Privacy-Preserving Greedy Link Scheduling for Wireless Networks

Anonymous Author(s)

ABSTRACT

Link Scheduling is an important branch of wireless scheduling algorithms for improving wireless throughput, with implications in domains such as Industrial and Agricultural Internet of Things (IIoT and AIoT), smart grids, and Vehicle Ad-Hoc Networks (VANETs). Current approaches for efficient link scheduling primarily focus on computing schedules that maximize the highest-weighted links in a network among all possible schedules. These algorithms are not designed for privacy, as they rely on information about the entire network topology and device link weights. This paper proposes a novel link scheduling algorithm called PriLink with built-in privacy protections, link scheduling performances close to high-performing greedy algorithms, and real-time execution times. PriLink provides privacy benefits over existing algorithms as the entire network topology is not shared, and devices share only links required for computing the schedule hiding everything else. To our knowledge, PriLink is the first implementation of a privacy-preserving link scheduling algorithm. A comparison with high-performing greedy algorithms (Greedy Maximal Scheduling, Local Greedy Scheduling, and Distributed Greedy Scheduling) shows that the PriLink algorithm achieves faster execution times than all algorithms and good scheduling performance with link schedules about 3% lower than the best-performing algorithm for wireless networks comprising 50 devices and about 5% lower for 250 devices. Regarding privacy, we observe that PriLink can hide nearly 85% of network links for networks containing 50 devices from an honest-but-curious adversary. For large networks containing 250 devices, the algorithm can hide more than 95% of the links providing significant privacy benefits for wireless network devices.

KEYWORDS

Wireless link scheduling, privacy preservation, greedy algorithms.

1 INTRODUCTION

The deployment of next-generation distributed multihop wireless networks is expected to be widespread across many applications, including Industrial and Agricultural Internet of Things (IIoT and AIoT), smart grids, Vehicle Ad-Hoc Networks (VANETs), and more [1, 7, 24]. An important consideration related to the efficiency of these wireless networks is the use of efficient scheduling algorithms. Link scheduling, in particular, is a good choice as it maximizes the weights of links that can be activated to transmit packets on a channel at any given moment. An efficient link schedule would ensure that the wireless spectrum and bandwidth are well utilized by wireless devices by (1) enabling high-priority transmissions to occur and (2) minimizing the interference in the network during such transmissions. The calculation of such an efficient schedule involves coordinating device transmissions and encompasses tasks such as link selection, ranking, and transmission (i.e., power allocation and coding schemes) [13, 21, 25].

Significant research has focused on link scheduling algorithms for multihop wireless networks. The most optimal solution for link scheduling requires solving a maximum weighted independent set (MWIS) problem, which is NP-Hard [5, 18, 34]. As a result, research efforts have focused on finding solutions that approximate an optimal solution. Earlier research efforts focused on greedy approaches for link scheduling [8, 33, 34], while more recent efforts have shifted to using ML-based solutions such as Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs), and Spatial Deep Learning [3, 32, 34]. In all the above approaches, the primary focus is on computing link schedules that enable the highest-weighted devices to transmit simultaneously without interference in the network. Another important consideration of the above approaches is execution time since these schedules must be computed in real time for dynamic wireless networks.

The above efforts have designed solutions requiring knowledge of the entire network to compute an efficient link schedule. In centralized link scheduling, all wireless devices must share their device and link information with a central server [12, 34]. In decentralized variations, devices must send information to other wireless devices in the network to decentralize the link schedule calculation [9, 23, 34]. There is no focus on privacy in both cases since the entire wireless network topology and the devices' transmission characteristics are known. We believe that privacy is an essential consideration for link scheduling algorithms. These algorithms may often be utilized in mission-critical activities (e.g., soldiers on a battlefield) where revealing the network topology and devices' transmission characteristics may be detrimental (e.g., loss of life or the battle). Knowing the network topology and devices' transmission characteristics also increases the potential for abuse by an adversary (e.g., a device on the network) who may use the information about the topology to cause disruptions in the network (e.g., update their link weight to always transmit). This highlights a need for link scheduling algorithms that provide privacy protections, good scheduling performance, and real-time execution.

To address the above need, this work proposes a novel privacy-preserving link scheduling algorithm called PriLink that has built-in privacy protections, approaches link scheduling performances close to greedy benchmark algorithms, and has faster execution times than benchmark algorithms. The intuition behind PriLink is that existing greedy algorithms (both centralized and distributed) focus on the highest-weighted links in wireless networks, and other links are discarded during the schedule computation. This led to the hypothesis that a privacy-preserving link scheduling algorithm that shares only the highest-weighted links can be designed. This

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–13

https://doi.org/XXXXXXX.XXXXXXX

has the privacy benefits that (1) the entire network topology is not shared with a central server, and (2) devices share only links required for computing the schedule hiding everything else from a potential adversary. In this paper, we first describe a PriLink base algorithm design that achieves an upperbound privacy performance where devices only reveal one link, their highest-weighted link. We show that this base algorithm is a lowerbound on link scheduling performance as it misses certain links found by greedy benchmark algorithms. Then, we discuss the design of a privacy tolerance-based PriLink algorithm that incorporates the notion of a privacy-scheduling tradeoff. This tradeoff enables the PriLink algorithm to achieve link schedules close to those achieved by greedy benchmark algorithms with execution times still shorter than the benchmark algorithms, with a slight compromise in privacy protections.

The PriLink algorithm was evaluated using simulated wireless networks that mimicked sparse and dense wireless networks and several different wireless network sizes. A comparison with high-performing greedy algorithms (Greedy Maximal Scheduling (GMS) [12, 13, 19], Local Greedy Scheduling (LGS) [10, 14], and Distributed Greedy Scheduling (DGS) [11]) was made in terms of the link scheduling performance, execution times, and privacy protections achieved by PriLink. We observe that the PriLink algorithm achieves good scheduling performance. For small networks (e.g., 50 devices), the link schedules obtained are about 3% lower than LGS (the best-performing algorithm). For large networks (e.g., 250 devices), the schedules are about 5% lower than LGS. The algorithm outperforms DGS in all our simulations. The execution times for PriLink were better than all evaluated benchmark algorithms indicating that the algorithm can be utilized for real-time scheduling. Regarding privacy, we observed that PriLink could hide nearly 85% of all links in small networks (e.g., 50 devices) and more than 95% in large networks (e.g., 250 devices).

In summary, our contributions are as follows.

- We propose a novel link scheduling algorithm called PriLink with built-in privacy protections. To our knowledge, the PriLink algorithm is the first design of a privacy-preserving link scheduling algorithm.
- We compare PriLink with high-performing benchmark algorithms. PriLink approaches scheduling performances close to greedy benchmark algorithms (e.g., GMS, LGS) and outperforms some algorithms such as DGS. It executes faster than the benchmark algorithms.
- The PriLink algorithm achieves significant privacy protections. We observed that it could hide nearly 85% of all links in small networks (e.g., 50 devices) and more than 95% of links in large networks (e.g., 250 devices).

The rest of this paper is organized as follows. Section 2 introduces the most relevant related work. Section 3 briefly discusses background information about link scheduling and privacy concerns with current greedy approaches. Section 4 details the intuitions and design of our privacy-preserving PriLink base and privacy tolerance algorithm. Section 5 presents the performance of our algorithms in terms of link scheduling performance, execution time, and privacy benefits. Section 6 discusses the limitations of our algorithm and future research directions. Finally, we conclude in Section 7.

2 RELATED WORK

In this section, we introduce the most relevant works to PriLink: specifically literature on link scheduling algorithms for wireless networks and literature on privacy in the wireless domain.

2.1 Link scheduling for wireless networks

A significant effort has been made to optimize wireless network link scheduling algorithms. The earlier algorithms in the literature focused on graph theory and greedy approaches to compute link schedules that approximated an optimal solution. Their focus centered on improving throughput while ensuring the algorithms operated in real-time. An optimal link schedule is infeasible since it requires solving a maximum weighted independent set (MWIS) problem, which is NP-Hard. More recently, the efforts have shifted to Machine Learning, which aims to train models that can compute better schedules than traditional greedy approaches. Most of these link scheduling schemes fall into two categories: (i) centralized link scheduling and (ii) distributed link scheduling. Next, we will focus on a subset of centralized and distributed algorithms that are most relevant to our work.

Centralized link scheduling schemes require a central server (serving as a coordinator) to gather all the devices and their link weight information. Using the link weights, this central server computes the link schedule using deterministic or heuristic approaches [18, 28, 34]. In terms of greedy approaches, one example of relevant work is Leconte et al. [19]. They implemented the Greedy Maximal Scheduling (GMS) algorithm and demonstrated improved bounds for computing link schedules. Their algorithm focuses on only local neighborhood information within the network graph instead of the entire graph. Focusing on local information yields good throughput and execution times, enabling their algorithm to compute schedules for wireless networks of any size in near real-time. In terms of Machine Learning, there has been a focus on Graph-based Deep Learning approaches such as Graph Neural Networks (GNNs). One example of a centralized GNN-based system is a paper by Zhao et al. [34]. They demonstrate that GNNs are well-suited for heuristic solutions for link scheduling. The intuition of their work is that training a GNN makes it possible to learn topological information about a wireless network and use the information for weighting links. They show that it is feasible to achieve schedules that outperform traditional link scheduling algorithms by incorporating topological information.

