Journal of
Sensor and Actuator Networks

MDPI

Article

MINDS: Mobile Agent Itinerary Planning Using Named Data Networking in Wireless Sensor Networks

Saeid Pourroostaei Ardakani

School of Computer Science, University of Nottingham Ningbo China, Ningbo 315100, China; saeid.ardakani@nottingham.edu.cn; Tel.: +86-(0)-574-8818-0000 (ext. 8102)

Abstract: Mobile agents have the potential to offer benefits, as they are able to either independently or cooperatively move throughout networks and collect/aggregate sensory data samples. They are programmed to autonomously move and visit sensory data stations through optimal paths, which are established according to the application requirements. However, mobile agent routing protocols still suffer heavy computation/communication overheads, lack of route planning accuracy and long-delay mobile agent migrations. For this, mobile agent route planning protocols aim to find the best-fitted paths for completing missions (e.g., data collection) with minimised delay, maximised performance and minimised transmitted traffic. This article proposes a mobile agent route planning protocol for sensory data collection called MINDS. The key goal of this MINDS is to reduce network traffic, maximise data robustness and minimise delay at the same time. This protocol utilises the Hamming distance technique to partition a sensor network into a number of data-centric clusters. In turn, a named data networking approach is used to form the cluster-heads as a data-centric, tree-based communication infrastructure. The mobile agents utilise a modified version of the Depth-First Search algorithm to move through the tree infrastructure according to a hop-count-aware fashion. As the simulation results show, MINDS reduces path length, reduces network traffic and increases data robustness as compared with two conventional benchmarks (ZMA and TBID) in dense and large wireless sensor networks.

Keywords: wireless sensor networks; named data networking; mobile agents; itinerary planning

Citation: Pourroostaei Ardakani, S. MINDS: Mobile Agent Itinerary Planning Using Named Data Networking in Wireless Sensor Networks. *J. Sens. Actuator Netw.* **2021**, *10*, 28. https://doi.org/10.3390/jsan10020028

Received: 17 March 2021
Accepted: 13 April 2021
Published: 22 April 2021

1. Introduction

Wireless Sensor Networks (WSN) aim to capture sensory recordings from event/source regions and deliver the results to the consumer access point (*Sink*) for further processing and/or decision making. They are deployed either as infrastructure-based or ad-hoc networks [1]. In the former, communications infrastructure is deployed to forward sensory data samples from source sensors to a sink. However, this is expensive, as it requires a great amount of resources to set up, especially in wide and out/hard-to-reach regions (e.g., rainforests, crowded cities or oceans) [2]. The latter focuses on ad-hoc network deployment, which is established with no need for a pre-existing infrastructure. The sensor nodes deploy interconnection links using Zigbee and/or Bluetooth ties for communication with no centralised control and according to a self-organising fashion. This minimises the deployment cost as compared to infrastructure-based networks. Sensor nodes are either static or mobile. Static sensory devices are localised, having no mobility to capture events that happen locally within their sensing range, whereas mobile sensors move across the field to collect sensory recordings. However, sensory nodes are highly resource constrained—mainly power and bandwidth.

Sensory data samples are transmuted to the sink according to two schemes: client/server and mobile agent (MA) [3]. Client/server deploys direct or multi-hop wireless links from sensory nodes to the sink to deliver sensory recordings. However, this increases the network communications and consequently increases the resource consumption (e.g., power).

MA sensory data collection utilises mobile devices to move across the network and capture sensory recordings. This offers benefits such as enhanced data robustness, load balancing, scalability, resource conservation and reduce delay as compared to client/server [4]. An MA has the ability to autonomously move and collect sensory data samples. This can be in two forms: software and hardware. Software MAs are supported by mobile code and/or remote object access techniques to move throughout a network [5], whereas hardware MAs (e.g., unmanned aerial vehicles) move over a network to visit the sensory data station and collect data samples [6]. However, path planning and network resource consumption (e.g., bandwidth) are still the key drawbacks of MA data collection. To resolve this, MAs should intelligently move throughout the network to capture data samples and avoid blind/random walking to conserve network resources.

Named data networking (NDN) is able to offer MAs a number of path planning benefits in sensory data collection applications [7,8]. Using NDN routing, MAs are forwarded to visit source nodes residing on data-centric routes and capture interesting/meaningful data samples. That way, MAs do not blindly walk throughout the network to find event regions; they only move through NDN routes which forward them into source nodes. Although NDN suffers from limited connectivity and node mobility in mobile sensor networks [9], this can be tuned by distributed adaptive caching strategy technique to reduce transmitted traffic and communication overheads [10].

NDN is a routing technique that is used to establish data-centric network paths [11]. This utilises data names rather than address configuration to establish the links [12]. In fact, an NDN route is established by a number of (source) nodes having correlated data names. According to [13], basic NDN forwards data interests (DIs) throughout the network to ask for data contents. As Figure 1 shows, a receiving node replies back and provides the requested data record if the DI matches. Otherwise, this checks its own pending interest table (PIT) and forwards DI if this is new and fits message forwarding strategies which are provided by the forwarding information base (FIB).

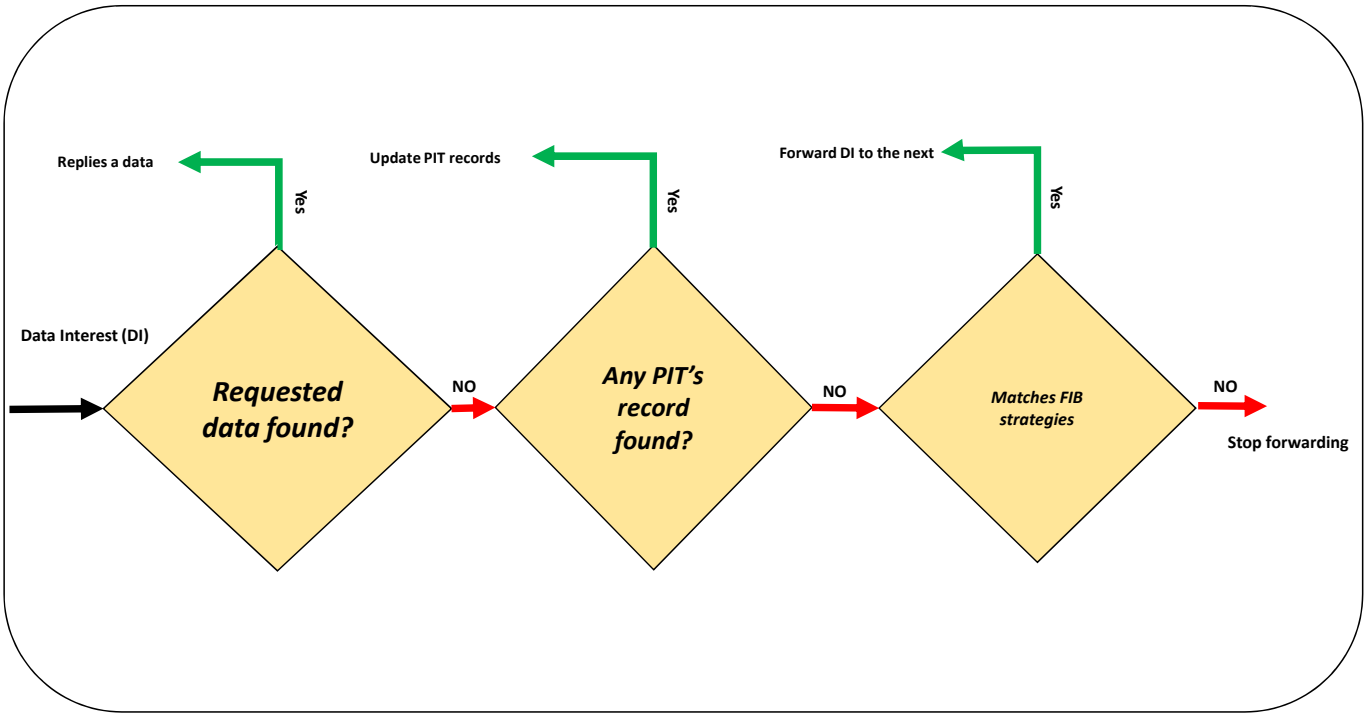

Figure 1. NDN conceptual diagram.

