J. Sens. Actuator Netw. **2013**, *2*, 653-674; doi:10.3390/jsan2040653

OPEN ACCESS

Journal of Sensor and Actuator Networks

ISSN 2224-2708

www.mdpi.com/journal/jsan/

Article

An Ontology-Based Context Model for Wireless Sensor Network (WSN) Management in the Internet of Things

Yang Liu, Boon-Chong Seet * and Adnan Al-Anbuky

Department of Electrical and Electronic Engineering, Auckland University of Technology, Private Bag 92006, Auckland 1142, New Zealand; E-Mails: yliu@aut.ac.nz (Y.L.); aalanbuk@aut.ac.nz (A.A.-A.)

* Author to whom correspondence should be addressed; E-Mail: bseet@aut.ac.nz; Tel.: +64-9921-9999; Fax: +64-9921-9973.

Received: 1 August 2013; in revised form: 14 September 2013 / Accepted: 22 September 2013 / Published: 30 September 2013

Abstract: Wireless sensor networks (WSNs) are an enabling technology of context-aware systems. The Internet of Things (IoT), which has attracted much attention in recent years, is an emerging paradigm where everyday objects and spaces are made context-aware and interconnected through heterogeneous networks on a global scale. However, the IoT system can suffer from poor performances when its underlying networks are not optimized. In this paper, an ontology model for representing and facilitating context sharing between network entities in WSNs is proposed for the first time. The context model aims to enable optimal context-aware management of WSNs in IoT, which will also harness the rich context knowledge of IoT systems.

Keywords: ontology; context model; wireless sensor networks; network management; Internet of Things

1. Introduction

The Internet of Things (IoT) can be seen as an evolving suite of technologies which make it possible to interconnect digitally augmented everyday objects and spaces—e.g., home appliances and environments instrumented with sensing, processing, and communication capabilities—and creating interactions between them over large-scale interconnected heterogeneous networks such as the Internet. The IoT gives rise to the possibility of an omniscient awareness of the fine details of our physical world by providing anytime anywhere access to information about any interconnected 'things'. Therefore, the IoT is a potentially large-scale context aware system, where such awareness is used for making intelligent decisions and autonomous responses to situations and events that have occurred.

The IoT is also a heterogeneous system where multiple types of devices can co-exist over different types of networks. Different devices can have their own operating mechanisms. Therefore, it is difficult to communicate between heterogeneous devices, but it is possible to make such interactions easier by contextualizing the information to be shared or exchanged in order to reduce the dimensions of information (e.g., range of possible values) to be communicated.

Existing context aware designs have focused mostly on the application level aspects of the IoT, *i.e.*, on the provisioning of automated services to end users under particular environment or user situations. The underlying network states and conditions which can impact on the application performance, however, are often ignored [1,2].

There exist designs for implementing context aware services and applications for the IoT, but there have been hardly any attempts to bring context awareness down to the underlying networks. The IoT can integrate physical and virtual objects seamlessly in an Internet-like structure, and let applications and services interact with those interconnected "smart objects" based on contextual information. Unfortunately, the network itself has been left out of this "context aware" paradigm, and network optimization designs are not yet considering this context aware approach. Optimization techniques can be difficult to apply in the network as heterogeneous devices operate over hierarchical networks and every network device of different types can operate with its own standards and mechanisms. Not only do modifications have to be made to devices of one type to understand and interact with devices of other types, but additional communication and processing by the devices is also often required to make this happen. By contextualizing information such as network states and conditions to be shared or exchanged between heterogeneous devices, interactions between such devices can be made easier and the application of network optimization techniques involving heterogeneous devices can be facilitated [3].

The motivation of this paper is to provide a context model for modeling different situations and conditions of an IoT system with the aim of optimizing the underlay network using contextualized information. The networks can react autonomously to the contextual information, executing appropriate tasks and actions according to the system contexts. In this paper, an ontology-based context model is proposed for facilitating context sharing between heterogeneous devices and network optimizations exploiting context awareness in WSNs. The remainder of the paper is organized as follows: Section 2 overviews the challenges of network management under the IoT and context-aware designs for WSNs. It also introduces the general concepts of ontology based context modeling. Section 3 defines the context classes that will be used by the proposed context model. Section 4 presents in detail the proposed ontology-based context model for WSNs. Section 5 illustrates with an application scenario how the proposed model can be applied to model the context used in a context-aware system designed for WSNs. Finally, Section 6 concludes the paper.

2. Background and Related Work

2.1. Network Management under the IoT

Due to the nature of the IoT environment—*i.e.*, large scale, dynamic, distributed, and resource constrained—it is challenging to perform network management functions such as configuration management, performance monitoring, and resource allocation. With heterogeneous devices, it is difficult to find a standard or unified solution for managing the entire network system. The following provides a concise overview of recent representative works that attempted to address the network management issue in IoT.

The original concept of the IoT is to interconnect a plethora of pervasive distributed devices through the Internet. Thus, the SNMP (Simple Network Management Protocol) was adapted initially to manage the networks in IoT [2]. As wireless sensor networks become carriers for the majority of pervasive distributed devices, a network management system designed to perform WSN-specific management tasks such as new sensor discovery, sensor configuration, and sensor database synchronization was proposed for the management of WSNs in IoT [1].

There exist other works that focused on the issue of resource allocation. In [4], the concept of cross-utilizing nodes' resources such as processor, memory, transceiver, and sensing platform for multiple applications over multiple WSNs was proposed as an approach to efficient resources re-use in IoT. In [5], the authors considered the IoT as a hyper-network and proposed a semantic hyper-network model which exploits semantic relations among resources to support resource finding and mash-up.

As the IoT network management is expected to evolve towards the distributed and more automated management paradigm, promising enablers of such automation—context awareness and ontologies—are introduced in the next section.

2.2. Context-Aware Designs for WSNs

The term "context" is defined as "any information that can be used to characterize the situation of an entity" [6]. This definition is traditionally associated with the design of context-aware applications, where contexts are information that describes the situation of any entity relevant to the application. The entities can be people, places, or things. The applications are context consumers that receive information from context producers, and may also produce their own context that they provide to other context consumers (e.g., other applications). By exchanging contexts, the applications can adapt their behaviors according to shared contexts, and overall performance can be optimized as a result.

