A Survey on Design Challenges of Scheduling Algorithms for Wireless Networks

Mojtaba Malekpourshahraki

BITS Networked Systems Lab, Department of Computer Science
University of Illinois at Chicago - Chicago, IL, USA
E-mail: mmalek3@uic.edu

Christopher Desiniotis

Internet of Things Research Lab, Department of Computer Engineering ,
Santa Clara University - Santa Clara, CA, USA
E-mail: cdesiniotis@scu.edu

Marjan Radi

Western Digital Research
Milpitas, CA, USA
E-mail: radi@ieee.org

Behnam Dezfouli

Internet of Things Research Lab, Department of Computer Engineering ,
Santa Clara University - Santa Clara, CA, USA
E-mail: bdezfouli@scu.edu

Abstract: Recent advances in wireless technologies open up new avenues toward many newer applications; however, existing wireless networks are not efficient enough to satisfy some new applications' requirements such as low energy consumption, high throughput, and low end-to-end delay. Scheduling algorithms are promising for performance improvement by addressing the limitations of standard networking protocols. Due to the widespread deployments and applications of wireless networks, a significant number of scheduling algorithms has been proposed to improve network performance. In this paper, we provide a systematic comparison among proposed scheduling algorithm with an exhaustive, network-independent framework to ease the comparison among different scheduler design and highlight important challenges in the term of algorithmic parameters, computational order, accuracy, and overhead. We also study a set of schedulers in two main categories: *low-power*, and *high-throughput*, concerning our framework to extract the weakness and strength of each proposed method compared to other scheduling algorithms.

Keywords: Scheduling algorithms; low-power wireless networks; High-throughput networks; MAC.

Biographical notes: Mojtaba Malekpourshahraki is a Ph.D. candidate in BITS Networked Systems Lab, at the University of Illinois at Chicago (UIC). He

is currently focusing on multi-tenant datacenter networks and low latency applications for data centers using programmable switches. Before joining UIC, he completed his Master from Iran University of Science and Technology, focusing on wireless mesh networks. His research interests are wireless networks, scheduling algorithms, datacenter networks, programmable switches, network processors, RDMA.

Christopher Desiniotis received his B.S. and is currently pursuing his M.S. in Computer Science and Engineering at Santa Clara University. His research interests include Internet of Things, virtualization, and edge computing.

Marjan Radi is an R&D Technologist in NVM Systems Architecture research group at Western Digital. Prior to joining Western Digital, She was a post-doc at the University of Iowa. She hold Bachelor of Science in Computer Software Engineering, Master of Science in Computer Software Engineering, and PhD of Computer Science. She is currently focusing on consistent and coherent distributed shared memory systems, fault-tolerant in-network caching and storage, distributed cloud data centers, and SDN enabled data centers.

Behnam Dezfouli is an Assistant Professor at the Department of Computer Science and Engineering, as well as the Director of SCU IoT Research Lab, at Santa Clara University (SCU), USA. Before joining SCU, he was a Postdoctoral Research Scientist and a Visiting Assistant Professor at the University of Iowa, US, during 2015 to 2016. Behnam Dezfouli served as a Postdoctoral Research Fellow and a Research Fellow at University Technology Malaysia, Malaysia, and Institute for Infocomm Research, Singapore, during 2014 and 2012, respectively. He received his PhD in Computer Science from University Technology Malaysia in 2014.

1 Introduction

Compared to wired networks, wireless technologies offer ease of deployment, mobility, and low infrastructure cost. These features motivate a major deployment of wireless networks in residential, and commercial environments. According to a study in Statista (2020), there are a total number of 21.5 billion devices connected to the internet while this number was 7.3 billion in 2014 and 7.9 billion in 2015 Cisco (2016). In addition, another study in Juniper-Research (2020) predicts that the number of IoT devices will reach 36.8 billion in 2025, which shows an overall growth rate of 107% compared to the number of devices in 2021. The total number of connected mobile devices, in conjunction with its rapid increase, is indicative of the growing ubiquity of wireless technologies.

Despite the potentials of wireless networks and the rapid growth of applications in this domain, they present unique challenges when compared to wired networks. The most prominent problem is shared medium, which adversely affects transmission reliability due to co-channel interference. Energy is another challenge in wireless networks. Unlike wired networks, the energy consumption of wireless transceivers is higher compared to other onboard components such as processors and sensors. This problem is critical in battery-powered wireless devices, such as wireless sensor nodes and smartphones. For instance, in a battery-powered multi-hop sensor network, energy depletion may result in a network partitioning, and consequently, losing ad-hoc functionality (Rao & Kamila (2021), Onuekwusi et al. (2020)).

Medium Access Control (MAC) protocols have been proposed to arbitrate medium access and control the operation of RF transceivers, and save power. MAC protocols falls into two main categories: Contention-free and Contention-based protocols (Jin et al. (2014), Lin, Shroff & Srikant (2006), Jones et al. (2020)). Nodes in contention-free Mac protocols do not compete with each other to gain the access to the channel and the channel is shared among them with some predefined policy. Time-division multiple access (TDMA) and Frequency-division multiple access (FDMA) are examples of Contention-free mac protocols in which the medium is shared based on time and frequency respectively (Pan & Liew (2020)).

State-of-the-art MAC protocols are designed to offer compatibility and simplicity; however, a simple MAC protocol cannot address the unique challenges of wireless networking. For instance, contention-based protocols perform poorly in dense networks due to the bandwidth waste caused by interference and hidden terminal problem (Pathak & Dutta (2011)). Contention-free protocols, on the other hand, require a synchronization algorithm to synchronize global information (e.g., time in TDMA or frequency in FDMA) across the network. Synchronization in a wireless network is not a trivial problem because of the dynamic changes in parameters such as size and mobility (Skiadopoulos et al. (2019), Tian et al. (2019)).

Scheduling algorithms – refereed to as schedulers – can address performance limitations of MAC protocols by offering fine-grain medium access policies. The scheduling element can vary based on the goal of the schedulers. *Link* scheduling is the most well-known type of schedulers to handle channel access and interference by assigning link activation orders (Rhee et al. (2006), Enqing et al. (2014), Zhou et al. (2012), Ahmad et al. (2014), Bhatia & Hansdah (2014)). An activation order is a deterministic transmission time for a set of links to prevent co-channel interference from other activated links or nodes (Malekpourshahraki et al. (2016), Zhou et al. (2012), Fateh & Govindarasu (2015), Nasser et al. (2013), Nabli et al. (2014), Gabale et al. (2011)). Similarly, *node* scheduling algorithms determine the activation order of the nodes. A node scheduler deactivates (power off or power save mood) a node to save the energy when the absence of a node does not cause a network partitioning (Hare et al. (2014), Zhang et al. (2014)).

Even though link and node schedulers improve throughput, they cannot address all networking concerns such as long queuing delay. Link and node schedulers cannot provide any control over queuing and packet transmission. Thus, other schedulers are required to offer a full QoS guarantee. For example, assume an application generates delay sensitive flows, and packets belonging to these flows need to be prioritized over regular packets. A common approach to implement delay sensitive transmission is *packet* scheduling in a priority queue (Malekpourshahraki et al. (2019*a*), MalekpourShahraki et al. (2019*b*)). In addition to deadline-aware transmissions, this type of scheduler enhances bandwidth utilization and fairness among all active flows (So-In et al. (2009), Ito et al. (2002), Kaur & Singh (2011), Lera et al. (2007), Stolyar & Ramanan (2001), Wang et al. (2008)).

Another type of scheduler is *task* scheduling, in which nodes schedule either network processing tasks or transmission tasks in a more efficient way. Some examples of network processing tasks are packet generation, forwarding, and data aggregation. The execution order of these tasks can dramatically affect network performance and energy consumption. If all of these tasks finish in a shorter time, nodes can aggregate a longer sleep interval and set radios off for lower energy consumption. For example, wireless sensor nodes may need to sense and aggregate data before transmission to a sink node. By coordinating the ordering and execution time of these tasks, a task scheduler extends processor idle periods or reduces the number of the transceiver's on and off cycles to enhance energy efficiency (Fateh & Govindarasu (2015), Hare et al. (2014), Zhang et al. (2014)).

Designing schedulers for wireless networks is particularly challenging due to the impact of dynamic factors such as mobility, interference, congestion, and energy level. Although previous surveys such as Pathak & Dutta (2011), Amjad et al. (2019), Kaur & Kumar (2019) address some of these challenges in detail, there is a gap in the literature for a comprehensive framework to cover all of these problems in a modular fashion. In other words, we believe a framework is needed to decouple all of the elements and factors required in a scheduler to ease the design and comparison of different schedulers. Towards an organized structure to study these algorithms and the factors affecting them, we propose a comprehensive framework that covers scheduler types and different network settings. In addition to offering design guidelines, this framework helps researchers compare their designs to existing approaches. We then review the main categories of schedulers and existing proposed solutions in a variety of networks including 802.15.4, Bluetooth, WiFi (IEEE 802.11), mesh (IEEE 802.11s) under two main categories: low-power and high-throughput wireless networks. Within the context of the proposed framework, we compare existing work based on performance, computational order, and algorithm design, as well as other influential factors of designing scheduling algorithms.

This paper is organized as follows: In Section 2, we propose our scheduling framework. In Section 3, we briefly summarise various approaches in low-power wireless networks and high-throughout wireless networks. We draw a comparison among all of the algorithms in Section 4 along with a discussion about opportunities and future work. Lastly, we conclude our paper in Section 5.

2 A General Scheduling Framework

Comparing scheduling algorithms is challenging since they are diverse in network types, goals, and settings. One way to approach this complexity is to break down the scheduling challenges and compare them separately. This underlines the need for a comprehensive framework that can cover all of the target networks and scheduling challenges. In this section, we propose a framework to provide a guideline for tackling scheduling challenges and illustrate the building blocks of all scheduling algorithms to ease comparison among existing approaches. Figure 1 shows our framework with five components. Each component in this figure demonstrates an essential part of a scheduling algorithm. We describe each component in the following sections.

2.1 Information Collection

This component is responsible for collecting and classifying required information for scheduling. In distributed schedulers, each node needs to collect its required information individually, which could be as simple as local information to the entire network. In centralized scheduling algorithms, only a single node collects and maintains a global view of the scheduling information. Some examples of nodes that may run central schedulers are: a cluster head (Pawar et al. (2012), Ahmad et al. (2014)), a gateway node (Dezfouli et al. (2017), Behnam Dezfouli (2016), Wei et al. (2005)) or an external proxy device (Qiu et al. (2011)). Although information collection from the entire network can increase the efficacy and accuracy of the schedules, the overhead of packet exchange is usually prohibitive,

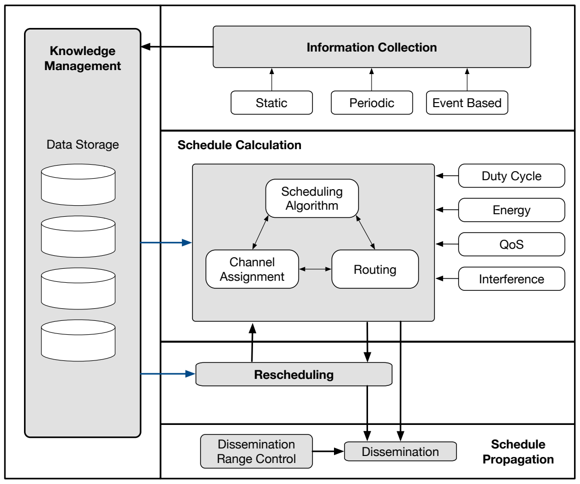

Figure 1 Comprehensive framework for scheduling algorithms in wireless networks.

especially in large-scale, multi-hop networks. Information collection is performed using one of the following approaches:

Static: Static information gathering is the simplest information collection approach, which the network manager hard codes the required information. For example, in a long-distance static wireless mesh network, nodes are manually deployed in a geographical area, and the neighborhood connectivity of each node is predetermined (Gabale et al. (2011), Narlikar et al. (2010), Nabli et al. (2014), Panigrahi & Raman (2009), Ramachandran et al. (2006), Dutta et al. (2008), Yu et al. (2008)).

Periodic: A scheduling node gathers required information by sending periodic control packets to inform other nodes about changes in network status. Beacon packets in mesh and ad-hoc networks are examples of the events that may carry information regarding a change in network status (Malekpourshahraki et al. (2016), Ahmad et al. (2014), Bhatia & Hansdah (2014)).

Event-Based: When a node detects a change in a parameter of interest, it generates an event packet to inform other nodes about its observation. For example, an event message may carry the list of unreachable nodes to show that the existing schedule is not valid anymore (Hedayati & Rubin (2012), Rhee et al. (2006), Enqing et al. (2014)).

2.2 Knowledge Management

The main responsibility of this component is to maintain the required knowledge for the scheduling algorithm in data structures that are compatible with scheduling algorithms. For instance, the topology of a mesh network can be stored in a graph or tree data structure showing reachability instead of coordination values. Required knowledge falls into three categories based on the distance of the source of the knowledge from the scheduling node, as follows.

Intra-Node: In this category, nodes use their local information to calculate a proper schedule. An example is a priority-based packet scheduling policy applied before outgoing packet transmission. In this case, each node can rely on its local operational information to perform scheduling (Semeria (2001)).

N-hop Neighbors: In this category, the scheduler obtains network knowledge from nodes that are N hops far from the scheduling node. In the majority of cases, the value of N is small due to the difficult information gathering from remote nodes. The first hop neighbors' knowledge (N=1) is commonly used in the literature since one-hop neighbors are in the interference range of the scheduling node, and they hold important scheduling knowledge. For example, a node may schedule packets at a different time than neighbors are using the channel to avoid collisions. To gain this knowledge, a node can passively listen to packets from neighboring nodes, or it actively exchanges control packets to update its neighborhood table (Rhee et al. (2006), Enqing et al. (2014)).

Network Wide: Obtaining global network knowledge enables the computation of an optimal schedule – despite being an NP-hard problem, the persistence of global knowledge is usually short, especially in mobile networks. This type of knowledge management is mostly used in centralized schedulers to ensure end-to-end timeliness and reliability guarantees (Chang et al. (2012), Takita (2013)).

2.3 Schedule Calculation

The schedule calculation component has one or more scheduling algorithms that use the provided knowledge as inputs and generate a schedule for a specific set of nodes in distributed scheduling or all of the nodes in centralized scheduling. Regardless of the scheduler type, scheduling algorithms utilize nodes' resources to execute the scheduling algorithm. We compare the time that these algorithms run in the term of computational order. Ideally, algorithms must have a smaller computational order such as logarithmic or linear rather than polynomial or exponential.

Designing a schedule calculation algorithm rises the following fundamental questions: (i) What to schedule? (ii) What are the problems that the scheduler should solve? We discuss these two questions in the following two sections.

2.3.1 Scheduling Types

We categories scheduling algorithms into three main categories based on what they schedule. Figure 2 shows these types and mechanisms that are used in each category to implement schedulers. We review these types of algorithms as follows:

Link Scheduling. Link Scheduling is the most frequently used type of scheduling algorithm in the literature. The state-of-the-art approaches focus on tackling interference by assigning an appropriate activation slot to link. Interference happens when two links transmit at the same time on the same channel. Hence, the scheduler assigns different activation time-slots to the interfering links to prevent concurrent transmission and interference.

Node Scheduling. In node scheduling, the scheduler assigns different activation slots to the nodes in the transmission range of each other to prevent co-channel interference.

