Smart Eagle - a Bird's-eye View on Heterogeneous Networks for Commercial and Industrial Sites

Bram Scheidegger[1], Yvonne-Anne Pignolet[2], Ettore Ferranti[2]
[1]*ETH Zurich,* [2]*ABB Corporate Research, Baden, Switzerland*
scheidegger@bram.ch, {*yvonne-anne.pignolet, ettore.ferranti*}*@ch.abb.com*

***Abstract*—IT infrastructures in commercial and industrial sites are recently undergoing an evolution, which requires them to integrate large numbers of heterogeneous devices, often using multi-protocol networks. One of the driver of this evolution is represented by a recent trend in the energy sector, where local sites are encouraged to build their own microgrids. Furthermore, the lowering costs of sites automation and monitoring systems are making their adoption by industrial and commercial sites much easier to justify.**

In order to make the most out of these innovations, the communication between all the players of these systems is vital, and a constant monitoring of it essential. However, despite the importance of the data network, there exist currently no vendor-independent analysis and management tools tailored for such networks, usually comprised of various, sometimes very different, network types.

Therefore, we present Smart Eagle, a distributed network analysis and monitoring tool capable of dealing with building automation and low-rate wireless personal area networks. We introduce the architecture of Smart Eagle, and report on our experience in taking measurements on an existing site management communication network. In particular, we were able to gather meaningful measurements from both IEEE 802.15.4 and KNX networks, without intervening on existing devices or on the underlying network infrastructure.

I. Introduction

Recently, the combination of small scale microgrids and sites automation and monitoring systems have received a lot of attention [12]. One main challenge present in such systems is the granularity of control and the amount of data to process: in fact, they usually need a fine-grained control over various components within a building (e.g. heating, lights, etc.) and are constantly receiving large amounts of data from all the sensors on the site.

Within such a system, the communication infrastructure is essential to enable the interaction between sensors, actuators, security controls and monitoring systems. For normal operation, the communication network is mainly used to control the heating system, lights, power levels, etc. However, in case of a power failure, the data network transports essential information to stabilize the system and keep vital systems operational. As the data network is the backbone of all these operations, it is crucial to keep it operational at all times. This requires a system which assists in the deployment, operation and maintenance of the communication infrastructure.

To this end, we introduce Smart Eagle, a communication network measurement system for large sites (source code available upon request). It has been designed to help site operators to monitor their infrastructure, i.e., to detect issues and react on them appropriately.

From a communication network point of view, a site management system is a large, heterogeneous network consisting of up to thousand different devices typically using more than one communication technology. The devices participating in communication can be grouped into two main categories: energy-related devices (e.g., power generation and storage units) and devices for building automation (e.g., end-user devices allowing fine-grained control, like electricity meters or power switches). As these devices are delivered by various manufacturers, it is likely that custom software deployment is limited or unavailable. Therefore, we choose a least invasive approach and perform our measurements by adding measurement devices only (i.e., no changes to existing devices or their communication equipment are necessary).

Smart Eagle adheres to a modular approach and can easily be extended to offer more functionalities. It offers a user interface combining all relevant measurements of the different devices. While most existing work on network measurement addresses traditional Ethernet and IP-based networks, we explicitly designed Smart Eagle to cope with a heterogeneous network. In its current implementation, Smart Eagle integrates IEEE 802.15.4 and KNX networks and provides the functionality summarized in Table I.

Apart from the general architecture, we focus on the components necessary to analyze the KNX home automation bus and its specific properties, since to the best of our knowledge few research has been done on it. We discuss several round trip time measurement techniques, topology discovery and sniffing. In addition we briefly mention how Smart

Functionality	IEEE 802.15.4	KNX
Link layer RTT measurement	✓	✓
Application layer RTT measurement	×	✓
Subnet sweep	✓	✓
Topology discovery	×	✓
Channel scan	✓	×
Sniffing	✓	✓
Monitoring	✓	✓
Link quality	✓	×

Table I
Smart Eagle functionality overview. RTT = round trip time.
✓: implemented, ×: not applicable.

