Monitoring the Internet of Things (IoT) Networks

Basma Mostafa Hassan

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En **Informatique**

École doctorale Information, Structures, et Systèmes de Montpellier (I2S)

Unité de recherche **Laboratoire d'Informatique, Robotique et Micro-électronique de Montpellier (LIRMM)**

En partenariat international avec **Faculté d'Informatique, Université du Caire**, EGYPTE

MONITORAGE DANS LES RESEAUX DE TYPE INTERNET DES OBJETS

Présentée par Basma Mostafa HASSAN

Le 20 Décembre 2019

Sous la direction de Miklos MOLNAR
et Mohamed Mostafa SALEH

Devant le jury composé de

Abderrahim BENSLIMANE, Prof., Université d'Avignon, France	Président du jury
Bernard COUSIN, Prof., Université de Rennes 1, France	Rapporteur
Omar SAAD, Prof., Université du Helwan, Egypte	Rapporteur
Miklos MOLNAR, Prof., Université de Montpellier, France	Co-Directeur
Mohamed SALEH, Prof., Université du Caire, Egypte	Co-Directeur

UNIVERSITÉ DE MONTPELLIER

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Cairo University
Faculty of Computers and Artificial Intelligence
Operations Research and Decision Support Department

Cairo University

CAIRO UNIVERSITY AND THE UNIVERSITY OF MONTPELLIER

DOCTORAL THESIS

Optimized Monitoring of Internet of Things Networks

Author:
Basma Mostafa Abdelghany HASSAN

Supervisor:
Prof. Mohamed Mostafa SALEH
Prof. Miklos MOLNAR

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Cairo University and the University of Montpellier

UNIVERSITÉ DE MONTPELLIER

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Declaration of Authorship

I, Basma Mostafa Abdelghany HASSAN, declare that this thesis titled, "Optimized Monitoring of Internet of Things Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for the PhD research degree at Cairo University and the Université de Montpellier.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at both universities or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Basma Mostafa

Date: February 5th, 2020

Abstract

The research of this Ph.D. is fulfilled in the context of the optimized monitoring of Internet of Things (IoT) networks. The IoT enables the interconnection of billions of sensors, actuators, even humans to the Internet, creating a wide range of services, some of which are mission-critical. However, IoT networks are faulty in nature; Things are resource-constrained in terms of energy and computational capabilities. Moreover, they are connected via lossy links. For IoT systems performing a critical mission, it is crucial to ensure connectivity, availability, and network reliability, which requires *proactive* network monitoring. The idea is to oversee the network state and functioning of the nodes and links; to ensure the early detection of faults and decrease in node *unreachability times*. It is imperative to minimize the resulting monitoring energy consumption to allow the IoT network to perform its primary function. Furthermore, to realize the integration of the monitoring mechanism with IoT services, the proposed models should work in tandem with the IoT standardized protocols, especially the IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN) and the Routing Protocol for Low-power and lossy networks (RPL). In this challenging context, the first step of analysis is to ensure the (*optimal*) placement of monitoring nodes (*monitors*) to cover the given domain. Leveraging the graph built by RPL (the DODAG), the monitoring coverage can be modeled as the classic Minimum Vertex Cover (MVC) on the DODAG. MVC is NP-hard on general graphs and polynomial-time solvable on trees. To reduce the computational complexity, we introduce an algorithm to convert the DODAG into *nice-tree* decomposition. We demonstrate that the monitor placement, in this case, is only Fixed-Parameter Tractable, and can also be polynomial-time solvable. The monitoring role should be distributed and balanced to prolong network longevity. To that end, assuming periodical functioning, we propose to assign Vertex Cover(VC) sets to time periods in a three-phase centralized monitoring approach. In the first phase, multiple minimal VC are computed. The assignment of the VC across the planning horizon is handled in the second phase; by modeling the scheduling as a multi-objective Generalized Assignment Problem (GAP). To minimize the state transitions in a duty-cycled monitoring approach, the optimal sequence between the sets of monitors across time periods is computed in phase three; by modeling it as a Traveling Salesman Path Problem (TSP-Path). In a third contribution, we provide the exact solution to the defined monitoring placement and scheduling problem via formulating a Binary Integer Program. The model serves as a benchmark for the performance evaluation of contemporary models. Moreover, the monitoring mechanism must adapt to real-time network instabilities. Therefore, in our final contribution, we propose a dynamic distributed monitoring scheduling mechanism, which is implemented in the Contiki OS for constrained networks, and experimented using the COOJA simulator; the *de facto* simulator for IoT applications. Results demonstrate the models' effectiveness in realizing full monitoring coverage with minimum energy consumption and communication overhead, and a balanced distributed monitoring role.

Acknowledgements

The author Basma Mostafa would like to thank the French Ministry of Foreign Affairs, the Embassy of France in Egypt, the French Institute in Egypt for their grant, and the Egyptian Ministry of Higher Education and Scientific Research. Also, she would like to thank the L'Oreal foundation and UNESCO for their award in the 2017 L'Oreal-UNESCO for Women in Science Levant and Egypt Fellowship.

Contents

List of Figures

List of Tables

Chapter **1**

Résumé

1.1 Contenu

Les réseaux « Internet des Objets » se composent de plusieurs millions d'objets qui possèdent une adresse IP et qui peuvent se connecter sur Internet [1]. En général, ces objets sont supposés d'être autonomes et peuvent résoudre des tâches : mesurer, traiter et fournir des informations pour les systèmes connectés et pour les utilisateurs. L'émergence de l'Internet des Objets (IdO) introduit de plus en plus de services et d'applications dans l'histoire dans les villes intelligentes [2] ; pour la santé [3] ; etc. Aussi, ces réseaux sont vulnérables (c.-à-d. : les éléments peuvent être mobiles et la topologie du réseau peut changer dynamiquement), les changements peuvent influencer le (bon) fonctionnement du réseau. De plus, ils peuvent être alimentés par des batteries de durée de vie limitée, ce que nécessite la réduction de leur consommation.

Ce travail de thèse aborde un sujet important dans le domaine de l'Internet des Objets, qui consiste à savoir comment assurer la robustesse et le fonctionnement tolérant aux pannes du réseau pour répondre aux exigences des missions critiques. Avec le large déploiement des services IdO, ce problème est devenu particulièrement crucial pour les applications telles que le monitorage intelligent de la santé ou de détection de pannes et de sécurité industrielle où l'état des objets communicants doit être constamment vérifié pour le rétablissement rapide en cas de problèmes de communication inattendus. Pour réussir à surmonter les défis mentionnés au-dessus, une des possibilités est le monitorage des éléments, la détection et la localisation des problèmes. Le monitorage peut être coûteux et peut ajouter des charges supplémentaires au réseau. On cherche alors de minimiser le coût du monitorage et l'utilisation de l'énergie, et aussi les charges additionnelles sur les réseaux. Pour un état statique donné du réseau, les problèmes d'optimisation correspondants sont souvent NP-difficiles. Et avec le compartiment dynamique du réseau, les problèmes deviennent encore plus difficiles à résoudre. Pour résumer, notre objectif est d'optimiser le monitorage des réseaux IdO pour être robuste avec une tolérance aux pannes.

1.2 Objectifs fixés dans la thèse

Cette thèse propose des architectures et des algorithmes pour le monitorage des réseaux Internet des Objets, pour améliorer la disponibilité des hôtes et des services, et pour la détection et localisation des événements anormaux dans les réseaux IdO. Cet outil recueillera en temps réel les données provenant de composants et, par conséquent, il peut être très utile pour corriger ou même prévoir les pannes avant qu'elles ne se produisent. Dans la thèse, plusieurs systèmes de monitorage dans les réseaux IdO sont étudiés. Le choix des techniques du monitorage, les problèmes de recouvrement des éléments sont examinés ; et finalement utilisant des méthodes passives sont sélectionnées. Afin de préserver l'énergie des éléments, la bande passante du réseau, il faut également minimiser l'intervention du monitorage. Les méthodes

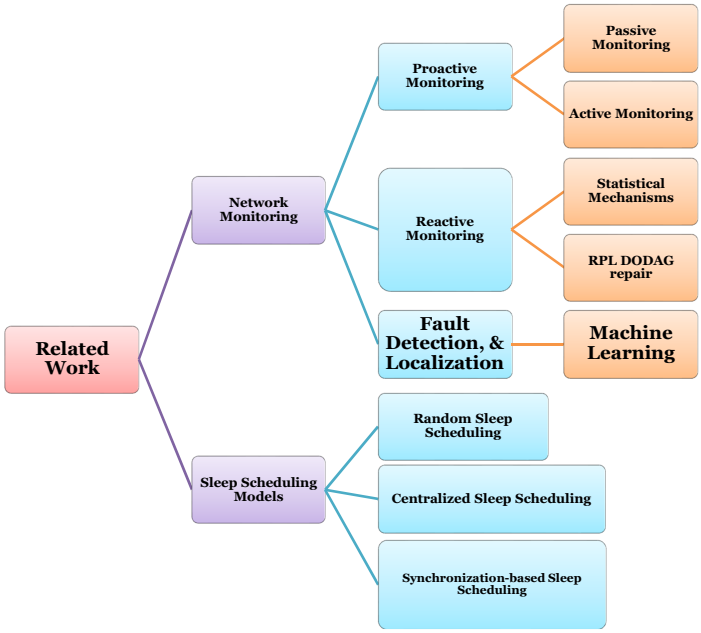


FIGURE 1.1: Résumé de la revue de littérature.

proposées coexistent et coopèrent avec des solutions existantes dans ces réseaux (découverte de la topologie par exemple). Dans la thèse ; on trouve la formulation exacte des problèmes liés et une décomposition de ce problème NP-difficile, des propositions algorithmiques distribuées. De cette façon, on trouve l'analyse de la complexité des différents cas formulés, la recherche des algorithmes exacts, des algorithmes heuristiques. Dans la thèse on trouve :

1. étude de la technologie des réseaux IdO,
2. étude des exigences des services et des mécanismes existants, et
3. propositions pour les architectures de monitorage et des solutions algorithmiques.

1.3 Contributions et travaux effectués

1.3.1 Revue de l'état de l'art de travail couvre les aspects suivants (voir Figure 1.1)

— Un état de l'art complet pour analyser les objectifs, les défis, les technologies et les protocoles spécialement conçus pour les réseaux IdO.
— Une analyse du rôle et des mécanismes des IPv6, 6LoWPAN, protocoles de routage pour les réseaux à faible puissance et fort taux de perte (RPL).
— Une étude du mécanisme de découverte de voisinage dans 6LoWPAN pour découvrir la topologie du réseau et une étude de son efficacité dans le cas de noeuds endormis ou défaillants.
— Une étude du mécanisme de la construction de graphes acycliques orientés dirigés vers une destination (DODAG) dans RPL et une étude de la suffisance de RPL pour la détection de pannes et l'autoréparation.
— Une analyse des techniques existantes de détection d'anomalies et de pannes dans les réseaux de capteurs sans fil.

— Une analyse des techniques de monitorage de réseau existantes, en particulier le monitorage actif par rapport au monitorage passif et combinatoire et le monitorage saut par saut par rapport au monitorage basé sur le chemin.
— Une étude des technologies de communication typiques dans les réseaux sans fil et l'IdO afin d'argumenter les modèles choisis pour l'optimisation.
— La détection des faiblesses et des points sensibles dans le protocole RPL proposé pour l'IdO, en termes de périodicité de la construction du graphe/réseau et la fréquence des réparations des DODAG dans le RPL et sa capacité à faire face au caractère dynamique et au manque de fiabilité du réseau.

1.3.2 Contribution n°1 : "Distributed Monitoring in 6LoWPAN-based Internet of Things"

Nous avons proposé un algorithme qui vise à réaliser un placement distribué des moniteurs avec une complexité minimale pour le calcul. L'algorithme proposé fonctionne avec RPL. L'objectif principal est d'augmenter la robustesse dans les réseaux IdO ciblant les applications critiques en temps réel via le monitorage des liaisons dans les DODAGs construits par RPL. Dans notre première contribution, le problème est modélisé comme un problème de couverture minimale des sommets (VCP) sur le DODAG. Ce problème est très connu comme un problème d'optimisation NP-difficile. Cependant, nous avons développé un algorithme à temps polynomial qui transforme le DODAG en une décomposition arborescente (Nice-Tree Decomposition) avec une largeur arborescente (treewidth) unitaire (voir Figure 1.2). Cette stratégie profite de la spécificité des DODAG et a abouti à une réduction significative de la complexité de la résolution du VCP sur les DODAG. Elle peut être résolue en temps polynomial. Le travail a été publié dans la conférence internationale MoWNet'2016 de l'IEEE, qui se focalise sur les réseaux sans fil [4].

1.3.3 Contribution *n*°2 : "An Energy-Efficient Multi-objective Scheduling Model for monitoring in Internet of Things"

La deuxième proposition est un modèle approché pour l'optimisation de l'ordonnancement du rôle de monitorage des nœuds dans les réseaux IdO, afin de maximiser la durée de vie des dispositifs embarqués à ressources limitées, tout en minimisant le coût global du monitorage de réseau. Le monitorage de réseau est très coûteux, en particulier pour les réseaux à ressources limitées tels que l'IdO. Par conséquent, le monitorage doit être économe en énergie et avec des frais généraux minimaux comparé à la performance normale du réseau. Notre travail correspondant contient une proposition d'un modèle mathématique en trois phases pour assurer l'exigence d'une couverture des moniteurs tout en minimisant la consommation d'énergie de monitorage et les frais de communication (voir Figure 1.3).

Notre modèle proposé décompose le problème abordé en trois problèmes d'optimisation bien connus, il s'agit du problème de couverture de sommets (VCP), problème d'affectation généralisé (GAP) multi-objectives et problème de voyageur de commerce (TSP). Ces problèmes difficiles sont discutés en détail dans la littérature et de nombreux algorithmes d'approximation avec de rapports d'approximation intéressants sont proposés. En outre, la résolution de ces problèmes sur des graphes spéciaux tels que les arbres et même les graphes acycliques orientés dirigés vers une destination (DODAG) qui sont construits par le protocole de routage normalisé RPL s'est avérée peu coûteuse. De plus, l'intégration du modèle proposé avec des protocoles normalisés facilite le déploiement à grande échelle du modèle. Les modèles proposés ont été mis en œuvre et les expérimentations ont été menées en utilisant des DODAG générés aléatoirement. Les résultats (voir Table 1.1) prouvent théoriquement

l'efficacité du modèle proposé. Cette contribution a été publié dans la revue : IEEE Internet of Things Journal [5].

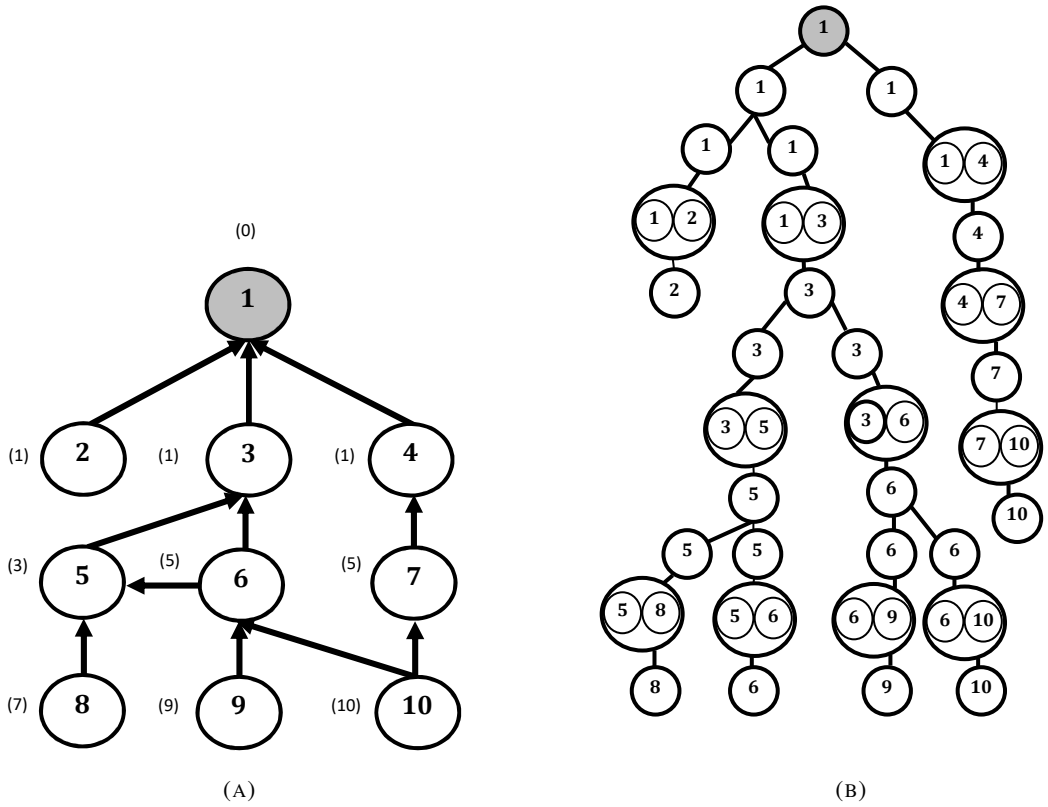


FIGURE 1.2: Décomposition d'un DODAG en un arbre avec une largeur arborescente de un. (A) Le graphe DODAG avant la décomposition ; (B) après la décomposition.

1.3.4 Contribution n°3 : "Exact Optimal Monitor Placement & Scheduling for Fault-Tolerant IoT Networks"

Dans cette troisième partie, une approche exacte est proposée pour résoudre le problème décrit dans la contribution n°2. Comme nous avons vu, la décomposition en trois phases ne donne pas la solution exacte. Nous avons donc proposé une formulation exacte du problème qui consiste en un problème de l'affectation minimum des tâches de surveillance avec un fonctionnement de surveillance cyclique. Pour cela, nous avons formulé un programme en nombres entiers binaires. L'ordonnancement optimal garantit la couverture du graphe pour la surveillance avec une consommation d'énergie minimale. La solution peut être intégrée dans un système centralisé, en utilisant un mécanisme de surveillance passif et interopérable avec les protocoles RPL et 6LoWPAN.

Cette solution est implémentée sur Julia et les programmes linéaires en nombres entiers (ILP) sont résolus en utilisant le solveur Gurobi. Les expérimentations sont conçues en utilisant des instances de réseau avec différentes topologies avec des différentes tailles (De 25 à 4941 nœuds et de 150 à 11535 arêtes). Les résultats (voir Table 1.2) démontrent l'efficacité des modèles proposés pour réaliser une surveillance avec une consommation d'énergie minimale et frais généraux du réseau ; tout en équilibrant le rôle de surveillance entre les nœuds.

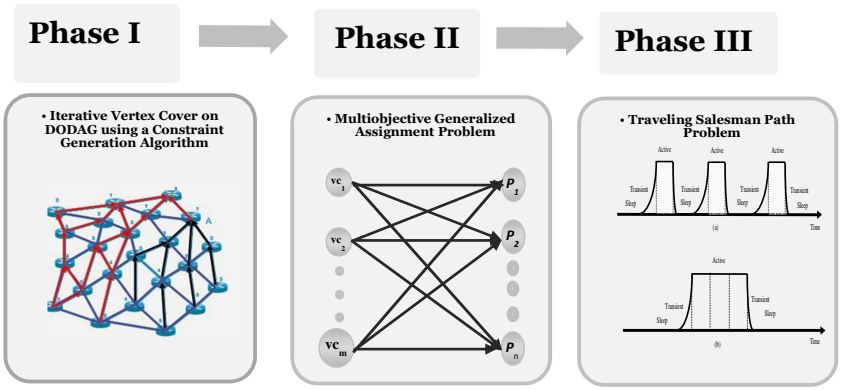


FIGURE 1.3: Figure 3 : Les trois phases des calculs.

TABLE 1.1: Résumé des résultats de l'optimisation en trois phases

$\|V\|$	$\|E\|$	p	% Monitors	% Residual battery	Execution time (sec)			% Reduction in state transitions
					Phases			
					I	II	III	
50	123	0.125	60	86.0	0.83	3	0.001	66.6
50	104	0.09	52	89.8	0.46	2	0.001	67.0
100	234	0.09	55	88.8	2.02	2	0.001	70.0
100	347	0.125	66	86.6	9.71	600	0.001	80.8
150	380	0.09	62	87.5	48.90	591	0.001	70.2
150	407	0.125	57	87.5	21.50	523	0.001	52.9
200	383	0.06	54	98.0	22.98	534	0.001	80.0
200	576	0.08	63	87.3	30.06	300	0.001	0.00

La conclusion la plus remarquable qu'on peut observer dans les résultats expérimentaux du modèle exact est que la limitation sévère de la batterie réservée à la surveillance présente l'avantage d'équilibrer la charge entre les nœuds. La consommation moyenne d'énergie par nœud diminue avec des contraintes d'énergie sévères. Cette constatation met en évidence le fait que le niveau de la batterie réservé à la surveillance doit être choisi avec beaucoup de soin. D'autre part, on constate que le calcul de la solution exacte du modèle pour les réseaux de grande taille est coûteux, correspondant au fait que le problème MVC est déjà NP-difficile. Ce fait implique une borne inférieure exponentielle sur le temps de calcul du Programme Linéaire en Nombres Entiers ou Binaires proposé. Malgré ces limites, le mécanisme proposé sert comme point de repère pour des comparaisons et des éventuelles évaluations de performance des modèles récents et futurs.

1.3.5 Contribution n°4 : "Dynamic Distributed Monitoring for 6LoWPANbasedIoT Networks"

Comme la topologie du réseau IoT est souvent dynamique, il est nécessaire de cibler des algorithmes dynamiques pour le monitorage dynamique dans l'avenir. En outre, il est

nécessaire aussi de tester l'efficacité des modèles proposés par rapport aux mécanismes de réparation de RPL. Nos propositions sont basées sur le monitorage passif. Cependant, on peut avoir des périodes et des parties des réseaux où le monitorage passif n'est pas possible par manque du trafic permanent. Il sera indisponible d'investir dans l'optimisation de techniques actives.

Des évaluations de performance et des estimations précises du niveau d'énergie des objets ont été réalisées dans Contiki / COOJA, qui est le simulateur de facto pour les réseaux IdO sous contrainte. Les expériences illustrent l'efficacité du modèle proposé pour atteindre une couverture complète du réseau de manière dynamique. La découverte des voisins et la récolte des connaissances sur l'état des voisins (et des deuxièmes voisins) sont obtenues par des communications locales entre les voisins. Ces communications sont réalisées au début de chaque période, pendant une courte période de négociation. Les nœuds dans une situation de surveillance critique (où la surveillance des voisins ne peut pas être réalisée par d'autres moniteurs) ne sont pas autorisés à dormir.

Des simulations ont été effectuées pour évaluer l'adaptabilité du modèle aux contraintes énergétiques sévères. Similairement au modèle exact, les résultats de l'heuristique dynamique soulignent aussi que plus la contrainte énergétique est sévère, plus la consommation moyenne d'énergie est favorable. En même temps, la couverture du réseau est garantie (voir Table 1.3). De plus, dans le cadre des expériences, une analyse de sensibilité a été réalisée pour obtenir la "meilleure" combinaison des paramètres ajustables et minimiser le compromis entre la consommation d'énergie et l'équilibre du rôle de surveillance sur l'ensemble des nœuds (voir Table 1.4). Une très courte longueur des périodes entraîne une répartition inéquitable du rôle de surveillance, où certains nœuds épuisent des quantités relativement élevées d'énergie pour la surveillance, tandis que d'autres ne consomment pas d'énergie pour la surveillance. En revanche, une période trop longue peut impliquer un risque important d'augmentation de la consommation moyenne d'énergie ; qui se justifie par la longue obligation des cycles de surveillance.

Par rapport à la décomposition en trois phases, l'heuristique distribuée dynamique produit des bons résultats concernant la complexité et l'évolutivité des calculs. Par rapport au modèle exact, l'inconvénient de l'algorithme dynamique est que la planification n'est pas optimale. Cependant, il a l'avantage d'atteindre une adaptabilité robuste et un suivi en temps réel des changements dans le réseau. Cette solution réduit également le temps de calcul nécessaire et la surcharge de communication à l'aide du mécanisme distribué, la performance de l'algorithme dynamique est remarquable. Cependant, des expérimentations supplémentaires avec les deux modèles sont nécessaires pour évaluer un facteur d'approximation expérimental de l'heuristique dynamique. Cette activité est la première dans la liste des travaux futurs. Par ailleurs, une conception et une expérimentation des surveillances actives et hybrides et une étude de ces approches sont intéressantes pour analyser la faisabilité, la complexité de calcul et la consommation d'énergie.

TABLE 1.2: Résultats expérimentaux du modèle exact

Instance	**Topology**	**|V|**	**|E|**	ρ	**monitor(%)**	**|N|**	**Energy cons.**(mJ)	**battery cons.(%)**	**msgs**	**time** (sec)
1	Random	25	150	0.250	80	13	9.936	19.872	5	109.66
2	Random	25	180	0.300	76	16	9.439	18.878	5	7.20
3	Random	25	210	0.350	84	18	10.432	20.865	6	28.82
4	Random	25	240	0.400	84	21	10.432	20.865	5	8.30
5	Random	25	270	0.450	88	23	10.929	21.859	6	8.75
6	Random	25	300	0.500	96	25	11.923	23.846	5	8.46
7	Random	27	330	0.550	88	26	11.040	22.080	5	56.76
8	Random	30	130	0.150	66	11	8.280	16.560	6	59.73
9	Random	30	174	0.200	77	13	9.522	19.044	5	153.82
10	KARATE [6]	34	78	0.101	41	5	5.607	11.214	5	66.90
11	Random	34	470	0.430	87	29	10.914	21.829	5	49.57
12	DOLPHINS [7]	62	159	0.024	56	5	7.013	14.026	5	175.30
13	DOLPHINS* [7]	62	207	0.054	58	9	7.211	14.850	7	104.95
14	POLBOOKS [8]	105	441	0.040	60	8	7.570	15.430	6	13.61
15	FOOTBALL [9]	115	613	0.047	82	11	10.152	20.304	5	78.55
16	POWER [10]	200	209	0.005	44	2	5.437	10.875	5	51.30
17	POLBOOKS* [8]	399	950	0.005	29	6	3.673	7.346	6	15.00
18	Complete Graph	500	249500	0.990	99	499	12.395	24.790	6	276.60
19	Random	600	179700	0.500	99	599	12.3993	24.798	5	466.90
20	NETSCIENCE [11]	1589	4331	0.002	57	6	45.005	90.011	5	2.82
21	POWER* [10]	4941	11535	0.0005	47	5	5.799	11.598	5	111.24

TABLE 1.3: Consommation d'énergie dans un réseau de 20 nœuds avec différents niveaux de batterie réservée pour la surveillance. Durée totale = 10000 ms, longueur des périodes = 500 ms, périodes de négociations = 250 ms, fréquence Tx = 30 ms

Reserved battery(%)	**Reserved battery(kJ)**	**Avg. consumption(kJ)**	**Max. consumption(kJ)**
1	114.21	132.91	411.25
5	571.05	79.97	488.30
10	1142.10	72.04	438.08
15	1713.15	68.78	413.36
20	2284.20	72.94	428.89
25	2855.25	72.99	454.73
30	3426.30	81.49	484.6
35	3997.35	67.01	398.49

TABLE 1.4: Consommation d'énergie dans un réseau de 40 nœuds avec des longueurs des périodes différentes. Durée totale = 10000 ms, périodes de négociations = 250 ms, fréquence Tx = 30 ms, batterie réservée = 10% (1142,1 (kJ))

Period Length (ms)	**Max. consumption (kJ)**	**Avg. consumption (kJ)**	**Standard Deviation (kJ)**
100	175.53	60.17	95.23
500	409.80	142.27	144.20
1000	443.17	156.82	31.34
2000	281.16	94.64	105.65
2500	318.86	125.79	45.37
3000	515.99	215.43	66.41

Chapter **2**

Introduction

2.1 Research Motivation

The Internet of Things (IoT) is a global network and service infrastructure with connectivity and self-configuring capabilities ; based on standard and interoperable protocols. The IoT consists of heterogeneous things that have identities, physical and virtual attributes, and are seamlessly and securely integrated into the Internet [1]. The goal of the IoT is to enable things to be connected anytime, anyplace, with anything and anyone, ideally using any network. By connecting billions of things to the Internet, IoT created a plethora of applications that touch every aspect of human life ; to name but a few : Wearables [12] and health monitoring [3], military applications as intrusion detection in remote or hostile environments, environmental monitoring, smart homes [13], smart cities [2], smart grids [14] and connected cars [15].

Contrasting the IoT and the standard Internet, the ultimate difference resides in the fact that IoT networks mainly use Low-power Lossy Networks (LLN). LLNs have several limitations, such as energy, memory, and computational constraints of incorporated devices, uncertain radio connectivity, and extensive protocol overheads against memory. The connection of such networks to the IPv6-based Internet required an adaptation layer between the MAC and the network layers ; hence came the IPv6 over LoWPAN (6LowPAN). 6LowPAN enabled the reliance on IPv6 for addressing, which in turn allowed the provision of the large address space that was required for connecting such a tremendous number for devices to the Internet. It is one of the most significant protocols which facilitated the emergence of the IoT.

IoT networks are characterized by :

— incorporated things are connected via unreliable, lossy channels with unpredictable bandwidth,
— stringent resource constraints with respect to energy, processing power and memory of devices,
— vulnerability to security risks from the Internet and the shared wireless medium,
— limited network lifetime that should be maximized by incorporating duty-cycling [16],
— highly dynamic network topology,
— possibly deployment in hostile, highly dynamic environments.
— eventual node unreachability, and
— self-configuration, lack of infrastructure, and complexity of the design of network protocols.

These characteristics of IoT networks make them vulnerable to faults and security attacks. Although a significant number of IoT applications are not time-sensitive, there is a whole class of mission-critical applications, especially the ones which target human safety. For instance, health monitoring [3], critical control and fault detection applications[17, 18, 19]. Taking health monitoring as an example (Fig. 2.1), it involves the gathering, aggregation, and mining of information indicative of the health of patients, such as a heart rate. In cases of

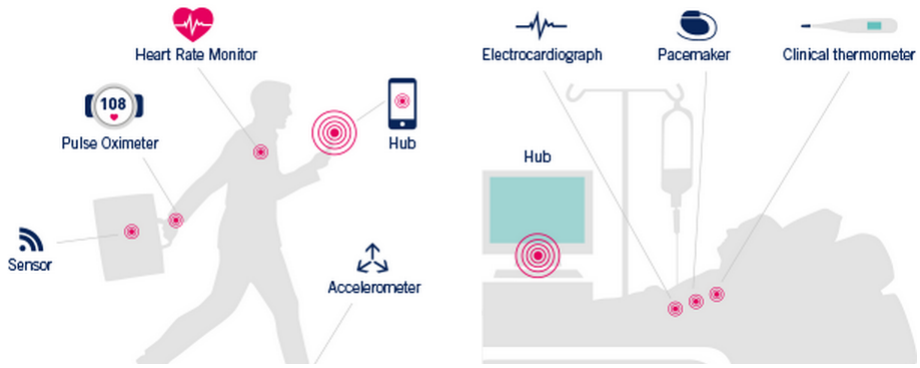


FIGURE 2.1: Health monitoring with IoT.

emergencies, an ambulance could automatically be informed and locate the patient via GPS signal. In this situation, high reliability is particularly crucial, where data must be processed and shared instantly and within strict reliability constraints.

Unfortunately, given the unreliable nature of LLNs, faults are common rather than rare events [20]. Therefore, maintaining robustness, continuous availability of devices, and reliability of communication, are critical factors to guarantee a constant, reliable flow of application data. According to the IEEE, reliability is *"the ability of a system or component to perform its required functions under stated conditions and for a specified period of time"* [21].

Furthermore, energy constraints impose hard duty cycles to maximize longevity, which in turn can cause unreliable connectivity [22]. In addition to unknown and dynamic network topologies, and unreliable connectivity, this leads to incomplete information about the current network state. The situation is considered a form of entropy, where a system deteriorates unless effort is invested in the development of monitoring and correction mechanisms to maintain a fault-tolerant system's performance [23].

The absence of any monitoring mechanism for detecting networks' faults would dramatically reduce the performance of the network, which renders monitoring the IoT network state a vital research area that will develop in significance. An effective and efficient monitoring mechanism could greatly improve robustness in network connectivity, reliability, and, eventually, Quality of Service (QoS), which will significantly increase the uptake of the technology by stakeholders.

2.1.1 Research Gap

To handle some of the aforementioned challenges, the Internet Engineering Task Force (IETF) has standardized the Routing Protocol for Low Power and Lossy Networks (RPL) for IP-connected IoT [24]. RPL is a self-healing routing and topology control protocol; it can respond to some node or link failures; through the application of route repair mechanisms for network recovery.

Several protocols for management and routing exist; however, RPL is the best candidate for critical systems that need fast recovery mechanisms. To minimize the cost of monitoring the links that are not being used; RPL favors the use of *reactive* repair approaches; thus trading fault-tolerance for routing stability and less control traffic [25], (*cf.* Chapter 3 for more details on RPL). Active mechanisms for regularly probing neighbors do not exist in the ContikiRPL implementation, and the neighbor unreachability detection (NUD) is not obligatory in neither 6LoWPAN nor RPL [26]; not until a node had already failed to reach its default router (parent) [24]. Only then, RPL triggers a repair mechanism. As a result,

the recovery time, which is the time required to establish a new route in the case of node unreachability could be relatively long. Moreover, the high signaling traffic volume during the route recovery mechanism is an overhead on network traffic.

"Knowing When Things Break Is Good. Knowing Before They Break Is Even Better". For mission-critical applications, *proactive* monitoring approaches are preferable. As a kind of preventive maintenance, proactive mechanisms enforce continual network monitoring; so that node and link failures are early detected, and alerts are promptly issued. Consequently, disconnectivity and service failures are prevented from occurring in the first place.

Nevertheless, all supplementary monitoring mechanisms should have minimal effect on energy consumption and traffic load; to leave the network unconstrained to perform its normal function of sensing, actuation, or transmission. If not applied carefully, proactively verifying network performance will negatively impact node resources. If ignored, this impact may lead to battery depletion due to idle listening to the radio channel and excessive control, increased congestion or delays in network traffic, which violate critical applications' requirements.

Consequently, to realize the intrinsic technical differences between conventional wireless and 6LoWPAN networks and their aforementioned mentioned vulnerabilities, optimizing the monitoring energy consumption and traffic overhead is crucial. Already existing monitoring mechanisms cannot be directly applied to the IoT. Moreover, to realize integration with already existing (and future) IoT services, it is detrimental that monitoring prepositions are entirely interoperable with the standardized IoT protocol suite; especially the IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN) and the Routing Protocol for Low-power and lossy networks (RPL). Interoperability is challenging mainly because IoT solutions are often tailored to specific scenario requirements without neither focusing on horizontal integration with other IoT services nor re-usability.

After a comprehensive literature review, and up to our knowledge, it is clear that there is a call for a new approach to optimized, energy-efficient, proactive monitoring, designed explicitly for IoT networks perform a critical monitoring mission [27].

2.2 Research Objectives & Methodology

2.2.1 Monitoring Requirements

Network monitoring mechanisms in general aim at detecting and localizing network faults. They should provide the appropriate tools for overseeing the network state, availability of, and connectivity between nodes. Through the mapping of symptoms of detected problems to possible root causes, the necessary corrective measures can be taken. Therefore, the monitoring infrastructure should be built such that monitoring nodes (*monitors*) are able to *cover* the entire network domain to successfully detect network failures.

IoT networks are enormous scale, consisting of potentially (hundreds of) thousands of nodes. Naturally, it is required that monitors are embedded (*placed*) in the correct locations to guarantee full monitoring coverage. Given the constrained resources of LLNs, it is also required to reduce the monitoring cost. Therefore, the number of monitors to be placed has to be minimized while satisfying the coverage condition.

The most common IoT architectures are completely centralized, mainly due to security reasons. The 6LoWPAN Border Router (6BR) is the central entity and is always assumed to be accessible [28]. Therefore, 6BRs can perform a potentially crucial role in centralized monitoring. Through multihop communication, the gathered data can be forwarded from monitors to the 6BR, then to a Network Operations Center (NOC), where sophisticated data analysis and mining can be performed.

However, energy is lost due to the communication between monitors and the 6BR, which is why it is necessary to find the shortest path in terms of the number of hops to the 6BR [29]. Furthermore, since the design of LLNs is ordinarily constrained by life span concerns, a prevalent approach toward expanding network longevity is by utilizing duty cycling. In duty-cycled networks, nodes enter sleep state frequently to conserve energy, and intermittently wake up to check for action [30]. High redundancy in network deployment is necessary to achieve this goal. Only then it is possible to identify small subsets of nodes that are active at a time, and put the major part of nodes into a sleeping state and thus saving energy. Various scheduling algorithms are used to organize the alternation of active and sleeping node sets to provide continuous service of the network.