Packet Scheduling. In this type of scheduling algorithm, the scheduler finds an efficient packet transmission order based on the network policies. For instance, in the case where a network has different applications with different priorities, a packet scheduler dequeues packets belonging to the applications with higher priorities. A common approach to implement packet scheduling is using multiple queues, where each queue represents a traffic class. Scheduler dequeues the packets belonging to the traffics with higher priority earlier than others. Kaur & Singh (2011), Ito et al. (2002), Lera et al. (2007), Stolyar & Ramanan (2001), Wang et al. (2008) are examples of packet scheduling algorithms.

Task and Sleep Scheduling. Network nodes have a simple operating system that schedules all of the nodes' tasks such as sensing, aggregation, and transmission. Authors in Zhu et al. (2012) proved that the order of running these tasks could dramatically affect

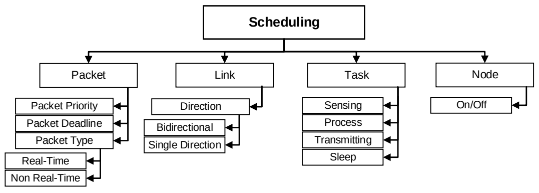

Figure 2 Four main types of schedulers in wireless networks.

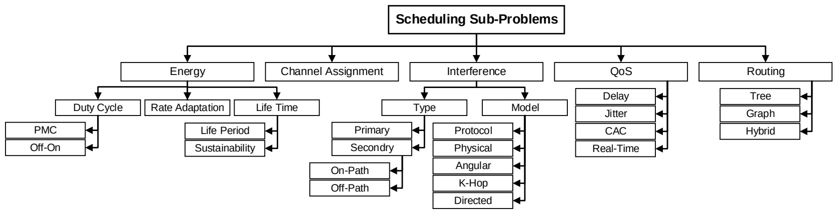

Figure 3 Scheduling sub-Problems. A scheduler can address one or more sub-problems in an algorithm.

network performance in the term of energy consumption. For example, a sensor node may have three different tasks: sensing the environment, processing incoming packets for aggregation, and data transmission. In this case, an ideal task scheduler could find the best task execution order and time to reduce the time that the radio must be on and optimize energy consumption. If nodes schedule all of their tasks at the end of a time interval (e.g., beacon interval), they can enter sleep mode for the rest of the interval and reduces energy consumption by removing idle listening.

2.3.2 Scheduling Sub-Problems

One Scheduler may address one or more challenges at the same time. These challenges tightly depend on each other, and solving one of them may show some improvement in other challenges. Figure 3 shows all of these sub-problems and techniques that are used in literature to address the problem. In this section, we review all these challenges in five different categories as follow:

Interference. Schedulers mitigate or eliminate interference among nodes and links to achieve higher throughput. There are two different types of interference: *primary* and *secondary*. Primary interference refers to the limitation imposed by half-duplex transceivers, which indicates that a node cannot send and receive at the same time. On the other hand, secondary interference refers to the interference among nodes during concurrent transmissions. There are two different types of secondary interference: *on-path* interference and *off-path* interference. In the on-path interference, packets from a particular end-to-end connection interfere with packets from the same connection, while in the off-path interference, packets from a particular connection interfere with packets from other connections.

Another way to categorize interference is to use the interference detection model. The interference model mainly describes how a node detects that interference happened or link or node is prone to interference.

- **Protocol:** In this interference model, nodes are considered to be in the interference range of each other if their distance is less than a certain threshold.

- **Physical:** Physical model detects interference with the exact signal-to-noise ratio at the physical layer. If the ratio is lower than a threshold, interference is detected.

- **Angular:** This model is proposed in Panigrahi & Raman (2009) to increase network throughput. In this model, interference is proportional to the angle between links. If the angle of two links is smaller than a threshold, links are considered to be interfering with each other.

- **k-hop:** This model considers any concurrent transmission within a radius of k-hop neighbors as an interference.

Among all of the interference models, the physical interference model can determine the interfering links accurately because nodes calculate the Signal to Interference Noise Ratio (SINR) in their physical layer.

Channel Assignment. Link and node schedulers avoid co-channel interference by controlling transmission time. Assigning a different channel for concurrent transmissions can also eliminate co-channel interference. Channel assignment provides a better throughput rather than link scheduling since nodes are allowed to transmit concurrently and it increases the overall throughput of the network (Gupta & Dhurandher (2020), Tian & Yoshihiro (2020)).

Quality of Service. Quality of service is essential to satisfy application requirements. If an application needs a very low delay, the packet scheduler may priorities packets from an application to others to ensure that the application can achieve the destination with the desirable delay. For example, video streaming can tolerate delay and jitter with buffering packets, but mission-critical applications such as process automation cannot tolerate long delays in their control loop. A packet scheduler could strictly prioritize packets belonging to the mission-critical applications over other applications.

Similar to application QoS, schedulers may address end-user quality of experience (QoE). Quality of experience is closely tight to the QoS and it is a concept to measure and model the end-user experience of using the system (Sousa et al. (2020)). Even though it is not a new concept in networking, it has been the main focus in state-of-the-art resource scheduling in the wireless network (Rao & Hency (2021), Nosheen & Khan (2020)).

Energy. For wireless devices that use battery or renewable energy (e.g., solar, wind, mechanical, or thermal sources) as their power supply, controlling energy consumption is important to guarantee long-term network functionality and prevent frequent battery replacement (Sah & Amgoth (2020)). One solution to tackle high power consummation is to switch the network card to Power Save Mode (PSM). In this mode, schedulers determine time intervals that a network card does not need to receive or send data (e.g., because other nodes are sending data) Malekpourshahraki et al. (2016). A scheduler may periodically choose a group of nodes to change their mode to PSM for a short slot of time. The remaining nodes must form a connected graph that can handle the overall network traffic. Another way to control energy consumption is to limit the sending rate. When nodes sending in a higher data rate, more energy is required. Thus, schedulers control the sending rate to reduce energy consummation.

2.4 Schedule Distribution

Schedulers need to distribute the calculated schedule to other nodes. In centralized algorithms, a scheduling algorithm (e.g., base station or sink node) transmits a packet to

Table 1 Summary of notations used in the tables

Abbreviation	Definition	Abbreviation	Definition
Pr	Protocol	e	Number of edges
Ph	Physical	D	Length of the tree
Fa	Fading	δ	Maximum size of a two-hop neighborhood
nH	N-hop	λ_t	Number of available discrete frequencies
Di	Distributed	λ_m	Number of available discrete modulation levels
Ce	Centralized	s	Slots in the scheduling interval
Cd	DCMA	c	Number of available channels
Td	TDMA	d	deadline in terms of number of slots
T	Tree	p	Spatial reuse
G	Graph	m	Number of available data rates
v	Number of nodes	L	Number of active links

other nodes and provides them with the new schedule. In distributed algorithms, nodes calculate the schedules separately and follow their local schedules. In some cases, distributed schedulers need to send their local scheduling information over at least one hop to provide up-to-date information for other nodes to update their schedules accordingly.

2.5 Rescheduling

Rescheduling methods fall into two main categories: periodic, and trigger-based. In the periodic method, nodes execute a rescheduling algorithm, even if network status has not changed since the last schedule and input are the same. Some examples of periodic method are Hedayati & Rubin (2012), Chang et al. (2012), Rhee et al. (2006), Zhang et al. (2014). In the trigger-based method, nodes execute rescheduling algorithms when they detect a new change in topology or any other parameters of interest (Bhatia & Hansdah (2014), Zhu et al. (2012)).

3 Scheduling Algorithms

In this section, we provide an in dept study of a set of scheduling algorithm with regard to the algorithm design. For simplicity, we divide the algorithm into two different category: low-power and high throughput wireless networks. We provide the details of these approaches in tables 2 and 3. Table 1 shows the abbreviations that we use in both tables.

3.1 Low-Power Wireless Networks

This category broadly includes wireless sensor networks and underwater wireless networks. Nodes in these two subcategories are energy sensitive, and they mostly rely on battery power or renewable energy. Table 2 shows the approaches in this category with the detailed comparison among their characteristic.

DRAND is a distributed TDMA Scheduling algorithm in Rhee et al. (2006) for wireless sensor and ad-hoc networks. DRAND proposes a negotiation procedure between nodes to find the best slot allocation order. This procedure does not require synchronization in the slot allocation phase, even though it uses TDMA slots and requires synchronization for the data transmission.

DRAND benefits from a state machine to track the states of the nodes. Four states are defined to control slot allocation: IDLE, REQUEST, GRANT, and RELEASE. Each node starts in IDLE state. A successful slot allocation scenario is shown in Figure 4. During the IDLE state, nodes run a lottery to start the slot allocation process. Let node *A* denotes the winner of the lottery. In Figure 4.A, node *A* changes its state to REQUEST and broadcasts

Table 2 Summary of the Scheduling Algorithms for Low-Power Wireless Networks

	Multi-Channel	Multi-Radio	Interference Model	Call Admission Control	Multi-Gateway	Scheduler Control	Channel Type	Topology	Application	Antenna Type	Computational Order	Technique
DRAND	N	N	Pr	N	N	D	T/C	G	General	Omni	$O(\delta)$	Message passing
IEJSH	N	N	Pr	N	N	C	T	T	Sensing	Omni	$O(vlogv + v^2 + v^2(\lambda_t + \lambda_m))$	LP, Heuristic
DLSPHY	N	N	Ph	N	N	D	T	G	Monitoring	Omni	Polynomial	Independent sets, Shifting sets
DSC	N	N	Pr	N	N	D	T	G	General	Omni	N/A	Message passing
M-GCF	N	N	N/A	N	N	D	T	C	General	Omni	N/A	Schedule separately
DOES	N	N	N/A	Y	N/A	D	G	N/A	General	Omni	Polynomial	Task decomposition and composition
PFSA	N	N	N/A	N	N	D	N/A	G	Tracking	Omni	N/A	Calculate probability of target location
AnE	N	N	N/A	N	N	D	N/A	G	Tracking	Omni	N/A	Adjusting sampling interval
TDCS	N	N	N/A	N	N	C	T/C	C	General	Omni	$O(v.e)$	Reducing dutycycle and LP
									Non-realtime			scheduling
SCoRe	Y	N	N/A	N	N	C	T	T	Realtime	Omni	N/A	Recursive scheduling
DDLS	N	N	Fa	N	N	D/C	N/A	G	Realtime	Omni	$O(ln(v))$	Sorting/Greedy
A2C	N	N	N/A	N	N	C	N/A	T	Realtime	Omni	N/A	Reinforcement learning

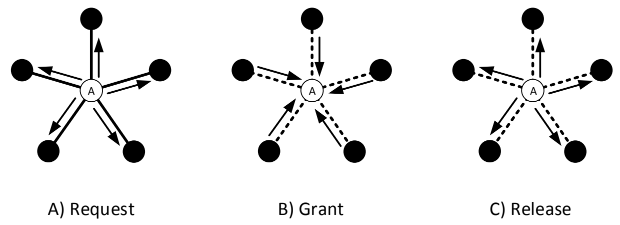

Figure 4 A successful scheduling scenario in DRAND. (A) node A sends a *request* packet to inform its neighboring nodes that it won the lottery. (B) Node A's neighbor sends a *grant* packet for node A. Node A waits for all grant packets from all of its one-hop neighbors. (C) Node A selects a slot and sends a *release* message to inform its neighbors about the assigned slot.

a *request* packet, which indicates that node *A* is ready for slot allocation. Each neighbor (e.g., *B*) receiving a *request* packet, changes its status to GRANT and sends a *grant* packet. This packet contains time slot information for node *B*. Node *A* chooses a new slot upon receiving the *grant* packets from all of its neighbors (Figure 4.B). After receiving all *grant* packets, node *A* has all the required information about its two-hop neighbors. Thus, it changes its status to RELEASE and allocates a slot that has not been allocated. In this case, all the available slots have been allocated, and the least frequently used slot by the two-hop neighbors is selected (Figure 4.C). If node *B* receives a request packet from node *A* while it is in the GRANT state for others, node *B* sends a *reject* packet. Consequently, node *A* fails in the slot allocation procedure and broadcasts a *failed* packet.

DRAND suffers from high energy consumption and high schedule convergence duration, due to high packet exchange overhead and several state transitions For instance, in a network with an average of eight neighbors for each node, DRAND needs six seconds to complete slot allocation. Similarly, schedule propagation in DRAND takes approximately up to one minute, which poses a considerable uncertainty in rescheduling mobile networks. As a result, DRAND slightly supports mobility, but it is not suitable for highly mobile networks.

SCoRe in Park & Paek (2021) is an on-demand scheduling of command and responses on multi-hop wireless network. This approach calculates the amount of time that is required by nodes to distribute commands and collect the results from all sensors. SCoRe assigns a schedule to all nodes to prevent collision in synchronous responses from IoT nodes.

SCoRe prevents collision by dedicating time slots to nodes in both command and response paths. Upon issuing a command, SCoRe recursively assigns a time slot for the child node level by level. When a node receives a command, it may generate a response or retransmit the command to their child nodes (next level in Breath first search - BFS). SCoRe improves data collection reliability by reducing the collision and increase the number of successfully delivered packets.

SCoRe is designed for low-power wireless networks and it was implemented in low-power embedded devices. Evaluations show that SCoRe achieves a 99% packet reception ratio (PRR), while this factor in basic flooding algorithm and Trickle in Levis et al. (2004) are 50% and 90% respectively. Similarly, SCoRe keeps the command-to-response latency as low as 1.5-2s for different tested scenarios and retransmissions is close to zero.

DDLS in Yu et al. (2020) proposes a centralized and a distributed link scheduling algorithm to maximize the number of concurrently scheduled links. These algorithms work based on the link distance of the nodes introduced in GHW in Goussevskaia et al. (2013). GHW algorithm sorts and traverses all links in ascending order of length to find a link that the effect of the link on selected links is smaller than a threshold. The link's effect is calculated by the path-loss exponent and SINR threshold. Since SINR cannot be determined without error due to the effect of fading, DDLS uses the Rayleigh-fading model. Authors reduce the Rayleigh-fading model to a simpler SINR model by constructing the connection between the acceptable failure probability and the needed received SINR. Distance-based Link Scheduling (DLS) is derived from GHW with two key observations to reduce the time complexity of the algorithm by removing the sort from GHW. The authors also proposed a distributed version of DLS (DDLS) that operates based on DLS to find the best feasible schedule in different iterations.

The evaluation shows that DDLS shows a better performance for all SINR thresholds. DDLS results in 22.8% more in successfully scheduled links compared to GHW Goussevskaia et al. (2013) and 50% more successfully scheduled links compared to LDP Qiu & Shen (2017).

A2C in Xu et al. (2020) is a radio resource scheduler based on deep reinforcement learning (DRL) to improve scheduling in down-link over the cellular network. A2C optimizes throughput, fairness, and packet drop for all user equipment (UE) using a multi-objective optimization problem concerning limited buffer size and link adaptation.

In the conventional transmission model, the cellular scheduler assigns a separate transmission queue for each UE. At each slot, the cellular scheduler picks one transmission queue and transmits the head of the line packet (Aka, FIFO) to the UE. This model can address throughput and fairness as two critical factors in the scheduling decision. In addition to these two, A2C also considers the packet drop for the case where the transmission buffer is full. Authors formulate a Pareto optimization to find the best trade-off between all these three parameters. Since solving this optimization problem is difficult due to the large solution space, the authors proposed two genie-aided heuristic methods to reduce the exploring efforts. This two heuristic, however, requires some inputs that are not available real-time such as packet arrival time. To solve this, a modified version of Advantage actor-critic A2C is proposed to solve the optimization in real-time. Details about the deep learning A2C is beyond the scope of this paper.

