Journal of
Sensor and Actuator Networks

Article

SDMob: SDN-Based Mobility Management for IoT Networks

Iliar Rabet [1,*], Shunmunga Priyan Selvaraju [1], Hossein Fotouhi [1], Mário Alves [2], Maryam Vahabi [1,3], Ali Balador [4] and Mats Björkman [1]

[1] School of Innovation, Design and Engineering, Mälardalen University, 721 23 Västerås, Sweden; shunmuga.selvaraju@mdh.se (S.P.S.); hossein.fotouhi@mdh.se (H.F.); maryam.vahabi@mdh.se (M.V.); mats.bjorkman@mdh.se (M.B.)
[2] Politécnico do Porto, School of Engineering (ISEP/IPP), 4249-015 Porto, Portugal; mjf@isep.ipp.pt
[3] ABB Corporate Research, 72226 Västerås, Sweden
[4] RISE Research Institutes of Sweden, 72212 Västerås, Sweden; ali.balador@ri.se
* Correspondence: iliar.rabet@mdh.se

Abstract: Internet-of-Things (IoT) applications are envisaged to evolve to support mobility of devices while providing quality of service in the system. To keep the connectivity of the constrained nodes upon topological changes, it is of vital importance to enhance the standard protocol stack, including the Routing Protocol for Lossy Low-power Networks (RPL), with accurate and real-time control decisions. We argue that devising a centralized mobility management solution based on a lightweight Software Defined Networking (SDN) controller provides seamless handoff with reasonable communication overhead. A centralized controller can exploit its global view of the network, computation capacity, and flexibility, to predict and significantly improve the responsiveness of the network. This approach requires the controller to be fed with the required input and to get involved in the distributed operation of the standard RPL. We present SDMob, which is a lightweight SDN-based mobility management architecture that integrates an external controller within a constrained IoT network. SDMob lifts the burden of computation-intensive filtering algorithms away from the resource-constrained nodes to achieve seamless handoffs upon nodes' mobility. The current work extends our previous work, by supporting multiple mobile nodes, networks with a high density of anchors, and varying hop-distance from the controller, as well as harsh and realistic mobility patterns. Through analytical modeling and simulations, we show that SDMob outperforms the baseline RPL and the state-of-the-art ARMOR in terms of packet delivery ratio and end-to-end delay, with an adjustable and tolerable overhead. With SDMob, the network provides close to 100% packet delivery ratio (PDR) for a limited number of mobile nodes, and maintains sub-meter accuracy in localization under random mobility patterns and varying network topologies.

Keywords: Internet of Things; Software Defined Networking (SDN); mobility management; localization; Kalman filter; particle filter; Contiki; COOJA; RPL; 6LoWPAN

check for updates

Citation: Rabet, I.; Selvaraju, S.P.; Fotouhi, H.; Alves, M.; Vahabi, M.; Balador, A.; Björkman, M. SDMob: SDN-Based Mobility Management for IoT Networks. *J. Sens. Actuator Netw.* **2022**, *11*, 8. https://doi.org/10.3390/jsan11010008

Academic Editor: Mohamed Elhoseny

Received: 10 December 2021
Accepted: 17 January 2022
Published: 21 January 2022

1. Introduction

With the advent of the Internet-of-Things (IoT) and its revolutionary role in numerous application domains such as healthcare, industrial automation, and environmental monitoring, there is an increasing demand for seamless support of mobile nodes (MNs). However, the de facto protocols for low-power and lossy networks (LLNs), including Routing Protocol for Lossy Low-power Networks (RPL) and IPv6 over Low -Power Wireless Personal Area Networks (6LoWPAN), have not been designed to cope with highly dynamic network topologies. These protocols are not able to handle rapid topological changes in the network in a timely and accurate manner. Further, it has been shown that various mobility patterns impact the performance of the standard RPL protocol significantly [1]. This occurs due to the continuous relocation of mobile nodes and the delayed readjustments of RPL by the 'trickle' algorithm. In practice, the trickle algorithm is responsible for adapting the transmission frequency of control packets to the rate of the topological changes [2]. Increasing the control packet rate could result in better responsiveness to mobility, but at the cost of higher resource consumption in terms of communication and energy overheads. Nevertheless, since no predictive measure is taken, the mobile nodes' routes are only updated after a period of disconnection, leading to network inaccessibility periods that will cause packets loss and higher delays.

A proactive approach to support seamless hand-off in MNs could rely on Bayesian filters (such as Kalman filters) or other predictive techniques to forecast the future position of MNs [3]. Here, a filter is referred to as the methods that estimate the state of a temporal variable, which is usually observed under noisy measurements [4]. It is common to have a fixed infrastructure of static nodes (in fixed and a priori known positions) that estimate the distance from the MNs. MNs are also usually assisted by an *Inertial Measurement Unit* (IMU) to measure velocity and direction of movement. In such a scenario, it is practical to exploit Bayesian filters to fuse these two sources of information and, thereby, benefit from an accurate localization which leads to improvement in network responsiveness [5].

The Kalman filter has been proved to be unbiased (the average error across all the recursive runs is zero), consistent (the filter is neither overconfident nor under-confident) and optimal (it minimizes the estimation error) [4]. However, when using a Kalman filter, the posterior distribution (after the observations) can be computed in closed form only when the relationship between states and observations is described by a linear function, and the measurement and prediction noise follow a Gaussian distribution [6]. To address nonlinear system models, *Extended Kalman Filter* (EKF) was introduced which uses *Taylor series* to linearize the equations, trading for a negligible approximation error. On the other hand, Unscented Kalman Filter (UKF) and Particle filter have shown higher accuracy in prediction with the bi-modal distribution of the noise [4]. In the literature, these filters are integrated with low power routing protocols mostly in distributed environments [7–9].

The overall architecture of distributed routing algorithms with location prediction in the literature is depicted in Figure 1a, where the anchors report back to the MN so it can predict its future parent on its own and modify its routing state accordingly. In this scenario, the accuracy of the location prediction significantly impacts the connectivity of the MN, and to implement accurate and predictive models, we require higher computation capacity than what mainstream IoT devices can afford. In practice, the resource-constrained devices barely manage to support simple data filters, such as Kalman filters. The next important limitation is flexibility in configuration, as the position information is required as an input for the localization of MN, and this can be done much easier with the SDN approach compared to a distributed situation where each node should be fed with this information. We argue that these limitations can be alleviated by offloading the computational burden of these data filters to a centralized external entity, such as an edge/fog device or an external Software-Defined Networking (SDN) controller, depicted in Figure 1b. These centralized devices have sufficient resources while being accessible for IoT devices with reasonable latency.

In our context, an edge or fog device is defined as a one-hop resource or service provider. It is placed in close proximity to the client devices to access their resources at a low delay and low accessibility cost. In recent years, there has been a roll-out of edge-based services to improve network reliability, services, and reactivity in the rapidly growing field of IoT applications. Specifically, it has been commonly used by external proprietary real-time localization systems (RTLS) such as Ubisene and Sewio [10] to be implemented in a centralized device at the network edge. SDMob can be integrated into such systems and exploit the information gathered through the distributed operation similar to RPL. However, the involvement of an external SDN controller can also raise new challenges such as additional control packets, leading to control overhead and reservation of system resources to robustly handle them [11].

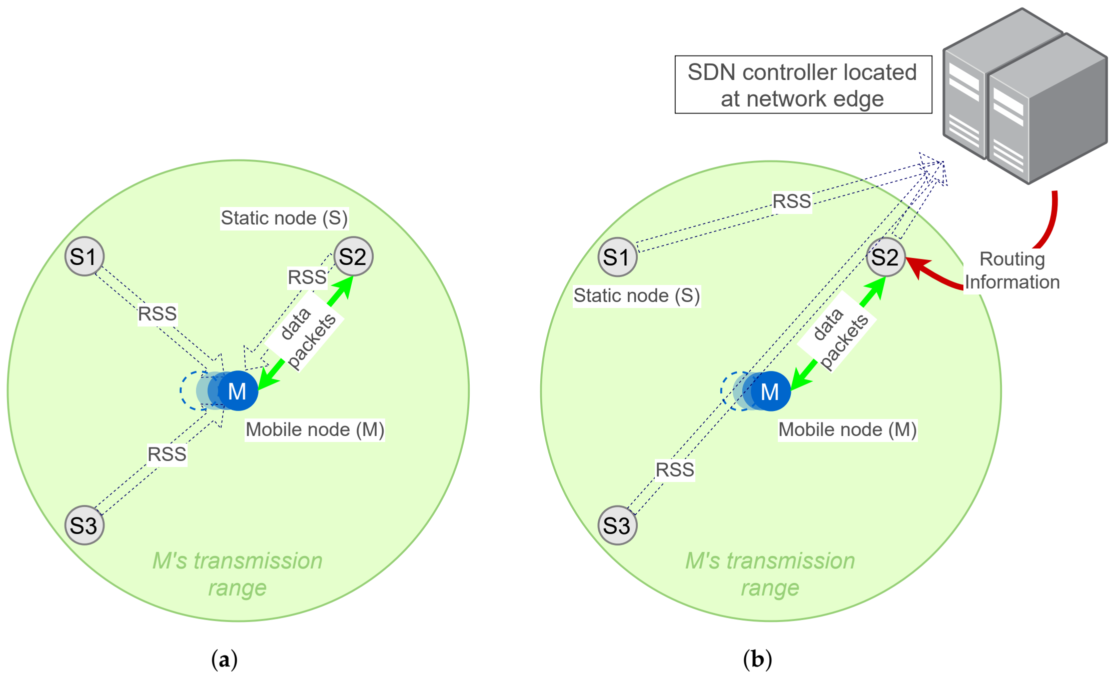


Figure 1. Illustration of mobility management strategies—(**a**) Distributed strategy—Mobile node (*M*) does the routing function by processing the Received Signal Strength (*RSS*) values from neighbouring Static nodes (*S*); and (**b**) Centralized strategy—SDN controller at network edge picks the next best parent for *M* to relay data packets.

This paper extends our previous work [12], where we had compared SDMob—*Software Defined Mobility management*, powered by either particle filter or unscented Kalman filter, with the baseline RPL and mRPL [13]. Our previous work only partially addressed mobility patterns (linear and circular trajectories), and scalability of the system in different aspects including the number of MNs, anchors, and their topology (density and distance) were not analyzed. In this work, we introduce new mechanisms to handle congestion in dense networks and to support multiple mobile nodes for enabling seamless handoffs in more realistic scenarios. The new mechanisms include smart buffer management to support multiple MNs, new timers for congestion and restructured route-installation packet format. We also evaluate the impact of other system parameters such as path loss variance and the velocity of the MNs. In this work, we have also included the analytical evaluation of the system in comparison with the simulation results.

The main contributions of this work are listed below:

- Enhancement of the SDMob architecture by introducing new timers and route installation format to achieve a better quality of service (QoS) in networks with varying topologies, realistic mobility patterns, and supporting multiple MNs.
- Comparison of SDMob with the benchmark ARMOR and baseline RPL.
- Modelling of the SDMob architecture through probabilistic analysis and comparing the analytical results with the simulation results.
- Evaluation of SDMob in various conditions, considering different mobility patterns, link quality fluctuations as well as network scalability in terms of hop distance from the MN to the controller, number of neighbors, and number of mobile nodes.
- Implementation of SDMob in the Cooja/Contiki environment, where the code is available online https://github.com/iliar-rabet/sdmob, (accessed on 9 December 2022).

The rest of the paper is organized as follows: Section 2 provides a brief description of RPL limitations upon mobility, and outlines some efforts to overcome them. Section 3 describes the SDMob architecture, and the employed filter for data processing. Section 4 takes an analytical approach to evaluate the system and measure the probability of packet loss during handoff. Section 5 describes the simulation environment and performance evaluation, comparing SDMob with the selected benchmark (ARMOR). Finally, in Section 6 conclusions are drawn.

2. Related Works

This section provides an overview of some of the main related works, which are specifically focused on mobility support in RPL, localization algorithms within the routing protocol, and edge or fog computing architecture in IoT networks enabled with SDN devices.