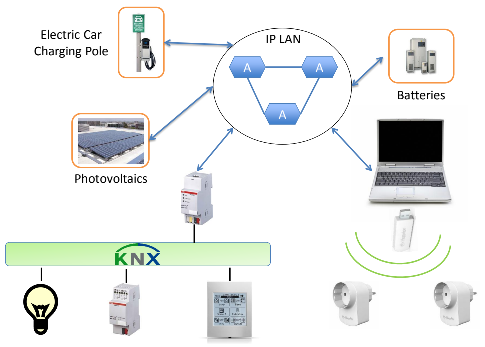

Figure 1. High level network overview. All smart grid demo lab devices are interconnected by an IP network. KNX is connected to the LAN using a KNXnet/IP gateway whereas the Plugwise power plugs communicate with the LAN by an intermediate computer and their associated USB dongle. The site management system is represented by the agent platform (A).

Eagle deals with wireless networks built on top of IEEE 802.15.4 and how to extend it for other technologies. In contrast to control platforms, we are only concerned with the communication network protocols and their performance, but not the semantics of the commands. Thus Smart Eagle is also applicable in areas other than site management.

To evaluate our work we use Smart Eagle to explore the communication network of a site management system (called Smart Grid Demo Lab, SGDL) deployed at the Swiss ABB Corporate Research Center. The global aim of the SGDL is to build a software infrastructure integrating both ABB Smart Grid solutions and third-party products, to provide a common platform for Smart Grid R&D activities. A multi-agent system integrating different appliances such as smart meters, solar panels, batteries, etc. interconnected with a multitude of different network technologies like Ethernet, KNX or Zigbee has been put in place (Figure 1). The site management system is represented by multiple interconnected agents and their attached appliances.

All smart grid devices are indirectly connected to a LAN, some of them via a smart meter (Figure 1). This IP network is a standard 100 Mbit/s Ethernet LAN. The KNX bus is connected to the IP network using a KNXnet/IP gateway. The right side of Figure 1 shows two special power plugs named *Plugwise Circle*[1]. These power plugs have integrated monitoring functionality and accept remote commands to turn the attached devices on and off. They communicate with a manufacturer-specific USB dongle using Zigbee. The communication is encrypted and the protocol is proprietary. The USB dongle is attached to a PC which serves as relay between the IP network and the Plugwise.

II. Related Work

There is a large body of work on network measurements and tools to perform them. Most of them focus on Ethernet and IP-based networks and consider methods for, e.g., topology discovery [4], bandwidth estimation [14], link capacities, packet round-trip times, jitter and packet loss on residential networks [6], sampling for passive measurements [7] to name but a few. For internet measurements the cooperative association for internet data anaylsis CAIDA[2] is a good starting point. These efforts have led to a wide variety of available tools for such networks. Thus we focus on additional communication technologies in this paper.

The vendors of microgrid and building automation devices often integrate some measurement functionality into their products as they are in full control over the software on the devices. However, often only vague information concerning network measurements is available. As a consequence, there exists no framework to combine these tools, to the best of our knowledge. For KNX networks, most measurement results concern congestion. E.g., Cavalieri presents measurements demonstrating that KNXnet/IP gateways form bottlenecks under ”traffic conditions not particularly critical” and telegrams get dropped [5]. Tools for KNX include ETS[3], the official configuration and deployment tool which supports sniffing and device discovery, ABB i-bus tool [3] to present end device functionality to the user and a Wireshark plugin [9]. None of these tools offer network measurement capabilities.

Performance analysis of wireless sensor networks has received more attention both in simulations, e.g., [15], and measurements on deployed networks, e.g., [11]. This paper also briefly surveys active and passive measurement techniques focussing on different technologies and aspects in recent work. To analyze low-power wireless networks like IEEE 802.15.4, a multi-sniffer system can be beneficial due to the limited radio range of nodes [17].