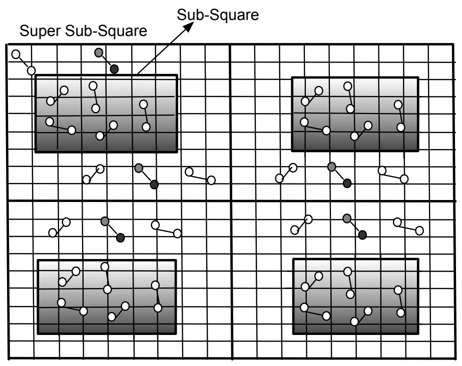

Figure 5 DLSPHY partitions the network into several smaller subsquares. Each super-subsquare contains several smaller subsquares. Nodes in each subsquare can transmit their packets without interference. Zhou et al. (2012).

Evaluations show that the modified A2C can perform as well as two genie-aided methods without using any unrealistic assumptions. In an evaluation with 56 UE, A2C outperform the baseline method in throughput, fairness, and packet drop rate, by 10.54%, 0.3%, 7.64% respectively.

DLSPHY in Zhou et al. (2012) aims to maximize network throughput using the physical interference model. The scheduler creates several non-interfering sets of links such that links belonging to each set can communicate concurrently without interfering. DLSPHY models the network as a grid to form these sets. Each set is the links in a square of the grid. Figure 5 shows an example of a grid. Two types of squares are drawn in this picture *Super-subSquare*, and *Sub-square*. DLSPHY calculates the lengths of both squares using the node's transmission range. If the lengths of the squares are longer than the transmission range of the nodes, DLSPHY can guarantee that the links in sub-squares never interfere with each other. The only remaining area that links may interfere is inside a sub-square. Since the number of links in each sub-square is small, it is easy to schedule all of the links using any scheduling method, even if it has high time complexity. Sub-squares can only cover a portion of the network. In order to schedule the entire network, *shifting strategy* is used, in which shaded regions change continuously to refresh the whole area.

Partitioning sub-squares in DLSPHY uses the physical interference model to calculate the transmission range, which is the most accurate method to calculate the transmission range. The major drawback of the proposed approach is that forming a grid depends on the location of the nodes, which requires either GPS or a pre-known fixed node deployment.

TDCS in Ahmad et al. (2014) is a lightweight algorithm that calculates the best activation order based on the tree structure in ZigBee.

IEEE 802.15.4/ZigBee has been widely used in low-cost, low-power wireless sensor networks. In this standard, the MAC layer supports two different modes: *Beacon-Enabled* and *No-Beacon*. In the beacon-enabled mode, the network forms a tree with a cluster-head at the root. The cluster-head periodically sends a beacon packet to stay synchronized with children. The time interval between two beacons contains two smaller intervals. (i) Superframe duration (SD) with 16 equal time slots: The first slot in SD is reserved for beacon transmission. The rest of the slots are used as *Contention Access Period* (CAP) and *Contention Free Period* (CFP). (ii) Inactive period: In this interval, the nodes do not send or

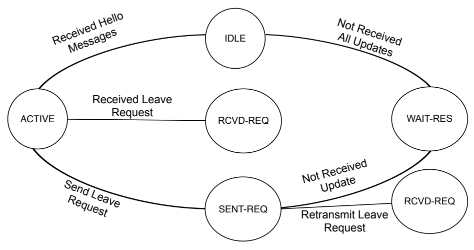

Figure 6 All nodes start in *idle* state. A node may change its state to the *active* state to perform schedule compaction. In this state, a node may switch its slot with the node with maximum ID within the two-hop neighborhood. Nodes also may receive a *leave request* message which indicates that the node must abandon the slot and set it free for others Bhatia & Hansdah (2014).

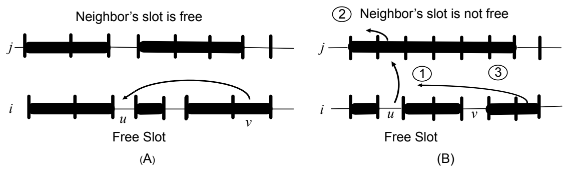

Figure 7 (A) node *i* finds a free slot, *u*, and it swaps its slot with *v* has not been used by the two-hop neighbors including node *j*. (B) Node *i* wants to swap its slot with *u* although the slot is equipped with *j*. Node *i* sends a request to *j* and asks it to leave this slot Bhatia & Hansdah (2014).

receive any packets (sleep interval). Each cluster-head can allocate Guaranteed Time Slots (GTS) in the CFP to its child. The beacon interval (BI) is fixed in all the cluster heads, while SD is varied.

TDCS increases the duration of the inactive period to provide longer sleep duration. The first step is to calculate the duration of the active portion (SD) and the beacon interval (BI) of each cluster. Every node calculates the number of GTS slots required in a BI and the minimum possible value of SD length. In the second step, scheduler models flow constraints to guarantee that the network can meet all of the flows' deadlines. In the third step, TDCS models flows and nodes as a directed graph and assign a weight to each path in the graph. The scheduler uses a standard weighted shortest path algorithm and the graph as an input, to find the best path to the root. Finally, to achieve the maximum possible sleep time for the nodes, the scheduler selects the best ordering of the cluster sleep time slots.

TDCS is an offline scheduling algorithm, and it cannot change the schedule based on network dynamics. As a result, it is not suitable for real-time and mobile networks. Moreover, this approach is implementable in-network with a ZigBee tree topology.

DSC in Bhatia & Hansdah (2014) is a schedule optimization algorithm that receives an existing schedule as an input and generates another schedule with shorter scheduling length. The input of this algorithm can be a very naive schedule generated by a simple greedy algorithm. Figure 6 shows the state machine of the schedule compaction process. All nodes are initialized in the IDLE state. To start the schedule compaction procedure, nodes change their status to the ACTIVE. In this state, all nodes move their scheduled slots toward the beginning of the time interval by finding the maximum slot ID (last used slot) among all two-hop neighbors and swapping it with a slot that is closer to the beginning of the interval. Figure 7 presents two examples of the schedule compaction process. In this figure, slot v has the maximum ID in node i. In Figure 7.A, node i leaves slot v and moves to a new slot

u. Note that slot u is free and no other node is using it in this scenario. If node i cannot find a free slot (see Figure 7.B), it sends a *request* message to all nodes in two hops away from i to leave slot v, and change its status to REQUEST (step 1 in the figure). Nodes that receive the *request* message, change their state to RCVD-REQ and finds the first free slot to leave u (step 3). When all nodes in two-hop neighbors leave slot u (e.g., node j), they send an *update* message, indicating that slot u is free. Finally, when i receive update message from all other nodes in WAIT-RES state, the scheduler preempts the free slot u and replaces it with the slot u that is causing the long scheduling delay. The whole procedure repeats for the next highest slot ID.

Since keeping the state machine updated depends on many messages passing for even a handful of nodes, the proposed approach has a high overhead for sending control messages. Furthermore, this approach may also lead to a situation in which a node has released its schedule and did not get a new one (known as an orphan node). An orphan node may have some overlapped slots with other nodes due to the algorithm failing at the presence of mobility.

M-GCF in Pawar et al. (2012) is a TDMA algorithm for WSN that utilizes a hierarchical structure to find conflict-free slots with energy considerations. The primary goal of this scheduling algorithm is to reduce run-time complexity by leveraging a divide and conquers approach. The scheduler divides the network into several smaller clusters and schedules them individually. Each cluster has a *cluster-head* (CH), and all of the nodes inside a cluster communicate through the CHs. Nodes with intra-cluster communication receive their schedule first. The nodes calculate neighboring nodes within three hops (neighbors that are one to three hops away) and add them to a conflict graph. Then, they use the graph coloring algorithm to schedule nodes. Although the complexity of graph coloring is NP-hard, it presents an acceptable performance once used inside small clusters. In the second step, the scheduler repeats the graph coloring process for cluster-heads to schedule the communication between them.

The performance of this algorithm might be significantly degraded in non-uniform deployments. Specifically, some partitions may end with a significantly higher number of nodes in them. Although M-CGF employs a fixed three-hop partitioning, it does not introduce a parameter for dynamic cluster size adjustment. In addition, large clusters increase the load of CHs and result in higher queuing delay and a lower lifetime of CHs.

PFSA in Hare et al. (2014) is a distributed sensor scheduling, proposed for energy-efficient target-tracking in 2-dimensional underwater wireless sensor networks. Since it is hard to replace or recharge underwater sensor nodes, energy conservation is an important part of these types of networks to reduce energy consumption and increase network lifetime. In PFSA, nodes estimate the target distance and start sensing targets if they are close to the sensing node. This procedure is implemented in a state machine with four states: SLEEP, LISTEN, LOW-POWER, and HIGH-POWER. Figure 8 shows PFSA state machine. All nodes are initially in the SLEEP state, and the only element in the node that consumes power is CPU to stay synchronized with other nodes. When there is no object to track, a sensor node remains in the sleep mode with the probability of P_{sleep}. In other words, a node starts listening (sensing) to the environment and changes the state to LISTEN with the probability of $P_{listen} = 1 - P_{sleep}$. The value of P_{sleep} is used to prevents continuous listening and decreases energy consumption. Nodes in LISTEN state calculate P_{region}, which is the estimation of the distance from the target node. The formulation of this probability is complicated, and its provided in Bar-Shalom & Campo (1986), Chang et al. (1997). The value of P_{region} is smaller for closer targets, and it is larger for remote targets. PFSA uses

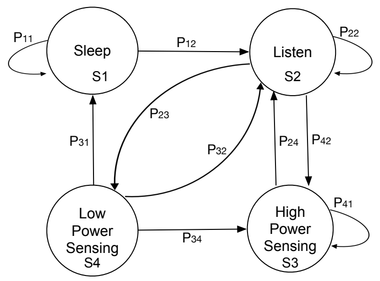

Figure 8 State machine of PFSA. Nodes start in the SLEEP state and choose their state based on the estimations to be in one of the states Hare et al. (2014).

P_{region} to calculate the probability of transition to the LOW-POWER state, HIGH-POWER state, or remaining in the LISTEN state. In the LOW-POWER state, the sensor node turns off its communication device and starts low power sensing. In the HIGH-POWER state, the sensor node uses a Sonar Sensing Device (SSD) Mukherjee et al. (2011) to increase the accuracy of target tracking in a high power sensing.

AnE in Zhang et al. (2014) is proposed for target tracking in 3-dimensional underwater wireless sensor networks. AnE guarantees an accurate track with low energy consumption. This scheduler chooses the best possible sensors for tracking a target and selects one of these sensors as the fusion sensor at each time slot to reduce the sensing burden. This strategy improves the tracking accuracy is compared to the state of the art algorithms by having at least four sensors in coordination. The authors also propose a sampling interval to adjust the accuracy level and node energy consumption. Nodes monitor sensing accuracy and increase the sampling if accuracy is low. Similarly, if accuracy is high enough, nodes decrease sampling to save energy.

IEJSH in Fateh & Govindarasu (2015) proposes the first joint task and packet scheduling algorithm for wireless sensor networks to minimize energy consumption. IEJSH proposes a Mixed Integer Linear Program (MILP) to address joint computation task packet scheduling. This model aims to minimize the total energy consumption of computational tasks and communication messages. MILP takes the following constraints into account: (i) task deadline: network must handle tasks before a certain deadline; (ii) task precedence constraint: controls the order of task execution to guarantees that tasks are in the correct order; (iii) interference constraints: each node starts transmission when no concurrent transmission is activated. MILP finds an optimal solution for the scheduling problem. However, solving LP is NP-hard, a heuristic algorithm named *Interference-aware Energy-aware Joint Scheduling Heuristic* (IEJSH) is used to solve LP in a timely manner. IEJSH converts the network topology to a *mixed-graph* to model task constraints. Mixed-Graph has $2V$ nodes, in which V is the number of nodes in the topology graph. There are two different types of vertices: interference and task. Likewise, there are two types of edges: directed edges that model the tasks' precedence constraints, and undirected edges model links' interference. In a mixed-graph, there is only one path to the root from each node. Once the mixed-graph is built, the scheduler assigns a path to each node in the new graph. Path selection gives the highest priority to children nodes to achieve the best performance.

The authors compared the performance of IEJSH with MILP as baseline. The simulation results show that IEJSH can perform as good as MILP. Besides, energy consumption of IEJSH with some additional energy-saving technologies is optimal. This approach consumes less energy for larger sized networks or when nodes in the network are more spread out. In

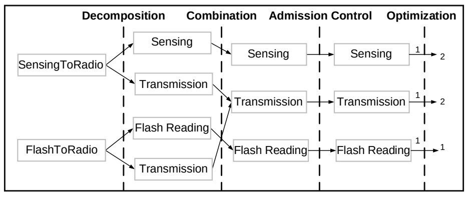

Figure 9 An example of DEOS scheduling. (i) jobs are decomposed into four tasks. (ii) combine tasks if possible. (iii) the algorithm checks if the capacity is still available for a new connection. (iv) scheduler optimizes the atomic tasks Zhu et al. (2012).

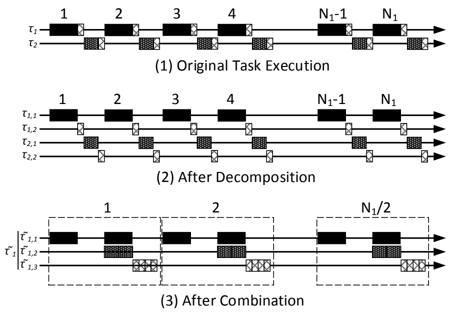

Figure 10 DEOS combination and decomposition Zhu et al. (2012).

the proposed case study, for a network with an average distance of 100m between each pair of nodes, IEJSH consumes 27% less energy compared to the baseline.

DEOS in Zhu et al. (2012) is a real-time, energy-aware, task scheduling to reduce nodes' energy consumption. DEOS dynamically schedules tasks based on the node's available energy and task's energy consumption.

DEOS has four steps: (i) *decomposition*, (ii) *combination*, (iii) *admission control*, and (iv) *optimization*. In the first step, the scheduling node decomposes available jobs into several smaller atomic tasks. In the second step, the scheduler combines all of the decomposed tasks based on two rules: combined tasks must have the minimum number of repetitive tasks, and the combined tasks must be able to execute concurrently. These rules guarantee that the scheduler executes all possible similar tasks consecutively. For example, if there are two tasks from two different jobs that need packet transmissions, the scheduler transmits both packets simultaneously. In the third step, the scheduler leverages admission control to reject a new call when the network cannot handle a new connection. Admission decisions are based on task priority, available energy, and a task's energy consumption. Finally, the scheduler optimizes the tasks to maximize the number of operations that are calculated in the previous steps. For instance, if there are one sensing task and two transmission tasks, the scheduler may execute two sensing tasks and four transmission tasks.

Figure 9 depicts an example of the DEOS scheduling algorithm. This example assumes that the nodes have two jobs: (i) sense an event and transmit information; (ii) read a block of information from flash storage and send it through the radio cards. In the first step, the scheduler decomposes the jobs into three atomic tasks: sensing, transmission, and flash reading. In the second step, the scheduler combines two packet transmission from two jobs into a single transmission task. In the admission control phase, the scheduler calculates the energy constraints and the maximum number of that node can handle, and it rejects the tasks if the total number of available tasks is more than the maximum. Finally, DEOS optimizes the task to find the number of required executions. Figure 10(a) show original jobs execution. After decomposition step, scheduler changes the order of the tasks (τ_1 to τ_4 in figure 10(b)). In the third step, the scheduler combines tasks (figure 10(c)) and provides more continuous idle gaps compared to the original task executions.