2.1. Overview of RPL.

RPL is considered as the de facto routing protocol for IoT. RPL maintains a distributed data structure, named Destination Oriented Directed Acyclic Graph (DODAG). The process starts with the root node transmitting a DODAG Information Object (DIO) that embeds the necessary information to construct a route towards the root (rank 0 for root node). Upon receiving the DIO packet, each node selects its preferred parent (based on some objective function) and schedules relaying a DIO packet with its non-decreasing rank to further advertise the network. Upward traffic can be routed after DIO packet transmission, but for downward routing, the root node (or parent in storing mode) gets notified about the high-rank nodes only after transmission of Destination Advertisement Object (DAO) packets.

RPL allows two modes of operation—storing and non-storing—for downstream traffic. In non-storing mode, it is only the root that maintains the downward routes. This mode scales better, since the memory footprint at intermediate nodes does not increase with the size of the network. In RPL, it is more challenging to support mobility for downstream traffic since a mobile node must notify the root (rather than only updating its parent for upstream traffic). SDMob uses the non-storing mode of RPL so that the controller can manipulate the source-routed downward packets.

In the original RPL, the most common Objective Functions (OFs) are (1) the Minimum Rank Hysteresis Objective Function (MRHOF), and (2) Objective Function Zero (OF0). Objective functions define how RPL nodes minimize the given routing metric: hop count, expected transmission count (ETX), or latency. MRHOF is designed to minimize the path cost while avoiding excessive churns in the network. The nodes using MRHOF only switch the best parent if the minimum is improved by a threshold (this is also known as the hysteresis mechanism) [14]. OF0 is faster in terms of updating the routes in a dynamic environment as it changes the best parent even after slight improvements in the routing metric [15]. Our proposed mobility solution (SDMob) does not enforce any specific requirement on the objective function or the routing metric. Rather, SDMob allows the baseline version of RPL to converge with arbitrary metrics and then injects higher priority routes from the controller.

2.2. Mobility-Aware RPL Routing

Mobility creates an inconvenience for all layers of the protocol stack, especially the routing protocol. While RPL supports infrequent joining and leaving of nodes, it performs poorly upon the dynamics imposed on the network topology. The authors in [16], survey the enhancements made to RPL to support mobility and focus on the specific mechanisms that have been altered. A more recent survey [17] classifies mobility extensions of RPL into solutions including (i) only mobile nodes (e.g., VANETs) and (ii) those with a fixed infrastructure as well as mobile nodes.

One of the extensions of RPL that addresses mobility of nodes is BRPL [18], which combines backpressure routing [19] with the objective functions of RPL. To support high-throughput traffic, BRPL takes into consideration the queue backlogs of the neighbors. This allows BRPL to utilize sub-optimal routes when the optimal route is congested.

In mRPL [13], MN operates in two phases of data transmission and discovery. In the *Data Transmission phase*, static nodes (SNs) constantly monitor the link quality and compare the Received Signal Strength Indication (RSSI) measurements to a threshold T_l that indicates the minimum RSSI threshold in a reliable channel. If the link quality crosses the threshold, the SNs notify the MN with a beacon, and then MN stops transmitting data packets and instead starts the *Discovery Phase* by sending DIS packets to request the neighboring SNs to respond with DIO packets. Then the MN analyses the received DIO packets, and if the RSSI values from other SNs are higher than T_h, it performs the handoff process and resumes data transmission, otherwise, it continues sending DIS packets. An enhanced version called mRPL+ [20], relies on overhearing of mobile nodes' packets by alternative parents. Then mRPL+ takes a similar approach to mRPL by selecting the preferred parent based on some thresholds.

For the case where all nodes are mobile, Tian et al. [21] adjust the Trickle timer according to mobile nodes' velocity, and utilize geographical information as the routing metric. If the mobile nodes are not equipped with IMU sensors, *Doppler Effect* can be used to estimate their velocity as explained in [22]. Murali et al. [23] introduce D-trickle to support mobility, where the chosen DIO interval depends on the number of neighboring nodes. In a dense network, D-trickle sends DIO packets less frequently but for a sparse network, frequent DIO packets boost the connectivity of the network.

Ancillotti et al. [24] propose RL-Probe for using reinforcement learning methods to determine when to send probing packets to estimate link qualities. Using RL and multi-armed bandit theory in RPL minimizes the communication overhead while keeping the network responsiveness at a high level. Another approach has been implemented in GTM-RPL, which is based on game theory to select the optimal transmission rate of the nodes [25]. The authors prove the existence of Nash Equilibrium. In other words, each node can reach an optimal strategy with no incentive to change while other nodes keep their current strategy. In the next step, nodes select the preferred transmission rate based on the mobility of nodes (detected by RSSI) and other parameters. This approach is only practical in applications where the transmission rates can be manipulated.

ARMOR [26] is a recent work that calculates the time each parent is available based on location data and selects the node with the so-called longest "Time-To-Reside". We use ARMOR as a benchmark in our work, and we will further explain it later in Section 5.1. RMA-RP [27] introduces a similar metric called "Time-To-Stay" using only two recent RSSI measurements. The choice of this metric is debatable since these two models may choose links that are active for a long time but are lossy since stability is interpreted with the duration of the connection rather than the link quality. This metric manages to reduce the number of performed handoffs and preserves energy consumption. RMA-RP also modifies the DIO intervals to be lower for low-rank nodes (close to the root) as they provide connectivity for the other nodes.

2.3. Location Estimation Models for Enhancing Routing Protocols

A class of extensions of RPL consists in boosting the mobility support by integrating a filtering/localization method into the routing protocol. For instance, in [7], authors have proposed Kalman-RPL to predict the future position of the mobile node and consequently estimate the future link qualities. In Kalman-RPL a mobile node transmits a beacon that includes its velocity information in specific intervals and is responded to by the receiving static nodes. After a positioning phase that estimates the current position of the mobile node using three static nodes in its vicinity, MN can predict its future position. EKF-RPL [9] takes a similar approach to Kalman-RPL, but it employs Extended Kalman Filter (EKF) within RPL to better support non-linear trajectories.

Some research works apply the Kalman filter for predicting link quality regardless of the routing protocol. Parasuraman et al. [28] model path loss and shadow fading of the wireless channel independently and then apply Kalman filter to fuse both models.

There are also some efforts on adopting *on-demand* routing strategies when a node starts searching for a route for transmitting data. The *Lightweight On-Demand Adhoc Distance-vector* routing protocol - Next Generation (LOADng) [29] is one such protocol specifically designed to support any-to-any communication in LLNs, which has received less attention compared to RPL. EKF-LOADng [8] enhances basic LOADng by predicting the link's RSSI after a positioning phase (triangulation algorithm) and running the EKF. In the triangulation phase, a mobile node broadcasts a message asking for packets from its static neighbors. Responses from static nodes experience a random waiting time to avoid collisions.

Compared to the EKF, particle filter leads to more accurate results and better resiliency against nonlinear moving trajectory and non-Gaussian noise [4]. The particle filter, which is also known as *Sequential Monte Carlo* uses hundreds to thousands of samples to predict the future state and fuse the measurement, hence requiring a higher computation capacity than most constrained IoT nodes can provide. In the context of cellular communications, the particle filter has been used in the literature to spot the mobile nodes and select the best service point as in [30]. This work also proposes a Rao-Blackwellised particle filter as a lightweight alternative to the baseline particle filter.

In a previous work [12], we proposed and implemented the basic version of SDMob for offloading the mobility solution to the edge devices in a centralized manner. The basic SDMob outperformed RPL and mRPL [13] in a topology with a single MN. This paper presents enhanced SDMob, supporting networks with multiple mobile nodes, high density of anchors, varying hop distances, and realistic mobility patterns.

2.4. SDN-Enabled IoT Network Architectures

There is growing popularity in using SDN-enabled solutions in IoT networks and a recent survey [31] addresses the proposals implemented by the community to integrate the SDN in IoT networking in a coherent and lossless manner. It is too expensive to simply integrate the common SDN solutions and standards within constrained IoT networks without re-designing the SDN to consider IoT limitations [32]. Therefore, there is a requirement for devising solutions targeting IoT networks with reduced complexity and operational cost.

Efforts have been made (including by standardization bodies) to design solutions for managing IoT networks. The *Internet Engineering Task Force* (IETF) has a recent draft for infusing data routes into the network that is called DAO projection [33]. It defines a framework for the root node to initialize some options in DODAG Advertisement Objects (DAO) through new control messages, namely Project DAO Request (PDR) and PDR-Acknowledgement (PDR-ACK). This enables the root node to install routes in either the source or intermediate nodes along the path. The mechanism is a low-overhead substitute for implementing centralized network management in IoT networks. We have presented a similar implementation of this draft in RPL-RP [34] that aims at optimizing any-to-any routes.

Coral SDN [35] is another RPL-based solution that allows an SDN controller to manipulate RPL routing parameters such as the interval used by the Trickle timer. The interval is the duration between successive DIS messages from a leaf node, which is an important configuration to adapt the responsiveness of the network.

Theodorou et al. [36] proposed SD-MIoT, in which RPL is assisted by an SDN controller. The SDN controller is responsible to detect the mobility of the nodes by maintaining an adjacency graph. The mobility detector assumes the node to be mobile if more than one row in the adjacency graph changes compared to the previous time step, but with a single connectivity change, it can not be determined if the node is mobile. Next, k-means clustering is performed to separate mobile and static nodes. The SDN will then constantly use the adjacency graph to update the forwarding rules.

In μSDN [37], the authors argue that an appropriate design for the centralized controller is to rely on the legacy RPL for basic connectivity. To deal with constraints in IoT networks, it introduced optimizations including avoiding packet fragmentation, source-routed control packets, and timers for fine-tuning.

SDN-WISE [38] is one of the research works on SDN-based low-power networks that has attracted some attention. It installs finite state machines on the constrained nodes to handle the rules. For connecting the controller to the mesh network it adds a layer called Topology Discovery that is responsible to interpret packets from neighbors that advertise their hop distance to the sink and remaining battery power. In some other works, such as μSDN, SDMob, and DAO projection, this basic connection is prepared by the RPL protocol. Some extensions to SDN-WISE try to address mobility by assigning a MAC layer schedule [39] or by using multicast routing, similar to MMF-SDN [40].

Mertens et al. [41] proposed SDN-(UAV)ISE in which a drone acts as a mobile sink (a.k.a mule). The SDN controller, which is based on SDN-WISE applies a decision tree learning algorithm on the data from sensors to predict the position of the drone. They also optimize the destinations that the drone is supposed to visit and this optimization can be reduced to the NP-complete "set cover problem".

MobiFog [42] is our previous work on centralized mobility management, where the discovery of alternative parents is performed using the RSSI measured from data packets. The predictions are independent of the position of the nodes and hence MobiFog does not require prior knowledge about the position of the static nodes. Generally, the approaches that rely on the data traffic including MobiFog and mRPL impose much less overhead, but the handoff process will depend on the traffic at the time of the handoff.

We have summarized the presented works in Table 1. Note that when it is mentioned in the table that all of the nodes can be mobile, it means that the protocol design does not require a set of static nodes, yet it does not mean that the authors have claimed or tested the work in such a scenario. Overall, the literature in IoT networks mostly neglects more sophisticated and computation-intensive filters for mobility management. Employing the SDN architecture to overcome challenges raised by the mobility of nodes remains a research gap that we address by introducing SDMob. We believe that SDMob paves the ground for employing more accurate filter/localization algorithms at the SDN controller towards improved performance upon mobility in IoT networks.

Table 1. Mobility management extensions for RPL.

