International Journal of Web and Grid Services (IJWGS)

NARUN: Noise Adaptive Routing for Utility Networks

Fabio Pagnotta*, Leonardo Mostarda*, Orhan Gemikonakli, Rosario Culmone*, Diletta Cacciagrano*, Flavio Corradini***

*Computer Science Division, Camerino University, 62032 MC, Italy
E-mail: {fabio.pagnotta,leonardo.mostarda,rosario.culmone,
diletta.cacciagrano,flavio.corradini}@unicam.it

**Final International University, Toroslar Avenue,
No: 6, 99370, Catalkoy, Kyrenia via Mersin 10 Turkey, TRNC
E-mail: orhan.gemikonakli@final.edu.tr

Abstract: Wireless Meter-Bus is an open standard for power-efficient smart metering. Data are collected from meters and transmitted to the collector for processing. In smart cities, placing meters with the best quality communication signal is often challenging for urban constraints and other communication signals. Meters can also have limited capabilities in terms of memory and CPU. Previous work has been addressing the reliability issue only in the context of direct collector-meter communication. This paper proposes a novel Noise Adaptive Routing for Utility Networks (NARUN) protocol for improved performance and efficient routing in a partially connected mesh network. The collector keeps a weighted graph of the whole network where weights define the link failure index. No keep-alive or control messages are used to update the weights. Meters eavesdrop on the surrounding environment and efficiently report link failure indexes to the collector with ordinary reading messages. We validate NARUN on a real case study.

Keywords: WM-Bus; Smart Metering; Routing.

Biographical notes:

Fabio Pagnotta is a PhD student at Computer Science Division of Camerino University where he received his BSc and Msc with full marks. His research interest includes the modeling and simulation of IoT systems.

Leonardo Mostarda is currently an Associate Professor at the Computer Science Division, Camerino University and CEO of Bilancio CO_2 zero. In 2007 he was Research Associate at the Computing Department, Imperial College London. In 2010 he was Senior Lecturer at networking department, Middlesex University (UK). His research activities are in the areas of IoT, middleware and security.

Prof. Orhan Gemikonakli obtained a PhD in Telecommunications from King's College London in 1990 where he worked as a post-doctoral research associate. Then, he joined Middlesex University, where he served as the Head of Department of Computer and Communications Engineering (2007-2014), and Professor of Telecommunications (2012-2018). Currently, he is the Dean of the Faculty of Engineering at Final International University. His current research is in telecommunications, performance evaluation of complex systems, computer networks, and algorithms.

Prof. Rosario Culmone is a researcher at the University of Camerino. He teaches programming languages, databases and software engineering. He has been interested in web interaction patterns and more recently in networking for Ambient Assisted Living and smart grid.

Diletta Romana Cacciagrano is Assistant Professor at the University of Camerino. Her research activities include formal methods for complex systems, knowledge modeling, and model-based design of cyber-physical systems.

Flavio Corradini is Professor at the University of Camerino. He was Assistant Professor (1996-2000) and Associate Professor (2000-2003) at the University of L'Aquila. From 1996 to 1997 he was Visiting Researcher at Sussex University (UK). His research activities include formal specification, verification of concurrent, distributed and real-time systems.

1 Introduction

The sheer size of today's cities makes the manual reading of gas, water, and electricity meters a significant challenge. Fortunately, the technological advances in communication networks gave rise to the development of automated meter reading systems. Network meter systems enable automated, periodic, and real-time readings, reducing the management costs and enabling real-time data analysis. This technology can also help in reducing resource consumption by raising the awareness of prosumers. Directive 2009/72/EC of the European Parliament requires the use of smart metering systems to empower the consumers in the electricity and gas supply markets [1]. A metering system consists of a set of distributed nodes connected via a communication network. A collector node gathers data from meter nodes that measure usage. Data is then sent to the collector node. Repeaters can be used to extend the node transmission range. The main electronic parts of a meter consist of a simple micro-controller with sensors. These devices are limited in terms of computation power, memory and can be battery powered. Hence, the highest cost is not associated with the device itself but with the subsystem of interconnection and communication. These subsystems can use optical fiber, WiFi, cellular networks, or other transmission technologies. The network deployment and technologies used in communication represent the critical parts of the reading service [2].

Meter Bus (M-Bus) is a European standard for metering systems that includes the physical layer, the data-link layer [3], and the application layer [4]. The network layer is presented as an optional one and there is no specification for any standard routing algorithm. No topology restrictions are stated, except for the token ring which cannot be used. Wireless Meter Bus (WM-Bus) is the wireless version of the M-Bus protocol. The EN 13757-4 standard describes its physical and data-link layer [5]. The EN 13757-5 provides node relaying [6] whereas the EN 13757-7 introduces transport and security services [7]. The line of sight communication range can be a few hundred meters (with 868 MHz or 433 MHz frequencies) or more than 5 km (using 169 MHz frequency). The former is widely used in Europe [8]. In this scenario, a reliable and energy-efficient routing protocol is needed to extend the coverage of smart meter networks [9].

In smart cities [10, 11, 12], meter installation introduces a trade-off between reliable connectivity and setup cost. Placing meters with the best quality communication signals is often expensive or it is not feasible because of urban constraints. WM-Bus networks can be

installed in dense urban areas where many other communication signals coexist. This can cause interference and generate poor quality links [13], which increases communication overhead and reduce the lifetime of the battery-operated meters. Poor quality links also affect the data reading time since a successful reading message can require several attempts. Previous work has been addressing the scarce link quality in the context of direct collector-meter communication [14, 15]. Often channels with Gaussian noise are assumed, using some error recovery strategy for improving communication performance. The WM-Bus standard suggests appending the routing path to the reading request that is sent by the collector. In fact, a meter is usually a simple device with limited CPU and a small amount of memory that cannot hold much information. The broadcasting of routing service messages should be also avoided in order to minimise the messages that are sent, thus, maximising the meter lifetime.

In this paper, we propose a novel protocol; Noise Adaptive Routing for Utility Networks (NARUN) for WM-Bus networks. In NARUN, communication occurs in a mesh topology without sending extra service messages. The routing path is appended to the read request message (as advised by the WM-Bus standard). A collector can request and receive data from networked meters. Meters can relay the request to the destination node and forward back the response. The collector decides the routing path depending on the internal representation of the network. This knowledge is represented as a graph where nodes are vertices and edges are interconnections. The edge weight gives a measure of the link failure index. More precisely, an infinite weight can denote a broken link, a weight equal to one denotes a perfect functioning link. Finally, a number greater than one denotes a noisy link that requires frequent re-transmission or error recovery. The routing path is calculated by minimising the failure index and path length. Meters eavesdrop on the surrounding environment and efficiently report information on link failure indexes back to the collector with ordinary reading messages. These are ordinary response WM-Bus packets. NARUN can run on simple meter devices with limited CPU and a small amount of memory. Hamming Error Correction Code (ECC) is used to increase communication reliability and measure the link failure index (this is denoted with lfi in the rest of the paper). To evaluate our approach, we compared the following four routing methods through simulation: (i) a WM-Bus routing where the network graph has not weighted edges (i.e., no link failure information is provided); (ii) an ECC-WMBUS routing where the network graph has no weighted edges, and the Hamming code is used to recover errors; (iii) the NARUN protocol where the network graph has edges with weight one or infinity (i.e.,we have only information about whether the link is working or not); (iv) the ECC-NARUN protocol where the network graph has weighted edges and the Hamming code is used to recover errors. We also compared NARUN with the Dynamic source routing (DSR) [16]. This is an efficient and well-known protocol that is suitable for wireless sensor networks [17, 18]. For simulations, a real-life topology is chosen from the San Paolo district of Camerino (MC, Italy). We analysed the collector failure rate and the collector reading rate. A collector failure is consequence of a path that fails to reach the destination meter. A collector can select a failing path when its network graph is not updated (i.e., a broken or noisy link is not correctly updated). The collector reading rate estimates the number of sensors that are correctly read by the collector. The simulations are performed by varying the percentage of noisy links (i.e., 30%, 15% and 5% of the total number of links). We also vary the noise power. Our results show that NARUN has the highest reading rate with the lowest failure rate and the least traffic load. For instance, when 30% of the links are noisy with a noise power equal to -70 dBm, ECC-NARUN and DSR read 99% of the sensors. WMBUS reads only to 71% of the

sensors. ECC-NARUN has a failure rate of 4.9 % while DSR has a failure rate of 37.14%. In fact, DSR generates a number of messages that are one order of magnitude higher than NARUN.