3.2 Scheduling Algorithms for High-throughput Wireless Networks

This category broadly includes the improvements over IEEE 802.11 standards MAC and physical layer protocols for wireless networks. IEEE 802.11-1997 is the first standard in this family, and it followed by three most popular, widely used amendments: 802.11a, 802.11b, 802.11g. Almost all of today's wireless devices support these standards. Other amendments have been proposed to improve the limitations of the IEEE 802.11 standards: IEEE 802.11n introduces MIMO to increase the throughput by spatial multiplexing. IEEE 802.11ac was proposed based on IEEE 802.11n with a broader channel (80/160 MHz) and high order modulation (256-QAM) to improve the throughput. Table 3 shows the approaches in this category with the detailed comparison among their characteristic.

Table 3 Summary of Scheduling Algorithms in High-throughput networks

	Multi-Channel	**Multi-Radio**	**Interference Model**	**Call Admission Control**	**Multi-Gateway**	**Scheduler Control**	**Channel Type**	**Topology**	**Application**	**Antenna Type**	**Computational Order**	**Technique**
DelayCheck	Y	Y	Pr	Y	N	Ce	T	G	Voice (Max call)	Omni	$O((vscd)^2)$	Auxiliary graph, Dijkstra
OddEven	Y	N	Pr	N	N	Ce	Td	T	General	Directional	Polynomial	LP, Heuristic SINR base heuristic
Cohabit	N	N	Ph	N	N	Ce	Td	G	General	Omni	$O(e^3)$, $O(ve^2)$	Cohabit graph
DPRL	N	N	Ph	N	N	De	Td	G	General	Omni	$O(m+(\frac{L}{p})^2)$	Convex optimization, SINR base Heuristic
RBT	N	N	Pr	N	N	Ce	Cs	G	Broadcast	Omni	N/A	Heuristic, Non-convex quadratic constrained programming,
IMM	N	Y(2)	Pr	N	N	De	Cs	G	General	Omni	N/A	Channel weight based scheduling
CS	N	N	Pr	N	N	Ce	Cs	G	General	Omni	N/A	Compressive sensing
FRACTEL	Y	N	Ang	N	N	Ce	Td	T	General	Omni	$O(v^{2.5})$	Angular model, Approximation
MCG	Y	Y	Pr	N	N	Di	Td	G	General	Omni	N/A	Multi-radio, Conflict graph
ROMA	Y	Y(2)	Pr	N	Y	Di	Cs	G	General	Omni	N/A	Routing
CSF	N	N	Pr	Y	N	Ce	Cs	G	Voip	Omni	$O(v^2)$	Measurement base capacity utilization
DEC	Y	N	Pr	N	N	Ce	Cs	G	General	Directional	Polynomial	Directed edge coloring
MIROSE	N	N	Pr	N	N	Ce	Cs	G	General	Omni	N/A	Coefficient graph between node and path
OTSLS	Y	Y	Pr	N	N	Ce	Td	G	FTP, Video	Omni	N/A	LP, Heuristic, Max-flow graph
Time-Split	N	N	Pr	Y	N	Di	Cs	T	General	Omni	$O(D)$	Divide and conquer LP, Heuristic, Greedy
QCBS	N	N	2H	N	N	Di	Cs	G	General	Omni	N/A	Greedy
TACK	N	N	N/A	N	N	Di	N/A	G	TCP	Omni	N/A	Transport ACK reduction
MC	N	N	N/A	N	N	Di	Cs	G	General	Omni	N/A	Load balancing
DSFC	N	N	N/A	N	N	Ce	Cs	G	General	Omni	$o(v^3)$	Machine Learning
EDFHOA	N	N	2H	N	N	Ce	Cs	G	General	Omni	N/A	Clustering

OddEven in Narlikar et al. (2010) is a joint link-scheduling and a routing algorithm to improve throughput, and guarantee end-to-end delay. Scheduling begins with labeling links either as *odd* or *even* using Breadth First Search (BFS) algorithm on network topology. Labels are used to divide links into two non-interfering groups. Authors assume that the interfering links are always in one-hop neighbors, and nodes within two-hop can transmit at the same time. If $l(u, v)$ denotes link l with the source node u and destination node v, the algorithm labels link l as odd if and only if u is odd and similarly it labels link l as even if and only if u is even. To address the first type of interference, the scheduler activates only one group of the links at a time slot. Algorithm benefits from routing to tackle the second type of interference. OddEven models routing as an integer linear programming (ILP) optimization problem proposes two different heuristics, *MinMax+SP* and *MinMax*, to reduce the time complexity of the ILP. Both of these heuristics find the shortest path to the root in the network using *Dijkstra* algorithm with interference constraints (only second type). The second algorithm, however, considers existing connections in the network as an input and uses the Dijkstra algorithm for networks that nodes already have some traffics.

In a network with mobility, even a slight movement of nodes games the accuracy of Odd-Even tagging and consequently causes a lot of interference with two interfering links within the same group. Moreover, Odd-even labeling may waste bandwidth in the case that a node does not have traffic to send, but it belongs to the group that is activated. As a result, the proposed algorithm is vulnerable to network dynamic and traffic patterns.

Cohabit is proposed in Nabli et al. (2014) to address interference and reduce scheduling complexity. The authors propose generating a *cohabit* graph from network topology and interference pattern to find the best non-interfering link activation order using two lightweight heuristic algorithms.

Cohabit models co-channel interference by *interference weight* between nodes. Interference weight (c_{ij}) for links i is defined as the reversed distance of link i to j. If two links i and j have a node in common, interference weight will be zero ($c_{ij} = 0$), because either sender or receiver handles the interference. The algorithm produces a *cohabit* graph $\Gamma_{cohabit}(u', v')$ based on the topology graph $G(u, v)$. In cohabit, vertices are the links in G. There is an edge with the weight of c_{ij} between i and j if they are in the interference range of each other. Two different algorithms are proposed for link scheduling. The first algorithm is the Greedy Random Physical Algorithm (GRPA), which generates a feasible schedule using the cohabit graph. GRPA selects a random edge in v, calculates all possible edges that can transmit without interference with the selected edge, and assigns all of these possible links to the first slot. GRPA selects another random node and follows the procedure mentioned above until v becomes empty. The second algorithm is the Cohabit Scheduling Algorithm (CHA). It chooses a vertex with the least number of adjacent vertices in the cohabit graph and accommodates all possible links to a slot. Then, it selects another vertex with the minimum number of adjacent vertices in the cohabit graph, runs the same procedure until all vertices are selected.

This approach benefits from the physical interference model, which is more accurate compared to other interference models. Authors compared their proposed algorithms with SGLS Yang et al. (2010) and IMTIR Gore (2008) in terms of spatial reuse. Their evaluation shows that SGLS, IMTIR, and GRPA demonstrate similar spatial reuse; however, GRPA has lower time complexity. Results also support that CHA has 15% higher spatial reuse compared to the other algorithms.

CS is a medium access control mechanism proposed in Takita (2013) based on *Compressive Sensing* (CS) Donoho (2006). CS is a simple signal processing technique for sampling data under the Nyquist rate. When CS senses data in smaller sizes, the number of multiplication/addition operations in the sensing phase decreases and the delay for generating compressed samples decreases accordingly. Recent works on CS improve the quality of recovered signal by recovering loss in decreasing the size of signal Zanddizari et al. (2018), Mitra et al. (2020). CS is also used to coding and decoding sparse information as a technique to recover a signal using fewer samples that are required by the Nyquist-Shannon theorem Amini et al. (2020), Candès & Wakin (2008). Schedulers benefits from CS features to transmit an time slot without any contention resolution mechanism.

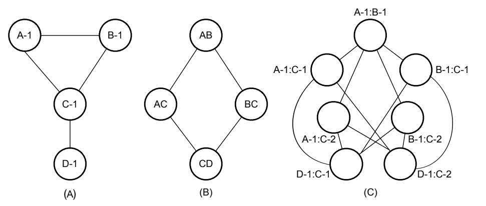

Figure 11 (a) A simple network graph. (b) A conflict graph created by the network graph. (c) A Multi-radio Conflict Graph usable for multi-radio scheduling Ramachandran et al. (2006).

Medium access control starts with broadcasting a solicitation packet. Nodes that need channel access reply to the solicitation packet by sending back a request packet to the coordinator. If a small number of nodes reply to the solicitation packet, the coordinator can decode the replies without any interference, and it assigns a slot to each access request by sending a separated *Ack* messages. If the number of nodes is more than a certain threshold, the coordinator cannot decode the request messages due to the interference and CS failure. In this case, the coordinator groups nodes and put non-interfering nodes in the same group to mitigate interference from the high number of requests. Scheduler sends a solicitation packet again, but only nodes belonging to a particular group must reply.

Simulation results show that CS outperforms the conventional IEEE 802.11 standard by 1.5x higher throughput. CS decreases synchronization procedures; however, it is a centralized approach and handshaking packets are required for each new connection.

MCG in Ramachandran et al. (2006) is one of the earliest attempts of node scheduling and channel assignment in multi-radio, multi-channel wireless networks. MCG uses an interference estimation technique to mitigate interference and increase throughput. Interference estimation takes both interference from other nodes and interference from co-located wireless networks into consideration to achieve a practical scheduler.

MCG estimates interference by counting the number of interfering radios on each channel. To achieve this number, each radio listens to the channel and count the number of distinct MAC addresses that it received. Scheduler identifies the channel with minimum interference with external networks and sets it as the default channel. Once all nodes settled on a default channel, the scheduler creates a Multi-radio Conflict Graph (MCG). Figure 11 shows an example of MCG graph. Figure 11(a) depicts the network topology. MCG generates a conflict graph using the graph topology, such that all edges in the topology graph are the vertices in the conflict graph. MCG uses the conflict graph to generate a multi-radio conflict graph. MCG is shown in figures 11(c). In this figure, node $(A-1:C-2)$ stands for the interference between the first radio in node *A* and second radio in node *C*. The authors propose an algorithm called Breadth-First Search Channel Assignment ($BFS-CA$) to assign channels to the radios of the nodes.

The proposed approach performs better, in terms of throughput, when compared to the static-multi-radio network; however, MCG uses estimation to consider the interference. In a mobile network with highly skewed traffic, the estimation may fail. This algorithm uses a two-hop interference model.

IMM in Tabbane (2012) is an *Interference Management Module* (IMM) for radio network interface. IMM has two sub-modules: *Packet Classifier* and *Packet Scheduler*. *Packet Classifier* is responsible for assigning a queue to the incoming packets. In addition, the classifier assigns a channel to each queue. Packets belonging to the queue must dequeue all the packets in the corresponding Chanel. *Packet Scheduler* is responsible for changing the channel periodically. The scheduler also calculates the minimum service time (T_s) for each channel. T_s is the minimum time that the scheduler must serve each queue in the corresponding channel. IMM has three different algorithms for scheduling channels and queues. The first approach is merely a round-robin algorithm, where the scheduler assigns the same amount of time to each queue. The second scheduling mechanism is static weighted round-robin, in which each queue has a different weight based on its priority. Finally, Dynamic Weighted Round-Robin is used to change the weight of the queues dynamically based on Enhanced Distributed Channel Access (EDCA). EDCA contains four different service levels: background, best efforts, voice, and video. Video and background traffic is assigned the highest and lowest priority, respectively Group (1999).

Although this approach dramatically reduces interference, by implementing channel scheduling, it only considers two network interfaces per node.

DPRL in Hedayati & Rubin (2012) is a rate and power adaptation distributed algorithm to maximize throughput and minimize energy consumption. This algorithm iteratively assigns the highest feasible rate to each link. Each iteration has three phases. In the first phase, all nodes announce their maximum SINR. The scheduling node marks the neighbor with the maximum SINR as the winner. In the second phase, the winner calculates its maximum possible power and transmission rates to transmit in the current time slot. Finally, the scheduled nodes broadcast their power margin levels. The power margin level in a scheduled node (the winner) is the difference between the selected transmission power and the minimum power level required for successful transmission. This value is used in calculating the maximum transmit power level that other nodes can use across the links. DPRL repeats these three phases, iteratively, to schedule other slots until all links are scheduled for a slot.

DPRL chooses the winner node by the SINR level. If nodes are the same in all scenarios, a node with a maximum SINR level always has the chance to send in maximum rate, which leads to an uneven energy level among nodes that may cause starvation, cause a specific link to be the winner always.

RBT in Chang et al. (2012) proposes a centralized, joint power control and scheduling algorithm for minimizing broadcast delay in mesh networks. Increasing transmission power reduces delay because a broader set of nodes are in the transmission range, and they receive broadcast messages in a single transmission; however, sending at full rate and full power increase the power consumption. RBT studies this trade-off by modeling the system as an LP optimization problem to find the best transmission power. The goal of the LP is to minimize the maximum delay in broadcasting packets with minimum power consumption. The authors observed that to minimize the broadcast delay, the transmission rate at the bottleneck paths must be maximum. To calculate the transmission rate, *remaining broadcast time* (*RBT*) is defined, which is the remaining time that the broadcast packets reach the last node of the network. The scheduler also tracks $\overline{RBT}$, which is the difference between broadcast times of various paths. If the value of $\overline{RBT}$ is more than a certain threshold, the scheduler chooses the *Data Rate First* algorithm. This algorithm sorts paths based on *RBT* and assigns a higher priority and data rate to paths with a higher *RBT*. If the value of $\overline{RBT}$ is less than the threshold, the scheduler uses the *Concurrency First* algorithm, which scheduler maximizes the number of non-interfering links. This helps to deliver the broadcast message faster to more number nodes.

In dynamic networks where nodes regularly change their coordination, RBT estimation may not work accurately, and broadcast may take longer than expected.

Time-Split in Malekpourshahraki et al. (2016) proposes a scheduler to control the interference in a wireless mesh network with a tree topology. The scheduler divides the

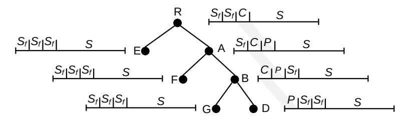

Figure 12 Transmission slots are selected greedily and based on the first available free space Malekpourshahraki et al. (2016).

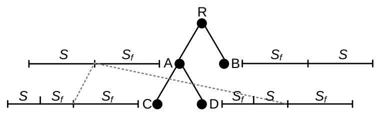

Figure 13 Time-Split algorithm. The interval of the beacon for node A is divided between two smaller intervals, and each child communicate through that internal Malekpourshahraki et al. (2016).

beacon interval into three smaller intervals: (i) sleep interval S_f, in which a scheduling node switches the radio interface to sleep mode. (ii) Child interval C, in which a node is allowed to communicate to its children and (iii) parent interval P, which a node is allowed to communicate with its parents. Each of these intervals could be either equal or greater than the required time for the transmission of a single packet at a given data rate. Initially, the entire beacon interval is marked as S_f. When two nodes want to communicate, the scheduler forms a virtual link as $C - P$ between them by assigning a P to the child node and a C to the parent node. Note that both of these intervals in parent and child nodes must have the same starting time, and they must not be occupied with other C or P intervals. When the schedulers finds the intervals, nodes can communicate using the virtual link. Two approaches are proposed to find the best intervals to form a virtual link: (i) *greedy* in which algorithm search for the first free overlapping sleep interval S_f between parent and child; (ii) *time-split* algorithm in which each node divides its available sleep interval among the number of its children nodes with a certain weight. The division continues to reach the leaf nodes.

