The Industrial Internet of Things as an enabler for a Circular Economy
Hy-LP: A novel IIoT Protocol, evaluated on a Wind Park’s SDN/NFV-enabled 5G Industrial Network

George Hatzivasilis\textsuperscript{a,b,*}, Konstantinos Fysarakis\textsuperscript{a}, Othonas Soultatos\textsuperscript{a}, Ioannis Askoxylakis\textsuperscript{a}, Ioannis Papaefstathiou\textsuperscript{b}, Giorgos Demetriou\textsuperscript{c}

\textsuperscript{a}ICS-FORTH, Vassilika Vouton, 70013, Heraklion, Crete, Greece
\textsuperscript{b}Department of Electrical & Computer Engineering, Technical University of Crete, Akrotiri Campus, 73100, Chania, Crete, Greece
\textsuperscript{c}Ecole des Ponts Business School, 77 rue de Miromesnil, 75008, Paris, France

Abstract

Smart interconnected devices, including Cyber-Physical Systems (CPS), permeate our lives and are now an integral part of our daily activities, paving the way towards the Internet of Things (IoT). In the industrial domain, these devices interact with their surroundings and system operators, while often also integrating industrial cloud applications. This 4\textsuperscript{th} Industrial Revolution guides new initiatives, like the introduction of 5\textsuperscript{th} Generation Mobile Networks (5G), to implement flexible, efficient, QoS- and energy-aware solutions that are capable of serving numerous heterogeneous devices, bringing closer the vision of a sustainable, Circular Economy. However, the lack of interoperable solutions that will accommodate the integration, use and management of the plethora of devices and the associated services, hinders the establishment of smart industrial environments across the various vertical domains. Motivated by the above, this paper proposes the Hy-LP - a novel hybrid protocol and development framework for Industrial IoT (IIoT) systems. Hy-LP enables the seamless communication of IIoT sensors and actuators, within and across domains, also facilitating the integration of the Industrial Cloud. The proposed solution is compared with existing standardised solutions on a common application, working around the protocols’ intrinsic characteristics and features to produce each variant. The developed systems are evaluated on a common testbed, demonstrating that the proposed solution is around 10 times faster for the same CPU usage level, while consuming 7 times less memory. Moreover, the applicability of the proposed solutions is validated in the context of a real industrial setting, analyzing the network characteristics and performance requirements of an actual, operating wind park, as a representative use case of industrial networks.

\textsuperscript{*}George Hatzivasilis
Email address: hatzivas@ics.forth.gr (George Hatzivasilis)

Preprint submitted to Journal of Computer and Communications networks January 14, 2018
1. Introduction

By 2020, the Internet of Things (IoT) will have an economic impact of more than 14 trillion\(^1\) with around 34 billion devices being deployed\(^2\) in a variety of domains (smart grid, city infrastructure, transportation, residential/home automation, industrial systems, military, and healthcare, among others \[1, 2, 3, 4\]). The convergence of the digital with the physical worlds and the establishment of Cyber-Physical Systems (CPS) unleashed an entirely new potential for the European industry that can be utilized in the effort of becoming a resilient, competitive and resource efficient economy.

Research and industry struggle to integrate this evolving technology and the exponential growth of IoT. Existing networking \[3\] and security mechanisms \[5\] are adapted to handle the vast population of IoT devices. Moreover, as high volumes of data are produced and processed, the integration of IoT solutions with cloud computing is becoming imperative \[6, 7, 8, 9, 10\]. Seamless and higher level machine to machine (M2M) and human-to-machine (H2M) interactions, are another important requirement in order to effectively monitor and manage the infrastructure, allowing the use of its full potential. The upcoming 5G technology tries to tackle these issues. The main design goals towards this new shift in technology include among others large network coverage with high data speed, energy efficient installations, and increased inter-organization communication between the machines \[11, 12, 13, 14\].