2 Related Work

Different frameworks are proposed for measuring, and analysing the failure rate of the WM-Bus for different cases. A framework for the analytical study of failure rates is presented as a case study where the suitability of WM-Bus for smart water grids is considered in [19]. The approach considered performance measures such as Packet Error Rate (PER), bandwidth, frequency, range, and energy requirements. The simulation framework for failure rate analyses of WM-Bus is also considered in [20]. The authors discussed the approach for developing a simulation of the WM-Bus protocol through extending NS-3[1] simulator. This work focused mostly on the physical and data-link layers of WM-Bus. The implementation details for this approach are given in the study [21]. The paper [22] analyses the WM-Bus for home automation systems by using 5G and M2M technologies. Technical details associated with the 868 MHz communication, such as range and indoor signal transmission efficiency under various interference levels, are studied in [23]. The same frequency is also used in this paper. Simulations were used to analyse the failure rate by varying the noise power in a smart city network.

Improvements to the reliability of the WM-Bus have been considered in some studies. In papers [14, 15], authors presented discussions on the inherent reliability problems related to the deployment of the WM-Bus network and they proposed a method for reliable data reception. In particular, the authors addressed the trade-off between reliability and the deployment cost for WM-Bus networks, which is one of the fundamental and inherent problems of the WM-Bus. The meter nodes are not mobile and since they are added to existing infrastructures, their optimum deployment becomes difficult. The authors proposed a data recovery scheme exploiting the use of a deterministic packet transmission interval from the meter nodes to deal with this issue. In [24], the authors proposed a method to increase the reliability and the network lifetime of WM-Bus. The proposed method extends the standard WM-Bus protocol making it suitable for “home energy management systems” which requires bi-directional real-time communication. Authors added functions such as asynchronous meter trigger, adaptive slot scheduling, and bitmap-wise re-transmission request to WM-Bus to achieve reliable and energy-efficient real-time communication. In this paper, the proposed approach uses along with the inherited single-hop reliable mechanism of WM-bus.

Routing protocols can be reactive or proactive. Proactive protocols can be link-state or distance vector protocols. These require periodic messages to maintain the information stored in nodes. In link-state protocols such as Open shortest path first (OSPF), the network topology is stored by nodes and is maintained through message flooding. Flooding notifies the whole network when link costs change or a link failure occurs. In distance-vector protocols, nodes store and maintain their local view of the network which includes the set of neighbours along with their respective link costs. This information is used to route messages in the network. Nodes periodically send messages to update the local view of neighbors. Although these protocols seem ideal for networks with a fixed topology (modulo failures), they are not suitable for battery-equipped devices with limited capacity in terms of memory and CPU since periodic messages can reduce the lifetime. PSA-HD [25] is

a proactive protocol used in MANET networks where nodes are mobile and links are frequently created and broken. This protocol tries to find the shortest stable path in the network. In detail, the shortest path is computed by minimizing the REfined Hamming DIStance (REHDIS) metric required to reach the destination node. The selection of the shortest path is performed amongst multiple paths which metrics are shared between nodes with periodical message exchanging. PSA-HD decreases the average end-to-end delay and increases the Packet Delivery Ratio (PDR) with respect to Proactive Source Routing (PSR) [26] and Predicted Probabilistic Coefficient Link Stability (PPCLSS) [27] protocols. PSR is a proactive protocol where the network topology is shared between nodes while PPCLSS is a routing protocol where stability is evaluated using the weighted sum of energy utilized by a node, link loss and the average path size. Like PSA-HD, the NARUN protocol presented in this paper, uses Hamming distance for shortest path computation but NARUN path updates are done without exchanging periodic messages. Generally speaking, proactive protocols are not suitable in smart cities composed of many battery-operated nodes that have limited processing capabilities. In this case the usage of periodic service messages (such as HELLO messages) are unsuitable since they require high bandwidth and high memory allocation.

Reactive protocols are designed to restrict the bandwidth consumed with service packets by removing the periodic service messages. Routes are generated on-demand when a node starts a new communication. Dynamic source routing (DSR)[16] is one of these reactive protocols. At the beginning, a node performs route discovery in order to reach the destination node. The route discovery floods the network with route request packets. The destination node that receives this type of packets responds by sending a route reply packet back to the source. The source node stores the paths used to reach the destination. Link fault is recognized when the node tries a faulty path. The maintenance of these paths does not include service messages used to update the routing tables. The major limitation of this protocol is the absence of a local routing maintenance mechanism of a broken link. Therefore, flooding is also required when the sender has no suitable path to reach a destination node. Such a process may restrict the bandwidth available in a dense network. The path found using the discovery process, may also not necessarily be the shortest one between the given pair of source and destination nodes. Ad hoc On-Demand Vector (AODV) [28] is quite similar to the DSR. Both protocols find routes through flooding. However, DSR stores the routes in the source node while AODV distributes and stores the routing information in the entire network. While AODV scales with a large population of nodes, the size of routing tables (stored in intermediate nodes) increases as well. This augmentation makes the protocol unsuitable for networks composed of devices with a limited amount of memory. In [29], the authors proposed an AODV extension. The protocol performs route discovery by partially flooding the network. Topology information is sent back in the response packet after a specific amount of time. The source node calculates the shortest path to reach a specific destination by minimising the packet reception rate and the number of hops. Their simulation results show an improvement in AODV packet reception rate when noisy network environments are considered. The packet reception rate depends on the distance between nodes and the nominal transmission range. While the latter is a known value, the former is fixed at the beginning or computed by exchanging the geographical coordinates. The optimal reactive routing protocol (ORRP) [30] is a reactive protocol that finds the shortest path between the source and destination nodes in a distributed fashion. The shortest path is calculated using Dijkstra algorithm. The protocol assumes symmetric links between neighbouring nodes and that each node maintains the cost of links. In [31], the authors propose a proactive extension of this protocol by introducing a periodic HELLO message exchange for sensing

neighbourhood as well as for the determination of cost list. However, this protocol generates a huge overhead when the network is dense.

Various routing approaches specifically designed for WM-Bus networks have been proposed. In [32], the study presented a model routing extension for the Wireless M-Bus Q-mode in TinyOS[2]. In [33, 34] the authors proposed an energy-aware routing that is implemented as an extension to the EN13757 Wireless M-Bus Q-mode [6]. A cost-efficient integrated energy harvesting system powered by the available water flow was developed to enable operation independent from the main grid. This eliminates the need for battery replacement with near-zero maintenance costs. A noteworthy idea for routing in low power-lossy link wireless networks comes from the study presented in [35]. The proposed routing considers metrics involving the residual energy and packet reception rate of neighbor nodes in ISA100.11a industrial wireless networks. In [36], the authors proposed a routing protocol for Wireless Mesh Networks by modifying the AODV routing protocol. The proposed protocol considers the load of nodes and link quality. In our work, we proposed novel extensions for reliability and routing of WM-Bus communication.

3 The NARUN protocol

We first introduce some notation that is used to describe the network model and the protocol. Let's denote the set of all network nodes $n_0, \ldots, n_z$ with N, and the WM-Bus read request frame packet with M. M is composed of three fields that are the header, the payload, and the footer which are denoted with $M.h$, $M.d$ and $M.o$, respectively.

$G(N, E, w)$ denotes a network graph where (i) N is the set of nodes; (ii) $E \subseteq N \times N$ is a finite set of links, $(n_i, n_j) \in E$ when n_i can directly communicate with n_j; (iii) $w : E \longrightarrow R$ is an edge weight function.

We define three weight functions: (i) constant; (ii) connection-based; and (iii) Hamming-based. The constant weight function, $w^c(n_i, n_j)$ assigns to each link (n_i, n_j) a constant number 1. This can be used to find the path with the least number of hops from the collector to any meter on the network. The connection-based weight function, $w^r(n_i, n_j)$ is ∞ when the link (n_i, n_j) is not working, and 1 otherwise. As shown in the next section, w^r can be used by the collector node to avoid broken links or faulty nodes.