Meanwhile, for critical applications, it is required that monitoring coverage is always guaranteed throughout the entire network lifetime, such that each link is monitored by at least one monitoring node, regardless of the lack of activity in the network. Therefore, effective and energy-efficient monitoring scheduling algorithms are necessary with duty cycling ; not only to satisfy the coverage constraints, but also to balance the distribution of the monitoring burden among nodes, and thus maximize longevity.

Consequently, so far, the monitoring objectives are three-fold : (1) minimize the number of placed monitors while satisfying the coverage condition, (2) optimally place the monitors such that the total communication energy consumed in the path between monitors and the 6BR is minimized, and (3) balance the monitoring role among nodes.

IoT network topology is often unstable due to node mobility, unreliable connectivity, and the fact that link connections between nodes are transient and unstable. [22]. Given that in LLNs, the nodes can only monitor the network's traffic within their radio transmission range, the monitoring coverage and scheduling connectivity problem is extremely challenging in duty-cycled LLNs. Moreover, it is inefficient to re-solve the monitoring placement and scheduling problems whenever the network topology is changed. Therefore, it is pertinent to target dynamic monitoring algorithms in order to handle these challenges. Dynamic algorithms include efficient, incremental methods that can adapt to real-time changes in the network without the requirement for re-optimization.

Finally, as stated before, to realize the integration with already existing (and future) IoT services, it is required that monitoring prepositions are completely *interoperable* with the standardized IoT protocol suite ; especially the IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN) and the Routing Protocol for Low-power and lossy networks (RPL).

To review the requirements for the proposed IoT network monitoring models (Fig. 2.2) :

1. monitors should be placed in the correct locations to guarantee full monitoring coverage,
2. the monitoring mechanism should support the presence of sleeping nodes,
3. the monitoring energy and communication costs should be minimal to satisfy the low-cost, low-power and scalable objectives of LLNs,
4. the mechanism should be dynamic to adapt to changing topologies without the need for complete re-optimization, and
5. monitoring should work in tandem with the standardized IoT protocol suite.

2.2.2 Research Methodology

To achieve the stated goals, our research methodology is the following (Fig. 2.3) :

— extensive reviews to the state of the art of monitoring wireless networks and especially Wireless Sensor Networks (WSNs) and the IoT protocol suite,
— identification of the monitoring requirements and research objectives,
— modeling and formulate the corresponding graph optimization problem,

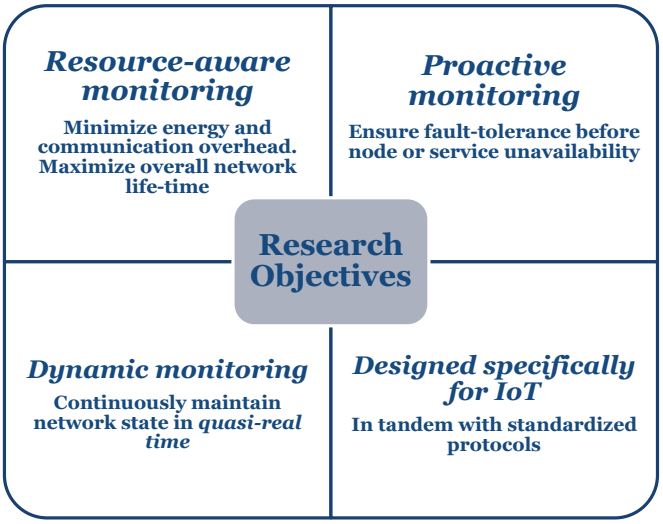


FIGURE 2.2: Research objectives.

— developing exact analytical solutions to the related graph problems,
— analysis of the proposed models for complexity and resolvability,
— integrated implementation of the monitoring mechanism with 6LoWPAN and RPL protocols, and
— performing extensive simulations for performance evaluations; to verify the effectiveness and efficiency of the proposed models.

2.3 Research Contributions

We target energy-efficient monitoring of 6LoWPAN-based IoT networks which utilize RPL for routing. Our first contribution, a Fixed-Parameter Tractable monitor placement algorithm, addresses the optimal location (placement) of the monitoring nodes. These monitors should be capable of observing the network traffic to infer the current network state; known as *passive monitoring*. Monitors can also initiate and collect link and node measurements, and control the routing of measurement packets, which is referred to as *active monitoring*. (*cf.* Chapter 3.3 for details on different monitoring approaches).

In this difficult context, the first step of analysis is to ensure the (*optimal*) placement of monitoring nodes (*monitors*) to cover the given domain. Monitors are responsible for anticipating failures by continuously tracking nodes and links and taking corrective actions beforehand. Leveraging the graph built by RPL, the DODAG (*cf.* Chapter 3), several graph-related optimization problems can be solved. We modeled the monitoring coverage as the classic Minimum Vertex Cover (MVC), which is NP-hard on general graphs and polynomial-time solvable on trees. To reduce computational complexity, we introduced an algorithm to convert the DODAG into *nice-tree* decomposition. We demonstrate that the monitor placement, in this case, is only Fixed-Parameter Tractable, and can also be polynomial-time solvable.This work has been published in the IEEE international conference MoWNet'2016, which focused on wirless networks [4].

The monitoring role should be distributed and balanced between the entire set of nodes to

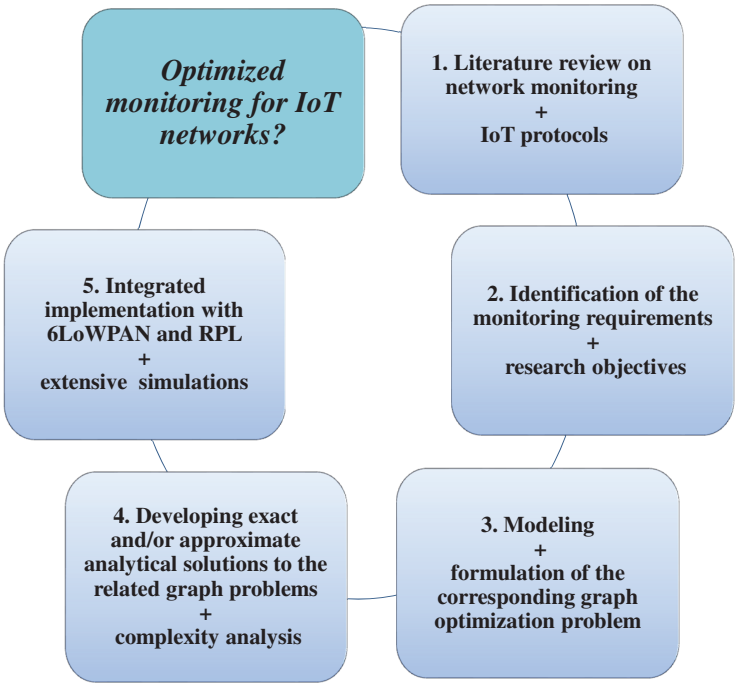


FIGURE 2.3: Research methodology.

prolong network longevity. To that end, assuming periodical functioning, we propose in a second contribution a three-phase centralized monitoring placement and scheduling approach; to optimally assign Vertex Covers to time periods. The first phase is responsible for computing several subsets of potential monitors by solving a *minimal* Vertex Cover Problem; by proposing a Constraint Generation algorithm. The optimal scheduling of the Vertex Covers is handled in the second phase, by modeling the scheduling as a multi-objective Generalized Assignment Problem. In a given period, if a node is not assigned to monitoring, nor does it have another role to perform, it can turn to a sleeping state. However, multiple state transitions consume extra energy. Therefore, to further minimize the energy consumption of the monitors, they are sequenced across time periods to minimize the state transitions of nodes. This part of the problem is modeled as a Traveling Salesman Path Problem (TSP-Path). Input to Phase III are the unique Vertex Covers assigned to monitoring in Phase II, while the output is the sequence that minimizes the total number of state transitions (from active-monitoring to sleep and vice versa). The sequence is generated using a dynamic programming implementation of the TSP-Path. This contribution has been published in the IEEE Internet of Things Journal [5].

In a third contribution, we provide an exact solution to the defined IoT monitoring placement and scheduling problem via formulating a Binary Integer Program, which has been missing from the literature. The objective of the formulated mathematical model is providing the *exact* solution to the optimal scheduling of the monitoring roles throughout a predetermined lifetime, using minimal monitor sets in each period. The formulation takes into account the presence of sleeping nodes by duty-cycling the monitoring, while *minimizing the monitoring state transitions*, and always respecting the monitoring *coverage requirement*, regardless of the lack of current network activity. Experimentation is designed using network instances of different topologies. Results demonstrate the effectiveness of the proposed model in realizing full monitoring coverage with minimum energy consumption and communication overhead; while balancing the monitoring role between nodes.

It is necessary that the monitoring mechanism adapts to real-time network instabilities

without re-solving the entire problem. Therefore, in our fourth contribution, we propose a dynamic, distributed monitoring scheduling mechanism that is completely interoperable with 6LoWPAN and RPL protocols. The model ensures real-time adaptation of the monitoring schedule to the frequent instabilities of networks, and the distributed feature aims at reducing the communication overhead between monitors and the Border Router, thus supporting *scalable* monitoring by reducing the computational complexity. The methodology is implemented in the Contiki operating system, and experimentation is performed using the COOJA network simulator, the *de facto* simulator for IoT applications. Simulations were performed to evaluate the model's adaptability to harsh energy constraints. The results highlight that the harder the energy constraint is, the lower is the average energy consumption; meanwhile, full monitor-network coverage is guaranteed. Experiments illustrate the effectiveness of the proposed model to achieve full network coverage dynamically.

Compared to the three-phase decomposition (*cf.* 4.3), the dynamic distributed heuristic achieves better results concerning computational complexity and scalability. Compared to the exact monitoring placement and scheduling (*cf.* 6.2.2), the only limitation is that the schedule is not *exact*, however, with the benefit of achieving robust, real-time adaptability to network changes and the reduced computational and communication overhead of the distributed mechanism, the performance of the dynamic model is superior.

To summarize our contributions to the optimized IoT monitoring problem, we propose :

1. a Fixed-Parameter Tractable monitor placement algorithm, by converting DODAGs into nice-tree decomposition,
2. a three-phase centralized monitoring placement and scheduling approach,
3. the exact solution to the IoT monitoring placement and scheduling problem via formulating a BIP, and
4. a dynamic distributed monitoring scheduling mechanism heuristic, with integrated Contiki implementation and experimentation using the COOJA network simulator.

Chapter **3**

Background & Related Work

3.1 The Internet of Things (IoT) : An Overview

3.1.1 Definitions of IoT

Despite the diversity of research on IoT, its definition remains somewhat fuzzy. There are different definitions that reflect different perspectives ; the following are the most prominent among them :

— *"IoT is a group of infrastructures interconnecting sensor(s) and/or actuator(s) with (limited) computing capabilities, and allowing their management, access to and transfer of the data they generate over the Internet without requiring human-to-human or human-to-computer interaction."* [31]

— *"IoT is a global network and service infrastructure of variable density and connectivity with self-configuring capabilities based on standard and interoperable protocols and formats [which] consists of heterogeneous things that have identities, physical and virtual attributes, and are seamlessly and securely integrated into the Internet."* [1]

— *"IoT is a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network.* [32]

— *"IoT is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment. The IoT comprises an ecosystem that includes things, communication, applications and data analysis."* [33]

— *"IoT refers to a loosely coupled, decentralized system of devices augmented with sensing, processing, and network capabilities."* [34]

3.1.2 Applications of IoT

Through connecting billions of things to the Internet, IoT created a plethora of applications that touch every aspect of human life ; to name but a few (*cf.* Fig. 3.1) :

— **Smart Cities** [2] - the utilization of sensors, networking structures, and Cloud-based integration systems to provide citizens services and infrastructure, and give them access to a wealth of real-time information about the urban environment ; upon which to base decisions and actions.

— **Smart Grids** [35] - electricity networks that can integrate the behaviour and actions of all users connected to it in a cost-efficient manner ; in order to ensure economically-efficient, sustainable power system with low losses and high levels of quality and security of supply and safety.

— **Connected Health** [3] - the gathering, aggregation, and effective processing and mining of rich information indicative of physical and mental health.
— **Smart Homes** [13] - a convenient home setup where appliances and devices can be automatically controlled remotely from any internet-connected place in the world using a mobile or other networked device.
— **Connected Cars** [15] - cars which are capable of internet connectivity and sharing of various kinds of data (location, speed, status of parts of the car, *etc.*) with back-end applications. Accordingly, various workflows can be created to take necessary actions, *e.g.* scheduling a service with the car service provider. Connected cars can also communicate with each other, and can send alerts to each other in certain scenarios like possible crash.

IoT applications can be divided with respect to their main requirements into three categories [36] :

1. **Real-time** - applications which contain time restrictions. For example, the Connected Health and Smart Farming require real-time monitoring of vital signs, and the Smart Supply Chain needs real-time for an efficient trading.
2. **Data analysis** - applications focused in analyzing data. For instance, Smart Retail, Smart City and Smart Grid rely on data analysis to optimize business, cities and the electrical grids, respectively.
3. **Device interaction** - applications focus on devices relations. In Smart Home, Wearables, and Industrial Internet, device interaction is a key aim.

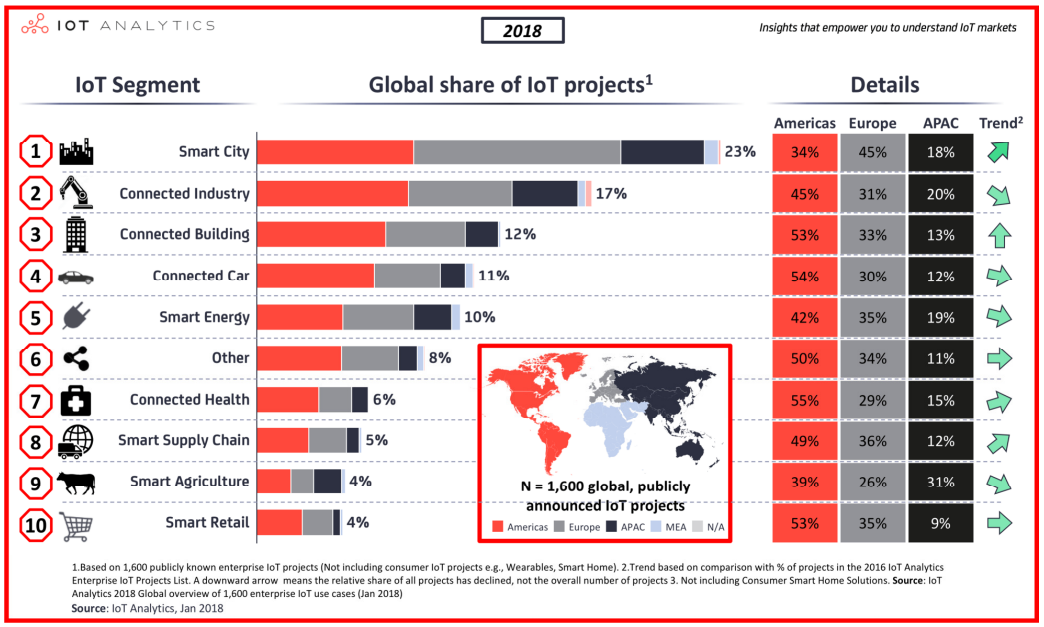


FIGURE 3.1: The most popular IoT applications (January 2018) [37].

3.1.3 Requirements & Challenges in IoT

— **Availability, Reliability & Denial-of-Service Resistance** - The hard energy, memory, and processing constraints of the things as well as the unreliable radio communication naturally alleviate node failures, long-term network unstability, Denial

of Service (DoS) and resource-exhaustion attacks. The recoverability of communicant objects, fault avoidance in the network, and application robustness in the face of uncertain information require effective defense mechanisms.

— **Energy-optimization** - For resource-constrained IoT entities, minimizing the energy consumed for communication and computing is a primary constraint. The need for devising solutions that tend to optimize energy consumption is aggravating everyday. Careful protocol (re)design and usage are required to reduce not only the energy consumption during normal operation but also under DoS attacks [38].

— **Self-configuration & self-organization** - The complexity and sheer dynamics of IoT systems calls for distributing the intelligence in the system and making IoT devices (or a subset thereof) able to autonomously react to critical situations. In order to ensure robust network function while minimizing human intervention [39], things should be able to perform device and service discovery, and tune protocols' behavior to adapt to the current difficult situation [40].

— **Scalability** - Due to the huge number of heterogeneous devices, it is required that IoT systems support the increasing number of connected devices, users, and analytics capabilities, without degradation in the QoS. However, scalability issues arise at, the data communication and networking, the service provisioning and management, and the naming and addressing of devices thus making it a difficult task to achieve.

— **Security** - The resource-constrained IoT devices are more vulnerable to attacks and threats. On the other hand, due to the strong entanglement with the physical realm, IoT networks should be secure and privacy-preserving without negatively impacting usability.

— **Interoperability** - It is a necessary requirement that information exchange takes place between all the interconnected IoT devices. However, the large number of heterogeneous devices and different technologies makes interoperability a key issue.

— **Semantic interoperability** - IoT involves exchanging and analyzing massive amounts of data. In order to turn them into useful information, ensuring interoperability among different applications, semantic description and provision of data with standardized formats is necessary for enabling the successful adoption of IoT [36].

3.2 Enabling Protocols & Standards for IoT

Features of the IPv6 design such as a simple header structure and its hierarchical addressing mode made it ideal for use in LLN with 6LoWPAN. However, LLN are particularly challenging for Internet protocols [41] due to the following reasons.

— Constrained devices need to keep low duty-cycles ; however, the basic assumption of Internet Protocols (IP) is that a device is *always connected.*

— *Unreliability* due to node failures, energy exhaustion and sleep duty cycles.

— The low-power radio technologies have constrained frame size ; while the minimum frame size for standard IPv6 is 1280 bytes ; which makes *header compression* and *data fragmentation* a necessity.

— *Multicasting* is crucial to the operation of many IPv6 features ; however, low-power radio technologies do not typically support multicast, and flooding in such a network is wasteful of power and bandwidth.

— The applications of low-power radio technology typically benefit from *multihop, mesh networking* to achieve the required *coverage* and *cost-efficiency*. However, current IP routing solutions may not be easily applicable to such networks.

— Low-power radio technology usually has *limited bandwidth*; in mesh topologies, bandwidth further decreases as the channel is shared and is quickly reduced by multihop forwarding.

Given the stated challenges for LLN to adopt IP, the connection of the resource-constrained things to the IPv6-based Internet required an *adaptation layer* between the MAC and network layers. To ensure the wide adoption of the IoT, these solutions have to be *interoperable* with the most widely used protocols in the Internet. The Internet Engineering Task Force (IETF) has formed several working groups to address these challenges and requirements. The *IETF is a standards* developing organization that creates voluntary standards to maintain and improve the usability and interoperability of the Internet. It has a major role in the development of a number of standards that are directly related to creating the environment needed for a successful, vibrant IoT.

Standardization has obvious advantages, namely :

— interoperability and compatibility with pre-existing architectures and web-services developed for standard IP networks,
— plug-and-play installation, and
— rapid connectivity and development of applications.

The IETF working groups targeted the :

— IPv6 over Low Power WPAN (6LoWPAN) [42],
— Routing Over Low Power and Lossy Networks (ROLL) [43], and
— Constrained Restful Environments (CORE) [44].

The 6LoWPAN group [WG-6LoWPAN] concentrates on the definition of methods and protocols for the efficient transmission and adaptation of IPv6 packets over IEEE 802.15.4; which is a standard for short-range low-power radios networks. The ROLL group develops IPv6 routing solutions for Low Power and Lossy Networks (LLNs), and the CoRE group aims at providing a framework for resource-oriented applications intended to run on constrained IP networks. Together, these protocols allow the IP-based integration of constrained devices into the Internet in a standardized way, as shown in 3.2.

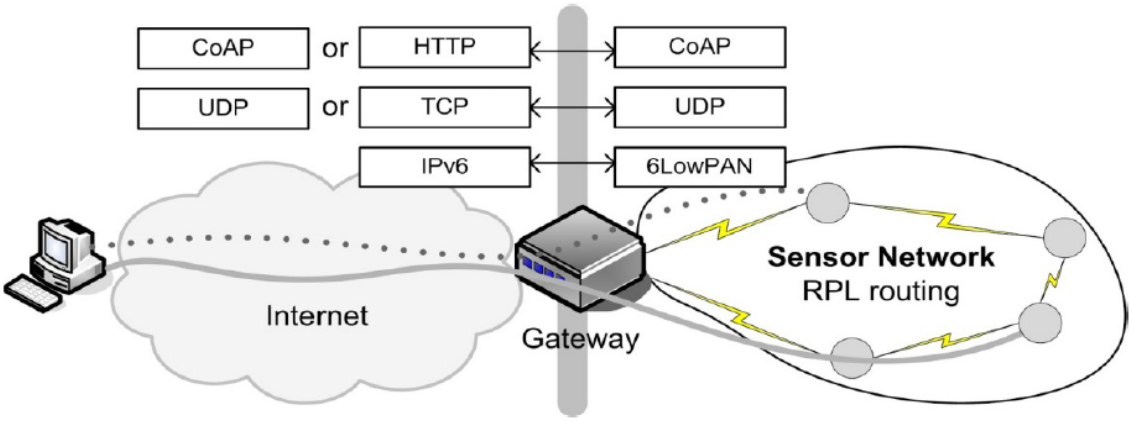


FIGURE 3.2: Internet protocols are extended to the sensor networks. The Gateway translates between the two standardized protocol stacks [45].

Before 6LoWPAN, a complex application-layer gateway was needed to make devices such as ZigBee, Bluetooth and proprietary systems connect to the Internet using IP. IP is the networking protocol used to provide all devices with an IP address to transport packets from one device to another. 6LoWPAN solves this dilemma by introducing an *adaptation layer* between the IP stack's link and network layers to enable transmission of IPv6 datagrams

over IEEE 802.15.4 radio links ; which radically changes the IoT landscape. By communicating natively with IP, 6LoWPAN networks are connected to other networks simply using IP routers.

The 6LoWPAN network is connected to the IPv6 network using an *edge router* or gateway (*cf.* Fig. 3.2). The edge router handles three actions [46] :

1. the data exchange between 6LoWPAN devices and the Internet (or other IPv6 network),
2. local data exchange between devices inside the 6LoWPAN, and
3. the generation and maintenance of the radio subnet (the 6LoWPAN network).

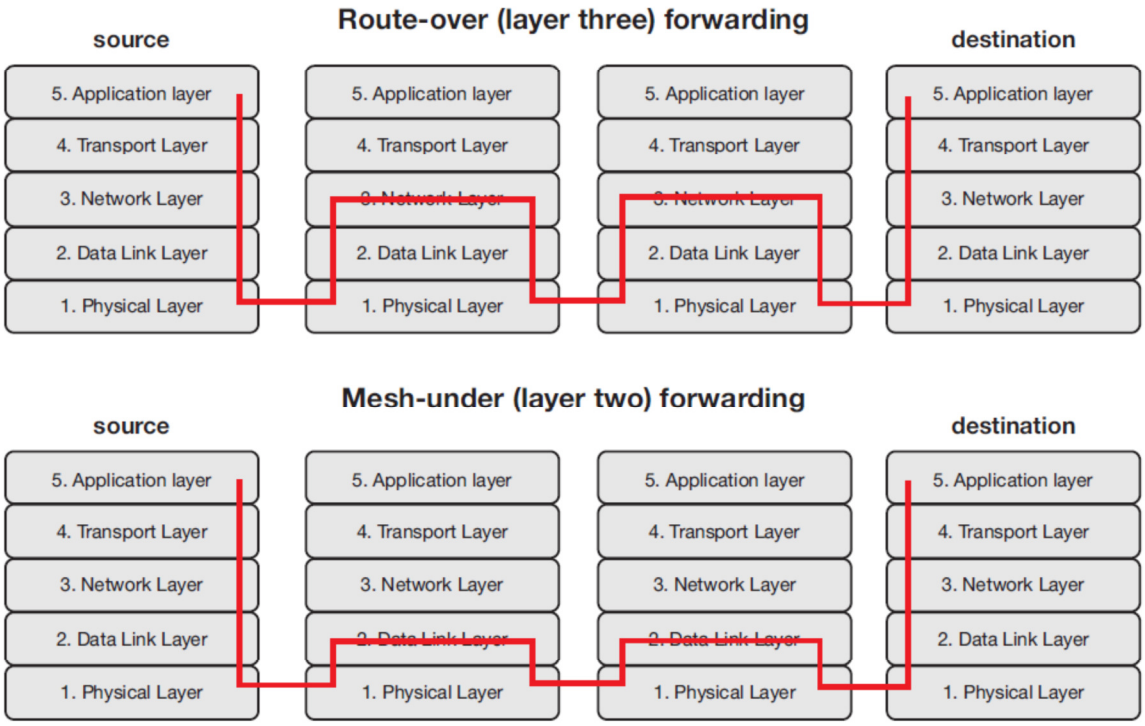


FIGURE 3.3: Route-over (layer three) forwarding versus Mesh-under (layer two) forwarding [41]

3.2.1 Routing over Low-power & Lossy Networks

Routing is the ability to send a data packet from one device to another device, sometimes over multiple hops. The routing protocols used in the network layer of IoT are similar to the network layer of standard Internet ; however, the network layer of IoT is specified towards LLN [47]. The unique requirements for routing over LLNs [48] are as follows.

— LLN applications may require the sending of messages for specific groups of nodes ; the routing protocol should support *constraint-based* routing and *multicast* forwarding.

— The routing protocol must support the addition of new nodes, real-time adaption to link failures, and route recomputing.

— the routing protocol must avoid *quick re-convergence* which unavoidably leads to the *lack of stability and unacceptable control plane*.

— It is common that the communication needs to be performed through several hops, requiring the support of *multihop routing* protocols.
— The routing protocol must support to mobility and scalability.
— The routing protocol must support different routing metrics, including link quality and throughput, and be able to adapt to the *dynamic nature* of the metrics.

In 6LoWPAN, depending on what layer the routing mechanism is located, two categories of routing are defined : *mesh-under* or *route-over* (*cf.* Fig. 3.3) [46]. Mesh-under uses the layer-two (link layer) addresses (IEEE 802.15.4 MAC or short address) to forward data packets ; while route-over uses layer three (network layer) addresses (IP addresses). In mesh under, one broadcast domain is established to ensure compatibility with higher layer IPv6 protocols ; using control messages which are broadcast to all devices in the network ; resulting in high network load. Thus, mesh-under networks are best suited for smaller and local networks. In route-over networks the routing takes place at the IP level as described above. The usage of IP routing provides the foundation to larger and more powerful and scalable network. The most widely used routing protocol for route-over 6LoWPAN networks today is the *Routing Protocol for Low-power lossy networks (RPL)*.

RPL is a distance vector IPv6 routing protocol for LLN (*cf.* [26] for a comprehensive review of RPL). It supports different modes of operation : *many-to-one* communication from the constrained nodes towards the root (typically the 6LoWPAN border router 6BR), *one-to-many* communication from the root to the constrained nodes and, *one-to-one* communication between the constrained nodes. RPL specifies how to build a *Destination Oriented Directed Acyclic Graph (DODAG)* ; which is a logical routing topology built over a physical network to meet a specific criteria ; using an *Objective Function (OF)*. The OF operates on a combination of *metrics and constraints* (such as link quality, node energy and link reliability) to compute the 'best' path ; via calculating the *rank* of each node [48]. Ranks determine the individual position of each node ir[illegible] d relative to other nodes (*cf.* Fig. 3.4).

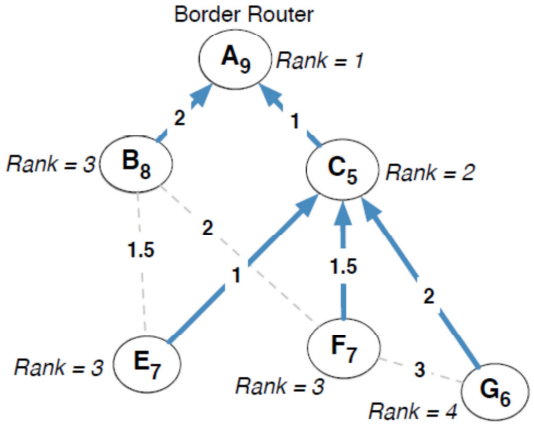


FIGURE 3.4: RPL DODAG construction [49].

RPL is designed to deal with incompatible networking requirements such that different applications are able to run concurrently over the same network infrastructure. *Multiple RPL instances* can co-exist within the same network for that purpose. The DODAG construction depends on the *Neighbor Discovery Protocol (NDP)* in the IPv6. NDP is a messaging protocol that facilitates the discovery of neighboring devices over a network. The protocol builds three logical sets of link-local nodes :

1. *candidate neighbor set* ; which is a subset of the nodes that can be reached through link-local multicast.

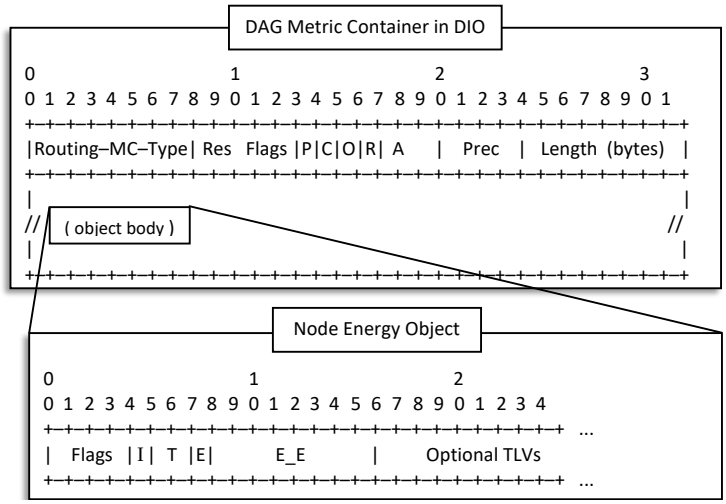


FIGURE 3.5: DIO Message with a DAG Metric Container option [50].

2. the *parent set*; which is a restricted subset of the candidate neighbor set; selected via using the objective function.
3. the *preferred parent set*; which is a member (or members) of the parent set that is considered as the preferred next hop in upward routes.

The RPL specification defines several types of Internet Control Message Protocol (ICMPv6) control messages; which are primarily used for topology maintenance and information exchange. The most important message among them is the DODAG Information Object (DIO) [26]; the fundamental source of information needed for topology (DODAG) construction. They primarily include information like the OF, the current rank of a node, the current RPL Instance, the IPv6 address of the root, etc. Usually, the construction of a DODAG is started by the DODAG root, which is usually a Border Router (BR), by broadcasting DIOs [50]. Any RPL implementation must support OF Zero (OF0); a metric-agnostic OF where the goal is for a node to join a DODAG that offers good enough connectivity to a specific set of nodes [51], though there is no guarantee that the path will be optimized according to a specific metric [52].

DIOs may contain an optional header for metrics and constraints objects, called DAG Metric Container (DAGMC) object Fig. 3.5) [50]. Node State and Attribute (NSA) and Node Energy (NE) are examples of the objects that could be included in the DAGMC. The NSA object can be used to provide information on nodes characteristics, for example, CPU overload and node available memory, while NE object provides information pertaining to energy and power mode (i.e. main-powered or battery-powered), and most importantly the estimated remaining power-level for nodes operating with batteries.

DODAG Repair Mechanisms

Repair mechanisms are significantly crucial for a routing protocol to update routes dynamically and adapt the network topology to potential failures. RPL supports two integral repair mechanisms; in particular *local repair* and *global repair* [26]. When a node detects a network failure (e.g. a link between two nodes fails), it triggers local repair (*cf.* Fig. 3.7). It consists in searching for a backup path urgently without attempting to repair the entire DODAG. However, this alternate recovery path may not be the best path with respect to the

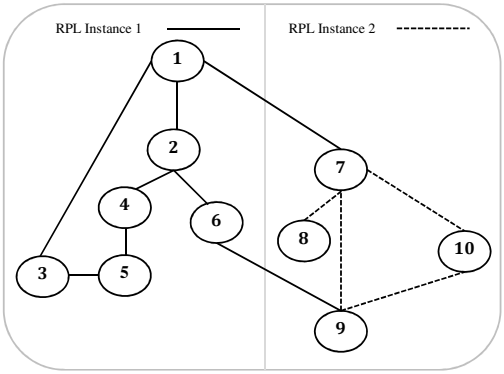


FIGURE 3.6: Example of the multiple instance feature of RPL. Nodes 7 and 9 participate in both instances.

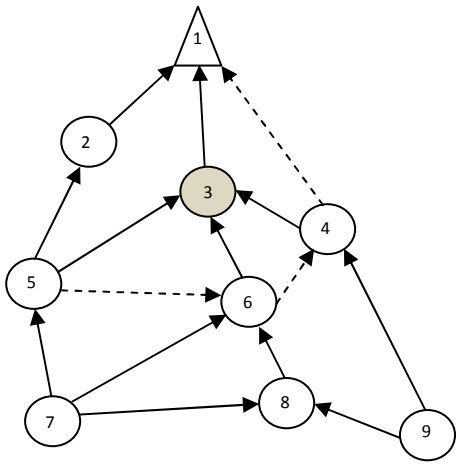


FIGURE 3.7: Example of local repair mechanism in RPL. Node 1 is the DODAG root, node 3 is unreachable, and the backup path for node 5 to the root is in dashes.

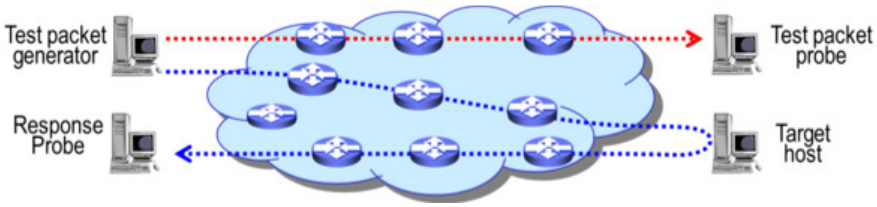


FIGURE 3.8: Active network monitoring

defined OF. The second mechanism is the global repair; it prompts the fundamental reconstruction of the entire network topology, but on the expense of additional control traffic in the network [53]. Nodes in the new DODAG version can choose a new position whose rank is not dependent on the previous one. The global repair is considered as an entire re-optimization of the routes.

There are times when the nodes are unreachable during the DODAG rebuilding process. Therefore, relying solely on global repairs is neither efficient, in terms of the number of control messages, nor reliable especially for mission-critical IoT applications where there is no tolerance for nodes unreachability.

RPL's connectivity repair mechanisms are considered to be *reactive*; since they are triggered after the failure has been detected. There are times when nodes might be unreachable; as reported in [25], the average unreachable time of the node during DODAG reconstruction is almost three and a half minutes. For critical applications, the recovery time could be detrimental to availability and QoS; where delay is intolerable, for instance health monitoring [54] and safety applications [55]. Thus to prevent the deterioration of IoT systems and maintain a fault-tolerant systems' performance, effort should be invested in the development of efficient network monitoring and correction mechanisms [23]. The state of the art of IoT network monitoring is presented in Chapter 3.

3.3 Related Network Monitoring & Scheduling Mechanisms

3.3.1 Network Monitoring : An Overview

Network monitoring mechanisms in general aim at detecting and localizing network faults, providing the appropriate tools for overseeing the network state, availability of, and connectivity between nodes. Through the mapping of symptoms of detected problems to possible root causes, the necessary corrective measures can be taken.

Generally, network monitoring mechanisms can be classified with respect to several attributes. Starting by their **reactivity** to network faults, monitoring mechanisms are either proactive or reactive [56]. In **proactive monitoring** [57, 58, 59], the mechanism maintains correctness in the network via scanning for potential points of failure or security risks, and resolving them before they develop into serious problems. With **reactive monitoring** [60, 61] [62], the mechanism adapts the network configuration based on detected faults, via collecting information about the current network state and past events, then taking the appropriate corrective measures.

Monitoring techniques can also be classified with respect to their **interaction** with the network into active versus passive monitoring [63]. **Active monitoring** keeps track of the network by directing a probe and collecting the response to and from the monitored nodes. The instrumentation of both monitoring and monitored nodes and the extra load on network traffic constitute a serious overhead when applied to LLNs. **Passive monitoring**, on the other hand, infers the state of monitored nodes and links from the traffic that is normally passing

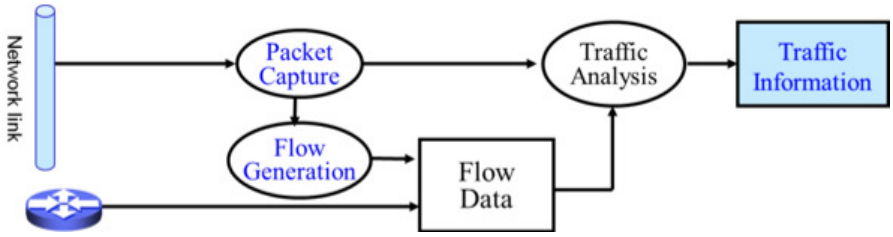


FIGURE 3.9: Passive network monitoring.

through the network; which reduces the number of devices required to be instrumented, and does not impose an extra overhead on network traffic. However, when compared to active monitoring, passive techniques are restricted in their ability to isolate the exact fault location [64]. **Hybrid active/passive monitoring** (combinational monitoring) can be convenient for fault localization with reduced overhead. Initially, passive monitoring is incorporated for fault detection; subsequently, active monitoring mechanisms are used for localization [65].

