Using Open Standards for Utilizing IoT Sensors in a Smart City Scenario

Frank T. Johnsen[a]
Trude H. Bloebaum[a]
Marianne R. Brannsten[a]
Ketil Lund[a]
and
Didrik E. Aubert[b]

[a]Norwegian Defence Research Establishment (FFI)
P.O. Box 25
2027 Kjeller, Norway

[b]Technical University of Denmark (DTU)
Anker Engelunds Vej 1, Building 101A
2800 Kgs. Lyngby, Denmark

Track 3, Paper 31

Point of contact: Frank-Trethan.Johnsen@ffi.no

ABSTRACT

Although the use of the Internet of Things (IoT) in a smart city context primarily stems from civilian applications like management and monitoring, there are also other uses. For instance, military forces may utilize sensors during natural disasters, terrorist attacks or insurgent attacks, in order to improve situational awareness. For this to be possible, the military forces must be able to discover the sensors, as well as utilize the data formats of the sensors in their Command and Control (C2) systems. It is therefore necessary to be able to describe the sensors in a standardized and machine-readable way, such that they can be automatically discovered and their capabilities assessed.

In this paper, we investigate using open standards such as the OpenAPI Specification (OAS) and Sensor Model Language (SensorML) to enable sensor sharing so that IoT can be leveraged by military forces in an agile and interoperable way, which is necessary to support coalition operations. Furtermore, we have started on a Proof-of-Concept implementation, and in the paper, we present the implementation thus far, as well as the results achieved.

1. INTRODUCTION

A key enabler for military sovereginity is Situational Awareness (SA).[1] In short, *SA is how the situations are understood by an individual*. Having a good and *shared* SA enables better and faster decisionmaking. As new technology becomes available, new and more efficient ways for information sharing, and thus shared SA, emerge. Already back in 1999, the United States Department of Defense started investigating how to make use of such technology in military operations, through their concept of Network-Centric Warfare.

Later, in 2005, the North Atlantic Treaty Organization (NATO) adopted this concept as NATO Network Enabled Capability (NNEC),[2] and in the following years, NNEC was embraced by the member nations. For example, in 2007 Norway included the concept in the Norwegian Armed Forces Joint Operational Doctrine calling it "Network Based Defence".[3] In these concepts, organizational boundaries are removed, while decision makers, sensors, and effectors collaborate to carry out the operation.[1–3] Because of this, machine-to-machine communication becomes more important, which in turn means that interoperability between the parties involved becomes an important issue.

On the civil side, cities around the world are steadily increasing in size and population, and for the first time in history, more than half of humanity lives in urban areas.[4] With the growth in the population and complexity of cities comes an increased need for management and monitoring tools, and such tools constitute a major part of the *smart city* concept. One area that is particularly important in the context of smart cities is the Internet of Things (IoT). Based on the idea of connectivity for everything, anywhere and anytime,[5] IoT offers the means to make such management and monitoring possible. The general concept of IoT is to connect any device to the Internet, allowing devices to collect and transmit data,[6] and the purpose is to *improve the quality and productivity of life, society, and industries*.[7]

With an increasing share of the population living in cities, the chance of serious incidents taking place within a city may increase, as well as the consequences of such incidents. Additionaly, more and more cities are deploying sensors and offering smart city capabilities. In such cases, the management and monitoring tools of a smart city may prove to be highly useful also for other purposes. For instance, military forces may utilize sensors during natural disasters, terrorist attacks or insurgent attacks, in order to improve SA. For this to be possible, the military forces must be able to discover the available sensors, and be able to utilize the data formats used. Once this is in place, the advent of the IoT can provide additional sensing capabilities, which in turn contribute to SA in military operations. In NATO, the Research Task Group (RTG) Information Systems Technology (IST) 147 "Military Application of the Internet of Things" (IST-147) aims to make use of civilian IoT trends for military purposes. In the context of this group, a technical architectural approach to using IoT to support enhanced SA has been developed.[8]

The IST-147 architecture is shown in Figure 1, illustrating how sensor data from a smart city can be leveraged in different ways. The right side of the figure shows the basic levels included in the architecture within the smart city domain, and from bottom to top, these are:[8]

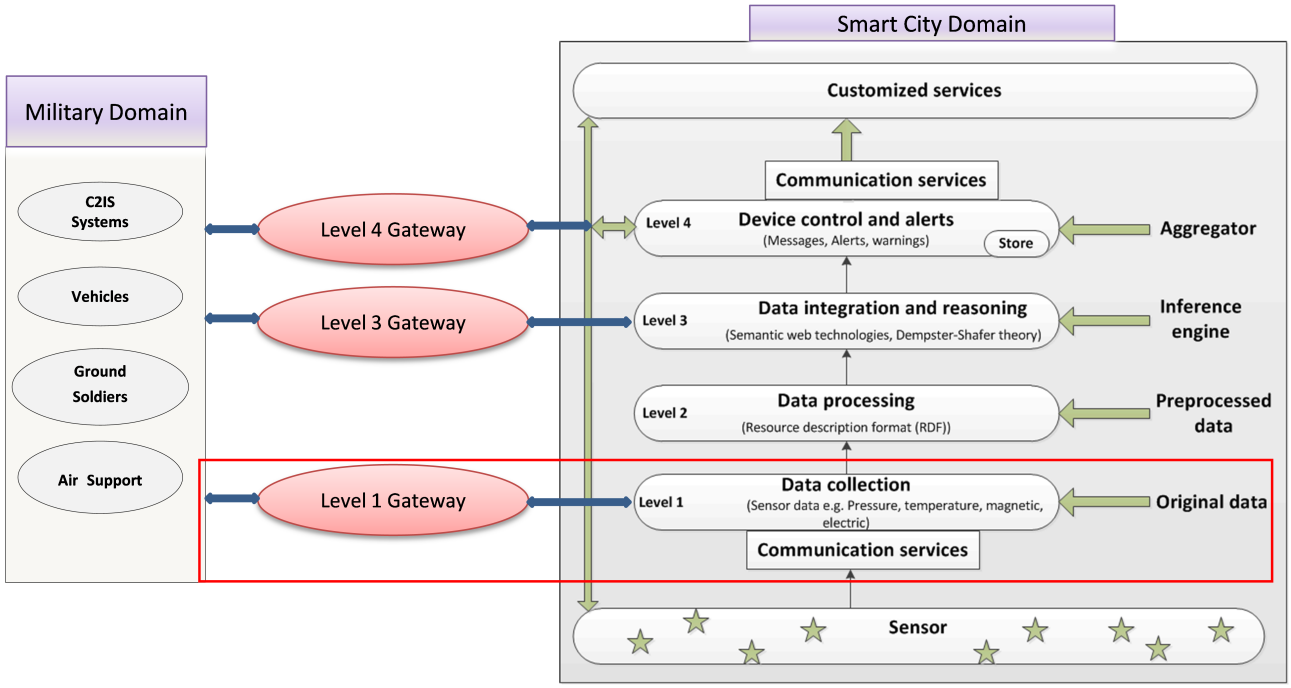


Figure 1: The IST-147 architecture.[8] The red box shows the main focus of this paper.