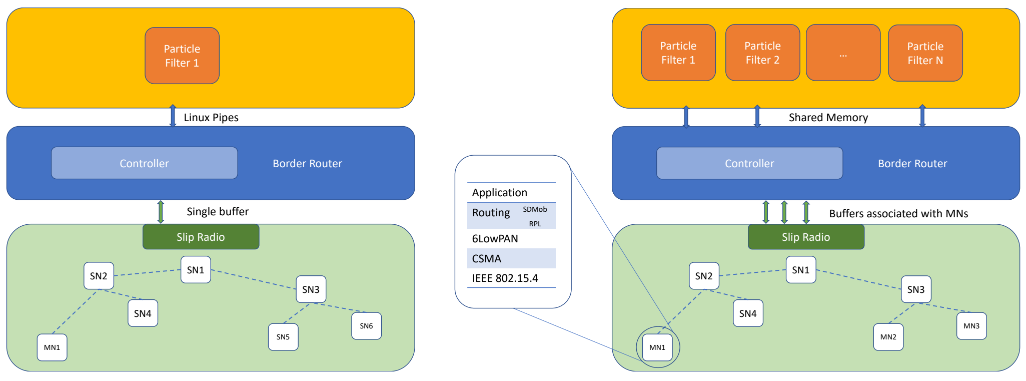
Mobility Solutions	Infrastructure	Metric	Predictive Mechanism
mRPL [13]	fixed and mobile	ETX and RSSI	average RSSI/SNR
mRPL+ [20]	fixed and mobile	ETX and RSSI	average RSSI/SNR (overhearing)
ARMOR [26]	all can be mobile	Time To Reside	Relative velocity
RMA-RP [27]	fixed and mobile	Time To Stay	2 consecutive RSSI values
D-trickle [23]	all can be mobile	ETX,ELT,RSSI,distance	-
Kalman-RPL [7]	fixed and mobile	predicted ETX	Kalman Filter
EKF-RPL [9]	fixed and mobile	position of MN	Extended Kalman Filter
EKF-LOADng [8]	fixed and mobile	position of MN	Extended Kalman Filter
DAO projection [33]	No mobility	priority for projected routes	-
Coral SDN [35]	fixed and mobile	OF and trickle set by controller	-
SD-MIoT [36]	fixed and mobile	link quality	proactive route installation
SDN-UAise [41]	Mobile sink	not RPL-based	Decision tree
MobiFog [42]	fixed and mobile	ETX and RSSI	Average RSSI
MMF-SDN [40]	fixed and mobile	not RPL-based	-
FTS-SDN [39]	fixed and mobile	not RPL-based	-
BRPL [18]	all can be mobile	backlog drift plus ETX	Lyapunov Optimization
GTM-RPL [25]	fixed and mobile	ETX	Nash Equilibrium
RL-Probe [24]	all can be mobile	ETX (same as RPL)	epsilon-greedy learning

3. SDMob Architecture

In this section, the structure of the SDMob architecture is addressed in more detail. First, we review the basic design of SDMob, which was presented in the previous work [12]. Next, we describe the mechanisms introduced in the enhanced SDMob to address the challenges that arise in networks with multiple mobile nodes, high density, and more realistic mobility patterns. For simplicity, we are using SDMob for the enhanced version throughout this paper. Finally, we analyze the tracking algorithm employed in the paper (particle filter).

3.1. Basic SDMob Architecture

In this subsection, we review the basic SDMob architecture that was presented in a previous work [12]. Figure 2 illustrates and compares the basic SDMob against the enhanced version. Inside the WSN network, SDMob uses Contiki's protocol stack including RPL/6LoWPAN networking with a 64-bit network prefix and IEEE 802.15.4. The WSN is composed of one MN in the basic SDMob (multiple MNs in the enhanced SDMob), and Static Nodes (SN). SNs are also known as anchors and provide multi-hop access for the MNs, and provide the controller with the required information for localization. The SDN controller is placed outside of the constrained network, separately as a Linux-server process. It admits to the control information collected via the sink node in the WSN simulated network. The SDN controller is bundled with the border router, which is responsible to decode messages from the low-power domain to be used by the controller and data server. The border router is also a Linux process that is connected to the WSN with the help of a node called Serial Line Internet Protocol (SLIP) radio. The Contiki's implementation of the SLIP protocol prepares the wireless IP packets to be transmitted over a serial line. The particle filter runs the location estimation algorithm to accurately locate future positions of the mobile node. The implementation of the border router and the particle filter are interfaced using Linux pipes (shared memory in the enhanced SDMob due to its fast inter-process communication).



(**a**) The basic version of SDMob (**b**) Enhanced SDMob

Figure 2. A schematic view of the basic SDMob architecture in (**a**) supporting a single mobile node compared with enhanced SDMob (**b**) with multiple mobile nodes and buffers.

In this paper, we assume the MNs are equipped with Inertial Measurement Unit (IMU) sensors, but the architecture can be generalized to nodes that do not benefit from IMU sensors by estimating the velocity based on previous RSSI measurements. However, measuring the velocity is more accurate compared to the estimation techniques, since it exploits the extra information from the sensors. On the other hand, many applications nodes are already using such sensors, so the overhead could be negligible.

This design is highly inspired by the ideas in an IETF draft titled "root-initiated routing state" [33] that allows a centralized entity to manipulate the routing states in the distributed operation of RPL. The routing rules that are installed by the controller are of a higher priority compared to the routes learned by receiving DIO packets. The SDN controller's traffic relies on RPL for its basic connectivity. It is only after the convergence of the standard RPL protocol that the MN can utilize the upward/downward routes to access the controller and the centralized tracking algorithm can notify the anchors to relay MN's traffic.

For the smooth operation of SDMob, certain mechanisms are required. Below we describe some of the challenges and the mechanisms adopted in basic SDMob to address them.

3.1.1. Collision Avoidance between Control Plane and Data Plane

The SDN-based architecture comes with an inevitable control overhead that can saturate the MAC layer congestion mechanism in both upward and downward directions.

To streamline the traffic, we have implemented a reserved period for control packets called Control Window (CW) out of which transmission of control packets is prohibited. CW can be adjusted based on network conditions; for instance in a network with a long delay between the MN and the controller, it is advised to select a longer CW to allow the control packets to reach the controller. A long CW limits the available time for data packets, and hence data packets are expected to be further delayed.

3.1.2. RPL-Aware Leaf (RAL)

In SDMob installing a new rule on the MNs is troublesome since their links are lossy. A recent IETF draft [43] defines RALs as a host that does not participate in further advertising the DODAG and relies on the RPL routers to forward its traffic. This can solve the problem of routing loops that could happen in the network, for example when MN is denoted as the parent node of another node. Another upside is the reduced memory footprint in the MN. Most importantly, it is beneficial as the controller is mandated to send routes to more reliable SNs rather than using MN's lossy links.

3.1.3. Downward Routing Process

Standard RPL favors upstream traffic as it is the predominant traffic pattern in the IoT domain. Most enhancements made to RPL also have weaker behavior when it comes to point-to-point or downward traffic. These shortcomings stem from design choices in RPL. If there is a topological change deep in the network, the root node will only be notified after a timer for sending the DAO packet expires. SDMob relies on the downward routes provided by RPL to relay the controller's commands. It can also reinforce the downward routes by the RSSI measurements that it collects.

3.1.4. Integration to the Objective Functions

Once the controller runs the filter and announces the new best parent, it will start serving the MN in the data window. Selecting the closest parent to the MN can be a naive strategy since the characteristics of the multi-hop path will be ignored. An alternative approach is to estimate the one-hop link quality (reverse relation with distance) and use it by adding to the classic metrics such as ETX.

3.2. Enhanced SDMob

Previously, basic SDMob design has focused merely on offloading the mobility solution to the resource-rich controller, yet it fails at performing the handoff in some challenging conditions. In this paper, we aim at improving and evaluating the scalability of the proposed system. We review those challenging conditions and introduce mechanisms to overcome those challenges. Table 2 summarizes these features and associates them with the versions of SDMob.

3.2.1. Handling Anchor Density

The RSSI measurements that provide the vital information for localization generate bursty traffic since all the anchors try to forward the control beacon as soon as they receive it. The constrained network cannot always deliver this bursty traffic to the controller especially if MN resides in a dense area. Although the CSMA-based MAC layer handles the shared spectrum but with bursty traffic in a dense network the delivery ratio drops significantly. With more burstiness in the traffic, CSMA is forced to retransmit more packets due to congestion in the network. This bursty traffic can push the CSMA-based MAC layer to its limits. If the number of retransmissions for a certain packet crosses a certain threshold, the packet will be dropped. Some mobility patterns tend to keep the mobile node in areas with more density. The situation gets worse with multiple mobile nodes moving in the same area.

The accuracy of the filter substantially depends on the RSSI reports arriving on time. However, this bursty traffic can be lost in the anchors or be delayed. By enabling the re-sampling procedure, the particle filter can converge again after a period of not receiving any/enough measurements. But most importantly, we introduce a timer called the congestion timer. The congestion timer randomly delays the transmission of the RSSI report to the controller in order to decrease the burstiness. This timer can be tuned based on the density of the neighbors but excessively increasing this timer may increase the handover delay. By setting the congestion timer to a reasonable value, the delivery ratio at the control plane gets boosted but on the other hand setting it too high may lead to not receiving the packets in time for the filter.

3.2.2. Handling Multiple MNs (Smart Buffer Management)

One of the most important features of enhanced SDMob is supporting multiple MNs that require multiple instances of the filter to run in parallel. On the other hand, with multiple MNs the control traffic increases dramatically sometimes even more than the capacity of the network. SDMob will require multiple independent buffers at the border router each associated with an individual MN. These buffers assign a priority to the newest reports and discard the old ones.

Another additional feature focuses on improving the efficiency of the inter-process communication between the border router and the filter. In the enhanced SDMob, the buffers (between the border router and the particle filter) are implemented using shared memory, instead of using Linux pipe files that were utilized in the basic SDMob. This feature drastically decreases the time spent for predicting the future parent and handoff delay, thus improving network reliability.

To increase the reception ratio of the control traffic, we implement a buffer timer that specifies the time that the border router waits for RSSI measurements. This timer can be tuned in accordance with the hop-distance of the mobile node, congestion in the network, congestion timer, and the interval for sending the control beacons. Each MN will have a buffer timer that can be tuned independently.

3.2.3. New Route Projection Packet Format

The packet format of the controller has some impact on the handoff process, mainly through notifying the previous parents. It is important to stop the previous parent from serving the MN immediately after switching to a new parent otherwise two parents will forward the data packets. This may be useful if the number of redundant parents does not increase in a way that too many replicas of the same packet congest the network. In some of the related works, uninstalling root-initiated routes is performed using timers, meaning that the routes are active for a specific time set by the controller. In our previous work, the controller sent the IP address of the MN's preferred parent to all potential parents. Taking these two approaches limits the flexibility in network management, but with low additional overhead due to the SET/UNSET commands in the current work, parents are managed independently.

Table 2. Comparing main features of basic SDMob and enhanced SDMob.

Feature	Basic SDMob	Enhanced SDMob
Control Window	✓	✓
RPL Aware Leaf	✓	✓
Downward routing	✓	✓
Integration with OF	✓	✓
Multi-instance filter	×	✓
Multiple buffers	×	✓
Buffer Timer	×	✓
Congestion Timer	×	✓
SET/UNSET commands	×	✓

In Figure 3, a timeline demonstration of the SDMob's handoff process is illustrated. The handoff process is carried out in 4 steps, as described below:

- Step 1: MN embeds its velocity in a control beacon and broadcasts it (during the Control Window).
- Step 2: All SNs in the vicinity of MN receive the beacons, append the measured RSSI values and transmit the beacon to the border router. Contention for the wireless channel towards the root node may overwhelm the network. To deal with this problem, we define a random congestion timer that randomly delays forwarding the RSSI values between 5 to 20 milliseconds. The SNs only relay the control beacon after the congestion timer expires.
- Step 3: Another challenge is deciding the amount of time that the border router waits for the RSSI values. We define a second timer called the buffer timer in a deterministic manner that specifies how long the border router waits for control packets. This timer can be tuned based on the distance from the mobile node to the controller and is set to 500 milliseconds by default. Once the buffer timer expires, the border router sends the accumulated buffer over the serial line to the controller for processing, and meanwhile, it assigns each RSSI measurement to its associated buffer.
- Step 4: Controller executes the particle filter, and selects the new best parent. Next, the new and old best parent get notified by a downstream packet containing a SET and UNSET command respectively. The new parent will continue serving the MN's data packets (during the data window) until it receives an UNSET command.