The Hamming-based weight function, $w^h(n_i, n_j)$ is calculated by using the Hamming code. The header $M.h$ and the payload $M.d$ of a frame M are divided into l equal parts $M_1, \ldots, M_l$. The sender of the packet M calculates the Hamming code $hmc(M_i)$ of each part M_i and adds it into the footer M_o. The format of the message (when the Hamming-based weight function is used) can be summarized as follows [3]: $M_t = M_1 || \ldots || M_l || hmc(M_1) || \ldots || hmc(M_l)$. We also define the correction function $R(M_i)$ as follows:

$$R(M_i) = \begin{cases} 0 & M_i \text{ is received with no error} \\ \infty & M_i \text{ has a non recoverable error} \\ 1 & M_i \text{ has a recoverable error} \end{cases} \quad (1)$$

and $w^h(n_i, n_j)$ by the following equation:

$$w^h(n_i, n_j) = \begin{cases} \infty & \text{if } \sum_{i=0}^{l} R(M_i) = \infty \\ \frac{\sum_{i=0}^{l} R(M_i)}{l} + 1 & otherwise \end{cases} \quad (2)$$

In other words, $w^h(n_i, n_j)$ is set to ∞ when M has a part M_i with no recoverable error. Otherwise, w^h is a number between one and two. It is one when no error is recovered, while it is two when each part M_i has a recoverable error. Effectively, $w^h(n_i, n_j)$ measures the link failure index lfi between n_i and n_j. As we are going to see in the next section, the weight function w^h can be used by the collector node not only to avoid faulty links or faulty nodes but also to select the path with the lowest link failure index.

In the rest of the paper we denote with $P_{n_k}(G) = \{n_0, \ldots, n_k, n_{k+1}, \ldots, n_0\}$ (with $k > 0$) a path that starts from the collector node n_0, reaches the node n_k and returns to the node n_0. This can be used by the collector to read the node n_k. The cost of the path can be calculated as follows:

$$W(P_{n_k}(G)) = \sum_{i=0}^{i=k-1} w(n_i, n_{i+1})$$

In this paper, we assume symmetric links that is the path P_{n_k} will be palindromic (i.e. the paths from n_0 to n_k and n_k to n_0 are the same). In fact, WM-Bus links are rarely asymmetric.

In the rest of the paper, $G_{n_0}(N_0, E_0, w_0)$ denotes the network graph of the collector node. Each meter node n_s also defines a meter network graph (from here onward, referred to as projection graph) which is denoted by $G_{n_s}(N_s, E_s, w_s)$. N_s is the set of neighbors of n_s that is $n_i \in N_s$ when n_i is in the communication range of n_s. The set E_s contains communication edges of the type (n_s, n_j) and (n_i, n_s) where the node n_s takes the role of sender and receiver, respectively. A projection graph is used by the meter n_s to store its local network view that is all its neighbors and the failure index of the related links. As we are going to see in the following, NARUN keeps the collector graph updated by performing an efficient merge of all projection graphs.

We assume that the collector node keeps a global timestamp t that is a sequence number. The collector node increases t by one when a meter reading is attempted. We assume the collector node n_0 reads all meter nodes $n_1, \ldots, n_z$ in turn. A timestamp t_q denotes the q^{th} attempt that is made by the collector to read a meter n_k. Two consecutive timestamps t_q and t_{q+1} may be related to the same meter node n_k when the routing path selected at time t_q fails to reach n_k. In this case, a different path can be tried at time t_{q+1}.

In the rest of the paper, $G_{n_k,t_q}(N_{k,t_q}, E_{k,t_q}, w_{k,t_q})$ denotes a meter graph last updated at time t_q. Each $n_j \in N_{k,t_q}$ is a neighbor of n_k added at time t_u with $t_u \leq t_q$. The link $(n_k, n_j)_{t_u}$ would also be added to E_{k,t_q} at time t_u. For the sake of simplicity, we do not show the protocol for adding new nodes into the network. This is done by broadcasting standard hello messages. The weight $w_{k,t_u}(n_k, n_j)$ of the link (n_k, n_j) denotes a weight that was updated at some time t_u with $t_u \leq t_q$.

3.1 Communication primitives and the message format

We denote the WM-Bus frame packet the collector node uses to read a meter node at time t_q by M_{t_q}. Then, the three fields; the header, the payload, and the footer are denoted by $M_{t_q}.h$, $M_{t_q}.d$, and $M_{t_q}.o$, respectively. The payload $M_{t_q}.d$ contains a NARUN application layer message d that complies with a standardized Payload structure. More precisely, the payload d is defined as Payload d = {Type y, node n, TimeStamp t, GraphList l, Numeric r, Path p} where: (i) d.Y specifies the types of the message, that is either REQ or $REPLY$ (the former defines a payload that contains a reading request sent by the collector to a meter

while the latter is the reading returned value); (ii) d.N contains the meter n_k to be read (this is often referred to as the destination node); (iii) d.P contains the path P_{n_k} that leads to the destination node and back to the collector node; (iv) d.T contains the global timestamp t_q; (v) d.L contains a list of projection graphs $G_{n_1,t_q}, \ldots, G_{n_h,t_q}$, each graph G_{n_s,t_q} is added by the node n_s as the payload travels along the routing path; (iv)d.R contains the reading of the node d.N (if any).

The list of projection graphs $G_{n_1,t_q}, \ldots, G_{n_h,t_q}$ is used to update the collector node graph G_{n_0,t_q}. This field is the NARUN addition to ordinary WM-Bus packet to updated the collector weights without extra messages. The algorithm of Figure 1 sketches the merge procedure. This considers each link (n_j, n_i) that is contained inside each projection graph G_{n_s,t_q} in the list. The procedure updates the weight of the collector link (n_j, n_i) when the one received from the meter has a fresher timestamp.

Algorithm 1 NARUN **merge** of projection graphs

```
procedure Merge(GraphList L)
  for each G_{n_s,t_q}(N_{s,t_q}, E_{s,t_q}, w_{s,t_q}) ∈ L do
    for each (n_j, n_i) ∈ E_{s,t_q} do
      Let t_u be the update time of w_{s,t_u}(n_j, n_i) ∈ G_{n_s,t_q}
      Let t_v be the update time of w_{0,t_v}(n_j, n_i) ∈ G_{n_0,t_q}
      if t_v < t_u then                          ▷ true if the collector receives a fresher weight
        w_{0,t_v}(n_j, n_i) ← w_{s,t_u}(n_j, n_i)
      end if
    end for
  end for
end procedure
```

NARUN uses send and receive primitives. The send(n,m) primitive can be used to send a message m from a node to its neighbor n. Acknowledgements are used in order to provide reliable one hop communication. An error is returned when the destination node cannot be reached after a pre-set number of communication attempts. The receive(n,m) primitive can be used by a node to receive a message m from its neighbor n. This blocks the node execution until the message is received or a timeout is generated. The send(n,m) communication primitive is used to define a more abstract NARUN send primitive that is sendN (see Algorithm 2). This can be used by the node n_s to send the message m to its neighbor n_d. The link weight between n_s and n_d is updated according to the $weight$ function that has been selected. More precisely, when n_s fails to send m to n_d the weight $w_{s,t_q}(n_s, n_d)$ is updated to infinity. This means that at current time t_q the link n_s, n_d is broken. Otherwise the function UpdateFailureIndex is executed which updates the weight $w_{s,t_q}(n_s, n_d)$ by considering the selected weight function that is constant; connection-based; or Hamming-based.

3.2 NARUN collector behavior

The collector node uses its network graph in order to read each sensor node. While the least cost path is obtained by using a simple Dijkstra's algorithm, NARUN uses a novel routing protocol strategy to update the collector network graph weights. More precisely, unlike most of the protocols, NARUN does not use any keep-alive or any control messages to update variation in link failure index. In NARUN, each node locally collects link failure index information in the form of projection graphs. This is performed by using meter nodes that continuously eavesdropped on surrounding communications and update their local

Algorithm 2 NARUN **Send** primitive primitive

```
1: procedure SendN(node n_s, node n_d, Graph G_{n_s,t_q},Payload d)
2:     M ← generateFrame(d)
3:     Result ← send(n_d, M)
4:     if Result=Error then
5:         w_{s,t_q}(n_s, n_d) ← ∞                    ▷ weight of link updated at time t_q
6:         return Failure
7:     else
8:         UpdateFailureIndex(w_{s,t_q}(n_s, n_d))    ▷ weight of link updated at time t_q
9:         return Success
10:    end if
11: end procedure
```

projection graphs. When the collector node selects a routing path to read a sensor node, all nodes in the routing path will add their local projections to the reply message. The projections will be merged and used by the collector to update its network graph. This strategy avoids the use of keep-alive or any other control messages.