Figure 12 shows a simple example of the greedy algorithm. Node R communicates with its children in C intervals, and node A communicates to its parent using P intervals. In each case two intervals form a virtual link. Virtual links guarantees that there is no interference among links as they are scheduled in a different time. Figure 13 shows and example of time-split algorithm. Node A has two children, and it splits its sleep interval into two parts. Node $Root(R)$ uses the first part of the interval to communicate with node A and the second part to communicate with node B (inside S interval). Node A transitions into sleep mode when a child node is transmitting (S_f or forced sleep). The main difference between these two approaches is that time-split is distributed, and it forms contiguous sleep intervals that result in better energy conservation in this approach.

Experimental results show that the bottleneck links (links that are connected to the root) are always fully utilized; however, both of the algorithms waste bandwidth for synchronization period in each beacon.

TACK in Li et al. (2020) is a transport layer approach to mitigate co-channel interference and improve TCP throughput. In particular, TACK tackles the inter-path interference caused by ACK packets from the receiver back to the sender.

TACK reduces the chance of collision among data and ACK packets by reducing the number of ACK packets generated from the receiver. TACK reduces the ACK packet in two different approaches: (1) when the bandwidth-delay product is high, the receiver sends the ACK packets in a fixed time interval regardless of the sender's sending rate. (2) when the bandwidth-delay product is low, the receiver sends an ACK packet when the number of received bytes reaches a certain level (byte counting). Reducing the number of ACK, however, cause some performance problems such as lost recovery, round trip time (RTT) calculation, and setting advertise window. To solve these problems, the authors introduce a new event-based ACK packet (IACK) that the receiver generates to recover the sending rate in the case that the ACK frequency is low. IACK can address lost recovery and rate control by sending an additional ACK packet in the case of recovery from the low advertise window and packet lost. To calculate the RTT, the authors proposed a new RTT calculation mechanism on the receiver side to precisely calculate the RTT without any time synchronization.

TACK evaluations show that this approach performs very well where the data rate is high. For instance, TACK can reduce the number of ACK in 802.11b by 90.5% while for 802.11ac this ratio is 99.8%. TACK also shows 20%, 26.3%, 27.7%, and 28.1% improvement in goodput for IEEE 802.11 b,g,n, and ac respectively.

MC is a multi-connectivity (MC) scheduling schemes studied in both Suer et al. (2020*a*) and Suer et al. (2020*b*). MC uses multiple paths to deliver packets at the same time to improve reliability and latency in wireless networks.

MC proposes three different packet schedulers: Load Balancing (LB), Packet Duplication (PD) and Packet Splitting (PS). In LB, a sender sprays the packets across different paths to reduce the queuing delay. In PD, MC duplicates the packets to increase the successful packet delivery by packet redundancy. Finally, in PS, the scheduler splits packet payload into smaller packets and sends each of them to a differed path. This mechanism reduces the transmission latency since the queue sizes in intermediate nodes receive fewer data packets to handle. In this approach, however, the receiver nodes need to aggregate the segments into a full packet before delivering them to the upper layer.

Authors in Suer et al. (2020*b*) evaluate all three approaches in a real scenario for both IEEE 802.11ac and LTE networks under homogeneous and heterogeneous links along with the effect of link correlation and SNR. Evaluations show that Link Homogeneity is beneficial for all three approaches. PS performs the best when link correlation is high and LB outperforms other approaches when the network is congested.

QCBS in Shu et al. (2020) is a scheduling algorithm based on node queue size for wireless mesh networks. Queue size has widely been used in literature as a parameter to avoid collision since it shows that each device is ready to utilize the channel (Takeda & Yoshihiro (2016), Nardelli et al. (2011), Chattopadhyay & Chockalingam (2010), Xu et al. (2014)).

Similar to Takeda & Yoshihiro (2016), QCBS leverages node activation and deactivation mechanisms. Each node can only send data if it is in active state. The state transition is based on the queue size of the nodes, and nodes with higher queue sizes have a better chance for state transition. Once a node is active it can send a fixed number of frames, and it passes the active state to the next node with the highest queue size. To avoid oscillation between the state, a threshold is defined and nodes check the queue size to be larger than the threshold. This eliminates the chance of frequent transition among different states among two nodes with a high queue size.

Even though the stateful mechanism along with queue size in QCBS mitigate interference, the hidden terminal problem still exists. One way to addresses this problem is predicting collision risk, especially when the network is loaded and the queue size information is not available or not updated in the scheduler. QCBS predicts the collision risk for each packet in the queue based on the state of receiver nodes. Then sender reorders the packets in the buffer such that the sending node has no hidden terminal when it is sending the head of the line packet.

Evaluations show that in 2Mbps of link rate, the delivery ratio is 22.5% higher than QECW in Takeda & Yoshihiro (2016). This approach however is prone to reordering which is not studied in the paper.

DSFC in Shi et al. (2020) is a machine learning-based technique to provide a joint bandwidth allocation and node scheduling problem. DSFC uses federated learning (ML) in McMahan et al. (2017) in which a base station (BS) coordinates all nodes in the network and updates the local ML model in each station and aggregates them in the BS. In Federation learning, however, nodes cannot update their model frequently since the resource are limited at the stations.

DSFC aims to address joint bandwidth allocation and scheduling problems concerning model accuracy and minimizing the expected time for FL training. To achieve this goal, DSFC decouples the joint scheduling and propose two greedy heuristic optimizations: one to calculate bandwidth allocation with a binary search to find the best bandwidth allocation with respect to the convergence time, and the second one to balance between the latency of the model and the accuracy of the FL. The overall time complexity of these two heuristics is $o(M^3)$ in which M is the total number of nodes participating in the FD.

Authors show that DSFC can provide accuracy up to 14.8% compared to the random policy assignment and it also outperforms other state-of-the-art approaches in Nishio & Yonetani (2019).

EDFHOA in Rajguru & Apte (2018) is an enhanced distributed load balancing and task scheduling framework for wireless networks to improve QoS. EDFHOA runs in two phases. First, forming clusters and choosing the cluster heads (CH). EDFHOA chooses CH based on a variety of parameters such as maximum residual energy, connectivity, buffer size, CPU speed, residual energy, density, and the maximum distance to the neighbors. EDFHOA also considers the position of the CH in CH selection to achieve the best performance in the load-balancing optimization. CH selection algorithm fits the CH in the middle of the cluster to achieve the best load balancing. Once CHs are determined, each CH finds the cluster members with *fitness metric* which represents distance, energy, and the connectivity level of the nodes.

In EDFHOA, CHs are responsible for task scheduling. Each CH stores two tables: one contains the cluster information and the other contains the information about the tasks. In EDFHOA, a task is a pair of client-server nodes in the network that need to be scheduled for data transmission. For the tasks that both client and servers are inside the same cluster, CH exchange some control messages to clients and servers to notify them and reserve time slots for data transmission. If the client and server are from different clusters, CHs exchange control messages toward other CHs to opt for information from remote clusters and find the remote CH that contains the peer node. Once the CH pair is found, EDFHOA schedules the task to pass through the CHs before reaching the destination.

Unlike other task schedulers that we discussed in this paper, EDFHOA schedules the transmission among a client and a server instead of operation systems' tasks. Evaluations in the EDFHOA shows that the clustering algorithm and outbalancing outperforms Guo et al. (2014) in energy, throughput, and delay for different scenarios and a different number of tasks.

DelayCheck in Gabale et al. (2011) is an online, joint, centralized scheduling algorithm, that maximizes the number of voice calls in a TDMA-based multi-radio, multi-channel

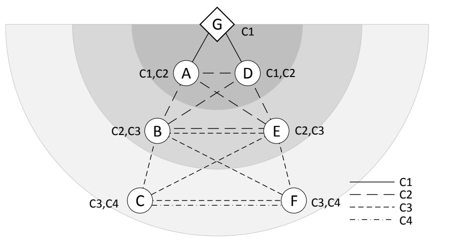

Figure 14 Channel assignment in ROMA. Channel $C1$ is assigned to all nodes in-depth one from the gateway. Channel $C2$ is assigned to all nodes in-depth one and two to make the communication in one-hop possible. Channel $C3$ is assigned to nodes in-depth two and three to connect these nodes. This channel assignment eliminates interference in two-hop links and is more based on the number of channels Dhananjay et al. (2009).

mesh network. DelayCheck formulates the scheduling problem as a linear programming optimization to gain a strict packet-level delay guarantee.

DelayCheck has three phases. In the first phase, scheduler uses the network graph $G(v, e)$ to generates an auxiliary graph $G'(v', e')$. Vertices in v' are quadrupled with four different parameters: (i) corresponding nodes in v, (ii) slot number, (iii) channel number, and (iv) the maximum tolerable delay. The scheduler starts with edge $x \in e$ and adds x as an edge of e' if e has no one-hop interference and is able to meet delay constraints. During the second phase, DelayCheck uses the Dijkstra algorithm to find the shortest path in G'. Note that the shortest path in the auxiliary graph is the best path between the given source and the destination that could meet the delay requirements. In the final phase, the scheduler monitors all of the paths in the network to find and eliminate two or more hops of interference. DelayCheck finds a series of links with the least interference (one hop) and then finds a possible path in this group using the Dijkstra algorithm. This mechanism results in a lower interference level.

DelayCheck introduces *scheduler efficiency*, a new performance parameter that shows the number of accepted calls. DelayCheck was compared to the number of accepted calls in an optimal LP solution and illustrated a higher scheduler efficiency in both IEEE 802.15.4 and IEEE 802.16 networks. The major shortcoming of DelayCheck is that it has a higher computational order compared to the number of hops needed to mitigate interference.

ROMA in Dhananjay et al. (2009) is a routing algorithm that minimizes end-to-end delay. ROMA assumes that all wireless devices have two network radio cards. The scheduler focuses on an efficient channel assignment to reduce network interference and consequently reduces the end-to-end delay.

ROMA addresses channel assignment by assigning channels based on the depth of the links. Figure 14 shows an example of a level-based channel assignment in ROMA. The nodes in the first level use both channels $C1$ and $C2$. The nodes in the second level use $C2$ and $C3$, and so forth. ROMA also addresses the routing sub-problem by defining a routing metric as a value pair of (ETT, L). ETT is Elapsed Transmission Time Draves et al. (2004), Bicket et al. (2005) and L is the external load called the *Transmission Load* of the nodes affecting a scheduling node. The scheduler calculates L by keeping track of both the average (L_a) and mean deviation (L_v) of its measured external load.

Empirical results show that ROMA is four times more efficient in term of throughput compared to a network with two separate channels. It is proven that ROMA eliminates intra-path interference and mitigates inter-path interference.

CSF in Kashyap et al. (2007) concerns VoIP applications with call admission control in wireless mesh networks. The authors propose a path selection mechanism for VoIP calls considering call quality and network capacity constraints. In other words, the goal of this work is to satisfy QoS metrics for VoIP calls and maximize the number of concurrent calls.

CSF uses the concept of capacity utilization in its scheduling algorithm. Capacity utilization is used as a routing metric, named carrier sense factor (*csf*), to calculate the node's residual capacity for transmitting packets. Formally speaking, csf_x^y is the actual rate of node x when both x and y are transmitting at the same time, compared to when node *x* transmitting alone. Since the rate is at its maximum when the node is transmitting alone, csf_x^y is always less than 1. When both *x* and *y* are out of each other's transmission range, *csf* is equal to 1 because there is no interference. When both *x* and *y* are transmitting, the *csf* is equal to 0.5 which means either *x* or *y* can transmit packets. Calculating csf_x^y is difficult as nodes cannot calculate the exact data rate, thus nodes use packet delivery ratio (*PDR*) as an estimation of the csf. Every node tracks its one-hop neighbors during a period of time to calculate their *PDR*. The estimated *csf* is used to calculate the overhead. The overhead metric is calculated using the *overheard traffic* and the *traffic load* based on the probability of successful transmission, retransmission, and interference. These two normalized parameters are shown as node *x*'s residual capacity. This parameter can be used in the path selection procedure to find the route with the least interference. For instance, a selected path may include nodes with the least possible csf. Experimental results show an increase in call acceptance rate by 20%, compared to the traditional shortest path routing that uses a hop count metric. However, the efficiency of this approach highly depends on the accuracy of estimating *csf*, which is an energy-consuming as it requires nodes to operate in the promiscuous mode.

MIROSE in Qiu et al. (2011) is a route selection algorithm for video calls. MIROSE takes light and heavy load (real-time video) traffic into consideration in the path selection mechanism. The authors claim that clients' video quality can vary based on the number of hops and the contention on the network resource for selecting a path. MIROSE benefits from a proxy node to address this problem. The *Proxy* chooses the best video compression parameters for the video call, which best suits the network condition. Moreover, MIROSE offers a path selection mechanism. A correlation function is defined as a quantitative value that shows interference among nodes. The correlation function applied to two paths, p_1 and p_2, is shown as $C_2(p1, p2)$, and denotes the number of interfering nodes between p_1 and p_2. The correlation function between several nodes is defined similarly. $G(p_1, p_2, ..., p_n)$ denotes the number of interfering nodes between all nodes in all of the supplied paths. Formally speaking, in equation, $G(p_1, p_2, ..., p_n) = \sum_{i=1}^{k} \sum_{j=1, i\neq j}^{k} C_2(p_1, p_2)$, G is the routing metric used to find the best path for any new connection request. The path with the minimum value of G is the best route for the path request.

OTSLS is an application-oriented scheduling algorithm proposed in Yu et al. (2008) to increase capacity through channel assignment and scheduling. OTSLS focuses on handling *ftp-type* and *video-type* information. OTSLS produces two different matrices: *channel assignment matrix* (CMT) and *link activation matrix* (LM). Each element in CMT shows whether a channel C is used by link l or not, while elements in LM show the number of activated links in time slot t. Each row in matrix LM shows a link schedule, called a *one-time-slot link schedule* (*OTSLS*). After generating this matrix, the proposed algorithm follows three phases. First, it generates an interference graph based on the topology graph. Each vertex in the interference graph represents a link in the topology graph. The algorithm then generates a *resource contention graph* (*RCG*), which reflects various contention regions by finding maximum cliques in the interference graph. There is an edge between a resource vertex and a link vertex when this link belongs to the contention graph represented by the resource vertex. During this phase, the RCG graph changes to a *Max-Flow* graph by adding source, sink vertices, image links, and nodes. In this graph, the edge capacity is the number of channels *k*, for the *k-first* levels of the max-flow graph, and the other three levels have the capacity of *R*, which is the number of radio interfaces. To solve the *Max-Flow* graph and find the best assignment, the authors use an ILP, while considering the number of channels, radios, flow conditions, and link capacity as ILP constraints. In the second phase, they benefit from two different heuristics, named *greedy set-covering* for *ftp-type* connections, and *link weight adjusting* for *video-type* connections. In the final phase, the scheduler assigns channels to the links.

FRACTEL in Panigrahi & Raman (2009) is an architecture for long distance mesh networks. FRACTEL proposes a new interference model that simplifies scheduling with delay considerations. FRACTEL groups each link and all of its two-hop neighbors as a spoke. A spoke may form different patterns based on the angle between them. There are four defined patterns by FRACTEL, and the coloring of these patterns are already known. If spokes form any of these patterns, then an available coloring is applied. To schedule a network, FRACTEL starts with a pattern with the minimum overhead (no interference) and then proceeds to the next pattern if no match was found. In the end, if there exist some uncolored spokes, a simple graph coloring is employed. FRACTEL claims that the angular model can be extended to more hops, but considering the number of possible spokes in a tree with three hops, the complexity of the algorithm increases dramatically in dense networks.