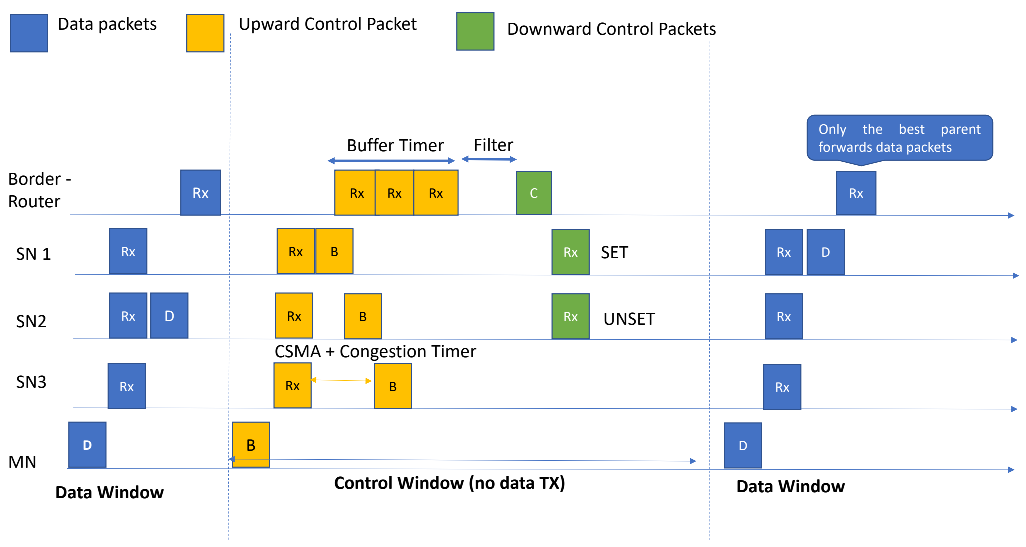


Figure 3. Time diagram of handoff process in SDMob showing how buffer timer and congestion timer and SET/UNSET commands improve the basic SDMob [12].

3.3. Filter Design

Tracking algorithms extend the localization of mobile entities in time via successive runs of localization and predictions. Higher error in localization/tracking of the mobile node leads to an erroneous calculation of the link quality and thus sub-optimal selection of parent nodes. The SDMob architecture offloads performing the filter to the more resourceful edge devices.

Bayesian filters such as Kalman and particle filter, model the state space as a Hidden Markov Model as depicted in Figure 4. A hidden Markov model is a graphical statistical model that relates the unobservable states (actual position) to the previous states (by velocity) and the noisy observations (based on RSSI reports). Within the model, the Markovian property holds, meaning that each state (k-th) at a given time only depends on the previous state ($k - 1$-th).

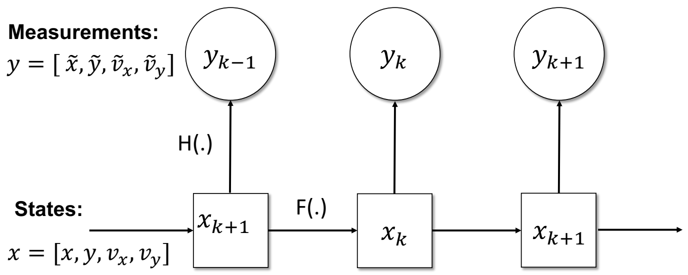


Figure 4. Markovian dependencies for the tracking problem.

The state vector and measurement vector at k-th time step are $\mathbf{x}_k = [x \quad y \quad v_x \quad v_y]^T$ and $\mathbf{y}_k = [\tilde{x} \quad \tilde{y} \quad \tilde{v}_x \quad \tilde{v}_y]^T$ respectively. x and y are positions within Cartesian coordinates. The transitional probabilities in a Markov process can be formulated using the Chapman Kolmogorov equation. Given a set of measurements before the current time slot $z_{1:k-1} = z_1, ..., z_{k-1}$, the prior probability or $p(x_k|z_{1:k-1})$ we have:

$$p(x_k|z1:k-1) = \int R^{n_x} p(x_k|x_{k-1})p(x_{k-1}|z1:k-1)dx_{k-1} \tag{1}$$

And then after receiving the measurement z_k, the posterior probability $p(x_k|z_{1:k})$ can be given by Bayes rule:

$$p(x_k|z1:k) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{z:k-1})} \tag{2}$$

The state vector can be related to the previous state (Markov Property) using a function denoted by F in Equation (3) and the relation of each state with the corresponding measurements can be described using a function denoted by H in Equation (4).

$$\mathbf{x}_k = F(\mathbf{x}_{k-1}) + \mathbf{n}_{k-1} \tag{3}$$

$$\mathbf{y}_k = H(\mathbf{x}_k) + \mathbf{r}_k \tag{4}$$

In this system model, $\mathbf{n}_k$ and $\mathbf{r}_k$ are the prediction and measurement noise respectively. These two noises are mutually independent. These equations are only analytically tractable if noise is Gaussian and functions are linear and Kalman Filter is only advantageous in linear functions and the presence of Gaussian noise. Some enhancements such as Enhanced Kalman Filter (EKF) focus on handling non-linear F and H functions. However, to counteract non-Gaussian noise under a non-linear trajectory in the controller, we are compelled to adopt some other techniques such as UKF or particle filter.

Particle filter is also known as Monte Carlo Sampling and maintains a set of fully random particles as in Equation (5). The filter can take advantage of any a priori knowledge of the obstacles and infeasible positions when initializing the samples.

$$p(x_t) = \sum_{i=1}^{N} w_i \delta(x - x_i) \tag{5}$$

Once it receives the RSSI measurements, first the distance can be estimated based on the radio model. Using the path loss model in Equation (7), the controller estimates the distance between MNs and distinct SNs and then applies triangulation. Here, P_1 denotes the received signal strength in a 1 meter distance, and α is a constant value, describing the radio propagation in the environment [44].

$$RSSI_{(d,t)} = RSSI_{d_0} - 10\eta log_{10}(\frac{d}{d_0}) + X_\sigma \tag{6}$$

$$d = 10^{\frac{RSSI - RSSI_{d_0}}{10\eta}} \tag{7}$$

Then the filter updates the weights based on

$$w_i = \frac{\tilde{w}_i}{\sum_{i=1} N\tilde{w}_i} \tag{8}$$

In the prediction step, the filter moves all the particles based on the IMU information. In this step, the particle filter defines obstacles or feasible areas for the MN by simply removing the samples that come to be outside the legit area. Consequently, an estimate of the mean of the posterior is calculated using simple weighted averaging. Finally, the filter selects one of the anchors to become the new preferred parent and initiates a route-projection packet, containing either a SET or UNSET rule. The SET rule is sent to the new parent to suggest accepting the MN's data packets and the UNSET rule signals the anchor to stop relaying. If an UNSET packet gets lost, it may cause the network to carry multiple instances of the same data packet and congest the network. But one can arrange such a multi-parenting scenario deliberately to have redundant anchors for a mobile node. Studying multi-parent routing is out of the context of this paper. SDMob sends the UNSET packet only if the preferred parent has changed, and the SET packet is transmitted every filtering interval.

With the probabilistic approach that particle filter takes, some of the samples may lose their importance and cause the filter to diverge. The filter should eliminate those irrelevant particles, and increase the number of credible particles. This can be performed in the re-sampling step, and to avoid the overheads it is carried out only when the number of effective samples drops below a certain threshold. There are a few alternative algorithms that the filter can select as re-sampling strategies such as systematic, stratified, and residual re-sampling. In SDMob we use systematic re-sampling in which N points are selected (with even distances) in the whole area and then randomly moved. It is believed that systematic re-sampling reduces the computational complexity while giving identical or improved estimates[45]. The number of effective samples can be defined using Equation (9).

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} w_k^{(i)}} \tag{9}$$

All the mentioned steps are performed for each MN, and each MN has its own set of particles. Each RSSI measurement contains the IP address of the MN, so the filter can easily associate RSSI values to the particle sets. The overall procedure is presented in Algorithm 1.

Algorithm 1 SDMob's algorithm with Particle Filter

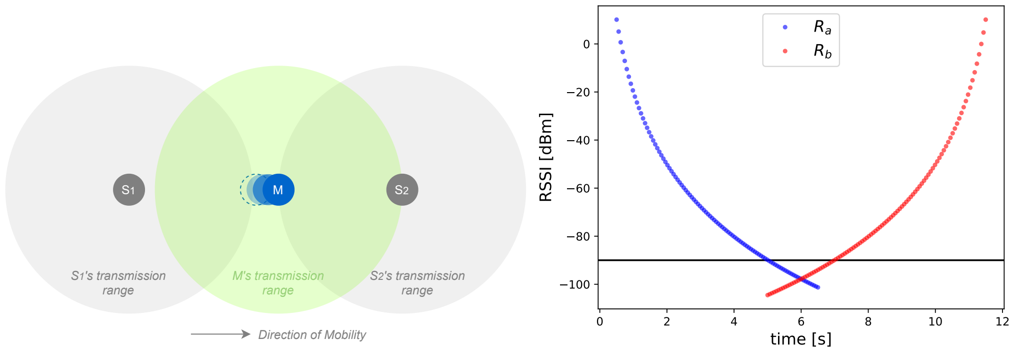
```
1:  Initialize a number of particles with random values for positions and velocities
2:
                          $X_0^i \leftarrow p(x_0)$
3:  while Termination condition not reached do
4:      Wait for the filter timer
5:      for each Mobile Node do
6:          if $N_{eff} \leq N/2$ then
7:              Re-sample
8:          end if
9:          Update weights:
10:         for Each particle $i$ do
11:             $\mathcal{L}(y_t|x_{t-1}^i) \leftarrow N(H(x_k), \theta)$
12:             $\tilde{w}_t^i \leftarrow w_{t-1}^i \mathcal{L}(y_t|x_{t-1}^i)$
13:         end for
14:         Predict samples:
15:         for Each particle $i$ do
16:             $x_k^i \leftarrow F(x_{k-1}^i)$
17:         end for
18:         Estimate by weighted averaging
19:         Send SET/UNSET packets to the Mobile Node
20:     end for
21: end while
```

4. Analytical Model and Evaluation

We have conducted a probabilistic analysis of the SDMob architecture. This model can guide us through the reasons why SDMob may lose or gain advantages in specific scenarios. In a previous work [46], we focused on comparing RPL with SDMob using the analytical evaluation. Now we aim at evaluating the impact of parameters such as path loss variance and the MN's velocity, and compare the results from the analytical model with simulation results.

4.1. Analytical Model

For modeling the radio and network, we try to resemble the simulations that are explained in the next section. The received signal strength at distance d can be estimated using Equation (7) with σ being the standard deviation in RSSI measurements due to shadowing. We consider a scenario in which an MN moves from the proximity of static node A, approaching its future parent, static node B that resides in a 10-meter distance, with a constant speed of V horizontally as depicted in Figure 5a. The transmission range of the nodes is 5 m. The upward control beacons are assumed to be delivered to the controller, and the particle filter tracks the MN (with positioning error = 0). The downward packet containing the new routing rule from the controller (route-projection packet) is assumed to be received at the MN at time $T_r x$ which follows a normal distribution. The expected signal quality ($\overline{R_A(t)}$) from node A deteriorates exponentially in time with increasing distance and contrarily rises for node B (see Figure 5b).



(**a**) (**b**)

Figure 5. The illustration of the scenario considered for the analytical evaluation—(**a**) mobile node M moves from vicinity of its current parent (S1) to the future parent (S2); (**b**) expected RSSI values with respect to the lower threshold of RSSI when M starts moving from vicinity of AP_a to AP_b (right).

Since the distribution of the noise (X_σ) follows the Gaussian distribution, the probability of the RSSI to be below a certain threshold (T_ℓ) can be derived by:

$$P(R_a(i) < T_\ell) = Q(\frac{-T_\ell + \overline{R_a(i)}}{\sigma})$$

where T_ℓ corresponds to the lower threshold of RSSI and Q function is the complementary distribution function of the standard Gaussian distribution.

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \frac{exp(u^2)}{2} du \tag{10}$$

Accordingly, the probability of packet loss at time t can be estimated. If the route-projection packet is not received at time t ($T_{rx} > t$), probability of packet loss equals ($R_a(t) < T_\ell$) meaning that if the link to the old parent gets disconnected. After reception of the route-projection packet, it is the link to parent B that matters and packets can get lost only if $R_b(t) < T_\ell$. We assume that the probability of receiving the route projection packet in time follows a normal distribution. Since the reception of the route-projection packet is independent of the link quality, we can derive the packet loss probability by multiplying these two as formulated in Equation (11).