This concept of context can be extended to describe the situation of any entity relevant to a network when designing context-aware networks. The entities can be nodes, links, or protocols. It may also include human users and the physical environment of the network. Similarly, context can be shared between context producers and consumers in the network. The resulting increased level of context awareness can empower the network to make more informed and intelligent decisions for its operations, which will lead to more automated management and optimized performance.

A number of context-aware designs for WSNs have been proposed, which aimed mostly at addressing, through context awareness, one specific network-related issue, such as medium access control (MAC) [7], network routing [8–10], and network formation (clustering) [11]. In [12], the authors attempted to adapt protocols at multiple layers to improve the transmission of video sensors through context-aware cross-layer optimization. Unlike these conventional layered architectures, a non-layered protocol architecture is proposed in [13] where protocols can be modularized and dynamically composed of functional blocks according to context information. At the system level, context has also been used for management of device energy [14] and storage allocation [15] in WSNs.

2.3. Ontology Based Context Modeling

Context modeling refers to the process of creating an abstract representation of situations in the real-world such that it can be interpreted and exchanged by machines. Various context modeling approaches exist, such as key-value models, object-role based models, and spatial models [16,17]. However, for many of these approaches, either they lacked the capacity for context to be structurally formed and thus, efficiently shared between system components, or they were too application/system dependent so that models created for one application/system cannot be easily reused for another application/system. To overcome these issues, ontology based context modeling was introduced, which is also the approach that has been adopted in this paper.

Ontology is a mechanism that allows a formal and structured description of a set of concepts and relationships between them for an entity [18]. By conforming to a common and well-defined set of description specifications, ontologies can make context generation using diverse types of data sources from different entities easier. An example is the semantic context service proposed for smart offices in [19] where high-level context information is generated from low-level data in heterogeneous formats from different entities such as users, sensors, and positioning systems. Ontology based models can also make context sharing easier between different entities, and can be reused in part or in whole for different applications/systems.

There exist few ontology-based context models specifically for WSNs. In [20], a centralized model for context management in WSN is proposed. In this model, all contexts are organized in an ontology structure and stored on a relational database of a base station that functions as sensor sink in the WSN. Context querying and reasoning processes are also performed on the base station. In [21], a scheduling approach for distributed description logic (DL) reasoning is proposed. Unlike in [20], where context reasoning is performed centrally at a base station, here the reasoning tasks can be offloaded to several sensor nodes in the WSN based on context parameters such as sensor node resource and network characteristics.

2.4. Open Issues

As discussed in Section 2.1, some attempts have been made to improve existing management of networks in IoT, although there still remain some limitations with these existing designs. Adopting IP based solutions such as SNMP [2] is only feasible if the entire heterogeneous network is operating as an IP network. However, existing major WSN protocols such as Zigbee are non-IP based. Even with the recent emergence of 6LoWPAN, an IP standard for sensor networking, there will be some WSNs that are more suited to a non-IP platform, in particular for applications where response speed and low energy requirement are critical factors. Furthermore, many of the discussed network management schemes are based on a centralized design, which lacks the scalability and flexibility necessary to effectively support a heterogeneous network as large and complex as the IoT.

Network heterogeneity can increase the difficulty of network management due to the large amount of different information needed to be shared and understood between heterogeneous entities in the network. While there exist ontology-based context models for mitigating this complexity issue through system context awareness in a heterogeneous operating environment, most if not all of them are focused on modeling application or system level contexts (e.g., [19,20,21]), with very few or no ontology models proposed for network level contexts. Therefore, to bridge this gap in research, this paper is proposing an ontology-based context modeling approach with the aim of supporting distributed and automated WSN management in IoT. The next section defines the context classes that will be used by the proposed model.

3. Context Classification

3.1. Defining the Context

Existing context definitions and models are not appropriate to be applied directly to many network scenarios since they target mainly the application or system levels, and do not encompass contexts about network elements and events. In this paper, we modified and extended the context definition in [6] to define context for the IoT system as follows:

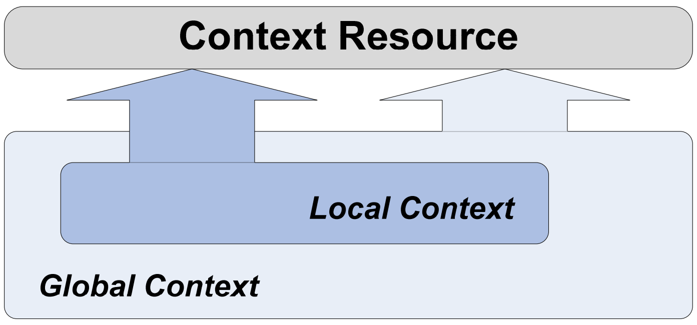
> **Definition 1**: *Context for the IoT system is any shareable knowledge to represent situations or conditions from different parts of the system. The range of contexts can include, but not limited to, device context, network contexts, system contexts, and environment contexts.*

The device context provides knowledge about local device conditions, such as the energy state, storage level, and services provided by a node. The network context represents network wide situations and states, such as network topology, overall transmission capacity, or path qualities in the network. The system context represents status of the IoT system, such as the current executing tasks of an application or the state of the IoT system performance, which can be shared with the underlying network. The environment context provides knowledge for a network to understand the changes of its environmental properties or attributes, such as the occurrence of a fire incident or detection of hazardous objects. Other types of context may also be included according to the system specifications. For example, user activity and user preference can be modeled under application context if the system has a user-centric design.

3.2. Local and Global Context

Generally, all contexts of an IoT system can belong to the category of either local or global contexts. We refer to the entire contextual information of an IoT system as Context Resource, as shown in Figure 1. The local context is the context that can be deduced locally within a single node device, and represents the conditions and states of the device as well as its constituent components, such as the state of the energy resource or the services carried by a node. On the other hand, the global context refers to contexts that cannot be deduced locally, but require the exchange of local contexts between multiple nodes. The global context may include the system and network level contexts, such as the network-wide conditions, or the user contexts generated at the system application level of the IoT system.

Figure 1. Local and global context.



Hence, a single node is aware of not only the contexts about itself, but also about other nodes, the underlying network and the IoT system based on which context-aware management of WSN in the IoT can be performed.