This article proposes MINDS, NDN-enabled "MA itinerary planning in wireless sensor networks". This aims to resolve existing MA routing drawbacks, such as increased data

collection robustness, decreased path length and network resource constraints—mainly bandwidth. MINDS contributions are outlined as below:

- Utilising a lightweight clustering algorithm (Hamming distance) with a minimised communication overhead to support early in-network data aggregation.
- Taking the benefits of named data networking to form a context-aware tree infrastructure with minimised network traffic in a self-organising fashion.
- Using a hop-count-aware Depth-First Search (DFS) algorithm [14] to forward MAs throughout the tree infrastructure to maximise the number of visiting nodes (results in increased data aggregation robustness) through minimised latency and loop-free routes.

MINDS has the capacity to offer Internet of Things (IoT) applications a number of benefits, especially in the context of sensory data collection and aggregation. The key duty of IoT is to provide Machine-to-Machine (M2M) communications which interconnect (sensory) devices via Internet links to transmit data samples. Agent-based protocols are commonly used in IoT-enabled WSNs to enhance application automaticity and network interoperability [15]. Indeed, IoT establishes an Internet-based infrastructure to forward MAs throughout a network and autonomously complete a mission [16]. IoT sensory systems would take the benefits of MINDS, which forwards multiple MAs through Internet links in order that they visit connected sensory stations for data collection and aggregation.

The rest of this paper is organised as follows. Section 2 reports a review on the applications of wireless sensor network data collection. Existing drawbacks and advances in this field of research are compared and contrasted. Section 3 introduces our proposed approach. Remarks on, features of and techniques within the designed MA itinerary planning are described. Section 4 explains the experimental plan and shows how the performance of MINDS was tested and evaluated. Section 5 provides the performance evaluation results of our proposed approach according to four key metrics: end-to-end delay, data robustness, path hop-count and traffic transmitted. The performance of MINDS is compared with two conventional protocols, ZMA [4] and TBID [17]. Section 6 explains MA routing benefits and outlines the key points of this research that have the potential to be addressed as further work.

2. Related Works

A great number of WSN data communication protocols have been proposed in a wide range of applications, from healthcare to urban resource management. They are typically categorised as conforming to one of two models: client/server and MA [3].

For client/server, source sensors (clients) forward data sample through direct or indirect links to the data collection points (server). However, client/server is not very beneficial for WSN applications, especially if the sensory devices are highly mobile and/or equipped with restricted resources—mainly power and bandwidth [18]. A frequent link update to keep/re-establish the communication ties results in increased power and bandwidth consumption. According to [19], a client/server WSN has been proposed to continuously monitor the health status of hypertensive patients using smartphones. Client-side preprocessing is performed to capture and extract electrocardiogram (ECG) features, whereas a cloud server-side computation is performed to analyse and predict healthcare patterns using machine learning techniques, such as neural network and random forest. In reference [20], they utilised a device-to-device paradigm to forward crowd sensing data packets in mobile networks over 4G links. The mobile sensory devices utilise a greedy algorithm to form location-based clusters. Sensory data records are transmitted within each cluster until the target device/station is reached.

MA sensory data collection offers a number of benefits as compared to client/server [3]: (1) The transmission rate is reduced. Client/server forwards raw data samples, whereas MAs collect data samples if they are meaningful and match the query. (2) reduced network resource consumption—mainly bandwidth, as the transmission rate is decreased. (3) Increased scalability due to the invariance of MA data collection performance to the network size. (4) Improved system extensibility, as MA is able to carry out task-adaptive

processes, aiming to extend the system capacities. (5) Enhanced network stability due to MA's ability to support offline (asynchronous) message delivery. (6) Increased load balancing by symmetric workload scattering as compared to client/server in which data communications focus on particular nodes—mainly crowd-sensing data stations.

MAs plan data collection itineraries in two ways: proactive and reactive [4]. The proactive scheme reduces MA migration delay and resource consumption, as routes are established in advance. However, this needs to have all the routing information required to establish paths. This would be challenging, especially in large networks where nodes are highly mobile. Reactive MA path planning is utilised to establish routes on-the-fly using acquired information in a data collection journey. However, MA's blind movements (or random walk) would result in increased migration delay and power consumption. Besides, this can reduce data collection robustness, as a number of source nodes, which have meaningful and interesting data records matching the consumer's query, might be missed by MAs. For this reason, MAs should be programmed to intelligently plan itineraries to collect and aggregate sensory data samples with minimum cost (e.g., power consumption) and delay, and maximum data robustness.

The near-optimal itinerary design algorithm (NOID) [21] utilises synchronous MAs to collaboratively collect sensory data samples from target regions. Each MA starts its journey from the sink and moves via routes which are established using a greedy algorithm to minimise a cost function of the path's hop count and the node's remaining power level. This approach monitors data collection volume at each node. This helps to control MA size and forwards an MA back to the sink if the data load is heavy. However, NOID still suffers from redundant data collection, as collaborative MAs blindly capture and report events in overlapping areas.

Tree-based itinerary design (TBID) [17] forms a zone-based network to forward MAs through multiple spanning trees (SPTs) for data aggregation. Under this protocol, a number of concentric zones are formed around the sink. In turn, each first-zone sensory node aims to establish an SPT with other neighbour source nodes. For this, a greedy tree establishment algorithm is used to incrementally interconnect nodes from outer to inner zones. Hence, a tree trunk is formed by the inter-zone ties, whereas branches are established using the intra-zone paths. An SPT is completed when the last zone is interconnected.

Zone-based mobile agent aggregation (ZMA) [4] reactively forms optimal paths for MA data aggregation according to a bottom-up approach. Similarly to TBID, ZMA forms zones with which to route MAs. However, this utilises a data-centric (DC) zone partitioning technique according to the consumer's DI. ZMA utilises a weight function to select a number of sensory nodes (zone mobile agent coordinators) at each zone to manage MA movements. MAs utilise a bottom-up migration approach to move from target regions to the sink for sensory data collection and aggregation. They record the traversed itineraries to avoid loop and collision.

There are a number of route planning protocols that are dedicatedly designed to forward hardware MAs—mainly UAVs [22,23]. Three-dimensional itinerary planning for UAVs was proposed by [24]. This utilises glow-worm swarm optimisation to route drones over a field of randomly located 3D objects. The target is to find an optimal path to fly over the field with no collisions. For this, a bio-inspired technique, glow-worm swarm, is utilised, which allows the UAVs to move through a path to achieve the journey targets, search the environment and avoid collisions with the existing obstacles on the path. MAs (e.g., drones) may use machine learning techniques for itinerary planning, path optimisation and communication link prediction [25]. One study [26] utilised a deep reinforcement learning technique for autonomous UAV path planning, and [27] extended the work to utilise reinforcement learning for accident avoidance. In [28] a reinforcement learning technique was combined with fuzzy logic to enhance the link connectivity and reduce the path length.

MINDS offers WSN data aggregation applications a number of contributions, especially minimised network traffic, reduced data collection latency and maximised data

aggregation robustness. They are achieved as MINDS establishes a data-centric, minimum hop-count tree infrastructure with reduced network transmission, for MAs to move and capture the maximum number of sensory recordings. The conventional MA data aggregation protocols still suffer from high network transmission rates in route planning. This results in increased network resource consumption, especially power and bandwidth. Moreover, they need to have a data-centric and hop-count-aware route planning through which MAs are forwarded via minimised hop-count paths to capture the maximum number of sensory data samples. This results in enhanced data aggregation robustness and reduced data collection latency at the same time.

3. The Proposed Protocol

A route planning approach is proposed to forward MAs for sensory data collection in rural or hard-to-reach areas. The objectives are to establish optimal MA migration paths, minimise data collection delay and increase data robustness. The proposed approach, called MINDS, forms a context-aware tree-structured route by interconnecting the sink and local sensory data stations (LDSs). This tree infrastructure is reactively formed according to a data-centric pattern to meet the consumer's queries/requirements. LDSs locally capture and aggregate sensory data from (mobile) sensory devices (e.g., wearable sensors) when they are interconnected (e.g., Zigbee ties). They are clustered using a data-centric and lightweight clustering technique, named Hamming distance, to limit network transmission. Each cluster-head joins the tree infrastructure using NDN routing if it has a data sample matching consumer's query. MAs start a data aggregation journey from the sink to visit LDSs residing on the tree-structured route according to a hop-count-aware Depth-First Search (DFS) algorithm. They collect and aggregate sensory data records according to the query and return to the sink to deliver the final results.