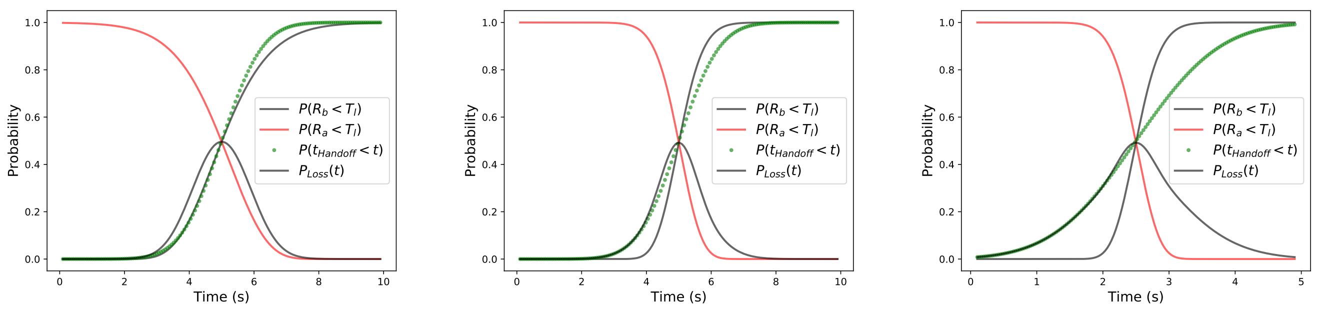
$$\begin{aligned} P_{Loss}(t) = P(R_a(t) < T_\ell) \times P(T_{rx} < t) \\ +P(R_b(t) < T_\ell) \times P(T_{rx} > t)) \end{aligned} \tag{11}$$

where the $P(T_{rx} < t))$ is simply the integral of the probability distribution function of T_{rx}:

$$P(T_{rx} < t) = \int_0^t P(T_{rx=\theta})d\theta) \tag{12}$$

4.2. Analytical Evaluation

With the model being devised, we can determine the probability of packet loss when changing the model parameters. With higher standard deviation in the RSSI measurements, it is expected that link quality fluctuations are observed more often and the probability of individual links being broken increases. As illustrated in Figure 6a,b, $P_{Loss}(t)$ increases with σ since it is the summation of availability of individual links according to Equation (11). In Figure 6c it can be seen that with increased velocity (2 m/s), $P_{Loss}(t)$ increases compared to the case with 1 m/s and the same σ depicted in Figure 6a. We can also calculate the expected probability of packet loss over the entire period ($E(P_{Loss})$) as depicted in Figure 7a to better see the increasing trend as a function of path loss variance.



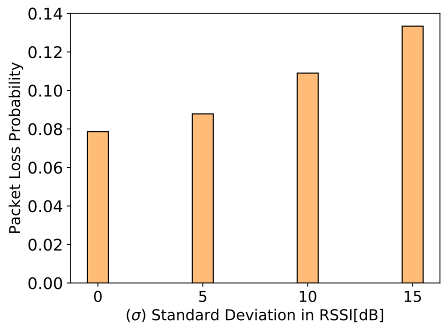
(**a**) $\sigma = 10$ and v = 1 m/s (**b**) $\sigma = 5$ and v = 1 m/s (**c**) $\sigma = 10$ and v = 2 m/s

Figure 6. Probability of packet loss, disconnections of individual links, and handoff in time. Comparing figures (**a**,**b**), we can observe higher packet loss probability stemming from fluctuations in the link quality. Comparing Figures (**a**,**c**), the higher packet loss probability is stemming from the increased velocity.

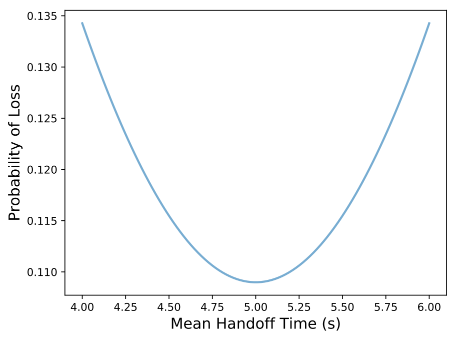
On the other hand, a vital parameter in the model is the expected instant of time when the route-projection packet is received (T_{rx}). Here we assume that once the route-projection packet is received, the nodes omit the old routing state from its neighbor table and insert the new one with negligible delay. We assume that T_{rx} follows a normal distribution and its mean value is interpreted as Mean Handoff Time. P_{Loss} for increasing Mean Handoff Time is illustrated in Figure 7b. This figure shows that scheduling the handoff either too early or too late increases the probability of packet loss and the best time for receiving the route-injection packet is when the mobile node is placed at an equal distance from both SNs.

4.3. Relation between the Analytical Evaluations and Simulations

To verify the mathematical abstractions that were presented in this section, it is both useful and common to validate the analytical results with simulations. Here we give a brief explanation of the simulations but the detailed explanation of the simulation environment can be found in the next section. The experiment includes 25 randomly placed anchors and 1 MN moving in a random trajectory.



(**a**) Loss probability with respect to standard deviation in RSSI

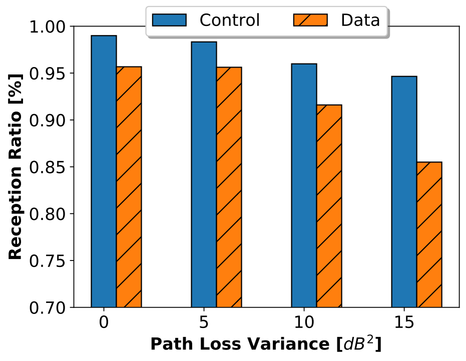


(**b**) Loss probability with respect to expected T_{rx}

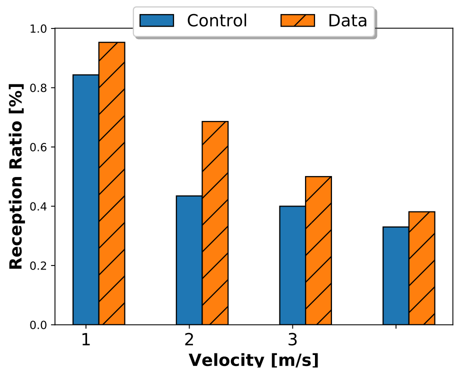
Figure 7. As illustrated in (**a**) with a higher variance of RSSI, it is expected to experience more packet loss in the system as the links are less stable. In (**b**), the packet loss probability with respect to the expected time for receiving the route-injection packet or T_{rx} (or Mean handoff time) is illustrated. The minimum packet loss happens when the packet is expected to be received in the middle of the trajectory (t = 5 for the scenario with 10 m distance).

The simulations showed the packet loss increases by path loss variance as illustrated in Figure 8a. The simulation results are aligned with the probability of packet loss in the analytical evaluation in Figure 7a.

As depicted in Figure 8b, increasing the velocity of the MN also impacts the reception ratio of the data packets in the same way as the analytical evaluations suggested (in Figure 6c).



(**a**)



(**b**)

Figure 8. Simulation results for increasing (**a**) path loss variance and (**b**) velocity.

5. Performance Evaluation

This section focuses on evaluating the performance of the SDMob. We have conducted a set of simulations based on Contiki's native emulator, Cooja [47]. The SDN controller has been implemented using a Linux machine with a Python-based filter, which connects to a C-based Contiki border router. For the IoT RPL/6LoWPAN network, we rely on the Contiki-NG/COOJA simulation environment [48]. Contiki-NG is an open-source embedded operating system, which is easily portable to commodity hardware. We have chosen the well-known Sky platform for the nodes as it is very constrained (in terms of memory), however, the results have also been verified using a more recent Zolertia platform. Sky motes are using an MSP430 F1611 micro-controller featuring 10 kB of RAM and 48 kB of flash memory. Zolertia motes run on an MSP430 F2617 MCU with 8 KB of RAM and 92 KB of flash memory. Both motes communicate using Chipcon's IEEE 802.15.4 compliant CC2420 Radio working in the 2.4 GHz ISM band. The radio model incorporated in Cooja is the Distance Loss mode of the Unit Disk Graph Medium (UDGM). This model considers two circles around each mote that define the distance in which the packets can be received or can interfere with other transmissions.

Table 3 highlights the selected configuration for the simulations. The following metrics have been extracted for SDMob.

- *Position estimation accuracy:* is the difference between the estimated position and the ground truth and is expressed by Root Mean Squared Error (RMSE) which is a measure that represents the quadratic mean of the error and penalizes the larger errors.
- *Handoff delay:* Measuring handoff delay for SDMob is non-trivial as the start time of handoff is not defined and SDMob may perform a handoff in a proactive way when the previous link is still active. To define an upper bound for handoff delay, we assume the time that a link's quality drops below a lower threshold to be the start time of handoff.
- *End-to-end delay:* The time for data packets to reach the destination (sink node) from MN.
- *Packet Delivery Ratio:* The ratio of received packets (in the sink) to the transmitted packets.
- *Communication Overhead:* The accumulated number of bytes for control packets.

Table 3. The parameters used in the simulations.

Parameter	Value
Network Simulator	Cooja under Contiki-ng
Radio model	UDGM
Number of anchor nodes	10, 20, 30, 40, 50
Simulation Area	20×20 m
Transmission range	5 m
Simulation time	300 s
Initialization time	300 s
Beacon Interval	1 s
I_{min}	2^{12}
I_{max}	2^{20}
Buffer Timer	500 ms
Congestion Timer	0–50 ms (uniform distribution)

5.1. An Overview of the ARMOR

ARMOR [26] is one of the most recent mobility solutions integrated within RPL, where it shows better performance compared with MA-RPL [23]. Thus, we picked ARMOR as the benchmark to compare with SDMob. Authors of ARMOR argue that the mobility solutions that are based on the RSSI routing metric fail to select the most stable routes. Hence the key idea behind ARMOR is in introducing a new routing metric, called Time-to-Reside (TTR) that estimates for how long each candidate parent is available. Each node embeds its GPS position and velocity in a DIO packet and broadcasts it. The neighbors exploit this information to calculate the TTR. Using this metric, ARMOR also proposes a parent selection mechanism. ARMOR modifies the Trickle algorithm to have short and constant intervals between DIO packets. ARMOR also allows mobile nodes to advertise the DODAG and act as parents, which is a reasonable design choice for a network with many mobile nodes. The fact that mobile nodes can be set as parents is inevitable in a topology with many mobile nodes and it may require more frequent DIO intervals for all the nodes. In the same way, ARMOR allows MNs to serve as parents with the cost of transmitting frequent update packets. The interval for sending the control beacons in SDMob is tunable according to the physical speed of the mobile node.

ARMOR, like some of its counterparts, assumes that nodes are equipped with GPS modules, by which the location of the nodes is fed to the routing protocol. First, GPS systems are not practical in many applications because they perform poorly in an indoor environment. It is also possible to integrate other localization methods such as fingerprinting or triangulation, but the impact of its overhead and accuracy on the routing protocol needs to be investigated.

Figure 9a highlights a motivational scenario for ARMOR where it can do better than baseline RPL. Using the TTR metric, node *M* selects node *S2* as its parent since it provides a long-lasting connection. On the other hand, in Figure 9b, ARMOR neglects the more stable links provided by *S1* and *S3* and sticks to the *S2* with long-lasting but lower quality link.

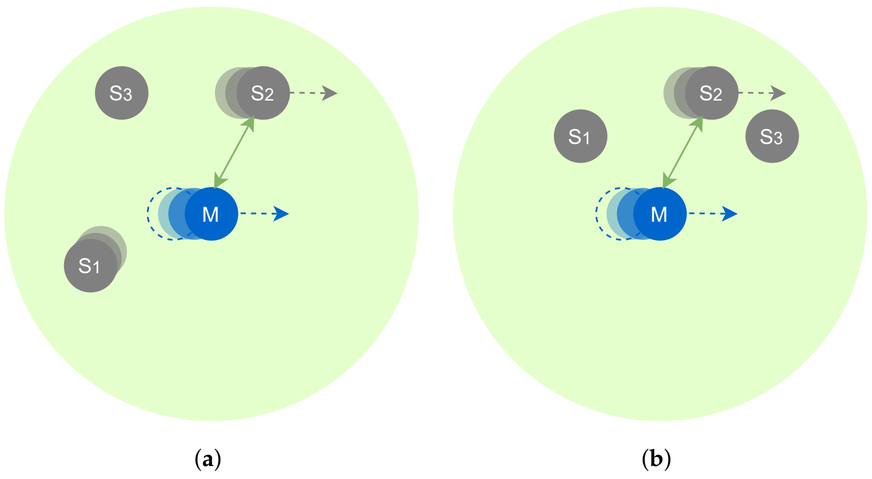


Figure 9. Two examples of ARMOR algorithm, (**a**) when mobile node *M* wisely selects *S2* as its parent since it provides the most long-lasting connection. RPL would have chosen *S1* by only considering the short-term link quality, and (**b**) depicts a scenario, where ARMOR is less effective as *M* ignores the stable (fixed) parents (*S1, S3*) and sticks to *S2*.