III. System Overview

The communication infrastructure is comprised of a multitude of different network technologies depending on the manufacturer and the environment. Before introducing our system architecture, we present an overview over the network assumptions to motivate our design decision. Afterwards, the network specific implementation of Smart Eagle is discussed in more detail.

A. Network Assumptions

As discussed in the previous section, implementing measurement capabilities into devices is a well-established topic. However, this approach depends on specific functionalities that vendors built in their products, and on the openness of the network stack. Hence it is restrictive and does not offer flexibility and future-proofness. We therefore assume that end nodes do neither implement any measurement functionality nor can they be modified (*uncooperative nodes*). However, to perform our measurements, we assume that we can

[1]http://www.plugwise.com

[2]www.caida.org

[3]http://www.knx.org/knx-tools/ets4/

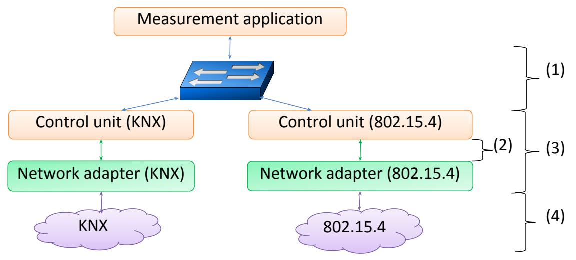

Figure 2. Smart Eagle architecture overview. (1) intermediate network, (2) interconnect link, (3) probe and (4) network under test (NUT).

use additional devices with *direct network access* in contrast to using a manufacturer-specific, high-level protocol. I.e., we assume that we can observe the messages exchanged instead of simply reading an aggregated value of multiple sensors by a high-level protocol command. Although we have direct access to the *network under test (NUT)*, we do not expect that we have the same network access capabilities as the devices of the network. For example, we may not know the encryption keys or the exact routing algorithm. The common denominator of a set of devices is the fact that they follow the specification of the underlying network. Thus, our measurements are derived from the analysis of standard compliant messages sent by additional devices, and of the responses sent by the devices in the NUT. This *external* approach ensures that our system is vendor-independent. A *distributed* approach is required as the various communication networks are spread over one or several buildings making it impossible to gather all measurements with a single machine.

B. Architecture

The cornerstone of our design is depicted in Figure 2. It comprises three building blocks: measurement application, control unit, and network adapter.

Measurement Application: controls the entire system, stores all the measurements gathered from the probes and offers a GUI for user interaction. As the measurement application has access to the entire dataset, it is not intended to run on embedded systems (in contrast to the probe).

Control unit: is a relay for format conversion and some basic preprocessing for a given network type. It is connected to the network adapter by a link we refer to as *interconnect link* which depends on the NUT (e.g., USB, serial, Ethernet). The protocol can either be predetermined by the manufacturer (in case of KNX) or chosen by the control unit programmer (in our example IEEE 802.15.4) if the network adapter is freely programmable.

Network adapter: provides access to the NUT. Depending on the type of network, this component may be a programmable piece of hardware or just an interface providing access to the network.

Subsequently, the bundle of network adapter and control unit is called a *probe*. A probe is network-specific and exchanges data with the measurement application using a (fast) intermediate network (in our case Ethernet). In the next sections we describe the communication over the intermediate network, followed by the description of the KNX probe and the 802.15.4 probe.

C. Intermediate Network Communication

The communication between the measurement application and the control units follows a client-server approach: the measurement application makes a request to the control unit and the control unit responds. This message exchange takes place over an IP network (intermediate network). We chose a polling-based communication over the HTTP protocol using the XML data format because that allows us to use open standards and existing components, e.g., the light-weight webserver Jetty for the control units.

On top of the webserver, we implemented a framework treating measurements as transactions. They are implemented by using HTTP status codes, for example "200 OK" is the response for a successful request whereas "500 Internal Error" is sent in case of an abort. We motivate this design choice in the next paragraph and briefly explain our implementation.