The key steps of MINDS are summarised as below:

1. Query Dissemination
   - Technique: An NDN routing to disseminate user's queries in the WSN according to the application features, e.g., time, location and data type.
   - Objective: Reduce transmitted network traffic.
2. Network Clustering
   - Technique: Hamming distance; lightweight, de-centralised and data-centric clustering.
   - Objective: Reduce communication delay and transmission.
3. Tree Route Establishment
   - Technique: Hop-count-aware NDN routing, to establish a data-centric spanning tree for MA migration.
   - Objective: Reduce path hop-count and establish data-centric links.
4. MA Path Planning
   - Technique: hop-count-aware DFS.
   - Objective: Maximise data aggregation robustness, avoid loops and minimise path length.

The network model is comprised of three components: sensory devices, a local data station (LDS) and a sink. Sensory devices are able to freely move and repeatedly capture body and/or ambient sensory data samples, such as body temperature, spo2, air pollution (e.g., PM2.5) and/or traffic congestion according to data collection application. The sensory devices report data samples to LDSs once there is a tie.

LDSs are static stations which collect sensory data records from sensor devices and aggregate them to deliver the results to a network data collector (e.g., MA). They are reactively labelled according to their available datasets (e.g., environmental condition, weather or fire status). For example, an LDS is tagged by "fire" if it recently received fire

status data from mobile sensory devices. However, it may remove the tag once the fire data record is expired.

The sink is the end-user access-point and has sufficient resources (e.g., power and bandwidth) for communication and computation. This retains the duties of consumer query dissemination, data collection/analysis and decision making. Table 1 outlines the acronyms which are used in MINDS algorithm.

Table 1. A glossary for MINDS.

Acronym	Definition	Acronym	Definition
LDS	Local sensory data station	DFS	Depth-First Search
Tm	Timer	CT	Clustering Table
CH	Cluster-head	CM	Cluster Member
SID	Sender's location address	DI	Data Interest
PIT	Pending interest table	FIB	Forwarding Information Base
PHC	Path hop count	ETE	Average End-To-End delay
DAR	Data robustness ratio	TTR	Total transmitted traffic
NDN	Named data networking	MA	Mobile Agent

3.1. Query Dissemination

The sink stays in the charge of consumer's query dissemination. A query messages is comprised of *data_type*, *region_address*, *hop_count* and *time_stamp*. This is disseminated by the sink to the first-hop LDSs. The network avoids utilising traditional broadcast routing approaches (e.g., flooding) to forward the queries. This should result in reduced redundant transmissions and enhanced network resource conservation—mainly bandwidth. For this, NDN routing is used to forward the consumer's query. Each LDS first checks its own routing table to figure out any previous record matching the query's *data_type* and/or *region_address*. It updates the routing table and discards the query message if there is still a fresh record of the query available. However, the LDS forwards the query message to the next-hop ones if this is a new query and matches FIB strategies. For example, an LDS stops forwarding a query message if this is old and/or expired. *hop_count* is used to calculate path length. This is incremented by each LDS when the query is successfully received. Path length is used by MAs later to figure out optimal/shortest route during data collection procedure. *time_stamp* allocates a time value to the query and this is used to support data freshness and/or discard expired ones.

Sensory devices utilise the Hamming distance technique [29] to avoid transmuting redundant and/or misleading data samples to LDSs. Moreover, they use Hamming distance to recognise data samples if they match the consumer's query [30]. This should result in reduced resource consumption and transmitted network traffic, as redundant, uninteresting and/or misleading data samples are not forwarded. According to Algorithm 1, this system counts flipped bits in a binary vector and shows the results as Hamming distance. Then, a data sample is transmitted if its Hamming distance from an existing record is greater than zero (if it is not a redundant value) and/or falls in the user's allowed distance range (for meaningful data based on the application).

Algorithm 1: Hamming distance algorithm [29].

```
Hamming function HAM(D1 ∧ D2)
/*starts from the first bit until end.*/
while D1(i) ≠ '\0' do
    /*checks each bit for both data items (D1 and D2).*/
    if D1(i) ≠ D2(i) then
        /*incremented if the data attributes differ.*/
        distance++;
    end
    /*next bit.*/
    i++;
end
return distance;
```

3.2. Network Clustering

A distributed data-centric clustering approach is utilised to partition an LDS network into a number of clusters. The clustering technique is commonly used to establish hierarchical routing infrastructures according to nodes' similarities—mainly location, sensor recordings and/or computation and communication capacities. Clustering aims to reduce network transmitted traffic and congestion, especially if bandwidth is restricted [18]. Using clustering, simultaneous requests to access wireless channel are reduced. This results in reduced concurrent communications and consequently decreased network congestion compared to flat infrastructure networks. Moreover, data robustness is enhanced in clustered networks [31]. Data robustness is reduced in flat networks, as packet failure is increased due to bottleneck congestion, whereas this is increased in clustered networks, as they avoid bottlenecks by restricting data transmissions into intra-cluster communications.

Algorithm 2 is utilised to form the data-centric clusters. This is a light-weight clustering algorithm that is executed by LDSs which has data records matching the sink's query in a distributed fashion. Unlike centralised clustering, LDSs do not need to transmit their information to a single node (e.g., sink) to form the clusters. This would reduce network transmitted traffic and consequently enhance network resource conservation—mainly regarding bandwidth. According to this algorithm, an LDS that receives a valid query message (in terms of region address and/or time stamp) introduces itself as a candidate cluster-member by broadcasting a *cluster_msg* message. This message is comprised of LDS's connectivity degree (the number of single-hop interconnected LDSs), data type and location address. Each LDS updates its clustering table once a *cluster_msg* is received.

Cluster-head (CH) selection starts at a particular time period after the clustering procedure in a distributed manner. This enhances network resource conservation—mainly bandwidth, as LDSs locally recognise their CHs with minimised communication. CH selection is executed when LDSs receive *cluster_msg* from their next-hop neighbourhood. According to [32], a two-second timer (Tm) is used to support allowed loss since a *cluster_msg* is sent out until a reply is received. That way, each LDS which receives *cluster_msg* checks its own clustering table to recognise the LDS with the maximum connectivity degree and shortest Euclidean distance to the sink. This LDS selects its own CH and joins the cluster as a cluster member (CM) by forwarding data records (*Data_pkt*) for aggregation. An LDS recognises itself as a CH when it receives a *Data_pkt* with no additional communications.

An LDS that already belongs to a cluster as a CM can be still selected as a CH. This is when a cluster member receives *Data_pkt* from an LDS. It marks itself as a CH and sends a *joint_msg* to its own CH. A CH which receives the *joint_msg* becomes a *joint_point*, which is used to form the tree branches and change MA direction. Indeed, an MA changes its direction to sweep three branches if a *joint_point* is visited. This MA route planning will be discussed in Section 3.4. However, an LDS becomes an isolated CH if it receives no *cluster_msg* message from its neighbours and Tm expires.

Algorithm 2: Clustering algorithm.

```
Data: Clustering Table (CT); Timer (Tm);

/*A Valid Query is Received.*/
if (Query.data == LDS. data) then
    Broadcast (cluster_msg);
    Tm.reset;
    /*Updates CT while Tm is not expired.*/
    while (Tm ≠ 0) do
        if cluster_msg ≠ Null then
            Update.CT();
        end
    end
end
/*Invalid Query is Discarded.*/
else
    Discard Query;
end

/*====CH Selection====*/
sort.CT(connectivity | distance);
CH ← CT.top();
if CH ≠ LDS.ID then
    /*This is selected already as CH by other LDSs, but has its own CH.*/
    if Data_pkt ≠ Null then
        Forward joint_msg to CH;
    end
    Forward Data_pkt to CH for aggregation;
end
/*A CH knows no other CHs.*/
else
    if Data_pkt ≠ Null then
        /*Data record is aggregated.*/
        Agg(Data_pkt);
    end
    /*Marks itself as Joint CH.*/
    else if joint_msg ≠ Null then
        joint.flag ← true;
    end
end
```

3.3. Tree Route Establishment

MAs move through a tree-structured route to visit LDSs and collect the data records using a hop-count-aware DFS algorithm. According to Algorithm 3, a CH that knows no CHs are around forwards a reply beacon toward the sink to form the tree path. This beacon is included by data type, the sender's location address (SID), hop count and time stamp. Hop count is used to avoid loops and recognise an optimal path for MAs to move through. This is incremented by each intermediate LDS until the sink is reached. A NDN routing is used to forward the beacon messages. Each intermediate LDS which receives the beacon message checks PIT to find a matching record. Similarly to [33], the LDS updates PIT and passes the beacon's information to FIB to check the message forwarding strategies if this is a new beacon. FIB allows the forwarding of the beacon and incrementation of *hop_count* if the LDS has a closer Euclidean distance than the beacon sender to the sink. Otherwise, the beacon is discarded. The beacons are forwarded until the sink is reached. An LDS still

becomes a *joint_point* if receives a beacon matching PIT records from LDSs which have farther distances to the sink.