Unlike their centralized counterparts, distributed link scheduling schemes are focused on enabling wireless devices to compute link schedules on their own instead of relying on a central server. They typically implement solutions where link schedules are obtained through an iterative process consisting of rounds of local changes between graph vertices and their neighbors [4, 16, 27]. Like centralized algorithms, this class of algorithms has also seen deterministic and heuristic approaches in the literature. In terms of greedy approaches, multiple relevant distributed greedy algorithms such as Local Greedy Scheduling (LGS), Local Greedy Scheduling Enhancement (LGS-E), Local Greedy Scheduling with Two contention mini-slots (LGS-Two), and Greedy Coloring have been proposed [14]. These algorithms are focused on building decentralized variations of the centralized GMS algorithm while achieving schedules close to

GMS. There are also ML-based distributed solutions published more recently. For instance, Zhao et al. propose a distributed scheduling MWIS solver using GNNs where a trainable GCN module can learn node topology to improve the topological information of the graph with good generalizability and a low increase in complexity [33]. The same group also proposed a delay-oriented distributed scheduler based on GCNs with deep Q learning, where the algorithm is aware of the dependency between the current backlogs of the network and the schedule of the previous time slot [35]. Joo et al. propose a distributed greedy approximation to MWIS for scheduling with fading channels, which has the throughput and the delay close to the optimal max weight that solves MWIS at each time [11]. Another work Cui et al. proposes a Spatial Deep Learning approach to compute interfering nodes during channel estimation and to schedule links efficiently based on the geographic locations of the transmitters and the receivers [3].

Our motivation for this effort aligns with the above-discussed prior works in that our goal is also to generate efficient link schedules that can be computed in real-time. At the same time, our approach differs significantly from prior works because our designed algorithm is the only one, to the best of our knowledge, that takes a privacy-preserving approach where devices only reveal the link weights required by the algorithm to compute the best possible schedule. In all the other cases, the implicit assumption is that the algorithm knows the entire network topology regardless of whether a centralized or decentralized approach is employed.

2.2 Privacy in wireless networks

In this section, we survey research efforts related to privacy in wireless networks. To the best of our knowledge, there are no works that focus on privacy in wireless link scheduling. As such, our overview will focus on general techniques used for privacy preservation in different domains of wireless networks.

Many research efforts have been devoted to preserving privacy from malicious attackers in wireless networks by designing new attack and defense mechanisms [30, 31]. Wang et al. proposed a probabilistic source location privacy protection scheme for Wireless Sensor Networks (WSNs). This work changes the packets' transmission directions by creating phantom nodes, fake sources, and weights to reduce an adversary's monitoring probability [29]. In another work, Koh et al. developed an optimal privacy-enhancing routing algorithm to prevent the inference of transmission routes. They considered global adversaries with lossless and lossy observations, and their techniques use the Bayesian maximum-a-posteriori estimation strategy to reduce the adversary's detection probability for WSNs [17]. Chakraborty et al. proposed a method for temporal differential privacy in WSNs focused on preventing an adversary from achieving extra information about the time of an event of a particular node. They do so by delaying traffic trace at the nodes [2] using differential privacy mechanisms.

Some research efforts have also adopted Machine Learning for privacy in wireless networks. For instance, Mohamed et al. proposed a privacy model for using federated learning over a wireless channel by using user sampling and a wireless gradient aggregation scheme [22]. Liu et al. proposed a gated recurrent unit neural network algorithm for accurate traffic flow prediction while preserving privacy. They combined a federated averaging algorithm with a joint-announcement protocol in the aggregation mechanism to improve privacy, keeping accuracy at a good level and decreasing the communication overhead [20]. Kim et al. presented over-the-air and broadcast adversarial attacks against deep learning to fool a modulation classifier. They also designed a certified defense method to reduce the impact of adversarial attacks on the modulation classifier performance [15]. Tang et al. proposed a privacy-preserving federated learning system by protecting local gradients and global models via efficient verification of local gradients. Their system also defended against passive and active inference attacks without incurring accuracy losses [26].

Our work differs significantly from the above in terms of motivation and methodology. First, our work focuses on link scheduling. In contrast, the above works focus on other wireless domains, such as protecting paths from a source to a destination, protecting the occurrence of events, protecting federated learning gradients and models, etc. Second, our notion of privacy is focused on revealing only relevant information required to compute a schedule. In contrast, the above works provide privacy by adding information (e.g., phantom nodes, temporal noise) to reduce the likelihood of an attack by an adversary. In their case, the information is transmitted in the system and is protected by specialized techniques. In our case, we note that the possibility of a privacy attack is minimized because the system does not transmit the information and is not revealed to an adversary in any form.

3 BACKGROUND

In this section, we briefly introduce some background about the existing state of link scheduling algorithms. In particular, we showcase a typical scenario where link scheduling is employed, describe the concept of a conflict graph used by many greedy link scheduling algorithms, and describe the construction of a benchmark algorithm called Local Greedy Scheduling (LGS) [14] as a representative of greedy algorithms in the literature. Our description will focus on an overview of the LGS algorithm to showcase the privacy concerns and develop an intuition about our privacy-preserving algorithm. We will then use the information from this section to describe our algorithms in detail in Section 4.

3.1 Link scheduling for wireless networks

The goal of wireless link scheduling algorithms is to, given a wireless network, maximize the count and weights of links that can be activated to transmit packets on a channel at any given moment. The schedule must be defined to minimize the possibility of interference in the channel to avoid the scenario of devices needing to retransmit packets. We note that link scheduling algorithms are ideal when (1) other collision sensing protocols, such as CSMA/CA, are unavailable, and (2) the wireless devices can transmit their weights to a server/other nodes to compute the schedule.

The most optimal wireless link schedule that can be computed in a wireless network would be a solution to the maximum weighted independent set (MWIS) problem. MWIS focuses on finding an independent set of the largest weights among all possible independent solutions. Intuitively, such a solution is resource intensive

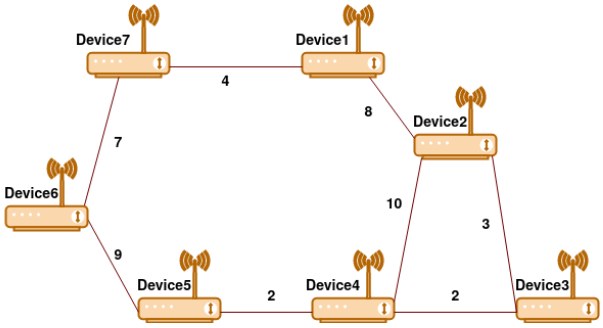

Figure 1: An example multihop wireless network.

and considered to be NP-Hard. As a result, prior research efforts have focused on greedy algorithms (e.g., Greedy Maximal Scheduling, Local Greedy Scheduling, and Greedy Coloring) that compute schedules that approximate MWIS solutions with execution times nearing real-time performance.

While link scheduling algorithms can conceptually be applied to various different topologies of wireless networks, it is more well-suited to multihop wireless networks. For instance, a typical small home Wi-Fi network is not well suited for link scheduling since the algorithms can select only one link at a time to avoid interference in the network. This defeats the purpose of using link scheduling algorithms. As a result, our work will focus on large multihop wireless networks that adopt orthogonal frequency-division multiplexing (OFDM) where the channel is divided into a set of orthogonal sub-channels and time slots. In this scenario, multiple device transmissions at the same time slot may cause interference in the wireless network that our privacy-preserving link scheduling algorithm is designed to mitigate.

3.2 Conflict graphs

A conflict graph is a graph construction technique used in wireless networks to define interference between network links. Using a conflict graph simplifies link scheduling computation because once a link has been selected for the schedule, all its neighboring links can be ignored. This is because choosing any neighbor will cause interference in the network.

To illustrate the construction of a conflict graph, we will consider a simple multihop wireless network shown in Figure 1. We denote this example wireless network as a graph $G = (V, E)$. The vertices $V = \{D_1, D_2, \ldots, D_7\}$ denote the set of all devices in the network. The edges $E = \{L(D_1, D_2, W_{1,2}), L(D_1, D_7, W_{1,7}), \ldots, L(D_6, D_7, W_{6,7})\}$ denote links between the devices, where $D_i \in V$ and $W_{j,k}$ denotes the weight of the link between devices D_j and D_k. In the case of our example network, some example weights are $W_{1,2} = 8$, $W_{1,7} = 4$, and $W_{6,7} = 7$. We note that the choice of weight is determined based on the scheduling requirements of the network.