Relative to the **architecture**, monitoring mechanisms can be classified into centralized versus distributed mechanisms [56]. In **centralized monitoring** [66], a base station performs the role of central management. It gathers information from the entire node set; granting it a comprehensive view of the network state. In case where the central node has unlimited resources, it can accomplish sophisticated monitoring tasks; which might otherwise exhaust the resources of the constrained nodes. Specialized softwares for centralized network management often help to monitor, analyze, and manage all of the network components and view the performance of individual components or the entire system on a graphical screen. However, powerful centralized network monitoring is achieved on the expense of high message overhead. There are scalability limitations of centralized models when the network starts growing significantly [67].

In **distributed monitoring** [68, 69, 70] a finite number of probe-clusters *(monitors)* participate in the monitoring tasks by "covering" their neighborhoods and gathering complete traffic information. Complete traffic information can be collected by network-wide traffic monitoring. However, network-wide traffic analysis requires the understanding of the order of messages appearing in the network. Therefore, time-stamping with small granularity and high precision are critical requirements of distributed monitoring [71]. Clearly, the message overhead is less than in centralized monitoring. However, finding feasible time synchronization solutions among the physically distributed monitors is not trivial. Moreover, since the monitors might be responsible for other primary tasks (sensing, transmission, and actuation), and given their constrained resources, they cannot perform complex monitoring tasks. Generally, distributed mechanisms are difficult to manage and may be computationally too expensive for resource-constrained nodes.

Based on the *model of link metrics and granularity of observations*, monitoring approaches can also be classified as hop-by-hop or end-to-end approaches. **Hop-by-hop** approaches use diagnostic tools such as traceroute, to measure hop-by-hop link metrics directly. By sending multiple probes with different time-to-live fields, traceroute can measure the delay of each hop on the probed path.

End-to-end link monitoring approaches use end-to-end metrics (*e.g.* path delays) to calculate unknown internal link metrics [72]. To distinguish between two possible sets of failures, there must exist a measurement path that traverses at least one element (node or link) in one set and none of the elements in the other set. It is highly nontrivial to place monitors, such that this condition is satisfied with minimum cost, due to the large solution space (all combinations of monitor locations) and large number of constraints (all pairs of sets of failure

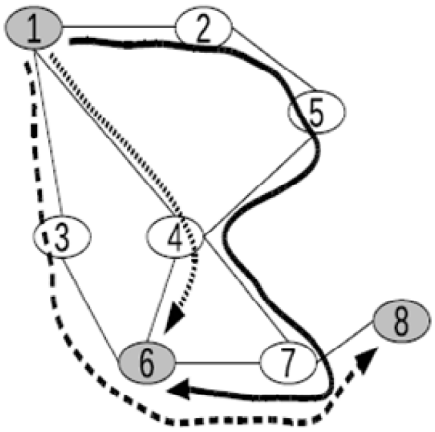


FIGURE 3.10: End-to-end monitoring approaches.

locations). For each monitored path (*cf. Fig*. 3.10), two probes (*monitors*) at the endpoints of the path keep track of the transmitted and received packets. End-to-end approaches do not rely on the cooperation of internal network elements; thus saving the resources of constrained nodes. However, network paths grow quadratically, $O(n^2)$, with the size of the network; which constitutes a high communication cost.

Network tomography (or **inferential network monitoring**) involves the estimation of fine-grained hop-by-hop link metrics, such as delay and packet reception ratio, using end-to-end measurements [73]. The basic idea is that a subset of nodes with monitoring capabilities (*monitors*), can initiate and collect end-to-end measurements of selected cycle-free paths. Unlike the approach of direct measurements that actively employ control packets, network tomography only relies on end-to-end performance (e.g., path connectivity) experienced by data packets, thus capable of reducing overhead and minimizing dependence on protocols [74].

Then, using these end-to-end metrics, network tomography techniques can decompose them to hop-by-hop link metrics by solving a linear system of equations; under the assumption that link metrics are additive and constant during the measurement, and measurement paths cannot contain cycles [75]. In order to successfully solve the linear system, sufficient measurement paths should be able to be conducted among the monitors; thus requiring the monitors assignment to comply with certain conditions [76]. All link metrics can be uniquely identified when the number of linearly independent measurement paths equals the number of links. It is, however, inefficient to collect measurements from all possible paths, whose number can grow exponentially in the number of links. The problem becomes challenging due to the existence of linearly dependent paths [77]. Nevertheless, inferring all link metrics may incur a high overhead and is not necessary in many applications. [75] developed a quadratic algorithm in the number of links which uses independent spanning trees to construct linearly independent, cycle-free paths between monitors without examining all candidate paths. A more practical and general case is only the "interesting" subset of the links, *a.k.a* critical links, is identified. However, assigning monitors to identify critical links faces non-trivial challenges due to the complex dependency between the link identifiability and each monitor [78].

3.3.2 Monitoring for Wireless Sensor Networks

The problem of assigning the minimum number of monitors to cover a given domain is known in the literature as the *minimum monitor assignment problem* (*cf.* [79, 80]); which has been proven to be NP-hard [76, 81]; implying that unless $P = NP$, efficient algorithms for solving it don't exist [82]. Monitoring costs can further be reduced by minimizing the overhead of monitoring flows on the underlying network through minimizing the number of monitored paths (*cf.* [83, 81]), and avoiding redundant measurements (*cf.* [84, 85]).

Several heuristics have been proposed to place monitors to uniquely localize a bounded number of link failures under specific probing mechanisms [73, 86, 87]. [73] proposed an efficient algorithm called MMP to assign the minimum number of monitors as well as an efficient path construction algorithm in [75]. [88] demonstrated using Integer Linear Programming formulation that there is a trade-off between minimizing the number of monitors and minimizing the communication overhead on the underlying. They proposed two greedy algorithms to efficiently balance the trade-off by jointly optimizing the two objectives.

Monitoring for Wireless Sensor Networks (WSN) have been approached frequently in the literature; the propositions extend over the aforementioned classes of monitoring mechanisms (*cf.* 3.3.1) [89, 56, 90]. Lee *et al.* [56], Paradis *et al.* [90] and Suriyachai *et al.* [91] provide surveys of network monitoring for WSNs. Fault tolerance can be attended to in the network layer, transport layer, or application layer [90].

The sniffer technology for WSNs is one of the distinguished passive real-time monitoring tools. Sniffers listen to the packets transmitted over the network; directly capture them from the shared medium in wireless networks. (*cf.* Fig. 3.11). The information obtained from the sniffed packets gives them real-time access to network operations, and the ability to promptly assess network performance and detect network malfunctions, without affecting the normal operation of the network. Information may include partial topology, routing information and, data content. There have been some related works on sniffing tools for WSNs in academia and industry, such as SNA [92], ZENA [93], SANMP [94] and SNDS [95]. However, their high costs and lack of analysis of integration with LoWPAN protocols such as 6LoWPAN and RPL diminish their application into the IoT domain [96].

Targeting efficient monitoring for WSNs, [97] utilized Software Defined Networks (SDN); which is a looming technology for network management. It benefits from the advantages of centralized monitoring without suffering from its pitfalls. SDN disconnects the network management and control functions from the packet routing processes. When applied to WSNs, this decoupling achieves centralized monitoring with its global view of the network state without exhausting the resources of constrained nodes. [98] and [99] provide a review on the application of SDNs to managing WSNs; whereas [100] and [101] survey its employment to IoT.

3.3.3 Monitoring for IoT Networks : Research Gap

Unlike WSNs, most of the research work pertaining to fault-tolerance in the IoT focus on security [27, 103, 104, 105], intrusion detection [106, 107], or anomaly detection [81]. These solutions are security-oriented and do not answer to the monitoring aspects that target the underlying network structure; specifically, guaranteeing node availability and stable, reliable, and scalable end-to-end connectivity. A survey of the state-of-the-art of security in IoT is conducted by [27].

The few researches which tackle the related IoT network-layer monitoring problems often suffer from being heavy-weight, or depend only on highly-powered nodes [104]; which are a minority in the IoT topologies. [105] investigated how to adapt existing IP-based network management protocols, namely SNMP and NETCONF, so they can be implemented on

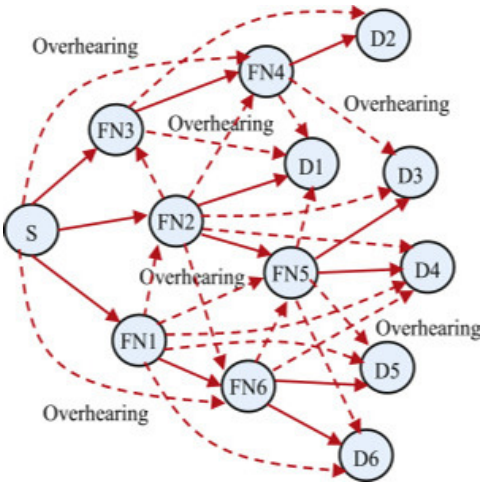


FIGURE 3.11: Overhearing (sniffing); **S** : source; **D** : destination and **FN** : forwarding nodes [102].

resource-constrained devices. Service interfaces were simplified to include a subset of their functions in order to minimize network overhead. The authors conclude that the time and memory requirements were low with only trivial security levels. However, enabling authentication and privacy increases message processing times significantly [104].

Our research is directed towards the optimal placement of monitoring devices to cover a given network topology. Even though the objective, and therefore the methodology, are different; works dealing with IoT and WSNs sensor coverage optimization problems are somehow related. For instance, Chafii *et al.* [108] proposed a method for enhancing the IoT sensing area coverage based on machine learning algorithms, and Dou *et al.* [109] proposed a connectivity model for improving the sensing coverage in WSNs.

The most relevant work is a mechanism for passive monitoring with RPL; proposed by Mayzaud *et al.* [104]. The monitoring responsibility is exclusively put on higher-order devices which are not limited in their resources; to reduce the overhead on the constrained IoT nodes. The mechanism imposes a hard constraint; since typically, higher-order devices do not constitute the majority of nodes in IoT networks. According to their location and the topology, it might be impossible to entirely cover the critical set of nodes and links using only highly powered devices. Moreover, determining the optimal placement of monitoring nodes was beyond the scope of their proposed work. After a comprehensive literature review and up to our knowledge, the optimal placement of monitors to cover a mission-critical IoT network has not been proposed so far. Particularly missing are monitoring models with optimized, energy-efficient role scheduling, and integration with RPL and 6LoWPAN protocols.

3.3.4 Related Sleep Scheduling Approaches

On account of the fact that idle listening to the radio channel is energy consuming; to maximize longevity, duty cycling (*a.k.a.* sleep scheduling) is frequently applied in constrained networks. In duty cycling, devices slip into a sleeping state and periodically wake up to perform their sensing, receiving, or transmission roles. Adding a monitoring functionality to the constrained devices should be duty cycled, as well. Therefore, another class of related works targets optimized sleep scheduling in LLNs. Several models were proposed for optimized duty cycling in WSNs [110] [111], [112] and for task scheduling in general [113], [114], [115].

The mainstream of research on sleep scheduling can be divided into : (1) coordinated (*a.k.a* synchronization-based) [**simon2007dependable**, 116]; where nodes synchronize with

each other to coordinate their wake up schedules, (2) asynchronous (*a.k.a.* random) [117]; which do not involve explicit synchronization, or (3) centralized; where it is assumed that a global clock and a predefined sleep-wake schedule is centrally available to all nodes [30].

In random scheduling nodes make local decisions on whether to sleep or stay awake to ensure communications [118]. Each node bases its decision on an estimate of how many of its neighbours will benefit from its being awake and the amount of energy available to it. Random mechanisms can lead to randomly covered/uncovered parts; which does not match with the hard requirement of mission-critical IoT applications [119].

Centralized solutions, on the other hand, need intensive communication between the nodes and a central node to collect data on the potential monitors and to send control messages from the central node to the monitors [30]. It assumes network-wide information distribution. In a large network it would mean excessive amount of messages transfer and thus the solution would impose an important communication communication overhead. For continuous monitoring systems, synchronization-based sleep scheduling schemes are often used because the traffic pattern is periodic. Fine-grained synchronization is required between the sender and the receiver, so that they can wake up at the same time to communicate.

A wide range of exact and heuristic approaches has been proposed to maximize network lifetime by using heuristic criteria [120, 121] and hybrid approaches as linear programming based rounding methods [122]. The WSNs lifetime considered in this work is the period of time through which the WSN is perfectly completing its function. This lifetime is affected by many factors including the amount of energy available, failure probability and components degradation. The amount of energy available becomes the most important factor in case of non renewable components applications [123]. Exact approaches based on column generation (CG) to solve coverage and scheduling problems in wireless sensor networks ([124, 125, 121]).

Chapter 4

Three-Phase Decomposition for Monitoring Placement & Scheduling

4.1 Problem Statement, Assumptions & Objectives

The problem addressed in this research is the efficient, full monitor coverage in IoT networks for a predetermined lifetime. We target the monitoring of mission-critical, 6LoWPAN-based LLN ; which utilize RPL for routing. (*cf.* Chapter 2 for examples of critical IoT network services). The ultimate objective of the monitoring mechanism is providing robust network connectivity. We aim to proactively and efficiently detect networks faults ; by continuously testing the availability of nodes ; via monitoring the status of the entire set of links present in the network.

It is crucial that the monitoring mechanism has minimal effect on energy consumption and traffic load ; to leave the LLN unconstrained to perform its primary function of sensing, actuation or transmission. To that end, this work aims to minimize the monitoring costs in terms of energy consumption and communication overhead. Monitoring costs depend on the type of the adopted monitoring approach, *i.e.* whether it is centralized or distributed, active or passive, ... *etc.* (*cf.* Chapter 3 for the different monitoring approaches). The most common IoT architectures are completely centralized ; mainly due to security reasons. The 6LoWPAN Border Router (6BR) is the central entity and is always assumed to be accessible [28].

Through multihop communication, the gathered monitoring data can be forwarded from monitors to the 6BR, then to a Networks Operations Center (NOC) ; where sophisticated data analysis and mining can be performed. Therefore, for the problem in hand, the communication cost comes in the form of the energy lost due to the communication between monitors and the 6BR. To minimize this cost, it is necessary to find the shortest path, in terms of the number of hops from monitors to the 6BR.

It is assumed that a link can only be monitored by its extremities (*i.e.* endpoints). The monitoring infrastructure should be built such that monitors are embedded (*placed*) in the correct locations to guarantee full monitoring coverage. To reduce the monitoring cost ; the number of monitors to be placed has to be minimized while satisfying the coverage condition. It also assumed that each node has several activities independent from monitoring ; therefore, the monitoring activity cannot consume more than a specified limited percentage of a node's battery.

Given that LLN are mostly constrained by energy consumption ; idle listening to the channel can quickly cause battery depletion. Hence, the incorporation of duty cycling mechanisms is necessary to reduce the average power consumption and maximize longevity.

Normally, the things' active primary functions of sensing, actuation or transmission are periodical across the network lifetime. In a given period, if a node is not assigned to perform a certain role, it can be put into a sleeping state. On the other hand, for critical applications, it is required that monitoring coverage is always guaranteed throughout the entire network lifetime ; such that each link is monitored by at least one monitoring node ; regardless of

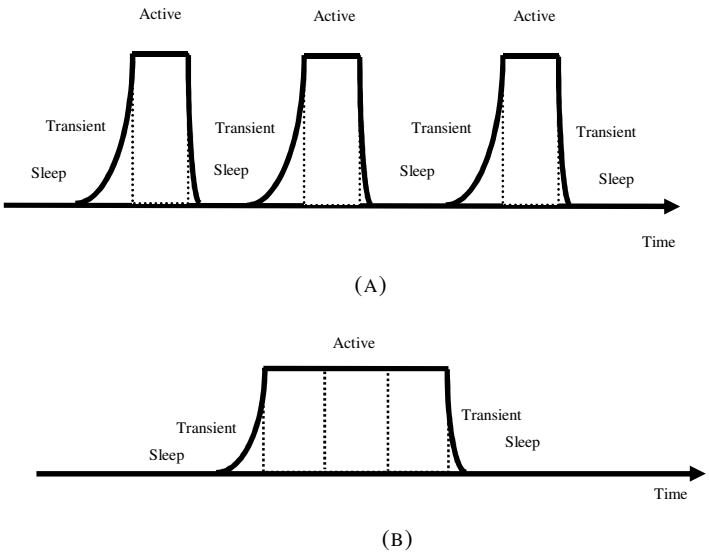


FIGURE 4.1: State transitions : (A) before merging active states, (B) after merging active states.

the lack of current network activity. Furthermore, energy consumption during the transient state can be high [126], and numerous state transitions for the nodes consume extra energy (*cf.* Fig. 4.1a). However, if the node's roles are merged together, the energy consumed in state transitions can be reduced (*cf.* Fig. 4.1b). Therefore, effective and energy-efficient monitoring scheduling algorithms are necessary with duty cycling; not only to satisfy the monitoring coverage constraints, but also to minimize state transitions and balance the distribution of the monitoring burden among nodes, and thus maximize longevity. It should be noted that the active/sleep alternation addressed in this context is the turn on/off of the monitoring activity of the node, regardless of the other activities a node may perform.

To recap, the monitoring mechanism is built based upon the following **assumptions** :

1. the IoT network is 6LoWPAN-based, uses RPL for routing, and performs a critical mission,
2. the monitoring approach is proactive and centralized; with the 6BR as the central entity,
3. a duty cycle mechanism is adopted, and the monitoring function is periodical across the planning horizon,
4. the active/sleep alternation is the turn on/off of the monitoring activity of the node, and
5. a link can only be monitored by its extremities.

Given the stated monitoring assumptions, the mechanism's **objectives** are :

1. minimize the number of placed monitors while satisfying the coverage condition,
2. place the monitors such that the total communication energy consumed in the path between monitors and the 6BR is minimized,
3. minimize the monitoring state transitions, and
4. balance the monitoring role among nodes.

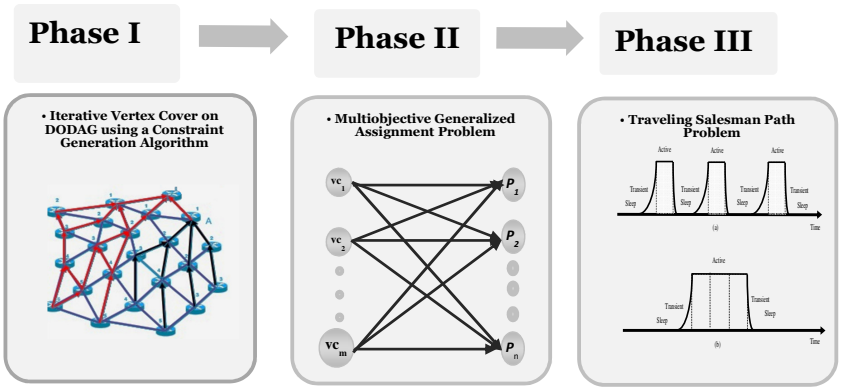


FIGURE 4.2: Three phases of the monitoring mechanism.

4.2 Centralized Monitoring Mechanism

To tackle the IoT networks monitoring problem. The first step of analysis addresses the location (placement) of the monitoring nodes to cover the entire set of edges required to be monitored; the edges corresponding to the links participating in the critical mission. However, a single subset of nodes may not have enough energy to monitor the network for the complete planning horizon; which is the pre-determined lifetime specified by the NOC. Therefore, multiple sets monitors are required.

Since a duty cycling approach is adopted for the primary functions of the things, we propose that the monitoring mechanism follows a duty cycle as well across a given planning horizon. We assume that the monitoring system reports the status of network components in prescheduled epochs. The frequency of epochs depends on the criticality of the application. At each epoch, a subset of monitors is active to perform its monitoring responsibility and then goes back to sleep-monitoring, then another continues the job. The NOC is responsible for planning the periodic monitoring schedule. Therefore, the optimization is external from the resource-constrained things.

We define the monitoring responsibility as observing the network traffic to verify the availability of the critical set of nodes; which is known as *passive monitoring*. If required, monitors can also collect link and node performance measurements; such as node energy level, end-to-end delay, link quality level, *etc.*. If the monitoring responsibility includes participating in the network traffic; via probing the monitored neighborhood and collecting the response, then *active monitoring* is assumed (*cf.* Chapter 3 for details on different monitoring approaches). Moreover, they can control the routing of measurement packets.

If detailed node and link performance metrics are required by the NOC, we introduce the utilization of RPL's control messages to the monitoring mechanism; which are broadcast to build and maintain the routing structure. RPL constructs a graph known as the DODAG [127, 128]; the name is an acronym for Destination Oriented Directed Acyclic Graph (*cf.* Chapter 3 for more details on RPL). The root of the DODAG is responsible for its construction; which starts by broadcasting a DODAG Information Object (DIO) that contains the configuration of the DODAG [28]. We propose to include a header in the DIO for metrics and constraints objects. This header is called DAG Metric Container object [129]. Node Energy and Link Reliability are examples of the metrics/constraints that could be included in the DAG Metric Container object. When the DIO traverses the DAG, each node augments a sub-object to the message, which expresses its value to the metric used.

During the lifetime of monitoring, communication and transition costs are incurred. We assume a centralized monitoring approach; meaning that the monitors are ultimately transmitting the gathered information about the monitored nodes to a central node. In our mechanism, we assume that the IoT network utilizes RPL for routing. Therefore, we leverage the routing topology, and the mechanism transmits the monitoring data to the root of the DODAG; which has the global view of the network. The transmission is done through multi-hop communication. Consequently, the longer the path to the root in number of hops, the higher is the communication cost.

At any time interval, the active node set is known from the DODAG. In a duty cycled monitoring approach, the transition cost is the cost of alternating from one Vertex Cover set of monitors in a monitoring period to another set in the next. The objective is to minimize the overall monitoring costs while ensuring monitor coverage of the entire DODAG throughout the planning horizon. The energy consumption of the monitors could further be minimized if the time periods where a node is actively monitoring are merged together (Fig. 4.1b). To that end, it is required to find the optimal sequencing between the sets of monitors across the time periods in a way that minimizes the state transitions of nodes. The modeling and mathematical formulation of the monitoring optimization are discussed in the following section.

4.3 Three-phase Modeling of Monitoring Optimization

The IoT network could be represented by its logical graph constructed by the routing protocol RPL, namely the DODAG [127, 128]. Consider an active DODAG $D = (V, E)$ where V represents all the vertices, $V = \{v_k, k = 1, 2, ..., q\}$ and E is the set of edges. For monitoring a time horizon $T = \{T_j, j = 1, ..., n\}$, the duty is cycled between several sets of nodes; each node has a reserved battery for monitoring ($reservedBattery_k$). The monitoring activity for one period consumes energy (eM). Scheduling between several sets of monitors for minimum energy consumption is an NP-hard problem. This remains true even in the very special case where the $reservedBattery_k$ for monitoring is sufficiently large; such that one set of monitors can cover the entire planning horizon. Finding the set of vertices to cover an entire graph is also NP-hard [130, 131].

We propose a three-phase decomposition of the monitoring mechanism (the proposition is published in [5]). The three proposed phases are modeled using well-known optimization problems in the literature (Fig. 4.2). Although these problems are NP-hard in nature, approximation algorithms to solve them are available in the literature (*cf.* in [130, 131, 132]). Moreover, decomposing the original problem allows a finer reduction of the search space and, hence, reduction in the solution's complexity.

As mentioned in section 4.1, the first step of analysis is to ensure the correct placement of monitors to cover the critical set of links; which is handled in the first phase, Phase I. We model the monitoring coverage as the classic Minimum Vertex Cover (MVC) problem [133]; which by definition computes the minimum Vertex Cover (VC) set that covers the entire set of edges. To address the periodical functioning of the monitoring role, several Vertex Covers are generated by iteratively solving the MVC. Only *minimal* Vertex Cover (VC) sets are interesting for monitoring. If a node set is not minimal for the MVC then it contains a subset which is minimal and can cover all the edges with less cost.

To prolong network longevity, the monitoring role should be distributed and balanced between the entire set of nodes. To that end, assuming periodical functioning, several monitoring sets (Vertex Covers) are required to achieve monitor scheduling while minimizing and balancing the energy consumption of the monitors. For this purpose, Algorithm 1 is developed to get multiple solutions of the same MVC, after incrementally adding new constraints to reduce the search space. Reducing the search space through incrementally adding new

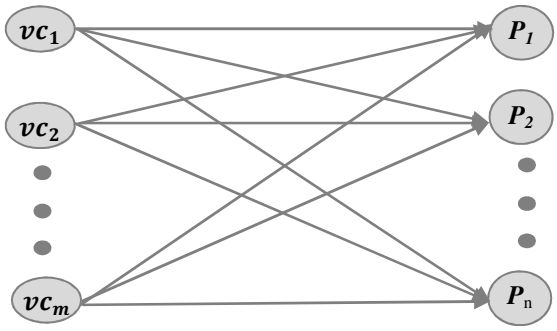


FIGURE 4.3: Assignment of Vertex Covers i to periods j

constraints is known as the Constraint Generation approach [134]. The output of Phase I is several sets of Vertex Covers.

The working schedule of the Vertex Covers across the planning horizon is still required. Given a planning horizon (defined by the NOC), divided into several time periods, we propose to optimally assign (a subset of) the Vertex Covers to time periods in Phase II ; by modeling the scheduling as a multi-objective Generalized Assignment Problem (GAP) [135]. GAP is a special type of optimization problems where agents are assigned to perform multiple tasks (Fig. 4.3). For the problem in hand, the agents are the Vertex Cover sets (or a subset of them) that should be assigned to time periods. The **objectives** of Phase II are :

— minimize the total energy consumption for monitoring, and
— minimize the communication cost across all Vertex Covers, from the monitors to the DODAG root.

The **assumptions** of Phase II are :

— any node can be selected as monitor, a node may be a member of several Vertex Covers,
— some Vertex Covers may monitor one period, more than one period, or none at all,
— each period is to be assigned exactly one Vertex Cover, and
— the monitoring cost is defined in terms of the communication cost incurred by monitors and the energy loss due to monitoring.

The last phase, Phase III, is sequencing the Vertex Covers with the objective of minimizing nodes' state transitions from one period to the next. The sequence generated determines the number of times a node's state is being changed from asleep to awake and vice versa. Phase III is modeled as a Traveling Salesman Path Problem (TSP-Path) [136]. We are interested in finding the minimum weighted Hamiltonian Path (in terms of nodes state transitions) from an arbitrary starting point. A Hamiltonian Path (HP) is a path that visits each vertex exactly once without the need to return to the starting vertex [137]. The vertices in this case are the selected Vertex Covers from Phase II. The starting vertex of the HP is completely arbitrary. The edges represent the transition costs. The result (the path) gives the optimal sequence (scheduling) of the selected VC.

4.4 Mathematical Formulation of Monitoring Optimization

The three phases include the following sets :

— Set V : represents all the vertices in the DODAG where $V = \{v_k, k = 1, 2, ..., q\}$.
— Set S : represents all the Vertex Covers (VC) obtained from Phase I ; where $VC \subset V$ represents the subset of monitoring nodes and $S = \{vc_i, i = 1, 2, ...m\}$.

— Set T : contains the disjoint time periods covering the entire planning horizon ; which the Vertex Covers are assigned to monitor. $T = \{T_j, j = 1, ..., n\}$.

4.4.1 Phase I : Generating Multiple Vertex Covers

Let $D = (V, E)$ be the DODAG where V is the set of vertices and E is the set of edges. A subset $VC \subset V$ is a minimal Vertex Cover of D if for every edge $(u, v) \in E$, either $u \in VC$ or $v \in VC$ or both $u, v \in VC$ and VC is irreducible, i.e. no vertex can be removed from the VC without losing the coverage property.

Phase I computes set S which includes the m Vertex Covers such that for each edge in the DODAG at least one of its endpoints belongs to vc_i. Moreover, no Vertex Cover in S should be a subset of another such that : $\forall vc_i \subset S, \nexists vc_j \subset S | vc_i \subset vc_j$.

Let the decision variables v_k express whether the vertex v_k is in the Vertex Cover or not. The objective is to minimize the total number of vertices in the Vertex Cover, subject to the constraint that at least one vertex of the edge (v_i, v_j) is a member of the VC. The problem is the binary optimization problem represented by Model 4.1. The binary integer program

TABLE 4.1: Model I, Minimum Vertex cover Problem

Decision Variables

$$v_k = \left\{ \begin{array}{ll} 1, & \text{if } v_k \text{ is chosen in a Vertex Cover} \\ 0, & \text{otherwise} \end{array} \right\} \tag{4.1}$$

Binary Program

$$\min \quad \sum_{k \in V} v_k \tag{4.2}$$

s.t.

$$v_i + v_k >= 1 \quad \forall (v_i, v_i) \in E \tag{4.3}$$

$$v_k \in \{0, 1\} \quad \forall v_k \in V \tag{4.4}$$

in Model 4.1 is solved to get the minimum set of monitors, i.e. the minimum Vertex Cover. In order to obtain multiple Vertex Covers, Algorithm 1 is used to solve the MVC iteratively. Each VC obtained using Algorithm 1 satisfies the constraints of Model 4.1.

Algorithm 1 Phase I, MVC Constraint Generation

Input: $D \longleftarrow \{v_j : v_j \in V, E\}$, **MVC** *Initial optimal solution* $VC_1^* \longleftarrow$ {*objective function* z_1^*, *matrix of constraints* A_1^*, *right hand side* b_1^*}
Output: Matrix of solutions M

```
while (feasible minimal VC exist) do
    M ⟵ vc_i ;
    A ⟵ A + vc_i^T ;
    b ⟵ b + (z_{i-1}) ;
    i ⟵ i + 1 ;
    [vc_i, z] ⟵ SOLVE-MVC(A, b) ;
end while
return M ;
```

Algorithm 1 works as follows : construct a matrix M where all solutions of the MVC are stored, and a z vector which contains the corresponding objective function value to each solution in M. Initially, M contains the optimal solution obtained after solving Model 4.1 using

branch-and-bound algorithm, and z stores the minimum VC. Then, the MVC is solved iteratively after adjusting the integer program by adding previous solutions to the set of constraints, thereby reducing the search space. The algorithm terminates when no other feasible minimal Vertex Covers can be found. In each iteration, a column representing the new solution, which has a cardinality greater than or equal to the previous solution, is appended to matrix M and the corresponding objective function value is added to the z vector, such that :

$$vc_i^T \cdot M < z \tag{4.5}$$

To illustrate, let the initial solution be vc_1, and the corresponding objective function value be z_1. To obtain vc_2, solve the integer program $Ax \leq b$ after adding the following constraint to the matrix of constraints (A) and the vector of RHS (b) :

$$vc_2^T \cdot vc_1 < z_1 \tag{4.6}$$

In this way, to solve for vc_3 the following two constraints are appended :

$$vc_3^T \cdot vc_1 < z_1 \tag{4.7}$$

$$vc_3^T \cdot vc_2 < z_2 \tag{4.8}$$

To solve for vc_m the following $(m-1)$ constraints are appended :

$$vc_m^T \cdot vc_1 < z_1. \tag{4.9}$$

$$vc_m^T \cdot vc_{m-1} < z_{m-1} \tag{4.10}$$

4.4.2 Phase II : Assigning time periods to Vertex Covers

In Phase II, the planning horizon is divided into several periods. Then, a mathematical model is developed to optimally associate the Vertex Covers to periods throughout the planning horizon. The assignment does not define any ordering or sequencing of the time periods.

Given a planning horizon denoted by $T = \{T_j, j = 1, ..., n\}$, define a binary decision variable $s_{i,j}$ that indicates whether a Vertex Cover vc_i is assigned to monitor a period T_j (Eq 14).

TABLE 4.2: Model II, Parameters

Term	Description
h_k	Number of hops traveled from each v_k in vc_i to the root of the DODAG
H_i	Total number of hops traveled by all v_k in vc_i, $H_i = \sum y_{k,i} \cdot h_k \forall k \in V$ (4.11)
eM	Energy loss per each monitoring period assigned for v_k
$reservedBattery_k$	Maximum battery allowed for monitoring

Any node v_k may be a member of several Vertex Covers. The current $consumedEnergy_k$ of node v_k is calculated using (Eq. (4.14)). Equation (4.14) states that : the current $consumedEnergy_k$ for monitoring, depends on the number of times a Vertex Cover set, including v_k, has been assigned to a period, multiplied by the energy loss (eM) per each monitoring period. The

TABLE 4.3: Model II, Multi-objective Generalized Assignment Problem

Parameters

$$y_{k,i} = \left\{ \begin{array}{ll} 1, & \text{if } v_k \text{ is chosen in a Vertex Cover} \\ 0, & \text{otherwise} \end{array} \right\} \tag{4.12}$$

Decision Variables

$$s_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } vc_i \text{ is assigned to period } j \\ 0, & \text{otherwise} \end{array} \right\} \tag{4.13}$$

Binary Program

$$\min F_1 = \sum_{k \in V} eM \cdot (\sum_{i=1}^{m} \sum_{j=1}^{n} y_{k,i} \cdot s_{i,j}) \tag{4.14}$$

$$\min F_2 = \sum_{i=1}^{m} \sum_{j=1}^{n} H_i \cdot s_{i,j} \tag{4.15}$$

s.t.

$$s.t. \quad \sum_{i=1}^{m} s_{i,j} = 1 \quad \forall j \in T \tag{4.16}$$

$$eM \cdot (\sum_{i=1}^{m} \sum_{j=1}^{n} y_{ki} \cdot s_{i,j}) \leq reservedBattery_k \quad \forall k \in V \tag{4.17}$$

$$s_{i,j} \in 0,1 \forall i,j \tag{4.18}$$

energy loss is the same for each active monitor and for each period. Define a binary variable $y_{k,i}$; which indicates whether a vertex v_k is a member of Vertex Cover vc_i or not (Eq. (4.16)). The variable $y_{k,i}$ is the output of Phase I, and hence it is a parameter in Phase II. When $y_{k,i}$ is multiplied by $s_{i,j}$ and summed over the Vertex Covers and the periods, the result is the number of times a vertex v_k has been assigned to monitor a period (Eq. (4.14)).

Energy is lost due to communication between the monitoring devices and the Border Router. The energy lost for communicating the monitoring data is not part of the $reservedBattery_k$ for monitoring. On the other hand, this energy loss affects the rest of the battery that is not dedicated to monitoring. Accordingly, it is necessary to find the shortest path, in terms of the number of hops to the BR. The objectives in Phase II are twofold (refer to Eq. (4.14) - (4.15)). At this stage, it is required to determine how all assignments should be made while minimizing the total number of hop counts travelled by all members of the Vertex Covers. This is done while minimizing the total energy spent for all nodes. Denote the number of hops travelled from each monitoring node v_k in vc_i to the root as h_k, and denote the total number of hops travelled by all v_k in vc_i by H_i (Eq. (4.11)). Constraint (4.16) indicates that each period must be monitored by one Vertex Cover. Constraint (4.17) ensures that the energy consumed for monitoring never exceeds the $reservedBattery_k$.

4.4.3 Phase III : Sequencing Between Assigned Vertex Covers

The objective of Phase III is to minimize the nodes' state transitions from one Vertex Cover to the next. If a node v_k belongs to more than one Vertex Cover set vc_i, the model

associates consecutive periods to the Vertex Cover sets containing the repeated node. Accordingly, the number of times a node needs to start up to perform its assigned monitoring is minimized. This Phase is modeled as TSP-Path. Details of the TSP-Path are shown in Model 4.4, where the Miller, Tucker and Zemlin (MTZ) mathematical formulation [132] is adopted. The decision variables and parameters are adjusted to in Model 4.4 such that :

— the cities here are the unique Vertex Cover sets assigned to periods $T = \{T_j, j = 1, ..., n\}$, *i.e.* the unique results from the previous assignment model (Model 4.3),
— define a binary decision variable $x_{i,j}$ denoting whether vc_j is selected in the path after vc_i (Eq. (4.20)), a feasible solution is a path (Hamiltonian Path) that passes through each set exactly once (Eq. (4.22) - (4.23)),
— the cost (distance) $C_{i,j}$ of moving from one city (Vertex Cover set) to the next; from vc_i to vc_j, is the total number of state transitions of the members of the sets, from active to sleep and vice versa (Eq. (4.19)),
— the objective is to find the least costly sequence of Vertex Covers over the planning horizon (Eq. (4.21)), and
— extra variables u_i are required for subtour elimination, in the constraints expressed in Eq. (4.24) - (4.25).