Typically, end-users do not possess the skills to configure and setup the devices that may be found in smart environments; in large-scale deployments, individually setting up devices is not even feasible. From the perspective of implementers, there is a need for rapid development and deployment, while simultaneously tackling issues of scaling and inherent limitations in terms of resources (CPU, memory, power etc.). However, at its current state, the ubiquitous computing landscape is segregated, consisting of numerous proprietary solutions, which are typically incompatible with each other. This makes setting up, managing and securing a smart device ecosystem, significantly challenging. Various IoT protocols proposed by researchers aim to address these issues, while standardization initiatives try to guarantee interoperability, through the wide and structured deployment of the proposed mechanisms.

The interplay of IoT with other domains such as the Circular Economy provides a fertile ground for innovation and value creation. Circular economy value

---


drivers include extending the useful life of finite resources and maximising the utilisation of assets, creating an emerging class of *looping assets*, and regenerating natural capital for more effective and efficient use. IoT value drivers enable for a new window of opportunity to be opened on the economic cycle as it becomes a serving enabler by collating knowledge about asset locations, conditions, quality and performance in real time and over time. IoT has been catalytic already in presenting solutions to many resource related challenges that have been faced by circular economy innovators. Moreover, the feedback-rich nature of circular economy models might conversely make them particularly suitable to help extract value from the large amount of data generated by IoT.

Motivated by the above issues, this paper presents the Hy-LP – a hybrid protocol and development framework for the Industrial IoT. The main contributions in the field include:

- The seamless operation of IIoT devices and services within and across domains
- Integration of IoT deployments with the Industrial Cloud and its associated applications
- Lightweight design, considering the limitations of constrained, heterogeneous embedded devices
- Integration of established and mature communication security solutions (e.g. SSL/TLS)
- Enabling the shift to Green networking and a sustainable, Circular Economy, via the increased efficiency and usability over existing solutions

Other prominent approaches that provide seamless and lightweight IoT interactions are also identified, highlighting a representative, standardized protocol for each of these approaches. Moreover, a CPS featuring M2M interactions is selected, designing and implementing the required entities and interactions, while adapting them to the intricacies of each protocol. Finally, the developed solutions are deployed and evaluated on a common testbed, while the performance is also assessed in the context of the network characteristics recorded on an actual, operating wind park, allowing valuable conclusions to be extracted.

The paper is organized as follows: Section 2 presents and compares widely used solutions for IoT interconnection and management. Section 3 details the proposed Hy-LP protocol. Section 4 evaluates the performance of our system, compared with other settings, over a real IoT application. Section 5 concludes and refers future work.

2. Current IoT Solutions

Surveying the academic literature and the web, reveals a plethora of protocols aiming to unify IoT devices and applications [15, 16, 17, 18]; some are still in their infancy, some are openly available, some are proprietary, and there are
also significant efforts to standardize some protocols. The main approaches are
detailed in the subsections below.

2.1. Synchronous Communication

An interesting approach for IoT interactions is the use of protocols following
the Representational State Transfer (REST) architecture, which currently
dominatestheWorldWideWeb.RESTfulimplementations typically use the
Hypertext Transfer Protocol (HTTP), but the latter is not appropriate for IoT
applications, when considering the resource, bandwidth and energy restrictions
of the target devices.

Thus, the Internet Engineering Task Force (IETF) Constrained RESTful
environments (CoRE) Working Group presented the Constrained Application
Protocol (CoAP [15]), now an IETF standard. CoAP is a specialized web trans-
fer protocol for use with constrained nodes and constrained networks in the
IoT, aiming to maintain compatibility with the existing Internet infrastructure,
through simple proxies. The protocol is often referred to as "the HTTP for the
Internet of Things". It follows a request/response model, where a client may
interact with the server using a subset of the HTTP methods, namely using
GET, PUT, POST and DELETE on the server’s resources. CoAP messages
are transported over UDP. The protocol features two layers: the Transaction
layer responsible for single message exchange between end points and the Re-
quest/Response layer which is responsible for request/response transmission and
resource management; thus providing reliability mechanisms and basic conges-
tion control. Moreover, basic publish/subscribe interactions are also supported,
as, by extending the HTTP GET method, a client can observe a specific re-
source. For security, CoAP applications support DTLS.