- Level 1: The raw data from the sensors is being collected at this level. Such data may be in a standardized format, but depending on the sensor, it may also be proprietary formats, which complicates the process of utilizing the information.
- Level 2: The data collected at level 1 is processed at this levels using various algorithms.
- Level 3: The data from various sources is integrated and reasoned over at this level using technologies such as Semantic Web.
- Level 4: The data from level 3 is used to generate alerts, control actuators, or issue commands to integrated devices. Such alerts can also be sent to recipient devices of interested partners of the smart city infrastructure. Examples of such partners are citizens of the city, third party service providers or administrative units of the city.

The left side of Figure 1 illustrates how systems in the military domain can utilize the information from the smart city domain. The figure shows that information can be obtained through gateways, which can be located on level 1, 2 or 4, depending on the type of information needed. In general, obtaining information at lower levels means that more processing must be done by systems within the military domain, while the higher levels provide processed information.

As a first step towards a Proof-of-Concept (PoC) implementation of this architecture, we have started at the bottom, investigating how to obtain data on level 1. A red box has been added to the figure to show which part of the architecture we address in this paper, and we aim to address other aspects in future work.

From the military point of view, the main question is: How can the original data from the smart city sensors be obtained and used? This question is then broken down into four key questions that must be addressed:

1. Which architectural approach should be taken for the sensor system? The approach must enable a network of sensors that allows for both sharing of raw sensor data (the Level 1 Gateway), but also supports higher level functionality, such as data processing. The latter will enable information sharing at Levels 3 and 4 later.

2. How can sensors be discovered? Not being a normal part of the smart city infrastructure, the military must be able to discover available sensors in order to retrieve the data.

3. How can sensors be described? Having discovered available sensors, an overall description of each sensor is needed, in order to assess its capabilities and characteristics.

4. How can sensor data be retrieved? In order to fetch data from a sensor, an application programming interface (API) for the sensor is needed.

In this paper we address these four key questions. First we discuss alternative architectures for sensor systems in Section 2, and identify the architecture that is most suitable in our scenario. Following this discussion Section 3 goes into detail on questions two and three. First, we investigate how sensors can be *described* in a standardized and machine-readable way, and how information about the sensor services can be shared so that they can be automatically *discovered* and have their capabilities assessed. This is done by identifying existing standards that can support this functionality, and then describing how these standards can be used to ensure interoperable discovery between sensors and information consumers.

After the sensors have been discovered we also need to support information retrival, where information consumers access the data offered by the sensors either directly, or after the data has been processed. In order to support as many sensors and consumers as possible, it is important to leverage best practices for how to do information exchange in large-scale distributed systems. A common approach in modern systems is to use REpresentational State Transfer (REST)[9] APIs, which are widely supported by both commercial sensor systems and potential consumer devices, such as smart phones. Section 4 goes into detail on how this approach can be used to support retrival of information from the sensor systems.

Following this discussion, we present our PoC implementation in Section 5. Section 6 sets our work in context with other related work, before Section 7 concludes this paper.

2. ARCHITECTURE DISCUSSION

In this section we discuss the alternative approaches to system architecture for sensor systems mentioned in question one above, and propose and discuss three different design approaches to how IoT sensors may be connected to existing systems that support SA.

First, we identify some factors that help set the scope of our discussion:

- Existing military sensor systems are typically purpose-built and expensive, and use their own propritary mechanisms for connecting to C2 systems. This paper aims to investigate the benefits of bridging civilian IoT approaches with military needs, and special purpose military sensor systems are thus out of scope.

- In a smart city, sensors may be either wired for power or be battery powered. In the case of a power failure or if the battery wears down, sensors will become unavailable if they run out of power. This means that being able to discover sensors that are currently available is important in a mission, since one does not want to waste time and resources attempting to use sensors that are no longer available.

- In our discussion we focus on integrating IoT sensing capabilities, so that information will flow from the IoT sensor into the SA systems. Though one could envision also allowing the SA systems to control actuators, that is currently beyond the scope of our PoC efforts.

Given the stated scenario and scope, an initial design of the system is sketched in Figure 2. As depicted, the only known components of the system are the IoT sensor devices and the SA application. The purpose of the design is to send data from the sensor devices so that the server may receive them. The nodes and communications in between may be approached in different ways.

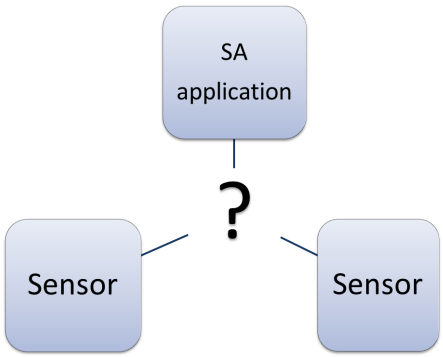


Figure 2: Design problem: Which configuration is suitable?

2.1 Thoughts on tactical networks

When integrating non-military sensor systems with military SA applications, one needs to consider the limitations imposed by the devices and networks that are likely to be used. Below are some constraints that networked devices can be subjected to in a military setting (note that these are also constraints in a civilian setting, but less so):

- The operation enviroment (topology, weather, signal interference etc.) may result in limitations in, or disruptions of, the network connectivity between any two devices.
- Cellular networks may be unavailable or reduced in different areas.
- Devices may rely on temporary power sources, which in turn means they must be resource efficient in terms of power consumption.

Different tactical networks are structured in different ways, based on the intended usage of the network, so there is no one structure that is representative for all tactical networks. However,[10] presents an emulated scenario for battalion-sized operations, which we use as our network example. Given the tactical network description presented in the paper, we know that tactical networks can support a number of different units, ranging from stationary through deployed to mobile units, which all have different limitations with repect to their network limitations, computational capabilites etc.