TABLE 4.4: Model III, Traveling Salesman Path Problem

Parameters

$$C_{i,j} = \sum stateTransitions_{i,j} \tag{4.19}$$

Decision Variables

$$x_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } vc_j \text{ is chosen for monitoring after } vc_i \\ 0, & \text{otherwise} \end{array} \right\} \tag{4.20}$$

Binary Program

$$\min \quad \sum_{i=0}^{n} \sum_{j=1,\, j!=i}^{n} C_{i,j} \cdot x_{i,j} \tag{4.21}$$

s.t.

$$\sum_{i=0,\, i!=j}^{n} x_{i,j} = 1 \quad \forall j \tag{4.22}$$

$$\sum_{i=0,\, i!=j}^{n} x_{i,j} = 1 \quad \forall i \tag{4.23}$$

$$u_i - u_j + (n-1) \cdot x_{i,j} \leq n-2, \quad i, j = 2, ..., n, i! = j \tag{4.24}$$

$$1 \leq u_i \leq n-1 \quad i = 2, ..., n \tag{4.25}$$

4.5 Implementation & Analysis

4.5.1 Problem Resolution & Implementation

Solving the problem starts with Phase I by generating a DODAG and outputs matrix M. Multiple solutions of the Vertex Cover Problem are stored in matrix M. Algorithm 1 runs until no more feasible minimal VC exist. However, for experimentation purposes, it has been

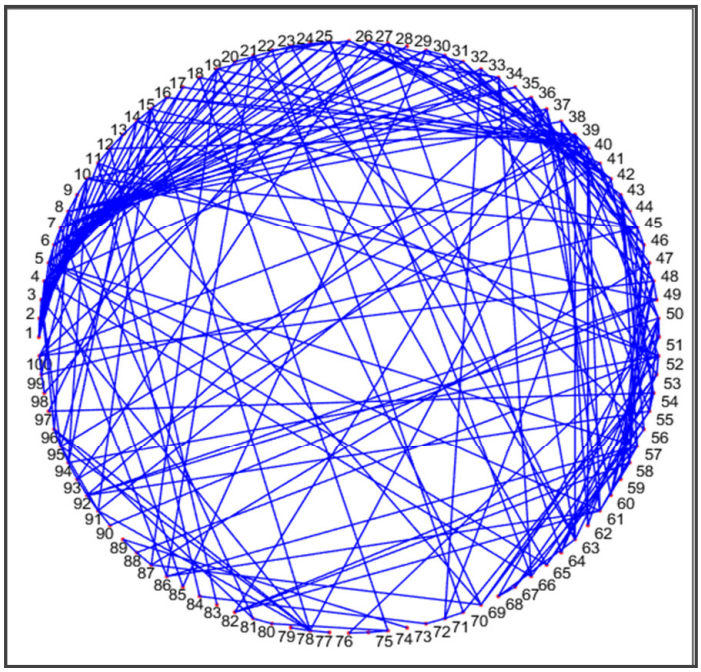


FIGURE 4.4: A DODAG of a network with 100 nodes and 234 links.
Node number 1 is the root.

shown to suffice to loop any number of times between the range [1.5|T| - 3|T|]. This range denotes the required number of Vertex Covers (θ).

Analytical simulation is conducted for networks of 50, 100, 150 and 200 nodes with varying numbers of links. Table 4.5 shows the characteristics of the corresponding DODAGs, a summary of the results of the three phases for 8 different DODAGs, as well as the metrics used for evaluation. DODAGs are constructed such that there is a root (BR) and each node has at least one path towards it. The edges on the DODAGs are constructed by randomly generating (x, y) positions of each node in a unit square (units do not matter in the graph). The distance between every two nodes is measured; if it is less than a certain Threshold parameter then the two nodes are connected. By varying the Threshold parameter, in the range [0–1], the DODAG gets sparser or denser. Figure 4.4 shows an example of a DODAG with 100 nodes and a Threshold of 0.09, which gives 234 links. Varying the density of the graph affects the number of nodes required to monitor the entire DODAG. Figure 4.5 depicts the effect of varying the density of a DODAG, with 100 nodes, on the percentage of monitors. It is evident that the more the number of communication links between the nodes, the more monitors are required.

Throughout the entire set of experiments, it is assumed that the planning horizon T is divided into 10 time periods, and that the energy loss eM per monitoring period assigned for v_k is 2% of its total energy. Running Phase I with DODAG number 3 in Table 4.5, for example, (DODAG shown in Fig. 4.4), outputs matrix M which includes 15 different Vertex Covers $(VC_1, ..., VC_{15})$. M is the input of the multi-objective Generalized Assignment Problem in Phase II.

The model in Phase II a Multi-objective Mathematical Programming (MMP) problem. There are several approaches to solving MMP problems in the literature. This research adopts the ϵ-constraint method. This is due to its several advantages over its rivals [138]. In the ϵ-constraint approach, only one objective function is optimized, while the others are added to the constraints. Pareto-optimal solutions are achieved by varying the right hand side of the

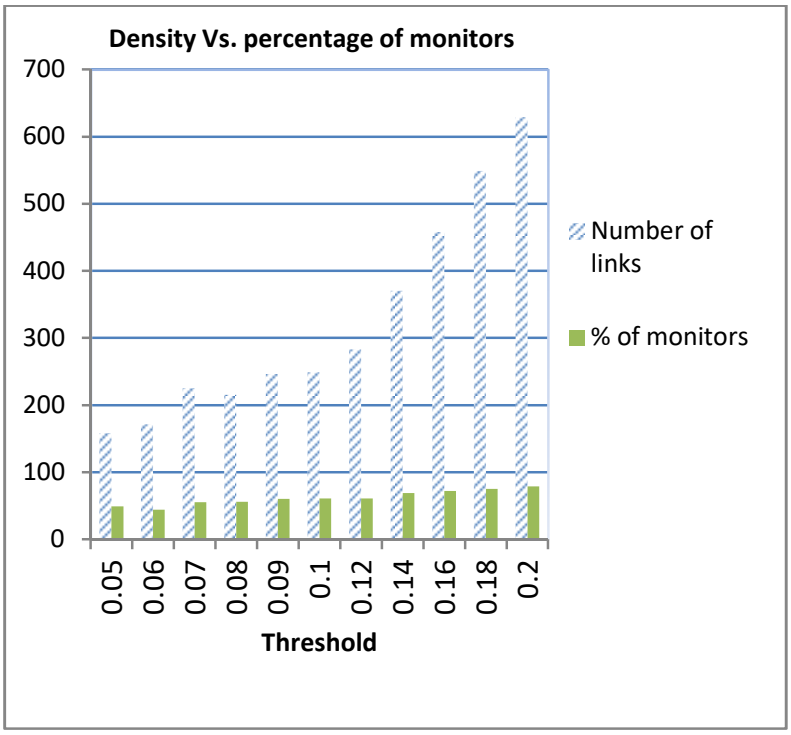


FIGURE 4.5: Effect of varying the density of the DODAG on the percentage of monitors.

constrained objective functions.

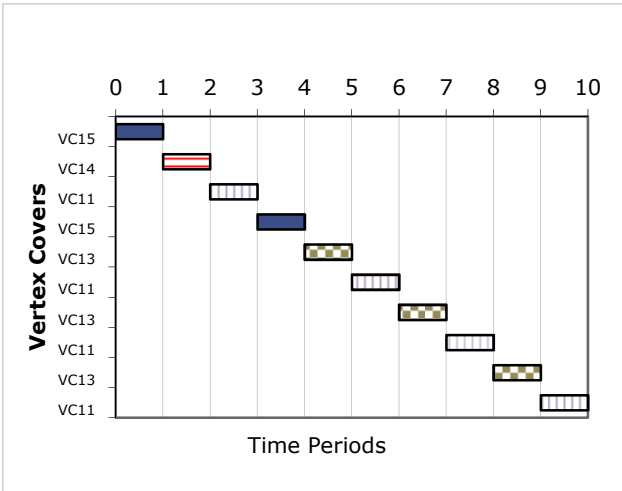
Although it is widely used, the ϵ-constraint method has its disadvantages. First of all, it is required to calculate the range of every objective function used as a constraint. The usual way is to build a payoff table. It includes the optimal solution for each objective function optimized individually. There is no guarantee that these optimal solutions are indeed non-dominated solutions [139]. Another weak point is that the optimal solution is not guaranteed to be an efficient solution if there are alternative optima. To overcome some of the limitations of the ϵ-constraint approach, (AUGMECON) [139] was developed. AUGMECON guarantees the Pareto-optimality of the solutions by using lexicographic optimization of the objective functions. Therefore, the augmented ϵ-constraint method (AUGMECON) is implemented in this work. Through using AUGMECON method for Phase II, the output is the scheduled assignment of the Vertex Covers over the periods, which is represented by a binary decision variable $s_{i,j}$.

Input to Phase III is the unique Vertex Covers assigned to monitoring in Phase II. Output of Phase III is the sequence that minimizes the total number of state transitions of the nodes. The sequence is generated using a dynamic programming implementation of the Traveling Salesman-Path Problem. Figure 4.6 shows a comparison between the sequencing of the Vertex Covers assigned in Phase II (Fig. 4.6a) on DODAG number 3, and the sequence after solving the TSP-Path of Phase III (Fig. 4.6b). The new sequence reduces the total number of state transitions by 70%.

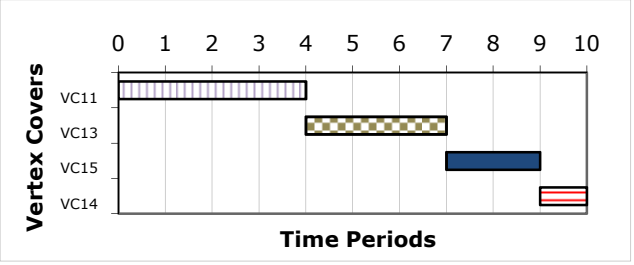
4.5.2 Performance Evaluation

The experiments were performed on a personal computer with 8 Gigabytes RAM and an Intel Core i7 processor @2.20 gigahertz. Considering the problem formulations presented in Section 4.4, the metrics used for evaluation (shown in Table 4.5) are :

— percentage of nodes selected as monitoring nodes,



(A)



(B)

FIGURE 4.6: Assignment of Vertex Covers : (A) before TSP, (B) after TSP.

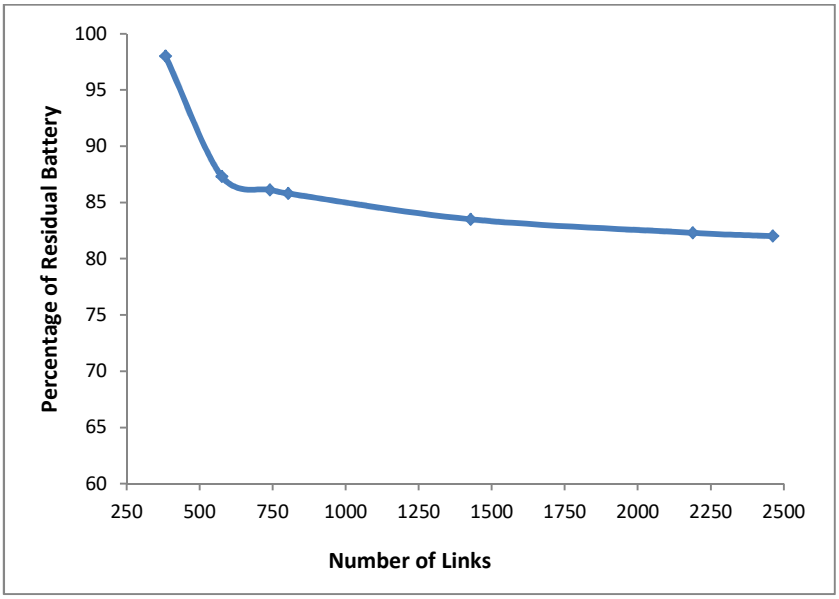


FIGURE 4.7: Average residual battery after monitoring for a network of 200 nodes and varying number of links.

— average residual energy among monitors, and
— percentage of reduction in the nodes' state transitions.

The originality of this work is mainly in the proper modeling of the defined problem. Modeling the monitor selection problem as MVC in Phase I guarantees optimal monitor placement to cover the entire set of links. Simultaneously, the percentage of deployed monitors is relatively small, (52% - 66%, depending on the density of the network). Consequently, it is possible to identify fine-grained performance link metrics, where monitors send passive probes in an end-to-end approach. This leads to reducing the need for active probes; thus less monitoring overhead. In addition, active probes can be used when needed, to check the availability of network parts where communication has not been established for long times.

Moreover, the proposition in Phase II is able to optimally assign monitors to periods with minimum energy consumption, depending on the number of periods ($|T|$), the energy loss per period (eM), and the reserved battery per node ($reservedBattery_k$). The values of the parameters used for the test instances have been chosen arbitrarily. This is because the main overriding objective of this part of the work is to emphasize that energy-efficient solution methods for this difficult problem do exist. That being said, it can be seen that, after setting the parameters to the values : ($|T| = 10, eM = 2\%, reservedBattery_k = 50\%$ and $initialBattery_k = 100\%$), the average residual battery throughout the 8 instances depicted in Table 4.5 is in the range [86% – 98%]. It is interesting to emphasize that when the battery dedicated for monitoring is sufficiently large, fewer Vertex Covers are assigned to periods and less monitor scheduling is required. On the other hand, when the $reservedBattery_k$ is relatively small, more Vertex Covers are required to monitor the same number of periods and scheduling for minimal energy consumption is critical. Figure 4.8 shows the minimum, average, and maximum remaining (residual) battery after monitoring for a network of 50, 100, 150 and 200 nodes. For $eM = 2\%$, the minimum residual battery does not fall below 80% and the average residual battery is between the range [83 – 90] %.

Also, when the model was tested without running Phase I, all the nodes were assigned to monitoring. Consequently, the nodes' battery level dropped to the minimum level. Figure 4.7, emphasizes the model's capability of monitoring a dense network of 200 nodes with

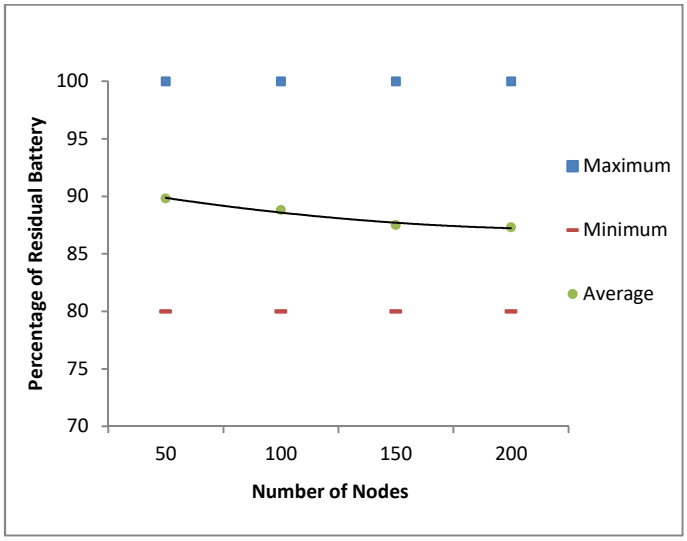


FIGURE 4.8: Residual battery after monitoring for different-sized networks.

increasing number of links, without depleting the nodes' reserved battery for monitoring. The figure shows that the density of the network grows up to 2463 links and yet the average residual battery is in the range [82 – 98]%.

The results after solving the TSP-Path, for the optimal sequencing of the Vertex Covers across the time periods, are very promising. The model is both effective and efficient in reducing the state transitions of nodes up to 80%. It is worth mentioning that in some instances, like DODAG line 8 in Table 4.5, the solution from Phase II is already optimal with respect to the number of state transitions. Therefore, the percentage reduction of state transitions after the TSP-Path is sometimes zero.

4.6 Complexity Analysis

The proposed model's time complexity depends on several factors :
— $|V|$: size of the DODAG, in terms of the number of nodes,
— $|M|$: number of Vertex Covers obtained from Phase I,
— $|T|$: number of periods in the planning horizon, and
— u : number of unique Vertex Covers assigned to monitor the periods (solution of Phase II).

Lemma 4.6.1. *Phase I has a total time complexity of* $O(2^{|V|}.|V|^2)$.

Démonstration. The Integer Programming solution for the MVC includes a nested loop that runs in exactly $\frac{1}{2}(|V|-1).|V|$. This nested loop is analogous to the constraint in (Eq. (4.3)) in Model 4.1. The Constraint Generation algorithm (Algorithm 1) has a constant running time of θ, where θ represents the required number of generated Vertex Covers. The branch-and-bound (BB) algorithm is used to solve the MVC. The complexity of BB is lower bounded by the total number of nodes, which is proportional to $2^{|V|}$ [140]. Hence, Phase I has a total running time of $\frac{1}{2}(|V|-1).|V| + \theta.2^{|V|}$ *.i.e.* $O(2^{|V|}.|V|^2)$. □

As stated before, it is noticed from the preliminary experimentation that it is sufficient to set θ between the range $[1.5|T|–3|T|]$. As MVC is NP-Hard, a lot of research has been conducted to produce efficient approximation algorithms (*cf.* in [141, 131]. The work in [4] develops

TABLE 4.5: Three-Phase Monitoring Optimization, Summary of Results

$\|V\|$	$\|E\|$	p	% Monitors	% Residual battery	Execution time (sec)			% Reduction in state transitions
					Phases			
					I	II	III	
50	123	0.125	60	86.0	0.83	3	0.001	66.6
50	104	0.09	52	89.8	0.46	2	0.001	67.0
100	234	0.09	55	88.8	2.02	2	0.001	70.0
100	347	0.125	66	86.6	9.71	600	0.001	80.8
150	380	0.09	62	87.5	48.90	591	0.001	70.2
150	407	0.125	57	87.5	21.50	523	0.001	52.9
200	383	0.06	54	98.0	22.98	534	0.001	80.0
200	576	0.08	63	87.3	30.06	300	0.001	0.00

a polynomial-time algorithm that converts the DODAG into a nice-tree decomposition with unity treewidth. Using the algorithm proposed in Chapter 5 [4] yields a significant reduction in the complexity of solving MVC on DODAGs. The problem becomes polynomial-time solvable, even though it is run iteratively using a Constraint Generation algorithm. It can be seen from Table 4.5 that the majority of computations is relatively centered in the multi-objective Generalized Assignment Problem (GAP) in Phase II. For example, DODAG number 4 with 100 nodes and 347 links requires around 10 seconds to reach the optimal solution for the MVC, and less than 1 second for the TSP Path, and 600 seconds to reach the optimal solution for the multi-objective GAP. The problem of large running time exists for instances having more than 200 nodes and 350 links. The optimal solution is still reachable, albeit slowly. Fortunately, there exist several approximation algorithms for GAP. For instance, [142] presented a polynomial-time 2-approximation algorithm for GAP.

Lemma 4.6.2. *Phase II has a total time complexity of* $O(2^{|M|.|T|}.|V| + |M| + |T|)$.

Démonstration. To prepare the coefficients for the communication objective function as represented by Eq. (4.15), the solution loops $|M|$ times. Moreover, a loop of $|T|$ times is required to enforce a period to be assigned once; which is represented in Eq. (4.17). Also, to constraint the energy used for monitoring of each node to be less than or equal to the $reservedBattery_k$ (Eq 20), the solution loops $|V|$ times. The number of variables in the GAP is $|M|.|T|$. Using the BB algorithm for solving the GAP, which is bounded by the total number of variables, gives an overall time complexity for Phase II of $O(2^{|M|}.|T|.(|V| + |M| + |T|)$. □

Lemma 4.6.3. *Phase III has time complexity of* $O(2^u.u^2)$.

Démonstration. Although the TSP Path is also NP-hard and there are approximation algorithms proposed in the literature (cf. in [22]), its running time in the proposed model is relatively small. This is due to the fact that the input of Phase III is only the unique Vertex Covers assigned to monitor the periods (u) from Phase II. TSP Path is solved using Held–Karp dynamic programming algorithm [143]. There are at most $2^u.u$ sub-problems, each of which takes linear time u to solve. Therefore, the time complexity of Phase III is $O(2^u.u^2)$. □

4.7 Summary

The problem addressed is the energy-efficient monitoring placement and scheduling of mission-critical IoT networks. Given its NP-hardness, we opted to tackle the problem via adopting a *Divide and Conquer* approach. The proposition decomposes the monitoring problem and maps it into three well-known sub-problems; for which approximation algorithms already exist in the literature. Thus, the computational complexity can be reduced. It is a proof of concept to emphasize that energy-efficient solutions for monitoring IoT networks do exist. Experimentation is performed using a number of test instances of different sizes, ranging from 50 to 200 nodes, and from 123 to 2463 links, prove that the proposed model is indeed effective and scalable in achieving the monitoring objective. The model succeeds in providing load balancing between monitors and minimizing the cost of monitoring in terms of energy and communication costs, and the number of nodes' state transitions.

However, the one major limitation of the proposed three-phase decomposition is that it is not an exact solution. Therefore, it does not guarantee global optimality. In fact, to the best of our knowledge, the exact solution to the defined problem is not yet known; which is proposed in the next chapter (Chapter 6). The global optimum will serve as a benchmark for comparisons and performance evaluation of future contemporary models.

Chapter **5**

Fixed Parameter Tractable Monitor Assignment Algorithm for IoT

5.1 Motivation & Objectives

One of the main objectives for the proposed monitoring mechanisms is determining where to place the monitors; such that the status of the entire set of links participating in the routing of mission-critical-related information is overseen. These monitors must have the possibility for passively (or actively) observing the regular traffic transmitted by the neighboring nodes (or verifying their availability; via injecting traffic and collecting their response) [64]. Therefore, the assignment of monitors must be optimized in order to ensure full monitoring coverage, as well as an efficient energy consumption of the constrained IoT devices; in terms of monitoring, communication, and monitoring state transitions.

Given the fact that IoT networks are very large scale; consisting of potentially (hundreds of) thousands of nodes, scalability will be key to handling this explosive growth. Scalability is defined as : *the ability to support an increasing number of connected devices, users, application features, and analytics capabilities, without any degradation in the quality of service* [144]. Consequently, scalable IoT monitoring mechanisms are essential to verifying the state of an increasing number of devices through a proportionate increase in the network resources. For minimum resource consumption, monitoring should be optimized.

To minimize the monitoring energy consumption, the number of monitors to be assigned should be minimized. The problem of assigning the minimum number of monitors to cover a given domain is known in the literature as the *minimum monitor assignment problem*; which has been proven to be NP-hard [76, 81]; implying that unless $P = NP$, efficient algorithms for solving it don't exist [82]. Given this fact, to achieve a scalable monitoring mechanism, we aim at proposing a new solution to solving the minimum monitor assignment problem with *minimal computational complexity*.

Moreover, remember that one of the objectives for our proposed methods is the interoperability with the standardized IoT protocol suite; most importantly with the IPv6 over Low-power Wireless Personal Area Networks (6LowPAN) and the Routing Protocol for Low-power and Lossy networks (RPL) (*cf.* Chapter 3 for an overview of IoT protocols). Therefore, we aim at analyzing the minimum monitor assignment problem while using the same routing topologies and message structures of these protocols.

To recap, the objectives for this part of the research are :

1. proposing a scalable monitoring mechanism via developing algorithms for solving the minimum monitor assignment problem with reduced computational complexity, and
2. achieving interoperability of the proposed mechanisms with 6LoWPAN and RPL protocols.

5.2 Fixed-Paramater Tractable Monitoring Mechanism

The network topology and communication between elements can be modeled by a graph. We leverage the logical routing topology constructed by RPL; the Destination Oriented Directed Acyclic Graph (DODAG). Fig. 5.1 depicts an example of a DODAG. Solving for the minimum number of nodes to monitor the whole network is analogous to solving for the minimum number of vertices in a graph which cover the entire set of edges. The latter is the definition of the classic Minimum Vertex Cover (MVC) problem [145]. Hence, we propose to map the monitor assignment to the MVC; which is an NP-hard, well-known, and well-studied graph optimization problem (*cf.* Fig. 5.2 depicts an illustration of the MVC). Formally, the MVC problem is defined as :

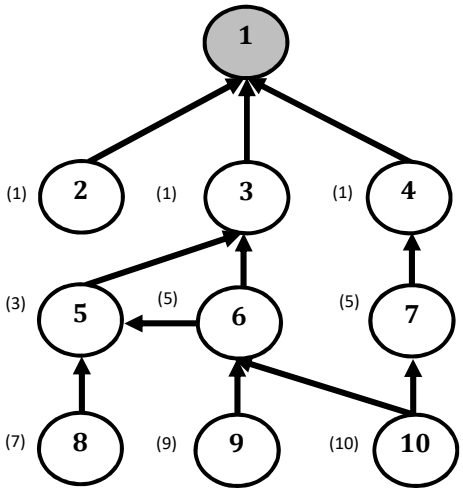


FIGURE 5.1: Logical routing topology (DODAG) built RPL, node 1 is the root, nodes' ranks are between brackets.

Definition 5.2.1. ***Minimum Vertex Cover (MVC) Problem***
The Minimum Vertex Cover (MVC) problem is defined over an undirected graph $G = (V, E)$ *and searches for a set of vertices* $S \subseteq V$*; such that for each edge* $e \in E$ *at least one of its endpoints belongs to S and* $|\ S\ |$ *is as small as possible.*

5.2.1 Tree Decomposition & the Minimum Vertex Cover Problem

There are several general approaches for attacking NP-hard problems, among them approximation algorithms [130, 131, 132], Fixed-Parameter algorithms [146, 147, 148], and heuristics [149]. The MVC problem is one of the best studied problems concerning Fixed-Parameter Tractability [148, 150]. Several techniques in parameterized complexity were successfully applied to the MVC problem, for instance data reduction, depth-bounded search trees, and dynamic programming [151].

Another approach for solving NP-hard graph optimization problems is the concept of *tree decomposition* for graphs; which was introduced by Robertson and Seymour [152], and plays an important role in algorithmic graph theory. Tree decomposition is the formal way to describe the *tree-likeness* of a certain graph (*cf.* Definition 5.2.2) [150]. The concept was motivated by the observation that many NP-complete problems are easy to solve on trees [153], connected graphs without cycles; due to the restriction in their structures when compared with general graphs.

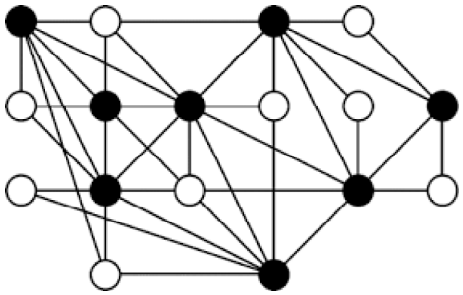


FIGURE 5.2: Solution of the Minimum Vertex Cover problem on a general graph. Black vertices are the members the optimal solution.

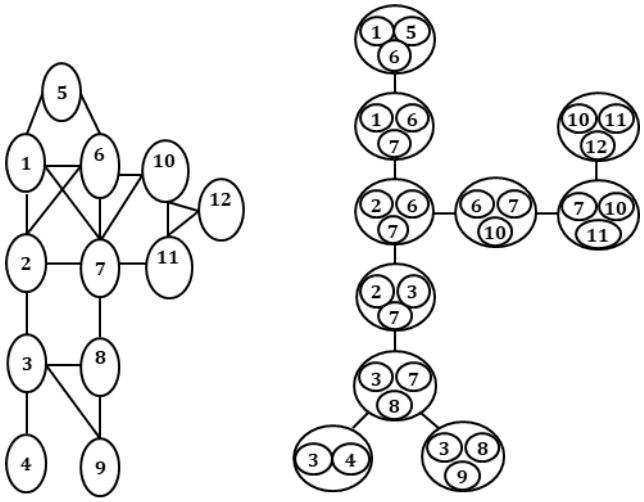


FIGURE 5.3: Example of nice-tree decomposition of a general graph.

Definition 5.2.2. ***Tree Decomposition*** *[153]*
Let $G = \{V, E\}$ be a graph. A tree decomposition of G is a pair $\{X_i : i \in I, T\}$, where each $X_i \subseteq V$ is called a ***bag****, and T is a tree with the elements of I as nodes. The following properties must hold :*

1. $\cup_{i \in I} X_i = V$,
2. *for every edge $e \in E, \exists$ a* ***bag*** *X_i with $i \in I, e \subseteq X_i$, and*
3. $\forall i, j, k \in I$, *if j lies on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.*

The third property is equivalent to the requirement that, for each $v \in V$, the nodes of all ***bags*** *containing v induce a* ***subtree*** *of T.*

Definition 5.2.3. ***Treewidth****[153]*
The width of a tree decomposition is the maximum of $|\ I(v)\ |$ - 1 $\forall v \in V_T$.

The authors in [152] addressed how hard problems can be solved for graphs that are *"like"* trees. In the following definitions, we give a brief description of the concepts of tree decomposition (Definition 5.2.2), treewidth (Definition 5.2.3), and nice-tree decomposition (Definition 5.2.4).

Definition 5.2.4. ***Nice-Tree***
A tree decomposition $\{X_i : i \in I, T\}$ is ***nice*** *if :*

1. *T is rooted at a designated node r ∈ I, called* ***root*** *node,*
2. *every vertex of T has at most two* ***children*** *(*i.e. *the tree is a* ***binary tree****),*
3. *if i is a* ***leaf*** *of T, | I(i) | = 1, I(i) is called a* ***start_bag****,*
4. *if i has two* ***children*** *j and k, I(i) = I(j) = I(k), in this case I(i) is called a* ***join_bag****, and*
5. *if i has one* ***child*** *j, then either :*
   — *| I(i) | = | I(j) | - 1 and I(i) ∪ I(j), in this case I(i) is a* ***forget_bag****, or*
   — *| I(i) | = | j | + 1 and I(j) ∪ I(i), in this case I(i) is an* ***introduce_bag****.*

Figure 5.3 shows an optimal tree decomposition T. T is optimal in the sense that there is no tree decomposition for the given graph G such that every *bag* X_i contains fewer than three vertices. Observe that the properties of tree decomposition as stated in Definition 5.2.2 hold.

Treewidth is a parameter which describes the tree-likeness of a graph (*cf.* Definition 5.2.3). The concept of treewidth has been proven to be very beneficial in algorithmic graph theory. Several problems that are NP-hard on general graphs are polynomial-time solvable, some even linear-time solvable, on graphs with bounded treewidth. These problems include Graph Coloring and the MVC [154]. If we restrict graph problems to graphs with a small tree decomposition, many NP-hard problems can be solved rather efficiently.

The fact that the running time of dynamic programming algorithms on tree decomposition is commonly of the complexity $O(c^k \cdot P(n))$; where $P(n)$ is a polynomial function of n, k is the treewidth of the tree decomposition, and n is the size of the instance, makes dynamic programming the prevalent approach for solving tree decomposition-based algorithms. Such problems are called *Fixed-Parameter Tractable (FPT)*; since they have algorithms that run in polynomial-time when the treewidth of the tree decomposition is fixed. Furthermore, algorithmic solutions are often easier when working with a *nice-tree* decomposition instead of a tree decomposition [155] (*cf.* Definition 5.2.4).

These facts are rather useful for our monitoring problem; since the monitoring mechanism can be executed in a reasonable amount of time on a nice-tree decomposition representing the network topology when the treewidth is relatively small (less than 20, for example) [155]. In light of this information, converting a DODAG into nice-tree decomposition with bounded treewidth yields the advantage of solving the generally NP-hard MVC problem; which in our model maps the minimum monitor assignment problem, on this special graph in polynomial time. Section 5.2.2 describes the modeling of and the proposed algorithms for the Fixed-Parameter Tractable minimum monitor assignment problem.

5.2.2 Modeling of the Minimum Monitor Assignment Problem

As mentioned above, for interoperability with standardized IoT protocols, we analyze the minimum monitor assignment problem while using the same routing structure constructed by RPL; the DODAG.

A DODAG $D = \{V, E\}$ consists of a set of vertices $v_d \in V$ and a set of edges E; where d is a unique identifier for each vertex in D. Each $v_d \in V$ is connected via an edge to one of three types of neighbors; depending on the rank of the neighbor in the DODAG (*cf.* Chapter 3 for how ranks are computed in RPL) :

1. *parents* $P(v_d)$ (preferred and alternative); which have lower rank than v_d in D,
2. *children* $C(v_d)$; which have lower rank than v_d in D, and
3. *siblings* $S(v_d)$; which have similar rank as v_d in D.

We model the minimum monitor assignment problem as a MVC problem. The objective of the proposed algorithms is to convert any DODAG, into *nice-tree* decomposition with

unity *treewidth*, while holding the properties in Definition 5.2.4. Bounding the *treewidth* to the value of one will have the effect of reducing the complexity of solving the normally NP-hard MVC problem on the DODAG to be linear-time solvable ; which is the main contribution of this part of the work, and a significant step towards a scalable monitoring mechanism.

TABLE 5.1: Glossary of Algorithmic Terms

Term	Description
V	Set of vertices, $V = \{v_d, d = 1, 2, ..., n\}$
E	Set of edges, $E = \{e_p, p = 1, 2, ..., m\}$
D	Destination Oriented Directed Acyclic Graph (DODAG) ; $D = \{V, E\}$
$C(v_d)$	Set of children of vertex v_d in the DODAG
$S(v_d)$	Set of siblings of vertex v_d in the DODAG
c_d	A child of vertex v_d, $c_d \in C(v_d)$
s_d	A sibling of vertex v_d, $s_d \in S(v_d)$
X_i	Bag containing a subset of V
I	Entire set of bags in the tree decomposition
T	A tree with the elements of I as nodes
NL	Set of non-leaf nodes in the DODAG
$required\_no\_of\_leaves$	Number of required leaves for v_d in the tree decomposition

Algorithm 2 PROCEDURE `INITIALIZE`

Input: $DODAGD \longleftarrow \{V, E\}$
begin procedure
2.1 $\forall v_d \in V$, Let $C(v_d)$ be the set of children of v_d , $S(v_d)$ be the set of siblings of v_d
2.2 Create binary tree $T \longleftarrow \phi$;
2.3 Create bag $X_1 \longleftarrow DODAG_{root}$;
2.4 Set root $T_r \longleftarrow X_1$;
2.5 Let non-leaf nodes $\longleftarrow \{NL : NL \longleftarrow \cup_{v_d \in D} \mid C(v_d) \mid + \mid S(v_d) \mid > 0\}$;
2.6 Sort NL according to the rank of v_d ;
2.7 **return** T
end procedure

The model is comprised of two auxiliary procedures, `INITIALIZE` (*cf.* Algorithm 2) and `CONSTRUCT_BINARY_TREE` (*cf.* Algorithm 3), and a main procedure `DODAG_INTO_NICE-TREE` (*cf.* Algorithm 4). TABLE 5.1 summarizes the variables of the three procedures.

Procedure `INITIALIZE` takes DODAG D as input and starts by creating a list of *children* $C(v_d)$ and another for *siblings* $S(v_d)$; for every vertex v_d in the $DODAG$ (step 2.1). An empty *binary tree*, t, is created ; its *root* t_r is set with a *bag* X_i that contains the DODAG *root* (steps 2.2 - 2.4). The *root* is considered as the destination of any node in D. The procedure then takes all *non-leaf* nodes in the DODAG, stores them in a vector NL, and then sorts them ascendingly according to their *ranks* in D (step 2.6). Finally, the procedure returns the initialized binary tree T.

The main logic of the model is implemented in the procedure `DODAG_INTO_NICE-TREE` (Algorithm 4) ; which takes the DODAG D and ensures that it is converted to a nice-tree decomposition with a unity $TREEWIDTH$. The **while** loop at step 4.2 iterates over the vector of non-leaf nodes NL, and searches the nice-tree T for the first vertex v_d ; the one which has

Algorithm 3 PROCEDURE `CONSTRUCT_BINARY_TREE`

Input: *bag* $X_i, required\_no\_of\_leaves$
Output: *binary nice-tree* t where $\{X_j : j \in I, t\}$ with the $required\_no\_of\_leaves$ and $X_j = X_i \quad \forall v_d \in V_t$
begin
3.1 $z \longleftarrow required\_no\_of\_leaves$;
3.2 Create two branches X_j, X_k where $X_j \longleftarrow X_k \longleftarrow X_i$;
3.3 **if** $z > 2$
3.4 $t \longleftarrow$ `CONSTRUCT_BINARY_TREE`$(X_i, z-1)$;
3.5 **end if**
3.6 **return** t;
end

the smallest *rank*. Next, it constructs a *binary tree* t for each of those vertices; by calling the recursive procedure `CONSTRUCT_BINARY_TREE` (Algorithm 3).