3.3. High and Low Level Contexts

Although contextual information can be classified according to their scope, *i.e.*, as local or global context, they can also be organized into levels, *i.e.*, as high or low level contexts. Depending on the scope of contextual information, high level contexts can represent the status of a node, network, or system as a coherent entity, while low level contexts can describe the status about the elementary component parts of a node, network, or system. The relationships between local/global and high/low level contexts can be summarized as shown in Figure 2.

Figure 2. Relationship between local/global and high/low level contexts.

	Local Context	Global Context
High Level Context	Device (node) level context	IoT system and network level context
Low Level Context	Sub-device (component) level context	System and network constituent level context

Low level local contexts refer to context about individual sub-components of a device, such as protocol states or sensor readings. Multiple low level local contexts can then be used to infer a high level local context, which represents the aggregated state of a device.

On the other hand, high level global contexts can cover the overall status of an IoT system and its underlying network, such as the overall QoS of the network, or the general performance of an application service provided by the IoT system. Low level global contexts can describe the status of those constituent parts of the system and network, such as the status of a routing path, the resource allocation states for data processing, or the environmental state from aggregated sensor readings, which are used to deduce high level global context.

For any context-aware network management scheme, when possible, it is essential to use high level contexts. Not only can high level contexts be inferred from multiple low level contexts, it is also possible to deduce low level contexts from a given high level context. For example, a single node can be constituted of multiple subsystems, such as the energy resource module, transceiver module, data storage module, and microcontroller module. Each subsystem is controlled and monitored by the operating system (OS) or firmware of the node. A high level local context such as "node is a cluster head" can be decomposed into corresponding low level local contexts for each subsystem, e.g., "fast energy depletion" for the energy resource module, "high inbound data traffic" for the transceiver module, "high memory load" for the storage subsystem, and "high CPU usage" for the microcontroller module. Thus, it is possible to make each single module aware of the status of other modules by deducing from high level contexts directly without having to exchange low level local contexts between them.

It will be even more necessary to use high level contexts when dealing with global contexts. Transmitting a piece of high level global context to be shared by all nodes in the network should incur less communication overhead than transmitting multiple low level global contexts. As communication can consume the majority of the overall energy resource for a node device [22], the amount of unnecessary communications should always be minimized by utilizing high level global contexts. In addition, data storage and processing requirements for the nodes can also be minimized, as the amount of information to be shared and processed from a single high level global context should be less than that from multiple low level global contexts.

4. Proposed Ontology-Based Context Model

The proposed context ontology model provides vocabularies to represent context knowledge about network related situations and states of the IoT systems. The model is designed to facilitate context exchange and the understanding of such exchanged contexts between heterogeneous nodes of an IoT system to enable optimal context-aware management of the system's underlying WSNs. Each node holds one instance of the model, which can use expression axioms to deduce the corresponding context knowledge according to the information exchanged at different levels and scopes.

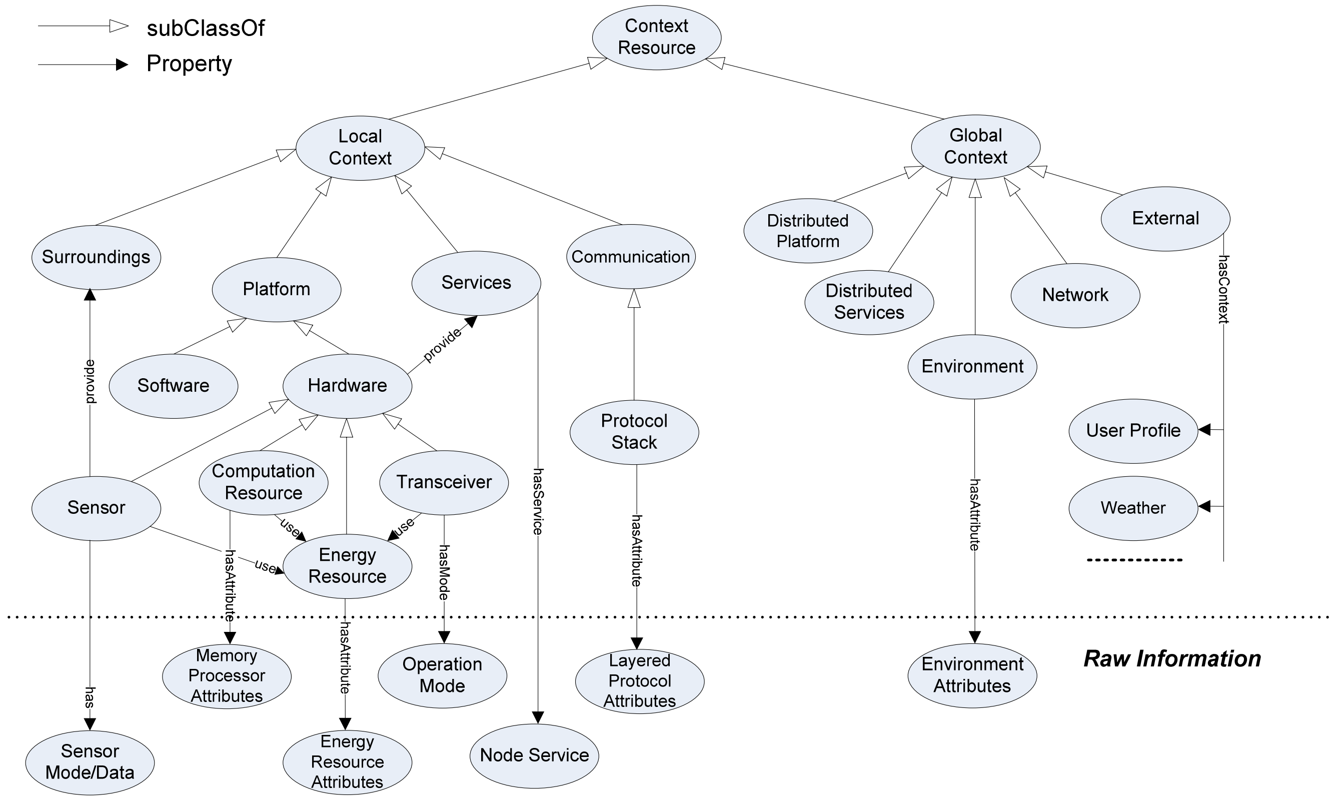
This model is designed as a hierarchical structure of context classes where each class characterizes the contextual information of one or more constituent parts of the IoT system. The bottom level of the model is raw information directly inherited from device components and system entities. The upper levels are the proposed context ontology to model and present inferred contexts for constituent parts of the system.