5.2. Scaling to Multiple Mobile Nodes

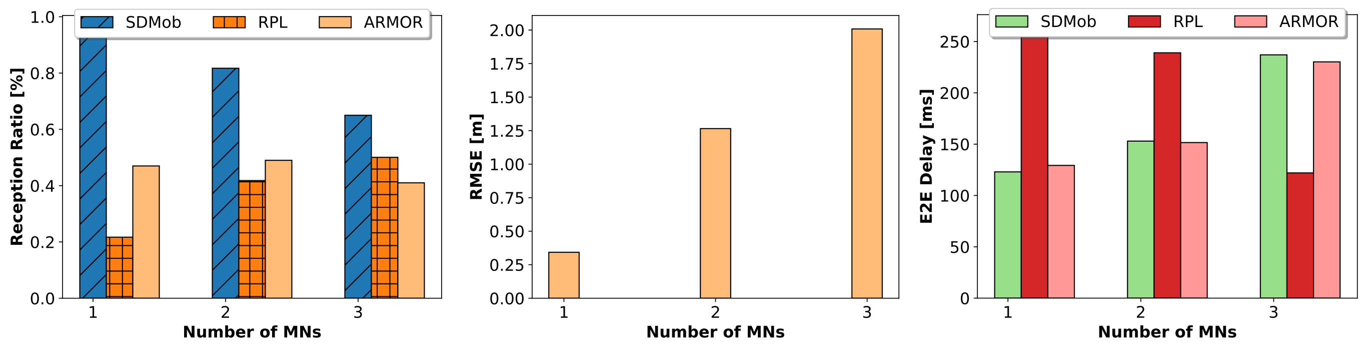
SDMob lies in a spectrum in which one extreme is RPL with its scalability to thousands of nodes and the other extreme are the protocols specifically designed for mobile ad-hoc networks such as AODV. SDMob is designed for IoT applications that take advantage of the static multi-hop infrastructure and hence it can support a limited number of MNs.

Figure 10 shows the comparison of the evaluation metrics for SDMob and RPL in a scenario with 25 randomly placed anchor nodes and a number of MNs moving with Truncated Levy Walk mobility pattern (for detail on mobility patterns see Section 5.4). From the results, we observe that although RPL shows an increasing pattern in packet delivery ratio (PDR), the proposed SDMob remarkably outperforms RPL. RPL performs better in terms of PDR for an increasing number of MNs since the Trickle algorithm resets its interval, which leads to more frequent DIO packet transmissions that congest the network. This also justifies the rise in average E2E delay for RPL. The congestion caused by the increasing MNs affects the accuracy of localization that happens due to the loss of a significant number of RSSI reports. However, the particle filter showed a good ability to converge again after not receiving RSSI reports, and the impact on localization accuracy was tolerable. The difference in the upper bound of handoff delay was negligible (100 milliseconds). Figure 11 shows SDMob's communication overhead in comparison with standard RPL for increasing number of mobile nodes.

5.3. Scaling to Networks with High Density or Hop-Distance

The localization algorithm works based on the RSSI measurements, thereby being spotted in an area lacking enough anchors to retrieve this information is fatal for the filter. On the other hand, there may be scenarios with too many anchors or the anchors are so far that the RSSI reports do not reach the controller in time.

SDMob's anchor nodes inherit RPL's design choices when it comes to handling high data rate traffic. Once an MN enters an area with a large number of anchors, it will congest the network and hence the local links will measure a higher ETX. Most standard-compliant implementations of RPL reset the DIO interval when there is a significant change in the rank value assuming that it has been caused by a lossy link. This leads to more frequent transmissions of DIO packets and further congesting the network. To avoid the aforementioned scenario, we have employed the congestion timer. The simulated set of scenarios consists of one MN moving in a linear trajectory and the anchors are manually positioned to keep the number of neighbors constant over time as depicted in Figure 12. Figure 13 shows the performance of SDMob in networks with an increasing number of neighbors. For up to 5 neighbors, we can see an increasing trend in the PDR but for 6 neighbors the network gets so congested that a sharp drop in the PDR is noticed both for control and data traffic. Localization and E2E delay also follow the same pattern.



(a) (b) (c)

Figure 10. Simulation results for increasing number of MNs with different solutions—(**a**) PDR; (**b**) localization error; and (**c**) end-to-end delay.

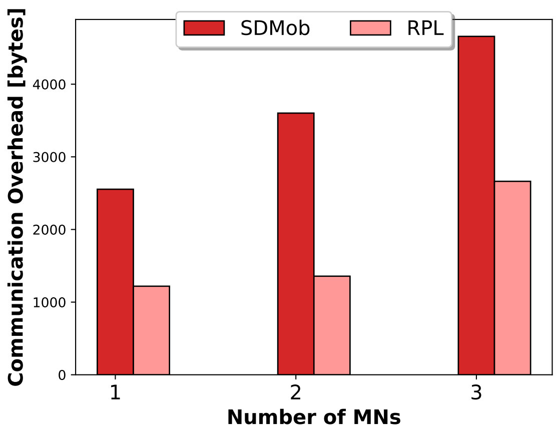
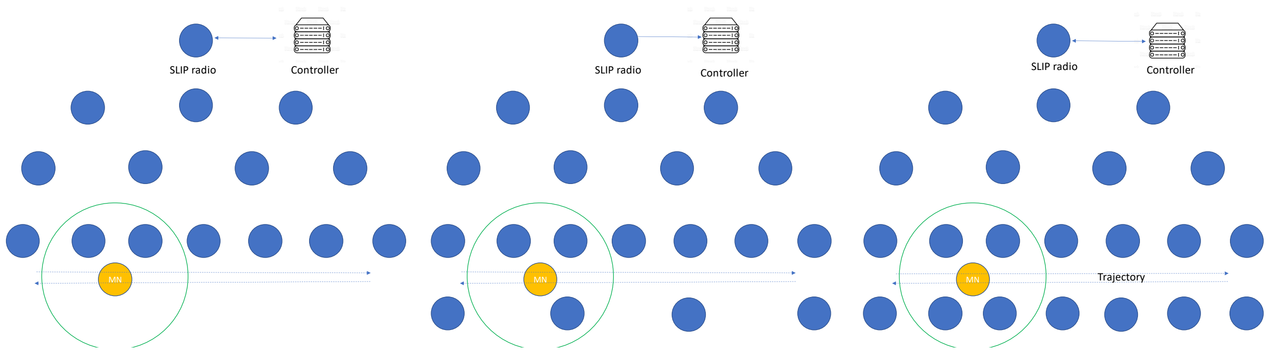
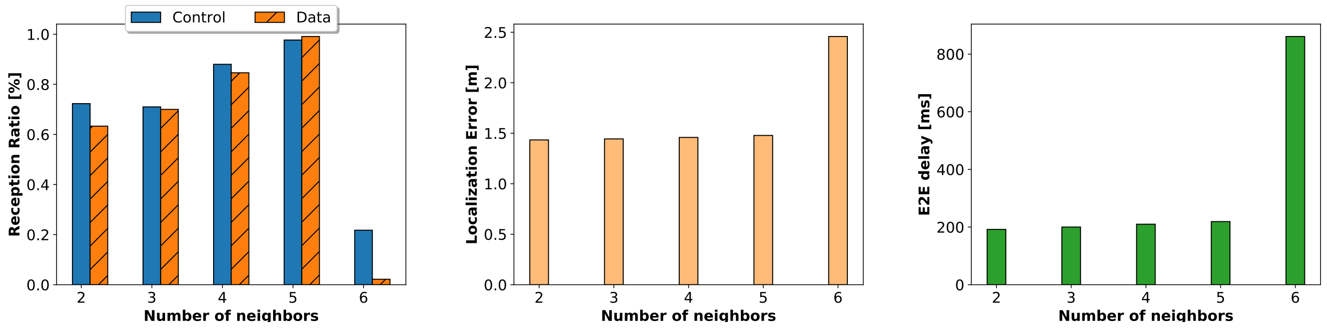


Figure 11. Communication overhead (in bytes) for increasing number of MNs with different solutions.



(a) (b) (c)

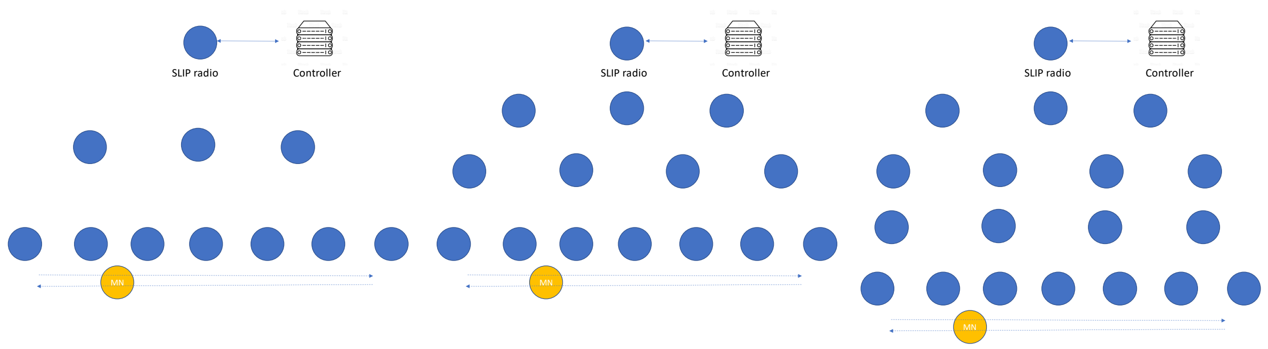
Figure 12. Simulation scenarios when increasing number of neighbors of a MN: (**a**) with 2 neighbors, (**b**) with 3 neighbors, and (**c**) with 4 neighbors.



(**a**) (**b**) (**c**)

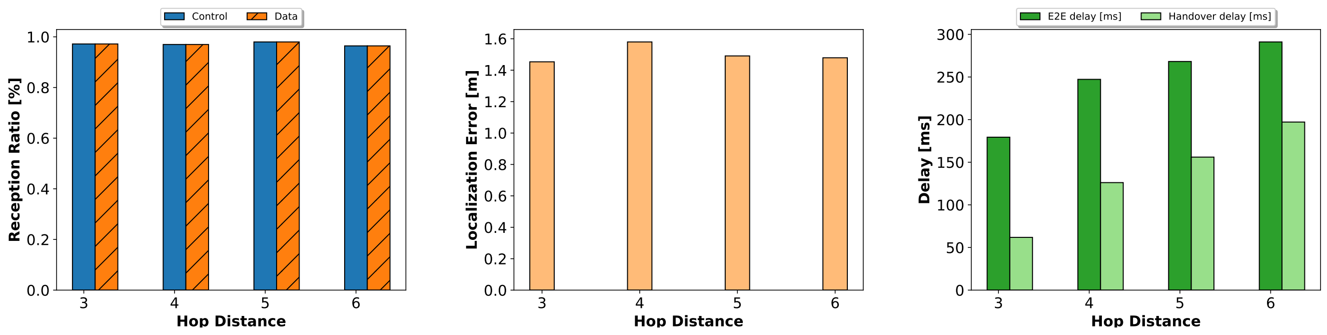
Figure 13. Simulation results for increasing number of neighbors with SDMob—(**a**) PDR; (**b**) localization error; and (**c**) end-to-end delay.

Another parameter that changes in scale is the hop distance from the MN to the controller. To examine the performance of SDMob in this regard, we manually placed the anchors to keep the hop distance of MN constant over time as illustrated in Figure 14 and experimented with different distances. As illustrated in Figure 15, the PDR and localization are more stable with increasing hop distance. However, there is a continual growth in E2E delay and handoff delay.