Taking the first vertex in NL at step 4.3, the procedure determines the $required\_no\_of\_leaves$ associated with it, which is the number of its children plus the number of siblings connected to it that are not already included in the tree decomposition (step 4.5).

With each call of the recursive subroutine `CONSTRUCT_BINARY_TREE`, it builds a *binary sub-tree* t with the required number of leaves and all its *bags* are set equal to X_i; which is sent as an argument to the procedure along with the $required\_no\_of\_leaves$ (steps 3.1 - 3.4). The returned *sub-tree* t is then augmented with the *nice-tree* T at the correct location; which is the location of the *bag* X_i at current iteration (step 3.9).

The inner **while** in the procedure `DODAG_INTO_NICE-TREE` loops for all the required leaves $required\_no\_of\_leaves$ (the vertex's *children* or *siblings* that are not already included in T). In step 4.13, the *leaf* $leaf_l$ is made into a $forget\_bag$ via branching out one *child* that contains two elements : the the current vertex v_d itself and one of its *children* or *siblings*, in the set of leaves L (step 4.13). Then, we make this *child* an $introduce\_bag$ via branching out a *child bag* that contains only one element; the vertex's *child* or *sibling*. Steps 4.13 and 4.14 are necessary to respect the rules in Definition 5.2.4 while ensuring a unity $TREEWIDH$.

Finally, vertex's v_d *child* or *sibling* is removed from the L list (step 4.15). After the algorithm exits this inner loop, the vertex v_d is removed from the vector NL at (step 4.18), and the outer loop continues until there are no more vertices left in NL. Fig. 5.4b shows a DODAG D and its corresponding *nice-tree* decomposition T with unity $TREEWIDTH$, the output of Algorithm 4.

5.3 Proofs of Termination

Lemma 5.3.1. *The conversion of a DODAG into nice-tree using the procedure* `DODAG_INTO_NICE-TREE` *(Algorithm 4) terminates.*

Démonstration. In order to prove that the procedure `DODAG_INTO_NICE-TREE` (Algorithm 4) terminates it is necessary to show that the two nested **while** loops starting at steps 4.2 and 4.12, respectively, eventually stop. The outer loop begins with a finite subset of vertices in the vector of *non-leaves* NL. The cardinality of NL is only altered at step 4.18; where it decreases by one. Since NL is finite and is strictly decreasing in cardinality with each iteration of the loop, the condition $(NL \neq \phi)$ is eventually falsified, and therefore the loop terminates.

Algorithm 4 PROCEDURE DODAG_INTO_NICE-TREE

Input: $D \longleftarrow \{V, E\}$
Output: $Nice-Tree\{X_i : i \in I, T\}; TREEWIDTH = 1$
begin
4.1 $T \longleftarrow$ INITIALIZE(D);
4.2 **while**($NL \neq \phi$) **do**
4.3 Select v_d from NL (its top vertex);
4.4 Search T for bag X_i : $X_i = v_d$;
4.5 $L \longleftarrow C(v_d) \cup S(v_d) : c_d, s_d \in NL$;
4.6 $required\_no\_of\_leaves \longleftarrow |\ L\ |$;
4.7 **if** $required\_no\_of\_leaves > 1$
4.8 $t \longleftarrow$ Construct_Binary_Tree($X_i, required\_no\_of\_leaves$);
4.9 At $X_i, T \longleftarrow T + t$;
4.10 **end if**
4.11 $l \longleftarrow 1$;
4.12 **while** $l \leq required\_no\_of\_leaves$ **do**
4.13 Make $leaf_l$ **forget_bag** via branching out X_k : $\{X_k = \{v_d, v_q\}, v_q \in L\}$;
4.14 Make X_k **introduce_bag** via branching out v_q;
4.15 $L \longleftarrow L - v_q$;
4.16 $l \longleftarrow l + 1$;
4.17 **end while**
4.18 $NL \longleftarrow NL - v_d$;
4.19 **end while**
end

The stopping condition of the inner **while** loop is when l is greater than the $required\_no\_of\_leaves$. Note that l initially has the value of one (step 4.11), and the value of the integer quantity $required\_no\_of\_leaves$ is the cardinality of the union of the finite subsets $C(v_d)$ and $S(v_d)$ (step 2.5). Therefore, the value of the expression ($required\_no\_of\_leaves$ - l) is an integer that strictly decreases with each iteration of the loop. Eventually, the value of the expression becomes a negative quantity, thus the condition, ($required\_no\_of\_leaves > 1$), is falsified, and the loop terminates. □

Lemma 5.3.2. *The construction of a binary tree using the procedure* CONSTRUCT_BINARY_TREE *(Algorithm 3) terminates.*

Démonstration. Since the procedure CONSTRUCT_BINARY_TREE (Algorithm 3) is a recursive function, to prove that it terminates, it is required to prove that its arguments get strictly smaller whenever there is a recursive call.

The argument z is initialized by the $required\_no\_of\_leaves$; which is a strictly positive finite integer that starts with a value greater than one. This restriction is ensured by the condition in step 4.7. With each call of the procedure, z strictly decreases until it reaches the value two; where the condition ($z > 2$) (step 3.3), is falsified and therefore, the recursive function terminates after returning the constructed *nice-tree*. □

5.4 Complexity Analysis

In every *nice-tree* decomposition produced by the procedure DODAG_INTO_NICE-TREE (Algorithm 4), there are two types of *bags* in the *nice-tree* namely :

— **single-element** *bags* containing one vertex of the DODAG, and

— **two-elements** *bags* containing two vertices that are connected via an edge in the DODAG.

The number of unique single-element *bags* is equal to the number of vertices in the DODAG. In addition, any two-elements *bags* will always be a unique *bags*; as the algorithm is designed to map an edge into a corresponding two-elements *bags* only once. Hence, we can conclude that the number of two-element *bags* is equal to the number of edges in the DODAG. Therefore, the number of unique *bags* in the *nice-tree* decomposition is simply the summation of the number of vertices and the number of edges in the DODAG.

Single-element *bags* are either *non-leaf* single-element *bags* or *Leaf* single-element *bags*. By experimental analysis, we identified the number of occurrences of any *non-leaf* single-element *bag* as $\{2 \cdot required\_no\_of\_leaves - 1\}$. Moreover, the number of occurrences of any *Leaf* single-element*bag* is determined by the number of alternative *parents* and *siblings* connected to its corresponding vertex, in the DODAG. Hence, we can conclude that the number of occurrences for both *Leaf* and *non-leaf* single-element *bags* is of the order of $O(v)$ at maximum; where v is the number of vertices. This leads to the conclusion that the total number of single-element *bags* is at maximum v^2.

The complexity of the proposed algorithm is measured by the number of *bags* generated for each DODAG to be a *nice-tree* decomposition. It is concluded that the proposed algorithm has a complexity of $O(m + v^2)$, where m is the number of edges in the DODAG; which is polynomial in time.

The running time of dynamic programming algorithms on tree decomposition FPT, and typically polynomial of the form $O(c(k) \cdot P(n))$; where k is the $TREEWIDTH$ and n is the size of the instance. Using the proposed procedures algorithm for the construction of the *nice-tree* decomposition from DODAGs, the parameter $TREEWIDTH$ is equal to one. Consequently, our particular MVC problem turns out to be just linear in time.

5.5 Discussion

The fundamental objective of this study is the enhanced robustness in IoT networks. In order to verify that mapping the minimum monitor assignment to the MVC problem the problem actually achieves the stated objective, it is required to prove that the DODAG maps the entire set of critical nodes and links corresponding to the critical mission, *i.e.* it comprises a vertex and an edge for every node and link, respectively, in the 6LoWPAN network.

The DODAG construction in RPL is based on the Neighbor Discovery Protocol (NDP) in 6LoWPAN; which provides the candidate neighbor set for each node, but the exact policies for selecting neighbors and parents is implementation-dependent and driven by the design of RPL's Objective Function (OF) [156]. The design of the OF is application-dependent. Therefore, it is the monitoring mechanism's responsibility to ensure the inclusion of the complete sets.

Let the candidate neighbor set be $N(v_j)$. In the procedure `INITIALIZE` (2; the sets of *parents*, *children* and *siblings* are defined for each vertex v_d as $P(v_d)$, $C(v_d)$ and $S(v_d)$, respectively. The selected RPL's objective function (OF0) [106] is designed to include all reachable neighbors, *i.e* the ones in the radio environment of the nodes. Consequently, the constructed DODAG maps the entire network, i.e. :

$$P(v_d) \cup C(v_d) \cup S(v_d) = N(v_d) \tag{5.1}$$

Therefore, solving the MVC problem on the general graph G that maps the network is equivilent to solving it on the DODAG D, *i.e.* :

$$MVC(D) \equiv MVC(G) \tag{5.2}$$

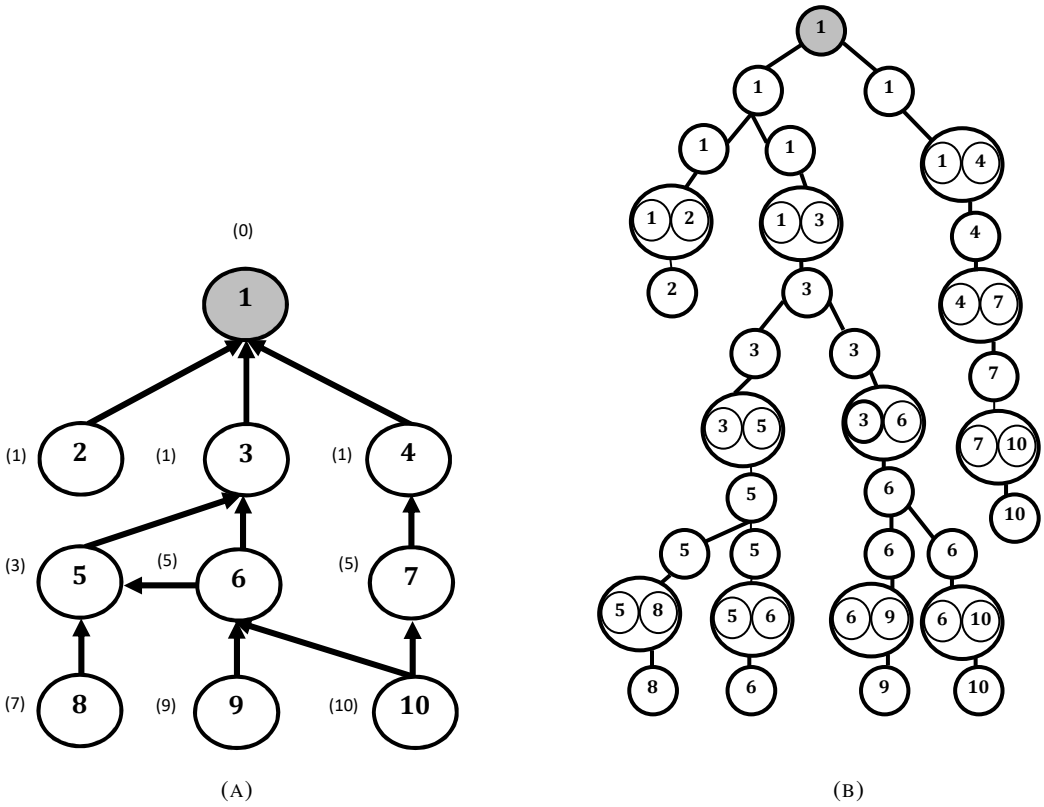


FIGURE 5.4: DODAG into nice-tree decomposition with unity tree-width. (A) DODAG before decomposition ; (B) DODAG after decomposition.

By definition of tree decomposition (*cf.* Definition 5.2.2), each vertex and edge in D is present in T, therefore,

$$MVC(T) \equiv MVC(D) \tag{5.3}$$

From 5.1, 5.2 and 5.3, we conclude that solving the MVC on the *nice-tree* achieves the principal objective of monitoring all the links of the network.

5.5.1 Summary

The principal objective is to monitor the unreliable links in a 6LowPAN-based IoT network ; via assigning monitoring nodes on the network to keep track of the status of the nodes and links in their vicinity. This is problem is known as the minimum assignment problem ; which is NP-hard. We tackle the problem by mapping it into the well-studied MVC problem, also NP-hard. However, it has been proven to FPT on graphs that are *"like"* tree ; tree decompositions. Targeting a scalable IoT monitoring mechanism, we proposed the algorithms 2, 3 and 4 to convert the RPL DODAG into *nice-tree* with its parameter $TREEWIDTH$ restricted to the value one ; thus achieving a linear-time solvable MVC problem ; which is the best case. Equations 5.1, 5.2, and 5.3 verify that the mapping of the monitor assignment problem into MVC problem and converting the DODAG into *nice-tree* decomposition actually achieve the stated principal objective.

Chapter **6**

Exact Passive Monitor Placement & Scheduling

6.1 Monitoring Objectives & Specifications

In the previous proposition towards the monitoring of mission-critical IoT networks (*cf.* Chapter 4), we tackled the problem using a *Divide and Conquer* approach ; via decomposing and mapping it into three well-known sub-problems. Solving each sub-problem separately is efficient, given that there exist several heuristics and approximation algorithms for each sub-problem.

However, the one major limitation of the proposed three-phase decomposition is that it is not an exact solution. Therefore, it does not guarantee global optimality. Here, we target the exact solution to the minimum monitor assignment problem and the optimal scheduling of the monitoring roles throughout a predetermined lifetime. The objective of the formulated mathematical model is minimizing the amount of energy consumed in monitoring the set of critical nodes, communication of the monitoring data to the central entity (6LoWPAN Border Router), and transition between monitoring states. To the best of our knowledge, the exact solution to the monitoring of mission-critical 6LoWPAN-based IoT networks is not yet known. The global optimum will serve as a benchmark for comparisons and performance evaluation of contemporary models.

For mission-critical IoT network services, the main problem is how network reliability, nodes availability, and robust connectivity can all be guaranteed. Performance metrics such as end-to-end delay and link quality level are not that much significant in this context as long as successful transmission of mission-critical-related information is always guaranteed. Therefore, the scope of the proposed monitoring mechanism is observing the network traffic to verify the availability of the critical set of nodes ; *i.e.* realize a passive monitoring mechanism. Node failures can be used to model failures of both physical nodes and links, with the latter represented as logical “nodes” connected to endpoints of the corresponding links [87].

In passive monitoring, the monitors will *listen* to the channel, and the monitoring energy is basically the cost energy consumed when the device is in receiving mode. The target is an optimized proactive, centralized, and passive monitoring mechanism for 6LoWPAN-based IoT networks which utilize the standardized RPL protocol for routing. The accurate specification of the model is required for a realistic estimation of energy consumption. In Section 6.4.1, the exact monitoring energy consumption of the model is computed. That said, our mechanism also supports the use of active monitoring ; where monitors participate in the network traffic ; via probing the monitored neighborhood and collecting the response. Active monitoring can be needed if the network traffic is sparse. However, it is not recommended for more efficient energy consumption of the resource-constrained things ; which is why we limit the experimentation to passive monitoring. The monitoring assumptions and objectives are somewhat similar to what have been stated earlier in Chapters 2 and 4.

In a nutshell, the monitoring mechanism is built based upon the following assumptions :

1. the IoT network is unstable, 6LoWPAN-based, uses RPL for routing, and performs a critical-mission,
2. in addition to the monitoring role, things' have primary functions of sensing, actuation and transmission,
3. things have stringent resource constraints with only a fraction of the battery reserved for monitoring,
4. a monitoring duty cycle mechanism is required, and the monitoring function is periodical across the planning horizon,
5. the active/sleep alternation is the turn on/off of the monitoring activity of the active node, and
6. links are lossy and can only be monitored by their extremities.

Given the stated assumptions, the monitoring objectives are listed as follows :

1. provide the *exact* solution to the optimal scheduling of the monitoring roles throughout a predetermined lifetime ; using minimal monitor sets in each period,
2. support the presence of sleeping nodes by duty-cycling the monitoring ; while *minimizing the monitoring state transitions*,
3. minimize the number of placed monitors while always respecting the monitoring *coverage requirement*, regardless of the lack of current network activity,
4. place the monitors such that in the *centralized* mechanism, the energy consumed in *relaying* the data to the 6BR is minimized,
5. *balance* the monitoring role among nodes,
6. support *passive* (and active) monitoring approaches (although passive monitoring is recommended),
7. ensure *interoperability* with 6LoWPAN and RPL protocols
8. support *scalable* monitoring via reducing the computational complexity, and
9. adapt *dynamically* to the changing topologies.

In the following sections, the exact mathematical model (Section 6.2.2) and the exact monitoring mechanism (Section 6.3) are described with regard to the above requirements and objectives.

6.2 Modeling & Mathematical Formulation

6.2.1 Decision Variables & Parameters

A graph can model the network topology. In tandem with RPL, the graph we use is the DODAG, $D = (V, E)$; where $V = \{ v_i, i = 1, 2, ..., n \}$ is the set of vertices representing the entire set of critical nodes, and E is the set of edges (objective 7). In a duty-cycled monitoring mechanism where a periodical functioning is assumed ; a planning horizon of several periods is defined and represented by $T = \{ T_j, j = 1, 2, ..., m \}$ (objective 2).

We formulate a linear Binary Integer Programming (BIP) model for the exact monitors' placement and scheduling across the planning horizon. The modeling terms are listed in Table 6.1. Binary Decision Variables (DV) are defined for the monitoring, relaying, and transition activities (objective 1). $xm_{i,j}$ represents whether a node v_i is assigned to monitor in a period T_j (Eq. (6.1)).

We opted to target passive monitoring ; which implies that to verify their neighbors' availability, monitors need only observe the regular traffic passing through the radio channel (objectve 6). Therefore, the monitoring cost comes in the form of the energy consumed while monitors *listen* to the messages transmitted by their neighbors. Depending on the total number of packets overheard from its neighbors set N, a monitor v_i consumes an amount of energy expressed as eM_i during a period T_j.

In the proposed centralized monitoring approach, monitors forward the monitoring data to their default parents in a path towards the DODAG root. A node routes monitoring packets if it is assigned to monitor, or if it is in the default route of an active-monitoring node in the same period. In the second case, the node is a *relay* node ; for which we define another binary DV $xr_{i,j}$ (objective 4). It indicates whether v_i acts as a relay in period T_j (Eq. (6.2)). eC_i stands for the amount of energy consumed by the relays while forwarding the monitoring data. The communication cost eC_i for each node v_i is a function of the number of times v_i forwards monitoring data through its default parent to the root.

For minimizing the cost of monitoring state transitions which are incurred as a result the required duty cycle, the transition DV $y^a_{i,j}$ and $y^s_{i,j}$ are introduced to the mathematical model (objective 2). $y^a_{i,j}$ identifies whether there is a monitoring state transition of v_i from sleep-monitoring in T_j into active in T_{j+1} (Eq. (6.3)) ; whereas $y^s_{i,j}$ denotes whether v_i is active-monitoring in T_j and asleep in T_{j+1} (Eq. (6.4)). The related costs are defined as $eActive$ and $eSleep$; they respectively denote the transition costs from sleep-monitoring to active and that from active-monitoring to asleep.

It is worth mentioning that the transition decision variables are dependent on $xm_{i,j}$ and $xm_{i,j+1}$; they are the product of the latter two binary variables. Equations (6.5) and (6.6) represent the dependence relationship. Consequently, with the addition of $y^a_{i,j}$ and $y^s_{i,j}$, the mathematical model becomes a quadratic Binary Integer Programming model ; which implies an expensive computational cost that goes against objective 8. However, we use standard techniques like the one mentioned in [157] to convert the quadratic formulation into a linear one (described in Section 6.2.2).

It is assumed that each node has a reserved battery ($reservedBattery_i$) for the monitoring activity across the entire planning horizon, apart from the energy dedicated to the main function of the thing (assumption 3). Thus, it is critical to ensure that all sorts of energy consumption of a thing never exceed such a threshold. This threshold is assigned by the network operator ; depending on the criticality of the monitoring role relative to the primary function. Section 6.2.2 describes the BIP model for the optimal monitor placement and scheduling problem.

$$xm_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } v_i \text{ is assigned to monitor in period } T_j \\ 0, & \text{otherwise} \end{array} \right\} \tag{6.1}$$

$$xr_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } v_i \text{ is a relay node in period } T_j \\ 0, & \text{otherwise} \end{array} \right\} \tag{6.2}$$

$$y^a_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } v_i \text{ is sleep-monitoring in period } T_j \\ & \text{and awake in period } T_{j+1} \\ 0, & \text{otherwise} \end{array} \right\} \tag{6.3}$$

$$y^s_{i,j} = \left\{ \begin{array}{ll} 1, & \text{if } v_i \text{ is active-monitoring in period } T_j \\ & \text{and asleep in period } T_{j+1} \\ 0, & \text{otherwise} \end{array} \right\} \tag{6.4}$$

TABLE 6.1: Glossary of Modeling Terms

Term	Description
V	Set of vertices, $V = \{v_i, i = 1, 2, ..., n\}$
T	Periodical planning horizon, $T = \{T_j, j = 1, 2, ..., m\}$
$xm_{i,j}$	Binary decision variable denotes whether v_i is assigned to monitor in T_j
$xr_{i,j}$	Binary decision variable denotes whether v_i acts as a relay in T_j
$y^a_{i,j}$	Binary decision variable denotes whether v_i changed its state from sleep in T_j into active-monitoring in T_{j+1}
$y^s_{i,j}$	Binary decision variable denotes whether v_i changed its state from active in T_j into sleep-monitoring in T_{j+1}
eM_i	Energy consumption of passive monitoring of neighbors
eC_i	Energy consumption of relaying messages to the preferred parent
$eActive$	Energy consumption of transitioning from sleep into active-monitoring state
$eSleep$	Energy consumption of transitioning from active into sleep-monitoring state
$reservedBattery_i$	Fraction of the available battery reserved for monitoring functions
$P(v_i)$	List of parents of v_i in the DODAG

6.2.2 Binary Integer Program

As stated before, the objective of the proposed model is to find the exact placement of monitors in the DODAG, as well as the optimal scheduling of multiple sets of monitors across the planning horizon; while minimizing the total energy consumed for monitoring, transmitting the monitoring data to the root, and the monitoring state transitions (objective 1). Remember that the overriding goals are to ensure full monitor coverage (objective 3), and a balanced (objective 5), energy-efficient duty-cycling of monitoring (objective 2) and relaying roles across the node set (objectives 4 & 6). Eq. (6.7) represents the model's objective function.

The constraint stated in Eq. (6.8) guarantees that the energy consumed by each monitoring node at the end of the planning horizon must never exceed the ($reservedBattery_i$) for monitoring.

Optimal monitor placement is finding out the *minimal* number of monitoring nodes placed on the graph while ensuring full monitor coverage. The coverage is forced in the optimal solution by including constraint (6.9); which states that each edge in the DODAG ((v_i, v_k) $\in E$) is incident to at least one active-monitoring node in the current period T_j.

To ensure the successful transmission of the monitoring data from the monitors to the DODAG root, there must be a connected path of active nodes from each monitor or relay node towards the root. Constraints (6.10) and (6.11) are introduced to the model for this purpose. They state that in each period T_j, if v_i is currently monitoring or relaying, then at least one member of its set of parents, denoted as $P(v_i)$, is also active in the same period.

Linear constraints are augmented to the mathematical model to eliminate the computational cost coming from the quadratic product of the binary variables $xm_{i,j}$ and $xm_{i,j+1}$. Constraints (6.12) and (6.13) ensure that $y^a_{i,j}$ takes the value of zero if either (1-$xm_{i,j}$) or $xm_{i,j+1}$ are zero, while (6.14) makes sure that $y^a_{i,j}$ takes the value of one if both binary variables are set to 1. Similar constraints, Eq. (6.15) - (6.17), are defined for $y^s_{i,j}$.

$$y^a_{i,j} = (1 - xm_{i,j})\, xm_{i,j+1} \tag{6.5}$$

$$y^{s}_{i,j} = xm_{i,j}\,(1 - xm_{i,j+1}) \tag{6.6}$$

$$\begin{aligned} \min \quad & \sum_{j\in T}\sum_{i\in V}\Big(eM_i\ xm_{i,j} + eC_i\ xr_{i,j}\Big) \\ & + \sum_{j=1}^{m-1}\sum_{i\in V}\Big(eActive\ y^{a}_{i,j} + eSleep\ y^{s}_{i,j}\Big) \end{aligned} \tag{6.7}$$

$$\begin{aligned} s.t. \quad & \sum_{j\in T}\Big(eM_i\ xm_{i,j} + eC_i\ xr_{i,j}\Big) \\ & + \sum_{j=1}^{m-1}\Big(eActive\ y^{a}_{i,j} + eSleep\ y^{s}_{i,j}\Big) \\ & \leq reservedBattery_i \qquad\qquad \forall i \in V \end{aligned} \tag{6.8}$$

$$xm_{v_i,j} + xm_{v_k,j} > 0 \quad \forall T_j \in T \text{ and } \forall \{v_i, v_k\} \in E \tag{6.9}$$

$$xm_{v_i,j} \leq \sum_{v_k \in P(v_i)} xm_{v_k,j} + xr_{v_k,j} \quad \forall v_i \in V \text{ and } \forall T_j \in T \tag{6.10}$$

$$xr_{v_i,j} \leq \sum_{v_k \in xr_{i,j}} xm_{v_k,j} + xr_{v_k,j} \quad \forall v_i \in V \text{ and } \forall T_j \in T \tag{6.11}$$

$$y^{a}_{i,j} \leq 1 - xm_{i,j} \tag{6.12}$$

$$y^{a}_{i,j} \leq xm_{i,j+1} \tag{6.13}$$

$$y^{a}_{i,j} \geq xm_{i,j+1} - xm_{i,j} \tag{6.14}$$

$$y^{s}_{i,j} \leq xm_{i,j} \tag{6.15}$$

$$y^{s}_{i,j} \leq 1 - xm_{i,j} \tag{6.16}$$

$$y^{s}_{i,j} \geq xm_{i,j} - xm_{i,j+1} \tag{6.17}$$

$$xm_{i,j}, xr_{i,j}, y^{a}_{i,j}, y^{s}_{i,j} \in \{0, 1\} \ \forall v_i \in V \text{ and } \forall T_j \in T \tag{6.18}$$

6.3 Exact Monitoring Mechanism

6.3.1 Centralized Passive Monitoring

Targeting full interoperability with 6LoWPAN-based IoT networks (objective 7), this section highlights the relations between the mathematical model and the implemented network monitoring mechanism. The BIP model is implemented using Julia ; a high-performance dynamic programming language for numerical computing [158] and solved using Gurobi solver [159] ; which is a powerful mathematical programming solver integrated into JuMP [160]. The latter is a modeling language for mathematical programming that extends Julia. Algorithm 5 describes the procedure `SOLVE_EXACT_MODEL` ; which takes as arguments the DODAG representing the topology of the mission-critical network, the formulated BIP (*cf.* Section 6.2.2), and the model's parameters (*cf.* Section 6.2.1). The BIP is solved by Gurobi and the optimal solution is returned to the calling procedure ; `SOLVE_EXACT_MODEL` (step 5.2). The solver's output is represented by the following matrices, the dimensions of each are the number of nodes × number of periods :

1. *optimal_monitors_solutions*,

2. *optimal_relays_solutions*,
3. *optimal_trans_to_sleep*, and
4. *optimal_trans_to_active*

SOLVE_EXACT_MODEL interprets the output of the solved mathematical model and translates it into the required *exact_monitoring_schedule* and *exact_relay_schedule* (steps 5.5 & 5.9); which are returned to EXACT_MONITORING (Algorithm 6) at step 5.13.

Algorithm 5 PROCEDURE SOLVE_EXACT_MODEL

Input: *BIP, DODAG, model_parameters*
Output: *exact_monitor_schedule, exact_relay_schedule*
begin
5.1 *exact_monitor_schedule* ⟵ *exact_relay_schedule* ⟵ ∅;
5.2 *optimal_monitors_solution, optimal_relays_solution* ⟵ GUROBI_SOLVER (*BIP, DODAG, model_parameters*);
5.3 **forEach** *period* ∈ *monitoring_timeline* **do**
5.4 **forEach** *node* ∈ *DODAG* **do**
5.5 **if** *optimal_monitors_solution*[*node, period*] == 1 **do**
5.6 *exact_monitor_schedule* ⟵ *exact_monitor_schedule* ∪ *node*;
5.7 **end if**
5.8 **if** *optimal_relays_solution*[*node, period*] == 1 **do**
5.9 *exact_relays_schedule* ⟵ *exact_relays_schedule* ∪ *node*;
5.10 **end if**
5.11 **end forEach**
5.12 **end forEach**
5.13 **return** *exact_monitor_schedule, exact_relay_schedule*;
end

Procedure EXACT_MONITORING (Algorithm 6) is the interface between the mathematical formulation and the monitoring mechanism. The procedure is functioning during the length of the monitoring timeline; represented by *timeline_timer*. Initially, the optimal duty cycle of monitors is obtained by calling the procedure SOLVE_EXACT_MODEL (Algorithm 5); which returns the *exact_monitor_schedule* (step 6.1). Next, following a passive monitoring approach, the neighbors' availability of each *monitor* is verified via calling the procedure the LISTEN_TO_NEIGHBORS (Algorithm 7); which returns the set of *unreachable_neighbors* detected during the period it is assigned to (step 6.4).

LISTEN_TO_NEIGHBORS (Algorithm 7) works as follows : during the period it is assigned to, given by *period_length*, the *monitor* checks whether each of its *neighbor*s had participated in the radio transmission (step 7.4). Note that the set of *neighbor*s (parents, children, and siblings) is known from the DODAG; which is passed as a parameter to the calling procedure. If there is a *neighbor* from which the *monitor* has not received any messages throughout the entire period, its address is appended to the list of *unreachable_neighbors* (step 7.5). After the expiry of the *period_timer*, the list of *unreachable_neighbors* is returned to the procedure EXACT_MONITORING (Algorithm 6).

In the centralized monitoring approach, the list of *unreachable_neighbors* of each *monitor* is forwarded to its default parent (step 6.6); which is responsible for relaying that message to its own default parent, and so forth until it reaches a central entity, the 6BR or the DODAG root. Having unconstrained resources and a global view of the network state, powerful analysis of the monitoring data and further corrective measures can be taken. On the other hand, leveraging RPL's repair mechanism, an attempt at a fast recovery of the DODAG is done by the *monitor* via calling LOCAL_REPAIR (step 6.7) [26] (objectives 4 & 7).

It is noteworthy that the *period_length* should be carefully chosen such that it is neither too short nor too long. Too short a *period_length* may result in false alarms. A false positive alarm occurs when a monitoring mechanism reports as fault a state that is in fact legitimate network activity [161]; whereas failure to detect a faulty state is termed as a false negative alarm. On the other hand, too long a *period_length* will unnecessarily exhaust the energy of monitors as they are awake-monitoring for quite a long time. Fortunately, there are some studies which focus on the optimal period length, such as the one in [162].

Algorithm 6 PROCEDURE `EXACT_MONITORING`

Input: $BIP, DODAG, model\_parameters$
Output: $unreachable\_neighbors \quad \forall monitors \in exact\_monitoring\_schedule$

```
begin
6.1   exact_monitoring_schedule ⟵
        SOLVE_EXACT_MODEL(BIP, DODAG, model_parameters);
6.2   while timer < timeline_timer do
6.3     forEach monitor ∈ monitoring_schedule do
6.4       unreachable_neighbors ⟵
            LISTEN_TO_NEIGHBORS(DODAG, period_length);
6.5       forEach neighbor ∈ unreachable_neighbors do
6.6         FORWARD_TO_PARENT(unreachable_neighbors);
6.7         LOCAL_REPAIR(DODAG);
6.8       end forEach
6.9     end forEach
6.10  end while
end
```

Algorithm 7 PROCEDURE `LISTEN_TO_NEIGHBORS`

Input: $DODAG, period\_length$
Output: $unreachable\_neighbors$

```
begin
7.1   unreachable_neighbors ⟵ ∅;
7.2   while timer < period_timer do
7.3     forEach neighbor ∈ DODAG do
7.4       if ! RECEIVE_MESSAGE(neighbor) do
7.5         unreachable_neighbors ⟵ unreachable_neighbors ∪ neighbor;
7.6       end if
7.7     end forEach
7.8   end while
7.9   return unreachable_neighbors;
end
```

6.3.2 Separate DODAG for Routing

Even though the centralized approach allows for sophisticated monitoring tasks; which might otherwise exhaust the resources of the things; they are achieved at the expense of high communication overhead. To take the burden of routing the monitoring data to the DODAG root off the constrained nodes, we leverage the multiple instance feature of RPL (*cf.* Fig. 6.1) and create another DODAG for that purpose. It is considered as an overlay structure within which the monitors communicate separately. The separate DODAG may consist of

only monitoring nodes, and the monitoring data is transmitted from monitors to the 6BR through the shortest path. Those routes may not have been defined by application-specific routing. However, according to the network topology, the separate instance of monitors might

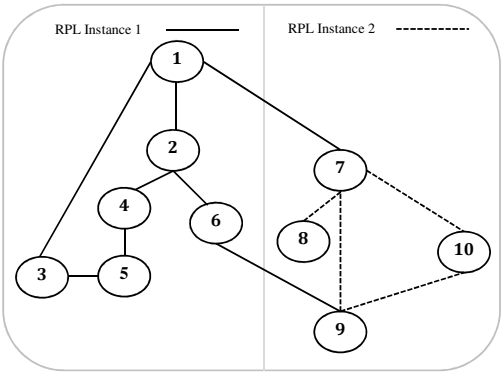


FIGURE 6.1: RPL's multiple instance feature.

not guarantee the presence of a connected path between themselves and the DODAG root. Therefore, for the successful transmission of monitoring data, *relay* nodes may be required. The optimal placement of the required relays is obtained from the $exact\_relay\_schedule$ given by procedure `SOLVE_EXACT_MODEL` (Algorithm 5). Even though it is probable that relay nodes are not currently involved in monitoring, they still share the monitoring burden as there is an amount of energy consumed by forwarding the monitoring data. This is cost is represented in the model's objective function by eC (*cf.* Section 6.2.2).

6.3.3 Active Monitoring

For detailed performance-related information of the nodes, the monitoring mechanism can exploit RPL's DIO control messages; defined in the RPL specification [26] (*cf. Chapter 3* for a review of RPL). DIOs are used for advertising topology-related information, typical examples are the Objective Function, the current rank of a node, the current RPL Instance ID, and the IPv6 address of the root. An optional header for metrics and constraints objects, known as DAG Metric Container (DAGMC), 6.2 can be contained inside DIOs) [50].

DAGMCs carry objects representing nodes' characteristics such as Node State and Attribute (NSA) and Node Energy (NE). Consequently, information such as CPU overload, available node memory, energy and power mode, and estimated remaining power level, as well as the DODAG configuration and ranks of the nodes located in the neighborhood, can all be accessed by sniffing the DIO control messages. Moreover, by looking at the source/destination addresses, nodes can have a local view of the topology and traffic patterns in the eavesdropped area. Appending DAGMCs, configuring the monitors to listen to DIOs, and analyzing the gathered data is a rather complicated and expensive procedure. It should be only implemented if it is necessary. Otherwise, a passive monitoring mechanism that tests neighbors' reachability and therefore guarantees service availability is recommended.

6.4 Experimental Evaluation

6.4.1 Parameter Settings

To verify and validate the BIP model, extended computational experiments are conducted using different network sizes and topologies. The parameters are carefully chosen to reflect a

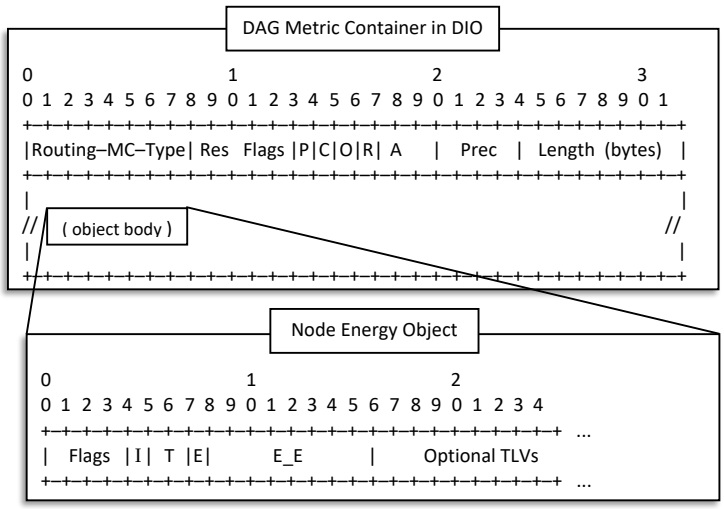


FIGURE 6.2: RPL's DAG metric container [50].

real-life mission-critical IoT network. This section explains how the energy costs are estimated. The parameters' settings are displayed in Table 6.2.