Here, raw information refers to any information, typically in numeric format, acquired directly from hardware and software components of a single node. In addition, raw information can be exchanged between nodes to infer low level global contexts, e.g., nodes in a given area can exchange raw sensor readings to derive the environmental context of their surroundings.

Normally, low level contexts can be derived by comparing numeric values of raw information with predefined thresholds. As a simple example, the "HIGH"/"LOW" energy state of a node is a low level context derived by comparing the level of residual energy on a node with a threshold value representing half of its full battery capacity. In addition, probabilistic frameworks such as Hidden Markov Models (HMM) [23,24] can be applied when deriving high level contexts, which are not sensed directly but inferred from lower level contexts, and thus have a certain level of uncertainty, depending on the accuracy of the lower level contexts used [16]. For instance, if the derived movement and location of a certain node *A* are believed to be mostly (but not 100%) true, rather than to infer that "*A* is leaving the network", an inference that takes into account of the level of uncertainty such as "With high probability, *A* is leaving the network", can be more appropriate to describe the condition of the event.

The structure of the proposed context ontology model is shown in Figure 3. The *Context Resource* class is the root entity, which has two direct descendant classes: *Local Context* and *Global Context* classes. The following sections describe the proposed model according to their scopes and levels.

Figure 3. Context ontology structure for the IoT systems.



4.1. Local Context

On the left side of this model shows the context ontology of the *Local Context* class, which describes the contextual information of a single node. It is a direct super class to four context classes, namely *Platform*, *Services*, *Surroundings*, and *Communication*.

4.1.1. Platform Context

The *Platform* context class provides a high-level description of a node's platform running state or capability based on context of its constituting hardware and software entities. The platform context can be utilized by a node to self-determine, or by other nodes (if provided with the context) to determine whether if it could undertake a certain role or task, such as serving as a cluster head or performing data aggregation in a WSN.

A. Hardware Context

The *Hardware* context class describes the general resource or performance state of a node's hardware platform, which consists of four hardware components: sensor, transceiver, computation resource and energy resource, each with its own context class.

- *Sensor*: this context class can describe the operation mode of the sensing unit or basic context about its surroundings as deduced from its raw sensor data.
- *Transceiver*: this context class mainly describes the operation mode of the transceiver, e.g., transmit, receive, idle, sleep, or off. The duration and frequency for which the transceiver operates in each mode directly impact the amount of energy that it consumes. Other communication attributes such as channel conditions and bit rate shall be described by the *Communication* context class.
- *Computation Resource*: this context class describes the state of the processing and storage resources, e.g., CPU, memory or buffer storage, of the hardware platform. Such contexts can be particularly useful to support in-network mechanisms such as in-network video processing [12] and data storage [15].
- *Energy Resource*: this context class describes the state of the energy resources of a node, which can be a battery, an energy harvesting device (e.g., solar cell), or other energy module. It is defined to provide energy-related context of a node, such as its residual energy level, energy consumption rate, or the energy generation rate of its harvesting hardware.

B. Software Context

The *Software* context class describes the state of the local OS and programs executing in a node. This can include the program configurations, performance of code executions, and other context that can be useful for WSN management. For instance, the code execution performance of a node, such as the time it takes by a program to process every 100 bytes of inbound data, can be used to determine whether the node can function as a distributed in-network processing node in the WSN.

4.1.2. Services Context

The *Services* context class describes the service roles or tasks that a node can perform. A single node may provide multiple services or carry out multiple tasks at the same time, e.g., a node can be a host to a software agent while acting as a data provider with its built-in sensing component. This node may also function as a cluster head in a hierarchical network, or a relaying node in a multi-hop path. By making the services context available within a node or to other nodes, internal or external functional entities may adapt accordingly to achieve better overall performance. For example, internal functional entities of a node located 1-hop away from a cluster head can adapt to increase the communication capacity in respond to the node's service context of being a frequent relaying node for inbound data to the cluster head.

4.1.3. Surroundings Context

The *Surroundings* context class describes the state of a node's surroundings as monitored by its built-in sensor. It is deduced from a time sequence of local sensor contexts, each of which only represents the state at the time when the raw sensor data is taken. For instance, from 10 consecutive sensor contexts provided by a built-in proximity sensor for human detection, of which eight are 'detected' and two are 'not detected', the context of the node's surrounding area over that time span could be inferred as 'highly active'.

4.1.4. Communication Context

The *Communication* context class provides a high-level description about the state of a node's communication with other nodes. The state can be in terms of the general quality, efficiency, security, frequency, availability, or pattern of communication. It is deduced based on the low level contexts from the node's communication protocol stack.

A. Protocol Stack Context

The *Protocol Stack* context class describes the state of each protocol layer in a node's protocol stack. The physical layer context may express characteristics such as signal quality, channel conditions, interference, and spectrum availability. The medium access control (MAC) layer context may describe the availability, quality, and utilization of the links to the node's direct neighbors, frame collision, and fairness of channel access. The network layer context may capture properties such as the quality, efficiency, and security of a node's multi-hop path to other nodes, traffic distribution pattern, and group membership if the node participates in group communication. The transport layer context may provide knowledge about end-to-end reliability of connection between the node and other nodes, or the occurrence of congestion along its path of communication. Finally, the application layer may present the IoT application's context of use, e.g., involving real-time or non-real time transmission, indoor or outdoor environment, mobile or static scenario, local area or wide area deployment, cooperative or non-cooperative nodes, *etc*. which can be utilized to infer the performance, efficiency, or security requirements of the node's communication for the application.

4.2. Global Context

On the right side of this model shows the context ontology of the *Global Context* class, which describes the contextual information of the IoT system based on local contexts exchanged between nodes and other external contexts. It is a direct super class to five context classes, namely *Distributed Platform*, *Distributed Services*, *Environment*, *Network*, and *External*.

4.2.1. Distributed Platform Context

The *Distributed Platform* context class provides a high-level description of a system's distributed platform running state or capability based on exchanged *Platform* context between nodes in the system. Each node of the distributed platform can adapt to this knowledge to improve the system performance. For example, in a distributed in-network storage system, this context can be used by a node to become aware of and adapt to the storage and computation resource levels of other nodes in order to balance the data storage and processing loads among them.