There is significant research interest in CoAP, with numerous efforts to lever-
age its extremely lightweight interactions in domains such as smart homes [19],
mobile IoT deployments [20], cloud services [21], healthcare [22], smart cities
[23] and industrial WSNs [24].

2.2. Asynchronous Communication

Message-oriented protocols typically focus on providing asynchronous data
transfers between distributed devices. Their focus is on reliable messaging,
including message buffers and Quality of Service (QoS) facilities, controlled by
centralized entities.

The Advanced Message Queuing Protocol (AMQP) implements this func-
tionality and is standardized by the Advanced Open Standards for the Infor-
mation Society (OASIS)\(^3\). It is designed for reliable communication via message
delivery guarantee primitives, like at-most-once, at-least-once, and exactly-once
delivery, and it is built upon a reliable transport protocol, such as TCP. The
protocol consists of two core components that handle communication: the ex-
changes and the message queues. Based on pre-defined rules, the exchanges

---

\(^3\)OASIS AMQP: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=amqp
route the messages to appropriate queues, which can store the data and later send it to the receivers. Moreover, AMQP provides a publish/subscribe communication model by defining a messaging layer on top of the underlying transport layer. Two message types are supported, the bare messages and the annotated messages respectively. The first type is supplied by the sender while the second type is seen at the receiver. In comparison with the REST approach, AMQP performs better under a high volume of message exchanges [25]. The protocol is mainly used in business messaging [26] while in the IoT context it is suitable for the control plane of the server-based analysis functions [25].

The MQ Telemetry Transport (MQTT [16]) is another message-oriented protocol, introduced by IBM in 1999 and recently standardized by OASIS, as the IoT developments brought it back into the limelight. It is also standardized as ISO/IEC 20922. MQTT was designed as an extremely lightweight publish/subscribe messaging transport, for small sensors and mobile devices, optimized for high-latency or unreliable networks. An MQTT Broker is responsible for handling and organizing all communications between the various devices. Messages are published with specific topics, and each client can subscribe to various topics (though the Broker may require username/password authentication before allowing subscription). Topics are organized in a hierarchical manner, like the folder structure in a file system; e.g. home/kitchen/oven/temperature could be a topic where a device can subscribe to get updates on the oven’s temperature. When a client publishes a message, the Broker then relays this message to all clients subscribed to the message’s topic. Thus, all interactions are asynchronous and clients only communicate directly with the Broker. MQTT relies on TCP and secure deployments support the use of TLS. As with the previous protocol, researchers have already studied MQTT in a variety of domains, including eHealth applications [27], [28], WSNs and smart grid [29], smart homes [30] and also mobile IoT contexts [31], among others. In comparison with AMQP in the IoT domain, MQTT is more suitable as it scales better and requires less computational resources and less effort to implement on a client [32].

The eXtensible Messaging and Presence Protocol (XMPP) is developed for instant messaging and connects people via text messages over TCP. It is standardized by the IETF as the RFC 6120 and is mainly utilized for chatting and other message exchange applications. In the IoT domain, XMPP implements an easy way to address devices [34] and has been used as a protocol for Software-Defined Networking (SDN). However, it was designed for near real-time, human-to-human (H2H) interaction, and it is not practical for M2M settings as it does not provide any quality of service guarantees. Moreover, the messages are formatted in XML, which further increases the computational and communication overhead due to lots of headers and tag formats. Thus, XMPP is rarely utilized in IoT applications due higher latency and resource consumption.