(**a**) (**b**) (**c**)

Figure 14. Simulation scenarios when increasing distance between MN and controller, (**a**) with 3-hop distance, (**b**) with 4-hop distance, and (**c**) with 5-hop distance.



(**a**) (**b**) (**c**)

Figure 15. Simulation results for increasing distance between MN and controller with SDMob—(**a**) PDR; (**b**) localization error; and (**c**) end-to-end delay.

5.4. Mobility Patterns

The previous tests required a deterministic configuration that could only be performed using a linear moving trajectory and a chosen topology. Now we shift the focus on a random placement of nodes and trajectory. We utilized an open-source library called pymobility [49] to create some of the most well-known mobility traces in the field. The number of neighbors and hop-distance depends on the mobility pattern and changes in time. There are 25 anchors and one MN in each of the scenarios below.

5.4.1. Random Way Point (RWP).

RWP is a synthetic mobility model in which MNs choose a random point in the simulation area and start moving toward it with a randomly chosen velocity. In random waypoint, nodes tend to be spotted in the middle of the simulation area. Since the newly selected point is random, sharp turns may happen that are not realistic. Another phenomenon called density wave [50] is that the number of neighbors for each MN fluctuates considerably. As the MN passes through the center of the simulation area, where it is usually more crowded (by both static and MNs), it may further converge the network. On the other hand, when the MN is spotted closer to the edges, it may be subject to packet losses in blind spots.

5.4.2. Random Direction Model (RDM)

In RDM, MNs randomly choose a direction until it reaches the boundary of the area, and after a pause, chooses a new direction as depicted in Figure 16a. If the number of MNs is high, this will decrease the probability of congestion in the center as the MNs tend to be spotted at the edges of the area, and thus it is claimed that RDM is known to be unaffected density wave problem. This paper aims at extending RPL to support a limited number of MNs. As a result, the fact that nodes are more often spotted at the edges does not considerably decrease the congestion, and RDM is even expected to reduce the number of RSSI measurements that arrive at the controller and are fed to the filter. The filter works reliably when it receives at least 3 measurements. The effect of the number of neighbors has been studied in the previous subsection.

5.4.3. Gauss Markov Model (GMM)

GMM is a memory-based mobility pattern in which the temporal dependency of the nodes can be defined using a parameter α (between 0 and 1). Higher values of α imply higher dependency and less harsh turns and speed changes. This leads to more realistic mobility patterns and configurable randomness.

5.4.4. Truncated Levy Walk (TLW)

It has been claimed that human mobility has the same characteristics as Levy Walks that follows the heavy-tailed Levy distribution [51]. This leads to a number of short flights followed by a long flight and fewer harsh turns as depicted in Figure 16b. Therefore it is expected for the filter to perform better under this mobility pattern.

Figure 17 compares the evaluation metrics for different mobility patterns. The distribution of velocity is different but we kept the same mean velocity for fairness. The PDR reached a peak during the simulation with TLW since the harsh turns are minimized in this mobility pattern and there are a lot of short flights that reduce the number of required handoffs. In RDM, the data traffic is not as well-received as the control traffic. This is caused by the fact that RDM challenges the localization algorithm with unpredictable churns. For RWP and RDM, the localization error is higher than TLW and GMM. This is due to the higher randomness in direction and velocity in these patterns. E2E delay for RDM is shown to be higher because of the MN residing in the edges of the simulation area.

5.5. Velocity

Another decisive parameter is the velocity of the MN. Target applications mandate supporting the physical speed of humans (for healthcare applications) averaging about 5 km/h or 1.34 m/s. For industrial applications, many machines such as forklifts are capable of speeds over 22 km/h but regulatory agencies such as Occupational Safety and Health Administration recommend a speed of 5 km/h for typical indoor environments [52]. We have evaluated the performance of the system with speeds up to 2.4 m/s.

We have tested a group of scenarios in which a single MN roams around according to the TLW model with its speed averaging to the values between 0.6 m/s to 2.4 m/s. What stands out in Figure 18a is the steady rise in localization error with increasing velocity. In the previous section in Figure 8b, we observed a general decreasing pattern of PDR with increasing velocity of the MN.

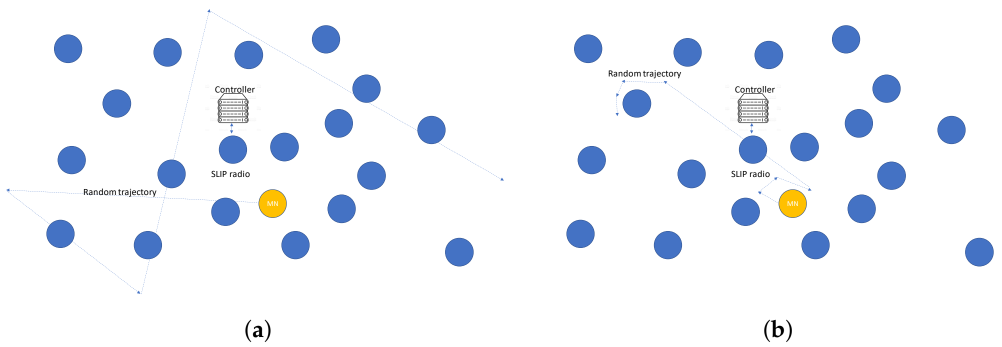


Figure 16. Simulation scenarios with different mobility patterns: (**a**) with RDM mobility pattern and (**b**) with TLW mobility pattern.

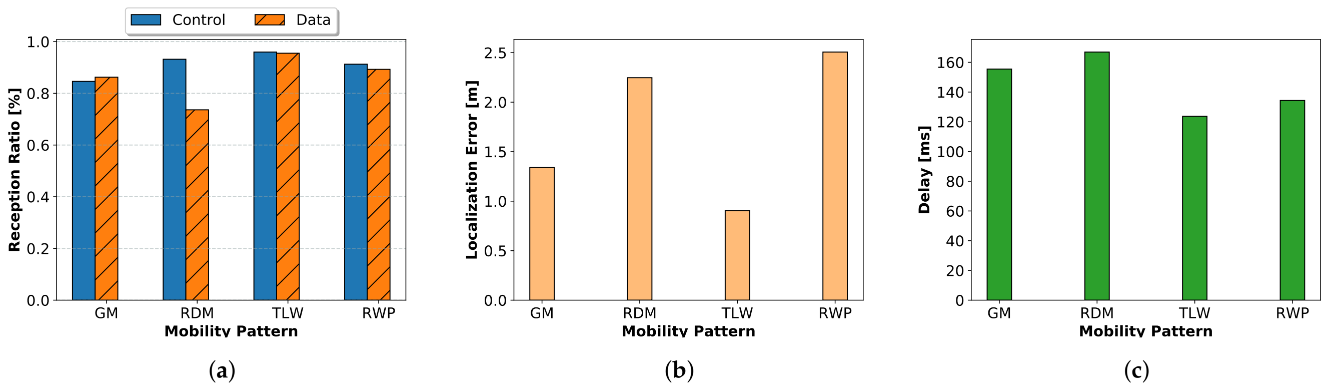


Figure 17. Simulation results for different mobility patterns with SDMob—(**a**) PDR; (**b**) localization error; and (**c**) end-to-end delay.

The loss in packet delivery ratio with an increased physical speed of MN stems from both (i) lower accuracy of the filter and (ii) higher probability for delayed reception of the SET packet. It is possible to increase the supported maximum speed of the MN. At a higher velocity, it is more challenging to keep the accuracy of the filter unless by transmitting more frequent control beacons. A more frequent control beacon is not always desirable since it imposes a higher communication overhead. This higher overhead may avoid the boosted localization accuracy from improving the PDR. We consider dynamically changing this beaconing interval for future work.

5.6. Path Loss Variance

For changing this parameter, we had to enhance the radio model in Cooja to support a zero-mean normal distribution. Figure 18b reveals that if the link qualities are fluctuating more (due to the environment characteristics) it gets more difficult to track the MN so SDMob shows a slight escalation in localization error.

In Section 4, the degrading impact of path loss variance on reception ratio in the simulations was compared with the analytical evaluations in Figure 8a. It is worth mentioning that although the reception ratio for control traffic is not affected much, the data traffic's PDR is about 10 percent less than control traffic. This can be explained by the fact that the RSSI fluctuations contribute to packet loss.

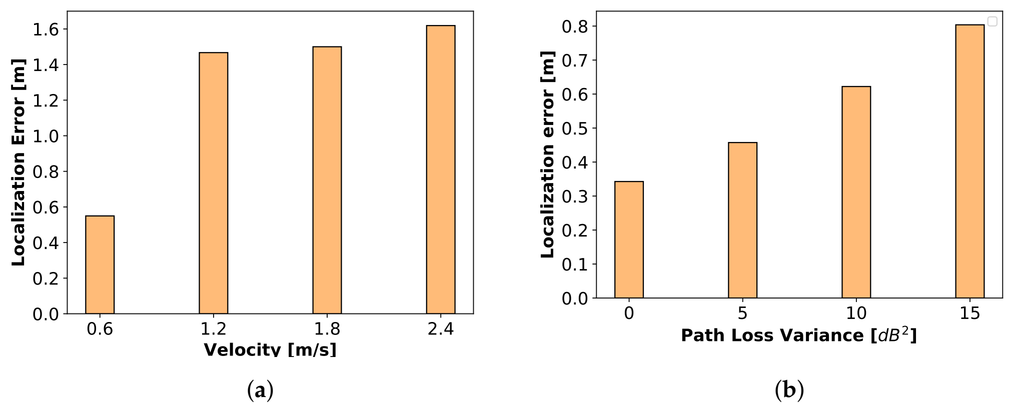


Figure 18. Average localization error as a function of (**a**) MN's velocity, and (**b**) path loss variance.

6. Conclusions

To address the low reliability of the RPL protocol in mobile IoT applications, we have proposed SDMob. In this proposed architecture, an edge device collaborates with the distributed nodes to provide seamless handoff for the mobile nodes. However, new challenges arise, such as delivering real-time link quality reports to the controller as well as the downward packets to install root-initiated routes. SDMob addresses these challenges by employing a lightweight controller that tunes its operation for a constrained environment. Having the controller deployed, the computation-intensive tasks can be offloaded to the controller to benefit from the global view and resources available at the centralized controller.

The results show that SDMob significantly improves RPL and outperforms state-of-the-art ARMOR with close to 100 percent PDR, with reasonable and adjustable overhead in scenarios with a varying number of mobile nodes. Given the requirements of the applications, SDMob by design aims at reliably maintaining the connectivity of a limited number of mobile nodes since with increasing mobile nodes the control traffic increases as well. The solutions in the literature that aim at a higher number of mobile nodes usually make a compromise by less reliable communication for mobile nodes. Through analytical evaluations, we analyzed the behavior of the system when exposed to increasing path loss variance in the radio environment, and velocity of the mobile node which was also verified by simulations. We extended the simulation results to networks with varying densities of neighbors, the distance of the mobile node from the controller (in terms of the number of hops), and mobility patterns.

For future work, we consider employing redundant controllers to avoid a single point of failure in the control layer, although, in the event of a failure in the controller, RPL resumes its normal operation. Another interesting direction is applying machine learning methods to either localize the mobile node and predict its link quality in time, or automatically optimize different system parameters such as beacon interval and congestion and buffer timers.

Author Contributions: Conceptualization, I.R. and H.F.; methodology, I.R., M.A., and H.F.; software, I.R. and S.P.S.; analytical evaluation, I.R.; writing—original draft preparation, I.R.; writing—review and editing, H.F., M.A., S.P.S., M.V., A.B., and M.B.; visualization, I.R. and S.P.S.; project administration, H.F.; funding acquisition, H.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Swedish Foundation for Strategic Research via the FiC project, by the Swedish Research Council (Vetenskapsrådet) through the MobiFog starting grant, Vinnova by the GREENER project, the ECSEL Joint Undertaking (JU) projects InSecTT, and DAIS under grant agreement No.876038 and No.101007273.