4.2.2. Distributed Services Context

The *Distributed Services* context class describes the service roles or tasks that can be performed by multiple nodes in a system based on exchanged *Services* contexts. This knowledge can be applied to assist in the selection of nodes to undertake certain networking roles or tasks. For example, in a cluster-based WSN, this context can be used by a departing cluster head to select the best node to take over its role without re-clustering the network.

4.2.3. Environment Context

The *Environment* context class describes the state of a system's physical environment based on exchanged *Surroundings* contexts. It provides nodes with a wider view of the event occurrences in their environment than is possible with only local *Surroundings* context. In turn, nodes can utilize this knowledge to make more informed networking decisions. For example, in an event detection WSN, nodes detecting an event occurrence will transmit data about the event to a sink. By adapting their routing decisions to event contexts, packet congestion in the network can be avoided by routing data through nodes which have not detected any events.

4.2.4. Network Context

The *Network* context class provides a high-level description about the state of an IoT system's network based on exchanged *Communication* contexts as well as contexts from any deployed network management station (NMS) for WSN, e.g., [25]. The heightened awareness of the network state can bring about more effective solutions to address problems, particularly those due to inherent constraints (e.g., resource constraints) and vulnerabilities (e.g., open distributed nature) of WSN. This may consequently give rise to new solutions such as network-state aware resource scheduling or intrusion detection techniques.

4.2.5. External Context

The *External* context class represents any context originated from a source external to the system. This may include user related contexts of IoT applications such as user's profile, preferences, and activity schedule, or contexts derived from weather forecast data, indoor or outdoor map information, which can be useful for WSN management.

5. Scenario Analysis

In this paper, a context-aware multi-path selection (CAMS) algorithm for video streaming in wireless multimedia sensor networks [12] is selected as a use case of our proposed ontology model. In this algorithm, a sensor node can generate video streams of its surrounding environment from its physical onboard sensor components comprising of an image camera and microphone. Thus, each single video stream can be decomposed into two sub-streams—image and audio streams, and transmitted over multiple node-disjoint paths simultaneously. The CAMS algorithm can choose the right number of paths for transmitting each stream so that the overall throughput is maximized. The CAMS prioritises the transmissions and the available routing paths according to the stream content, and end-to-end delay of the path, respectively. The aim is to transmit high-priority content over low delay paths whenever possible.

The original CAMS algorithm does not explicitly consider the issue of heterogeneous nodes. However, in this analysis, we consider a network composed of heterogeneous video sensors of different resolutions. As a result, differences between video sensors in their end-to-end delay requirements can be expected. The end-to-end delay requirement of a high resolution video sensor will be more stringent than a low resolution video sensor as more information bits will be transmitted for a given image or audio frame, *i.e.*, more bits per image pixel or digitized sound sample.

Based on the proposed ontology structure, the context model for CAMS as proposed in [12] is shown on the left side of Figure 4, which involves only local contexts, as the algorithm does not perform any exchange of priority related information. The right side of Figure 4 shows the context model for CAMS that has been extended to utilize global context (to be explained later). The associated syntaxes used are defined in Table 1. The local context resource, CAMS priority, is constituted of *Content priority* and *Delay priority*, which can be seen as corresponding to the *Surroundings* context, and *Communication* context, respectively, of the context ontology structure shown in Figure 3.

Figure 4. Context ontology model for context-aware multi-path selection (CAMS).

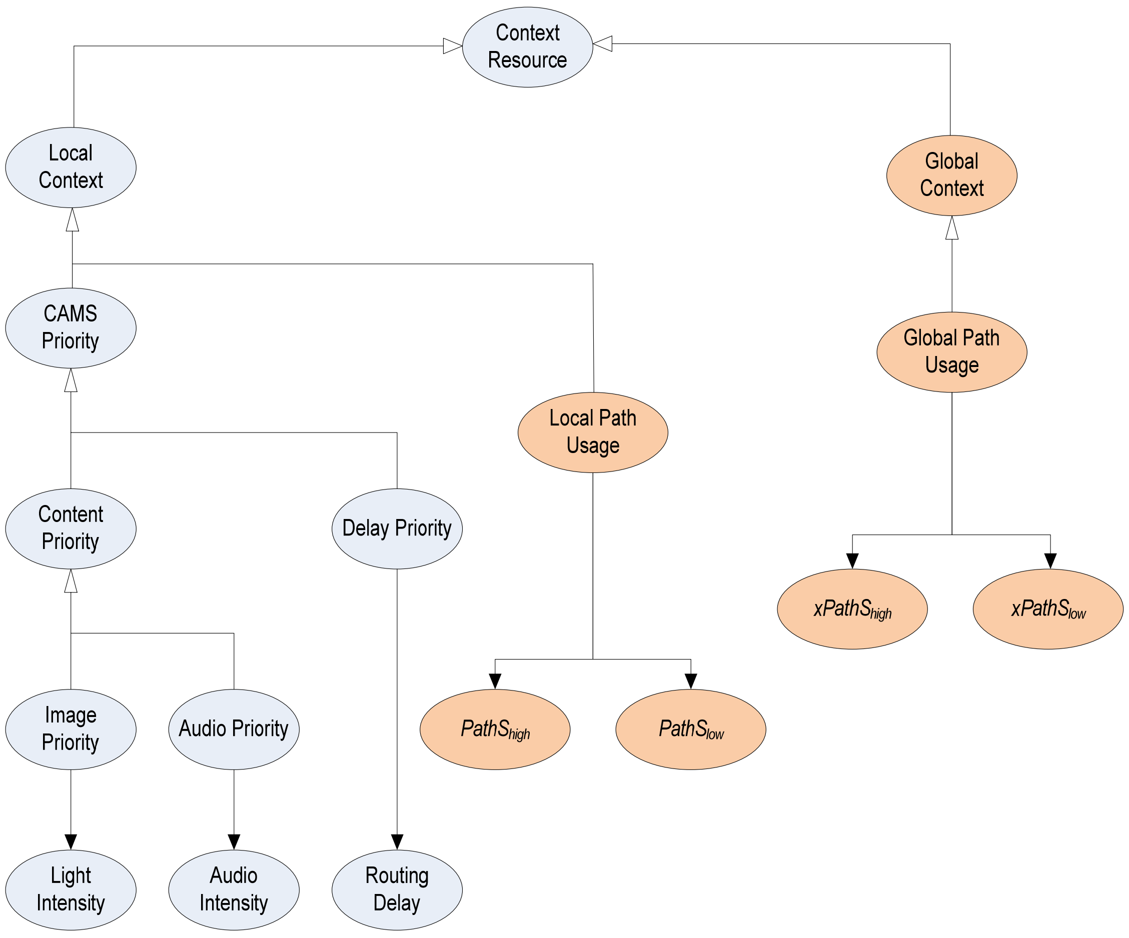


Table 1. Syntax definitions for CAMS algorithm.