The amount of energy consumption is platform-dependent. TelosB, also known as the TMote Sky motes, are selected as a target platform for regular and monitoring nodes; since its computational resources allow it to function as an RPL router node within the Contiki RPL implementation [104]. Tmote Sky motes use 2 AA batteries with total available energy of 30780 Joules. Referring to Tmote Sky's data sheet [163], its measurements in different modes of operation are shown in Table 6.3.

While monitors passively listen to their neighbors, they consume a significant amount of energy; which can be expressed as e_{listen}. The total monitoring cost, eM_i, is a function of the listening cost, the cardinality of a monitor's set of a neighbors $|N|$, and the number and size of the sniffed packets. Based on the framework of 6LoWPAN [53], the size of the data packet, hereby denoted s_{msg}, is 17 bytes for the header frame in addition to the size of the data payload. Assuming that the payload is 2 bytes, s_{msg} is 19 bytes. Acknowledgment packets are not mandatory in IEEE 802.15.4. Nevertheless, since the target is a more reliable communication, acknowledgment costs are considered in our computations. 6LoWPAN dedicated 11 bytes to the acknowledgment packet; which is expressed here as s_{ack}.

To compute the communication cost, eC_i for IEEE 802.15.4 networks that employ Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), there is a fixed energy cost associated with the Clear Channel Assessment; symbolized as e_{CCA}. Moreover, it is necessary to estimate the costs related to the radio energy consumption of receiving and relaying messages; expressed as $e_{receive}$ and e_{send}, respectively.

Given Tmote Sky measurements in Table 6.2, we compute the costs of receiving eM_i and relaying one message of size 19 bytes eC_i using Equations 6.19 and 6.20, respectively. Accordingly, the default values of the BIP model are obtained and given in Table 6.4.

$$eM = e_{listen} + e_{receive} \times s_{msg} \tag{6.19}$$

$$eC = e_{CCA} + e_{send} \times s_{msg} + e_{receive} \times s_{ack} \tag{6.20}$$

TABLE 6.2: Default Values of Physical Network Parameters

Parameter	Description	Default Value
e_{listen}	Energy cost of listening to the radio channel	0.58 mJ
e_{CCA}	Energy cost of Clear Channel Assessment	0.08 mJ
e_{send}	Radio energy consumption of sending 1 byte	0.0020 mJ
$e_{receive}$	Radio energy consumption of receiving 1 byte	0.0022 mJ
s_{msg}	Size of the data packet	19 B
s_{ack}	Size of acknowledgment packet	11 B

TABLE 6.3: Tmote Sky Measurements in Typical Operating Conditions [163]

Parameter	Default Value
Maximum supply voltage	3.6 V
Radio transmitting current consumption	17.4 mA
Radio receiving current consumption	19.7 mA
Radio on current consumption	365 μA
Power down current consumption	1 μA
Maximum startup time	860 μs
Maximum supply energy	30780 J
Transmit bit rate	250 $kbps$

6.4.2 Results & Discussion

Experiments are performed on a personal computer with 16 Gigabytes of RAM and 2.20 Gigahertz Intel Core i7 processor. The proposed model is tested using 21 instances of 8 network topologies with variable densities. Results are displayed in Table 6.5; where the values of the following metrics are recorded.

— Number of nodes (|**V**|), links (|**E**|), and graph's density (ρ).
— Percentage of monitors (**monitor(%)**).
— Average number of neighbors per monitor (|**N**|).
— Average energy consumption per node in monitoring, relaying, and state transitions (**Energy cons.** (mJ)).
— Percentage of battery consumed for monitoring, relaying and state transitioning from $reservedBattery$ per node (**Battery cosns.** (%)).
— Average number of overheard packets per monitor ($msgs$).
— Model execution time in seconds ($time$).

The experiments are designed to test the model's ability to optimally place monitors and relays on the DODAGs that represent the nodes are associated with the critical mission. Considering an example of the KARATE [6] network of 34 nodes and 114 links. Fig. 6.3 pinpoints the placement of three types of monitors; specifically, the ones that : (1) transitioned from active-monitoring in the previous period T_{j-1} to sleep in T_j, (2) transitioned from sleep-monitoring to active in said periods, and (3) the ones that did not transition at all, *i.e.*, they are monitoring in both periods T_{j-1} and T_j. The rest of the vertices correspond to the monitored nodes.

We tested the model on the Dolphins [7] network topology; consisting of 62 nodes where node number 1 is the 6BR. This topology demonstrates the benefit of using a separate DODAG for routing the monitoring data to the 6BR. Creating a separate instance consisting of

TABLE 6.4: Default Values of Model Parameters

Parameter	Default Value
eM_i	0.621 mJ
eC_i	0.486 mJ
$eActive$	0.0011 mJ
$eSleep$	0.02 μJ
$reservedBattery_i$	50 mJ
T	20

monitors only does not provide a connected path between the monitors and the 6BR (except for monitors 14 and 61). Fig. 6.5 displays the required relay nodes for an arbitrary period T_j ; corresponding to the values of $xr_{i,j}$ in the solution optimal solution of the proposed mathematical model.

To test the model's response towards variations in the Right Hand Side (RHS) vector of the $reservedBattery_i$ for monitoring constraint (Section 6.2.2 - (6.8)). For the KARATE [6] topology depicted in Fig. 6.3, we run 14 trials while varying the node's $reservedBattery_i$ in the range between 1539 - 307800 mJ ; which corresponds to 0.05% - 10% of Tmote Sky total power. This particular range is selected as the nodes are incapable of monitoring the network for the length of the planning horizon with a $reservedBattery_i$ below than its lower bound and the model converges to the same solution with the upper bound and beyond. The results of these trials are displayed in figures 6.6a, 6.6b, and 6.7.

Fig. 6.6a shows a clear trend ; the more the $reservedBattery_i$ constraint is hardened ; the more is the execution time. Fig. 6.6b, on the other hand, shows the significant effect of the size of the $reservedBattery_i$ on the total energy consumption. The interpretation is that the harder the battery constraint, the more is the number of monitoring state transitions since a node does not have enough power to continue monitoring for several periods.

The most remarkable conclusion to emerge from the experimental results is that hardening the $reservedBattery_i$ constraint has the advantage of balancing the monitoring role among nodes. Consequently, the average energy consumption per node decreases. This result is depicted in Fig. 6.7 ; which shows the average energy consumption per node after 20 periods as the $reservedBattery_i$ is varied between 307800 mJ, 2052 mJ, and 1641.6 mJ ; corresponding to 10 %, 0.07% and 0.05% of the total power of Tmote Sky, respectively. This finding highlights the fact the size of $reservedBattery_i$ should be set with considerable care.

Fig. 6.7 emphasizes the efficiency of the proposed passive monitoring mechanism ; since the average energy consumption per node does not exceed 300 mJ (approximately 0.01% of its total power) ; regardless of the size of $reservedBattery_i$ for monitoring.

6.5 Summary

This part of the work presents the exact solution to the minimum monitor assignment problem with a duty-cycled monitoring approach. The optimal schedule guarantees monitoring coverage with minimum energy consumption. The solution is incorporated into a centralized, passive monitoring mechanism that is interoperable with RPL and 6LoWPAN protocols.

The overall findings confirm the effectiveness of the proposed model in achieving full monitor coverage with minimum energy consumption in all tested network topologies (objective 3). The results in Table 6.5 also confirm that the execution times are tolerable even for relatively large or dense networks. These conclusions are crucial for the adoption of network monitoring into critical-mission IoT networks.

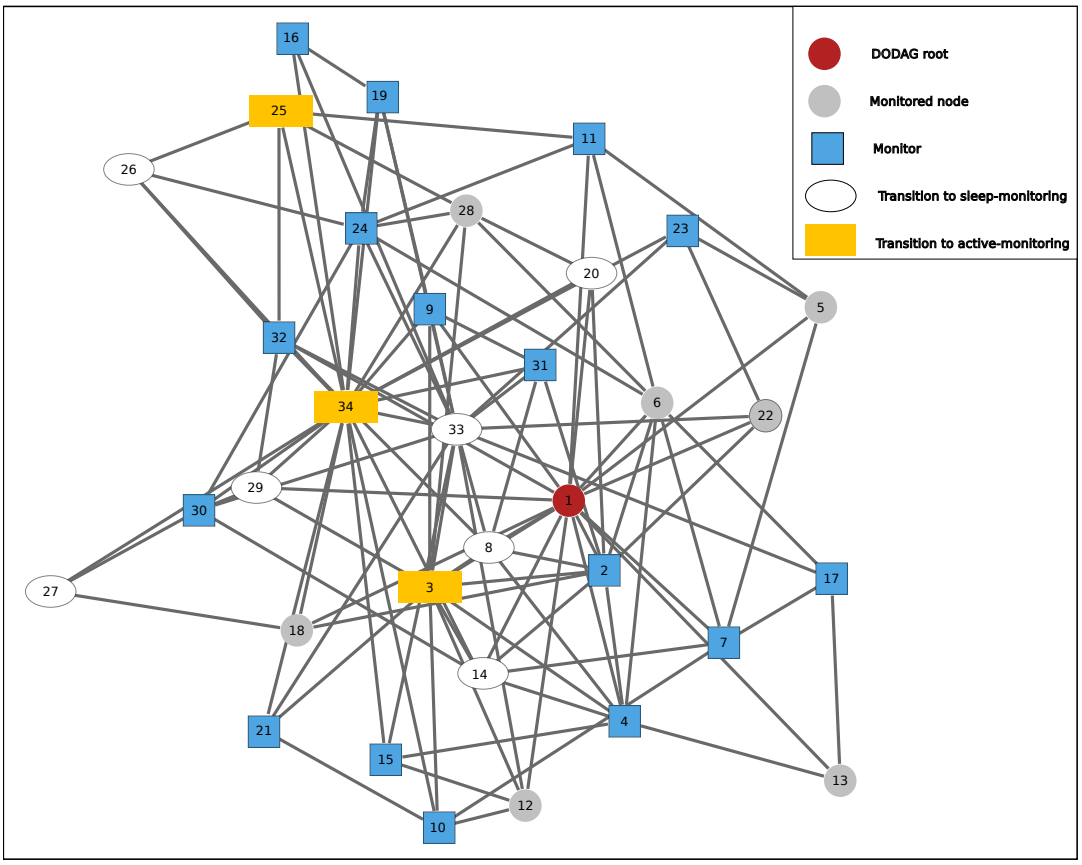
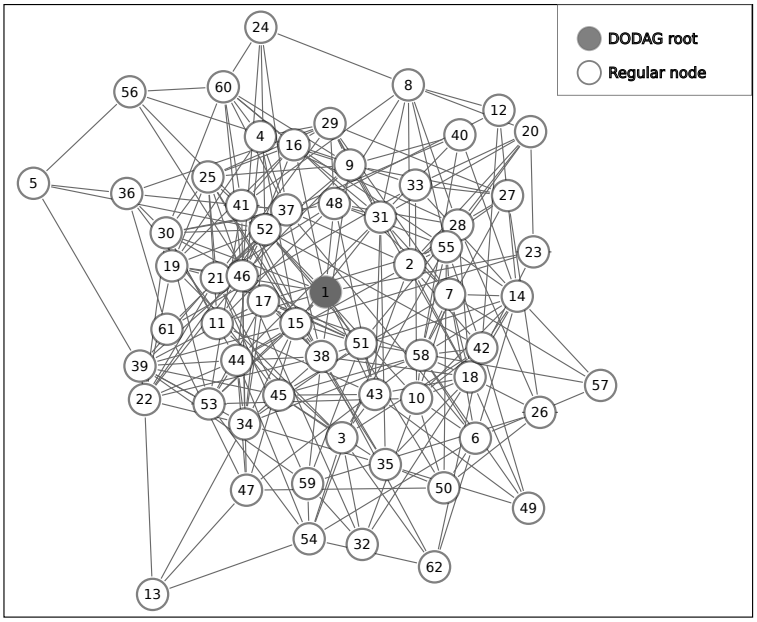


FIGURE 6.3: Optimal solution in a given period for the KARATE [6] network topology with 34 Vertices and 114 edges. $reservedBattery_i$ = 1641.6 mJ (0.05 % of total power).

Even though the optimization is computed off-line, for benchmarking, it is essential to test how scalable the model is, concerning the size and density of networks. It can be seen in Table 6.5 and Fig. 6.8, the execution time for a dense network of 600 nodes and 17970 links is approximately 8.5 minutes, which is not very expensive. However, it is a fact that computing the exact solution to the BIP model for large-sized networks will be computationally expensive, a result of the NP-hardness of the MVC problem, which is represented in constraint 6.9 [164]. This fact implies an exponential lower bound on the running time of solving the proposed linear BIP model. Despite computational limitations, the proposed mechanism serves as a benchmark for comparisons and performance evaluation of contemporary models, which has been missing from the literature.

So far, the proposed models represent a more static view of the IoT network; rendering the optimal solution unrepresentative to the current situation in case a significant change in the network topology is detected. To avoid a complete re-optimization of problem, dynamic models are required; which is proposed in the next chapter.

Nevertheless, it is worth mentioning that the model's performance was tested under a hard battery constraint; with a $reservedBattery_i$ of 50 mJ from a total available power of 30780 $Joules$ of the TMote Sky mote.



(A)

FIGURE 6.4: The DOLPHINS [7] 62 Vertices and 159 edge.

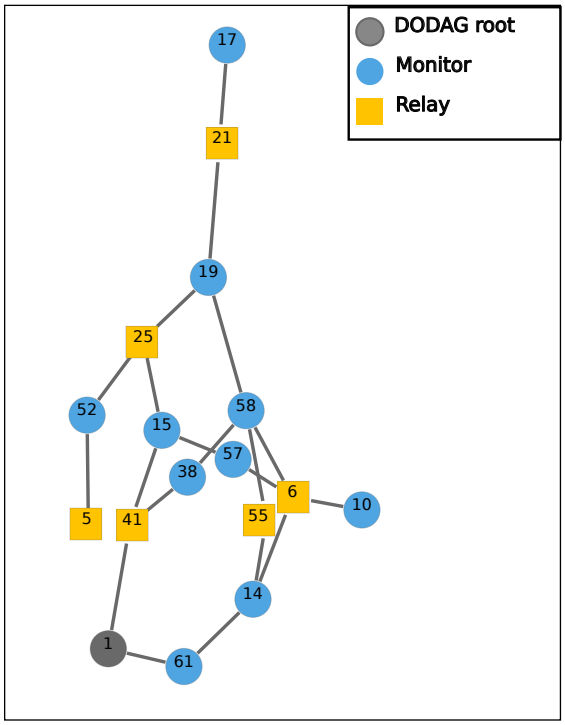
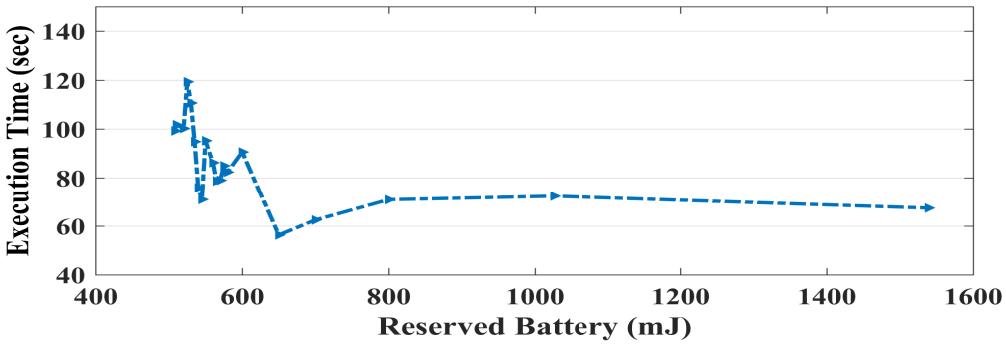
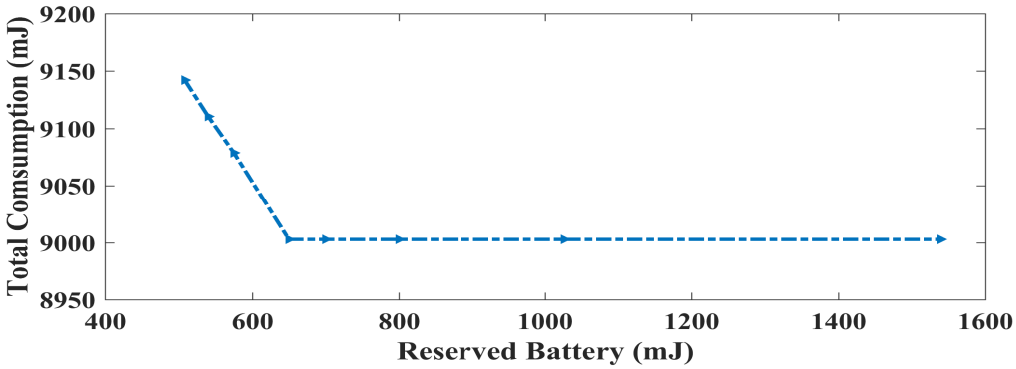


FIGURE 6.5: Monitoring DODAG with the necessary relays for the DOLPHINS [7] topology of 62 Vertices and 159 edges. Node labeled 1 is the root.



(A)



(B)

FIGURE 6.6: Effect of variation of $reservedBattery_i$ for the KA-RATE [6] topology; $T = 20$; (a) $reservedBattery_i$ versus model execution time; (b) $reservedBattery_i$ versus network total energy consumption.

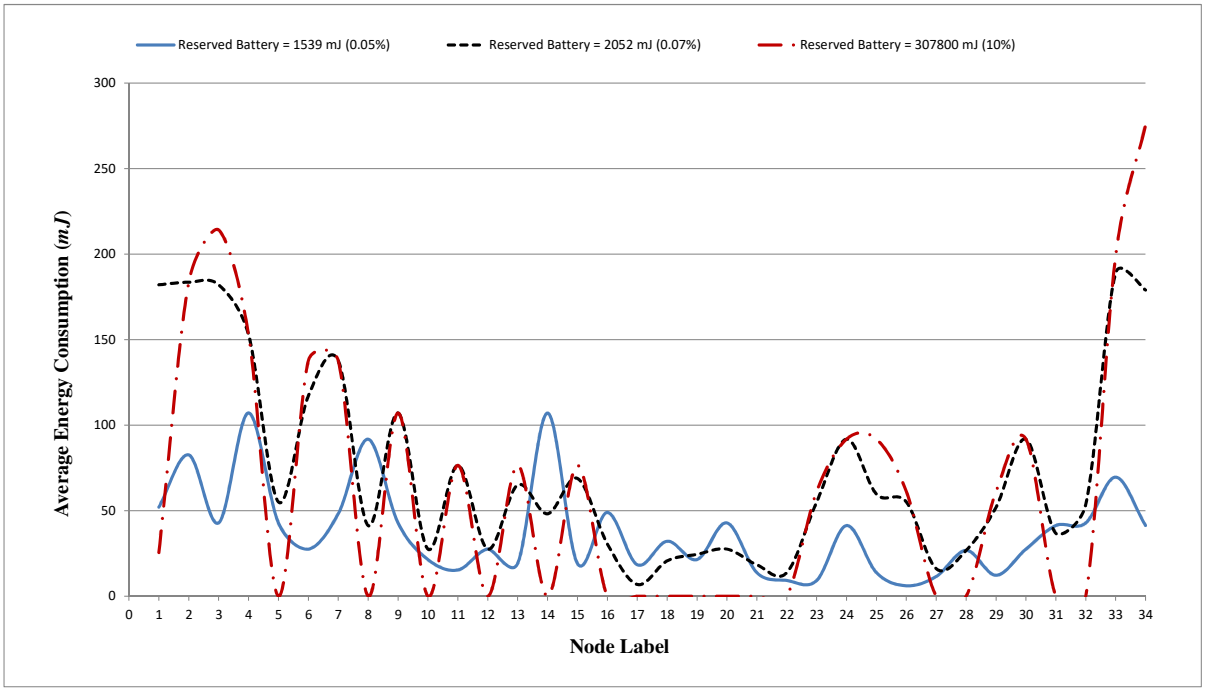


FIGURE 6.7: Average energy consumption per node for the KARATE [6] topology; $reservedBattery_i$ = 307800 mJ, 2052 mJ and 1641.6 mJ; corresponding to 10 %, 0.07% and 0.05% of Tmote Sky total power, respectively; $T = 20$.

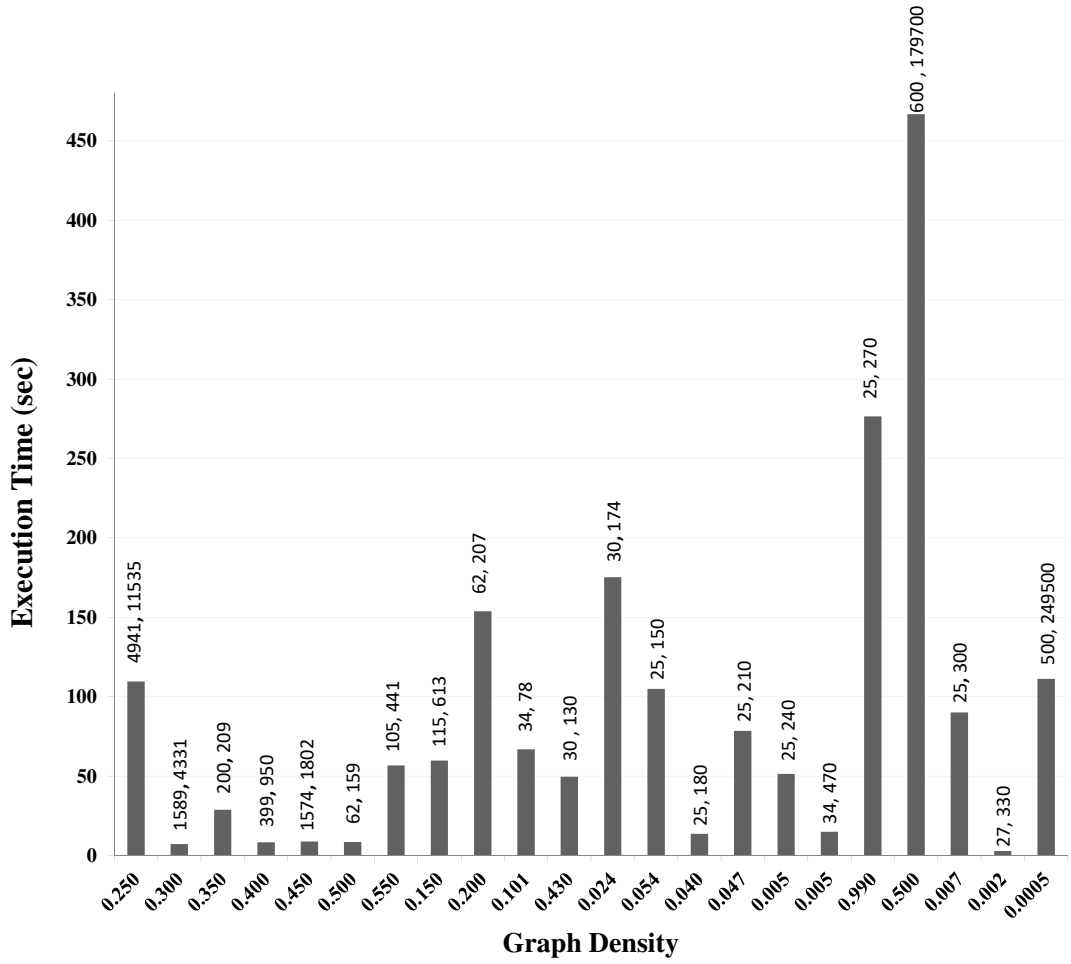


FIGURE 6.8: Effect of varying network density on the execution time.
Data labels are the number of nodes and links.

TABLE 6.5: Experimental Results of Exact Monitor Placement & Scheduling

Instance	Topology	|V|	|E|	ρ	monitor(%)	|N|	Energy cons.(mJ)	battery cons.(%)	msgs	time (sec)
1	Random	25	150	0.250	80	13	9.936	19.872	5	109.66
2	Random	25	180	0.300	76	16	9.439	18.878	5	7.20
3	Random	25	210	0.350	84	18	10.432	20.865	6	28.82
4	Random	25	240	0.400	84	21	10.432	20.865	5	8.30
5	Random	25	270	0.450	88	23	10.929	21.859	6	8.75
6	Random	25	300	0.500	96	25	11.923	23.846	5	8.46
7	Random	27	330	0.550	88	26	11.040	22.080	5	56.76
8	Random	30	130	0.150	66	11	8.280	16.560	6	59.73
9	Random	30	174	0.200	77	13	9.522	19.044	5	153.82
10	KARATE [6]	34	78	0.101	41	5	5.607	11.214	5	66.90
11	Random	34	470	0.430	87	29	10.914	21.829	5	49.57
12	DOLPHINS [7]	62	159	0.024	56	5	7.013	14.026	5	175.30
13	DOLPHINS* [7]	62	207	0.054	58	9	7.211	14.850	7	104.95
14	POLBOOKS [8]	105	441	0.040	60	8	7.570	15.430	6	13.61
15	FOOTBALL [9]	115	613	0.047	82	11	10.152	20.304	5	78.55
16	POWER [10]	200	209	0.005	44	2	5.437	10.875	5	51.30
17	POLBOOKS* [8]	399	950	0.005	29	6	3.673	7.346	6	15.00
18	Complete Graph	500	249500	0.990	99	499	12.395	24.790	6	276.60
19	Random	600	179700	0.500	99	599	12.3993	24.798	5	466.90
20	NETSCIENCE [11]	1589	4331	0.002	57	6	45.005	90.011	5	2.82
21	POWER* [10]	4941	11535	0.0005	47	5	5.799	11.598	5	111.24

Chapter 7

Dynamic Distributed Monitoring for 6LoWPAN-based IoT Networks

7.1 Motivation & Objectives

Given the constantly changing, dynamic nature of the IoT networks, to provide robust network monitoring it is necessary that the monitoring mechanism adapts itself to the real-time network instabilities. The previously presented models, either the three-phase decomposition or the exact solution to the monitoring placement and scheduling problem, represent a rather static view of the IoT network, rendering the optimal (or approximate) solution unrepresentative of the current network state. Consequently, in the case where significant changes in nodes' state or network topology are detected, for maintaining robust coverage of the IoT network, the entire problem has to be re-solved, which is a waste of time and resources. To avoid a complete re-optimization of problem, dynamic models are required. The objective of the proposed dynamic monitoring mechanism is ensuring real-time monitoring coverage while respecting the limited *and* changing power resources of the devices in order to prolong the network lifetime.

On account of the fact that the 6LoWPAN Border Router (6BR) is the central entity that is always assumed to be accessible. The most common IoT architectures are completely centralized. Even though generally centralized monitoring architectures allow for simpler network management, they also impose some limitations [38] :

1. they represent a single point of failure,
2. they limit the possibility of creating ad-hoc domains without dedicated infrastructures, and
3. they represent a more static world view, where device roles are fixed, rather than a dynamic world view that recognizes that networks and devices, and their roles, may change over time.

The proposed exact solution to monitoring placement and scheduling (*cf.* Chapter 6) guarantees energy-efficient, global optimal solution, which can act as a benchmark for comparisons and performance evaluation of future contemporary models. However, there are limitations to the exact model which we target to remedy in the dynamic model discussed here, namely :

1. it is assumed that a global and a predefined sleep-wake monitoring schedule is centrally available, thus requiring network-wide information distribution to exchange the monitoring schedule and the monitoring data from and to the central node, respectively,
2. in a large network the information distribution is considered as a significant communication overhead on the constrained devices, and
3. there are scalability issues due to the computational complexity of the NP-hard BIP.

In response to the stated limitations, in this part of the work we propose a *dynamic distributed* monitoring placement and scheduling mechanism for 6LoWPAN-based IoT networks.

7.2 Background

The Controlled Greedy Sleep (CGS) algorithm proposed in [118] targets Wireless Sensor Networks (WSN) that are used to monitor an area in space. Leveraging the high redundancy feature usually present in sensor networks, the objective is ensuring that a required number of sensors are able to provide measurements from each point in space during a periodical network life-time. This class of problems can be treated as the k-coverage problem, where every point of the target area must be covered by at least k sensors (k is determined by the application). The life-time of the WSN is prolonged by applying a scheduling (duty-cycling) mechanism, meanwhile the k-coverage is provided by the active sensors. The CGS provides a quasi-optimal sensor coverage solution, while respecting the deployment and energy constraints of sensor nodes.

In the CGS algorithm, the sensing assignment in a sensor network is represented by a bipartite graph $G(S \cup R, E)$, where the two disjoints sets of vertices represent the nodes S and geographical regions R (*cf.* Fig. 7.1). In G there is an edge e between sensor $s \in S$ and region $r \in R$ if and only if s covers region r.

The algorithm applies the notion of a *drowsiness factor*, which models the state of the sensors and their "desire" to go to sleep. The factor is computed at the beginning of each period for each node. Supposing that a sensor node s has E_s remaining energy, its drowsiness factor D_s is defined as follows :

$$D_s = \left\{ \begin{array}{cc} \frac{1}{E_s^{\alpha}} \sum_{r \in R} \Phi_r & \text{if } \Phi_r > 0, \forall r \\ \text{-1} & \text{otherwise.} \end{array} \right\} \tag{7.1}$$

where α is a positive constant (e.g. $\alpha = 2$), and Φ_r is the coverage ratio of the element r, defined as follows :

$$\Phi_r = \left\{ \begin{array}{cc} \frac{1}{C_r - k} & \text{if } C_r > k \\ \text{-1} & \text{otherwise.} \end{array} \right\} \tag{7.2}$$

Here, C_r is the degree of the object r in G and k is an integer lower bound indicating at least how many times the observed object should be covered, i.e. how many objects in S should simultaneously cover an object in R. This so called "coverage ratio" Φ_r is positive if the element r is over-covered, i.e. more than one sensor could cover it, and negative otherwise.

In reference to D_s, each node s selects a Decision Time Delay (DTD_s), broadcasts it to its neighbor, meanwhile collecting its neighbors' DTD and AM. From the received DTD and AM messages, each node builds a Delay List (DL_s) and a List of Awake Neighbors (LAN_s). After DTD_s time elapsed, each node s makes a decision based upon LAN_s and DL_s :
For all $r \in R_s$, if R_s can be covered using only nodes present in LAN_s and/or nodes j present in DL_s for which DTD_j is greater than DTD_s, then node s goes to sleep. Otherwise, s decides to be active and broadcasts an AM to inform other nodes of its decision.

In the last step, nodes go to sleep in a greedy manner, *i.e.* if the coverage problem can be solved with the already known awake nodes in the LAN (due to their higher drowsiness factor these nodes have decided earlier on their sleep/awake status) and some of the neighbors with lower drowsiness factor (these nodes will decide their sleep/awake status later), then the node greedily selects to sleep and leaves the sensing to those already being awake and those who haven't decided their status yet.

Briefly, the CGS algorithm works as follows.

1. Run the network for a period of T

2. Wake up all sensors
3. Nodes with energy enough for at least one more period broadcast local Hello messages containing node geographical location
4. Each node s calculates its own drowsiness factor D_s
5. Based on D_s each node selects a Decision Time Delay (DTD_s)
6. Each node s broadcasts its DTD_s and collects other nodes' Awake Messages AM
7. From the received AM, each node builds a List of Awake Neighbors (LAN_s)
8. After DTD_s each node s decides its state

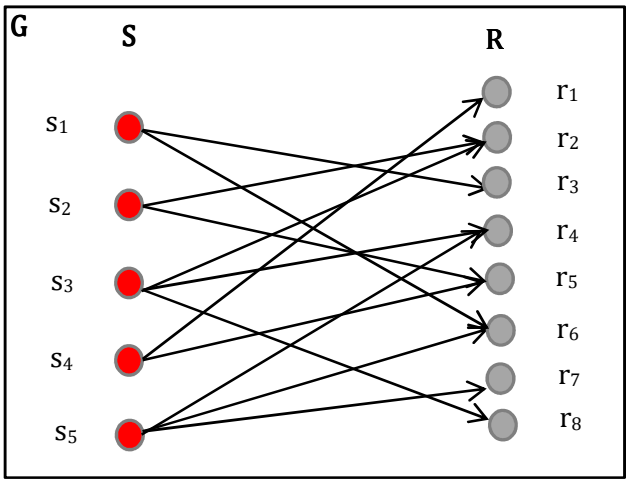


FIGURE 7.1: Bipartite graph showing the coverage of the sensors ($s_1, ..., s_5$) and the sensing regions ($r_1, ..., r_8$)

.

7.3 Proposed Model

7.3.1 Problem Definition

The objectives of this proposed model are stated as follows.

— Ensuring real-time monitoring coverage, via updating the monitoring set to the dynamic changes in the network topology (*cf.* Chapter 2 requirements 1 & 4).

— Since the monitoring activity is only one of the activities of the things, the power of batteries is consumed not only by monitoring, but also by other activities (sensing, transmission and reception). These are the primary activities defined by the original mission of the IoT network. They are not controlled by the monitoring mechanism, however, the eventual changes in the power resources as a result of the primary function of the things should be followed dynamically, since they affect the monitoring scheduling (*cf.* Chapter 2 requirements 2 & 4).

— Moreover, we aim at a monitoring mechanism that is fully inoperable with the standardized IoT protocol suite, especially the IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN) and the Routing Protocol for Low-power and lossy networks (RPL) (*cf.* Chapter 2) requirement 5.

Targeting the stated requirements, our main contribution in this work is the adaptation of the Controlled Greedy Sleep (CGS) [118] to conform to the full monitor coverage objective of

mission-critical 6LoWPAN-based IoT networks. To guarantee the required network coverage for a planning schedule, a coordinated, dynamic and distributed monitoring sleep scheduling algorithm is proposed. The algorithm contains two major elements :

— a cooperation protocol between nodes to assure the distributed scheduling, and
— an efficient computation algorithm to prepare the monitors' awake/sleep decisions.

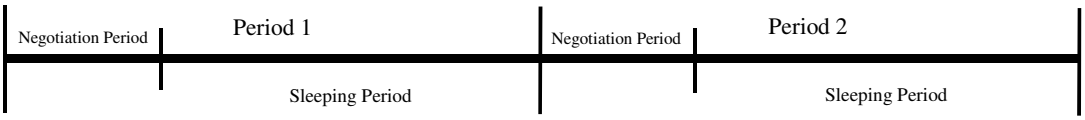


FIGURE 7.2: Monitoring Timeline.

The monitoring activity is organized in a timeline that is decomposed into a sequence of periods, $T = \{t_1, t_2, ..., t_m\}$. Each period is characterized by the set of active monitors and the duration of the period : $t_m = (S_j^a, t_j)$, where the set S_j^a is the active monitor subset during t_j that solves the coverage of the graph representing the *current* network topology. In our first proposition, we suppose that the period length is the same for all periods.

Each node s has its own candidate neighbor set which is the set of the nodes that can be reached via linklocal unicast, comprising of other potential monitors S_s and targets R_s. A monitoring node covers (a subset of) its candidate neighbor set. At the beginning of each monitoring period, there is a small negotiation period (*cf.* Fig. 7.2), where the communication between neighbors is established. It is imperative that this periodical communication between neighboring nodes is accomplished successfully for the following purposes :

— updating the candidate neighbor set S_s, so that the current List of Awake Neighbors LAN is known, which constantly changes due to the lossy nature of 6LoWPANs or simply because of the applied duty cycling mechanism,
— informing neighboring nodes of updated coverage ratio of each node, and
— informing neighboring nodes of the monitoring awake/sleep decision of each node for the next period.

Our problem is analogous to the one dealt with by the CGS algorithm [118], however, with some differences.

— In CGS, the sets of monitors and observed sensors are disjoint, and represented by geographical regions (*cf.* Fig. 7.1). In the proposed model, the two sets are actually the same, where a monitored node can also be a monitor for another.
— In CGS, sensors are identified by their geographical locations, instead in 6LoWPAN-based IoT networks, the nodes' radio communication ranges are defined by link-local reachability, where nodes are discovered by the 6LoWPAN *Neighbor Discovery Protocol (NDP)* [1], and identified by unique RIME/IPv6 addresses.
— There is an edge between monitor s and element r in G, if and only if r is within the radio environment of s. For monitor placement, the direction of edges is irrelevant, which implies that if there is a directed edge from s to r, s will be able to monitor r and also r can monitor s. Therefore, neighborhood and communication links are a symmetric notion. The undirected graphs are used in several routing protocols as in the models of [165] and [166].
— In our dynamic monitor scheduling algorithm, it is important to increase the knowledge of every node, such that it knows the neighbors of its neighbors. One neighbor of one of its neighbors is called Neighbor-of-Neighbor (NoN) [167]. A node's awake/sleep scheduling in the next period t_j is affected by the state of its NoN_s. The requirement of the knowledge of NoN_s can be illustrated in Fig. 7.3. Let the List of Awake Neighbors be called LAN. For v_1, $LAN_1 = \{2, 3, 5\}$, and $LAN_3 = \{1, 6\}$.