A conflict graph $G_C = (V_C, E_C)$ can be easily derived from the network graph G. Here, the vertices V_C are simply the edges E of the network graph. If we represent V_C as $\{V_C^{1,2}, V_C^{1,7}, \ldots, V_C^{6,7}\}$, then some example vertices for our example wireless network are $V_C^{1,2} = L(D_1, D_2, W_{1,2})$, $V_C^{1,7} = L(D_1, D_7, W_{1,7})$, and $V_C^{6,7} = L(D_6, D_7, W_{6,7})$. The edges E_C now denote interference between the vertices. In our example network, some example edges associated with vertex $V_C^{1,2}$ are $I(V_C^{1,2}, V_C^{1,7})$, $I(V_C^{1,2}, V_C^{2,3})$, and $I(V_C^{1,2}, V_C^{2,4})$, where I denotes interference between two vertices. Visually, Figure 2 shows the conflict graph constructed for the wireless network in Figure 1. In simple words, if a link between devices D_1 and D_2 is activated, then the links between devices D_1 and D_7, devices D_2 and D_3, and devices D_2 and D_4 should not be activated. Activating either of those links will cause interference in the network, violating link scheduling constraints.

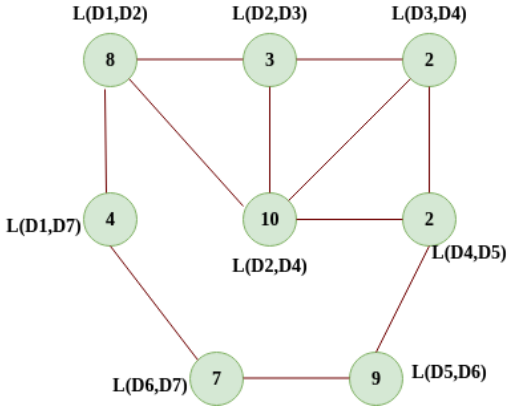

Figure 2: Conflict graph constructed for the example wireless network shown in Figure 1.

3.3 Privacy concerns with existing algorithms

This section will provide an overview of the Local Greedy Scheduling (LGS) algorithm to (1) represent how traditional greedy link scheduling algorithms function and (2) develop an intuition of the privacy concerns associated with traditional algorithms. Our choice of the LGS algorithm is driven by the fact that the distributed algorithm has been shown to output schedules that approach MWIS by prior research efforts [34].

Local Greedy Algorithm Overview: The LGS algorithm uses the conflict graph $G_C = (V_C, E_C)$. Let v denote the set of all devices that have been visited. The first step in the algorithm is to identify a set of devices r that are remaining (not visited). We note that $r = V_C$ in the first iteration of the algorithm. The set r is sorted by weights in the next step, and the maximum weighted vertex is chosen as part of the schedule. In our example network, the algorithm would choose $V_C^{2,4} = L(D_2, D_4, W_{2,4})$ since its weight $W_{2,4}$ is the highest weight at 10. The algorithm would then mark the vertex and its neighbors as visited in set v. In our example network, the set v would contain $\{V_C^{2,4}, V_C^{1,2}, V_C^{2,3}, V_C^{3,4}, V_C^{4,5}\}$ after the first iteration. After each iteration, the set r is updated with the remaining devices ($r = r - v$), sorted by weight, and the maximum weighted vertex is selected for the schedule. This process continues till no devices are remaining. In our example network, the algorithm would choose vertex $V_C^{5,6}$ for the schedule in the second iteration since the vertex has the highest weight at 9 among the remaining devices. The algorithm will also set v as $\{V_C^{2,4}, V_C^{1,2}, V_C^{2,3}, V_C^{3,4}, V_C^{4,5}, V_C^{5,6}, V_C^{6,7}\}$ and r as $\{V_C^{1,7}\}$ after the second iteration. Since only the vertex

$V_C^{1,7}$ remains in r after the second iteration, it is selected by the algorithm in the third iteration completing the final schedule. As a result, the final schedule is computed as $[V_C^{2,4}, V_C^{5,6}, V_C^{1,7}]$ (i.e., $[L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$) with a total weight of 23. In Section 4, we will demonstrate that the same results can be obtained in a privacy-preserving manner using our design and implementation of the PriLink algorithm.

Privacy Concerns: The current link scheduling algorithms have not been designed with privacy in mind. Using the LGS algorithm as an example, we can observe that the algorithm requires the traversal of all the graph vertices to compute the link schedule. In fact, all other greedy algorithms we analyzed during our research required knowledge of the entire graph to calculate the schedule (e.g., Greedy Maximal, Distributed Greedy, Color Greedy). This is undesirable from a privacy perspective since knowledge of the entire wireless network can be abused by any adversary.

In a centralized system, the threat is an honest-but-curious central server. This adversary computes the correct schedule but is interested in learning about the wireless network topology, the devices in the network, and their transmission characteristics. In centralized link scheduling, we note that this adversary may not even be part of the wireless network and can be at a remote location. A suitable example would be the deployment of link scheduling in an adversarial scenario, such as a battlefield. In such a scenario, if a central server (e.g., a 5G gNodeB base station) is used and cannot be trusted, then the leakage of the wireless topology and the devices' transmissions can potentially result in the loss of life and the battle. We believe privacy is vital in such scenarios and warrant using privacy-preserving algorithms like PriLink.

In a distributed system, the threat is any device participating in the wireless network. Since each device in the network knows the wireless topology, an adversarial device can abuse the information. A simple example could be a device aiming to disrupt network communications. Such an adversary can position themselves where they can cause significant interference within the network. After this, if the adversary's device is selected for transmission, they may decide to transmit or not transmit. In any case, they manage to degrade the throughput and efficiency of the wireless network with their attack. Another advantage for the adversary is low detection of the attack since they follow the protocols defined by the wireless network instead of sending unexpected packets. A privacy-preserving algorithm like PriLink will reduce the likelihood of an attack (although it is still possible) since the adversary will not be aware of the network's topology and cannot position themselves for maximum network degradation.

4 PRILINK: PRIVACY-PRESERVING GREEDY LINK SCHEDULING ALGORITHM DESIGN

This section describes the system model and design of the PriLink privacy-preserving link scheduling algorithm. Section 4.1 discusses the system model and our assumptions about the wireless network. Section 4.2 describes the capabilities of adversaries that our algorithm is designed to protect. In Section 4.3, we describe a base PriLink algorithm in detail. The goal of the section is to introduce a simplified version of the algorithm that provides upperbound privacy and lowerbound link scheduling performance of our algorithm. Section 4.4 describes a tolerance-based PriLink algorithm that accepts a tolerance value to incorporate a privacy-scheduling tradeoff in the algorithm.

4.1 System Model and Assumptions

The PriLink algorithm can replace existing greedy link scheduling algorithms as a drop-in replacement. It can be used with multihop wireless networks where the privacy of the wireless network topology and the devices' transmission characteristics is vital (see Section 3.3 for examples). We assume dynamic multihop wireless networks where the topology changes over time, requiring real-time link schedule computation. Even though we assume dynamic networks, we note that our algorithm can yield similar performance in static networks. The choice of dynamic networks showcases modern and dynamic infrastructures that evolve continually. We do note that our algorithm's privacy guarantees are somewhat weakened in static networks since an adversary continually monitoring the schedules can infer the topology over time. This is not the case in dynamic networks, where an adversary cannot infer the topology at any instant when our PriLink algorithm is utilized. The link schedule can be computed by a central server in a centralized setup. The schedule can also be distributed, where devices can broadcast their links, and each computes its schedule. We note that a centralized setup is better as it requires less network traffic; i.e., devices need only transmit their weights to a single server rather than other network devices, and it can also better protect the privacy of the devices from other devices in the network. As such, all further discussions will focus on centralized link scheduling with a central server computing the schedules.

4.2 Adversary Model

We assume an adversary that is honest-but-curious. Since we assume a centralized system, the adversary is a central server that receives the input link weight information from the network devices and outputs an efficient link schedule corresponding to the received inputs. As the adversary is honest, we assume that the schedule will be calculated correctly. As a result, our algorithm does not focus on protecting the link information that is transmitted to the server. As the adversary is also curious, the algorithm limits the link weight information received by the server to only those required to compute the schedule. This way, the algorithm prevents the adversary from inferring the wireless network topology and the devices' transmission characteristics.

4.3 PriLink Base Algorithm

Our intuition for PriLink was derived from the observation that existing greedy link scheduling algorithms focus on finding the highest-weighed link in every neighborhood, ignoring all the other links. This makes intuitive sense since the goal is to find a set of links within the wireless network that maximizes the schedule's total weight among all possible link schedules. This observation drove us to experiment with an algorithm where *the devices only share their highest weighted link* and hide all the other links from a central server. We hypothesized that the server would still be able to compute the schedule. However, this schedule may not approach

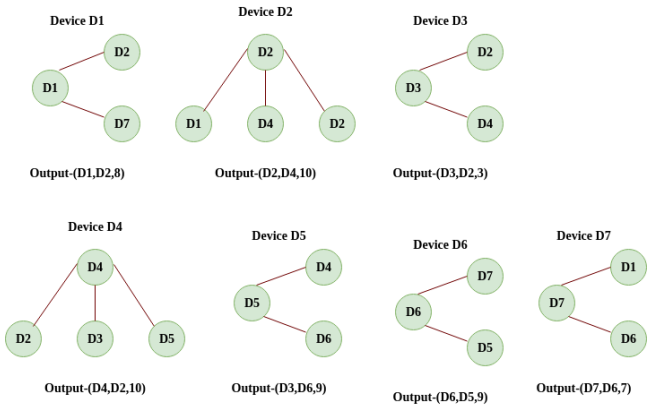

Figure 3: Breaking down the wireless network graph in Figure 1 into small per-device graphs. Each device in the network maintains its link weight information.