Algorithm 3: Tree establishment.

```
Data: beacon (data_type, SID, hop_count, time_stamp);

/*beacon forward.*/
if LDS.CH == CH then
    Broadcast beacon;
end

/*until sink is not reached.*/
while LDS ≠ sink do
    /*beacon is received from farther area.*/
    if beacon.SID ≥ LDS then
        /*This is a new beacon.*/
        if beacon ∉ LDS.PIT then
            /*Matches FIT.*/
            if beacon ∈ FIT then
                hop_count++;
                forwards beacon;
            end
        end
        /*A true beacon, but a copy is already available.*/
        else
            joint.flag ← true;
        end
    end
    /*A wrong beacon from closer area is received.*/
    else
        Discard beacon;
    end
end
```

3.4. MA Path Planning

MAs go on data collection journeys from the sink to LDSs residing on the tree. MA migration is similar to ZMA, in which MAs move through each data zone to visit sensory data stations [4]. However, MINDS is a top-bottom MA route planning as compared to ZMA, which is bottom-up. Moreover, MINDS utilises a hop-count-aware DFS algorithm to select the paths, whereas ZMA works according to a particular weight function of connectivity degree and residual energy level to reactively establish the routes.

MINDS forms a data-centric tree path which forwards MAs to only LDSs whose data records match the query. That way, MAs avoid random walking that consequently results in reduced MA power consumption. Hence, this offers great benefits to hardware MA applications that usually are equipped with restricted power resources. Each MA is allocated to stack memory, called *re_visiting*, which works according to last-in-first-out data retrieval. This is used to keep the address of *joint_points* to visit tree branches as next move, yet MAs record their own itinerary to avoid loops and visiting overlapping areas.

Algorithm 4 illustrates MINDS which forwards MAs to collect sensory data samples. According to this, each sub-tree is allocated to an MA to move from the sink. An MA visits LDSs residing on the allocated tree one-by-one until a *joint_point* is visited. Then, this updates its own *re_visiting* by the region address of the *joint_point*. This address is used to change the direction and collect data records from the non-visiting LDSs which are interconnected to the *joint_point*. Indeed, an MA returns to *joint_point* and visits other non-visiting branches when an LDS with no next-hop link is visited. A hop-count-aware DFS algorithm is used to select the next direction at each *joint_point*. According to this, a LDS with minimum *hop_count* is selected as the next visiting LDS and other ones are left as non-visiting until next move. An MA removes a *joint_point* from *re_visiting* when all the tree branches are visited. This is repeated until *re_visiting* becomes empty with no next-hop link to visit. The MA eventually reruns to the sink to deliver the results. Figure 2 depicts the MA route planning approach. However, a UAV might return to the sink in the middle of a journey if the power resources are running out. For this, the address of the last visited LDS is recorded for a return later. The MA resumes its journey from the last visited LDS when it is fully re-charged.

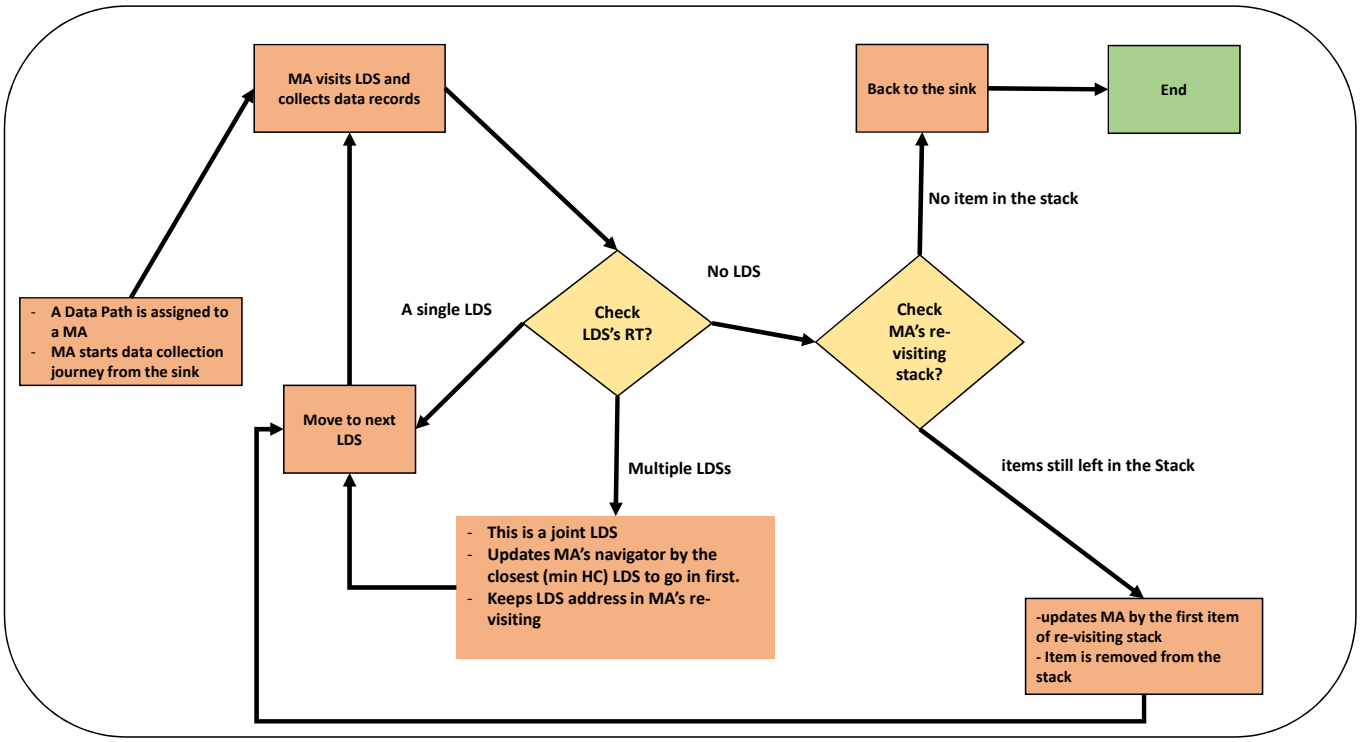

Figure 2. MA route planning diagram.

Algorithm 4: UAV migration algorithm.

```
Data: Routing Tables (RT);
MA(ID, Data, Code, Itinerary);
Next LDS(NLDS); Last Visited (LV);
/*MA starts to move.*/
Move ← true;
while (Move == true) do
    /*Run-out power.*/
    if MA.power ≤ threshold then
        Move ← false;
        LV ← LDS.ID;
        MA.itinerary(sink);
    end
    /*Resume a move.*/
    else if LV ≠ Null then
        MA.itinerary(LV);
        LV ← Null;
        Move ← true;
    end
    else
        /*Multiple branches.*/
        if joint.flag == true then
            if LDS.ID == sink then
                /*MA allocation at sink.*/
                while RT.LDS ≠ Null do
                    NLDS ← RT.LDS.top();
                    RT.LDS.remove();
                    MA.ID(i).itinerary(NLDS);
                    i++;
                end
            end
            else
                /*Hop-count DFS.*/
                MA.re-visiting() ← LDS.ID;
                sort.RT.min(HC);
                NLDS ← RT.top();
                NLDS.MA ← MA.ID;
                MA.itinerary(NLDS);
            end
        end
        /*A single route is available.*/
        else if RT.LDS.count() == 1 then
            NLDS ← RT.top();
            MA.itinerary(NLDS);
        end
        /*Preparing the return path.*/
        else
            NLDS ← MA.re-visiting();
            /*No re_visiting is left.*/
            if NLDS == Null then
                MA.itinerary(sink);
                Move ← false;
            end
            /*A re_visiting is found.*/
            else
                re-visiting.remove(NLDS);
                MA.itinerary(NLDS);
            end
        end
    end
end
```

4. Experimental Plan

The performance of MINDS was evaluated using a simulation technique. Real MA route planning applications (e.g., drones) need additional resources (e.g., infrastructure and equipment) and coordination, especially in a wide rural and/or hard-to-access areas. Hence, simulations are usually used to test MA route planning approaches and evaluate

results and performance. OMNET++ [34] is a well-known component-based framework for network communication simulation (e.g., WSN). This utilises an INET [35] framework which supports node mobility patterns, MAC protocols and communication models. This supports mobile ad-hoc networks in which MAs are simulated to move throughout the network to perform particular tasks. In addition, there are implementations of well-known MA itinerary planning protocols—mainly ZMA and TBID in OMNET++. A metropolitan grid (M-Grid), the Manhattan model [36], was used for LDS localisation. The simulation parameters are discussed in the next section.