D. Transaction Framework

Fault handling is of major concern as network measurements in heterogeneous environments can fail in various ways, e.g., failure of intermediate network, sudden termination of network connection, etc. To simplify reasoning about measurement requests, all queries to the control units are treated as transactions: either the request completes successfully or it fails. The measurement application is provided with a clear semantic and does not have to worry about the state of the control unit. To implement transactions on the control units, our framework offers various locking schemes controlling the interaction between the measurement application and the control unit. It is easier to control concurrency before starting the actual measurement as this way no low-level concurrency handling (e.g., mapping responses from concurrently running measurements to their respective requensts) has to be implemented.

With regard to ACID properties, our focus is on atomicity and to some extent consistency as we want to avoid that the control unit remains in an invalid state due to some half-completed measurement. Isolation comes naturally with the one thread per request schema of the webserver and durability is not relevant because the control units do not store data.

IV. KNX Probe

The KNX probe consists of a KNXnet/IP gateway attached to the KNX bus and a PC running the control

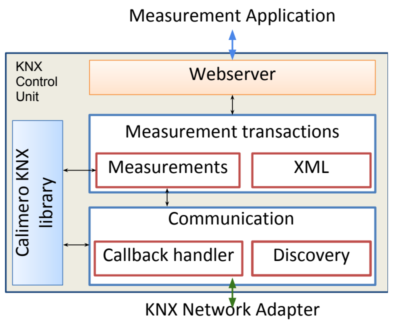

Figure 3. KNX control unit architecture. Red boxes indicate functionality used by the surrounding component, the webserver runs in a separate thread.

unit software programmed in Java using the library Calimero [13]. The KNXnet/IP gateway is a standardized, off-the-shelf KNX component used to connect the KNX bus to an IP network. The KNX control unit serves as a broker between the KNXnet/IP gateway and the measurement application. In the remainder of this section, we describe its architecture and measurements implementation.

A. Architecture

The KNX control unit consists of four basic building blocks (Figure 3) which are briefly reviewed in a top-down manner before discussing gateway discovery.

After a HTTP get request, the webserver invokes a callback which executes the corresponding measurement transaction. On completion, the measurement results are packed into predetermined data structures, converted to XML and sent back. The communication layer is responsible for KNXnet/IP gateway discovery, traffic sniffing and measurement execution. To communicate with the KNX bus and to implement the measurements, we use the Calimero library. It maintains the connection with the KNXnet/IP gateway and offers a set of network, transport and application layer functionalities. To spare the user the effort of passing the IP address of the KNXnet/IP gateway to the control unit upon deployment, the gateways are located automatically. The discovery procedure is part of the KNXnet/IP specification [1] and implemented in Calimero. The discovery process of Calimero returns a set of gateway IP addresses. However, the control unit must address the following issues: 1) KNXnet/IP gateway failover, in case some gateways fail or are occupied; 2) Multiple network interfaces as Calimero needs the IP address of the PC network interface attached to the network containing the gateway. Our approach to resolve these problems uses the gateway discovery functionality to find all KNXnet/IP gateways, no matter to which network interface they are attached. Next, ping is used to determine if the gateway is alive. By setting the source IP parameter in ping, we control the control unit source address to use. Finally, the control unit tries to connect to each gateway.

B. Measurement implementation

Sniffing: After successfully establishing a connection to the KNXnet/IP gateway, we provide a callback implementing the NetworkLinkListener interface. It forces us to implement a function for indication and confirmation frames. Indication frames are received data frames and confirmation frames are link layer ACKs. KNXnet/IP gateways support two monitoring modes: bus monitoring and group monitoring. In bus monitoring mode, all link layer traffic on the bus is sniffed (including ACK) whereas in group monitoring mode only group communication is forwarded to the IP client. Our KNXnet/IP gateway supports group monitor mode only and hence we cannot sniff link layer ACKs not intended for us.

Topology discovery: The KNX topology consists of a three level hierarchy: backbone line, main line and subnet. Each level can contain network elements such as repeaters, bridges and routers as well as end devices. The one-to-one correspondence between the physical hierarchy (wiring) and the network address of each device is specified by the KNX protocol. Routers are the only network elements interconnecting different hierarchical levels and they have dedicated addresses in the addressing schema allowing us to detect them through device discovery. Each hierarchical level can contain end devices, which are again discovered through device discovery and placed in the correct position using their device address.