Algorithm 1 PriLink Base Algorithm

1: **Input:** $devices \leftarrow \{D_1, D_2, \ldots, D_N\}$
2: **Output:** $schedule \leftarrow []$
3: $links \leftarrow \emptyset$
4: $visited \leftarrow \emptyset$
5: **for all** $D_i \in devices$ **do**
6: $\quad receive\ msg(m = D_i, D_j, W_{i,j})$ ▷ $D_j = max\ E(D_i)$
7: $\quad links \leftarrow links \cup \{m\}$
8: **end for**
9: $sorted\_links \leftarrow sort(links)$
10: **for all** $(D_i, D_j, W_{i,j}) \in sorted\_links$ **do**
11: $\quad$ **if** $!(D_i \in visited \,||\, D_j \in visited)$ **then**
12: $\quad\quad schedule \leftarrow schedule + [(D_i, D_j, W_{i,j})]$
13: $\quad\quad visited \leftarrow visited \cup \{D_i, D_j\}$
14: $\quad$ **end if**
15: **end for**

the performance of traditional greedy algorithms such as the Local Greedy Scheduling (LGS) algorithm. The resulting algorithm from our experiment is shown in Algorithm 1. We note that this algorithm serves as a baseline for PriLink. The reason is that since every device is sharing just one link, we achieve an upperbound on privacy benefits and a lowerbound on link scheduling performance. Revealing at least one link per device is an unavoidable privacy leakage since a central server cannot perform any computations without this information. The maximum weighted device is also necessary to determine which link is selected so the server does not select another link with the same device. In Section 4.4, we will describe an extended algorithm that uses a privacy tolerance value to improve the scheduling performance that approaches existing greedy algorithms' performance with a slight compromise in privacy. The evaluation results for both algorithms (base and privacy tolerance) are reported in Section 5.

The PriLink base algorithm (Algorithm 1) is described below.

Setup: We will describe the setup required for our algorithm using the example wireless network from Figure 1. Given that our algorithm's primary goal is privacy, we note that the entire graph is never shared with the server. Instead, in our setup, each device maintains its own network state. This decentralization ensures that the devices can restrict access to most of their transmission characteristics and reveal only what the server requires to compute the link schedule. A complete decentralization of the network topology from Figure 1 is shown in Figure 3. We can observe that in the figure, device D_1 maintains a graph $G_{D_1} = (V_{D_1}, E_{D_1})$. The vertices comprise the device and its neighbors, i.e., $V_{D_1} = \{D_1, D_2, D_7\}$. The edges only connect the device with its neighbors, $E_{D_1} = \{L(D_1, D_2, W_{1,2}), L(D_1, D_7, W_{1,7})\}$. Similarly, a device D_3 maintain the graph $G_{D_3} = (V_{D_3}, E_{D_3})$ comprising vertices $V_{D_3} = \{D_2, D_3, D_4\}$ and edges $E_{D_3} = \{L(D_2, D_3, W_{2,3}), L(D_3, D_4, W_{3,4})\}$. Similarly, all other devices D_2, D_4, D_5, D_6, and D_7 maintain their own graphs to control their privacy.

Step 1: The first step of the algorithm corresponds to *lines 5-8* in Algorithm 1. In this step, each device from the network shares the following link weight information with the server - $(D_i, D_j, W_{i,j})$ (lines 5 and 6). Here, D_i is the source device. D_j is a neighbor of D_i such that weight $W_{i,j}$ is the maximum weight in the graph G_{D_i}. To illustrate with an example, consider the graphs for devices D_1 and D_3 in Figure 3. The device D_1 will share the link $(D_1, D_2, 8)$ with the server since the maximum weight in the graph G_{D_1} is 8 with device D_2. Similarly, D_3 will share the link $(D_2, D_3, 3)$ since the maximum weight in the graph G_{D_3} is 3 with D_2. The server will add these links to a set called *links* (line 7) which will be used in the next steps to compute the final schedule.

Step 2: This step of the algorithm corresponds to *line 9* in Algorithm 1. In this step, the algorithm sorts all the links available in the set *links* and stores them in a new list called *sorted_links*. Note that the sorting step arranges all the links such that the highest-weighted links are at the top and the lowest-weighted ones at the bottom. For the setup in Figure 3, the result after the sorting step is shown in Figure 4. This step is beneficial since it enables the algorithm to select the highest-weighted links first by traversing the sorted list of links.

Step 3: This algorithm step corresponds to *lines 10-15* in Algorithm 1. In this step, the algorithm iterates over all the sorted links (line 10), ignores links that can cause interference in the network given the current schedule (line 11), adds non-interfering links to the schedule (line 12), and then adds the non-interfering links to the set of visited devices (line 13). We will illustrate these steps using the sorted list of links from Figure 4. The algorithm will have seven iterations since seven network devices share one link each.

Iteration 1: The algorithm returns true for line 11 during this iteration since no visited devices exist. It adds the link $(D_2, D_4, 10)$ to the list containing the schedule in line 12, i.e., $schedule \leftarrow [L(D_2, D_4, 10)]$. It updates the visited set with D_2 and D_4 in line 13, i.e., $visited \leftarrow \{D_2, D_4\}$. The outputs of the first iteration are shown in Figure 5.

Iteration 2: The algorithm returns false for line 11 since both D_4 and D_2 are visited. All other operations are skipped.

Iteration 3: The algorithm returns true for line 11 during this iteration since both D_5 and D_6 are not visited. It adds the link $(D_5, D_6, 9)$

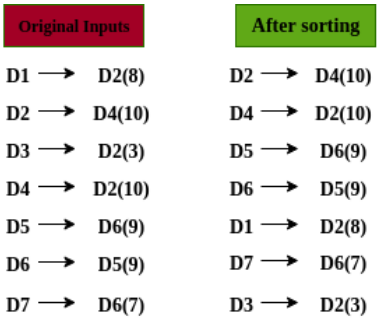

Figure 4: PriLink Algorithm Step 2 - Sorting of network devices by weight.

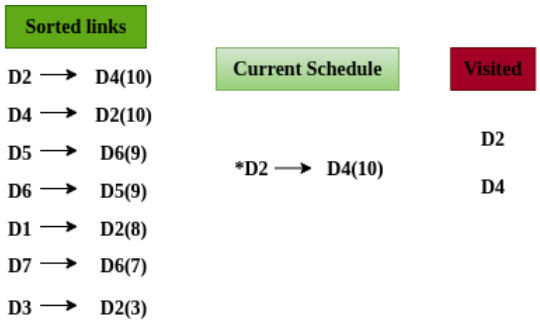

Figure 5: PriLink Algorithm Step 3 - Schedule after the first iteration.

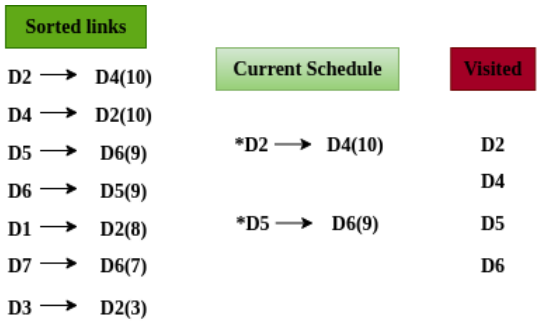

Figure 6: Final link schedule calculated by the PriLink Base Algorithm.

to the list containing the schedule in line 12, i.e., $schedule \leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9)]$. It updates the visited set with D_5 and D_6 in line 13, i.e., $visited \leftarrow \{D_2, D_4, D_5, D_6\}$.

Iterations 4 - 7: The algorithm returns false for line 11 for all these iterations, and all other operations are skipped. In iteration 4, both D_6 and D_5 are visited. In iteration 5, device D_2 is visited. In iteration 6, device D_6 is visited. In iteration 7, device D_2 is visited. The link schedule is finalized at this point as $schedule \leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9)]$. The outputs of the final iteration are shown in Figure 6.

Algorithm 2 PriLink Privacy Tolerance Algorithm

1: **Inputs:** $devices \leftarrow \{D_1, D_2, \ldots, D_N\}$, $tolerance\ (\tau) \in \mathbb{R}$
2: **Output:** $schedule \leftarrow []$
3: $remain \leftarrow copy(devices)$
4: $visited \leftarrow \emptyset$
5: **for all** $t \leftarrow 0$ **to** τ **do**
6: $\quad links \leftarrow \emptyset$
7: $\quad$ **for all** $D_i \in remain$ **do**
8: $\quad\quad receive\ msg(m = D_i, D_j, W_{i,j})$ ▷ $D_j = sort(E(D_i))[t]$
9: $\quad\quad links \leftarrow links \cup \{m\}$
10: $\quad$ **end for**
11: $\quad sorted\_links \leftarrow sort(links)$
12: $\quad$ **for all** $(D_i, D_j, W_{i,j}) \in sorted\_links$ **do**
13: $\quad\quad$ **if** $!(D_i \in visited\ ||\ D_j \in visited)$ **then**
14: $\quad\quad\quad schedule \leftarrow schedule + [(D_i, D_j, W_{i,j})]$
15: $\quad\quad\quad visited \leftarrow visited \cup \{D_i, D_j\}$
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: $\quad remain \leftarrow remain - visited$
19: **end for**

Recall that we had previously described this PriLink algorithm as a baseline that provides an upperbound on privacy benefits and a lowerbound on link scheduling performance. In our example wireless topology from Figure 1, we observe that the server sees 5 unique links because links $L(D_2, D_4, 10)$ and $L(D_4, D_2, 10)$ are the same, and so are links $L(D_5, D_6, 9)$ and $L(D_6, D_5, 9)$. Even in a small graph with 8 links, about 3 links are not disclosed to an adversary. Section 5 shows that a large network can hide more than 95% of its links using the PriLink algorithm.