There are four metrics that were measured in this research: average end-to-end delay, data robustness, path hop-count and total transmitted traffic. According to the literature [3], these metrics are commonly used to measure and evaluate the performances of MA route planning protocols.

Simulation Setup

This simulation aimed to run the data collection experiments on a field of 5.5 km $\times$5.5 km. As Figure 3 shows, there are two parameters to simulate experiments: sensory device count and data density. That way, varying numbers of sensory devices, 256, 1024 and 4096, were randomly distributed throughout the network. This allowed us to observe and evaluate the route planning approach behaviour, scalability and performance if the node count changes.

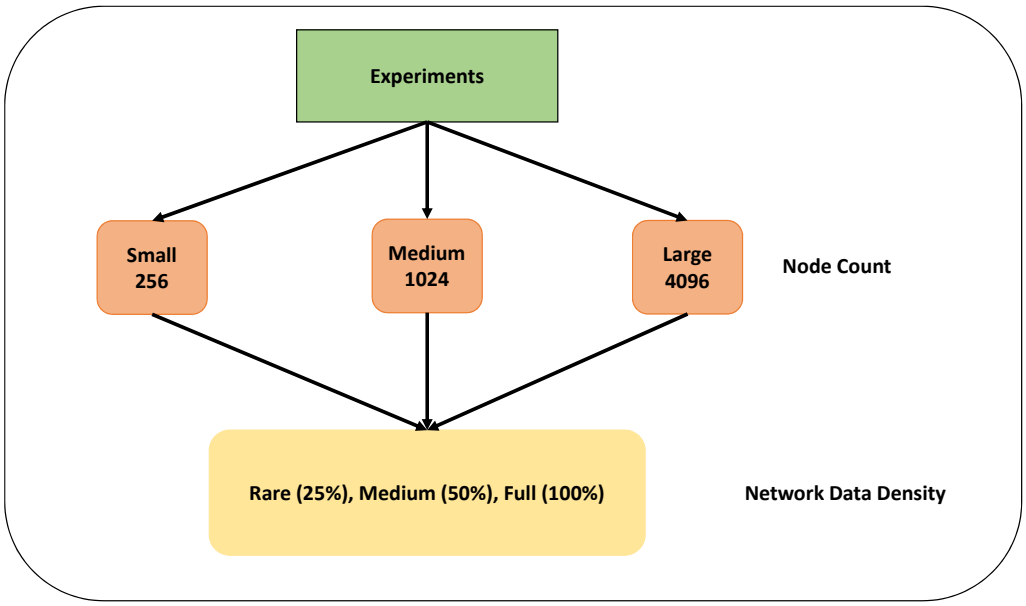

Figure 3. Parameter sweeping plan.

Each sensory device count was assigned by three data density proportions, including 25% (low), 50% (medium) and 100% (full). This means the network's node count consists of a proportion of source nodes reporting data samples to LDSs. For example, a network of 256 and 25% data density is comprised of 64 active sensory devices which transmit data records if a tie is found to a LDS, whereas other ones (192 nodes) have no relevant data to report and MA collection. This allows us to study the ability of route planning protocols to reactively detect event regions according to requested data samples (defined by the user on-demand) and forward MAs with minimum random walk to capture them. Under this simulation, each sensory device is assigned by a unique ID and randomly moves with a constant speed through a pre-defined road map, which is provided according to M-Grid mobility pattern.

As Figure 4 shows, the data collection map was simulated according to a Manhattan model. The blocks (11$\times$11) had a size of 400 m. The roads were two-lane with 100 m width. LDSs were localised at the middle of junctions (500 m distance between each pair) and

formed a grid communication infrastructure. Each one was assigned by an ID containing grid row and column number. The sink was located in the centre (block) of the grid.

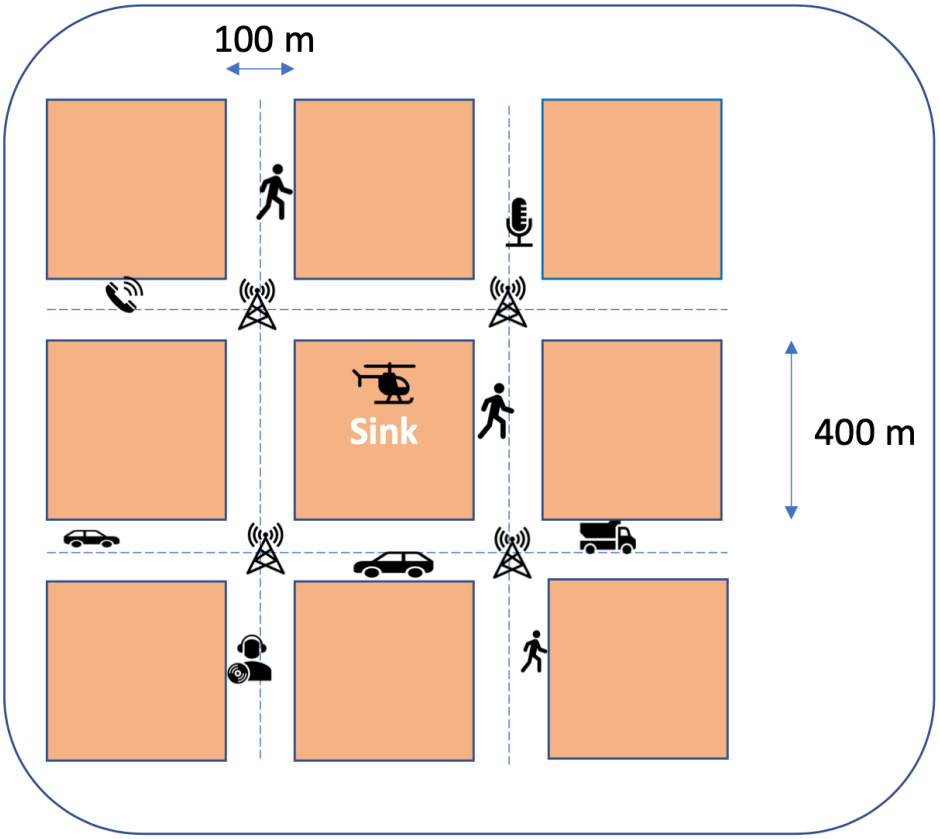

Figure 4. M-Grid mobility model.

This was a collision-free 2D simulation. The network nodes (including MAs and sensory devices) were set up to communicate through wireless transceivers (IEEE 802.11) with a limited 200 m range of communications. A line of sight (LOS) wireless signal propagation model was used for all the simulation experiments. This model assumed no wireless interference by obstacles such as trees/buildings or ambient factors (e.g., noise and weather). A Carrier Sense Multiple Access (CSMA/CA) MAC protocol was used in the simulation experiments to provide wireless channel sharing and avoid collisions. MAs move with a constant speed of 30 m/s.

The statistical power analysis technique [37] was used to compute experimental repetitions required in this research. For this, population standard deviation was calculated for a (randomly selected) subset of experiments. This helped to compute the number repetitions based on the experiment's confidence level. For this, the simulation results of 20 repetitions (the experiment subset) were collected and analysed. According to the subset results, at least 250 repetitions (sample size) are recommended to achieve 90% confidence for all the experiments. Table 2 summarises the simulation parameters.

Table 2. Simulation parameters.

Parameter	Range	Parameter	Range
Simulation time	3600 s	network initialisation	100 s
repetition	250	ambient noise and obstacles	disabled
communication range	200 m	node distribution model	random
LDS distance	500 m (grid)	Area size	5.5 km × 5.5 km
MAC	CSMA/CA	Communication protocol	IEEE 802.11
mobility pattern	minimum traffic	Data density	25%, 50%, 100%
Node count	256, 1024, 4096	MA speed	30 m/s

5. Results and Discussion

The performance of MINDS is evaluated in this section according to four metrics: ETE, DAR, PHC and TTR. It is compared with ZMA and TBID to understand the superiority and inferiority of the proposed approach. ZMA and TBID were selected, as there are available on OMNET++.