---

2.3. Service-Oriented Approach

Service Oriented Architectures (SOAs) provide an attractive option for IoT node interactions. This approach has already been successful in business environments, as web services allow stakeholders to focus on the services themselves, rather than the underlying hardware and network technologies. The OASIS standard Devices Profile for Web Services (DPWS [17]) is the most attractive solution. It should be noted that DPWS was originally conceived and introduced as a successor to UPnP, but nowadays is actively pushed by industry stakeholders as the solution of choice for large-scale enterprise (e.g. industrial) deployments, while UPnP is mostly targeted to the home environment (printers, home entertainment etc. [35]). Also, like UPnP, DPWS is natively integrated into the various versions of the Windows operating system.

The specification defines a minimal set of implementation constraints to enable secure Web Service messaging, including discovery, description, synchronous (via operation invocations) and asynchronous (via subscription and event-driven changes) interactions on resource-constrained devices. The profile’s architecture includes hosting and hosted services. A single hosting service is associated with each device while the same device may accommodate various hosted services. The latter represent the device’s various functional elements and rely on the hosting service for discovery. For security, DPWS supports the use of TLS. The mechanisms detailed in the WS-Security specification are also applicable, as with any other Web Services deployment.

DPWS enables the adoption of a SOA approach on embedded and sensor devices with limited resources, allowing system owners to leverage the SOA benefits across heterogeneous systems that may be found in smart environments. The use and benefits of DPWS have been studied extensively in the context of various applications areas, which, other than the ones already mentioned, include automotive [37] and railway systems [38], industrial automation [39], eHealth [40], smart cities [41] and smart homes [42] and buildings [43].

2.4. Comparison

CoAP, MQTT, and DPWS share some important characteristics which make them good candidates for IoT and Industrial IoT (IIoT) applications, and which motivated us to consider them for this study. More specifically, all three protocols: are open standards with significant traction in the research community and the industry; are designed with constrained environments in mind; can offer seamless M2M interactions; run on IP; and have a range of implementations readily available to developers and researchers alike. While CoAP messages are transported over UDP, MQTT relies on TCP, and DPWS uses a combination of both (TCP for the bulk of the device interactions, and UDP for device discovery and other auxiliary functions); with each protocol inheriting different characteristics from the underlying transport mechanisms. For example, DPWS is constrained for local network communication only, as the implication of relying the multicasting features of UDP restricts the standard’s application for Internet-wide. The underlying protocols also affects the available security mechanisms, with DPWS and MQTT deployments supporting the use of TLS, and
CoAP applications supporting DTLS. In the case of DPWS, the mechanisms detailed in the WS-Security specification are also applicable, as with any other Web Services deployment. MQTT is suited, by design, for publish/subscribe interactions, CoAP also has support for observing resources, partly covering such functionality, but it is better suited for synchronous interactions, instead of event-based ones. DPWS is more flexible in this regard, as the WS-Eventing specification enables a feature-rich publish/subscribe functionality, including interactions that are triggered at pre-defined intervals and/or when a specific event takes place. Moreover, QoS is an important aspect in MQTT, with the protocol supporting three different modes of message delivery (Fire and forget, Delivered at least once and Delivered exactly once), whereas CoAP only offers a rudimentary choice between Confirmable and Non-confirmable messages. The former have to be acknowledge by the received with an ACK packet, in applications where it is necessary to cater for UDP’s unreliable transport. DPWS has no such features built-in, relying solely on TCP’s delivery mechanisms. Various extensions enhance the reliability and QoS features of Web Services (e.g. [35]), but these have not been integrated into DPWS yet.