RTT measurement: Implementing RTT measurements requires understanding the exact behavior of KNX on the link layer, network layer and partially the application layer. Network elements differ in their link layer ACK behavior. Bridges and repeaters do not support selective ACK, i.e., they acknowledge all frames, even if the corresponding end device does not exist. In contrast, routers have a filter table and only send a link layer ACK if a destination device exists in its configuration. If the filter table is deactivated, e.g., for debugging, the router sends a link layer ACK for all frames.

On the transport layer, KNX defines connectionless and connection-oriented communication. A connectionless RTT measurement is based on a device descriptor read operation (specified in the KNX application layer [1], Section 3/3/7). This is the closest to a network layer RTT measurement we can get. However, the implementation of connectionless communication is not mandatory [2]. Connection-oriented communication is the official procedure for KNX device discovery ([1], Section 3/5/2). It tries to establish a connection to the remote device. If the remote device exists, it responds with a disconnect, otherwise, no response is received. We slightly modified this procedure and perform a device descriptor read operation after the connection has been established. This ensures that the detected device is indeed capable of interaction using this connection and it makes our measurement comparable to the connectionless case as the request and payload is the same.

Device Discovery: is implemented on top of the RTT measurements by scanning through the address ranges. However, it is not required to scan all possible addresses as we can exclude entire lines and subnetworks if the corresponding router is not available.

V. IEEE 802.15.4 Probe

IEEE 802.15.4 is a low-rate wireless network technology enabling low-power and low-cost communication [10]. The specification is publicly available and defines the physical layer and the link layer. ZigBee and 6LoWPAN are popular personal area network technologies defined on top of IEEE 802.15.4. Although analyzing higher layers may reveal more information about the network (e.g., routing information), we focus on IEEE 802.15.4 for two main reasons: first, it is a more general approach allowing us to deal with a wider range of networks. The other problem for higher layer network analysis is the link layer encryption offered by IEEE 802.15.4. When it is enabled, the payload of the IEEE 802.15.4 frame is encrypted and hence we cannot access higher-layer protocol information. However, on link layer we still have the cleartext IEEE 802.15.4 header information (which is only protected by a checksum) revealing valuable information. In the next paragraph, we provide a broad overview over the architecture of the IEEE 802.15.4 probe and afterwards we briefly present the concepts of the measurement functionality.

A. Architecture

The 802.15.4 probe consists of a freely programmable network adapter which is attached to a PC running Linux and hosting the control unit as depicted in Figure 4. The network adapter is connected to the PC through USB but registers as a serial console. The control unit is a Java application providing an interface between the network adapter and the measurement application. In addition, it deploys and launches the Contiki OS on the network adapter.

B. Measurement implementation

To avoid inaccuracies due to the USB connection between network adapter and control unit, the measurements are directly implemented into the Contiki network stack [8].

Sniffing: The sniffer is implemented by putting the radio chip into promiscuous mode and fetching the header information upon receival of a frame. However, putting the radio chip into promiscuous mode has a disadvantage for RTT measurement as it disables hardware ACK handling.

RTT Measurement: RTT measurements are performed by sending a data frame with a one byte payload (without side effects) to the target device with the ACK request flag set. According to the specifications, the target device responds with an ACK. As the mapping between data frame and its corresponding ACK is performed by 8-bit sequence numbers, timing is important: if we send a data packet and wait too long for a response, other hosts may send ACKs not intended for us with the same sequence number (collision). In our case, hardware ACK handling is disabled (promiscuous mode) and we have to do the mapping in software losing the accuracy provided by the hardware. One way to reduce the probability of a collision is performing a second RTT measurement for each discovered device.