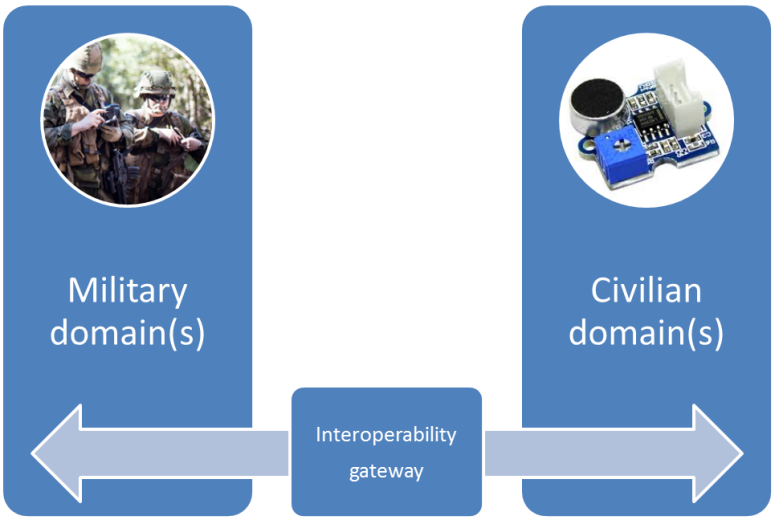


Figure 3: Users from different domains are often connected via an interoperability gateway.

A common feature is that the communication between units in the military domain and civilian systems (or between military units from different domains, e.g. from different nations or organizations) often goes through interoperability gateways that are responsible for bridging between data formats, communication protocols, security policies etc., as shown in Figure 3. For the purpose of our further discussion of the architecture alternatives, we assume a similar high-level structure with a common network (Internet or otherwise) connecting the components using a variety of different bearer technologies.

In the smart city scenario we assume that the main information flow will be going from the sensors towards the SA applications via the gateway. Typical smart city sensors are deployed ahead of time, by entities such as local governments, organizations and businesses, and are thus likely to be stationary. The interoperability gateways are also predetermined, and thus likely to be stationary as well.

The amount of latency that is acceptable in a smart city scenario depends on a number of factors, with the intended usage of the information often being the determining factor. When the information is used as a basis for rapid decission making, the timely delivery of the information is important. This means that any approaches that introduce significant latency overhead should be avoided. The reliability requirements of the data transport also depends on the information content and intended usage of the information.

2.2 Possible system architecture approaches

Below we present three approaches towards disseminating data from the sensors to SA applications and discuss which approach is best suited for our scenario.

2.2.1 Approach 1: Peer-to-Peer Systems (P2P)

In a P2P system, the peers (nodes) communicate and exchange data with each other, without a central server that possesses all the data. Each peer can act as both a client and a server, meaning that it can provide data to, as well as requesting it from, other peers. Data is eventually guaranteed to any peer that requests it. This guarantee is ensured by replicating subsets of data across multiple peers when network connections are available. One way of implementing distributed storage across the peers is distributed hash tables (DHT).[11]

A P2P system involves an *overlay network*, which is an implementation/application that resides on top of the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. P2P overlay networks can be grouped into two types; *unstructured* and *structured* overlay networks.

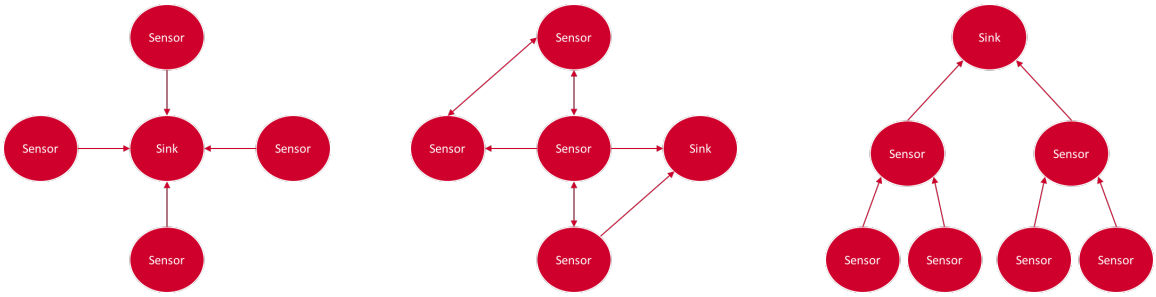
In *unstructured* P2P overlay networks, the peers connect to each other randomly, and as indicated by the name, the nodes do not form or maintain a particular structure. This type of P2P network is simple to set up, as it is easier to configure than a structured network. However, searching for data is more complex, and often means generating a substantial amount of network traffic. The simplest approach to searching such networks is to use flooding, which one can consider as a brute force operation.[12]

Structured networks arrange the peers in a specific topology. Typically structured overlay networks implement a DHT for storage and retrieval of data (key/value pairs distributed across the peers). This type of distributed storage allows peers to retrieve values of the corresponding keys efficiently. However, with high rates of *churn*. Churn is the frequency of peers joining and leaving a P2P network.[12] This type of network overlay has significant downsides. Each peer must maintain a set of neighbors, which will be hard to update frequently.[12]

Based on the fact that most data producing nodes are stationary in our scenario, a structured P2P system is better suited than an unstructured one. Thus, in our below evaluation of this architecture alternative, we consider a structured approach.

2.2.2 Approach 2: Fog Computing

Fog computing is the concept of extending cloud computing capabilities to the edge of the network, close to the end devices. This means that fog nodes can perform on-site data analytics and/or provide services to edge devices without continuous interaction with the cloud.



(a) Star topology (b) P2P mesh topology (c) Tree topology

Figure 4: Topologies in Wireless Sensor Networks

In this approach, there is a potentially large amount of data producing *edge nodes* with limited computational capabilities. Because of this, the raw data is pushed towards the core of the network, where the more computationally capable *fog nodes* can perform data processing etc. before passing the processed data on to the final receiver of the data, which is often a data storage cloud.

In our scenario, the sensors are the edge nodes, as they produce the raw data. The data reciever is either the interoperability gateway (which in turn forwards the information to other systems, after having done the needed interoperability transformations, for instance between transport protocols as described in[13]) or the actual SA application if it is directly reachable. The role of the fog node can in some cases be fulfilled by the gateway, but a more likely scenario is that the fog nodes would be an additional capability between the sensors and the gateway that implements metadata tagging, data processing and/or other functionality from Layer 2 in the smart city architecture in Figure 1. In this case, the fog nodes would be a key component in enabling message exchange at the higher layers in the same architecture.

2.2.3 Approach 3: Wireless Sensor Networks (WSN)

A WSN consists of sensor nodes and sink(s). Sinks are merely the destination to which sensor nodes transmit their recorded signals. Sinks also act as gateways, and upon receiving signals, these are sent over the Internet to some server. In our scenario, the sensor nodes forward information to the gateway, which takes on the role as the sink, which in turn forwards the data to the SA application.

Figure 4 depicts three common WSN topologies:

- In the star topology (subfigure 4a), all the sensors are directly connected to the sink.
- In the mesh topology (subfigure 4b), the sensors are interconnected, which means that more than one hop may be required to reach the sink. A node can connect to more than one node in this topology. Consequently, the transmission path from a node to the sink might differ over multiple signal recordings.
- In the tree topology (subfigure 4c), the nodes are structurally interconnected - if an intermediate sensor goes down, this might risk that all child nodes may not reach the sink nodes, as there might not be any alternative transmission paths.