DEC in Dutta et al. (2008) focuses on continuous full-duplex data transmission without any synchronization across links. DEC improves throughput dramatically and reduces end-to-end delay. To achieve this goal, the network graph is modified such that each edge is replaced with two directed edges. Directed edges are colored with two different colors to provide nodes with full-duplex interference-free communication. The new graph is directed, and the edge coloring algorithm is used to assign channels to the edges. Due to the fact that the graph is directed, the edge coloring algorithm is called $Directed\ Edge\ Coloring$ (DEC). Let G be the new directed graph. The authors define G' as G's none-directed graph. DEC starts with coloring G' and assigning all possible colors to its nodes. Edges in G are colored with the colors in their corresponding nodes. In short, colors are assigned to edges with the constraint that input and output edges are to be different colors. Evaluations showed a 50% to 100% increase in throughput and a considerable decrease in end-to-end delay. Even though the approach attains near-optimal performance, it is an offline solution, and it does not apply to real dynamic networks.

4 Discussion and Observation

In this section, we study the existing state-of-the-art schedulers to see how they match up against our proposed framework. We compare and contrast all the approaches based on three different viewpoints on scheduler design: First, we compare existing works regarding our comprehensive framework on scheduling algorithms to study how many module each approach implements. Second, we compare schedulers based on the target network. Finally, we compare proposed approaches based on the type of algorithms that the authors use to tackle the scheduling problem.

4.1 Framework Based Analysis

Schedulers in different networks have different settings and requirements. Each scheduler addresses one or more sub-problems regarding the network type and objective. For instance, network requirement in a low-power network is energy consumption control, and in a long-distance wireless network, the requirement is utilization and throughput. The following sections draw a comparison between scheduling features in each category based on our framework.

Table 4 Summary of Scheduling Approaches for Low-Power Wireless Networks Based on the Proposed Framework

	Scheduling Type	**Scheduling Goal**	**Reschedule**	**Knowledge**	**Monitoring**	**Propagator**
DRAND	Link	Interference (2-Hop)	N/A	2-Hop (Packet exchange)	Event-based (Packet exchange)	Packet exchange (Distributed)
DLS	Link	Interference	N/A	1-Hop (Location information)	Event-based	Packet exchange (Distributed)
IEJSH	Packet, Task	Interference, Energy, Routing	N/A	Network graph - global	Static	Routing
DLSPHY	Link	Interference	N/A	Network graph - global	Static (fixed nodes)	Whole network
DSC	Link	-	New hello message	2-Hop (Hello message exchange)	Periodic (Packet exchange)	Packet exchange (Distributed)
M-GCF	Link	Interference (3hop), Energy	N/A	Network graph - global	Static (3 hop)	Inter and intera cluster
DOES	Task	Energy	Incoming new task	Tasks list	Local	Distributed
PFSA	Task/Node	Energy, Target tracking	N/A	Local	Static (Locations of nodes)	Distributed
AnE	Task/Node	Energy, Target tracking	Adaptive	Local	Static	Distributed
TDCS	Link	Energy, Duty-cycle	N/A	Local	Periodic (Beacon)	Distributed
SCoRe	Node	Interference	Incoming new command	Local	Local	Centralized
DDLS	Link	Interference	Incoming Packet	Local	Local	Centralized /Distributed
A2E	Packet	QoS	Incoming	Global	Global	Centralized

Table 4 shows different features of the proposed algorithms for low-power wireless networks. This table illustrates that link and task scheduling are the most frequently used strategies for scheduling low power wireless networks in the literature. Task scheduler provides a course grain sleep intervals that leads to an contiguous sleep intervals and efficient energy-saving. Link scheduling schedules links to remove interference and save energy by preventing retransmission.

Energy consumption is closely bounded to throughput in low-power networks. Different papers leverage different strategies to reduce power consumption and increase throughout; however, interference elimination is the mostly used strategy to tackle these problems.

Table 4 also shows that reducing unnecessary retransmission in the MAC layer and preventing concurrent transmission are the best strategies to address interference.

Rescheduling plays an essential role in low-power wireless networks since wireless networks are mobile, and rescheduling is required to renew the schedule. A large number of approaches do not consider any rescheduling mechanism, or they use the original algorithm to generate a new schedule (Enqing et al. (2014), Fateh & Govindarasu (2015), Zhou et al. (2012)); however, some approaches have a separate algorithm that is simpler than the main scheduler algorithm, and it helps a scheduler to adopt new changes in a dynamic network with a lower complexity (Zhang et al. (2014)).

Information gathering from a limited number of hops (less than the length of the network) is the dominant approach in the literature. The accuracy of a scheduler increases when the scheduler has more information abort the network; however, schedulers are sensitive to the accuracy of the received information, which increases the scheduling errors in dynamic networks Pawar et al. (2012), Rhee et al. (2006), Enqing et al. (2014).

A centralized schedule calculation is prone to high overhead and low scheduling efficiency, and there are a few approaches that have a centralized scheduler. To address these limitations, a limited set of nodes (e.g., cluster heads) is used to perform scheduling tasks. Elected nodes consume more power to run the scheduling algorithm, and it leads to a skew power distribution among nodes. To maximize the network lifetime, all nodes must have a uniform power usage which is hard to achieve this in centralized algorithms. Hence, the majority of the proposed approach are distributed.

Table 5 Summary of Scheduling Approaches for High-throughput Wireless Networks Based on the Proposed Framework

	Scheduling Type	**Scheduling Goal**	**Reschedule**	**Knowledge**	**Monitoring**	**Propagator**
DelayChack	Link	Interference, QoS, Routing	N/A	Network graph - global	Static	Routing
OddEven	Link	Interference, Delay, Routing	N/A	Network graph - global	Static	Routing
Cohabit	Link	Interference, Routing	N/A	Network graph - global	Static	Dissemination
DPRL	Link	Interference, Energy	Adaptive	SINR one hop	Event-based	Distributed
RBT	Link	Interference, QoS, Energy Energy	Adaptive	Remaining broadcast time	Estimation	Dissemination
IMM	Link	Channel	N/A	Traffic profile	One hop	Distributed
CS	Link	Routing, Interference	N/A	Request access	Pooling	One hop
FRACTEL	Link	Interference	N/A	Angle of links (Network graph)	Static	Dissemination
MCG	Link-radio	Interference, Channel	N/A	Conflict graph	Static	Dissemination
ROMA	Link	Interference, Channel, Routing	N/A	External traffic, Link load	One hop	Routing
CSF	Link	Interference, QoS, Routing, Duty-cycle	N/A	Packet delivery ratio	One hop	Routing
DEC	Link	Interference	N/A	Network graph - global	Static	Distributed
MIROSE	Link	Interference, Routing	N/A	Network graph - global	Static	Routing
OTSLS	Link	Interference, Channel	N/A	Network graph - global	Static	Distributed
Time-Split	Link	Interference	Adaptive	Global	One hop	Distributed
QCBS	Node	Interference, QoS	Adaptive	Two hops queue size	Two hops	Distributed
TACK	Node	Interference	N/A	End-end	N/A	N/A
MC	Packet	QoS	N/A	Local	Local	Distributed
DSFC	Node	Bandwidth	N/A	One hop	One hop	Centralized
EDFHOA	Task	QoS	N/A	Inside cluster	Cluster tasks	Distributed

A summary of high-throughput wireless network futures are shown in Table 5. This table shows that the main focus in this category is improving throughput using a link scheduler. Moreover, interference is the most frequently addressed sub-problem, and a few works address other sub-problems along with interference. Unlike two other types of the network in this paper, mobility is not the major source of network dynamic. Hence, rescheduling is not required frequently. In the case of a change in the network, schedulers can use the main scheduling algorithm.

A significant number of papers in the literature are centralized algorithms. These work are motivated by the fact that in static networks such as long-distance mesh networks, the scheduler can collect the required information from the entire network without any performance concern. Schedulers can be centralized, which allows accurate interference elimination. In this category, some existing approaches also benefit from routing algorithms to propagate the calculated schedule.

4.2 Approach Integrity

Our framework in Section 2 covers all of the main problems and challenges that are discussed in the literature. We classify the proposed approaches based on the problems they tackled in table 6 to show different components of the framework along with different network types. Approaches under low-power networks show low integrity since they implement a limited number of components. Monitoring is static and limited, and most of the algorithms are distributed. On the other hand, schedulers in high-throughput networks address more number of components, since mobility is less of a problem in many scenarios. The main component in high-throughput networks is interference, and most of the approaches in this category leverage global information along with routing algorithms.

Table 6 Summary of Scheduling Algorithms in Our Proposed Framework

Framework		**Proposed Approaches**	
		Low-power	**High-throughput**
Scheduling	Channel	N/A	IMM, OTSLS, MCG, ROMA
	Interference	DRAND, DLS, IEJSH, DLSPHY, M-GCF	DelayChack, OddEven, Cohabit, DPRL, RBT, CS, MCG, ROMA, CSF, DEC, MIROSE, OTSLS, DEC, MIROSE, OTSLS, FRACTEL
	Routing	IEJSH	DelayChack, OddEven, Cohabit, CS, ROMA CSF, MIROSE
	Duty-cycle	TDCS	CSF
	Energy	IEJSH, PFSA, AnE, TDCS, M-GCF, DOES	DPRL, RBT
	QoS	DMP	DelayChack, RBT, CSF
Monitoring	Static	IEJSH,DLSPHY,PFSA, AnE,DMP,M-GCF	DelayChack,OddEven,Cohabit ,FRACTEL,MCG,DEC, OTSLS
	Periodic	N/A	TDCS, DSC
	Event Based	DRAND DLSDSC	DPRL
Knowledge Required	Limited-Hop	DRAND, DLS, DSC, M-GCF	DPRL, IMM, ROMA, CSF
	Global	IEJSH, DLSPHY, M-GCF	DelayChack, OddEven, Cohabit, DEC, MIROSE, OTSLS
Schedule Propagation	Routing	IEJSH	DelayChack,OddEven,Cohabit,CS, ROMA, CSF, MIROSE
	Packet Exchange	DRAND, DLS, DSC	N/A
	Diemmision		MCG , RBT, FRACTEL, Cohabit
	Distributed	DRAND, PFSA, AnE, DMP, DOES	DEC, OTSLS, DPRL, IMM
Reschedule	Main Algorithm	DSC	DOES, CS
	Adaptive (No need)	AnE	DPRL, RBT

4.3 Network Based Analysis

Scheduling algorithms in wireless networks are usually a part of the MAC layer; however, schedulers may use other network layer information to achieve an accurate schedule. Figure 15 shows some approaches from previous sections. In this picture, different approaches are categorized by network type and where the scheduling algorithm is implemented. Papers on low-power wireless networks are almost always in Mac or Physical layer. On the other hand, approaches in the high-throughput wireless network, used various layers of the network. IMM is the only approach that uses all four layers in the network protocol stack. Cross-layer approaches also show good performance as they benefit from additional information compared to the methods that are defined in a single network layer. For instance, approaches that benefit from routing information in the network layer show a better performance in interference sub-problem.

Table 7 Strategies used in the literature for designing algorithms.

Greedy algorithms	Yu et al. (2020) Chang et al. (2012) Gabale et al. (2011) Liu & Luo (2010) Hong et al. (2007) Li et al. (2005) Zhang et al. (2006) Malekpourshahraki et al. (2016) Stolyar & Ramanan (2001) Wei et al. (2005) Gabale et al. (2011) Enqing et al. (2014) Bhatia & Hansdah (2014) Nabli et al. (2014) Hedayati & Rubin (2012)
Divide and conquer	Liu et al. (2008) Zhu et al. (2007) Chang et al. (2015) Pawar et al. (2012) Rajguru & Apte (2018) Ahmad et al. (2014) Zhu et al. (2012) Zhou et al. (2012) Malekpourshahraki et al. (2016)
Randomization	Wang et al. (2008) Zhang et al. (2014) Hare et al. (2014) Rhee et al. (2006)
Local search	Chang et al. (2012) Qiu et al. (2011) Kashyap et al. (2007)
Graph traverse	Ramachandran et al. (2006) Narlikar et al. (2010) Dhananjay et al. (2009)
Artificial intelligence	Xu et al. (2020) Shi et al. (2020) Chou & Hou (2014) Zhao et al. (2021) Song et al. (2009) Karaboga et al. (2012)
Network flow	Yu et al. (2008) Cai et al. (2017) Marašević et al. (2017)
Dynamic programming	Du et al. (2007) Wang et al. (2016)

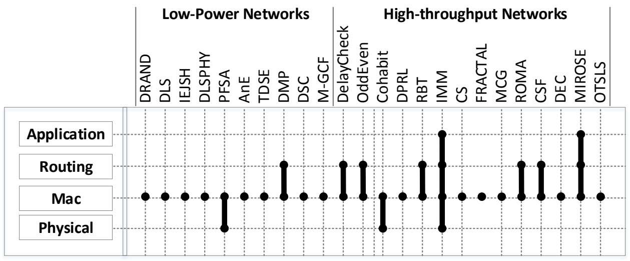

Figure 15 Summary of a few different scheduling algorithms based on the network layer that they are implemented. All of the approaches use the MAC layer information. Some approaches also use other layers of networks to increase the accuracy in schedule calculation.

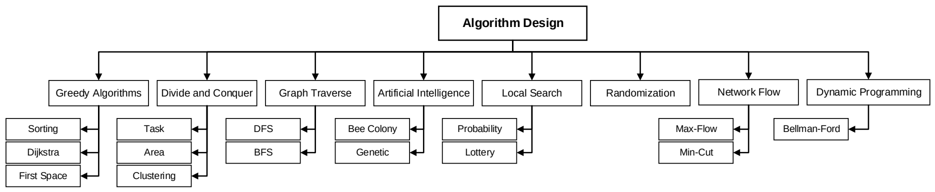

Figure 16 Types of the algorithm design strategies that are used in designing scheduling algorithms.

4.4 Algorithm Design Analysis

Finding an optimal solution for a scheduling algorithm is NP-hard (Cormen (2009)), and it is computationally hard to solve NP-hard problems in polynomial time. This motivates lots of papers to tackle the complexity with some technique that reduces the complexity by finding an sub-optimal solution such as heuristic algorithms or artificial intelligence (Zhao et al. (2021), Chou & Hou (2014), Song et al. (2009), Karaboga et al. (2012)). These technique with the most important approach to implement them are shown in Figure 16; All of the techniques that are used in the literature can be categorized as one of the algorithm design techniques mentioned in Kleinberg & Tardos (2006). We categorize all approaches into the following eight categories:

Greedy algorithms: Algorithms in this category make scheduling decisions based on the current view of the problem. Greedy algorithms produce output that is either the maximum or minimum of one or a set of parameters. Nodes individually sort the parameters of interests and use the minimum (e.g., interference, noise) or maximum (e.g., throughput, signal strength) to achieve the best performance. For example, Chang et al. (2012) sorts data rates to assign the higher data rate to nodes with the larger *RBT*. The well known Dijkstra algorithm Dijkstra (1959) is also a greedy algorithm, which is widely used in the literature of scheduling algorithms (Gabale et al. (2011), Liu & Luo (2010), Hong et al. (2007)), and in computer networks as a polynomial solution to find the shortest path between two nodes. Another greedy concept in the literature is used in Malekpourshahraki et al. (2016), Li et al. (2005), and Zhang et al. (2006), in which slots are allocated to the first available space for sleep, to maximize throughput.