Algorithm 3 Collector n_0 behavior

```
Require: discard any received message with less than timestamp t
Require: update global variable G_{n_0,t}(N_{0,t}, E_{0,t}, w_{0,t}) with new connected/disconnected nodes
1: procedure CollectorLoop(int m)
2:     t ← 0
3:     for each node n_k inside cluster N_{0,t} do
4:         Result ← readOperation(n_k,max)          ▷ try at most max times to read the same sensor n_k
5:     end for
6: end procedure
7: procedure readOperation(node n_k, int m)
8:     i ← 0                                        ▷ counter for making sure that no more than max attempts are done
9:     toBeMerged ← false                           ▷ true when a unary graph must be merged with G_{n_0,t}
10:    G1_{n_0,t} ← G_{n_0,t}                       ▷ A reference to collector graph G_{n_0,t} is put into G1_{n_0,t}
11:    res ← null
12:    do
13:        P_{n_k} ← Dijkstra(G1_{n_0,t},n_k)       ▷ best path P_{n_k} = {n_0, ..., n_k, ..., n_0}
14:        if P_{n_k} is not empty then             ▷ A path leads to n_k
15:            t ← t + 1                            ▷ timestamp incremented when a read is attempted
16:            i ← i + 1
17:            res ← ReadMeter(n_k,G1_{n_0,t},P_{n_k})
18:            if res ≠ Failure then                ▷ a value was read
19:                Break
20:            end if
21:        else
22:            G1_{n_0,t} ← Clone(G_{n_0,t},1)      ▷ a unary graph
23:            toBeMerged ← true
24:        end if
25:    while i ≤ m                                  ▷ Try all possible paths to n_k
26:    if toBeMerged then                           ▷ read a value
27:        Merge(G1_{n_0,t})
28:    end if
29:    return res
30: end procedure
```

Algorithms of Figure 3 describe the collector behavior. This discards all messages that have an old timestamp. An external procedure that connects new nodes and disconnects leaving ones is assumed. WM-Bus uses hello and disconnect messages. The variable $G_{n_0,t}$ is assumed to be a global one that contains the collector network graph. Without loss of generality, we assume the collector reads the data of all sensor nodes in turn. The procedure CollectorLoop(int m) performs a for loop (lines 3-5) that reads all sensor nodes in turn. This is performed by using the procedure readOperation(node n_k, int m) that tries to read each sensor n_k at most max times (this is denoted with m in the algorithm of Figure 3).

More precisely, when a reading attempt fails, readOperation will try again until a read is performed or the max number of attempts max has been reached. The result of line 4 can be either the value read or the error failure. The lines 7-30 of Figure 3 describes the code of readOperation. The do-while cycle of lines 12-25 will try at most m routing paths to read the meter n_k. More precisely, the shortest path P_{n_k} is selected by considering the current network graph $G1_{n_0,t}$ (line 13). When a path is found, the procedure ReadMeter is called (line 17). This procedure tries to reach the node n_k by using the selected path. In case n_k is successfully read, the do-while cycle terminates (line 19), otherwise another path is tried. The variable P_{n_k} is empty when each path that leads to n_k has a link with infinity weight. In this case, lines 22-23 are executed. These set the pointer of the temporary variable $G1_{n_0,t}$ to a unary clone of the global network graph $G_{n_0,t}$. $G1_{n_0,t}$ is a copy of $G_{n_0,t}$ that has the same edges, nodes and timestamps but all the weights set to one. The unary graph $G1_{n_0,t}$ allows the collector to retry paths that have an infinity link weight in the global network graph $G_{n_0,t}$. When the unary graph is used, the merge algorithm of Figure 1 is used to update the weights of the network graph $G_{n_0,t}$ with the new discovery weights from the unary graph (line 27). The procedures terminates by providing the result that is either the sensor reading (line 19 was executed), or the error Failure (max paths were tried but none of them reached the sensor), or null (no path exists). This last option should never happen since we assume that each node connects and disconnects correctly from the collector.

Algorithm 4 Reading procedure of the collector node n_0

```
1: procedure ReadMeter(node n_k,Graph G_{n_0,t}(N_{0,t}, E_{0,t}, w_{0,t}), Path P_{n_k})
2:     Payload d                                  ▷ {Type y, node d, TimeStamp t, GraphList l, Numeric r, Path p}
3:     d.y ← req                                  ▷ WM-Bus request of reading
4:     d.n ← n_k
5:     d.p ← P_{n_k}
6:     d.t ← t                                    ▷ sequence number of attempted reading
7:     d.l ← ∅                                    ▷ local-graph set l = {G_{n_1}, ..., G_{n_h}}
8:     Result ← SendN(n_0, n_1, G_{n_0,t}, D)
9:     if Result = Failure then
10:        return Failure                         ▷ Collector cannot contact next hop n_1
11:    end if
12:    if TimeOut receive(n_1, M) then            ▷ Reply message not received
13:        w_{0,t}(n_1, n_0) ← ∞                  ▷ Message was not successfully sent
14:        return Failure                         ▷ No path leads to the node
15:    end if
16:    Merge(G_{n_0,t},m.L)                       ▷ Update graph G with received local-graphs
17:    if Contains(n_k, M.L) then                 ▷ Data value received from n_k
18:        return M
19:    else
20:        return Failure
21:    end if
22: end procedure
```

Algorithm 4 outlines the collector node's reading procedure. The collector node reads the sensor value of a node n_k by taking into account the collector network graph G_{n_0}. The procedure starts by defining the payload. This contains the type of message (in this case, a request REQ of reading), the destination node, the routing path, the timestamp and the list of projection graphs that is set to empty. A local projection graph G_{n_i} is added to the reply list by any node in the path in order to keep the weight of the collector node graph updated. The collector uses the sendN communication primitive in order to forward the request to the node n_1. This is the next node in the routing path. When the sending fails, the collector node returns an error (in this case, sendN updates the weight of the link between n_0 and n_1 to infinity); otherwise, the collector waits for the reply message (line 12 of the

algorithm). When no $reply$ is received the collector node assumes its communication link to the first node of the path (i.e., n_1) is not working; thus the collector sets the weight of the link $w_{0,t}(n_1, n_0)$ to infinity and returns a Failure error. When a reply value is returned, the collector calls the Merge function in order to update its local network graph G_{n_0} with the list of projection graphs $M.L = \{G_{n_1,t}, \ldots, G_{n_h,t}\}$. The list of projection graphs is further analyzed. More precisely, when the projection graph G_{n_k} of the node to be read (i.e., n_k) is in the list, it means the data was successfully read and the reply message M is returned. When the projection graph G_{n_k} is not in the list, it means the data was not successfully read and a Failure error is returned. The complexity of the Collector behaviour is polynomial w.r.t. the number of nodes and the number of edges. In fact, the Dijkstra algorithm (which is polynomial) is executed a constant number of times for each sensor reading.

3.3 Meter behavior

A meter n_s uses an eavesdrop procedure to sniff each message that is exchanged between neighbors (see Algorithm 5 for details). More precisely, this is a message that is sent from n_j to n_i with $n_i \neq n_j \neq n_s$. This indirect observation allows n_s to update the link failure index of its projection graph.

Algorithm 5 Narun eavesdrop routine

```
1: procedure eavesdrop(node n_s, Graph G_{n_s}(t)(N_s, E_s, w_s))
2:     Message m
3:     while true do
4:         sniff(m)                          ▷ sent from n_j to n_i with n_i ≠ n_j ≠ n_s and n_j ∈ E_s
5:         if message from n_j to n_i is Readable then
6:             UpdateFailureIndex(w_{s,m.T}(n_j, n_s))
7:         end if
8:         if ack from n_i to n_j is Readable then
9:             UpdateFailureIndex(w_{s,m.T}(n_i, n_s))
10:        end if
11:    end while
12: end procedure
```

The Algorithm 6 shows the reading behavior of a meter n_s. This loops forever waiting for incoming payloads (line 3 of the algorithm). When a payload D is received from a node n_{s-1} the timestamp of the local graph $G_{n_s,t}$ is updated with the received one (i.e., D.t.) and the failure index of the link n_{s-1}, n_s is also updated. When the meter is the destination one (i.e., the one to be read), a read is performed and the type of the message is changed to reply (lines 9-10 of the algorithm). The local projection graph is added to the received payload (this is needed to update the collector node). Finally, the payload is forwarded to the next hop (line 14). When the next hop cannot be reached, the payload is updated with the new projection graph since the procedure sendN would set the weight of the link (n_s, n_{s+1}) to infinity. The Reply message is sent back to the previous node in the path. In this case, no reception of the message is checked.