Which topology is most suitable depends on how the information flow in a given scenario is structured, and how the underlying transport technologies used in the scenario function. There are different types of transport protocols in use in IoT, ranging from standard request/response message exchange, to both direct and brokered event based message exchange (see Section 4 for further details on these message exhange patterns). If each sensor node needs to communicate directly with the sink (due to for instance using request/response), either a star topology or a mesh topology is likely advantageous, while a brokered exchange pattern fits well with the tree topology.

Specifically for the tree topology,[14] proposes a multi-hop based routing approach that has several benefits for military applications, including:

- Faster to deploy and usually does not rely on infrastructure.
- No centralized control gives the possibility to set up the network automatically.
- Can be attached to the existing system to provide more detailed information.
- Configuring multi-hopping properly can preserve energy.

A routing algorithm is utilized by the nodes involved, to continuously self-organize the tree.

A multi-hop based routing approach consists of *source nodes*, *cluster head nodes* (CH), and the sink node. Source nodes record sensor signals and transmit them to the parent (a cluster head). CH nodes are intermediate nodes that serve as relay nodes, responsible for forwarding the data up the tree, eventually arriving the sink node.[15]

2.3 Discussing the three approaches

2.3.1 Approach 1

A structured P2P network can be of either static or dynamic topology; which is best is decided by the churn rate (low churn rate leans towards static, and high towards dynamic). In our scenario we can assume that the churn rate is low, since the sensor components will be stationary when deployed, and the topology of a structured network will be (seemingly) static.

Even though the structured network will take some proximity measures into account when the peers are building their neighbor sets, the approach may suffer from excess latency as data may be relayed across multiple peers. Considering tactical networking conditions, having to send information via a number of application level peers rather than directly to the receiver might generate an additional and undesired network load in addition to the increase in latency.

P2P overlay networks reside on top of the TCP/IP stack, implying that the peers must implement the TCP protocol. TCP provides reliable and ordered delivery of messages over the network. In networks with high data rate errors or frequent disruptions, this behaviour might not be desirable for many information types, as it comes at a cost of additional network load.

The combination of additional traffic to support the P2P structure and the additional overhead of sending messages via other peers, means that implementing a P2P system comes with a risk of excessive overhead when it comes to data exchange, and concerning retaining routing tables.

In a P2P network, false or compromised peers can conspire against the network and can prevent correct message delivery. Note that depending on the specific type of P2P network chosen, a single peer may not be able to do this, in which multiple peers must cooperate to achieve the attack. In cases where the overlay network is DHT based, a single false peer can compromise it merely by returning wrong data upon query from a legitimate peer.[11]

2.3.2 Approach 2

Given the smart city scenario, once the sensor nodes are deployed, they will remain stationary and connected to the closest fog device. This can be both a benefit and a drawback, as few structure changes means that information flow can be easily managed. The failure of a fog node might however lead to the loss of information from a whole set of sensors, and compensating for that failure might require physical interaction with the fog or sensor nodes.

An additonal feature of using a fog approach is that data can be processed close to the sensor that produced the information. Depending on the sensor type in question, this mean that the raw sensor information flow (which is often large and/or transmitted frequently) does not have to traverse the entire network. The processed information is often smaller in size (as only the relevant data is included in the processed data product), meaning that less information in total is transmitted across the network. The cost of this can include higher power consumption on the fog nodes and an increase in latency (to allow for computation time on the fog node). The degree to which these factors occur depends on the specific data flow and processing required.

For edge devices to communicate with fog devices, and for the fog devices to forward the data, the fog devices act as routers. This behavior introduces a spectrum of attack vectors that Local Area Networks (LAN) are vulnerable to, e.g., Address Resolution Protocol (ARP) spoofing, man-in-the-middle, eavesdropping, etc. To be able to attack, the attacker needs to authenticate with the router, and one possible way of doing this is to employ the *evil twin* attack (in case the fog/router is a *wireless* access point) on the edge devices. The attack fools devices to connect to a fake access point that appears as a legitimate one. A successful evil twin attack allows attackers to exploit the mentioned LAN vulnerabilities.

2.3.3 Approach 3

The WSN topology is autonomous and self-organized. Source nodes and CH nodes will, in turn, change their roles – meaning that the topology will change continuously.

For a source node to send data and for it to eventually end up at a sink (gateway), multiple hops may be required, in which the latency can suffer. One advantage of using a multi-hop approach is that the nodes can be strategically placed to avoid Line-of-Sight limitations (for instance in cases where the altitude varies).

WSN networks in a military context would require attention to security aspects, typically the standard Confidentiality, Integrity, and Availability (CIA) triad. Proper confidentiality and integrity require several cryptographic elements, such as authentication protocols, nonces, and key management.These aspects can be problematic to enforce on the sensor nodes, as they are constrained in the sense that they have limited power supply, computational power, and space available. This means that specially tailored solutions that provide some security within the capability bounds of the devices is needed.

2.3.4 Selected Approach

Based on the expected structure and limitations of tactical networks, it can be seen that the first approach will be least appropriate. This is because the overhead of maintaining a structure (or of searching for information if one considers an unstructured approach) can be significant. Additionally, disruptions in the networks might lead to nodes coming and going from the overlay network despite the nodes being stationary, something which will increase the churn at the overlay level. Security is also an issue in such P2P networks.

Both the other two approaches matches well with the structure of the information flows we expect to see between sensors and end users, namely that the main flow is unidirectional from the sensor towards the application systems. Topologies that either connect sensors directly to the sink (WSN star topology), or hierarchical topologies that streamline information flow towards the sink (WSN mesh and tree topologies, and the fog architecture) support this unidirectional information flow well.

Based on the above discussion, the fog architecture seem to have the highest potential when connecting military users to smart city sensors. This is due to the support for data processing, which can both be used to reduce network load and to enable information exchange at higher layers of the architecture in Figure 1. Because of this, we chose to follow this architecture apporoach in the remainder of this paper.

3. DESCRIBING AND DISCOVERING SERVICES

A uniform approach to describing and discovering sensor assets is needed to ease interoperability.

3.1 Describing services

In IST-147, we consider SensorML[16] for this purpose. It is a standard for describing sensors, approved by the Open Geospatial Consortium. The original motivation behind SensorML was as a means to process observations without a-priori knowledge of the sensor or processor characteristics. This would contribute to avoiding stovepipe systems for processing sensor data within different communities.

However, SensorML provides a number of other advantages as well: In its simplest form, it provides a standard way of describing sensors. Through a collection of metadata, SensorML also provides a means for sensors to become discoverable, through a rich set of metadata. It can either be used to describe individual sensors, aggregated sensor capabilities in a gateway level, or both. However, it must be coupled with an ontology and

reasoning to automate sensor descriptions, data interoperability and provide a real-time overview of available sensors.