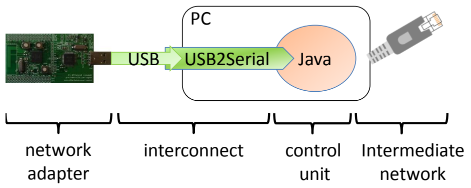

Figure 4. 802.15.4 probe: the network adapter is connected through USB (appears as serial console) to the control unit which is connected by Ethernet to the measurement application.

VI. System Evaluation

This section presents the results obtained from running the system on a real testbed. Unfortunately, we could not compare our measurement accuracy against vendor built-in solutions as this information is not publicly available.

A. System setup

The following equipment was used to run Smart Eagle: Lenovo-PC: dual core Intel Pentium G6950 (2.80 GHz) with 3 GB of RAM. It is equipped with a dual boot system, Ubuntu Linux 11.10 i386 with Java openjdk-i386 1.7.0 (used to evaluate Smart Eagle) and Windows XP with ETS 3.0f for KNX configuration. FitPC: dual core Intel Atom Z530 (1.60 GHz) with 1 GB of RAM. It runs Ubuntu Linux 12.10 i676 with the Java openjdk-i386 1.7.0_09 (used to test 802.15.4) For networking, we used a Netgear EN-104-TP 10-Mbit Ethernet hub to connect the two machines to each other and to the main ABB Smart Grid Demo Lab installation.

Note that there is no overhead for sniffing on KNX or 802.15.4 networks because KNX is a bus and 802.15.4 sends the data to all devices in range anyway. The overhead of probing traffic depends on the measurement intensity (configurable).

Due to the lack of space, we do not report our evaluation of the system while monitoring 802.15.4 communication. We focus instead on measurements on our KNX network consisting of 207 devices in total (see [16] for more details).

B. KNX

1) Topology discovery: Smart Eagle offers two types of topology discovery (i) a full scan scanning all possible network device addresses and (ii) smart scan scanning only subnetworks with a parent.

Both scans discovered one area coupler having a line containing four line couplers. As expected, the smart scan (5:08 min $\pm$ 0 sec) was much faster than the full scan (50:40 min $\pm$ 1 sec). We repeated each measurement three times.

2) Discovery accuracy: To see how reliably Smart Eagle detects KNX devices, we evaluated the subnet device discovery feature on each of the four subnetworks. Device discovery was performed in connection-oriented mode with

Subnet	ETS DB	SE miss	SE extra	SE Total
1.0.0	11	1	0	10
1.1.0	59	0	0	59
1.2.0	43	0	3	46
1.3.0	43	1	0	42
1.4.0	50	0	0	50

Table II
SMART EAGLE DISCOVERY ACCURACY COMPARED TO THE ETS PROJECT DATABASE. THE NUMBERS IN THE TABLE REPRESENT THE NUMBER OF DEVICES. LEGEND: DATABASE (DB), SMART EAGLE (SE), MISSING NODES (MISS), ADDITIONAL NODES (EXTRA)

Number of threads	1	2	3	4
Time (min:s)	39:59	20:07	13:29	10:14
Avg number of devices found	59.0 (100%)	58.75 (99.6%)	58.3 (98.2%)	55.25 (93.6%)

Table III
SPEED AND ACCURACY OF SMART EAGLE FOR THE 1.1.0 SUBNET.

one thread only to maximize the discovery precision. We repeated each measurement three times. As baseline, we compared the results of Smart Eagle against the ETS project database and found several discrepancies (Table II).

3) Performance & threading: Next, we investigate Smart Eagle's discovery speed and accuracy (Table III). We varied the number of threads (1 - 4) to observe the influence of concurrency. The two results of this experiment are: 1) Doubling the number of threads cuts the discovery speed by half; and 2) when increasing the number of threads, discovery accuracy decreases.