A detailed and hands-on comparison of the protocols in the context of an actual application, as presented in [36], revealed that, performance-aside, DPWS was the benchmark in terms of the ease in designing the framework. Its robust and flexible discovery, subscription and eventing mechanisms meant that the entities and their interactions could be designed in an intuitive manner. This is also true for the end application, as it is the most hassle-free variant from the end users’ perspective; minimal setup is required and the entities discover each other and interact seamlessly, no matter where they are deployed on the network. CoAP was intuitive to work with, especially considering that as most developers nowadays have experience with RESTful applications. Still, careful study of the protocol and its limitations (theoretical and/or in terms what is supported in the existing APIs) is needed, as it is not as mature as the other two protocols considered. Lastly, MQTT’s lack of synchronous interaction support, meant that we had to follow a not so elegant approach in designing the entities’ interactions, with too many interactions happening in order to bypass the limitation of only supporting asynchronous interactions that have to be routed through a Broker.

3. The Hy-LP Protocol

As indicated by the comparison above, the protocol choice necessitates careful consideration of the target application, as no ideal protocol exists; some protocols have more features and are more mature than the alternatives, while others are more lightweight, some are ideally suited to aggregating data from a variety of sensors, while others are better suited for end-user (e.g. consumer) applications, etc. Thus, a complex deployment could benefit from the use more than one protocol or the development of a custom protocol that combines the benefits and alleviates the disadvantages of the standalone protocols. The latter
approach does not fully sacrifice interoperability with existing solutions, combining one or more protocols, delegating to each one a task that it’s more suited for. Thus, for example, CoAP could be used for the lightweight M2M interactions it can provide, MQTT for the cross-domain communications and the SOA-based approach of DPWS could be used for M2H interactions (to leverage the ubiquity of web service support in all human-operated devices).

Motivated by this insight, we propose the Hy-LP – a novel service-oriented framework for the IoT setting. The framework addresses the main constraints of the individual protocols, enabling seamless communication of IoT and IIoT devices and actuators over the Internet. Hy-LP is then directly compared to DPWS, since, as described above, it stood out in previous works as the most flexible, mature and feature-rich protocol, despite the inherent performance limitations and LAN-only operation.

To overcome the latter, instead of HTTP and UDP multicasting of DPWS, Hy-LP leverages the CoAP and MQTT protocols for synchronous and asynchronous communication respectively. Moreover, Hy-LP inherits the Broker architecture of MQTT. Messages are passed through a central server (the Broker), enabling one-to-many and many-to-many communications. A publish/subscribe protocol is constructed, which accomplishes service/device discovery and eventing functionality.

Knowledge of the topics and the message format is assumed. As aforementioned, DPWS utilizes SOAP to form messages. Our framework utilizes the JavaScript Object Notation (JSON) open-standard format (a replacement of XML) to structure the messages’ context. JSON is simpler, human-readable and consumes less resources than XML [44]. The average packet size of Hy-LP

Figure 1: Hy-LP setting and operation mapping.
is around six times smaller than the relevant DPWS packets, resulting in lower processing time, memory requirements and energy consumption. Moreover, QoS is an important aspect, with the Hy-LP framework supporting three different modes of message delivery, similar with MQTT (Fire and forget, Delivered at least once, and Delivered exactly once). Figure 1 illustrates a typical Hy-LP deployment.

First, the devices publish their profile information to the broker, including the relevant IP address (in contrast to the local network addresses of the DPWS multicast). The broker can be either local or remote, enabling cross-domain interaction. In order to discover a device or service, the actuator sends a request message to all public devices through the broker, who implements the corresponding multicasting functionality. The compatible entities respond to the request by sending descriptive metadata. Figure 2 illustrates the sequence diagram of the discovery operation.

For synchronous communication, the actuator invokes the appropriate operations and communicates directly with the entity via CoAP. For asynchronous operation, subscribe or eventing, the messages are passed through the broker over MQTT. Figure 3 illustrates the sequence diagram of the event subscription operation.