1. NDP is a messaging protocol that facilitates the discovery of neighboring devices over a network [41].

Supposing that it is decided that v_6 is sleep-monitoring in the next period t_j, if v_1 goes to sleep in the same period v_3 will not be covered. Therefore, v_1 should know its neighbors of neighbors, which includes $NoN_1 = \{v_4, v_5, v_6, v_7\}$. For v_1, knowing that a member of its NoN_1, namely, v_6, is sleeping *should* decide to stay active-monitoring. Otherwise, a neighbor of v_1, which is v_3, cannot be covered.

The proposed monitoring mechanism is described in Algorithms 8, 9, 10, and 11. The monitoring is functioning during the length of the timeline, represented by $Timeline\_Length$. At the beginning of a new monitoring period t_m, all nodes wake up (Algorithm 9 Step 9.1), and estimate their remaining power (E_s) and initialize their parameters. Nodes with remaining energy level high enough for monitoring (greater than a given $Energy\_Threshold$) for at least one more period locally broadcast an Awake Message (AM) (Step 9.8). Otherwise, to conserve the remaining power for its primary function (sensing, actuation, and/or transmission), it broadcasts a Sleep Message SM and sleeps (Steps 9.9 & 9.10). It is noteworthy that the monitoring mechanism does not influence the node's primary duty cycle, *i.e.* the radio is turned off only if the node is idle with respect to its primary function.

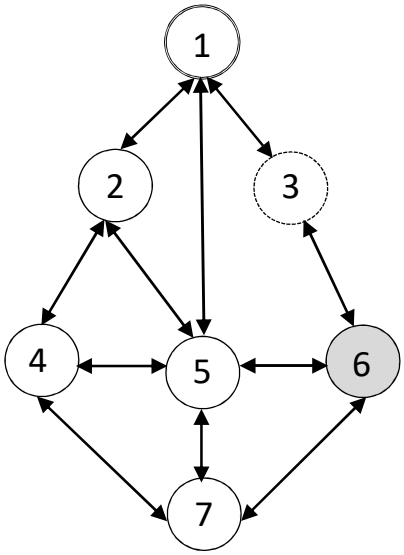


FIGURE 7.3: For v_1, knowing that a member of its NoN_1, namely, v_6, is sleeping decides to stay active-monitoring to ensure that its neighbor, v_3, is covered.

When a node receives a message from a neighbor, there are several tasks to perform : (1) it updates its List of Awake Neighbors (LAN_s), either by adding or removing this neighbor's address according to the neighbor's received state (active or asleep) (Algorithm 10 Step 10.2). Then, (2) updates its list of Neighbors of Neighbors (NoN_s) from the received list of neighbors, $LAN_{neighbor}$ (Step 10.3). (3) It computes its own coverage ratio (*cf.* Equation 7.3), and (4) updates its Delay List DL_s (Steps 10.4 & 10.5). Finally, (5) it broadcasts the updated parameters to its neighbors (Step 10.8).

At the end of the negotiation period, each node has to make a decision, whether it will be awake or asleep for the rest of the monitoring period. Algorithm 11 describes how the decisions are made which basically depend on the $drowsiness\_factor_s$ (*cf.* Equation 7.1). The drowsiness factor for each monitor includes the sum of coverage ratios of the objects it is able to monitor. Negative drowsiness indicates that a monitor is not allowed to sleep. The smaller the energy E_s of a monitor candidate, the larger is its drowsiness factor. On the

Algorithm 8 PROCEDURE DYNAMIC_DISTRIBUTED_MONITORING

Input: $Energy\_Threshold, Timeline\_Length, Period\_Length, Negotiation\_Period$
Output: Real-time monitoring schedule of 6LoWPAN-based IoT network

```
begin
8.1   while timeline_timer < Timeline_Length do
8.2     while period_timer < Period_Length do
8.3       while negotiation_timer < Negotiation_Period do
8.4         forEach s ∈ G do
8.5           START_UP();
8.6           RECEIVE_MESSAGE();
8.7           DECIDE_STATE();
8.6         end forEach
8.7       end while
8.8     end while
8.9   end while
end
```

Algorithm 9 PROCEDURE START_UP

Input: $Energy\_Threshold$
Output: Initialize node state

```
begin
9.1   RADIO_ON();
9.2   if E_s > Energy_Threshold do
9.3     state_s ⟵ 1;
9.4     drowsiness_factor_s ⟵ -1;
9.5     coverage_ratio_s ⟵ -1;
9.6     DTD_s ⟵ 0;
9.7     LOCAL_BROADCAST(AM, state_s, drowsiness_factor_s, coverage_ratio_s, DTD_s);
9.8   else
9.9     LOCAL_BROADCAST(SM, state_s ⟵ 0);
9.10    RADIO_OFF();
9.11  end if
end
```

contrary, a small drowsiness means large Decision Time Delay (DTD) (*cf.* Equation 7.4). These delays provide priorities when nodes announce their Awake Messages (AM).

A monitor participating in several critical coverages is likely to engage in more possible solutions than another covering objects which are simultaneously covered by alternative nodes. Therefore, they have larger drowsiness factors (*cf.* Equation 7.3). This property enforces the nodes in critical situations to deactivate the monitoring whenever it is possible, and permits to load monitors being in less critical situations.

Each node $s \in S$ has received the $coverage\_ratio_{neighbor}$ of the members of its LAN_s and NoN_S. If at least one of its neighbors or neighbors of neighbors is under-covered *i.e.* has a negative coverage ratio, it indicates that at most one node is able to monitor it, therefore s decides to stay awake to maintain successful coverage (Step 11.9). Accordingly, it broadcasts an AM (Step 11.10). Otherwise, it *can* sleep depending on the comparison between its own DTD (*cf.* Equation 7.4) with respect to its neighbors' $DTD_{neighbor}$ that were previously received and saved in the Delay List DL (Step 11.4). In the case where s has smallest DTD_s, it broadcasts a SM and goes to sleep (Steps 11.5 - 11.7).

Algorithm 10 PROCEDURE RECEIVE_MESSAGE

Input: Address of $neighbor$, $state_{neighbor}$, $LAN_{neighbor}$, $coverage\_ratio_{neighbor}$, $DTD_{neighbor}$
Output: Update LAN_s, NoN_s, $coverage\_ratio_s$, DL_s ; LOCAL_BROADCAST updated parameters

begin
10.1 **if** $state_{neighbor} > 1$ **do**
10.2 $LAN_s \longleftarrow LAN_s \cup neighbor$;
10.3 $NoN_s \leftarrow NoN_s \cup LAN_{neighbor}$;
10.4 UPDATE_COVERAGE_RATIO($coverage\_ratio_{neighbor}$);
10.5 UPDATE_DL($DTD_{neighbor}$);
10.6 **else** $LAN_s \longleftarrow LAN_s/neighbor$;
10.7 **end if**
10.8 LOCAL_BROADCAST($state_s, coverage\_ratio_s, LAN_s, NoN_s$);
end

Algorithm 11 PROCEDURE DECIDE_STATE

Input: LAN_s, NoN_s, DL_s
Output: Node s decides whether to stay active monitoring or sleep in t_j and accordingly broadcast AM or SM

begin
11.1 **if** $coverage\_ratio_{neighbor}$ & $coverage\_ratio_{NoN} >= 0$ $\forall neighbor \in LAN_s, \forall NoN \in NoN_s$ **do**
11.2 COMPUTE_DROWSINESS_FACTOR();
11.3 COMPUTE_DTD();
11.4 **if** $DTD_s < DTD_{neighbor}$ $\forall neighbor \in LAN_s, \forall DTD_{neighbor} \in DL_s$
11.5 LOCAL_BROADCAST($SM, state_s \longleftarrow -1$);
11.6 WAIT(DTD_s);
11.7 RADIO_OFF();
11.8 **end if**
11.9 $drowsiness\_factor_s \longleftarrow -1$;
11.10 LOCAL_BROADCAST(AM);
11.11 **end if**
end

$$\Phi_r = \left\{ \begin{array}{cc} \frac{1}{C_r - 1} & \text{if } C_r > k \\ -1 & \text{otherwise.} \end{array} \right\} \tag{7.3}$$

$$DTD = \left\{ \begin{array}{cc} \frac{1}{D_s} & \text{if } D_s > 1 \\ 0 & \text{otherwise.} \end{array} \right\} \tag{7.4}$$

7.4 Implementation & Experimental Setup

The methodology is implemented in the Contiki Operating System. For dynamic monitoring placement and scheduling, the current power level of nodes should be estimated as accurately as possible. The device's power is related to the monitoring as well as the activities related to the primary function of sensing, actuation, processing and transmission. The monitoring scheduling cannot influence the energy consumption of the primary function however, it must be taken into consideration. The WisMote is taken as a candidate platform for

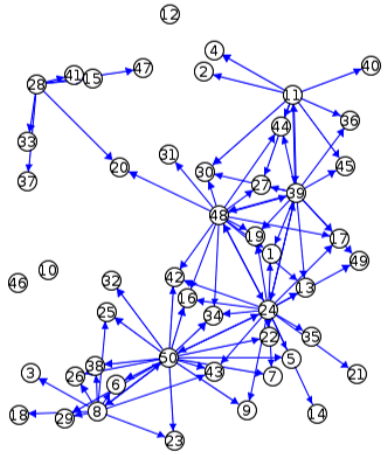


FIGURE 7.4: Radio communication within a network of 50 devices of type WisMote.

Mote output

File Edit View

Time	Mote	Message
00:00.345	ID:2	Rime started with address 2.0
00:00.357	ID:2	MAC 02:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 2.
00:00.360	ID:6	Rime started with address 6.0
00:00.366	ID:2	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.370	ID:2	Starting 'Broadcast process'
00:00.371	ID:6	MAC 06:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 6.
00:00.374	ID:2	START MONITORING TIMELINE
00:00.376	ID:2	START PERIOD
00:00.379	ID:2	START NEGOTIATION
00:00.381	ID:6	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.385	ID:6	Starting 'Broadcast process'
00:00.387	ID:2	INITIAL STATE -1, NEIGHBOR DISCOVERY -> BROADCAST AM
00:00.389	ID:6	START MONITORING TIMELINE
00:00.391	ID:6	START PERIOD
00:00.394	ID:6	START NEGOTIATION
00:00.401	ID:6	INITIAL STATE -1, NEIGHBOR DISCOVERY -> BROADCAST AM
00:00.436	ID:18	Rime started with address 18.0
00:00.447	ID:18	MAC 12:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 18.
00:00.457	ID:18	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.457	ID:4	Rime started with address 4.0
00:00.461	ID:18	Starting 'Broadcast process'
00:00.465	ID:18	START MONITORING TIMELINE
00:00.467	ID:18	START PERIOD
00:00.469	ID:4	MAC 04:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 4.
00:00.470	ID:18	START NEGOTIATION
00:00.477	ID:18	INITIAL STATE -1, NEIGHBOR DISCOVERY -> BROADCAST AM
00:00.478	ID:4	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.482	ID:4	Starting 'Broadcast process'
00:00.487	ID:4	START MONITORING TIMELINE
00:00.489	ID:4	START PERIOD
00:00.491	ID:1	Rime started with address 1.0
00:00.491	ID:4	START NEGOTIATION
00:00.499	ID:4	INITIAL STATE -1, NEIGHBOR DISCOVERY -> BROADCAST AM
00:00.503	ID:1	MAC 01:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 1.
00:00.511	ID:7	Rime started with address 7.0
00:00.512	ID:1	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.516	ID:1	Starting 'Broadcast process'
00:00.520	ID:1	START MONITORING TIMELINE
00:00.522	ID:1	START PERIOD
00:00.522	ID:7	MAC 07:00:00:00:00:00:00:00 Contiki 3.0 started. Node id is set to 7.
00:00.525	ID:1	START NEGOTIATION
00:00.532	ID:7	nullmac nullrdc, channel check rate 128 Hz, radio channel 26
00:00.533	ID:1	INITIAL STATE -1, NEIGHBOR DISCOVERY -> BROADCAST AM

FIGURE 7.5: COOJA motes output : monitoring negotiation period started.

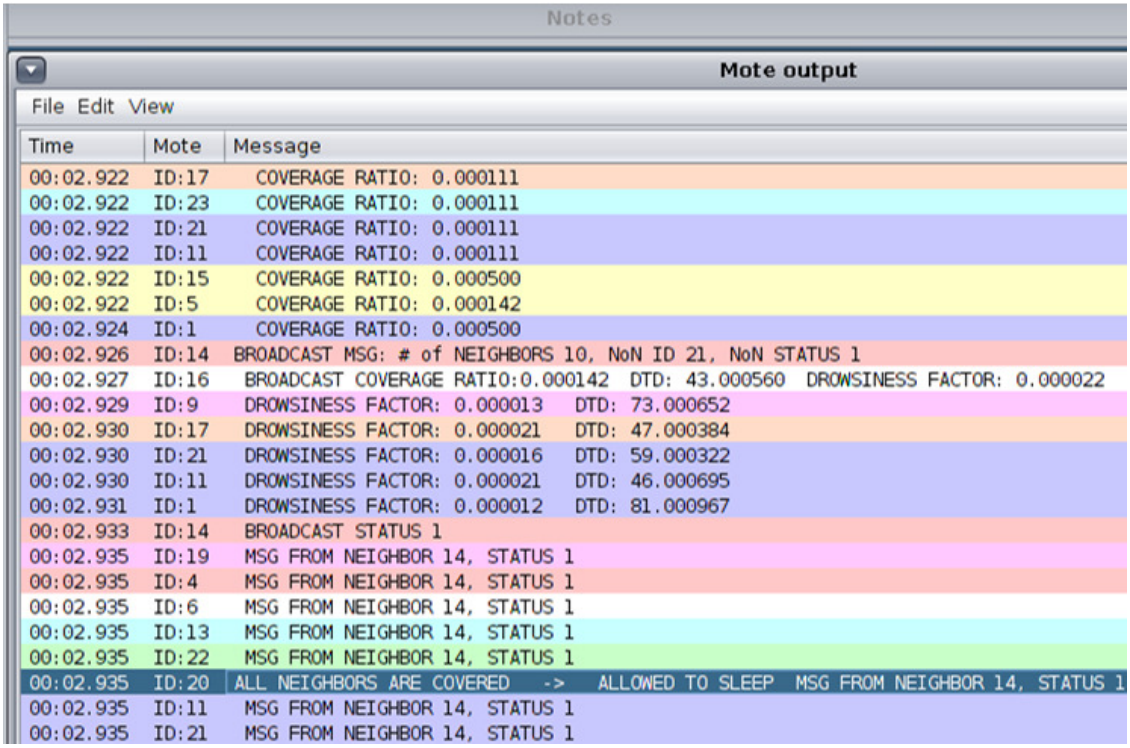


FIGURE 7.6: COOJA motes output : mote ID 20 is allowed to sleep.

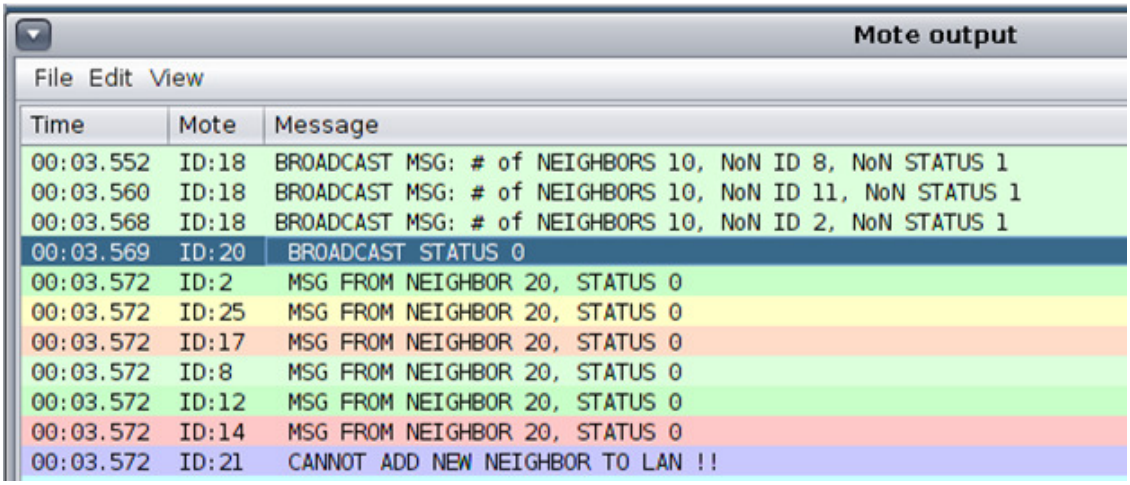


FIGURE 7.7: COOJA motes output : broadcast SM.

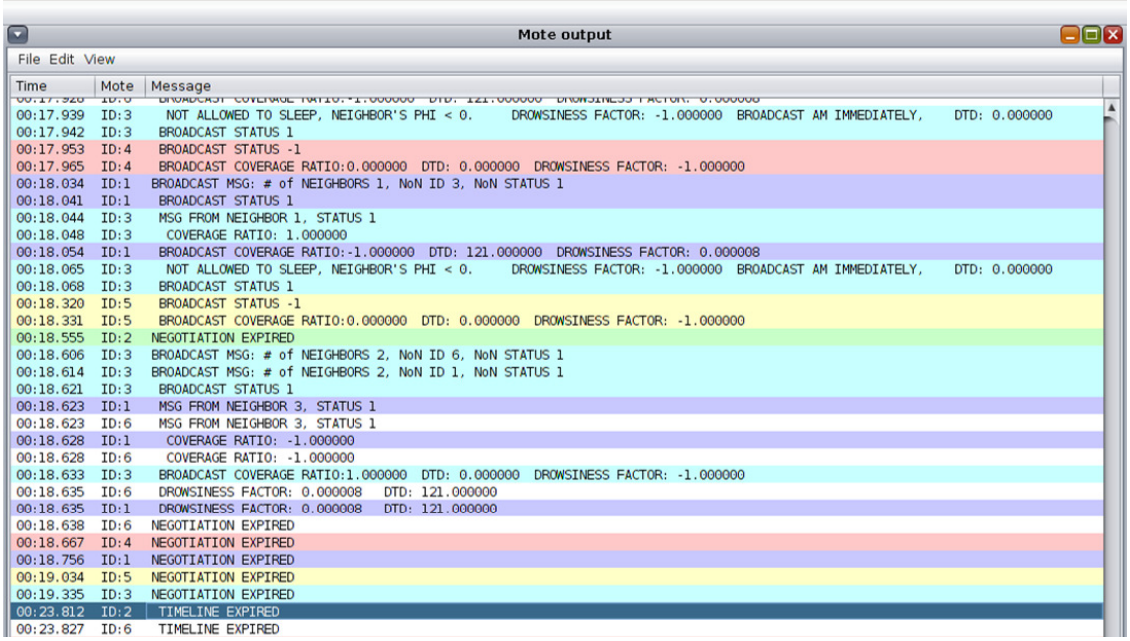


FIGURE 7.8: COOJA motes output : expiration of the monitoring timeline.

the monitoring mechanism. It features a 16-bit MSP430 with 20-bit support, 16 kB RAM, a nominal 128 kB, 192 kB or 256 kB ROM and CC2520 radio transceiver, with light, battery, and radio sensors. It is powered by a pair of AAA batteries with 3 volts. The total energy available by the WisMote is

$$2 \times (1.15\ Ah) \times (1.5\ V) \times (3600\ s) = 11421\ J = 11421000\ mJ \tag{7.5}$$

We Use the `POWER_TRACE` procedure, embedded in Contiki, to estimate the current energy level of the nodes. Its output is printed in timer ticks as follows,

- tx - the number of ticks the radio has been in transmit mode *(ENERGEST-TYPE-TRANSMIT)*
- rx - the number of ticks the radio has been in receive mode *(ENERGEST-TYPE-LISTEN)*
- cpu - the number of ticks the CPU has been in active mode *(ENERGEST-TYPE-CPU)*
- cpu-idle - the number of ticks the CPU has been in idle mode *(ENERGEST-TYPE-LPM)*

With each call of the `START_UP` procedure, `POWER_TRACE` is called and the current energy level E_s is estimated by executing the following computations (Algorithm 9 Step 9.2).

$$\textit{ticks-in-tx-mode} = \textit{energest-type-time(ENERGEST-TYPE-TRANSMIT)} \tag{7.6}$$

$$\textit{seconds-in-tx-mode} = \frac{\textit{ticks-in-tx-mode}}{\textit{RTIMER-ARCH-SECOND}} \tag{7.7}$$

To compute the average current consumption (in milliamperes, mA), multiply each of $tx, rx, cpu, cpu-idle$ with the respective current consumption in that mode in mA (obtain the values from the

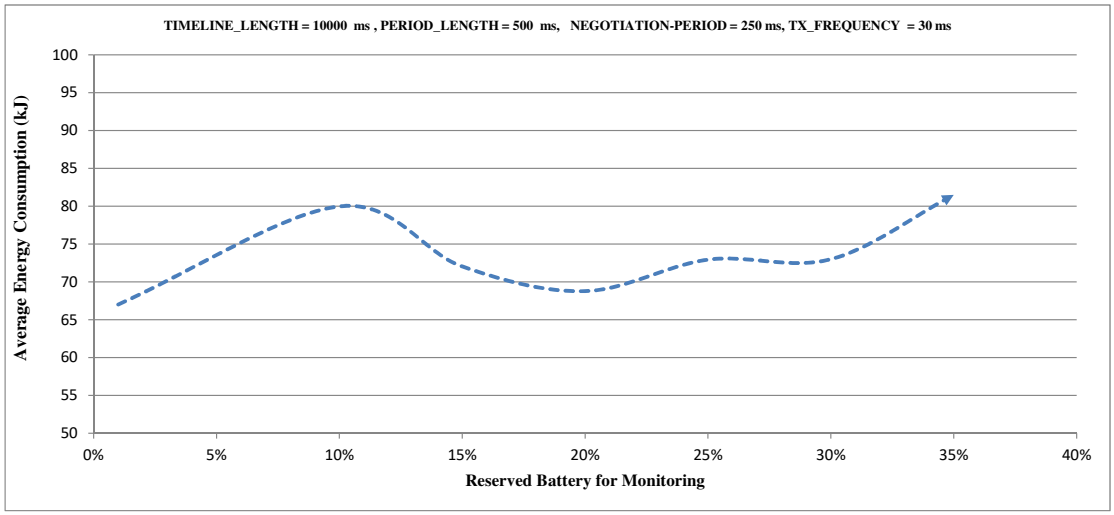


FIGURE 7.9: Effect of varying the size of the reserved battery for monitoring on the average energy consumption. Timeline Length = 10000 ms, Negotiation Period = 50 ms, Tx Frequency = 30 ms.

datasheet of the node), sum them up, and divide by RTIMER-ARCH-SECOND,

$$current = \frac{(tx \times current\text{-}tx\text{-}mode + rx \times current\text{-}rx\text{-}mode + cpu \times current\text{-}cpu + cpu\text{-}idle \times current\text{-}idle)}{RTIMER\text{-}ARCH\text{-}SECOND} \tag{7.8}$$

$$charge = \frac{current \times (cpu + cpu\text{-}idle)}{RTIMER\text{-}ARCH\text{-}SECOND} \tag{7.9}$$

To compute the power (in milliwats, mW) multiply the average current consumption with the voltage of the device :

$$power = current \times voltage \tag{7.10}$$

Finally, to compute the energy consumption (in millijoules, mJ), multiply the power with the duration in seconds or multiply the charge with the voltage of the system :

$$energy = charge \times voltage \tag{7.11}$$

7.4.1 Experimental Results

Experimentation is performed within the Contiki OS using COOJA network simulator the *de facto* simulator for constrained-IoT applications. The dynamic distributed monitoring mechanism is tested using network instances of random sizes and topologies (network size ranges from 20 to 200 nodes). Fig. 7.4 illustrates the radio communication and states in a network of 50 nodes of type WisMote.

The "Mote output" window in Cooja shows the motes' serial port printouts. Fig. 7.5 highlights the beginning of the monitoring $Negotiation\_Period$. Fig. 7.6 illustrates the case where a node knows that its neighbors are covered after computing its $coverage\_ratio_s$ according to Equation 7.2. Fig. 7.7 shows when a node has decided to sleep following the `DECIDE_STATE` (*cf.* Algorithm 11), and broadcasts a Sleep Message SM including its updated sleeping $state_s$. The `DYNAMIC_DISTRIBUTED_MONITORING` procedure (*cf.* Algorithm 8) keeps running until the end of the monitoring timeline (*cf.* Fig. 7.8).

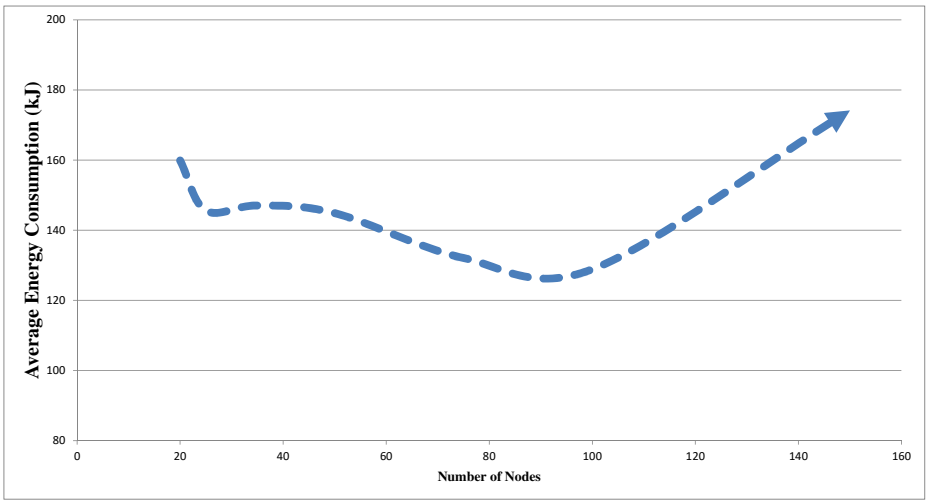


FIGURE 7.10: The distribution of energy consumption over different-sized networks, (a) average energy consumption. Timeline Length = 10000 ms, Period Length = 2000 ms, Negotiation Period = 50 ms, Tx Frequency = 30 ms, Reserved Battery = 10%.

Similar to the previously proposed monitoring mechanisms, 6.2.2 and 4.3, it is assumed that each node has a reserved battery for the monitoring activity across the entire timeline length, apart from the energy dedicated to the main function of the thing. During experimentation, we tested the model's sensitivity towards variations in the reserved battery for monitoring. 8 trials were run for which the reserved battery ranged from 1% to 35% of the total available battery of the WisMote, which corresponded to 112.10 - 3997.35 kJ. The $Timeline\_Length$, $Period\_Length$, $Negotiation\_Period$, and the frequency of transmission ($Tx - frequency$) were set in these trials to 10000 ms, 500 ms, 250 ms, and 30 ms, respectively. The results are displayed in Table 7.1.

Comparing between the two extreme thresholds, one where the reserved battery is tightened the most (1%), and another where it is stretched to 35%, produced an interesting result : the average energy consumption in the case of the 1% reserved battery is reduced by 21.55%. This result highlights the model's adaptability towards hard energy constraints, as it strives to preserve the scarce resources by effectively distributing the monitoring load. Some nodes decided not to participate to the monitoring activity at all thus rendering a zero level energy consumption. This decision was taken by the nodes after ensuring that the entire set of neighbors are covered by other monitors.

Another set of experiments were designed to test the correlation between the period length and the average energy consumption. The $period\_length$ should be carefully chosen such that it is neither too short nor too long. Too short a $period\_length$ may result in false alarms, and if it is too long it may unnecessarily exhaust the energy of monitors as they are awake-monitoring for quite a long time. Table 7.2 displays the average energy consumption and the standard deviation in response to varying the $Period\_Length$. The $Timeline\_Length$, $Negotiation\_Period$, and $Tx - frequency$ were fixed in all trials to

10000 ms, 250 ms, and 30 ms. A subset of those trials are displayed Fig. 7.11.

There is a trade-off between the level of energy consumption and the balance of the monitoring load among the nodes. It can be seen in Fig. 7.11 (also *cf.* the Standard Deviation column in 7.2) that a very short $Period\_Length$, 100 ms, results in an unfair distribution of the monitoring load, where some nodes exhaust comparatively high amounts of energy for monitoring while others are at a zero level consumption. This is illustrated by the high standard deviation value in Table 7.2. On the other hand, too long a $Period\_Length$ of 3000 ms revealed a significant rise in the average energy consumption, which is justified by the long monitoring duty cycles. It is detected that the best combination of a relatively low average energy consumption and a good balance between the monitoring loads is achieved when the $Period\_Length$ is set to 1000 ms.

The final set of experiments were performed to test the effect of the network size on energy consumption, as well as the model's scalabilty. The network size was increased to 150 nodes and 3200 links. The results are shown in Table 7.3. It is interesting and noteworthy that the percentage of energy consumption from the total available battery if WisMote never exceeds 1.36% regardless to the network size. Fig. 7.10 depicts the increase in the average energy consumption with respect to the network size, which is almost negligible.

TABLE 7.1: Energy consumption of a network of 20 nodes with different levels of reserved battery for monitoring. $Timeline\_Length$ = 10000 ms, $Period\_Length$ = 500 ms, $Negotiation\_Period$ = 250 ms, $Tx - frequency$ = 30 ms.

Reserved battery(%)	**Reserved battery(kJ)**	**Avg. consumption(kJ)**
1	114.21	67.01
10	1142.10	79.97
15	1713.15	72.03
20	2284.20	68.78
25	2855.25	72.94
30	3426.30	72.99
35	3997.35	81.45

TABLE 7.2: Energy consumption of a network of 40 nodes with different period lengths. $Timeline\_Length$ = 10000 ms, $Negotiation\_Period$ = 250 ms, $Tx - frequency$ = 30 ms, reserved battery = 10% (1142.1 kJ).

Period Length (ms)	**Avg. consumption (kJ)**	**Standard Deviation (kJ)**
100	60.17	95.23
500	142.27	144.20
1000	156.82	31.34
2000	94.64	105.65
2500	125.79	45.37
3000	215.43	66.41

TABLE 7.3: Average and percentage of energy consumption of different-sized networks. $Timeline\_Length = 10000\ ms$, $Period\_Length = 500\ ms$, $Negotiation\_Period = 250\ ms$, $Tx - frequency = 30\ ms$.

Number of nodes	Avg. consumption(kJ)	% of consumption from available energy
20	159.89	1.36
25	145.61	1.27
50	144.89	1.26
75	132.09	1.27
100	128.89	1.14
150	174.21	1.13

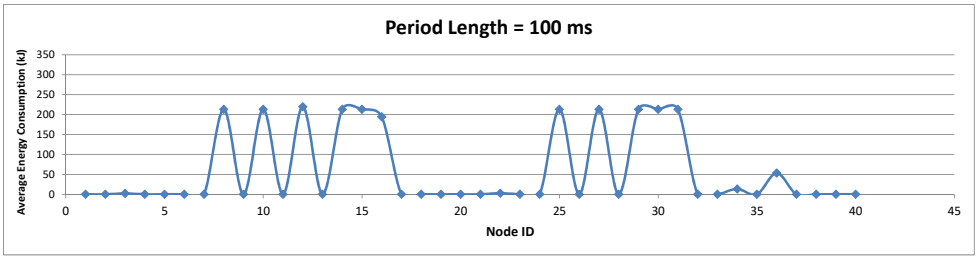
7.5 Summary

The proposed model targeted the dynamic distributed monitoring placement and scheduling of mission-critical IoT network, with complete interoperability with the IoT standardized protocols. The dynamic feature of the model ensures the real-time adaptation of the monitoring schedule to the frequent network instabilities without requiring to re-solve the monitoring placement and scheduling problem with each abrupt change in the network topology and nodes' availability. The distributed feature aims at reducing the communication overhead between monitors and the Border Router, a result of centralized monitoring mechanisms.

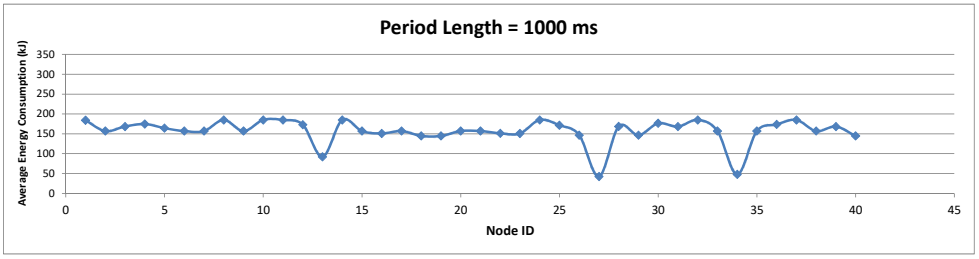
The dynamic monitoring mechanism follows the basic idea of the Controlled Greedy Sleeping (CGS) algorithm proposed in [118], but necessary adaptations for the scheduling of monitoring activities have been proposed. The monitoring awake/sleep schedule of nodes is computed using the notions of coverage ratio and drowsiness factor, which ensure the coverage of the entire set of critical nodes while prioritizing the awake/sleep decision based on coverage and energy levels. Successful neighbor-discovery and knowledge about the neighbors' state are achieved by inter-communication between nodes, which is scheduled at the beginning of each period, the negotiation period. Nodes with critical monitoring coverage (monitoring neighbors that are not covered by other monitors) are not allowed to sleep.

Performance evaluations and accurate energy levels estimation are achieved within Contiki/COOJA, the *de facto* network simulator for constrained IoT. Simulations were performed to evaluate the model's adaptability to harsh energy constraints. The results show that the harder the energy constraint is, the lower is the average energy consumption while ensuring full monitor-network coverage. A sensitivity analysis was conducted within the experiments to obtain the "best" combination between the parameters and minimize the trade-off between them.

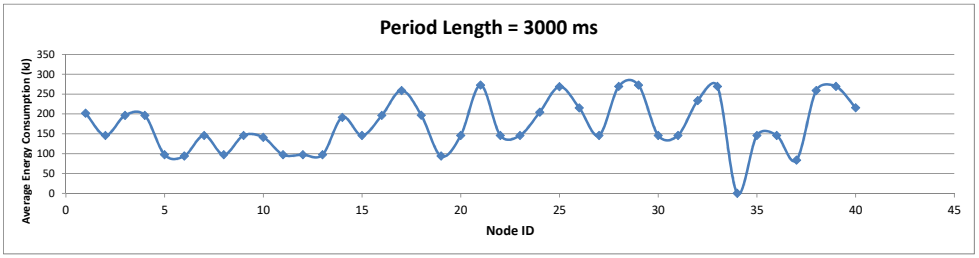
Compared to the three-phase decomposition (*cf.* 4.3), the dynamic distributed heuristic achieves better results with respect to computational complexity and scalability. Compared to the exact monitoring placement and scheduling (*cf.* 6.2.2), the only limitation is that the schedule is not *exact*, however, with the benefit of achieving robust, real-time adaptability to network changes and the reduced computational and communication overhead of the distributed mechanism, the performance of the dynamic model is superior. Further experimentation and comparisons between the two models are required to evaluate the approximation factor of the dynamic heuristic, which is the first target on the list of future work.



(A)



(B)



(C)

FIGURE 7.11: Effect of varying the period length on the average energy consumption, (a) period length = 100 *ms*; (b) period length = 1000 *ms*; (c) period length = 3000 *ms*. Timeline Length = 10000 ms, Negotiation Period = 50 ms, Tx Frequency = 30 ms, Reserved Battery = 10%

Chapter 8

Conclusions and Perspectives

By connecting billions of things to the Internet, IoT created a plethora of applications that touch every aspect of human life. Time-sensitive, mission-critical services (*e.g.* health monitoring [3], safety control and fault detection [17, 18, 19]), require robust connectivity and strict reliability constraints. On the other hand, the IoT relies mainly on Low-power Lossy Networks (LLN), which are unreliable by nature due to their limited resources, hard duty cycles, dynamic topologies, and uncertain radio connectivity. Faults in LLNs are common rather than rare events [20], therefore, maintaining continuous availability of devices and reliability of communication, are critical factors to guarantee a constant, reliable flow of application data.

The absence of any monitoring mechanism for detecting networks' faults would dramatically reduce network performance, which renders monitoring the IoT network state a vital research area that will develop in significance [23]. A *proactive* monitoring mechanism could greatly improve robustness, reliability and eventually, Quality of Service (QoS), which will significantly increase the uptake of the technology by stakeholders. After a comprehensive literature review, and up to our knowledge, it is clear that there is a call for a new approach to monitoring the unreliable nodes and links in an optimized, energy-efficient, proactive manner, and complete interoperability with IoT protocols [27].