Divide and conquer algorithms: Algorithms in this category divide the problem into several smaller sub-problems (usually two), solve each sub-problem separately, and combine the results to gain the overall solution. Divide and conquer can be used in node placements in Zhou et al. (2012), Liu et al. (2008) or for dividing tasks into several atomic tasks in Zhu et al. (2012, 2007). Another divide and conquer approach is used in Malekpourshahraki et al. (2016) to divide the beacon interval into the number of children. Schedule each part recursively and merge them again.

Another significant scheduling method is clustering the network. Clustering is categorized as a divide and conquer algorithms (Pawar et al. (2012)). Many other approaches presented in the literature also use divide and conquer to tackle the scheduling problem (Lin, Javidi, Cruz & Milstein (2006), Gardellin et al. (2011), Wu et al. (2016), Wan et al. (2006)).

Randomization: Randomization is a stochastic approach to address the scheduling problem for papers in this category. Zhang et al. (2014) and Hare et al. (2014) utilize the probability that an object is present in a certain place. In addition Rhee et al. (2006) benefits from a stochastic lottery for assigning channels which falls into the same category.

Local Search: In this category, an algorithm searches for an optimal solution locally to simplify the computational order. Examples of local search in the literature are Chang et al. (2012), Qiu et al. (2011), Kashyap et al. (2007).

Graph Traversal: The most prominent graph traversal algorithms are: Depth First Search (DFS) and Breadth First Search (BFS) (Tarjan (1972), Cormen (2009)). Both of the algorithms have a linear time complexity for traversing a graph. These two algorithms are used in the literature to solve different problems in scheduling. In Dhananjay et al. (2009) and Ramachandran et al. (2006), the authors use BFS for assigning channels to radios according to their depth, which consequently achieves a non-interfering channel assignment. Narlikar et al. (2010) also used BFS to tag nodes as odd or even and achieve a non-interfering schedule.

Artificial Intelligence: A great number of approaches use different methods of artificial intelligence to optimize scheduling sub-problems. The authors in Karaboga et al. (2012) use the artificial bee colony algorithm for routing in wireless networks. Scheduler in Song et al. (2009) and Chou & Hou (2014) are based on genetic algorithm.

Dynamic Programming: Dynamic programming reduces time complexity, using data structures such as arrays Cormen (2009). The authors in Wang et al. (2016) and Du et al. (2007) use the bellman-ford algorithm, which utilizes dynamic programming to find the shortest path in a graph. Another example is Chang et al. (2015), which leverages a dynamic programming algorithm to merge two sub-optimal problems to form a new optimal solution.

Network Flow: Ford Fulkerson is a polynomial algorithm to find the maximum flow (aka, minimum cut) in a directed graph (Ford Jr & Fulkerson (2015)). Almost all sub-problems in scheduling are maximization or minimization problems, which are reducible to a max-flow problem. For instance, throughput can be modeled as a flow in a max-flow network to maximize throughput. Authors in Yu et al. (2008) design a directed graph and reduce the scheduling problem to a max-flow by adding source and sink to the graph. Similarly Maraševič et al. (2017) uses Max-flow to find the best rate allocation. Max-flow is also used in Cai et al. (2017) which is a routing algorithm for energy harvesting networks that uses max-flow to find the optimal path.

Table 7 shows a comparison between some of the proposed approaches in the literature and the technique that they used.

4.5 Popularity

Designing a scheduling algorithm in a wireless network arises two crucial questions: What are the scheduling goals? And, how to attain our desirable goals? This section provides a brief discussion of the most popular answers to these questions. First, we discuss the most popular scheduling goals across different network types. Second, we briefly discuss the most popular techniques that are used to meet these scheduling goals.

4.5.1 What to Solve

We use our framework to illustrate the most popular scheduling type for each of the two different network settings. Table 8 shows the most used scheduling types. This table shows link scheduling is the most popular approach to tackle interference problems.

- **Low-power networks:** In low power networks, link and task scheduling are used to solve interference and energy sub-problems. Most of the existing approaches use information from a limited number of hops via packet transmissions.
- **High-throughput networks:** In high-throughput wireless networks, link-scheduling is the most popular form of the scheduling algorithm. The scheduler mostly tackles interference and channel assignment problems in this type of network. Because of the static nature of these networks, network information is usually collected globally via static data collection.

Table 8 Scheduling Types Used in the Literature

	Type	Goal	Knowledge	Data Collection
Low-power	Link, Task	Interference, Energy	Limited hop	Packet-based, Static
High-throughput	Link	Interference, Channel	Global	Static

4.5.2 Solving Technique

Scheduling algorithms require a technique to tackle the complexity of the NP-hard of the scheduling algorithms. A handful of methods are employed in the literature. Here we categorize the most common methods in the literature based on our observation in different types of networks.

- **Low-power Networks:** The most observed technique in this category is message passing (Bhatia & Hansdah (2014), Rhee et al. (2006), Enqing et al. (2014)), and dividing the scheduling problem to smaller problems that are easy to solve optimally (Zhou et al. (2012), Pawar et al. (2012), Zhu et al. (2012)).
- **High-throughput networks:** Linear Programming is the most used technique in this category. Approaches in Malekpourshahraki et al. (2016), Narlikar et al. (2010), Chang et al. (2012), Yu et al. (2008) use LP. Since finding the optimal solution for a LP is NP-hard, one or several heuristic algorithms are used to reduce the computational order.

5 Conclusion

Wireless networks can provide the infrastructure for a wide range of applications; however, they cannot meet the requirements of some emerging applications. Scheduling algorithms can address this problem by micromanaging the packet transmission and node behaviour over the network. Designing schedulers are difficult since each scheduler needs to optimize some parameters and address many problems. This complexity motivate us to proposed a comprehensive framework that shows all component that an ideal scheduler needs to implement. We study some of the proposed approaches in two main categories of networks, and we compare and contrast approaches in each category based on our framework. Our framework can open up new avenues toward future research in designing scheduling algorithms by decoupling all required components in a scheduler.

References

Ahmad, A., Hanzálek, Z. & Hanen, C. (2014), A polynomial scheduling algorithm for ieee 802.15. 4/zigbee cluster tree wsn with one collision domain and period crossing constraint, *in* 'Emerging Technology and Factory Automation (ETFA)', IEEE, pp. 1–8.

Amini, F., Hedayati, Y. & Zanddizari, H. (2020), 'Determining the number of measurements for compressive sensing of traffic-induced vibration data', *Measurement* **152**, 107259. **URL:** *https://www.sciencedirect.com/science/article/pii/S0263224119311236*

Amjad, M., Musavian, L. & Rehmani, M. H. (2019), 'Effective capacity in wireless networks: A comprehensive survey', *IEEE Communications Surveys Tutorials* **21**(4), 3007–3038.

Bar-Shalom, Y. & Campo, L. (1986), 'The effect of the common process noise on the two-sensor fused-track covariance', *IEEE Transactions on Aerospace and Electronic Systems* **AES-22**(6), 803–805.

Behnam Dezfouli, Marjan Radi, O. C. (2016), Mobility-aware real-time scheduling for low-power wireless networks, *in* 'IEEE 35th International Conference on Computer Communications, INFOCOM', IEEE.

Bhatia, A. & Hansdah, R. (2014), A fast and fault-tolerant distributed algorithm for near-optimal tdma scheduling in wsns, *in* 'Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on', IEEE, pp. 294–301.

Bicket, J., Aguayo, D., Biswas, S. & Morris, R. (2005), Architecture and evaluation of an unplanned 802.11 b mesh network, *in* 'Proceedings of the 11th annual international conference on Mobile computing and networking', ACM, pp. 31–42.

Cai, B., Mao, S.-L., Li, X.-H. & Ding, Y.-M. (2017), 'Dynamic energy balanced max flow routing in energy-harvesting sensor networks', *International Journal of Distributed Sensor Networks* **13**(11), 1550147717739815.

Candès, E. J. & Wakin, M. B. (2008), 'An introduction to compressive sampling', *IEEE signal processing magazine* **25**(2), 21–30.

Chang, C.-Y., Li, M.-H., Huang, W.-C. & Lee, S.-C. (2015), 'An optimal scheduling algorithm for maximizing throughput in wimax mesh networks', *Systems Journal, IEEE* **9**(2), 542–555.

Chang, K.-C., Saha, R. K. & Bar-Shalom, Y. (1997), 'On optimal track-to-track fusion', *Aerospace and Electronic Systems, IEEE Transactions on* **33**(4), 1271–1276.

Chang, Y., Liu, Q., Jia, X., Tang, X. & Zhou, K. (2012), Joint power control and scheduling for minimizing broadcast delay in wireless mesh networks, *in* 'Global Communications Conference (GLOBECOM)', IEEE, pp. 5519–5524.

Chattopadhyay, A. & Chockalingam, A. (2010), Past queue length based low-overhead link scheduling in multi-beam wireless mesh networks, *in* '2010 International Conference on Signal Processing and Communications (SPCOM)', IEEE, pp. 1–5.

Chou, Z.-T. & Hou, Y.-J. (2014), Genetic algorithm-based energy efficient multicast scheduling for wimax relay networks, *in* 'Computing, Networking and Communications (ICNC), 2014 International Conference on', IEEE, pp. 1061–1065.

Cisco (2016), 'Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020 white paper'.
URL: *http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html*

Cormen, T. H. (2009), *Introduction to algorithms*, MIT press, Cambridge.

Dezfouli, B., Radi, M. & Chipara, O. (2017), 'Rewimo: A real-time and reliable low-power wireless mobile network', *ACM Transactions on Sensor Networks (TOSN)* **13**(3), 17.

Dhananjay, A., Zhang, H., Li, J. & Subramanian, L. (2009), Practical, distributed channel assignment and routing in dual-radio mesh networks, *in* 'ACM SIGCOMM Computer Communication Review', Vol. 39, ACM, pp. 99–110.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische mathematik* **1**(1), 269–271.

Donoho, D. L. (2006), 'Compressed sensing', *Information Theory, IEEE Transactions on* **52**(4), 1289–1306.

Draves, R., Padhye, J. & Zill, B. (2004), Routing in multi-radio, multi-hop wireless mesh networks, *in* 'Proceedings of the 10th annual international conference on Mobile computing and networking', ACM, pp. 114–128.

Du, P., Jia, W., Huang, L. & Lu, W. (2007), Centralized scheduling and channel assignment in multi-channel single-transceiver wimax mesh network, *in* 'Wireless Communications and Networking Conference, WCNC.', IEEE, pp. 1734–1739.

Dutta, P., Jaiswal, S., Panigrahi, D. & Rastogi, R. (2008), A new channel assignment mechanism for rural wireless mesh networks, *in* 'INFOCOM 2008. The 27th Conference on Computer Communications.', IEEE.

Enqing, D., Fulong, Q., Jiaren, W., Zongjun, Z., Dejing, Z. & Huakui, S. (2014), An energy efficient distributed link scheduling protocol for wireless sensor networks, *in* 'Electrical & Electronics Engineers in Israel (IEEEI)', IEEE, pp. 1–4.

Fateh, B. & Govindarasu, M. (2015), ‘Joint scheduling of tasks and messages for energy minimization in interference-aware real-time sensor networks’, *Mobile Computing, IEEE Transactions on* **14**(1), 86–98.

Ford Jr, L. R. & Fulkerson, D. R. (2015), *Flows in networks*, Princeton university press, Princeton.

Gabale, V., Chiplunkar, A., Raman, B. & Dutta, P. (2011), Delaycheck: Scheduling voice over multi-hop multi-channel wireless mesh networks, *in* ‘Communication Systems and Networks (COMSNETS), 2011 Third International Conference on’, IEEE, pp. 1–10.

Gardellin, V., Das, S. K., Lenzini, L., Cicconetti, C. & Mingozzi, E. (2011), ‘G-pamela: A divide-and-conquer approach for joint channel assignment and routing in multi-radio multi-channel wireless mesh networks’, *Journal of parallel and distributed computing* **71**(3), 381–396.

Gore, A. D. (2008), ‘On wireless link scheduling and flow control’, *arXiv preprint arXiv:0812.4744* .

Goussevskaia, O., Halldórsson, M. M. & Wattenhofer, R. (2013), ‘Algorithms for wireless capacity’, *IEEE/ACM Transactions on Networking* **22**(3), 745–755.

Group, I. . W. (1999), ‘Part 11: wireless lan medium access control (mac) and physical layer (phy) specifications: higher-speed physical layer extension in the 2.4 ghz band’, *ANSI/IEEE Std 802.11* .

Guo, W., Li, J., Chen, G., Niu, Y. & Chen, C. (2014), ‘A pso-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks’, *IEEE Transactions on Parallel and Distributed Systems* **26**(12), 3236–3249.

Gupta, N. & Dhurandher, S. K. (2020), ‘Cross-layer perspective for channel assignment in cognitive radio networks: A survey’, *International Journal of Communication Systems* **33**(5), e4261.

Hare, J., Gupta, S. & Song, J. (2014), Distributed smart sensor scheduling for underwater target tracking, *in* ‘Oceans-St. John’s,’, IEEE, pp. 1–6.

Hedayati, K. & Rubin, I. (2012), ‘A robust distributive approach to adaptive power and adaptive rate link scheduling in wireless mesh networks’, *Wireless Communications, IEEE Transactions on* **11**(1), 275–283.

Hong, C.-Y., Pang, A.-C. & Wu, J.-L. C. (2007), Qos routing and scheduling in tdma based wireless mesh backhaul networks, *in* ‘Wireless Communications and Networking Conference, WCNC’, IEEE, pp. 3232–3237.

Ito, Y., Tasaka, S. & Ishibashi, Y. (2002), Variably weighted round robin queueing for core ip routers, *in* ‘Performance, Computing, and Communications Conference, 2002. 21st IEEE International’, IEEE, pp. 159–166.

Jin, H., Seo, J.-B. & Sung, D. K. (2014), ‘Stability analysis of p-persistent slotted csma systems with finite population’, *IEEE Transactions on Communications* **62**(12), 4373–4386.

Jones, A. et al. (2020), 'Survey: energy efficient protocols using radio scheduling in wireless sensor network.', *International Journal of Electrical & Computer Engineering (2088-8708)* **10**(2).

Juniper-Research (2020), 'Industrial iot connections to reach 37 billion globally by 2025, as 'smart factory' concept realised'.
URL: *https://www.juniperresearch.com/press/industrial-iot-iiot-connections-smart-factories*

Karaboga, D., Okdem, S. & Ozturk, C. (2012), 'Cluster based wireless sensor network routing using artificial bee colony algorithm', *Wireless Networks* **18**(7), 847–860.

Kashyap, A., Ganguly, S., Das, S. R. & Banerjee, S. (2007), Voip on wireless meshes: Models, algorithms and evaluation, *in* 'INFOCOM 2007. 26th IEEE International Conference on Computer Communications.', IEEE, pp. 2036–2044.

Kaur, H. & Singh, G. (2011), 'Implementation and evaluation of scheduling algorithms in point-to-multipoint mode in wimax networks 1'.

Kaur, T. & Kumar, D. (2019), 'Qos mechanisms for mac protocols in wireless sensor networks: a survey', *IET Communications* **13**(14), 2045–2062.

Kleinberg, J. & Tardos, E. (2006), *Algorithm design*, Pearson Education India.