In civilian use of IoT, the challenges identified are *interoperability*, *complexity*, and *scalability* issues. Recent international standardization and research and development initiatives aim to address these challenges. Notably, these challenges are the same as we have identified for the military applications of IoT in IST-147. In a recent keynote,[17] Drira has pointed to six out of 400 existing ontologies that are considered relevant for civilian IoT applications:

- Base ontology
- Smart Appliances REFerence (SAREF) ontology
- IoT-O
- Spitfire
- IoT ontology
- IoT-lite

This narrowing down of the plethora of choices out there is helpful to further studies in this direction, though eventually it will be up to NATO as a community to decide if SensorML should be used, and if so, exactly which ontology should be applied to it. Our impression so far is that SensorML seems to be a useful tool to provide overarching metadata, but it cannot be used without a suitable ontology.

3.2 Discovering services

Being able to discover the IoT assets (e.g., services that provide sensor information) in run-time is important. Since sensors can break, run out of power, etc. it is important to be able to find the sensors that are actually available when one wants to consume data from them.

There exist many service discovery protocols, each with their own benefits and drawbacks. One approach in particular has been promoted by the Technology for Information, Decision and Execution superiority (TIDE) community is multicast Domain Name System (mDNS). TIDE has developed several specifications on how to use mDNS for service discovery, along with approaches to tunnel the information over wide area networks and also integrating it with other service discovery approaches like Universal Description, Discovery, and Integration (UDDI). These specifications are available at Tidepedia[18] to registered users. NATO Allied Command Transformation (ACT) is the custodian of TIDE specifications, and as such it makes sense to consider applying these in a NATO context.

4. USING SERVICES

There are two fundamentally different communication paradigms which may be employed to share sensor data: A pull-approach, typically implemented as the request/response pattern where the client initiates a request to a server. This approach is described in Section 4.1. A push-approach, typically implemented as a publish/subscribe pattern where the client sets up a subscription and data is later pushed according to the active subscriptions. This approach is described in Section 4.2.

4.1 Retrieving sensor data – API approaches

Using open standards is an important precursor to exploiting IoT information from a smart city, since interoperability between data sources and consumers is necessary. The formats required depend on the type of sensor, whether it provides discrete or streaming data. Sensors offering streaming data would need different formats from those delivering discrete data, for example, a sensor capable of video streaming (e.g., a surveillance camera) could use H.264[19] coding which is a standard for efficient video coding.

With IoT, discrete sensors are often exposed as Web APIs (Web services). Examples of open data formats are JavaScript Object Notation (JSON), typically used with REST Web services, and eXtensible Markup Language (XML), typically the foundation of SOAP Web services. Of these, JSON is currently much used to provide smart city data from sensors presenting discrete data through a REST API. There are several API specification formats that can be used to describe such APIs: The OpenAPI Specification (OAS),[20] RESTful API Modeling Language (RAML),[21] API Blueprint,[22] Web Application Description Language (WADL),[23] OData,[24] and Web Services Description Language (WSDL).[25] Of these, the latter, WSDL, is specific to SOAP Web services, as it provides an XML standard for describing the service API. The others may all be used with a REST Web service approach, though OAS is gaining ground these days and is much used for describing REST APIs. In our opinion, the single most important tool today is the Swagger framework, which implements OAS and provides a metadata API to describe the assets made available through the framework. This combination can be found in many smart cities. For example, OAS is used by the Finnish Transport Agency to describe the Digitraffic API.[26] This API provides access to sensors deployed throughout Finland. By using the metadata provided sensors can be found based on their type (weather stations, traffic cameras, road surface condition sensors, visibility meters, precipitation meters, wind sensors, etc.), their geographical location, current status, and other parameters.

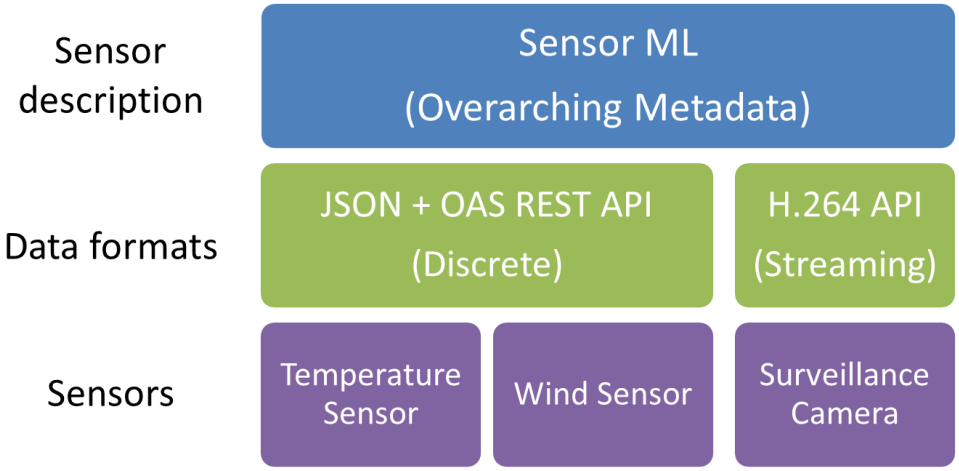


Figure 5: Describing smart city sensor assets.

Figure 5 gives an example overview of how smart city assets may be described. Here, we have both discrete and streaming sensors available. These sensors need to offer data according to a standard format and API. Finally, overarching metadata is needed to allow for uniform description and discovery of sensors.

4.2 Subscribing to sensor data

While the request/response pattern requires the consumer to actively fetch data from the producer (i.e., the consumer pulls data), the publish/subscribe pattern implies that the consumer expresses interest in a given type of data by setting up a *subscription*. The type of data of interest is typically specified using either topics or content filtering, and this specification is included in the subscription request. When new data on a given type (e.g., a new temperature reading) becomes available, the producer sends the new data to all consumers that have expressed interest in that type of data. This pattern is particularly well suited for IoT, as many sensors produce information more or less periodically, and thus, a push pattern can reduce network activity considerably.

In addition, a *broker* is commonly used between producer and consumer, in order to unload administrative tasks from the producer, something which is particularly important in IoT, as many sensors are relatively resource-constrained. The broker typically handles subscription management and dissemination, so that the producer only has to send new data to the broker, which then handles all further dissemination. Figure 6 shows a publish/subscribe information flow, where one or more publishers publish data to a broker. A set of subscriptions have been set up a-priori, and the broker uses these subscriptions to determine which messages should be sent to a given subscriber.

There exist a number of publish/subscribe standards, three of the most well-known being Advanced Message Queuing Protocol (AMQP),[27] Message Queuing Telemetry Transport (MQTT)[28] and Web Services Notification (WS-Notification).[29] We have evaluated all these three standards, and this is further described in the next section.