The PriLink base algorithm also provides a lowerbound of link scheduling performance. Comparing the algorithm's schedule (i.e., $[L(D_2, D_4, 10), L(D_5, D_6, 9)]$) for the example network from Figure 1 with LGS schedule (i.e., $[L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$), we observe that the base algorithm computes a suboptimal schedule missing the link $L(D_1, D_7, 4)$. The problem occurs because the algorithm does not select the highest weighted links for D_1 (i.e., $(D_1, D_2, 8)$) and D_7 (i.e., $(D_7, D_6, 7)$) as D_2 and D_6 are already visited. The missing link $(D_1, D_7, 4)$ is never transmitted. If D_1 and D_7 are able to retransmit their second-highest-weighted links $(D_1, D_7, 4)$ and $(D_7, D_1, 4)$, the algorithm will now be able to select this missing link in the next iteration. This limitation of the PriLink base algorithm is solved using the privacy tolerance algorithm (described next) that enables devices not yet part of the schedule to retransmit another link which may get selected by the algorithm.

4.4 PriLink Privacy Tolerance Algorithm

The PriLink privacy tolerance algorithm is shown in Algorithm 2. A large section of the algorithm (lines 6 - 17) is similar to Algorithm 1. Here, we will only focus on the differences between the two algorithms. The privacy tolerance algorithm now accepts a tolerance value $\tau \in \mathbb{R}$ as an input parameter (line 1). This τ parameter specifies the maximum count of links that devices can expose to

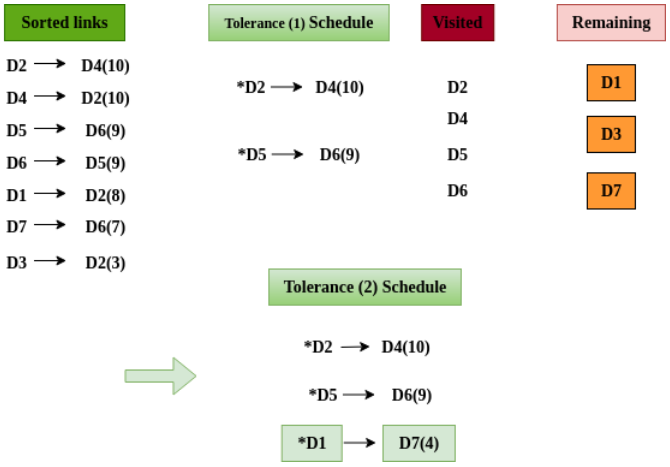

Figure 7: Final link schedule calculated by the PriLink Privacy Tolerance Algorithm.

the algorithm and can be adjusted based on the privacy requirements of the wireless network. Line 3 of the algorithm initializes a new set called *remain* that tracks the devices not scheduled in a previous iteration. In the first iteration, *remain* is set to all devices in the network to ensure that all are traversed at least once. Line 5 of the algorithm creates a new loop t from index 0 to $\tau - 1$. The loop ensures the algorithm does not request more links than the specified privacy tolerance value. At the end of each iteration of the loop in line 18, the set *remain* is updated only to use unscheduled devices for the next iteration of line 5. Another difference between Algorithms 1 and 2 is line 8, where the devices now send their link information based on the index value t. For example, if $t = 0$, the devices will send their highest-weighted link information. At $t = 1$, the devices will now send their second-highest-weighted link information if one is available, and so on. We note that some edge cases are not shown in Algorithm 2 for brevity. One example is that the algorithm terminates early (i.e., $t < \tau - 1$) if *remain* contains only one or no devices so that devices do not send more links than what is required to compute the schedule.

We will now apply the PriLink privacy tolerance algorithm on the example wireless network from Figure 1. The first loop of the algorithm, $t = 0$, functions the same as Algorithm 1 and produces the output shown in Figure 6. At this point, observe that $schedule \leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9)]$ and $visited \leftarrow \{D_2, D_4, D_5, D_6\}$. The set *remain* at the end of this loop will now contain $\{D_1, D_3, D_7\}$. This state is visually shown in the top half of Figure 7. The next loop $t = 1$ at line 5 will request the second-highest-weighted link from D_1, D_3, and D_7 at line 8. These links will be $(D_1, D_7, 4)$, $(D_3, D_4, 2)$ and $(D_7, D_1, 4)$. At the end of this loop, we will have $schedule \leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$, $visited \leftarrow \{D_2, D_4, D_5, D_6, D_1, D_7\}$, and $remain \leftarrow \{D_3\}$. Recall that the algorithm terminates when one or zero devices remain, which is the case now. Therefore, the final schedule is $[L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$ which is the same result obtained using the Local Greedy Scheduling algorithm. This schedule is visually shown in the bottom half of Figure 7. This example illustrates how the privacy tolerance value can improve the link scheduling performance approaching the performance of greedy benchmark algorithms.

Table 1: Wireless networks simulation parameters.

Methodology	Random topologies using ER model
Graph sizes	[10, 50, 90, 130, 170, 210, 250]
Probabilities (P)	[0.2, 0.8]
Number of runs	250 runs per graph size and probability

Table 2: Benchmark algorithms.

1. Greedy Maximal Scheduling (GMS)
2. Local Greedy Scheduling (LGS)
3. Distributed Greedy Scheduling (DGS)

5 PERFORMANCE EVALUATION

This section describes the wireless network simulation methodology and parameters, the performance metrics, and the results of the evaluation of our PriLink algorithms compared with existing high-performing benchmark greedy algorithms.

5.1 Wireless Network Simulation Methodology

Multiple wireless networks were generated using simulations, enabling us to test the performance of our algorithms using random but realistic topologies. These random wireless networks were generated using the Erdős–Rényi (ER) [6] model. In the ER model, given a count of nodes n and a probability p, a network graph is constructed by connecting n nodes randomly with probability p independent of other edges. Other works have also previously used the ER model for assessing link scheduling performance [33, 34].

Simulation parameters: Table 1 shows the parameters used for the simulation. In this work, we have evaluated our algorithms using multiple graph sizes ranging from small graphs of 10 devices to large graphs of 250 devices. Many other graph sizes between 10 and 250 are also considered to assess the effect of different graph sizes, as shown in the table. The ER model probabilities are chosen as 0.2 and 0.8. Here, a probability of 0.2 will enable us to assess the performance of our algorithms on sparse graphs. On the other hand, a probability of 0.8 will help us evaluate the performance on dense graphs. For each graph size and probability pair (e.g., $(10, 0.2)$, $(130, 0.8)$, $(210, 0.2)$), we generate 250 graphs and run our algorithms and benchmarks for each graph.

Benchmark algorithms: Table 2 shows the greedy benchmark algorithms used for the evaluation. These algorithms were chosen because they demonstrated high-scheduling performance and low execution times. We do not evaluate specific greedy algorithms if their (1) link scheduling performance is lower than selected algorithms and (2) the execution time is relatively high. One such algorithm is Color Greedy Scheduling [14] which underperforms compared to Local Greedy Scheduling (LGS) and has a worst-case complexity of $O(K * E)$, where K is the count of broadcast messages and E are the graph edges. Similarly, ML-based works were not

chosen as they require high training time and are not well-suited for real-time execution on dynamic wireless networks.

(1) Maximum Weighted Independent Set (MWIS) [18, 28]: This is the upperbound of link scheduling performance. MWIS is not implemented as the problem is NP-Hard and infeasible to evaluate for experiments with many runs.
(2) Greedy Maximal Scheduling (GMS) [12, 19]: GMS is a centralized greedy algorithm with link scheduling performance that approximates MWIS. The algorithm is centralized as it requires global knowledge of link weights.
(3) Local Greedy Scheduling (LGS): LGS is a distributed greedy algorithm with link scheduling performance very similar to GMS. The algorithm uses only local neighborhood information and can be computed simultaneously by multiple devices on a network.
(4) Distributed Greedy Scheduling (DGS): DGS is another distributed greedy algorithm with good link scheduling performance. It is not as accurate as LGS but provides faster computation as the algorithm can be parallelized.

5.2 Performance Metrics

The following metrics were used to measure the performance of the PriLink algorithms compared with high-performance greedy benchmark algorithms.