Lera, A., Molinaro, A. & Pizzi, S. (2007), 'Channel-aware scheduling for qos and fairness provisioning in ieee 802.16/wimax broadband wireless access systems', *Network, IEEE* **21**(5), 34–41.

Levis, P., Patel, N., Culler, D. & Shenker, S. (2004), Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, *in* 'Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation', Vol. 25.

Li, T., Zheng, K., Xu, K., Jadhav, R. A., Xiong, T., Winstein, K. & Tan, K. (2020), Tack: Improving wireless transport performance by taming acknowledgments, *in* 'Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication', pp. 15–30.

Li, Y., Todd, T. D. & Zhao, D. (2005), Access point power saving in solar/battery powered ieee 802.11 ess mesh networks, *in* 'Quality of Service in Heterogeneous Wired/Wireless Networks, Second International Conference on', IEEE, pp. 5–pp.

Lin, X., Shroff, N. B. & Srikant, R. (2006), 'A tutorial on cross-layer optimization in wireless networks', *IEEE Journal on Selected areas in Communications* **24**(8), 1452–1463.

Lin, Y.-H., Javidi, T., Cruz, R. L. & Milstein, L. B. (2006), Distributed link scheduling, power control and routing for multi-hop wireless mimo networks, *in* 'Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on', IEEE, pp. 122–126.

Liu, B., Dousse, O., Wang, J. & Saipulla, A. (2008), Strong barrier coverage of wireless sensor networks, *in* 'Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing', ACM, pp. 411–420.

Liu, X. & Luo, J. (2010), Joint channel assignment and link scheduling for wireless mesh networks: Revisiting the partially overlapped channels, *in* 'Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on', IEEE, pp. 2063–2068.

Malekpourshahraki, M., Barghi, H., Azhari, S. V. & Asaiyan, S. (2016), 'Distributed and energy efficient scheduling for ieee802.11s wireless edca networks', *Wireless Personal Communications* pp. 1–23.

Malekpourshahraki, M., Stephens, B. & Vamanan, B. (2019*a*), Ether: Providing both interactive service and fairness in multi-tenant datacenters, *in* 'Proceedings of the 3nd Asia-Pacific Workshop on Networking', ACM.

MalekpourShahraki, M., Stephens, B. & Vamanan, B. (2019*b*), Ward: Implementing arbitrary hierarchical policies using packet resubmit in programmable switches, *in* 'Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies', ACM, pp. 34–36.

Marašević, J., Stein, C. & Zussman, G. (2017), 'Max-min fair rate allocation and routing in energy harvesting networks: Algorithmic analysis', *Algorithmica* **78**(2), 521–557.

McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. (2017), Communication-efficient learning of deep networks from decentralized data, *in* 'Artificial Intelligence and Statistics', PMLR, pp. 1273–1282.

Mitra, D., Zanddizari, H. & Rajan, S. (2020), 'Investigation of kronecker-based recovery of compressed ecg signal', *IEEE Transactions on Instrumentation and Measurement* **69**(6), 3642–3653.

Mukherjee, K., Gupta, S., Ray, A. & Phoha, S. (2011), 'Symbolic analysis of sonar data for underwater target detection', *Oceanic Engineering, IEEE Journal of* **36**(2), 219–230.

Nabli, M., Abdelkefi, F., Ajib, W. & Siala, M. (2014), Efficient centralized link scheduling algorithms in wireless mesh networks, *in* 'Wireless Communications and Mobile Computing Conference (IWCMC)', IEEE, pp. 660–665.

Nardelli, B., Lee, J., Lee, K., Yi, Y., Chong, S., Knightly, E. W. & Chiang, M. (2011), Experimental evaluation of optimal csma, *in* '2011 Proceedings IEEE INFOCOM', IEEE, pp. 1188–1196.

Narlikar, G., Wilfong, G. & Zhang, L. (2010), 'Designing multihop wireless backhaul networks with delay guarantees', *Wireless Networks* **16**(1), 237–254.

Nasser, N., Karim, L. & Taleb, T. (2013), 'Dynamic multilevel priority packet scheduling scheme for wireless sensor network', *Wireless Communications, IEEE Transactions on* **12**(4), 1448–1459.

Nishio, T. & Yonetani, R. (2019), Client selection for federated learning with heterogeneous resources in mobile edge, *in* 'ICC 2019-2019 IEEE International Conference on Communications (ICC)', IEEE, pp. 1–7.

Nosheen, S. & Khan, J. Y. (2020), High definition video packet scheduling algorithms for ieee802. 11ac networks to enhance qoe, *in* '2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)', IEEE, pp. 1–5.

Onuekwusi, N. C., Ndinechi, M. C., Ononiwu, G. C. & Nosiri, O. C. (2020), ‘An energy balanced routing hole and network partitioning mitigation model for homogeneous hierarchical wireless sensor networks’, *International Journal of Interdisciplinary Telecommunications and Networking (IJITN)* **12**(1), 28–42.

Pan, H. & Liew, S. C. (2020), ‘Information update: Tdma or fdma?’, *IEEE Wireless Communications Letters* **9**(6), 856–860.

Panigrahi, D. & Raman, B. (2009), Tdma scheduling in long-distance wifi networks, *in* ‘INFOCOM’, IEEE, pp. 2931–2935.

Park, M. & Paek, J. (2021), ‘On-demand scheduling of command and responses for low-power multihop wireless networks’, *Sensors* **21**(3), 738.

Pathak, P. H. & Dutta, R. (2011), ‘A survey of network design problems and joint design approaches in wireless mesh networks’, *IEEE Communications surveys & tutorials* **13**(3), 396–428.

Pawar, P. M., Nielsen, R. H., Prasad, N. R., Ohmori, S. & Prasad, R. (2012), M-gcf: Multicolor-green conflict free scheduling algorithm for wsn, *in* ‘Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on’, IEEE, pp. 143–147.

Qiu, C. & Shen, H. (2017), Fading-resistant link scheduling in wireless networks, *in* ‘2017 46th International Conference on Parallel Processing (ICPP)’, IEEE, pp. 312–321.

Qiu, X., Liu, H., Ghosal, D., Mukherjee, B., Benko, J., Li, W. & Bajaj, R. (2011), ‘Enhancing the performance of video streaming in wireless mesh networks’, *Wireless Personal Communications* **56**(3), 535–557.

Rajguru, A. A. & Apte, S. (2018), ‘Qos enhanced distributed load balancing and task scheduling framework for wireless networks using hybrid optimisation algorithm’, *International Journal of Communication Networks and Distributed Systems* **21**(2), 241–265.

Ramachandran, K. N., Belding-Royer, E. M., Almeroth, K. C. & Buddhikot, M. M. (2006), Interference-aware channel assignment in multi-radio wireless mesh networks., *in* ‘Infocom’, Vol. 6, pp. 1–12.

Rao, D. S. & Hency, V. B. (2021), ‘A novel qoe based cross-layer scheduling scheme for video applications in 802.11 wireless lans’, *SN Applied Sciences* **3**(3), 1–10.

Rao, M. & Kamila, N. K. (2021), ‘Cat swarm optimization based autonomous recovery from network partitioning in heterogeneous underwater wireless sensor network’, *International Journal of System Assurance Engineering and Management* pp. 1–15.

Rhee, I., Warrier, A., Min, J. & Xu, L. (2006), Drand: distributed randomized tdma scheduling for wireless ad-hoc networks, *in* ‘Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing’, ACM, pp. 190–201.

Sah, D. K. & Amgoth, T. (2020), ‘Renewable energy harvesting schemes in wireless sensor networks: A survey’, *Information Fusion* **63**(1), 223–247.
URL: *https://www.sciencedirect.com/science/article/pii/S156625352030316X*

Semeria, C. (2001), ‘Supporting differentiated service classes: queue scheduling disciplines’, *Juniper networks* pp. 11–14.

Shi, W., Zhou, S. & Niu, Z. (2020), Device scheduling with fast convergence for wireless federated learning, *in* ‘ICC 2020-2020 IEEE International Conference on Communications (ICC)’, IEEE, pp. 1–6.

Shu, R., Yi, H., Liu, L. & Liu, D. (2020), A queue-length and collision-risk-prediction based scheduling for wireless mesh networks, *in* ‘2020 Information Communication Technologies Conference (ICTC)’, IEEE, pp. 70–74.

Skiadopoulos, K., Tsipis, A., Giannakis, K., Koufoudakis, G., Christopoulou, E., Oikonomou, K., Kormentzas, G. & Stavrakakis, I. (2019), ‘Synchronization of data measurements in wireless sensor networks for iot applications’, *Ad Hoc Networks* **89**, 47–57.

So-In, C., Jain, R. & Tamimi, A.-K. (2009), ‘Scheduling in ieee 802.16 e mobile wimax networks: key issues and a survey’, *Selected Areas in Communications, IEEE Journal on* **27**(2), 156–171.

Song, J., Li, J. & Li, C. (2009), A cross-layer wimax scheduling algorithm based on genetic algorithm, *in* ‘Communication Networks and Services Research Conference, 2009. CNSR’09. Seventh Annual’, IEEE, pp. 292–296.

Sousa, I., Queluz, M. P. & Rodrigues, A. (2020), ‘A survey on qoe-oriented wireless resources scheduling’, *Journal of Network and Computer Applications* **158**, 102594.

Statista (2020), ‘Internet of things (iot) connected devices installed base worldwide from 2015 to 2025’.
URL: *https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/*

Stolyar, A. L. & Ramanan, K. (2001), ‘Largest weighted delay first scheduling: Large deviations and optimality’, *Annals of Applied Probability* pp. 1–48.

Suer, M.-T., Thein, C., Tchouankem, H. & Wolf, L. (2020*a*), Evaluation of multi-connectivity schemes for urllc traffic over wifi and lte, *in* ‘2020 IEEE Wireless Communications and Networking Conference (WCNC)’, IEEE, pp. 1–7.

Suer, M.-T., Thein, C., Tchouankem, H. & Wolf, L. (2020*b*), Reliability and latency performance of multi-connectivity scheduling schemes in multi-user scenarios, *in* ‘2020 32nd International Teletraffic Congress (ITC 32)’, IEEE, pp. 73–80.

Tabbane, N. (2012), A load and qos aware scheduling algorithm for multi-channel multi-radio wireless mesh networks, *in* ‘Wireless Communications and Networking Conference (WCNC)’, IEEE, pp. 2043–2047.

Takeda, T. & Yoshihiro, T. (2016), A queue-length based distributed scheduling for csma-driven wireless mesh networks, *in* ‘2016 International Conference on Computing, Networking and Communications (ICNC)’, IEEE, pp. 1–6.

Takita, D. (2013), Centralized scheduling for wireless mesh networks with contention-reduced media access, *in* ‘Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on’, IEEE, pp. 493–496.

Tarjan, R. (1972), ‘Depth-first search and linear graph algorithms’, *SIAM journal on computing* **1**(2), 146–160.

Tian, Q., Feng, D.-Z., Hu, H.-S., Yang, F. & Wei, L. (2019), ‘Bi-iterative algorithm for joint localization and time synchronization in wireless sensor networks’, *Signal Processing* **154**, 304–313.
URL: *https://www.sciencedirect.com/science/article/pii/S0165168418303037*

Tian, Y. & Yoshihiro, T. (2020), ‘Traffic-demand-aware collision-free channel assignment for multi-channel multi-radio wireless mesh networks’, *IEEE Access* **8**, 120712–120723.

Wan, P.-J., Yi, C.-W., Jia, X. & Kim, D. (2006), ‘Approximation algorithms for conflict-free channel assignment in wireless ad hoc networks’, *Wireless Communications and Mobile Computing* **6**(2), 201–211.

Wang, Y., Chan, S., Zukerman, M. & Harris, R. J. (2008), Priority-based fair scheduling for multimedia wimax uplink traffic, *in* ‘Communications, 2008. ICC’08. IEEE International Conference on’, IEEE, pp. 301–305.

Wang, Y., Sheng, M., Lui, K.-S., Wang, X., Shi, Y. & Liu, R. (2016), ‘Joint spectrum-efficient routing and scheduling with successive interference cancellation in multihop wireless networks’, *Wireless Networks* **22**(4), 1299–1314.

Wei, H.-Y., Ganguly, S., Izmailov, R. & Haas, Z. J. (2005), Interference-aware ieee 802.16 wimax mesh networks, *in* ‘Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st’, Vol. 5, IEEE, pp. 3102–3106.

Wu, Q., Tao, M. & Chen, W. (2016), ‘Joint tx/rx energy-efficient scheduling in multi-radio wireless networks: A divide-and-conquer approach’, *IEEE Transactions on Wireless Communications* **15**(4), 2727–2740.

Xu, C., Wang, J., Yu, T., Kong, C., Huangfu, Y., Li, R., Ge, Y. & Wang, J. (2020), Buffer-aware wireless scheduling based on deep reinforcement learning, *in* ‘2020 IEEE Wireless Communications and Networking Conference (WCNC)’, IEEE, pp. 1–6.

Xu, Y., Chin, K.-W., Raad, R. & Soh, S. (2014), A novel queue length aware distributed link scheduler for multi-transmit receive wireless mesh networks, *in* ‘Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014’, IEEE, pp. 1–3.

Yang, D., Fang, X., Xue, G., Irani, A. & Misra, S. (2010), Simple and effective scheduling in wireless networks under the physical interference model, *in* ‘Global Telecommunications Conference, GLOBECOM’, IEEE, pp. 1–5.

Yu, H., Mohapatra, P. & Liu, X. (2008), ‘Channel assignment and link scheduling in multi-radio multi-channel wireless mesh networks’, *Mobile Networks and Applications* **13**(1-2), 169–185.

Yu, J., Yu, K., Yu, D., Lv, W., Cheng, X., Chen, H. & Cheng, W. (2020), ‘Efficient link scheduling in wireless networks under rayleigh-fading and multiuser interference’, *IEEE Transactions on Wireless Communications* **19**(8), 5621–5634.

Zanddizari, H., Rajan, S. & Zarrabi, H. (2018), 'Increasing the quality of reconstructed signal in compressive sensing utilizing kronecker technique', *Biomedical Engineering Letters* **8**(2), 239–247.

Zhang, F., Todd, T., Zhao, D. & Kezys, V. (2006), 'Power saving access points for ieee 802-11 wireless network infrastructure', *IEEE Transactions on Mobile Computing* **5**(2), 144–156.

Zhang, S., Chen, H. & Liu, M. (2014), Adaptive sensor scheduling for target tracking in underwater wireless sensor networks, *in* 'Mechatronics and Control (ICMC), 2014 International Conference on', IEEE, pp. 55–60.

Zhao, Z., Verma, G., Rao, C., Swami, A. & Segarra, S. (2021), Distributed scheduling using graph neural networks, *in* 'ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 4720–4724.

Zhou, Y., Li, Z., Liu, M., Li, Z., Tang, S., Mao, X. & Huang, Q. (2012), Distributed link scheduling for throughput maximization under physical interference model, *in* 'INFOCOM, 2012 Proceedings', IEEE, pp. 2691–2695.

Zhu, J., Li, J. & Gao, H. (2007), Tasks allocation for real-time applications in heterogeneous sensor networks for energy minimization, *in* 'Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Eighth ACIS International Conference on', Vol. 2, IEEE, pp. 20–25.

Zhu, T., Mohaisen, A., Ping, Y. & Towsley, D. (2012), Deos: Dynamic energy-oriented scheduling for sustainable wireless sensor networks, *in* 'INFOCOM, 2012 Proceedings IEEE', IEEE, pp. 2363–2371.