5.1. Average End-to-End Delay

Average end-to-end delay (ETE) is calculated to study the performance of MA route planning protocols to capture and aggregate sensory recordings and deliver results to the sink. The number of aggregated sensory recordings and the length of MA migration path are two key parameters that influence ETE. As Figure 5 shows, MINDS increases ETE compared to the benchmarks, especially in large and crowded networks where the data density proportion is high. This is because of the protocol's ability to find a greater number of event regions to collect data records. Indeed, delay is increased as MINDS spends a greater deal of time to collect a greater number of data records and report the results to the sink.

ZMA reduces ETE as compared to MINDS. This is because of utilising bottom-up MA movement through which mobile agents are forwarded from the middle of event regions to collect data records. That way, MAs do not need to establish a journey from the sink to the event regions; they collect sensory data from each region and deliver the aggregated results to the sink. ZMA assumes that each (zone leader) LDS dispatches an MA to a data zone for data collection. This will increase the parallelism degree of MA data collection and reduce ETE when a greater number of MAs move to collect sensory data samples. However, MINDS slightly underperforms when the data density is low and the network deployed is small. This is because of the ability of MINDS to minimise path hop-count from the sink to LDSs. MAs do not need to reactively establish a path; they migrate through the minimum hop-count routes which are already available. This reduces ETE when a small number of LDSs have sensory data to be collected.

ETE is reduced in TBID compared to ZMA and MINDS. This is because TBID works according to an address-centric fashion, and therefore has no ability to find the event regions (or LDSs) throughout the network. That way, TBID leaves a number of LDSs non-visited, especially if the network is large and crowded. Moreover, TBID offers no minimum hop-count paths to forward MAs to collect and aggregate sensory recordings.

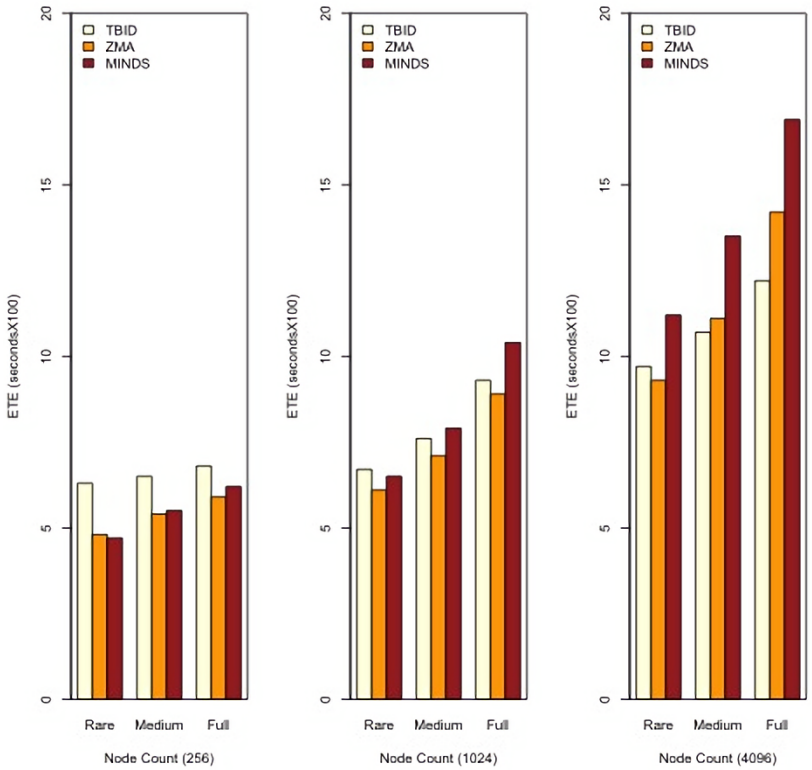

Figure 5. Average end-to-end delay.

5.2. Data Robustness Ratio

The data robustness ratio (DAR) is measured to study the ability of MA route planning protocol to detect event regions and collect data records which match the consumer's query. A protocol offers benefits to remote-sensing applications if it is able to have a high DAR.

According to Figure 6, MINDS increases DAR as compared to the benchmarks. This is rooted in the protocol's ability to visit event regions and LDSs which have data records matching the consumer's query. MINDS works according to a data-centric path planning that establishes a tree route with LDSs which have interesting data samples to collect. That way, MAs should be able to collect meaningful data samples if they only move throughout the path.

MINDS outperforms ZMA because of utilising clustering technique and NDN routing. Clustering forms data zones which are interesting for MAs to visit. However, NDN routing forms data-centric paths which interconnect LDSs whose data records match the consumer query. That way, MAs should be able to find event regions and capture data samples if they move through the paths. Indeed, MAs avoid random walking. They visit only the minimum number of LDSs whose data records match the consumer's query.

TBID underperforms compared to MINDS and ZMA for DAR. This is because of TBID establishing a tree structure according to an address-centric manner. This utilises no data-centric feature to establish the routes. For this, the tree is formed with a number of LDSs which might have no data records matching the query. This results in MA migration to a hole which has no meaningful data records to report. Hence, DAR is reduced especially when data density is low.

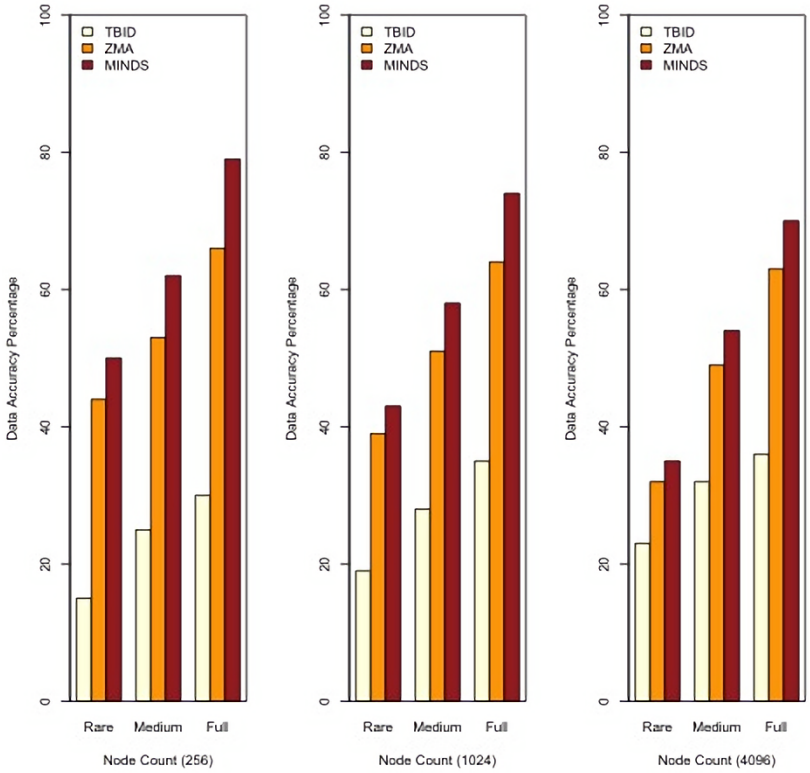

Figure 6. Data robustness percentage.

5.3. Path Hop Count

This measures the path hop-count through which MAs move to collect data records and return the results. Path hop-count is used to study the protocol's capacity in minimising hop-count and/or establishing optimal MA migration paths. This results in reduced power consumption and decreased ETE. That way, MAs should avoid randomly moving and blindly visiting the source sensor to capture recordings.

According to Figure 7, MINDS outperforms ZMA. This is because MINDS forms a minimum hop-count data-centric tree. That way, MAs avoid blindly moving throughout the network to collect data records which match the consumer's queries. Indeed, this tree allows MAs to move toward event regions which already have the requested data samples. Moreover, the clustering technique limits MA migration only into a restricted area which is formed according to a data-centric manner. That way, MAs do not need to visit all LDSs which have the data records; they only visit CHs to collect the aggregated results. However, ZMA plans MA itineraries using all LDSs which have data samples according to the consumer's query. For this, MAs need to visit all LDSs at each data zone one by one until the sink is reached.

TBID reduces PHC as compared to MINDS and ZMA, especially in sparse networks. This is because TBID is not able to detect event regions and/or LDSs which have sensory data matching the consumer's query when the network deployed is sparse. However, this increases when the network deployed is crowded and data density if full. This is rooted in that a greater number of LDSs join to form a longer address-centric path. This results in increased path length as MAs blindly move throughout the network to visit LDSs and capture data records, especially when the network is crowded.