- **Link Scheduling Performance:** We measure the link scheduling performance as the total weight of the links that are selected by an algorithm for the schedule, in comparison to the total weight of the schedule of the best performing greedy benchmark algorithm - the Local Greedy Scheduling (LGS). The performance is measured as a percentage value for ease of display. If W_a is the total weight of the schedule computed by an algorithm under test and W is the total weight computed by LGS, then the performance is measured as $\frac{W_a}{W} * 100$.
- **Execution time:** We measure the execution time of PriLink as the time taken by the algorithms (compared with Local Greedy) to compute the final schedule given the list of devices and a privacy tolerance value as inputs. In other benchmark algorithms, the execution time is measured from when the algorithm receives a graph as input and starts computing the schedule.
- **Privacy Cost:** We define privacy cost as the count of unique links disclosed by wireless devices compared to the total links in the wireless network. If N denotes the total links in a network and N_d denotes the count of links disclosed to the algorithm, then the privacy cost is measured as $\frac{N_d}{N} * 100$. We note that, except for PriLink, all benchmark algorithms in our set have a privacy cost value of 100% for every run as the entire network topology and link weights information is shared with the algorithm.

5.3 Evaluation Results

This section will report the results obtained for the evaluation of the simulations using the methodology from Section 5.1 and the metrics from Section 5.2. The scheduling performance results are reported in Section 5.3.1. The execution times are reported in Section 5.3.2. The privacy costs results are then reported in Section 5.3.3. Finally, the impact of privacy tolerance values on the scheduling performance is reported in Section 5.3.4.

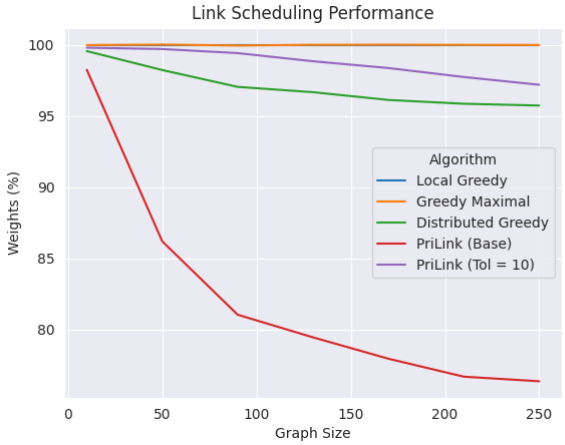

Figure 8: Link scheduling performance of PriLink compared with benchmark algorithms for sparse networks ($P = 0.2$).

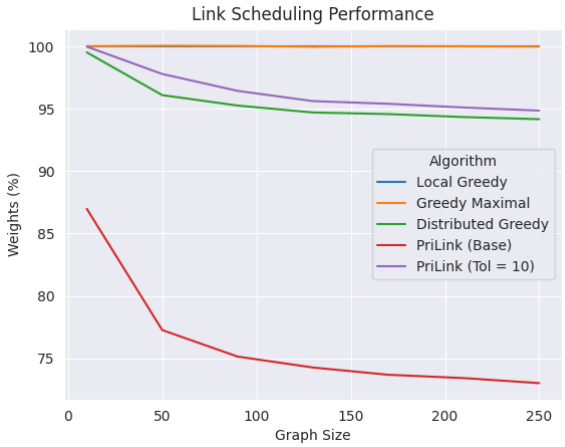

Figure 9: Link scheduling performance of PriLink compared with benchmark algorithms for dense networks ($P = 0.8$).

5.3.1 Link Scheduling Performance Evaluation Results. This evaluation focuses on measuring the link scheduling performance of PriLink as compared to existing greedy algorithms specified in Section 5.1. Measuring this metric is essential since prior research in link scheduling has primarily focused on improving this metric. Intuitively, a privacy-preserving algorithm is most beneficial if it provides good link scheduling performance comparable to prior algorithms from the literature.

Figure 8 shows the performance of the PriLink base and PriLink privacy tolerance algorithms for sparse networks ($P = 0.2$), as compared to Local Greedy Scheduling (LGS), Greedy Maximal Scheduling (GMS), and Distributed Greedy Scheduling (DGS). The graph

sizes range from 10 to 250 in increments of 40, and the number of runs per graph size is set to 250 (see Table 1 for parameters). For the PriLink tolerance algorithm, we chose a privacy tolerance value of 10 as our privacy costs evaluation (see Section 5.3.3) showed that the privacy leakage is only slightly higher than the PriLink base algorithm with less than a 1% difference in disclosed links for a graph size of 250 devices. The figure shows that LGS (blue) and GMS (orange) are very similar in link scheduling performance, with less than 0.1% difference between the two algorithms. We use LGS as the reference point as manual analysis indicated that it produces schedules that are just slightly higher than GMS for our simulations. The figure shows that the PriLink base algorithm is ineffective at obtaining link schedules close to the benchmarks for large graphs. The average link scheduling performance is about 98% of LGS for a graph size of 10 but degrades significantly with a larger graph size of 50 with a performance of about 86% of LGS. The performance further degrades with a graph size of 250, about 76% of the LGS performance. On the other hand, the PriLink tolerance algorithm with a tolerance value of 10 shows performance that approaches LGS. The tolerance algorithm's performance is about 99% of LGS for a graph size of 50 and about 97% for a larger graph size of 250. In all cases, it outperforms the DGS algorithm indicating that our algorithm outputs schedules that are more efficient than some well-cited greedy algorithms.

Figure 9 shows the performance of the PriLink base and PriLink privacy tolerance algorithms compared with LGS, GMS, and DGS for dense networks ($P = 0.8$). The link scheduling performance of the PriLink privacy tolerance algorithm for dense networks shows a slight degradation in performance respective to sparse networks. For instance, for a graph size of 50, the average scheduling performance degrades from 99% of LGS in sparse networks to about 97% of LGS in dense networks. The effect is more pronounced in a graph size of 250, where the average link scheduling performance reduces from 97% of LGS in sparse graphs to about 94.5% of LGS in dense graphs. Despite this, the performance is close to DGS. It indicates that the algorithm is still quite efficient and can be used for real-world link scheduling scenarios where the privacy of the network topology and link weights is important.

5.3.2 Execution Times Evaluation Results. This evaluation measures the execution time of PriLink as compared with the Local Greedy Scheduling (LGS), Greedy Maximal Scheduling (GMS), and Distributed Greedy Scheduling (DGS) algorithms. The goal is to evaluate whether PriLink can be used for real-time privacy-preserving link scheduling in dynamic multihop wireless networks. If τ denotes the privacy tolerance value and n denotes the total count of devices in a wireless network, then the worst-case time complexity of the PriLink privacy tolerance algorithm is $O(\tau n log n)$ as shown in Algorithm 2. We note that the execution time is low since τ and n comprise small numbers that can be computed efficiently on modern systems.

Figures 10 and 11 show the average execution time of the PriLink algorithms compared with the benchmark algorithms. These execution times were derived from the same experiment discussed during the link scheduling performance evaluation. For this execution time evaluation, we again use the baseline as LGS and compare each algorithm's time as a percentage value with respect to the LGS time.

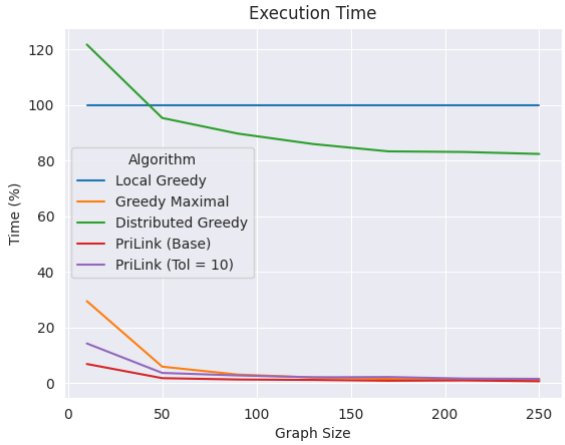

Figure 10: Execution time comparison between PriLink and benchmark algorithms for sparse networks ($P = 0.2$).

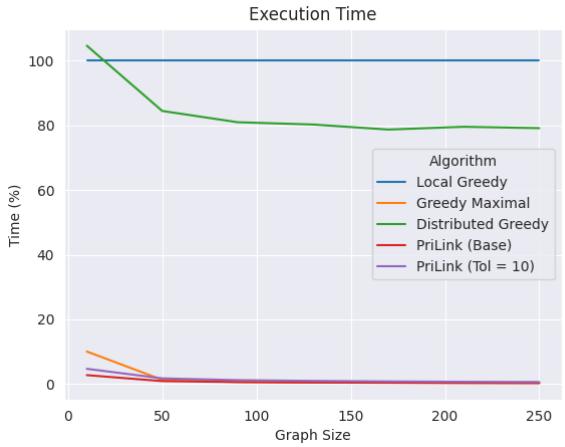

Figure 11: Execution time comparison between PriLink and benchmark algorithms for dense networks ($P = 0.8$).

Using the percentage value simplifies comparison since algorithms like LGS are quadratic, and plotting their link schedules directly will result in severe deviations between algorithms that differ in time complexities. We also acknowledge that the implementations of LGS and DGS are centralized[1] even though they can be parallelized. This helps test their performance in a centralized setting like the one we defined in Section 4.1.