---

**Figure 2: Sequence diagram of Hy-LP’s discovery operation.**

**Figure 3: Sequence diagram of Hy-LP’s event subscription operation.**
Figure 4, illustrates the equivalent DPWS setting. Actuators send multicasting requests to discover the active devices and their services. The compatible devices respond with their profiles and available services, including the relevant local network addresses. Then, the actuator communicates directly with the selected services via HTTP and TCP. For the invoke operation, the device sends the required information once, when requested. For the eventing operation, the device transmits information when an event occurs (e.g. change of monitored parameters).

Figure 4: DPWS setting and operation mapping.

4. Evaluation

In this section, a comparative performance evaluation is conducted for the proposed Hy-LP and DPWS. To this end, two relevant versions of an IoT system are implemented on embedded devices and evaluated under a common testbed. Utilizing the proposed deployment, the user can access the aforementioned functionality through the Industrial Cloud. The application is implemented on the Greek Research and Academic Community cloud service, named okeanos\(^5\). With Hy-LP, the user can access the devices directly through Internet. The broker is deployed on the cloud to enhance scalability. For the DPWS version, devices are accessed through a LAN-gateway. Regarding security, all the communication of the gateways with any backend systems is encrypted with TLS.

\(^5\)okeanos: https://okeanos.grnet.gr/home/
Moreover, to evaluate the performance on an actual application, the policy-based access control (PBAC) framework presented in [41] is developed and deployed on the testbed. Said framework provides access control to the resources of IoT nodes, based on policy constraints centrally managed by the system owner. The policies are modelled based on the OASIS-standardized eXtensible Access Control Markup Language (XACML [46]). The policy repository is maintained in the cloud along with the broker for Hy-LP and in the LAN-gateway for DPWS.

Figure 5 illustrates a typical IIoT deployment spanning different wind parks; a typical use case required in the specific domain. Each distinct wind park installation features several wind turbines, while several sensors, industrial control systems and communication equipment exchange data in the wind park’s LAN. Moreover, exchanges take place between the wind parks and a central location, and, in the future, the industrial cloud, through a gateway at the edge of the wind parks’ networks. The wind parks are connected and managed over an SDN and Network Function Virtualization (NFV)-enabled network infrastructure that aggregates all services, providing business applications from centralized SDN controllers.

The variants of Hy-LP and DPWS are depicted in the upper left corner. The Hy-LP user (green color) can access all the sensory devices through Internet, while the DPWS user’s (orange color) flexibility is constrained as she can gain access only to a specific wind park LAN.

4.1. Testbed

We examine the effects of the different approaches and evaluate the performance of the aforementioned frameworks (Hy-LP and DPWS) over a common benchmark suite. We implement two relevant versions of the PBAC framework.

As an example of the framework’s flow of interactions, consider the case of a wind turbine. The turbine is a device that hosts a service which supports multiple operations such selecting operation mode, getting the current rotations per minute status or even events such as notifications when the tilt of the wings changes and/or some threshold has been reached. As soon as the available device and its services are discovered, a user can request access to the node that is of particular interest, e.g. in order to extract the latest values from the sensor attached to it. The request is intercepted by the node’s policy enforcement (PEP) module which then forwards the request to the policy decision point (PDP), the latter running on the wind park’s trusted device. The PDP has to consider all applicable policies, enriched by any relevant information, from the central policy administrator and information repository. The Policy Information Point (PIP) and Policy Administrator Point (PAP), which are deployed at this end, act as a source of attribute values and are used for creating and managing policies or policy sets. Once all the required information has been collected, the PDP issues a decision which is sent back to the node’s PEP. Based on that decision the PEP may or may not allow the guest to access said nodes data of interest.
The IoT devices are BeagleBone nodes [47] – ARM Cortex-A8 single core CPU running at 720MHz (throttled at 500MHz during testing) with 256MB DDR2 RAM. The testbed for the Service Orchestration was the slightly more powerful and versatile Beagleboard-xM [48] – 1GHz ARM Cortex-A8 processor (throttled to run at 600MHz during testing) and 512MB DDR2 RAM, also running a minimal Linux-based operating system. The access control infrastructure was deployed on a desktop system as these are expected to run on more resource-rich devices (e.g. the main system used to control and configure our smart home) – Core i5 CPU at 3.3GHz, 8GB DDR3 RAM. Finally, all systems were interconnected via wired Ethernet to minimize the network’s impact on the reported performance figures. The setup for all variants of the application appears in Figure 6.