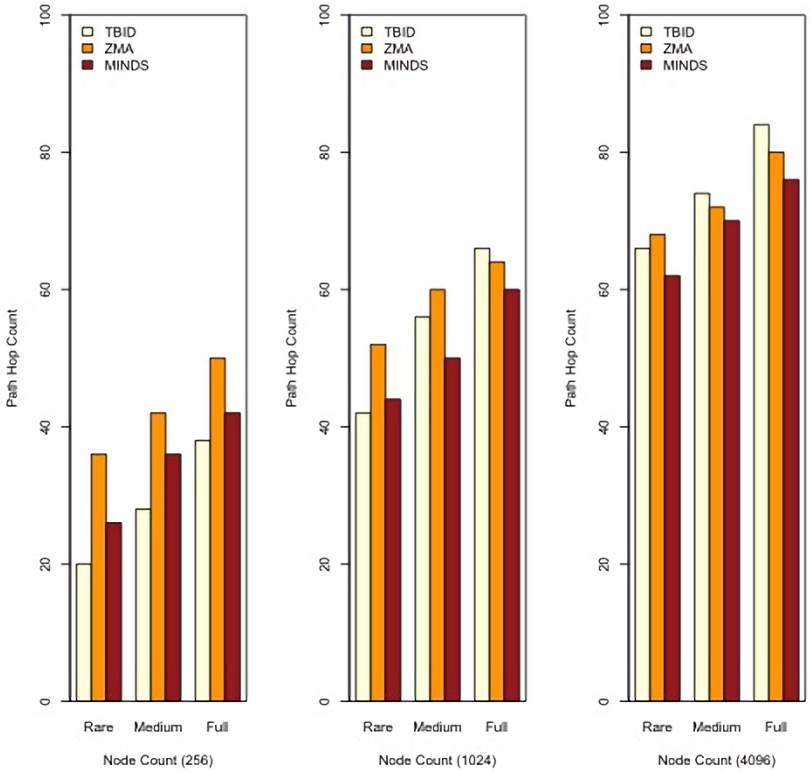

Figure 7. Data collection path hop-count.

5.4. Total Transmitted Traffic

Network message collision/failure is a key drawback for a network with restricted bandwidth, especially when transmitted traffic is increased. Total transmitted traffic (TTR) includes network and data messages which are forwarded during MA route planning and data collection.

According to Figure 8, MINDS reduces TTR as compared with the benchmarks. This is because of utilising NDN routing, which avoids transmitting redundant and/or meaningless messages. NDN routing utilises a data-centric technique which only allows forwarding network traffic if it matches a routing forwarding strategy. That way, LDSs avoid forwarding/broadcasting the network traffic blindly. Moreover, this utilises a clustering technique which limits the network communication to intra-cluster messages. This means that network traffic is reduced, as only a few LDSs participate in route planning. Indeed, CHs are the only nodes which forward the route planning messages, and CMs do not individually participate in route planning. The Hamming distance technique is a lightweight clustering technique which utilises a reduced number of network messages to form the clusters. It is performed in a distributed manner and avoids widely transmitting network messages for clustering.

TBID forwards network messages with no restrictions to establish address-centric routes. This increases transmitted network traffic if the network deployed is crowded and several LDSs participate in the route planning procedure.

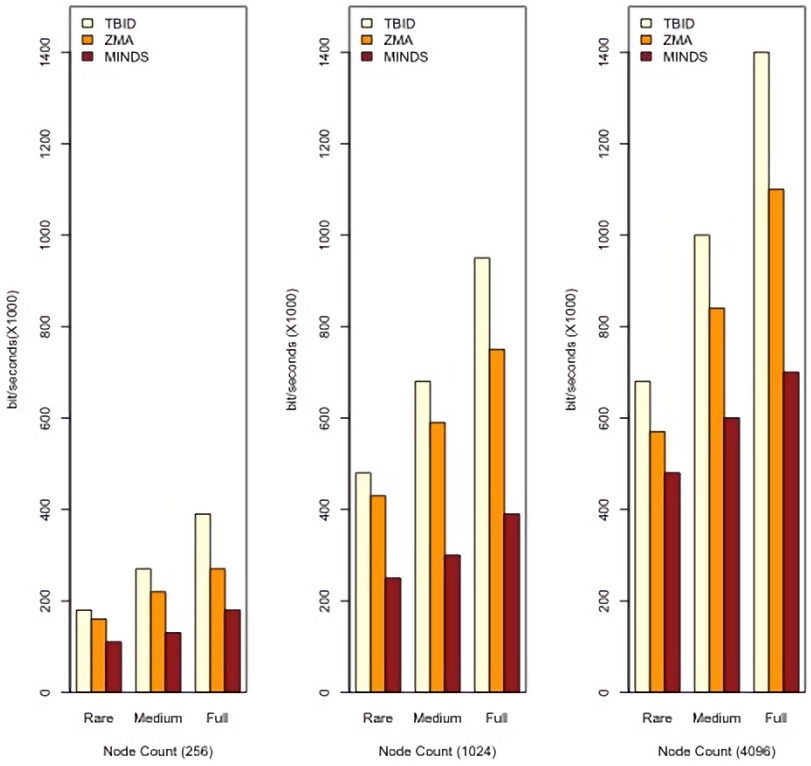

Figure 8. Transmitted network traffic.

6. Conclusions and Further Work

The proposed approach forms a data-centric and hop-count-aware tree infrastructure by interconnecting the sink and LDSs. This tree is reactively updated using data-centric NDN communications according to the consumer's queries/requirements. Under MINDS, MAs use a hop-count-aware Depth-First Search (DFS) algorithm to move through the tree and visit LDSs, which locally capture and aggregate sensory data from (mobile) sensory devices. LDSs are clustered using a data-centric and lightweight clustering technique named Hamming distance, to limit network transmission.

MINDS offers a number of benefits, including increased data robustness, reduced path length and decreased transmitted network traffic as compared to two benchmarks—ZMA and TBID. It avoids blind/random walk, as it forwards MAs through a minimum hop-count and data-centric tree. MINDS enhance data aggregation robustness, as MAs will be able to capture a greater number of interesting/meaningful sensory recordings when they move through the data-centric tree. In addition, MINDS reduces the network transmissions, as it uses the Hamming distance technique to form data-centric clusters. That way, the network traffic is limited only into intra-cluster communications. MINDS avoids forwarding MAs into overlapped/loop areas to collect redundant data records, because it allows MAs to record itineraries and utilise a hop-count-aware DFS algorithm to monitor data collection paths. However, MINDS underperforms compared to ZMA in ETE. ZMA supports bottom-up MA routing, as it assumes each LDS is equipped with an MA. This reduces ZMA's ETE because there is no need for MAs to move from the sink to LDSs.

In future, LDS synchronisation needs to be studied further. Synchronisation plays a key role in MINDS, especially for LDS communications. The network will experience frequent message conflicts, network disconnections and/or transmitted traffic congestion with lack of synchronisation. Each LDS should utilise a private timer for synchronisation, especially during clustering and data collection, whereas LDS infrastructure still needs a public or partial synchronisation.

MINDS still needs to be implemented as a real-world application to investigate the performance of the proposed approach in real test scenarios (e.g., real-world network stress

test) and study new findings. Due to cost and risk concerns, a simulation was used in this research to test and evaluate MINDS. However, MINDS still needs further evaluations according to a physical implementation using real MAs (e.g., drones) and sensory data collection stations.

The proposed approach needs to be re-designed if LDSs are mobile. Localising an LDS infrastructure in a wide and crowded urban area is expensive. That way, mobile LDSs—mainly vehicles—would offer benefits to crowd-sensing applications, as they do not need particular infrastructure to set-up. They are able to play LDS roles in an ad-hoc fashion. Indeed, the vehicles move throughout the city and collect crowd-sensing data samples from crowd-sensing devices through an infrastructure-less network.

Further investigation on MINDS with different LDS network topologies is still required. This research establishes an LDS network in a grid which is widely used to study the average networking behaviour. However, it would be interesting to study the performance of MINDS when LDS networks are established in a mesh or star formation.