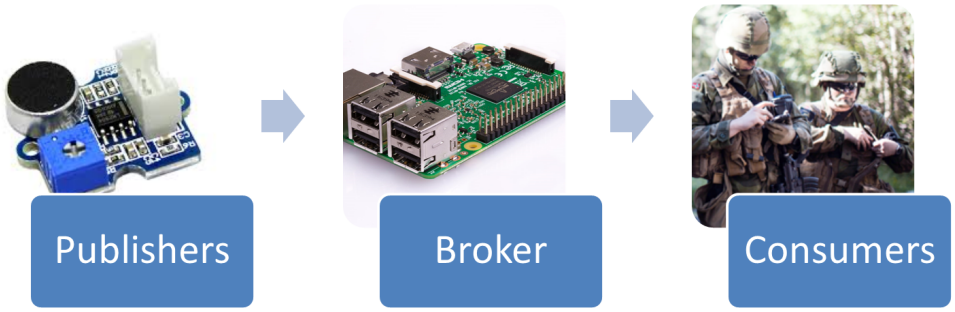


Figure 6: Publish/subscribe information flow.

5. PROOF OF CONCEPT

In IST-147 we chose to use MQTT as the publish/subscribe protocol since it is already much used for civilian IoT applications. Starting our PoC, we did a comparative protocol evaluation between three civilian standards: AMQP, MQTT, and WS-Notification. We chose to measure the end-to-end delay, i.e., the time from the message was sent from the publisher until it was received in the consumer via the broker. The reason for choosing this metric was that we wanted to identify the protocol that would induce the least delay from data was produced until it could be consumed, as timeliness is important for achieving SA. Table 1 shows the protocol comparison. Here, we see that for our purpose (lowest delay of sensor data) we should opt for MQTT.

AMQP	MQTT	WS-Notification
3.103124	2.576307	11.085122

Table 1: Comparison of end-to-end delay between AMQP, MQTT, and WS-Notification. The time is measured in seconds, and constitutes the time from the start till the end of a burst of 100 messages is sent till it is received.

As our PoC, shown in Figure 7, we chose to use Raspberry Pi 3 single-board computers as the hardware platform. The reason for this was that the Raspberry Pi is a cheap, capable, and representable device for IoT applications. There are also many sensors available, making it a good platform for rapid IoT prototyping. Our setup consists of three Raspberry Pi 3: One equipped with a Sense Hat[30] and another equipped with an Enviro pHat.[31] These two provide the sensing capabilities of our PoC. We have exposed the sensor capabilities with Swagger and OAS. The third unit does not have any sensors, but serves as a gateway towards a common MQTT bus using the Mosquitto[32] MQTT client implementation. Consequently, this unit also serves as the gateway between the request/response and the publish/subscribe protocols In key with TIDE specifications the Raspberry Pi 3 devices are discoverable using mDNS. The components work together in the following manner:

1. The Mosquitto Client (i.e., the Raspberry Pi 3 functioning as the gateway to MQTT) initiates a REST request to the service exposing the Sense Hat API.
2. The REST service providing the Sense Hat API processes the request, and responds with a payload of JSON-formatted discrete sensor data.
3. The Mosquitto Client initiates a REST request to the Raspberry Pi 3 exposing the Enviro pHat API.
4. The REST service providing the Enviro pHat API processes the request, and, just like the Sense Hat service, returns a JSON payload.

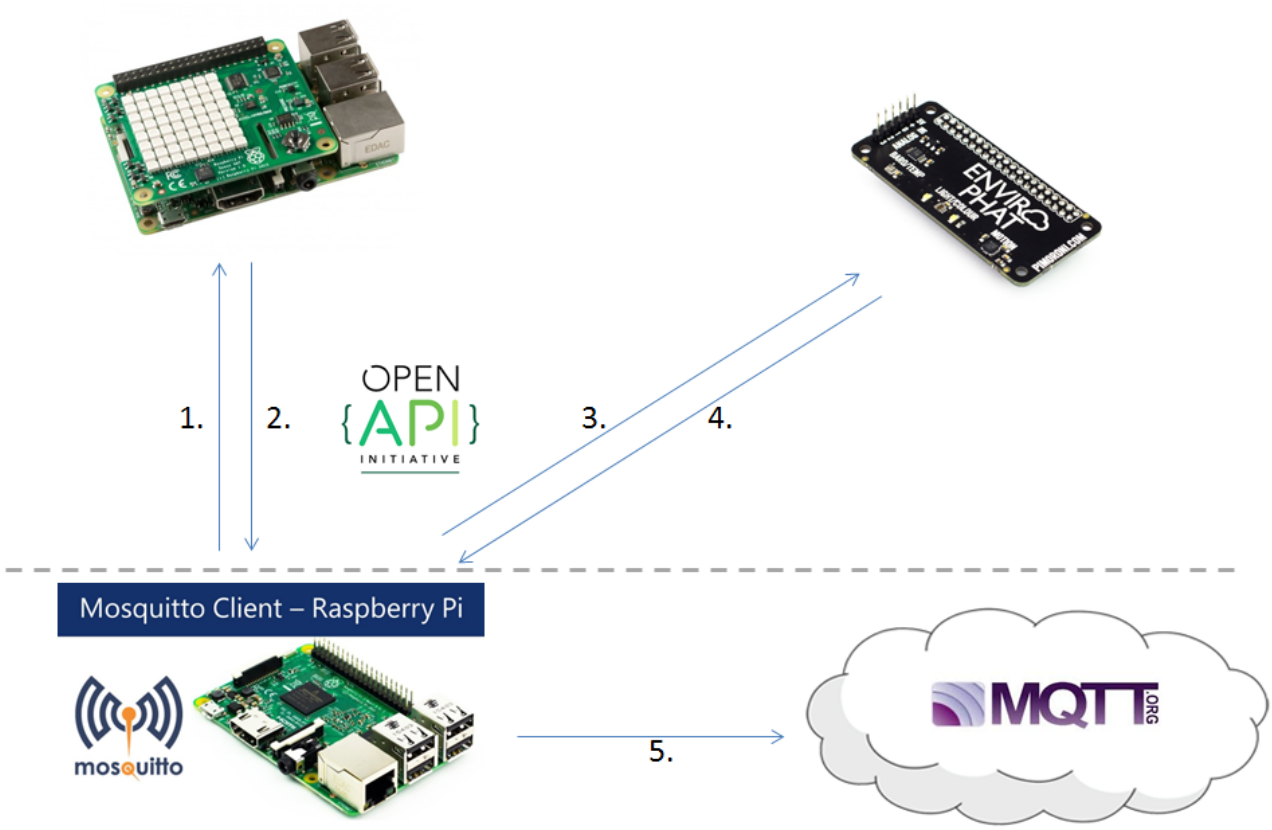


Figure 7: Proof-of-Concept implementation overview. Our two sensors provide data according to the Open API Specification (steps 1-2 and 3-4) to the MQTT gateway, which publishes the data to a common MQTT bus (step 5).