3.4 NARUN connectivity

As we are going to see in the simulation Section, NARUN is suitable for dense networks that are composed of many links between meter nodes. This setting can be found in many applications, such as smart metering in smart cities. Dense networks ensure connectivity and fast reading since the collector network graph always has updated links. Figure 1 shows

Algorithm 6 Meter n_s reception behavior after bootstrap

Require: discard any message with less than t
Require: update $G_{n_s,t}(N_s, E_s, w_s)$ with new connected/disconnected nodes
1: **procedure** meterReceive
2: **Payload** d ▷ {Type y, node d, TimeStamp t, GraphList l, Path p}
3: **while** true **do**
4: WaitReceive(n_{s-1}, D) ▷ sent by the previous node in the routing path
5: $G_{n_s,t}(t \leftarrow$ d.T)
6: UpdateFailureIndex($w_{s,d.T}(n_{s-1}, n_s)$)
7: **if** d.y=REQ **then** ▷ A request from collector
8: **if** d.n $= n_s$ **then** ▷ I am the destination
9: d.r $\leftarrow$ read() ▷ fill the reply with my reading
10: d.y $\leftarrow$ reply ▷ set message type to reply
11: **end if**
12: **end if**
13: d.L $\leftarrow$ d.L $\bigcup G_{n_s,t}$ ▷ add local graph
14: $n_{s+1} \leftarrow$ getNextHop(D.p) ▷ get the next hop
15: **if** SendN(n_s, n_{s+1}, d)$\neq$ Success **then** ▷ when the next hop is unreachable
16: d.y $\leftarrow REP$ ▷ a reply is sent back
17: d.L $\leftarrow$ d.L $\bigcup G_{n_s,t}$ ▷ add local graph
18: SendN(n_s, n_{s-1}, d)
19: **end if**
20: **end while**
21: **end procedure**

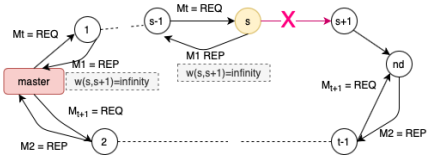

Figure 1: Updating a broken link

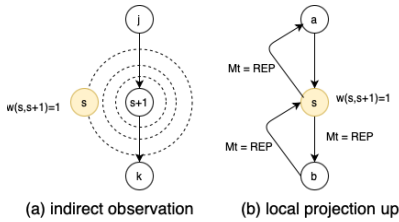

Figure 2: Indirect update of s by eavesdropping $s+1$ communication

the detection of a broken link by the collector node. The collector node tries to read a node n_d at time t by using a message M_t. The first broken link in the path (the link $(s, s+1)$ in Figure 1) will cause a timeout and the sending of a reply back to the collector. This is used to update the collector graph with the broken link (i.e., $w_{s,t}(s, s+1) = \infty$ in the example of Figure 1). The collector will retry to read n_d by using an alternative path at time $t+1$ (see message M_{t+1} of Figure 1). NARUN can update the weight of the link $(s, s+1)$ without the need of any extra service messages. This is performed in two steps that are (i) indirect observations, and (ii) local projection updates.

Indirect observations will eventually allow the detection of a link that starts functioning again. For instance in the example of Figure 1 the collector node will eventually try to route a message via the nodes s and $s+1$. This must necessarily happen when the collector tries to read these nodes. For instance node s can update the link failure index of $(s+1, s)$ when the routing path includes $s+1$ (see Figure 2). In this case, s can eavesdrop on the messages/ACK sent by $s+1$ to another node and update its local projection graph. This update can be sent back to the collector node when s is part of a routing path. This allows the collector to correctly update the link $(s, s+1)$. Symmetrically, $s+1$ can update the link failure index of $(s, s+1)$ when the routing path includes s. Similar observations can be done for a node n_s that is not working. In these cases, all links that have n_s as destination

node will be detected as not working. NARUN will be able to detect that n_s is functioning again when a path that goes via n_s will be tried.

We can easily prove that if a path leading to a node exists, the collector node will eventually find it. When the collector network graph is updated (through indirect observations), the collector node will eventually try the path (see the do-while cycle, lines 12-25 of Algorithm 3). When the collector network graph is not updated and all of the paths leading to the node appears to have been broken, the use of the unary graph of lines 22-23 of Algorithm 3 will force the collector to retry all possible paths until one is found working. While this behavior also updates link qualities, it can cause additional communication. As we are going to see in the simulation section; when the network is dense and the error is low, indirect observations always keep the collector network graph updated, and the unary graph is rarely used. This can be proved by showing a low percentage of failing paths. These are paths the collector node believes that lead to the destination node but they do not.

4 Simulation Setup

NARUN has been simulated in the San Paolo district, Camerino City (MC, Italy) where various performance measures have been evaluated. Figure 3 shows the considered smart meter network where the red and white markers represent the collector and meter nodes, respectively. The white lines represent connections between nodes. We use a free space propagation model [37]. The WM-Bus collector node is positioned in order to reach all

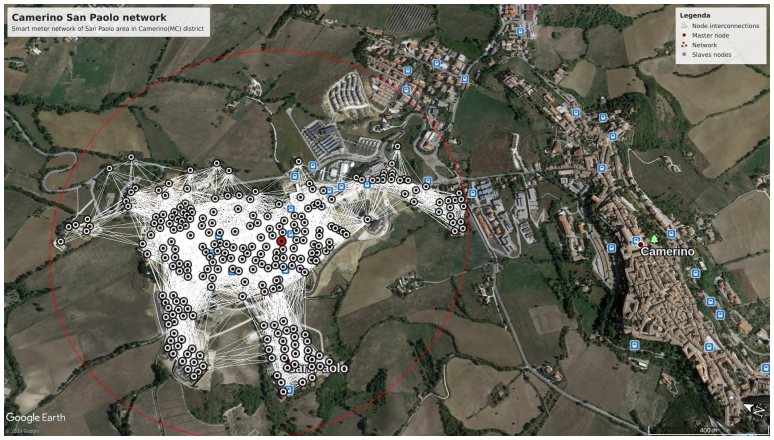

Figure 3: Camerino topology.

meters by using few hops. The characteristics of this topology are summarized in Table 1. The following input parameters are used for the simulations: (i) the noise power measured in dBm; (ii) ECC: a boolean parameter that defines the error correction code used by the simulation (i.e., ECC=true when Hamming is used and ECC=false for CRC); (iii) routing: a boolean parameter that determines whether the meter includes the link failure index lfi in a response packet.

Simulations consider a specific configuration for the physical model. WM-Bus supports frequencies of 169MHz (Narrowband), 433 MHz, and 868 MHz. The frequency of 169 MHz

is a novel frequency that allows distance up to 10 km. The frequency of 433 MHz instead is widely used in Europe [8] and reaches up to a few hundred meters of distance. The frequency of 433 MHz is intended for the sectors where 868MHz is not allowed. Each frequency has different modes that specify the communication baud rate and transmitter capabilities. In our simulation, we chose the widely used frequency of 868 MHz. For channel modeling, we used the Additive white Gaussian noise and the free space model. The transmitter is assumed to use Frequency Shift Keying (FSK) modulation, a transmission power of 10 dBm, and an antenna with 0 dB gain. Equation 3 shows the path loss computation formula, which depends on the frequency f, the distance between the source and the destination d, and the gain of antenna g.

$$Pathloss_{dB} = 20 \times \log_{10}(d) + 20 \times \log_{10}(f) - 27.55 - g \tag{3}$$

With the path loss value, we can compute the signal to noise ratio at the receiving end and the bit error rate as we can see in equation 4. This determines the success of communication.

$$BER = \frac{1}{2} \times erfc\left(\sqrt{\frac{SNR_{rec}}{2}}\right) \tag{4}$$

The probability of a success, a recoverable, or unrecoverable packet by assuming a binary symmetric channel [38] can be calculated as $(1-r)^n$, $n \times r \times (1-r)^{n-1}$ and $1-(1-r)^n - n \times r \times (1-r)^{n-1}$ where n represents the size of the packet, whereas r is the bit error rate.

4.1 Assumptions and simulation methodology

In this section we summarise our simulation methodology and all assumptions. The basic unit of our experiments is a collector reading attempt. This is done by using the ReadMeter procedure of Algorithm 4. We denote with a^i a reading attempt that is related to the sensor i. The attempt a^i is a random variable that is equal to Failure (in the following denoted with 0) when the collector fails to read the sensor i, Success (in the following denoted with 1) otherwise. The DSR attempt may include the discovery of new paths (this is done via broadcasting). These additional messages are sent when no path is known or all available paths failed.