To target this research gap, our contributions address the correct assignment (placement) of the monitoring nodes. Monitors should be capable of covering the entire critical network, such that they can observe the traffic and infer the current network state. This problem is known as the minimum assignment problem, which is NP-hard. We target scalable monitoring by mapping the assignment problem into the well-studied MVC problem, also NP-hard. However, it is known that MVC is polynomial-time solvable on trees, and has been proven to be Fixed-Parameter Tractable (FPT) on graphs that are *"like"* tree, known as *tree decompositions*. In tandem with the Routing Protocol for Low-power and Lossy Networks (RPL), we proposed algorithms 2, 3, and 4 to convert the DODAG into a *nice-tree* decomposition with its parameter (treewidth) restricted to the value one. As a result of these propositions, the monitor placement becomes only Fixed-Parameter Tractable, and can also be polynomial-time solvable.

To prolong network longevity, the monitoring role should be distributed and balanced between the entire set of nodes. To that end, assuming periodical functioning, we propose in a second contribution to schedule between several subsets of nodes; each is covering the entire network. Given its NP-hardness, we opted to tackle the problem by adopting a *Divide and Conquer* approach. A three-phase centralized computation of the scheduling was proposed (*cf.* Section 4.3). The proposition decomposes the monitoring problem and maps it into three well-known sub-problems, for which approximation algorithms already exist in the literature. Thus, the computational complexity can be reduced. It is also a proof of concept to emphasize that energy-efficient solutions for monitoring IoT networks do exist.

The first phase is responsible for computing several subsets of potential monitors by solving a *minimal* Vertex Cover Problem; by proposing a Constraint Generation algorithm.

The optimal scheduling of the Vertex Covers is handled in the second phase, by modeling the scheduling as a multi-objective Generalized Assignment Problem (GAP). For further reduction of energy consumption, monitors are sequenced across time periods to minimize the state transitions of nodes, which generally consume extra energy. This part of the problem is modeled as a Traveling Salesman Path Problem (TSP-Path), which outputs the sequence that minimizes the total number of state transitions (from active-monitoring to sleep and vice versa). The sequence is generated using a dynamic programming implementation of the TSP-Path.

Experimentation using several test instances of different sizes, up to 200 nodes and 2463 links, prove that the proposed model effectively provides full monitor coverage with minimal monitoring energy consumption and communication overhead. For all instances, the minimum residual battery does not fall below 80%, and for some, the reduction of the number of nodes' state transitions reach up to 80 %. It is interesting to emphasize that when the battery dedicated for monitoring is sufficiently large, fewer Vertex Covers are assigned to periods and less monitor scheduling is required. On the other hand, when the reserved battery is relatively small, more Vertex Covers are required to monitor the same number of periods and scheduling for minimal energy consumption is critical.

However, the one major limitation of the proposed three-phase decomposition is that it is not an exact solution. Therefore, it does not guarantee global optimality. In fact, to the best of our knowledge, the exact solution to the defined problem is not yet known, which we target in our third contribution (*cf.* Section 6.2.2). We provide the exact solution to the minimum monitor assignment problem with a duty-cycled monitoring approach, by formulating a Binary Integer Program (BIP). The optimal schedule guarantees monitoring coverage with minimum energy consumption. The solution is incorporated into a centralized, passive monitoring mechanism that is interoperable with RPL and 6LoWPAN protocols. Experimentation is designed using network instances of different topologies and sizes that ranged from 25 to 600 nodes, and from 78 to 179700 links. Results demonstrate the effectiveness of the proposed model in realizing full monitoring coverage with minimum energy consumption and communication overhead while balancing the monitoring role between nodes.

The most remarkable conclusion to emerge from the experimental results of the exact model is that tightening the reserved battery for monitoring has the advantage of balancing the monitoring load between nodes. Consequently, the average energy consumption per node decreases with hard energy constraints. This finding highlights the fact the size of reserved battery should be set with considerable care. On the other hand, it is a fact that computing the exact solution to the BIP model for large-sized networks is computationally expensive, a result of the NP-hardness of the MVC problem (*cf.* constraint 6.9). This fact implies an exponential lower bound on the running time of solving the proposed linear BIP model. Despite computational limitations, the proposed mechanism serves as a benchmark for comparisons and performance evaluation of contemporary models.

Nevertheless, the proposed models so far represented a more static view of the IoT network, rendering the optimal solution unrepresentative to the current situation in case a significant change in the network topology is detected. Dynamic models are required to avoid a complete re-optimization of problems. Chapter 7 targeted the dynamic distributed monitoring placement and scheduling of mission-critical IoT networks, with complete interoperability with the IoT standardized protocols. The dynamic feature of the model ensures real-time adaptation of the monitoring schedule to the frequent instabilities of networks, and the distributed feature aims at reducing the communication overhead between monitors and the Border Router, which is a common side-effect of centralized monitoring mechanisms.

Performance evaluations and accurate energy level estimations are achieved within Contiki/COOJA, the *de facto* network simulator for constrained IoT. Experiments illustrate the

effectiveness of the proposed model to achieve full network coverage dynamically. Successful neighbor-discovery and knowledge about the neighbors' state are achieved by intercommunication between nodes, which is scheduled at the beginning of each period, the negotiation period. Nodes with critical monitoring coverage (monitoring neighbors that are not covered by other monitors) are not allowed to sleep.

Simulations were performed to evaluate the model's adaptability to harsh energy constraints. Similar to the exact model, the results of the dynamic heuristic highlight that the harder the energy constraint is, the lower is the average energy consumption, meanwhile, full monitor-network coverage is guaranteed. Moreover, a sensitivity analysis was conducted within the experiments to obtain the "best" combination between the parameters and minimize the trade-off between energy consumption and the balance of the monitoring role among the nodes. A very short period length results in an unfair distribution of the monitoring role; where some nodes exhaust comparatively high amounts of energy for monitoring while others are at a zero level consumption. On the other hand, too long a period length revealed a significant rise in the average energy consumption; which is justified by the long monitoring duty cycles.

Compared to the three-phase decomposition (*cf.* Section 4.3), the dynamic distributed heuristic achieves better results concerning computational complexity and scalability. Compared to the exact monitoring placement and scheduling (*cf.* Section 6.2.2), the only limitation is that the schedule is not *exact*, however, with the benefit of achieving robust, real-time adaptability to network changes and the reduced computational and communication overhead of the distributed mechanism, the performance of the dynamic model is superior. However, further experiments with both models are needed to evaluate an experimental approximation factor of the dynamic heuristic. This activity is the first in the list of future works. In addition, a design and experimentation of active and hybrid monitoring and a study of these approaches are interesting to analyze the feasibility, computational complexity and energy consumption.

Bibliography

[1] Roberto Minerva, Abyi Biru, and Domenico Rotondi. “Towards a Definition of the Internet of Things (IoT)”. In: *IEEE Internet Initiative* 1 (2015), pp. 1–86.

[2] J. Jin et al. “An Information Framework for Creating a Smart City Through Internet of Things”. In: *IEEE Internet of Things Journal* 1.2 (2014), pp. 112–121. ISSN: 2327-4662. DOI: 10.1109/JIOT.2013.2296516.

[3] Moeen Hassanalieragh et al. “Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-based Processing: Opportunities and Challenges”. In: *2015 IEEE International Conference on Services Computing* (2015), pp. 285–292.

[4] B. Mostafa et al. “Distributed Monitoring in 6LoWPAN-based Internet of Things”. In: *2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT)* (2016), pp. 1–7. DOI: 10.1109/MoWNet.2016.7496626.

[5] B. Mostafa et al. “An Energy-Efficient Multiobjective Scheduling Model for Monitoring in Internet of Things”. In: *IEEE Internet of Things Journal* 5.3 (2018), pp. 1727–1738. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2792326.

[6] Wayne W Zachary. “An Information Flow Model for Conflict and Fission in Small Groups”. In: *Journal of Anthropological Research* 33.4 (1977). dataset: pp. 452–473.

[7] David Lusseau et al. “The bottlenose dolphin community of Doubtful Sound Features a Large Proportion of Long-lasting Associations”. In: *Behavioral Ecology and Sociobiology* 54.4 (2003). dataset: pp. 396–405.

[8] Ryan A. Rossi and Nesreen K. Ahmed. “The Network Data Repository with Interactive Graph Analytics and Visualization”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015). dataset: URL: http://networkrepository.com.

[9] Michelle Girvan and Mark EJ Newman. “Community Structure in Social and Biological Networks”. In: *Proceedings of the National Academy of Sciences* 99.12 (2002). dataset: pp. 7821–7826.

[10] Duncan J Watts and Steven H Strogatz. “Collective Dynamics of "Small-World" Networks”. In: *Nature* 393.6684 (1998). dataset: p. 440.

[11] Mark E.J. Newman. “Finding Community Structure in Networks Using the Eigenvectors of Matrices”. In: *Physical review E* 74.3 (2006). dataset: p. 036104.

[12] David Metcalf et al. “Wearables and the Internet of Things for Health: Wearable, Interconnected Devices Promise More Efficient and Comprehensive Health Care”. In: *IEEE pulse* 7.5 (2016), pp. 35–39.

[13] Hemant Ghayvat et al. “WSN and IoT-based Smart Homes and their Extension to Smart Buildings”. In: *Sensors* 15.5 (2015), pp. 10350–10379.

[14] Miao Yun and Bu Yuxin. “Research on the Architecture and Key Technology of Internet of Things (IoT) Applied on Smart Grid”. In: *2010 International Conference on Advances in Energy Engineering* (2010), pp. 69–72.

[15] Rohit Dhall and Vijender Solanki. “An IoT Based Predictive Connected Car Maintenance”. In: *International Journal of Interactive Multimedia & Artificial Intelligence* 4.3 (2017).

[16] Shibo He, Jiming Chen, and Youxian Sun. “Coverage and Connectivity in Duty-Cycled Wireless Sensor Networks for Event Monitoring”. In: *IEEE Transactions on Parallel and Distributed Systems* 23.3 (2011), pp. 475–482.

[17] F. Wu et al. “WE-Safe: A Wearable IoT Sensor Node for Safety Applications via LoRa”. In: *IEEE 4th World Forum on Internet of Things (WF-IoT)* (2018), pp. 144–148. DOI: 10.1109/WF-IoT.2018.8355234.

[18] Fan Wu, Taiyang Wu, and Mehmet Yuce. “An Internet-of-Things (IoT) Network System for Connected Safety and Health Monitoring Applications”. In: *Sensors* 19.1 (2019), p. 21.

[19] Murtaza Cicioğlu and Ali Çalhan. “IoT-based Wireless Body Area Networks for Disaster Cases”. In: *International Journal of Communication Systems* (2018), e3864.

[20] Farzad Kiani. “A Survey on Management Frameworks and Open Challenges in IoT”. In: *Wireless Communications and Mobile Computing* 2018 (2018).

[21] Miloš Stanisavljević, Alexandre Schmid, and Yusuf Leblebici. *Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures*. Springer Science & Business Media, 2010.

[22] Antonio J Jara, Latif Ladid, and Antonio Fernandez Gómez-Skarmeta. “The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities.” In: *JoWua* 4.3 (2013), pp. 97–118.

[23] John A Stankovic. “Research Directions for the Internet of Things”. In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 3–9.

[24] Jie Lin et al. “A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications”. In: *IEEE Internet of Things Journal* 4.5 (2017), pp. 1125–1142.

[25] Kevin Dominik Korte, Anuj Sehgal, and Jürgen Schönwälder. “A Study of the RPL Repair Process using ContikiRPL”. In: *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer. 2012, pp. 50–61.

[26] Olfa Gaddour and Anis Koubâa. “RPL in a Nutshell: A Survey”. In: *Computer Networks* 56.14 (2012), pp. 3163–3178.

[27] Fadele Ayotunde Alaba et al. “Internet of Things Security: A Survey”. In: *Journal of Network and Computer Applications* 88 (2017), pp. 10–28.

[28] Zhengguo Sheng et al. “A survey on the IETF Protocol Suite for the Internet of Things: Standards, Challenges, and Opportunities”. In: *IEEE Wireless Communications* 20.6 (2013), pp. 91–98.

[29] B. Mostafa. “Monitoring Internet of Things Networks”. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. Apr. 2019, pp. 295–298. DOI: 10.1109/WF-IoT.2019.8767203.

[30] Prasan Sahoo, Hiren Thakkar, I Hwang, et al. “Pre-scheduled and Self Organized Sleep-scheduling Algorithms for Efficient K-coverage in Wireless Sensor Networks”. In: *Sensors* 17.12 (2017), p. 2945.

[31] Bruno Dorsemaine et al. “Internet of Things: A Definition & Taxonomy”. In: *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*. IEEE. 2015, pp. 72–77.

[32] Ovidiu Vermesan et al. “Internet of Things Strategic Research Roadmap”. In: *Internet of Things-Global Technological and Societal Trends* 1.2011 (2011), pp. 9–52.

[33] *Internet of Things Defined – Tech Definitions by Gartner*. https://www.gartner.com/it-glossary/internet-of-things.htm. Accessed: 2019-09-11.

[34] Gerd Kortuem et al. “Smart Objects as Building Blocks for the Internet of Things”. In: *IEEE Internet Computing* 14.1 (2009), pp. 44–51.

[35] Patrick Guillemin et al. *Internet of Things Global Standardisation-State of Play*. River Publishers, 2014.

[36] Manuel Dıaz, Cristian Martın, and Bartolomé Rubio. “State-of-the-art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing”. In: *Journal of Network and Computer applications* 67 (2016), pp. 99–117.

[37] IOT Analytics. *IoT 2018 in Review: The 10 most Relevant IoT Developments of the Year*. 2018.

[38] Tobias Heer et al. “Security Challenges in the IP-based Internet of Things”. In: *Wireless Personal Communications* 61.3 (2011), pp. 527–542.

[39] Daniele Miorandi et al. “Internet of Things: Vision, Applications and Research Challenges”. In: *Ad hoc networks* 10.7 (2012), pp. 1497–1516.

[40] Simon Dobson et al. “A Survey of Autonomic Communications”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 1.2 (2006), pp. 223–259.

[41] Zach Shelby and Carsten Bormann. *6LoWPAN: The Wireless Embedded Internet*. Vol. 43. John Wiley & Sons, 2011.

[42] Carles Gomez et al. “Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing”. In: (2012).

[43] J.-P. Vasseur and D Culler. “Routing Over Low Power and Lossy Networks (roll)”. In: *IETF Working group* (2008).

[44] Zach Shelby. “Constrained RESTful environments (CoRE) link format”. In: (2012).

[45] Isam Ishaq et al. “IETF Standardization in the Field of the Internet of Things (IoT): A Survey”. In: *Journal of Sensor and Actuator Networks* 2.2 (2013), pp. 235–287.

[46] Jonas Olsson. “6LoWPAN Demystified”. In: *Texas Instruments* 13 (2014).

[47] Chris Lu. “Overview of Security and Privacy Issues in the Internet of Things”. In: *Internet of Things (IoT): A vision, Architectural Elements, and Future Directions* (2014).

[48] J. Vasseur et al. “RPL: The IP Routing Protocol Designed for Low Power and Lossy Networks”. In: *Internet Protocol for Smart Objects (IPSO) Alliance* 36 (2011).

[49] Oana Iova, Fabrice Theoleyre, and Thomas Noel. “Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network”. In: *Ad Hoc Networks* 29 (2015), pp. 45–62.

[50] Jean-Philippe Vasseur et al. *Routing Metrics used for Path Calculation in Low-Power and Lossy Networks*. Tech. rep. 2012.

[51] Hamidreza Kermajani and Carles Gomez. “On the Network Convergence Process in RPL over IEEE 802.15. 4 Multihop Networks: Improvement and Trade-offs”. In: *Sensors* 14.7 (2014), pp. 11993–12022.

[52] Pascal Thubert. *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*. Tech. rep. 2012.

[53] Nandakishore Kushalnagar, Gabriel Montenegro, Christian Schumacher, et al. “IPv6 Over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals”. In: (2007).

[54] M Shamim Hossain and Ghulam Muhammad. “Cloud-assisted Industrial Internet of Things (IIoT)–enabled Framework for Health Monitoring”. In: *Computer Networks* 101 (2016), pp. 192–202.

[55] Xu Li et al. “Smart Community: An Internet of Things Application”. In: *IEEE Communications Magazine* 49.11 (2011).

[56] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver. “Network Management in Wireless Sensor Networks”. In: *Handbook of Mobile Ad Hoc and Pervasive Communications* (2006), pp. 1–20.

[57] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver. “WiNMS: Wireless Sensor Network-management System, an Adaptive Policy-based Management for Wireless Sensor Networks”. In: (2006).

[58] J. Zhang et al. “Resource Management of Task Oriented Distributed Sensor Networks”. In: 3 (2001), pp. 513–516.

[59] Zhou Ying and Xiao Debao. “Mobile Agent-based Policy Management for Wireless Sensor Networks”. In: *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing* 2 (2005), pp. 1207–1210.

[60] Tae-Hyung Kim and Seongsoo Hong. “Sensor Network Management Protocol for Statedriven Execution Environment”. In: *In International Conference on Ubiquitous Computing* (2003).

[61] C-L Fok, G-C Roman, and Chenyang Lu. “Mobile Agent Middleware for Sensor Networks: An Application Case Study”. In: (2005), pp. 382–387.

[62] Chieh-Yih Wan et al. “Siphon: Overload Traffic Management Using Multi-Radio Virtual Sinks in Sensor Networks”. In: *Proceedings of the 3rd international conference on Embedded networked sensor systems* (2005), pp. 116–129.

[63] Alisha Cecil. “A Summary of Network Traffic Monitoring and Analysis Techniques”. In: *Computer Systems Analysis* (2006), pp. 4–7.

[64] Nicola Bressan et al. “The Deployment of a Smart Monitoring System using Wireless Sensor and Actuator Networks”. In: *Smart Grid Communications(SmartGridComm), 2010 First IEEE International Conference on*. IEEE. 2010, pp. 49–54.

[65] Björn Landfeldt, Pipat Sookavatana, and Aruna Seneviratne. “The case for a Hybrid Passive/Active Network Monitoring Scheme in the Wireless Internet”. In: *Proceeding of IEEE International Conference on Networks(ICON 2000)* (2000), pp. 139–143.

[66] H. Fouchal, M. Herbin, and F. Blanchard. “Centralized Energy Monitoring over Wireless Sensor Networks”. In: *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2015), pp. 788–792. ISSN: 2376-6492. DOI: 10.1109/IWCMC.2015.7289183.

[67] Anish Saini, Atul Mishra, and AK Sharma. “Distributed Network Management Architectures: A Review”. In: *International Journal of Computer Applications* 68.3 (2013).

[68] Pouria Zand, Arta Dilo, and Paul Havinga. “D-MSR: A Distributed Network Management Scheme for Real-time Monitoring and Process Control Applications in Wireless Industrial Automation”. In: *Sensors* 13.7 (2013), pp. 8239–8284.

[69] Chenxi Qiu, Haiying Shen, and Kang Chen. "An Energy-Efficient and Distributed Cooperation Mechanism for k-Coverage Hole Detection and Healing in WSNs". In: *IEEE Transactions on Mobile Computing* 17.6 (2018), pp. 1247–1259.

[70] Umar Ibrahim Minhas et al. "A WSN for Monitoring and Event Reporting in Underground Mine Environments". In: *IEEE Systems Journal* 12.1 (2018), pp. 485–496.

[71] Ferenc Nandor Janky and Pal Varga. "Time Synchronization Solution for FPGA-based Distributed Network Monitoring". In: *Infocommunications Journal* 10.1 (2018), pp. 1–9.

[72] M. Selmchenko et al. "Development of Monitoring System for End-to-End Packet Delay Measurement in Software-Defined Networks". In: *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*. Feb. 2016, pp. 667–670. DOI: 10.1109/TCSET.2016.7452147.

[73] Liang Ma et al. "Identifiability of Link Metrics based on End-to-End Path Measurements". In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. ACM. 2013, pp. 391–404.

[74] Rui Castro et al. "Network Tomography: Recent Developments". In: *Statistical Science* (2004), pp. 499–517.

[75] Liang Ma et al. "Efficient Identification of Additive Link Metrics via Network Tomography". In: *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE. 2013, pp. 581–590.

[76] Xiaojin Liu et al. "Robust Monitor Assignment with Minimum Cost for Sensor Network Tomography". In: *International Journal of Distributed Sensor Networks* 11.8 (2015), p. 512463.

[77] Yan Chen et al. "An Algebraic Approach to Practical and Scalable Overlay Network Monitoring". In: *ACM SIGCOMM Computer Communication Review*. Vol. 34. 4. ACM. 2004, pp. 55–66.

[78] Yi Gao et al. "Preferential Link Tomography: Monitor Assignment for Inferring Interesting Link Metrics". In: *2014 IEEE 22nd International Conference on Network Protocols*. IEEE. 2014, pp. 167–178.

[79] Ritesh Kumar and Jasleen Kaur. "Efficient Beacon Placement for Network Tomography". In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. ACM. 2004, pp. 181–186.

[80] Yan Chen, David Bindel, and Randy H Katz. "Tomography-based Overlay Network Monitoring". In: *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM. 2003, pp. 216–231.

[81] Shipra Agrawal, KVM Naidu, and Rajeev Rastogi. "Diagnosing Link-level Anomalies Using Passive Probes". In: *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE. 2007, pp. 1757–1765.

[82] Irit Dinur and Samuel Safra. "On the Hardness of Approximating Minimum Vertex Cover". In: *Annals of mathematics* (2005), pp. 439–485.

[83] KVM Naidu, Debmalya Panigrahi, and Rajeev Rastogi. "Detecting Anomalies Using End-to-End Path Measurements". In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 1849–1857.

[84] Paul Barford et al. "Network Performance Anomaly Detection and Localization". In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 1377–1385.

[85] Yao Zhao et al. “Towards Efficient Large-scale VPN Monitoring and Diagnosis Under Operational Constraints”. In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 531–539.

[86] Sava Stanic et al. “Active Monitoring and Alarm Management for Fault Localization in Transparent All-optical Networks”. In: *IEEE Transactions on Network and Service Management* 7.2 (2010), pp. 118–131.

[87] Liang Ma et al. “On Optimal Monitor Placement for Localizing Node Failures via Network Tomography”. In: *Performance Evaluation* 91 (2015), pp. 16–37.

[88] Emna Salhi, Samer Lahoud, and Bernard Cousin. “Joint Optimization of Monitor Location and Network Anomaly Detection”. In: *IEEE Local Computer Network Conference*. IEEE. 2010, pp. 204–207.

[89] Rakesh Ranjan Swain, Pabitra Mohan Khilar, and Sourav Kumar Bhoi. “Heterogeneous Fault Diagnosis for Wireless Sensor Networks”. In: *Ad Hoc Networks* 69 (2018), pp. 15–37.

[90] Lilia Paradis and Qi Han. “A Survey of Fault Management in Wireless Sensor Networks”. In: *Journal of Network and Systems Management* 15.2 (2007), pp. 171–190.

[91] Petcharat Suriyachai, Utz Roedig, and Andrew Scott. “A Survey of MAC Protocols for Mission-critical Applications in Wireless Sensor Networks”. In: *IEEE Communications Surveys & Tutorials* 14.2 (2011), pp. 240–264.

[92] DaintreeNetworks. *Daintree Sensor Network Analyzer*. http://www.daintree.net. Accessed: 2018-10-02. 2007.

[93] Microchip. *Zena Network Analyzer*. http://www.microchip.com//. Accessed: 2018-10-02. 2007.

[94] Carsten Buschmann et al. “Spyglass: A Wireless Sensor Network Visualizer”. In: *Acm Sigbed Review* 2.1 (2005), pp. 1–6.

[95] Xin Kuang and Jianhua Shen. “SNDS: A Distributed Monitoring and Protocol Analysis System for Wireless Sensor Network”. In: *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*. Vol. 2. IEEE. 2010, pp. 422–425.

[96] Zhonghua Zhao, Wei Huangfu, and Linmin Sun. “NSSN: A Network Monitoring and Packet Sniffing Tool for Wireless Sensor Networks”. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*. IEEE. 2012, pp. 537–542.

[97] Kgotlaetsile Mathews Modieginyane et al. “Software Defined Wireless Sensor Networks Application Opportunities for Efficient Network Management: A Survey”. In: *Computers & Electrical Engineering* 66 (2018), pp. 274–287.

[98] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke. “A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements”. In: *IEEE Access* 5 (2017), pp. 1872–1899. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2666200.

[99] Alejandro De Gante, Mohamed Aslan, and Ashraf Matrawy. “Smart Wireless Sensor Network Management based on Software-Defined Networking”. In: *2014 27th Biennial Symposium on Communications (QBSC)* (2014), pp. 71–75.

[100] Ala Al-Fuqaha et al. “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”. In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376.

[101] Nikos Bizanis and Fernando A Kuipers. “SDN and Virtualization Solutions for the Internet of Things: A Survey”. In: *IEEE Access* 4 (2016), pp. 5591–5606.

[102] Rami Halloush et al. “Hop-by-Hop Content Distribution with Network Coding in Multihop Wireless Networks”. In: *Digital Communications and Networks* 3.1 (2017), pp. 47–54.

[103] Parwinder Kaur Dhillon and Sheetal Kalra. “Secure Multi-factor Remote User Authentication Scheme for Internet of Things Environments”. In: *International Journal of Communication Systems* 30.16 (2017), e3323.

[104] Anthéa Mayzaud et al. “Using the RPL Protocol for Supporting Passive Monitoring in the Internet of Things”. In: *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE. 2016, pp. 366–374.

[105] Anuj Sehgal et al. “Management of Resource Constrained Devices in the Internet of Things”. In: *IEEE Communications Magazine* 50 (2012).

[106] Bruno Bogaz Zarpelão et al. “A Survey of Intrusion Detection in Internet of Things”. In: *Journal of Network and Computer Applications* 84 (2017), pp. 25–37.

[107] Chih-Fong Tsai et al. “Intrusion Detection by Machine Learning: A Review”. In: *Expert Systems with Applications* 36.10 (2009), pp. 11994–12000.

[108] Marwa Chafii, Faouzi Bader, and Jacques Palicot. “Enhancing Coverage in Narrow Band-IoT Using Machine Learning”. In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)* (2018), pp. 1–6.

[109] Runliang Dou and Guofang Nan. “Optimizing Sensor Network Coverage and Regional Connectivity in Industrial IoT Systems”. In: *IEEE Systems Journal* 11.3 (2015), pp. 1351–1360.

[110] Xu Lu et al. “Square Partition-based Node Scheduling Algorithm for Wireless Passive Sensor Networks”. In: *International Journal of Communication Systems* 31.8 (2018), e3531.

[111] S. Chowdhury and A. Benslimane. “Relocating Redundant Sensors in Randomly Deployed Wireless Sensor Networks”. In: *2018 IEEE Global Communications Conference (GLOBECOM)* (2018), pp. 1–6. ISSN: 2576-6813. DOI: 10.1109/GLOCOM.2018.8647974.

[112] Moslem Amiri. *Measurements of Energy Consumption and Execution Time of Different Operations on Tmote Sky Sensor Nodes*. PhD Thesis. 2010.

[113] Mehran Ashouraei et al. “A New SLA-Aware Load Balancing Method in the Cloud Using an Improved Parallel Task Scheduling Algorithm”. In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* (2018), pp. 71–76.

[114] Negar Dordaie and Nima Jafari Navimipour. “A Hybrid Particle Swarm Optimization and Hill Climbing Algorithm for Task Scheduling in the Cloud Environments”. In: *ICT Express* (2017).

[115] Bahman Keshanchi, Alireza Souri, and Nima Jafari Navimipour. “An Improved Genetic Algorithm for Task Scheduling in the Cloud Environments Using the Priority Queues: Formal Verification, Simulation, and Statistical Testing”. In: *Journal of Systems and Software* 124 (2017), pp. 1–21.

[116] Chih-fan Hsin and Mingyan Liu. “Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithms”. In: *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. ACM. 2004, pp. 433–442.

[117] Benjie Chen et al. “Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks”. In: *Wireless networks* 8.5 (2002), pp. 481–494.

[118] G. Simon et al. “Dependable k-coverage algorithms for sensor networks”. In: *2007 IEEE Instrumentation Measurement Technology Conference IMTC 2007*. May 2007, pp. 1–6. DOI: 10.1109/IMTC.2007.379153.

[119] Chih-fan Hsin and Mingyan Liu. “Randomly Duty-Cycled Wireless Sensor Networks: Dynamics of Coverage”. In: *IEEE Transactions on Wireless Communications* 5.11 (2006), pp. 3182–3192.

[120] Ali Kadhum Idrees et al. “Distributed Lifetime Coverage Optimization Protocol in Wireless Sensor Networks”. In: *The journal of supercomputing* 71.12 (2015), pp. 4578–4593.

[121] Raffaele Cerulli, R De Donato, and Andrea Raiconi. “Exact and Heuristic Methods to Maximize Network Lifetime in Wireless Sensor Networks with Adjustable Sensing Ranges”. In: *European Journal of Operational Research* 220.1 (2012), pp. 58–66.

[122] Mihaela Cardei et al. “Maximum Network Lifetime in Wireless Sensor Networks with Adjustable Sensing Ranges”. In: *WiMob’2005), IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, 2005.* Vol. 3. IEEE. 2005, pp. 438–445.

[123] Yousif Elhadi Elsideeg Ahmed. “Modeling, Scheduling and Optimization of Wireless Sensor Networks lifetime”. PhD thesis. 2016.

[124] Arianna Alfieri et al. “Maximizing System Lifetime in Wireless Sensor Networks”. In: *European Journal of Operational Research* 181.1 (2007), pp. 390–402.

[125] Fabian Castaño et al. “A Column Generation Approach to Extend Lifetime in Wireless Sensor Networks with Coverage and Connectivity Constraints”. In: *Computers & Operations Research* 52 (2014), pp. 220–230.

[126] Junchao Ma et al. “Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks”. In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 630–638.

[127] Cristanel Razafimandimby et al. “Efficient Bayesian Communication Approach for Smart Agriculture Applications”. In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE. 2017, pp. 1–5.

[128] Shifeng Fang et al. “An Integrated System for Regional Environmental Monitoring and Management based on Internet of Things”. In: *IEEE Transactions on Industrial Informatics* 10.2 (2014), pp. 1596–1605.

[129] Dominique Barthel et al. “Routing Metrics used for Path Calculation in Low-Power and Lossy Networks”. In: (2012).

[130] Nicolas Bourgeois et al. “Fast Algorithms for Min Independent Dominating Set”. In: *Discrete Applied Mathematics* 161.4-5 (2013), pp. 558–572.

[131] Matti Åstrand and Jukka Suomela. “Fast Distributed Approximation Algorithms for Vertex Cover and Set Cover in Anonymous Networks”. In: *Proceedings of the Twenty-second Annual ACM Symposium on Parallelism in Algorithms and Architectures* (2010), pp. 294–302.

[132] Gilbert Laporte. “The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms”. In: *European Journal of Operational Research* 59.2 (1992), pp. 231–247.

[133] S Balaji, V Swaminathan, and K Kannan. "Optimization of Unweighted Minimum Vertex Cover". In: *World Academy of Science, Engineering and Technology* 43 (2010), pp. 716–729.

[134] Byron Boots, Geoffrey J Gordon, and Sajid M Siddiqi. "A Constraint Generation Approach to Learning Stable Linear Dynamical Systems". In: *Advances in Neural Information Processing Systems*. 2008, pp. 1329–1336.

[135] David W Pentico. "Assignment Problems: A Golden Anniversary Survey". In: *European Journal of Operational Research* 176.2 (2007), pp. 774–793.

[136] Fumei Lam and Alantha Newman. "Traveling Salesman Path Problems". In: *Mathematical Programming* 113.1 (2008), pp. 39–59.

[137] Debasish Roy and G Visweswara Rao. *Stochastic Dynamics, Filtering and Optimization*. Cambridge University Press, 2017.

[138] Kaveh Khalili-Damghani, Madjid Tavana, and Soheil Sadi-Nezhad. "An Integrated Multi-Objective Framework for Solving Multi-Period Project Selection Problems". In: *Applied Mathematics and Computation* 219.6 (2012), pp. 3122–3138.

[139] George Mavrotas. "Effective Implementation of the ε-Constraint Method in Multi-Objective Mathematical Programming Problems". In: *Applied Mathematics and Computation* 213.2 (2009), pp. 455–465.

[140] Fatme El-Moukaddem, Eric Torng, and Guoliang Xing. "Maximizing Network Topology Lifetime Using Mobile Node Rotation". In: *IEEE Transactions on Parallel and Distributed Systems* 26.7 (2014), pp. 1958–1970.

[141] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th Paschos. "Approximation of Max Independent Set, Min Vertex Cover and Related Problems by Moderately Exponential Algorithms". In: *Discrete Applied Mathematics* 159.17 (2011), pp. 1954–1970.

[142] David B. Shmoys and Éva Tardos. "An Approximation Algorithm for the Generalized Assignment Problem". In: *Mathematical programming* 62.1-3 (1993), pp. 461–474.

[143] Miklós Molnár, Gyula Simon, and László Gönczy. "Quasi–Optimal Scheduling Algorithm for Area Coverage in Multi–Functional Sensor Networks". In: *International Journal of Ad Hoc and Ubiquitous Computing* 14.2 (2013), pp. 109–122.

[144] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. "Low Power Wide Area Networks: An Overview". In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 855–873.

[145] Katrin Casel et al. "Extension of Vertex Cover and Independent Set in some Classes of Graphs and Generalizations". In: *arXiv preprint arXiv:1810.04629* (2018).

[146] Jochen Alber et al. "Fixed Parameter Algorithms for Dominating Set and Related Problems on Planar Graphs". In: *Algorithmica* 33.4 (2002), pp. 461–493.

[147] Rolf Niedermeier and Peter Rossmanith. "On Efficient Fixed-parameter Algorithms for Weighted Vertex Cover". In: *Journal of Algorithms* 47.2 (2003), pp. 63–77.

[148] Maw-Shang Chang et al. "Fixed-parameter Algorithms for Vertex Cover P3". In: *Discrete Optimization* 19 (2016), pp. 12–22.

[149] Yan Y Liu and Shaowen Wang. "A Scalable Parallel Genetic Algorithm for the Generalized Assignment Problem". In: *Parallel Computing* 46 (2015), pp. 98–119.

[150] Hannes Moser. "Exact Algorithms for Generalizations of Vertex Cover". In: *Master's Thesis, Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität Jena* (2005).

[151] Magnus M Halldorsson. *Algorithm Theory-SWAT 2000: 7th Scandinavian Workshop on Algorithm Theory Bergen, Norway, July 5-7, 2000 Proceedings*. Springer, 2003.

[152] Neil Robertson and Paul D. Seymour. "Graph Minors. II. Algorithmic Aspects of Tree-width". In: *Journal of Algorithms* 7.3 (1986), pp. 309–322.

[153] Hans L Bodlaender and Arie MCA Koster. "Combinatorial Optimization on Graphs of Bounded Treewidth". In: *The Computer Journal* 51.3 (2008), pp. 255–269.

[154] Shiva Kintali and Sinziana Munteanu. "Computing Bounded Path Decompositions in Logspace". In: *Electronic Colloquium on Computational Complexity (ECCC)* (2012).

[155] Stephen P Carroll. "Domain Specific Language for Dynamic Programming on Nice Tree Decompositions". PhD thesis. Ohio University, 2013.

[156] Tim Winter et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. Tech. rep. 2012.

[157] Andrew Adamatzky. *Advances in Unconventional Computing: Volume 1: Theory*. Vol. 22. Springer, 2016.

[158] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[159] Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018. URL: http://www.gurobi.com.

[160] Iain Dunning, Joey Huchette, and Miles Lubin. "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Review* 59.2 (2017), pp. 295–320.

[161] Zhijiang Chen et al. "A Cloud Computing based Network Monitoring and Threat Detection System for Critical Infrastructures". In: *Big Data Research* 3 (2016), pp. 10–23.

[162] Gábor Bergmann et al. "Optimal Period Length for the CGS Sensor Network Scheduling Algorithm". In: *2010 Sixth International Conference on Networking and Services* (2010), pp. 192–199.

[163] Tmote Sky Datasheet. "Moteiv Corporation". In: *Saatavissa (viitattu 1.10. 2017): http://www. eecs. harvard. edu/~ konrad/projects/shimmer/references/tmote-sky-datasheet. pdf* (2006).

[164] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. "The Resolution Complexity of Independent Sets and Vertex Covers in Random Graphs". In: *Computational Complexity* 16.3 (2007), pp. 245–297.

[165] Olivier Ruas. "Neighbor-of-Neighbor Routing In Small-World Networks With Power-Law Degree". PhD thesis. INRIA-IRISA Rennes Bretagne Atlantique, équipe ASAP, 2013.

[166] Pierre Fraigniaud and George Giakkoupis. "Greedy Routing in Small-World Networks with Power-Law Degrees". In: *Distributed computing* 27.4 (2014), pp. 231–253.

[167] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. "Know Thy Neighbor's Neighbor: The Power of Lookahead in Randomized P2P Networks". In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM. 2004, pp. 54–63.