5. The Mosquitto Client publishes the data from the sensors to the MQTT broker, splitting the data into different sensor values and corresponding MQTT topics.

As a result of this process, the information from the sensors is availble through the MQTT broker. This information can then be forwarded into the military domain by a gateway, which can, if needed, perform a transformation from MQTT to the protocols supported in the military domain.[13] Note that steps 1-2 and 3-4 are pull-pattern request/response, whereas step 5 is push-pattern publish/subscribe. That said, the pull steps do not need to go across the network, as the REST to MQTT functionality can co-reside with the REST service on the actual device, so that only the messages published with MQTT traverse the network. This is a matter of how and where the Mosquitto Client software is deployed.

It is important to note that due to the current lack of agreement on the "best" ontology to use together with SensorML, we chose a slightly different scheme for identifying sensor data in our PoC. Here, we instead let the topic that we publish data under indicate the type of data to expect in the corresponding payload. This solution works where this is the agreed-upon approach (i.e., IST-147 bespoke topics and data format), but in the long run NATO, as a community, needs to find a viable approach to describing sensors, as discussed earlier in this paper.

In IST-147, we decided to arrange topic names so that they carried meaning beyond merely being a name for a data dissemination channel. Lacking a standardized approach, we chose a scheme where the topic could be used to indicate how the data received should be interpreted. So, the JSON data we send is always structured the same way for these discrete sensors, but the meaning of the "value" changes from topic to topic. Also, the

sensor owner would be indicated in the topic, so a topic with "NOR" in it would imply a Norwegian sensor. This approach to structuring topics is further discussed in.[33] The JSON message template is shown in Listing 1. For example, a temperature reading from the Enviro pHat is shown in Listing 2. There we can see that the message conveyed as JSON contains a "lat" and a "lon" value to give the position of the sensor reading. A timestamp ("UTC"-field, denotes that Universal Time Coordinated (UTC) is the timezone used) gives the time of the sensor reading, while the "msg_id" gives a unique identifier for this particular message. The identifier is generated as per uuid4 described in RFC4122.[34] Finally, the "value" is the actual sensor reading. Here, and also for our Sense Hat, it should be interpreted as Celsius, but another sensor type might indicate a different interpretation. The different topic suffixes provided in our PoC are summarized in Table 2.

```
{
    "lat": latitude,
    "UTC": timestamp,
    "msg_id": uuid,
    "lon": longitude,
    "value": value
}
```

Listing 1: JSON message template: The latitude and longitude are decimal values, the timestamp and uuid are strings, and the value differs with the sensor type.

```
{
    "lat": 52.245621,
    "UTC": "2018-02-01 10:17:30.227125",
    "msg_id": "4427ccee-ab06-4614-92b9-af209edd2b1c",
    "lon": 21.012371,
    "value": 27.4
}
```

Listing 2: This example is taken from a message exchange where the topic suffix is 'sensor/enviro/temp'. This indicates that this is a sensor of type Enviro pHat and that the reading is a temperature, implying that the value in the JSON above should be interpreted as Celcius.

Origin sensor	Topic suffix	Value interpretation
Enviro pHat	sensor/enviro/temp	Degrees Celsius
Enviro pHat	sensor/enviro/light	Lumen
Sense Hat	sensor/astro/temp	Degrees Celsius
Sense Hat	sensor/astro/humidity	%rH (relative humidity)
Sense Hat	sensor/astro/pressure	Millibars

Table 2: Sensors, topic suffixes used, and how to interpret the JSON "value" field.

6. RELATED WORK

The overall domain of interest emerges from Suri et al.[35] In the paper, the authors have investigated potential issues and benefits of applying Commercial off-the-shelf (COTS) IoT devices in operational military scenarios.

For an overview of applications of IoT in the context of defense and public safety, see the paper by Fraga-Lamas et al.[36] Security is a challenge in IoT systems. Barcena and Wueest give an overview of such challenges and the

requirements of IoT systems.[37] Wrona discusses security from a mission-critical systems perspective,[38] whereas perspectives of integrating commercial IoT with such systems is covered by Sorniotti et al.[39]

Karagiannis et al. have performed a survey of relevant IoT data protocols.[40] Their paper examines the following protocols: The Constrained Application Protocol (CoAP), MQTT, Extensible Messaging and Presence Protocol (XMPP), REST, AMQP, and Web socket. Important characteristics of each protocol are identified, and strengths and weaknesses explained.

Fog computing is one of three paradigms (the two others being cloudlet and mobile edge computing) that fall under the term *edge computing*. These paradigms are compared in work by Dolui and Datta using the following six parameters: physical proximity, logical proximity, non-IP support, context awareness, power consumption and computational time.[41] Based on the parameters, they build a decision tree to be used to determine which of the three paradigms are best suited to given scenarios. They conclude that despite the amount of research done in these fields, there is a lack of specific implementation standardizations of the three paradigms. This limitation increases the uncertainty and difficulty when choosing the appropriate paradigm for implementation in a system.

Data mining in edge devices is a strategy used to reduce the number of packets transmitted from edge nodes through the network. Gaura et al. propose edge mining: an approach in which the edge devices drop sensor recordings instead of transmitting them to a fog node.[42] This is done by only passing sensor data once a new recording has reached a certain threshold, or a specific time has passed.

Raafat et al. present an approach for classifying sensor data as normal or abnormal signals.[43] Their paper tested the approach on four sensor types; light, temperature, CO_2 and humidity. The authors conclude that the approach is promising, and processing sensor data on fog nodes can result in more efficient transfer to the cloud.

7. CONCLUSION

In this paper we have presented the technical architecture approach from the NATO RTG IST-147, and discussed how the original data from the smart city sensors could be obtained and used.

We raised four questions in the paper, that we aimed to answer through discussion and an initial PoC using open standards and best practices.

To address each question in turn:

1. *Which architectural approach should be taken for the sensor system?*
   We think the Fog computing approach is most suitable.

2. *How can sensors be discovered?*
   We suggest looking into the mDNS specifications for service discovery as proposed by the TIDE community.

3. *How can sensors be described?*
   We think using the SensorML standard as overarching metatdata for all smart city sensors could be a good idea. This, however, raises a new question: Which ontology should be used together with SensorML? This question must be addressed by NATO as a community if SensorML is to be used.

4. *How can sensor data be retrieved?* We have discussed different communication paradigms (request/response and publish/subscribe), and in our PoC show how we can integrate both to provide data. It is important to base APIs on open standards to ensure interoperability.