A collector reading operation is a finite sequence of reading attempts $r^i = a^i_1 a^i_2 \dots a^i_k$ ($0 < k < max$) where max is the maximum number of consecutive attempts the collector performs on the same sensor. The details of the readOperation are described in the algorithm of Figure 3. This models that when a collector reading attempt fails, the collector will try again until a read is performed or a max number of attempts has been reached (ten in our simulation). Hence, a reading operation failure is a sequence that contains all zeros. A successfully reading operation is any sequence of zeros (also an empty one) that ends with a one (the length cannot exceed max). Equation 5 formally defines the reading operation failure rate when reading the sensor i. $F(r^i)$ measures the number of failed attempts during the reading operation of the sensor i. As we are going to see, in the following a higher failure rate corresponds to a higher number of network messages. Equation 6 formally defines the successful reading operation rate when reading the sensor i.

$$F(r^i) = \frac{\sum_{k=0}^{max}(1-a^i_k)}{max} \tag{5}$$

$$O(r^i) = \frac{\sum_{k=0}^{max} a^i_k}{max} \tag{6}$$

This can be either zero or one. It is worth mentioning that a 100% reading rate does not imply a 0% failure rate. For instance in the sequence of attempts 0000000001 we have 90% failure rate and 100% reading rate.

We can now formally define a collector $round$. Suppose that our system is composed of z sensors. We denote with $N = \{n_1, \ldots, n_z\}$ the set of sensors. In a round a collector tries to read all sensors in turns by producing a sequence of z reading operations. We use the notation $R = \{r^1, \ldots, r^z\}$ to denote a round where each r^i is the reading operation on the sensor i. Equations 7 and 8 formally define the failure round rate and the reading round rate.

$$F(R) = \frac{\sum_{i=1}^{z} F(r^i)}{z} \quad (7) \qquad O(R) = \frac{\sum_{i=1}^{z} O(r^i)}{z} \quad (8)$$

We can define a simulation run that is a sequence of h rounds. This is denoted with $U = \{R_1, \ldots, R_h\}$. Equations 9 and 10 formally define the failure run rate and the reading run rate.

$$F(U) = \frac{\sum_{i=1}^{h} F(R_i)}{h} \quad (9) \qquad O(U) = \frac{\sum_{i=1}^{h} O(R_i)}{h} \quad (10)$$

An experiment is a sequence of q runs. This is denoted with $E = \{U_1, \ldots, U_q\}$. Equations 11 and 12 formally define the failure experiment rate $F(E)$ and the reading experiment rate $O(E)$.

$$F(E) = \frac{\sum_{i=1}^{q} F(U_i)}{q} \quad (11) \qquad O(E) = \frac{\sum_{i=1}^{q} O(U_i)}{q} \quad (12)$$

We can finally define a simulation that is a sequence of p experiments. This is denoted with $S = \{E_1, \ldots, E_p\}$. Equations 13 and 14 formally define the failure rate $F(S)$ and the reading rate $O(S)$.

$$F(S) = \frac{\sum_{i=1}^{p} F(E_i)}{p} \quad (13) \qquad O(S) = \frac{\sum_{i=1}^{q} O(E_i)}{p} \quad (14)$$

The experiments $S = \{E_1, \ldots, E_p\}$ are a sequence of independent and identically distributed random variables. In order to stop a simulation, we use the criteria that has been presented in [39]. At each run , we consider all the t experiments that have been performed so far (i.e, $S_t = \{E_1, \ldots, E_t\}$) and we calculate $F(S_t)$ and $O(S_t)$. We stop when for a sequence of k consecutive experiments (i.e., E_t, E_{t+1} ... E_{t+k}) the following conditions hold: (i) $|F(E_i) - F(E_{i+1})| < \epsilon$; (ii) $|O(E_i) - O(E_{i+1})| < \epsilon$ (with $t < i <$t+k). This stop criteria makes sure that the sample average converge within a threshold ϵ. We compared the following protocols: (i) WMBUS; (ii) ECC-WMBUS; (iii) NARUN; (iv) ECC-NARUN; (v) DSR. These are briefly described in the following.

WMBUS routing method allows node forwarding by including the path in the request. The weight function is constant-based (see Section 3 for details). The path chosen by the collector minimises the number of hops. Hop to hop communication uses the send described in the Algorithm of Figure 2. This uses CRC to detect erroneous data frame and tries re-

Table 1 Network characteristics

Network characteristic	Value
Collector latitude	$43.146\,509\,423\,809$
Collector longitude	$13.061\,599\,661\,224$
Number of meter nodes	$254\,\mathrm{nodes}$
Node to node minimum distance	$8.86\,\mathrm{m}$
Node to node maximum distance	$249.99\,\mathrm{m}$
Node to node average distance	$151.14\,\mathrm{m}$
Collector to meter minimum path length*	$1\,\mathrm{hop}$
Collector to meter maximum path length*	$4\,\mathrm{hops}$
Collector to meter average path length*	$2\,\mathrm{hops}$
Network radius	$778.10\,\mathrm{m}$
Network density	$134.28\,.\mathrm{nodes}/\mathrm{km}^2$

* Path lengths computed by finding the shortest path with the lowest number of hops.

transmission 4 times. ECC-WMBUS is similar to WMBUS one except for the ECC use. Hop to hop communications use Hamming forward correction code to validate and recover messages. We emphasise that for both WMBUS and ECC-WMBUS routing the weights are never updated but are always set to 1.

NARUN uses a connection-based weight function (see Section 3 for details) while ECC-NARUN uses a Hamming-based one. The algorithms are described in section 3. We have implemented the DSR algorithm as described in [16]. This uses a connection-based weight function.

5 Simulation results

We compare the protocols when some of the links are out of use. More precisely, we consider the case where a subset of the links are disconnected. Each simulation is a sequence of experiments $S = \{E_1, E_2, \ldots, E_k\}$. Each experiment E_i is composed of a sequence of runs (i.e., $E_i = \{U_1, U_2, \ldots, U_q\}$ with $q = 50$). At each run U_i we randomly select a subset of the links and we consider them as disconnected ones (any communication via those links fails). All the remaining links always deliver messages. For each run U_i we perform h rounds $\{R_1, \ldots, R_h\}$ ($h = 50$). When the run U_i is completed, a new one U_{i+1} is performed where the same is repeated (a subset of the edges are randomly picked and disconnected). When moving to a run U_{i+1} from the previous one (i.e., U_i), the protocol is not restarted (i.e., its state is kept). This allows us to test the ability of NARUN to update correctly all links and converge to the new network state. We have performed simulations for 30%, 15% and 5% of disconnected links. We have considered the WMBUS protocol and the NARUN protocol. Table 2 shows the results of our simulations. We can observe that an increase in the percentage of failing links results in a higher failure rate $F(S)$ and lower reading one $O(S)$. WMBUS has always a higher failure rate and lower sensor reading one with respect to NARUN. In fact, NARUN marks failing links and always tries alternative routes (if any). NARUN improves the WMBUS reading rate by 42.67% and 26.86% when 30% and 15% of the links are not usable. NARUN improves the WMBUS reading rate by 9.77% when 5% of the links are not working. We can conclude that NARUN eavesdropping capabilities allow the update of the collector network graph with the right link information. This allows NARUN to find an alternative path and to have a higher reading rate.

Table 2 Failure rate F(S) and reading rate O(S) with disconnected links

failing links	protocol	F(S)	O(S)
30%	WMBUS	90.0373 %	51.66%
	NARUN	29.9265 %	94.35%
15%	WMBUS	77.6022 %	72.93%
	NARUN	2.1864 %	99.79%
5%	WMBUS	48.4440 %	90.22%
	NARUN	0.1245 %	99,99%

We compare the protocols when some of the links are noisy. More precisely, we consider the case where a certain percentage of the links are affected by an increased noise power (i.e., from -80 dBm to -70 dBm). Each simulation is a sequence of experiments $S = \{E_1, E_2, \ldots, E_k\}$. Each experiment E_i is composed of a sequence of runs (i.e., $E_i = \{U_1, U_2, \ldots, U_q\}$ with $q = 50$). At each run U_i we randomly pick a subset of the noisy links. These have a noise power equal to Y ($-70 \leq Y < -80$) while all the rest of the links always deliver messages. For each run U_i we perform h rounds $\{R_1, \ldots, R_h\}$ ($h = 50$). When the run U_i is completed, a new one U_{i+1} is performed where the same is repeated (a subset of the edges are randomly picked which have noise power equal to Y). When moving to a run U_{i+1} from the previous one (i.e., U_i), the protocol is not restarted (i.e., its state its kept). This allows us to test the ability of the protocols (e.g., NARUN and DSR) to learn the new network conditions. We have performed simulations for 30%, 15% and 5% of noisy links with noise power values between -70 and -80 dBm. We have considered the following protocols: (i) WMBUS; (ii) ECC-WMBUS; (iii) NARUN; (iv) ECC-NARUN; (v) DSR.