Syntax	Definition
BrightnessLevel	Brightness of the split image frame
LoudnessLevel	Loudness of the split audio frame
$I_{brightness}$	Brightness threshold for deciding the frame priority
$I_{loudness}$	Loudness threshold for deciding the frame priority
Path	A single routing path between a source and destination
PathS	Set of *Path* between a source and destination
$Delay_{path}$	End-to-end delay of a path
$DelayS_{path}$	Set of $Delay_{path}$ for each available path in *PathS*
$T_{high\text{-}priority\_max}$	Maximum time for end-to-end transmission of a high-priority frame
$T_{low\text{-}priority\_max}$	Maximum time for end-to-end transmission of a low-priority frame
$PathS_{high}$	Set of available paths for high-priority frame transmission
$PathS_{low}$	Set of available paths for low-priority frame transmission
N	Total number of available paths in *PathS*
M_{high}	Number of paths in $PathS_{high}$
M_{low}	Number of paths in $PathS_{low}$
$xPathS_{high}$	Set of exchanged available paths for high-priority frame transmission
$xPathS_{low}$	Set of exchanged available paths for low-priority frame transmission

In CAMS, a video stream can be presented in Description Logic [26] as:

$$Video \equiv Image \sqcap Audio$$

which expresses that a single video stream is composed of an image stream and an audio stream, each being a sequence of image frames, and audio frames, respectively. A video source node has to decide the priority of each outbound image and audio frame based on their importance. Two qualitative context values for frame importance, *High_Priority* and *Low_Priority*, can be assigned by the video source nodes. A high priority image frame is defined as:

$$High\_Priority.Image \equiv (>I_{brightness}\ BrightnessLevel.Image) \sqcap (\leq I_{loudness}\ LoudnessLevel.Audio \sqcup High\_Priority'.Image)$$

where an image frame is assigned a high priority if its brightness (*BrightnessLevel*) is higher than predefined brightness threshold $I_{brightness}$, and, either the loudness (*LoudnessLevel*) of the associated audio frame is equal or lower than predefined loudness threshold $I_{loudness}$, or the priority of the immediate previous image frame is high as denoted by boolean parameter *High_Priority'.Image*. Similarly, a high priority audio frame is defined as:

$$High\_Priority.Audio \equiv (>I_{loudness}\ LoudnessLevel.Audio) \sqcap (\leq I_{brightness}\ BrightnessLevel.Image \sqcup High\_Priority'.Audio)$$

where *High_Priority'.Audio* is the equivalent boolean parameter denoting the priority of the immediate previous audio frame. Both *High_Priority'.Image* and *High_Priority'.Audio* are initialized to False and updated to True or False according to the respective high or low priority of each transmitted image and audio frame.

Based on the above definitions, only one type of frame—image or audio frame—of the same split video frame can be assigned as high priority, *i.e.*, both image and audio frames cannot be assigned as high priority at the same time. In addition, when both image and audio frames are above (or below) their respective brightness and loudness thresholds, they will inherit the respective priority level assigned to their immediate previous image and audio frames (*High_Priority'.Image, High_Priority'.Audio*) in order to maintain stability of the video streaming as specified in [12].

All available node-disjoint paths between a source-destination pair can also be assigned with a qualitative context based on their transmission latencies. Two qualitative context values used are: *Guaranteed_Trans._Delay* and *Non_Guaranteed_Trans._Delay*. A path is assigned with a non-guaranteed transmission delay context (*Non_Guaranteed_Trans._Delay*) if it neither satisfies the end-to-end delay requirement of the high-priority frame nor low-priority frame:

$$Non\_Guaranteed\_Trans.\_Delay.Path \equiv (>T_{high\text{-}priority\_max}\ Delay_{path}) \sqcap (>T_{low\text{-}priority\_max}\ Delay_{path})$$

If a path satisfies the end-to-end delay requirement of either the high- or low-priority frame, the path is assigned with a guaranteed transmission delay context (*Guaranteed_Trans._Delay*):

$$Guaranteed\_Trans.\_Delay.Path \equiv (\leq T_{high\text{-}priority\_max}\ Delay_{path}) \sqcup (\leq T_{low\text{-}priority\_max}\ Delay_{path})$$

The available routing paths for high-priority frame transmission ($PathS_{high}$) between a source-destination pair is defined as a set of paths in *PathS* whose end-to-end delay is equal or less than the end-to-end delay requirement of high-priority frame ($T_{high\text{-}priority\_max}$):

$$PathS_{high} \equiv PathS \sqcap \forall DelayS_{path}. \leq T_{high\text{-}priority\_max}$$

Similarly, the available routing paths for low-priority frame transmission ($PathS_{low}$) between the same source-destination pair is defined as:

$$PathS_{low} \equiv PathS \sqcap \forall DelayS_{path}. \leq T_{low\text{-}priority\_max}$$

It should be noted that the above $T_{high\text{-}priority\_max}$ and $T_{low\text{-}priority\_max}$ should be appropriately initialized for each node's instance of the ontology model based on its video resolution. This will ensure that all frames are transmitted over paths whose end-to-end delay satisfies the end-to-end delay requirement corresponding to the priority and resolution of the frames.

The CAMS supports multi-path routing, and the relationship between the number of available paths for high-priority frame transmission (M_{high}), low-priority frame transmission (M_{low}), and total number of available paths (N) can be shown as:

$$N \geq M_{high} + M_{low}$$

$$M_{high} < M_{low}$$

which expresses that the number of paths for frame transmission (high and low priority) is bounded by the total number of available paths, and due to the more stringent delay requirement of high priority frame, *i.e.*, $T_{high\text{-}priority\_max} < T_{low\text{-}priority\_max}$, there will be fewer paths in $PathS_{high}$ (M_{high}) for high-priority frame transmission than in $PathS_{low}$ (M_{low}) for low-priority frame transmission.