For future work we aim to perform larger scale experiments with MQTT, and also to investigate the opportunities of meshing brokers which some MQTT implementations offer.

REFERENCES

[1] Alberts, D. S., Garstka, J. J., and Stein, F. P., "Network centricwarfare. developing and leveraging information superiority.." 2nd. Library of Congress Cataloging-in-Publication Data, Aug. 1999.

[2] P. Bartolomasi et al., "Nato network enabled capability feasibility study.." Version 2.0. Oct. 2005.

[3] Diesen, S., "Forsvarets fellesoperative doktrine 2007." Forsvarsstaben. Utarbeidet av Forsvarets stabsskole, 2007. ISBN: 978-82-92566-01-5. (in Norwegian).

[4] Demographica, "Demographia world urban areas 13th annual edition: 2017:04." 2017, Available: `http://www.demographia.com/db-worldua.pdf`.

[5] ITU, "The internet of things." ITU Internet Reports 2005, Available: `http://www.itu.int/internetofthings/2010`.

[6] Morgan, J., "A simple explanation of 'the internet of things'," (2014).

[7] Kashyap, S., "10 Real World Applications of Internet of Things (IoT) Explained in Videos," (2016).

[8] Pradhan, M., Fuchs, C., and Johnsen, F. T., "A survey of applicability of military data model architectures for smart city data consumption / aggregation." IEEE 4th World Forum on Internet of Things, 05-08 February 2018, Singapore.

[9] Fielding, R., "Architectural styles and the design of network-based software architectures." Doctoral Dissertation, University of California, Irvine, 2000. Available: `https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

[10] Suri, N., Hansson, A., Nilsson, J., Lubkowski, P., Marcus, K., Hauge, M., Lee, K., Buchin, B., Mısırhoğlu, L., and Peuhkuri, M., "A realistic military scenario and emulation environment for experimenting with tactical communications and heterogeneous networks," in [*Military Communications and Information Systems (ICMCIS), 2016 International Conference on*], 1–8, IEEE (2016).

[11] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S., "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials* **7**(2), 72–93 (2005).

[12] Wikiwand, "Battalion | Wikiwand," (2018).

[13] Eirik Bertelsen et al., "Federated publish/subscribe services." 9th IFIP International Conference on New Technologies, Mobility & Security 26 to 28 February 2018. Paris, France.

[14] Lee, S. H., Lee, S., Song, H., and Lee, H. S., "Wireless sensor network design for tactical military applications: Remote large-scale environments," in [*Military Communications Conference, 2009. MILCOM 2009. IEEE*], 1–7, IEEE (2009).

[15] Arioua, M., el Assari, Y., Ez-Zazi, I., and El Oualkadi, A., "Multi-hop cluster based routing approach for wireless sensor networks," *Procedia Computer Science* **83**, 584–591 (2016).

[16] Open Geospatial Consortium (OGC), "Sensor model language (sensorml)." Retrived: December 19, 2017, Available: `http://www.opengeospatial.org/standards/sensorml`.

[17] Drira, K., "Toward open smart iot systems – an overview of recent initiatives and future directions." Keynote speech at NTMS 2018, Paris, France.

[18] TIDE Community, "Tide specifications on mdns." Available: `https://tide.act.nato.int/tidepedia/index.php?search=mdns`.

[19] ITU-T, "Advanced video coding for generic audiovisual services." ITU-T Recommendation H.264, 2003. Available: `http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-200305-S!!PDF-E&type=items`.

[20] OpenAPI Initiative, "Homepage." Available: `https://www.openapis.org/`.

[21] RAML Workgroup, "RAML – The simplest way to design APIs." Available: `https://raml.org/`.

[22] api blueprint, "API Blueprint Specification." Available: `https://apiblueprint.org/documentation/specification.html`.

[23] W3C, "Web Application Description Language." W3C Member Submission 31 August 2009, Available: `https://www.w3.org/Submission/wadl/`.

[24] Odata, "OData – the best way to REST." Available: `http://www.odata.org/`.

[25] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language." Available: `https://www.w3.org/TR/wsdl/`.

[26] Finnish Transport Agency, "Digitraffic Road metadata API." Available: https://tie.digitraffic.fi/api/v1/metadata/documentation.

[27] RabbitMQ, "Amqp 0.9.1 protocol specification." Available: https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf.

[28] OASIS, "Mqtt 3.1.1 protocol specification." Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[29] OASIS, "Wsn specifications." Available: https://www.oasis-open.org/committees/wsn/.

[30] pythonhosted.org, "Sense hat api reference." Available: https://pythonhosted.org/sense-hat/api/.

[31] Pimoroni, "Python libraries and examples for the pimoroni enviro phat." Available: https://github.com/pimoroni/enviro-phat.

[32] Eclipse Foundation, "Eclipse mosquitto." Available: https://mosquitto.org/.

[33] Manso, M., Johnsen, F. T., and Brannsten, M. R., "A smart devices concept for future soldier systems," tech. rep., Los Angeles, CA, USA (2017).

[34] P. Leach et al., "A universally unique identifier (uuid) urn namespace." Available: https://tools.ietf.org/html/rfc4122.html.

[35] Suri, N., Tortonesi, M., Michaelis, J., Budulas, P., Benincasa, G., Russell, S., Stefanelli, C., and Winkler, R., "Analyzing the applicability of internet of things to the battlefield environment," in [*Military Communications and Information Systems (ICMCIS), 2016 International Conference on*], 1–8, IEEE (2016).

[36] P. Fraga-Lamas et al., "A review on internet of things for defense and public safety." Sensors, vol. 16, no. 10, pp. 1-44, 2016.

[37] Barcena, M. B. and Wueest, C., "Insecurity in the internet of things." Symantec, 2015. Online: https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf.

[38] Wrona, K., "Securing the internet of things: A military perspective." IEEE World Forum on Internet of Things, 2015.

[39] Sorniotti, A., Gomez, L., and Wrona, K., "Secure and trusted in-network data processing in wireless sensor networks: a survey." J. Inf. Assur. Secur., vol. 2, no. 4, pp. 189-199, 2007.

[40] Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., and Alonso-Zarate, J., "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud Computing* **3**(1), 11–17 (2015).

[41] Dolui, K. and Datta, S. K., "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in [*Global Internet of Things Summit (GIoTS), 2017*], 1–6, IEEE (2017).

[42] Gaura, E. I., Brusey, J., Allen, M., Wilkins, R., Goldsmith, D., and Rednic, R., "Edge mining the internet of things," *IEEE Sensors Journal* **13**(10), 3816–3825 (2013).

[43] Raafat, H. M., Hossain, M. S., Essa, E., Elmougy, S., Tolba, A. S., Muhammad, G., and Ghoneim, A., "Fog intelligence for real-time iot sensor data analytics," *IEEE Access* (2017).