Figure 4 shows the reading rate (we recall that the total number of sensors is 254) when 30% of links have noise. When the noise power is lower than -73 dBm, all protocols are able to read all the 254 sensors (i.e., 100% of sensor reading rate). When the noise power is -70 dBm the reading performance of the WMBUS drastically decreases to 180 sensors out of 254 (i.e., 71.29%). The use of Hamming code (i.e., ECC-WMBUS) improves the reading rate of WMBUS. ECC-WMBUS reads 210 sensors out of 254 (i.e., 83.1% reading rate). NARUN reads 96% of the sensors while ECC-NARUN and DSR read 99% of the sensors. These protocols perform better since they are able to discover a path with no noisy links unlike the WMBUS which always tries the shortest path. Figure 5 show the failure rate with 30% of noisy links. When the noise power is higher than -75 dBm, all protocols have a reading failure rate close to 0%. When the noise power is between -70 and -74 dBm, WMBUS has a high failure rate since it always tries the shortest path without considering alternative paths with less noise. The introduction of the Hamming code (i.e., ECC-WMBUS) reduces the failure rate by 14%. The DSR and NARUN protocols reduce the failure rate of the WMBUS by 30% and 78%, respectively. In fact, both DSR and NARUN will eventually find the least noisy paths. We recall that NARUN and DSR mark a noisy link with infinity when it fails to deliver a message. The caching strategy of DSR will make paths with less noisy links to become a stable choice while NARUN indirect observations can update noisy links and make them a possible choice. This explains the slightly higher failure rate of NARUN (it retries a shorter path with noisy links). When all paths that lead to a node are noisy, both NARUN and DSR can keep discovering new paths. In this case the flooding discovery strategy of DSR will cause a higher amount of messages when compared with the indirect observations and retrial collector strategy (see Algorithm 3) of NARUN. This behaviour has been validated by checking the total amount of messages received by the sensors in a round (see Figure 6 and Figure 11). DSR generates the highest traffic load

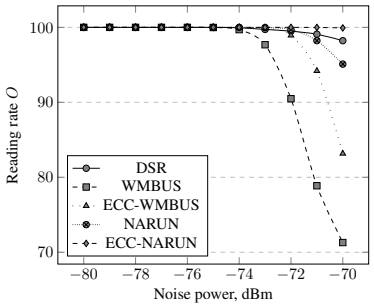

Figure 4: Reading rate O with 30% noisy links

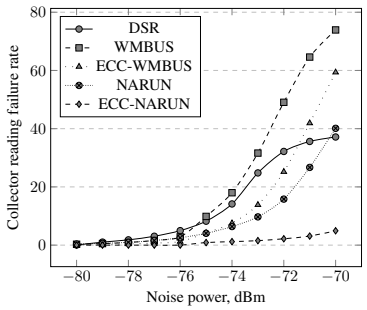

Figure 5: Collector reading failure rate with 30% of noisy links

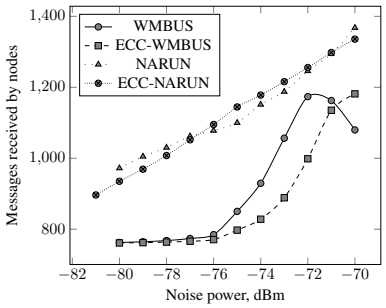

Figure 6: Average messages received by the sensors with 30% of noisy links

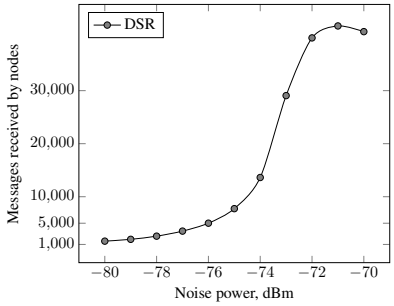

Figure 7: Average messages received by the sensors with 30% of noisy links using DSR protocol

in order to perform various flooding discoveries. Although WMBUS has a very low reading rate, its traffic load is not high. In fact, although it performs several attempts to read the same sensors, no flooding is involved and the shortest path is always used.

The addition of weights that are based on hamming (i.e.,ECC-NARUN) results in the lowest failure rate. ECC-NARUN reduces the failure rate by 70% and 32% when compared to WMBUS and DSR. ECC-NARUN has a low traffic load since it has the least failure rate and no overhead messages are used (i.e., link weights are updated via indirect observations).

Figure 9 and 8 shows the reading rate and the failure when 15% of the links are noisy. Figure 10 and Figure 11 show the traffic load. The trend of the protocols is the same of the 30% noisy links case that we have already discussed.

We can conclude the following: (i) DSR and NARUN are able to read the highest number of sensors since they eventually select paths with less noise ; (ii) WMBUS reads the lowest number of sensors since always selects the shortest path without considering the link noise; (iii) ECC-NARUN has the lowest failure rate and a low traffic load. In fact, indirect link observations allow a quick update of the collector network graph without the need of overhead messages; (iv) DSR has the highest traffic load since various floodings may be needed to converge to the paths with the lowest noise. The use of the Hamming code seems effective for improving the reading rate and reducing the failure one.

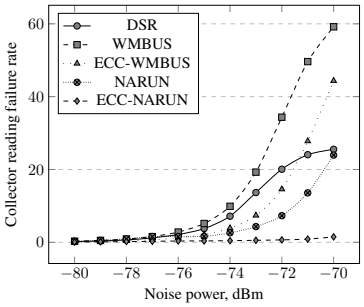

Figure 8: Collector reading failure rate with 15% of noisy links

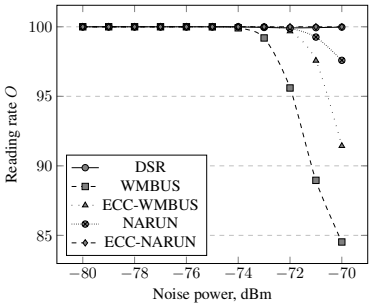

Figure 9: Reading rate O with15% noisy links

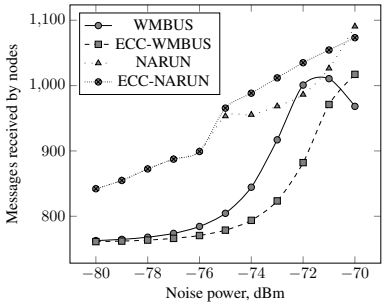

Figure 10: Average messages received by the sensors with 15% of noisy links

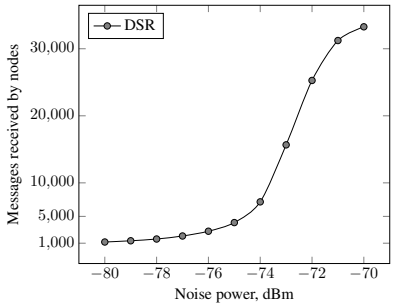

Figure 11: Average messages received by the sensors with 15% of noisy links using DSR protocol

6 Conclusions and Future Work

The WM-Bus is a widely used meter protocol in utility networks. However, in a noisy environment, faults become a serious issue resulting in excess volumes of transmissions. The increased energy consumption due to this activity is also problematic for battery-operated nodes. This work proposes NARUN that aims at reducing reading failure through the use of the Hamming ECC combined with a noise adaptive routing protocol. The employed Hamming ECC reduces re-transmissions by correcting the single-bit errors. The proposed methods are benchmarked in simulations by changing the noise power value between -80 and -70 dBm. Our results show that DSR and NARUN are able to read the highest number of sensors since they eventually select paths with low noise while WMBUS reads the lowest number of sensors since always selects the shortest path without considering the link noise. ECC-NARUN has the lowest failure rate and a low traffic load. In fact, meters eavesdrop on the surrounding environment and efficiently report information on link failure index back to the collector with ordinary reading messages. DSR has the highest traffic load since various floodings may be needed to converge to the paths with the lowest noise. As future work, we plan to include a caching strategy inside ECC-NARUN.