Analyzing the figures, we observe that the PriLink algorithms (both base and privacy tolerance) outperform all other benchmark algorithms. The LGS algorithm is the slowest among all benchmark algorithms. On the other hand, the centralized GMS algorithm is quite efficient for link scheduling computations. We also observe that the PriLink privacy tolerance algorithm is almost similar in execution time to the PriLink base algorithm. One primary reason

[1]Source code for LGS, GMS, and DGS were implemented by [33] and available at https://github.com/zhongyuanzhao/distgcn

is that, even with high tolerance values, only a small percentage of devices transmit their second, third, or higher weighted links. For example, in a graph size of 250, about 75%-80% of devices send just one link - their highest weighted link. As such, the number of devices and links executed by Algorithm 2 in the second or higher loop is low, adding minimal additional computation. Our results indicate that the PriLink privacy tolerance algorithm with a tolerance value 10 can compute link schedules very efficiently, averaging about 0.2 seconds for a graph size of 250. These tests were performed on a Dell Latitude desktop with 16 CPUs, 16 GB RAM, and Intel Core I7 processors. For small graph sizes 10 and 50, the average execution times are almost instantaneous at 0.0005 seconds and 0.0065 seconds, respectively.

5.3.3 Privacy Costs Evaluation Results. This evaluation measures the impact of the privacy tolerance values on the privacy costs in the PriLink algorithm, for different graph sizes and densities. As noted earlier, a lower privacy cost indicates higher privacy benefits since devices can hide more information from a central server. We do not evaluate this metric for other greedy benchmark algorithms because the privacy costs with other algorithms are always 100% as they need the entire wireless topology and link weights information to calculate the link schedule.

Figure 12 shows the average privacy costs associated with the PriLink base and privacy tolerance algorithms, using the probability $P = 0.2$ denoting sparse wireless networks. For this evaluation, we chose five different values for the PriLink privacy tolerance algorithm (2, 4, 6, 8, and 10) to show the impact on privacy costs for these values. All other simulation parameters were the same, as shown in Table 1. In addition, we chose the PriLink base algorithm to show the lowerbound on privacy costs and upperbound on privacy benefits. Note that the PriLink base algorithm is essentially the same as the PriLink privacy tolerance algorithm with a tolerance value of 1. Returning to the figure, observe that the lower percentage of disclosed links means the same as lower privacy costs since the privacy cost measures the percentage of disclosed links out of the total links. We can see that the PriLink base results outperform all other tolerance results for privacy costs. However, the privacy tolerance results also exhibit low privacy costs indicating that the privacy benefits of the algorithm uphold even when large tolerance values are used. For example, considering the graph size of 250, the difference in privacy costs between tolerance value 1 (base) and tolerance value 10 is less than 1%, which is minor privacy leakage considering the scheduling performance gains we observed in Section 5.3.1. Also, the privacy costs reduce significantly for large graph sizes compared to small graphs. For example, using the tolerance value of 10, we can see that the privacy costs for a small graph of size 10 are almost 56%. However, this cost reduces significantly to 15% for a graph size of 50 and becomes even lower than 5% for a larger graph size of 250.

Figure 13 shows the privacy costs for the probability $P = 0.8$ denoting dense wireless networks. All other simulation parameters are the same, as shown in Table 1. We see that the privacy costs are even lower for dense networks, signifying the high privacy benefits of using the PriLink algorithm for dense networks. For example, the privacy costs for a graph size of 10 in sparse networks was about 56% which was reduced significantly to 20% in dense networks.

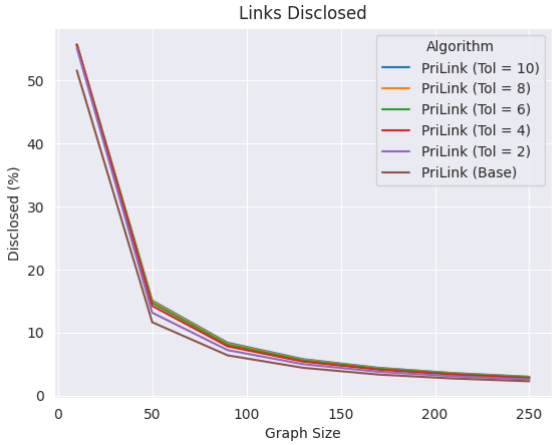

Figure 12: Impact of different privacy tolerance values on privacy protections using the PriLink algorithm for sparse networks ($P = 0.2$).

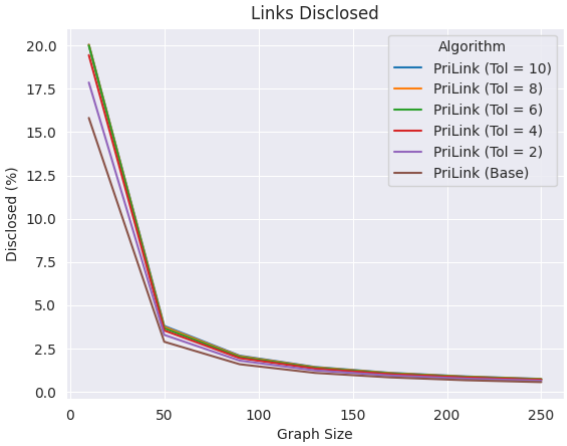

Figure 13: Impact of different privacy tolerance values on privacy protections using the PriLink algorithm for dense networks ($P = 0.8$).

The results are even more pronounced for large graph size 250, where the privacy costs are 5% for sparse networks and reduced to only about 1% for dense networks. These results validate that the PriLink algorithm is a viable alternative to the greedy benchmarks when privacy protection of the wireless network topology and link weights is an important objective.

5.3.4 Impact of Privacy Tolerance Values. This section aims to analyze the impact of privacy tolerance values on the link scheduling performance of PriLink. Recall that in Section 5.3.1, we showed that the PriLink base algorithm's scheduling performance is suboptimal. We also showed that the PriLink privacy tolerance algorithm with a tolerance value of 10 reported results that approached the performance of the Local Greedy Scheduling algorithm.

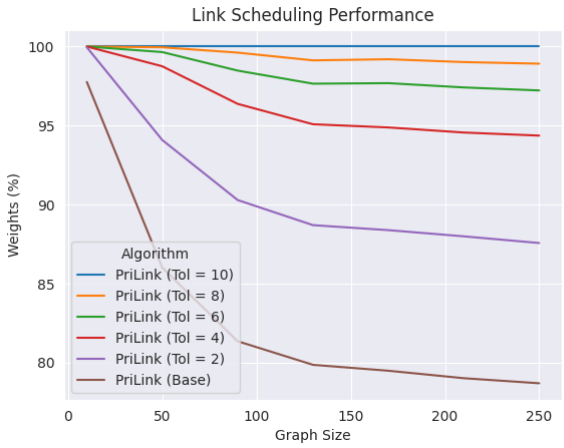

Figure 14: Impact of different privacy tolerance values on link scheduling performance using the PriLink algorithm for sparse networks ($P = 0.2$).

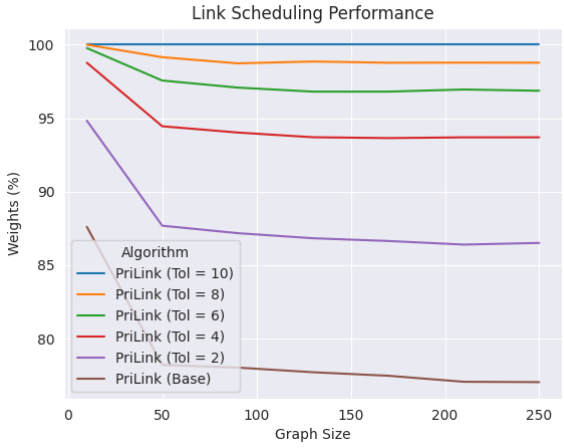

Figure 15: Impact of different privacy tolerance values on link scheduling performance using the PriLink algorithm for dense networks ($P = 0.8$).

Figures 14 and 15 show the setup and results of our evaluation of the impact of privacy tolerance on link scheduling performance for sparse ($P = 0.2$) and dense networks ($P = 0.8$), respectively. For this evaluation, we chose the privacy tolerance values of 2, 4, 6, 8, and 10 to demonstrate the impact on the scheduling performance from these tolerance values. We also use these values to identify whether the results converge at a specific privacy tolerance. Additionally, we chose the PriLink base algorithm as it serves as the lowerbound of scheduling performance for PriLink. We ran the evaluation using the same simulation settings as all previous experiments, as shown in Table 1. Analyzing Figures 14 and 15, we see the expected result that the scheduling performance for both sparse and dense networks improves as the tolerance values increases. This is because the algorithm gets access to a higher count of links, enabling it to compute more efficient schedules. We also see that the improvement is significant for lower tolerance values (e.g., 1, 2, and 4) and then diminishes for higher values (e.g., 6, 8, and 10). The reason for this is that the number of unscheduled devices significantly decreases with each loop, and this causes the algorithm to identify a lower count of potential links to schedule at higher tolerance values. Even though the algorithm performance may improve further if we keep increasing the privacy tolerance values, we ran all our experiments with a value of 10 as it provided a good balance of scheduling performance and privacy protection. We note that identifying an ideal privacy threshold value is outside the scope of this work since it will depend on the scenario of the wireless network and its privacy requirements.