The original CAMS algorithm is modified to use our proposed context model as discussed above and shown on the left side of Figure 4. The following shows how frames are transmitted by our modified CAMS algorithm under three case scenarios:

```
// Case 1: no transmission if none of the available paths meets the end-to-end delay requirement.
if (∀Non_Guaranteed_Trans._Delay.PathS)
  {No_Transmission}
end if
```

// Case 2: if all available paths meet the end-to-end delay requirement, transmit the high-priority stream simultaneously over the paths in PathS$_{high}$. If there are still unused paths remaining in PathS, transmit the low-priority stream simultaneously over these paths. Otherwise, discard the transmission of the low-priority stream.

```
if (∀Guaranteed_Trans._Delay.PathS)
  {Transmit the High-Priority Stream (sequence of high-priority frames)
    Simultaneously over $M_{high}$ number of paths in $PathS_{high}$
      if ($M_{high} < N$)
        {Transmit the Low-Priority Stream (sequence of low-priority frames)
          Simultaneously over ($N - M_{high}$) number of paths in ($PathS \sqcap (\neg PathS_{high})$)}
    end if
}
end if
```

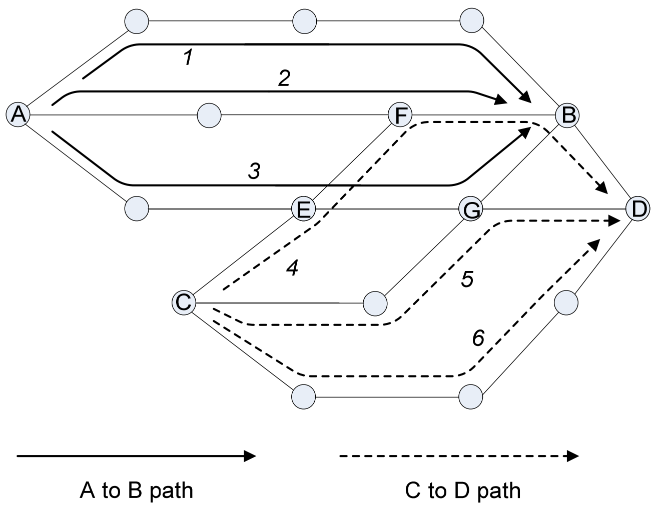
// Case 3: if only a subset of available paths meet the end-to-end delay requirement, transmit the high-priority stream simultaneously over the paths in PathS$_{high}$. If there are still unused paths remaining in (PathS$_{low}$ $\sqcap$ ($\neg$PathS$_{high}$)), transmit the low-priority stream simultaneously over these paths. Otherwise, discard the transmission of the low-priority stream.

```
if (∃Guaranteed_Trans._Delay.PathS)
  {Transmit the High-Priority Stream (sequence of high-priority frames)
    Simultaneously over $M_{high}$ number of paths in $PathS_{high}$
      if ($M_{high} < M_{low}$)
        {Transmit the Low-Priority Stream (sequence of low-priority frames)
          Simultaneously over ($M_{low} - M_{high}$) number of paths in ($PathS_{low} \sqcap$
           $(\neg PathS_{high})$)}
    end if
  }
end if
```

As mentioned earlier, the CAMS as proposed in [12] does not perform any exchange of priority related information, and therefore its selection of node-disjoint paths for frame transmissions is based only on local contexts, *i.e.*, CAMS priority. However, it is conceivable that if individual nodes can be made aware of and adapt their behavior to not only their local context, but also the global context of other nodes, a more coherent and optimal management of the network can be achieved. To illustrate the usage of global contexts, CAMS has been extended for nodes to utilize another type of local context (local path usage), which can be shared or exchanged between nodes as global context. Therefore, a new case scenario has been designed and its corresponding context model is shown on the right side of Figure 4.

The scenario involves multiple pairs of source-destination nodes performing CAMS at the same time. As in previous scenarios, the *PathS* of the source node will hold the available node-disjoint paths to its destination, and these paths can be further placed into set $PathS_{high}$ or $PathS_{low}$ depending on whether they satisfy the delay requirement of the high-priority frame, or low-priority frame, respectively. To motivate the need for an improved CAMS, consider the case where multiple source-destination pairs performed CAMS only according to their local contexts, resulting in some node-disjoint paths between different communicating pairs to become 'node-joint' or overlapped as shown in Figure 5.

Figure 5. Overlapping node-disjoint paths of two communicating pairs.



The common relay nodes *E*, *F*, *G*, and *B* can potentially become traffic bottlenecks for Paths 2 and 3, and Paths 4 and 5 of node pair *A*–*B*, and node pair *C*–*D*, respectively. In order to avoid the occurrence of such situations, nodes can be permitted to exchange context about their available routing paths and the CAMS can be extended to harness the knowledge of such global contexts.

Under the extended CAMS, nodes will behave as follows: After the source node determines its routing paths for high-priority ($PathS_{high}$) and low-priority ($PathS_{low}$) frame transmissions, but before it transmits any frame according to the three case scenarios, the node will share or exchange its local $PathS_{high}$ and $PathS_{low}$ information with other nodes, received upon which will be stored as $xPathS_{high}$ (exchanged $PathS_{high}$) and $xPathS_{low}$ (exchanged $PathS_{low}$). On receiving, the node can determine if any of its paths in $PathS_{high}$ and $PathS_{low}$ are 'node-joint' or overlapped with those in $xPathS_{high}$ and $xPathS_{low}$.

As shown on the right side of Figure 4, a new local context class *Local Path Usage* (constituted of local $PathS_{high}$ and $PathS_{low}$) and a global context class *Global Path Usage* (constituted of $xPathS_{high}$ and $xPathS_{low}$) have been introduced, which can be seen as corresponding to the *Communication* context, and *Network* context, respectively, in the context ontology structure shown in Figure 3.