References

[1] European Parliament and Council of European Union. Directive 2009/72/ec of the european parliament and of the council of 13 july 2009 concerning common rules for the internal market in electricity and repealing directive 2003/54/ec. *Official Journal of the European Union*, L 211:55–93, 2009.

[2] Nico Saputro, Kemal Akkaya, and Suleyman Uludag. A survey of routing protocols for smart grid communications. *Computer Networks*, 56(11):2742 – 2771, 2012.

[3] European Committee for Standardization. Communication systems for and remote reading of meters, Part 2: Physical and link layer. Standard EN 13757-2, European Committee for Standardization, 2005.

[4] European Committee for Standardization. Communication systems for and remote reading of meters, Part 3: Dedicated application layer. Standard EN 13757-3, European Committee for Standardization, 2013.

[5] European Committee for Standardization. EN 13757-4 Communication systems for meters and remote reading of meters - Part 4: Wireless meter readout (Radio meter reading for operation in SRD bands). Standard EN 13757-4, European Committee for Standardization, 2013.

[6] European Committee for Standardization. EN 13757-5 Communication systems for meters - Part 5: Wireless M-Bus relaying. Standard EN 13757-5, European Committee for Standardization, 2015.

[7] European Committee for Standardization. EN 13757-7 Communication systems for meters - Part 7: Transport and security services. Standard EN 13757-7, European Committee for Standardization, 2018.

[8] P. Masek, D. Hudec, J. Krejci, A. Ometov, J. Hosek, and K. Samouylov. Communication Capabilities of Wireless M-BUS: Remote Metering Within SmartGrid Infrastructure. In V. M. Vishnevskiy and D. V. Kozyrev, editors, *Distributed Computer and Communication Networks*, pages 31–42. Springer International Publishing, 2018.

[9] Amr Kassab. Latency optimization in smart meter networks. Master's thesis, American University in Cairo, 3 2021.

[10] Zaib Ullah, Fadi Al-Turjman, Leonardo Mostarda, and Roberto Gagliardi. Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, 154:313–323, 2020.

[11] Fadi Al-Turjman, Leonardo Mostarda, Enver Ever, Ahmed Darwish, and Naziha Shekh Khalil. Network experience scheduling and routing approach for big data transmission in the internet of things. *IEEE Access*, 7:14501–14512, 2019.

[12] Fadi Al-Turjman, B. D. Deebak, and Leonardo Mostarda. Energy aware resource allocation in multi-hop multimedia routing via the smart edge device. *IEEE Access*, 7:151203–151214, 2019.

[13] F. Abate, M. Carratù, C. Liguori, and V. Paciello. A low cost smart power meter for iot. *Measurement*, 136:59 – 66, 2019.

[14] R. M. Jacobsen and P. Popovski. Data recovery using side information from the wireless M-Bus protocol. In *IEEE Global Conference on Signal and Information Processing*, pages 511–514, Dec 2013.

[15] R. M. Jacobsen and P. Popovski. Reliable Reception of Wireless Metering Data with Protocol Coding, 2013.

[16] David B Johnson, David A Maltz, Josh Broch, et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, 5(1):139–172, 2001.

[17] Diletta Cacciagrano, Rosario Culmone, Matteo Micheletti, and Leonardo Mostarda. *Energy-Efficient Clustering for Wireless Sensor Devices in Internet of Things*, pages 59–80. Springer International Publishing, Cham, 2019.

[18] Leonardo Mostarda and Alfredo Navarra. Distributed intrusion detection systems for enhancing security in mobile wireless sensor networks. *International Journal of Distributed Sensor Networks*, 4, 04 2008.

[19] S. Spinsante, M. Pizzichini, M. Mencarelli, S. Squartini, and E. Gambi. Evaluation of the Wireless M-Bus standard for future smart water grids. In *2013 9^{th} International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1382–1387, July 2013.

[20] Z. Kuder and R. M. Jacobsen. Feasibility of Wireless M-Bus Protocol Simulation. *Elektrorevue*, 3(3):1–5, 2012.

[21] Z. Kuder. Routing Protocols For Lossy Wireless Networks. Master's thesis, Faculty Of Electrical Engineering And Communication Department Of Radio Electronics, Brno University Of Technology, Czech Republic, 2012.

[22] P. Masek, K. Zeman, Z. Kuder, J. Hosek, S. Andreev, R. Fujdiak, and F. Kropfl. Wireless M-BUS: An Attractive M2M Technology for 5G-Grade Home Automation. In B. Mandler et al., editors, *Internet of Things. IoT Infrastructures*, pages 144–156. Springer International Publishing, 2016.

[23] P. Masek, D. Hudec, J. Krejci, A. Ometov, J. Hosek, and K. Samouylov. Communication Capabilities of Wireless M-BUS: Remote Metering Within SmartGrid Infrastructure. In Vishnevskiy, V. M. and Kozyrev, D. V., editor, *Distributed Computer and Communication Networks*, pages 31–42. Springer International Publishing, 2018.

[24] K.-I. Hwang and S.-W. Nam. Bitmap-wise wireless M-Bus coordination for sustainable real time energy management. *Sustainability*, 6(7):4326–4338, 2014.

[25] Government Arts College, Calduwel Pitchai, Nismon Robert, and Bishop Heber College. PSA-HD: Path Selection Algorithm based on Hamming Distance to Enhance the Link Stability in Mobile Ad-hoc Networks. *International Journal of Intelligent Engineering and Systems*, 11(1):259–266, February 2018.

[26] Zehua Wang, Cheng Li, and Yuanzhu Chen. Psr: Proactive source routing in mobile ad hoc networks. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–6, 2011.

[27] M Deva Priya and P Priyanka. Ppclss: probabilistic prediction coefficient link stability scheme based routing in manets. *Int J Comput Sci Eng Technol*, 6(4):246–256, 2015.

[28] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Rfc3561: Ad hoc on-demand distance vector (aodv) routing, 2003.

[29] Chenxi Liu, Yong Li, Wei Cheng, and Ge Shi. An improved multi-channel aodv routing protocol based on dijkstra algorithm. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 547–551, June 2019.

[30] Sagnik Dutta, Rituparna Chaki, and Nabendu Chaki. Optimal reactive routing protocol (orrp): A new reactive routing protocol for the shortest path in ad hoc networks. In *2006 Annual IEEE India Conference*, pages 1–4, 2006.

[31] Soma Saha and Tamojay Deb. Study and improvement on optimal reactive routing protocol (orrp) for mobile ad-hoc networks. In *International Journal of Computer Science & Emerging Technologies (IJCSET)*, volume 1, pages 134–138, 2010.

[32] P. Digeser, M. Tubolino, M. Klemm, and A. Sikora. Management of Routed Wireless M-Bus Networks for Sparsely Populated Large-Scale Smart-Metering Installations. In Thampi, S. M. and Zomaya, A. Y. and Strufe, T. and Alcaraz Calero, J. M. and Thomas, T., editor, *Recent Trends in Computer Networks and Distributed Systems Security*, pages 385–395. Springer Berlin Heidelberg, 2012.

[33] A. Sikora, R. Werner, and J. O. Grafmüller. Design and implementation of an energy aware routing extension for energy autarkic Wireless M-Bus networks. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1446–1451, 2013.

[34] A. Sikora, R. Werner, P. Lehmann, and J. O. Grafmüller. Design, Implementation, and Verification of an Energy Autarkic, RF-based Water Meter with Energy Aware Routing. In *Energy self-sufficient Sensors; 7^{th} GMM-Workshop*, pages 1–5, Feb 2014.

[35] Tung-Linh Pham and Dong-Seong Kim. Lossy link-aware routing algorithm for isa100.11a wireless networks. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pages 624–629, July 2013.

[36] Y.-M. Xie, D.-G. Zhang, X.-D. Song, and X. Wang. An EAODV routing approach based on DARED and integrated metric. *Wireless Networks*, 20(8):2455–2467, 2014.

[37] Krishna Doddapaneni, Enver Ever, Orhan Gemikonakli, Ivano Malavolta, Leonardo Mostarda, and Henry Muccini. Path loss effect on energy consumption in a WSN. In *2012 UKSim 14th International Conference on Computer Modelling and Simulation*. IEEE, March 2012.

[38] Y. Bazlov. Coding Theory, Part2: Hamming Distance - 2010. Lecture from The University of Manchester, School of Mathematics, Manchester, 2010.

[39] Adam Brentnall. Discrete-event system simulation (international edition). *Journal of Simulation*, 1(3):223–223, August 2007.