4.2. Performance

The two frameworks enable the seamless communication among the user, the IoT devices and the central policy repository. We implement the Hy-LP framework with the programming language Golang (a compiled statically typed language design by Google). For DPWS, we utilize the novel NodeDPWS
library we had presented in previous work, built using Node.js (an interpreted JavaScript-like language for server-side web applications, also designed by Google). Both implementations support IPv6, necessary for IoT applications.

One of the most important aspects compared during benchmarking was the client-side response time, as this refers to the delay a user would experience in each case when trying to access the protected resource (e.g. the turbine’s rotations per minute). The testbed also featured a client application developed to discover and query the devices, recording response times, for benchmarking purposes. In total, 1000 requests were made to each device from our benchmark client while we measured response times, CPU and memory. Figure 7 presents the delay of Hy-LP and NodeDPWS. Random spikes are observed due to the garbage collector counterparts of both Golang and Node.js. The average response time of Hy-LP is 2.32ms and NodeDPWS is 23.63ms. Based on the response time, Hy-LP is significantly faster than the NodeDPWS (one order of magnitude faster). This is the result of the design approach for Hy-LP where a high amount of interactions is performed once and maintained in the broker, like the discovery and eventing operations, while for DPWS has to broadcast the requested information every time it is needed.

The CPU usage is similar for both frameworks, as illustrated in Figure 8, with Hy-LP exhibiting slightly lower utilization. Figure 9 illustrates the memory consumption. The use of lightweight primitives in Hy-LP (e.g. CoAP and JSON) instead of mainstream ones in NodeDPWS (e.g. HTTP and XML) results in significant reduction of memory demands. Our framework consumes around 7.1 MB where NodeDPWS requires as high as 51MB.

Energy consumption is essential for IoT networks and becomes a major issue as the number of devices rises, since they require constant maintenance for battery charging/replacing. Hy-LP consumes around 1472 mJ on average while NodeDPWS requires around 14963 mJ to operate (an order of magnitude
higher), which is strongly affected by the aforementioned performance results. The low energy requirements can enable the employment of wireless energy harvesting techniques [50] to prolong the lifetime of such networks in an environmental friendly manner.

![Execution time](image1)

**Figure 7:** Execution time of Hy-LP and NodeDPWS.

![CPU utilization](image2)

**Figure 8:** CPU utilization of Hy-LP and NodeDPWS.

4.3. Performance Analysis – The Wind Park Use Case

As is evident from the analysis above, Hy-LP is more efficient and consumes less resources than the competing DPWS. However, to analyze the protocol’s performance in the context of an actual vertical application, an SDN/NFV-enabled wind park is examined, as a characteristic use case of next generation industrial networks operating over 5G communication infrastructures. In the wind park, several sensing devices are deployed (see Figure 10) that transmit
periodically information to the backend local Supervisory Control and Data Acquisition (SCADA) servers. The sensing data is used to monitor and/or react on environmental or other operational parameters. The role of this operation is to read data from analog or digital sensors and transfer it to the backend via a gateway device. These links are currently wired, but in the context of upcoming industrial environments [51] and other critical applications [52] they will be replaced with wireless connections.