6 LIMITATIONS & FUTURE WORK

To our knowledge, the focus on privacy protection in the PriLink algorithm limits the usage of the algorithm in one scenario where greedy link scheduling algorithms have recently been utilized. For example, one recent effort [33] on Graph Neural Networks (GNNs) for link scheduling utilizes the Local Greedy Scheduling algorithm to train a model for learning the topological weights of links in the network. Our PriLink algorithm cannot be substituted in this scenario for privacy protection because our algorithm does not use a graph for computing link schedules. This limitation would also be valid for other ML-based implementations where the expectation is for graphs to be used for link scheduling. One of our future objectives is to study whether Deep Learning approaches can be designed (e.g., using Federated Learning) for link scheduling extending our privacy-preserving PriLink algorithm, where we can train effective models to improve the scheduling performance and do so in a privacy-preserving manner.

7 CONCLUSION

This paper proposes a privacy-preserving link scheduling algorithm called PriLink for multihop wireless networks. PriLink is designed with built-in privacy protections as the entire network topology is not shared with a central server, and devices share only links required for computing the schedule. To our knowledge, PriLink is the first known implementation of a privacy-preserving link scheduling algorithm. It can be used instead of existing greedy approaches where the privacy of the wireless network is important. Through network simulations and comparisons with high-performing greedy algorithms, we show that the algorithm achieves good link scheduling performance, has faster execution times than existing greedy benchmarks, and does this in a privacy-preserving manner. The PriLink privacy evaluation shows that it can hide nearly 85% of network links for networks containing 50 devices and more than 95% in large networks comprising 250 devices achieving significant privacy benefits in wireless link scheduling.

REFERENCES

[1] Khaled Ali Abuhasel and Mohammad Ayoub Khan. 2020. A secure industrial internet of things (IIoT) framework for resource management in smart manufacturing. *IEEE Access* 8 (2020), 117354–117364.

[2] Bodhi Chakraborty, Shekhar Verma, and Krishna Pratap Singh. 2020. Temporal differential privacy in wireless sensor networks. *Journal of Network and Computer Applications* 155 (2020), 102548.

[3] Wei Cui, Kaiming Shen, and Wei Yu. 2019. Spatial deep learning for wireless scheduling. *ieee journal on selected areas in communications* 37, 6 (2019), 1248–1261.

[4] Ahmed Douik, Hayssam Dahrouj, Tareq Y Al-Naffouri, and Mohamed-Slim Alouini. 2017. Distributed hybrid scheduling in multi-cloud networks using conflict graphs. *IEEE Transactions on Communications* 66, 1 (2017), 209–224.

[5] Peng Du and Yuan Zhang. 2016. A new distributed approximation algorithm for the maximum weight independent set problem. *Mathematical Problems in Engineering* 2016 (2016).

[6] Paul Erdős and Alfréd Rényi. 1961. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica* 12, 1 (1961), 261–267.

[7] Fuad A Ghaleb, Anazida Zainal, Murad A Rassam, and Fathey Mohammed. 2017. An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications. In *2017 IEEE conference on application, information and network security (AINS)*. IEEE, 13–18.

[8] Manan Gupta, Anil Rao, Eugene Visotsky, Amitava Ghosh, and Jeffrey G Andrews. 2020. Learning link schedules in self-backhauled millimeter wave cellular networks. *IEEE Transactions on Wireless Communications* 19, 12 (2020), 8024–8038.

[9] Libin Jiang and Jean Walrand. 2008. A distributed algorithm for optimal throughput and fairness in wireless networks with a general interference model. (2008).

[10] Changhee Joo. 2008. A local greedy scheduling scheme with provable performance guarantee. (2008), 111–120.

[11] Changhee Joo, Xiaojun Lin, Jiho Ryu, and Ness B Shroff. 2015. Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels. *IEEE/ACM Transactions on Networking* 24, 3 (2015), 1476–1488.

[12] Changhee Joo, Xiaojun Lin, and Ness B Shroff. 2009. Greedy maximal matching: Performance limits for arbitrary network graphs under the node-exclusive interference model. *IEEE Trans. Automat. Control* 54, 12 (2009), 2734–2744.

[13] Changhee Joo, Xiaojun Lin, and Ness B Shroff. 2009. Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks. *IEEE/ACM Transactions on Networking* 17, 4 (2009), 1132–1145.

[14] Changhee Joo and Ness B Shroff. 2011. Local greedy approximation for scheduling in multihop wireless networks. *IEEE Transactions on Mobile Computing* 11, 3 (2011), 414–426.

[15] Brian Kim, Yalin E Sagduyu, Kemal Davaslioglu, Tugba Erpek, and Sennur Ulukus. 2021. Channel-aware adversarial attacks against deep learning-based wireless signal classifiers. *IEEE Transactions on Wireless Communications* 21, 6 (2021), 3868–3880.

[16] Joongheon Kim, Giuseppe Caire, and Andreas F Molisch. 2015. Quality-aware streaming and scheduling for device-to-device video delivery. *IEEE/ACM Transactions on Networking* 24, 4 (2015), 2319–2331.

[17] Jing Yang Koh, Derek Leong, Gareth W Peters, Ido Nevat, and Wai-Choong Wong. 2017. Optimal privacy-preserving probabilistic routing for wireless networks. *IEEE Transactions on Information Forensics and Security* 12, 9 (2017), 2105–2114.

[18] Sebastian Lamm, Christian Schulz, Darren Strash, Robert Williger, and Huashuo Zhang. 2019. Exactly solving the maximum weight independent set problem on large real-world graphs. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 144–158.

[19] Mathieu Leconte, Jian Ni, and Rayadurgam Srikant. 2009. Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks. (2009), 165–174.

[20] Yi Liu, JQ James, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. 2020. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal* 7, 8 (2020), 7751–7763.

[21] Antonio G Marques, Nikolaos Gatsis, Georgios B Giannakis, N Zorba, C Skianis, and C Verikoukis. 2011. Optimal cross-layer design of wireless fading multi-hop networks. *Cross Layer Designs in WLAN Systems* (2011), 1–44.

[22] Mohamed Seif Eldin Mohamed, Wei-Ting Chang, and Ravi Tandon. 2021. Privacy amplification for federated learning via user sampling and wireless aggregation. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3821–3835.

[23] Jian Ni and R Srikant. 2009. Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks. (2009), 250–250.

[24] Elif Ustundag Soykan, Gurkan Soykan, Emrah Tomur, and Mete Akgün. 2023. A Privacy-Preserving Scheme For Smart Grid Using Trusted Execution Environment. *IEEE Access* (2023).

[25] Jian Tang, Guoliang Xue, Christopher Chandler, and Weiyi Zhang. 2006. Link scheduling with power control for throughput enhancement in multihop wireless networks. *IEEE Transactions on Vehicular Technology* 55, 3 (2006), 733–742.

[26] Xiangyun Tang, Meng Shen, Qi Li, Liehuang Zhu, Tengfei Xue, and Qiang Qu. 2023. PILE: Robust Privacy-Preserving Federated Learning via Verifiable Perturbations. *IEEE Transactions on Dependable and Secure Computing* (2023).

[27] Taha Ameen ur Rahman, Mohamed S Hassan, and Mahmoud H Ismail. 2020. A Queue-Length Based Approach to Metropolized Hamiltonians for Distributed Scheduling in Wireless Networks. In *2020 Wireless Telecommunications Symposium (WTS)*. IEEE, 1–6.

[28] Kilian Verhetsela, Jeanne Pellerina, Amaury Johnena, and Jean-Francois Remaclea. 2017. Solving the Maximum Weight Independent Set Problem: Application to Indirect Hexahedral Mesh Generation. *26th International Meshing Roundtable, Research Notes, Barcelona, Spain* (2017).

[29] Hao Wang, Guangjie Han, Wenbo Zhang, Mohsen Guizani, and Sammy Chan. 2019. A probabilistic source location privacy protection scheme in wireless sensor networks. *IEEE Transactions on Vehicular Technology* 68, 6 (2019), 5917–5927.

[30] Sulei Wang, Zhe Chen, Yuedong Xu, Qiben Yan, Chongbin Xu, and Xin Wang. 2019. On user selective eavesdropping attacks in MU-MIMO: CSI forgery and countermeasure. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1963–1971.

[31] Xue Yang, Yan Feng, Weijun Fang, Jun Shao, Xiaohu Tang, Shu-Tao Xia, and Rongxing Lu. 2022. An accuracy-lossless perturbation method for defending privacy attacks in federated learning. In *Proceedings of the ACM Web Conference 2022*. 732–742.

[32] Shuai Zhang, Wenlong Shen, Max Zhang, Xianghui Cao, and Yu Cheng. 2019. Experience-driven wireless D2D network link scheduling: A deep learning approach. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[33] Zhongyuan Zhao, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. 2021. Distributed scheduling using graph neural networks. *ieee journal on selected areas in communications* 37, 6 (2021), 4720–4724.

[34] Zhongyuan Zhao, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. 2022. Link scheduling using graph neural networks. *IEEE Transactions on Wireless Communications* (2022).

[35] Zhongyuan Zhao, Gunjan Verma, Ananthram Swami, and Santiago Segarra. 2022. Delay-Oriented Distributed Scheduling Using Graph Neural Networks. (2022), 8902–8906.