Institutional Review Board Statement: Ethical review and approval were not applicable for this study.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Aliasghar Mohammadsalehi for sharing his implementation of ARMOR.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Safaei, B.; Mohammadsalehi, A.; Khoosani, K.T.; Zarbaf, S.; Monazzah, A.M.H.; Samie, F.; Bauer, L.; Henkel, J.; Ejlali, A. Impacts of Mobility Models on RPL-Based Mobile IoT Infrastructures: An Evaluative Comparison and Survey. *IEEE Access* **2020**, *8*, 167779–167829. [CrossRef]
2. Levis, P.; Clausen, T.; Hui, J.; Gnawali, O.; Ko, J. *The Trickle Algorithm*; RFC6206; Internet Engineering Task Force: Fremont, CA, USA, 2011.
3. Paul, A.K.; Sato, T. Localization in wireless sensor networks: A survey on algorithms, measurement techniques, applications and challenges. *J. Sens. Actuator Netw.* **2017**, *6*, 24. [CrossRef]
4. Särkkä, S. *Bayesian Filtering and Smoothing*; Cambridge University Press: Cambridge, UK, 2013; Volume 3.
5. Li, L.; Yang, W.; Wang, G. Intelligent fusion of information derived from received signal strength and inertial measurements for indoor wireless localization. *AEU - Int. J. Electron. Commun.* **2015**, *70*, 1105–1113. [CrossRef]
6. Cappe, O.; Godsill, S.J.; Moulines, E. An overview of existing methods and recent advances in sequential Monte Carlo. *Proc. IEEE* **2007**, *95*, 899–924. [CrossRef]
7. Barcelo, M.; Correa, A.; Vicario, J.L.; Morell, A.; Vilajosana, X. Addressing Mobility in RPL with Position Assisted Metrics. *IEEE Sens. J.* **2016**, *16*, 2151–2161. [CrossRef]
8. Gonçalves, A.J.; Rabêlo, R.A.; Rodrigues, J.J.; Oliveira, L.M. A mobility solution for low power and lossy networks using the LOADng protocol. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3878. [CrossRef]
9. Bouaziz, M.; Rachedi, A.; Belghith, A. EKF-MRPL: Advanced mobility support routing protocol for internet of mobile things: Movement prediction approach. *Future Gener. Comput. Syst.* **2019**, *93*, 822–832. [CrossRef]
10. Ruiz, A.R.J.; Granja, F.S. Comparing ubisense, bespoon, and decawave uwb location systems: Indoor performance analysis. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 2106–2117. [CrossRef]
11. Kobo, H.I.; Abu-Mahfouz, A.M.; Hancke, G.P. A survey on software-defined wireless sensor networks: Challenges and design requirements. *IEEE Access* **2017**, *5*, 1872–1899. [CrossRef]
12. Rabet, I.; Selvaraju, S.P.; Adeli, M.H.; Fotouhi, H.; Balador, A.; Vahabi, M.; Alves, M.; Björkman, M. Pushing IoT Mobility Management to the Edge: Granting RPL Accurate Localization and Routing. In Proceedings of the 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 14 June–31 July 2021; pp. 338–343.
13. Fotouhi, H.; Moreira, D.; Alves, M. mRPL: Boosting mobility in the Internet of Things. *Ad Hoc Netw.* **2015**, *26*, 17–35. [CrossRef]
14. Gnawali, O.; Levis, P. *The Minimum Rank with Hysteresis Objective Function*; RFC 6719; Internet Engineering Task Force: Fremont, CA, USA, 2012; p. 13.
15. Thubert, P. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks. RPL [Online]. Available online: https://tools.ietf.org/html/rfc6552 (accessed on 18 January 2022).
16. Oliveira, A.; Vazao, T. Low-power and lossy networks under mobility: A survey. *Comput. Netw.* **2016**, *107*, 339–352. [CrossRef]
17. Kamgueu, P.O.; Nataf, E.; Ndie, T.D.; Kamgueu, P.O.; Nataf, E.; Djotio, T.; Survey, N. Survey on RPL enhancements: A focus on topology , security and mobility. *Comput. Commun.* **2018**, *120*, 10–21. [CrossRef]
18. Tahir, Y.; Yang, S.; McCann, J. BRPL: Backpressure RPL for high-throughput and mobile IoTs. *IEEE Trans. Mob. Comput.* **2017**, *17*, 29–43. [CrossRef]
19. Neely, M.J. Stochastic network optimization with application to communication and queueing systems. *Synth. Lect. Commun. Netw.* **2010**, *3*, 1–211. [CrossRef]
20. Fotouhi, H.; Moreira, D.; Alves, M.; Yomsi, P. mRPL+: A mobility management framework in RPL/6LoWPAN. *Comput. Commun.* **2017**, *104*, 34–54. [CrossRef]
21. Tian, B.; Hou, K.M.; Shi, H.; Liu, X.; Diao, X.; Li, J.; Chen, Y.; Chanet, J.P. Application of modified RPL under VANET-WSN communication architecture. In Proceedings of the 2013 International Conference on Computational and Information Sciences, Shiyang, China, 21–23 June 2013; pp. 1467–1470.
22. Park, J.; Kim, K.H.; Kim, K. An algorithm for timely transmission of solicitation messages in RPL for energy-efficient node mobility. *Sensors* **2017**, *17*, 899. [CrossRef]
23. Murali, S.; Jamalipour, A. Mobility-aware energy-efficient parent selection algorithm for low power and lossy networks. *IEEE Internet Things J.* **2018**, *6*, 2593–2601. [CrossRef]
24. Ancillotti, E.; Vallati, C.; Bruno, R.; Mingozzi, E. A reinforcement learning-based link quality estimation strategy for RPL and its impact on topology management. *Comput. Commun.* **2017**, *112*, 1–13. [CrossRef]
25. Kharrufa, H.; Al-Kashoash, H.; Kemp, A.H. A game theoretic optimization of RPL for mobile Internet of Things applications. *IEEE Sens. J.* **2018**, *18*, 2520–2530. [CrossRef]

26. Mohammadsalehi, A.; Safaei, B.; Monazzah, A.M.H.; Bauer, L.; Henkel, J.; Ejlali, A. ARMOR: A Reliable and Mobility-aware RPL for Mobile Internet of Things Infrastructures. *IEEE Internet Things J.* **2021**, *9*, 1503–1516. [CrossRef]
27. Farag, H.; Österberg, P.; Gidlund, M.; Han, S. RMA-RP: A Reliable Mobility-Aware Routing Protocol for Industrial IoT Networks. In Proceedings of the 2019 IEEE Global Conference on Internet of Things (GCIoT), Dubai, United Arab Emirates, 4–7 December 2019.
28. Parasuraman, R.; Oegren, P.; Min, B.C. Kalman Filter Based Spatial Prediction of Wireless Connectivity for Autonomous Robots and Connected Vehicles. In Proceedings of the IEEE Vehicular Technology Conference, Chicago, IL, USA, 27–30 August 2018.
29. Clausen, T.; Yi, J.; De Verdiere, A.C. LOADng: Towards AODV version 2. In Proceedings of the 2012 IEEE Vehicular Technology Conference (VTC Fall), Quebec City, QC, Canada, 3–6 September 2012; pp. 1–5.
30. Mihaylova, L.; Angelova, D.; Honary, S.; Bull, D.R.; Canagarajah, C.N.; Ristic, B. Mobility tracking in cellular networks using particle filtering. *IEEE Trans. Wirel. Commun.* **2007**, *6*, 3589–3599. [CrossRef]
31. Jurado Lasso, F.F.; Marchegiani, L.; Jurado, J.F.; Mahfouz, A.A.; Fafoutis, X. A Survey on Software-Defined Wireless Sensor Networks: Current status, machine learning approaches and major challenges. *TechRxiv* **2021**. [CrossRef]
32. Latif, Z.; Sharif, K.; Li, F.; Karim, M.M.; Biswas, S.; Wang, Y. A comprehensive survey of interface protocols for software defined networks. *J. Netw. Comput. Appl.* **2020**, *156*, 102563. [CrossRef]
33. Thubert, P.; Jadhav, R.; M., G. *Root Initiated Routing State in RPL*; draft-ietf-roll-dao-projection-09; Internet Engineering Task Force: Fremont, CA, USA, 2019.
34. Rabet, I.; Fotouhi, H.; Vahabi, M.; Björkman, M.; Alves, M. RPL-RP: RPL with Route Projection for Transversal Routing. In Proceedings of the IEEE World Forum IoT 2021, New Orleans, LA, USA, 14 June–31 July 2021.
35. Violettas, G.; Petridou, S.; Mamatas, L. Routing under Heterogeneity and Mobility for the Internet of Things: A Centralized Control Approach. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018. [CrossRef]
36. Theodorou, T.; Mamatas, L. SD-MIoT: A software-defined networking solution for mobile Internet of Things. *IEEE Internet Things J.* **2020**, *8*, 4604–4617. [CrossRef]
37. Baddeley, M.; Nejabati, R.; Oikonomou, G.; Sooriyabandara, M.; Simeonidou, D. Evolving SDN for low-power IoT networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 71–79.
38. Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 513–521.
39. Bello, L.L.; Lombardo, A.; Milardo, S.; Patti, G.; Reno, M. Experimental assessments and analysis of an SDN framework to integrate mobility management in industrial wireless sensor networks. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5586–5595. [CrossRef]
40. Orozco-Santos, F.; Sempere-Payá, V.; Silvestre-Blanes, J.; Albero-Albero, T. Multicast Scheduling in SDN WISE to Support Mobile Nodes in Industrial Wireless Sensor Networks. *IEEE Access* **2021**, *9*, 141651–141666. [CrossRef]
41. Mertens, J.; Milotta, G.; Nagaradjane, P.; Morabito, G. SDN-(UAV) ISE: Applying Software Defined Networking to Wireless Sensor Networks with Data Mules. In Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 31 August–3 September 2020; pp. 323–328.
42. Fotouhi, H.; Vahabi, M.; Rabet, I.; Björkman, M.; Alves, M. MobiFog: Mobility Management Framework for Fog-assisted IoT Networks. In Proceedings of the IEEE Global Conference on Internet of Things GCIoT'19, Dubai, United Arab Emirates, 4–7 December 2019.
43. Thubert, P.; Richardson, M. *Routing for RPL leaves*; Work in Progress, draft-thubert-roll-unaware-leaves-05; Internet Engineering Task Force: Fremont, CA, USA, 2018.
44. Zuniga, M.; Krishnamachari, B. Analyzing the transitional region in low power wireless links. In Proceedings of the 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004, Santa Clara, CA, USA, 4–7 October 2004; pp. 517–526.
45. Hol, J.D.; Schon, T.B.; Gustafsson, F. On resampling algorithms for particle filters. In Proceedings of the 2006 IEEE Nonlinear Statistical Signal Processing Workshop, Cambridge, UK, 13–15 September 2006; pp. 79–82.
46. Rabet, I.; Selvaraju, S.P.; Fotouhi, H.; Vahabi, M.; Bjorkman, M. Poster: Particle Filter for Handoff Prediction in SDN-based IoT Networks. In Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks, EWSN 2020, Lyon, France, 17–18 February 2020.
47. Osterlind, F.; Dunkels, A.; Eriksson, J.; Finne, N.; Voigt, T. Cross-level sensor network simulation with cooja. In Proceedings of the 2006 31st IEEE Conference on Local Computer Networks, Tampa, FL, USA, 14–16 November 2006; pp. 641–648.
48. Österlind, F. *A Sensor Network Simulator for the Contiki OS*; SICS Research Report; Swedish Institute of Computer Science: Stockholm, Sweden, 2006.
49. Panisson, A. *Pymobility v0.1—Python Implementation of Mobility Models*; Istituto per l'Interscambio Scientifico I.S.I.: Torino, Italy, 2014. [CrossRef]
50. Royer, E.M.; Melliar-Smith, P.M.; Moser, L.E. An analysis of the optimum node density for ad hoc mobile networks. In Proceedings of the ICC 2001, IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240), Helsinki, Finland, 11–14 June 2001; Volume 3, pp. 857–861.

51. Rhee, I.; Shin, M.; Hong, S.; Lee, K.; Kim, S.J.; Chong, S. On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.* **2011**, *19*, 630–643. [CrossRef]
52. Occupational Safety & Health Administration; US Department of Labor. *Training Requirements in OSHA Standards and Training Guidelines*; Number 2254 in 2254; Occupational Safety & Health Administration: Washington, DC, USA, 1998.