In the context of project VirtuWind, we analyzed traces from an actual, operational wind park (in Brande, Denmark), in order to evaluate the specificities of industrial traffic. The examined setting acquires four wind turbines, connected in a redundant star topology. The turbines themselves consist of two switches in series, one at the bottom and the other at the top. Numerous measurements systems, sensors, and actuators are linked to these switches, communicating with a SCADA server also connected to the star topology. The connection between the central switch and the Internet is enabled through a router. Figure 5 shows the topology of the turbines and the typical networks (Ethernet and Profinet) within a wind turbine. The Park Control System consists of two main parts:

- **Wind Farm SCADA System** – enables the reporting, supervision, acquisition, and storage of data from the turbines
- **Wind Farm Grid Control System** – controls the power output of the different wind turbines and adapts it to the grid operator demands

In the context of IIoT, the traces focus on traffic to/from the SCADA server, which was captured for approximately 1000 seconds of operation. Analyzing these network traces, in conjunction with the application requirements and how the wind parks currently function, enhance the interpretation of the evaluation outcomes in the context of the actual specific application.
Several connections with low data rates (about 20.0000) that are contained in these traces can be ignored in the context of IIoT wireless sensor motes and their applications, including services such as Network Time Protocol (NTP), Dynamic Host Configuration Protocol (DHCP), and Simple Network Management Protocol (SNMP) exchanges. The remaining traffic includes TCP and UDP connections between the wind turbines and the SCADA server. The TCP ones, though critical, only have end-to-end requirements of 100 ms, 250 ms and 500 ms depending on the specific service, while the latter (i.e. instantaneous single-packet UDP exchanges) have a more stringent end-to-end delay requirement of 10 ms. These numbers can be compared with the end-to-end delay observed by Hy-LP, which is, on average, much lower, as illustrated in Figure 7. The ordinary delays in the round-trip timings dominate the trace communications. As is evident from the observed performance figures, the proposed Hy-LP protocol, constitutes a viable solution even for the most stringent connections in the context of the specific industrial setting examined (i.e. those requiring a delay of maximum of 10ms).

Figure 10: Typical Wind Turbine sensors and data networks.

5. Conclusions

This work presented Hy-LP – a hybrid protocol and framework for IIoT networks. Hy-LP enables seamless and lightweight interactions with and between smart IIoT devices, in inter- and intra-domain deployments. It provides devices and their associated services with discover, publish/subscribe and eventing (i.e. asynchronous), as well as synchronous (via direct invocation of exposed features)
capabilities. Security is also taken into account and all communications can be protected with mature, robust solutions, such as TLS. Moreover, a policy-based access control framework is supported, as demonstrated, that facilitates the access rights of different users. As a proof of concept, a CPS is implemented on a wind park setting, where IIoT devices and actuators communicate information through cloud. The protocol addresses the main obstacle of the LAN-based DPWS and accomplishes interactions of devices over Internet. Moreover, HY-LP is faster and consumes less resources than DPWS. The proposed solution is lightweight and can be deployed on resource constrained embedded devices, respecting the strict QoS requirements observed in the target, operational wind park. As future work, the overhead imposed by the various security mechanisms will be examined in more detail, while the protocol and development framework itself will be made available to the researchers’ and developers’ communities.

Another important aspect of consideration for future work is the impact of the 5G communication technologies’ umbrella to the IoT and in particular the existing technological IoT building blocks, such as the presented hybrid protocol. This impact, should not be examined in a technological isolation, but it should also be seen in the wider context of the collateral interplay that this evolution will have in a cross sectoral and cross disciplined manner. IoT and intelligent assets, being considered as key enablers of Circular Economy are highly affected by the technological advancements that the 5G technology will bring. The enabling competitiveness, as well as the industrial transformation, will have positive effects also in the context of the circular economy. New technologies help to make products, services, manufacturing and processing cleaner, safer, more secure while contributing in the use of materials and energy as efficiently as possible given the need for reduced waste and emissions.

6. Acknowledgement

This work has received funding from the European Union’s Horizon 2020 research and innovation programme VirtuWind under grant agreement No. 671648. The authors would also like to thank the network engineers maintaining the subject wind park in Brande, Denmark for their valuable input in interpreting the network traces and defining the application requirements.

References