4) Link layer analysis: As expected due to the KNX link layer ACK semantics (Section IV), a link layer sweep is not usable for device discovery. When running a topology discovery, all network elements except the ones in the 1.0.0 line are marked as existing. In the 1.0.0 line, only the actually existing network elements (line couplers) are found. The difference between the 1.0.0 line and the other lines is that our KNXnet/IP gateway is member of the 1.0.0 line. Hence, the probing traffic does not pass the 1.0.0 area coupler for device discovery inside the 1.0.0 line. We checked the 1.0.0 area coupler device configuration and found that it is configured to ACK all frames (filtering is disabled) which is not an optimal configuration for a production environment. Furthermore, we found that filtering in the line couplers of the 1.0.0 line is disabled as well. This corresponds to our measurements, showing all addresses in the 1.1.0 to 1.4.0 subnet as existing. We conclude that link layer sweeps are not suitable for device discovery but depending on their position in the KNX bus, we can learn something about the configuration of the network elements. Link layer analysis can be used as building block to find non-optimal KNX network configurations (e.g., filtering disabled, errors in the filter table).

VII. CONCLUSION

Reliable power supply is essential in critical areas like data centers, industrial sites, hospitals, etc. An interruption of communication in a Smart Grid may impact failover and load balancing and hence ensuring Smart Grid data network operability at all times is essential. Towards this end, we implemented a prototype demonstrating how to build a monitoring and measurement system for heterogeneous communication networks. We showed that our external, distributed network measurement approach is feasible and we evaluated it on a existing communication infrastructure for KNX and IEEE 802.15.4. During the analysis of our KNX probe, we discovered that: the ETS project database is outdated and that filtering for certain network elements is disabled. The experience we gained on the test networks confirmed that our model and architecture fit site management systems requirements well, laying the cornerstone to add support for more network types in the future.

REFERENCES

[1] Knx standard, system specs - vol 3: Architecture, 2009.

[2] Knx standard, system specs - vol 6: Profiles, 2009.

[3] Abb i-bus tool: A professional service tool for knx system integrators. http://www.abb.com/cawp/seitp202/eec9ea9d970bf95dc125799f003bf7f4.aspx, 2012.

[4] K. Ahmat. Ethernet topology discovery: A survey. *CoRR*, abs/0907.3095, 2009.

[5] S. Cavalieri. Analysing congestion in knxnet/ip communication system. In *IEEE Conf. on Industrial Technology*, 2011.

[6] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. Internet Measurement Conference*, 2007.

[7] N. Duffield. Sampling for passive internet measurement: A review. *Statistical Science*, 19(3):472–498, 2004.

[8] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proc. IEEE Conference on Local Computer Networks*, 2004.

[9] D. L. Harald Weillechner. Knxnet/ip wireshark dissector. http://knxnetipdissect.sourceforge.net/index.html.

[10] A. Koubaa, M. Alves, and E. Tovar. Ieee 802.15.4 for wireless sensor networks: A technical overview. Technical report, Polytechnic Institute of Porto (ISEP-IPP), July 2005.

[11] G. Kunzel, J. M. Winter, I. Muller, C. E. Pereira, and J. C. Netto. Passive monitoring software tool for evaluation of deployed wirelesshart networks. In *Brazilian IEEE Symposium on Computing System Engineering (SBESC)*, 2012.

[12] R. H. Lasseter and P. Paigi. Microgrid: A conceptual solution. In *IEEE Power Electronics Specialists Conf.*, 2004.

[13] B. Malinowsky, G. Neugschwandtner, and W. Kastner. Calimero: Next Generation. In *KNX Scientific Conference*, 2007.

[14] R. S. Prasad, M. Murray, C. Dovrolis, K. Claffy, R. Prasad, and C. D. Georgia. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17:27–35, 2003.

[15] A. V. S. L. Ullo and G. Velotto. Performance analysis of ieee 802.15.4 based sensor networks for smart grids communications. *Journal of Electrical Engineering: Theory and Application*, 2010.

[16] B. Scheidegger. Smart eagle: Advanced external monitoring of heterogeneous networks. Master's thesis, Swiss Federal Institute of Technology (ETH) Zurich, 2013.

[17] Y. Yang, P. Xia, L. Huang, Q. Zhou, Y. Xu, and X. Li. Snamp: A multi-sniffer and multi-view visualization platform for wireless sensor networks. In *IEEE Industrial Electronics and Applications Conference*, 2006.