Funding: The author would like to extend sincere thanks to University of Nottingham Ningbo China for supporting this research project under UNNC FOSE Grant: I01200900006 and I01200300007.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Al-Karaki, J.; Kamal, A. Routing techniques in wireless sensor networks: A survey. *IEEE Wirel. Commun.* **2004**, *11*, 6–28. [CrossRef]
2. Alvear, O.; Calafate, C.T.; Cano, J.C.; Manzoni, P. Crowdsensing in Smart Cities: Overview, Platforms, and Environment Sensing Issues. *Sensors* **2018**, *18*, 460. [CrossRef]
3. Ardakani, S.P. Wireless Sensor Network Routing Protocols for Data Aggregation. Ph.D. Thesis, Computer Science Department, University of Bath, Bath, UK, 2014.
4. Ardakani, S.P.; Padget, J.; Vos, M.D. A Mobile Agent Routing Protocol for Data Aggregation in Wireless Sensor Networks. *Int. J. Wirel. Inf. Netw.* **2017**, *24*, 27–41. [CrossRef]
5. Lingaraj, K.; Biradar, R.V.; Patil, V.C. Eagilla: An Enhanced Mobile Agent Middleware for Wireless Sensor Networks. *Alex. Eng. J.* **2018**, *57*, 1197–1204. [CrossRef]
6. Yanmaz, E.; Yahyanejad, S.; Rinner, B.; Hellwagner, H.; Bettstetter, C. Drone networks: Communications, coordination, and sensing. *Ad Hoc Netw.* **2018**, *68*, 1–15. [CrossRef]
7. Bian, C.; Zhao, T.; Li, X.; Yan, W. Boosting named data networking for data dissemination in urban VANET scenarios. *Veh. Commun.* **2015**, *2*, 195–207. [CrossRef]
8. Li, T.; Ota, K.; Wang, T.; Li, X.; Cai, Z.; Liu, A. Optimizing the Coverage via the UAVs With Lower Costs for Information-Centric Internet of Things. *IEEE Access* **2019**, *7*, 15292–15309. [CrossRef]
9. Chen, M.; Mau, D.O.; Zhang, Y.; Taleb, T.; Leung, V.C.M. VENDNET: VEhicular Named Data NETwork. *Veh. Commun.* **2014**, *1*, 208–213. [CrossRef]
10. Yang, N.; Chen, K.; Liu, Y. Towards Efficient NDN Framework for Connected Vehicle Applications. *IEEE Access* **2020**, *8*, 60850–60866. doi:10.1109/access.2020.2981928. [CrossRef]
11. Ahed, K.; Benamar, M.; Lahcen, A.A.; Ouazzani, R.E. Forwarding strategies in vehicular named data networks: A survey. *J. King Saud Univ. Comput. Inf. Sci.* **2020**. [CrossRef]
12. Mau, D.O.; Zhang, Y.; Taleb, T.; Chen, M. Vehicular Inter-Networking via Named Data—An OPNET Simulation Study. *Lect. Notes Inst. Comput. Sci. Soc. Inform. Telecommun. Eng.* **2014**, *137*, 116–125.
13. Saxena, D.; Raychoudhury, V.; Suri, N.; Becker, C.; Cao, J. Named Data Networking: A survey. *Comput. Sci. Rev.* **2016**, *19*, 15–55. [CrossRef]
14. DFS. Depth First Search—Iterative and Recursive Implementation. 2019. Available online: https://www.techiedelight.com/depth-first-search/ (accessed on 15 February 2020).
15. Savaglio, C.; Ganzha, M.; Paprzycki, M.; Bădică, C.; Ivanović, M.; Fortino, G. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Gener. Comput. Syst.* **2020**, *102*, 1038–1053. [CrossRef]
16. Yousefi, S.; Derakhshan, F.; Karimipour, H.; Aghdasi, H.S. An efficient route planning model for mobile agents on the internet of things using Markov decision process. *Ad Hoc Netw.* **2020**, *98*, 102053. [CrossRef]
17. Konstantopoulos, C.; Mpitziopoulos, A.; Gavalas, D.; Pantziou, G. Effective Determination of Mobile Agent Itineraries for Data Aggregation on Sensor Networks. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1679–1693. [CrossRef]

18. Ardakani, S.P.; Padget, J.; Vos, M.D. CBA: A cluster-based client/server data aggregation routing protocol. *Ad Hoc Netw.* **2016**, *50*, 68–87. [CrossRef]
19. Jovanovic, S.; Jovanovic, M.; Skoric, T.; Jokic, S.; Milovanovic, B.; Katzis, K.; Bajic, D. A Mobile Crowd Sensing Application for Hypertensive Patients. *Sensors* **2019**, *19*, 400. [CrossRef]
20. Wang, Y.; Li, H.; Li, T. Participant selection for data collection through device-to-device communications in mobile sensing. *Pers. Ubiquitous Comput.* **2016**, *21*, 31–41. [CrossRef]
21. Gavalas, D.; Mpitziopoulos, A.; Pantziou, G.; Konstantopoulos, C. An approach for near-optimal distributed data fusion in wireless sensor networks. *Wirel. Netw.* **2010**, *16*, 1407–1425. [CrossRef]
22. Thibbotuwawa, A.; Bocewicz, G.; Nielsen, P.; Banaszak, Z. Unmanned Aerial Vehicle Routing Problems: A Literature Review. *Appl. Sci.* **2020**, *10*, 4504. [CrossRef]
23. Khan, M.F.; Yau, K.L.A.; Noor, R.M.; Imran, M.A. Routing Schemes in FANETs: A Survey. *Sensors* **2020**, *20*, 38. [CrossRef]
24. Goel, U.; Varshney, S.; Jain, A.; Maheshwari, S.; Shukla, A. Three Dimensional Path Planning for UAVs in DynamicEnvironment using Glow-worm Swarm Optimization. In Proceedings of the International Conference on Robotics and Smart Manufacturing (RoSMa2018), Chennai, India, 19–21 July 2018; pp. 230–239.
25. Bithas, P.S.; Michailidis, E.T.; Nomikos, N.; Vouyioukas, D.; Kanatas, A.G. A Survey on Machine-Learning Techniques for UAV-Based Communications. *Sensors* **2019**, *19*, 5170. [CrossRef]
26. Wang, J.; Xiao, X.; Lu, P. A Survey of Vehicular ad Hoc Network Routing Protocols. *J. Electr. Electron. Eng.* **2019**, *7*, 46–50. [CrossRef]
27. Li, Y.; Zhang, S.; Ye, F.; Jiang, T.; Li, Y. A UAV Path Planning Method Based on DeepReinforcement Learning. In Proceedings of the IEEE USNC-CNC-URSI North American Radio Science Meeting (Joint with AP-S Symposium), Montreal, QC, Canada, 5–10 July 2020; pp. 93–94.
28. He, C.; Liu, S.; Han, S. A Fuzzy Logic Reinforcement Learning-Based Routing Algorithm For Flying Ad Hoc Networks. In Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC), Big Island, HI, USA, 17–20 February 2020. [CrossRef]
29. Hamming, R.W. Error Detecting and Error Correcting Codes. *Bell Syst. Tech. J.* **1950**, *29*, 147–160. [CrossRef]
30. Xiong, Z.; Liveris, A.D.; Cheng, S. Distributed Source Coding for Sensor Networks. *IEEE Signal Process. Mag.* **2004**, *21*, 80–94. [CrossRef]
31. Sambo, D.W.; Yenke, B.O.; Forster, A.; Dayang, P. Optimized Clustering Algorithms for Large Wireless Sensor Networks: A Review. *Sensors* **2019**, *19*, 322. [CrossRef]
32. Chakeres, I.D.; Belding-Royer, E.M. The utility of hello messages for determining link connectivity. In Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC), Honolulu, HI, USA, 27–30 October 2002; Volume 2, pp. 504–508.
33. Amadeo, M.; Campolo, C.; Molinaro, A. Forwarding strategies in named data wireless adhoc networks: Design and evaluation. *J. Netw. Comput. Appl.* **2015**, *50*, 148–158. [CrossRef]
34. OMNET++. OMNET++ Simulator. 2019. Available online: http://www.omnetpp.org/ (accessed on 21 March 2019).
35. INET. INET Framework. 2019. Available online: https://inet.omnetpp.org/Introduction.html (accessed on 21 March 2019).
36. Martinez, F.J.; Cano, J.C.; Calafate, C.T.; Manzoni, P. CityMob: A mobility model pattern generator for VANETS. In Proceedings of the IEEE International Conference on Communications Workshops, Beijing, China, 19–23 May 2008.
37. AUSVET. Sample Size to Estimate a Single Mean with Specified Precision. 2014. Available online: https://epitools.ausvet.com.au/onemean?page=1Mean&Stdev=45&Conf=0.95&Error=20 (accessed on 15 February 2020).