An available routing path cannot be categorized into $PathS_{high}$ and $PathS_{low}$ at the same time. On the other hand, the same routing path may not be categorized into either $PathS_{high}$ or $PathS_{low}$ if it does not satisfy the delay requirement of either high-priority or low-priority frame. Each *Path* in $PathS_{high}$ and $PathS_{low}$ can be assigned with one of the following two qualitative context values:

$$RelayNodeShared.Path = \exists\ xPathS_{high}{:}hasRelayNode(\exists Nodeof(Path)) \sqcup \\ \exists\ xPathS_{low}{:}hasRelayNode(\exists Nodeof(Path))$$

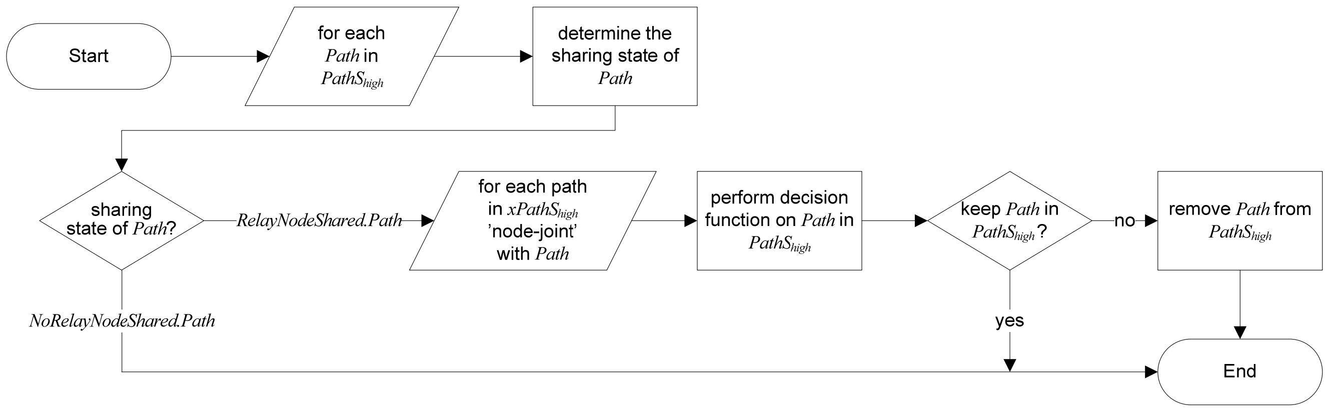
$$NoRelayNodeShared.Path = \exists\ xPathS_{high}{:}hasNoRelayNode(\exists Nodeof(Path)) \sqcap \exists\ xPathS_{low}{:}hasNoRelayNode(\exists Nodeof(Path))$$

which expresses that if any relay node of a *Path* in *PathS_{high}* and *PathS_{low}* is also a node of a path (e.g., source, destination, or relay node) in *xPathS_{high}* and *xPathS_{low}*, this *Path* will be assigned with the state *RelayNodeShared*, otherwise it will be assigned with the state *NoRelayNodeShared*.

For a path 'marked' as having one or more shared nodes, the source node may perform a decision function to determine whether or not it should keep this path for frame transmission. The design of the decision function is often application/scenario specific, which may be based on probabilistic models, fuzzy logic, decision trees, or other reasoning mechanisms.

Figure 6 shows the flow of steps to handle shared paths in CAMS with global context. For each *Path* in $PathS_{high}$ 'node-joint' with other paths in $xPathS_{high}$, the node will perform a decision function to decide whether or not it should keep this *Path* locally in its $PathS_{high}$. The same procedure is applied for each *Path* in $PathS_{low}$ 'node-joint' with other paths in $xPathS_{low}$. However, if a *Path* in $PathS_{low}$ is 'node-joint' with paths in $xPathS_{high}$, this *Path* will be removed from $PathS_{low}$ of this node. In other words, priority for using this Path is given to nodes that will be using it for high-priority frame transmission, *i.e.*, as *Path* in $PathS_{high}$. This step will be taken as well by other nodes with *Path* in their $PathS_{low}$ 'node-joint' with paths in their $xPathS_{high}$. On the other hand, if a *Path* in $PathS_{high}$ is a 'node-joint' with any paths in $xPathS_{low}$, the node will keep this *Path* in its $PathS_{high}$.

Figure 6. Handling of shared paths in CAMS with global context.


Start
for each Path in PathS_high
determine the sharing state of Path
sharing state of Path?
RelayNodeShared.Path
for each path in xPathS_high 'node-joint' with Path
perform decision function on Path in PathS_high
keep Path in PathS_high?
no
remove Path from PathS_high
NoRelayNodeShared.Path
yes
End

As a formalism for ontology representation, the RDF/XML serialization of the proposed ontology in Figure 4 is shown. However, XML is seen as a 'heavy' syntax for resource-constrained devices. Thus, for implementing the proposed ontology on sensor nodes, more compact XML representations such as binary XML formats should be used [27]. Another promising approach uses streaming HDT as lightweight serialization format for RDF and Wiselib Tuplestore for storing RDF data locally on embedded IoT devices such as sensor nodes is proposed in [28].

```
<owl:Class rdf:ID="CAMSPriority">
 <owl:Class rdf:ID="LocalContext">
  <owl:Class rdf:ID="ContentPriority">
   <rdfs:subClassOf rdf:resource="#LocalContext"/>
```

```
</owl:Class>
 <owl:DatatypeProperty rdf:ID="PriorityState" >
  <rdfs:domain rdf:resource=" #ContentPriority"/>
  <rdfs:range rdf:resource="xsd:string"/>
 </owl: DatatypeProperty >
 <owl:ObjectProperty rdf:ID="ImagePriorityState">
  <rdfs:domain rdf:resource="#ContentPriority"/>
  <rdf:range rdf:resource="#PriorityState"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="AudioPriorityState">
  <rdfs:domain rdf:resource="#ContentPriority"/>
  <rdf:range rdf:resource="#PriorityState"/>
 </owl:ObjectProperty>
 <owl:Class rdf:ID=" DelayPriority">
  <rdfs:subClassOf rdf:resource="#LocalContext"/ >
</owl:Class>
 <owl:ObjectProperty rdf:ID="RoutingDelay">
  <rdfs:domain rdf:resource="#DelayPriority">
  <rdf:range rdf:resource="xsd:double">
 </owl:ObjectProperty>
 <owl:Class rdf:ID=" LocalPathUsage">
  <rdfs:subClassOf rdf:resource="#LocalContext"/ >
</owl:Class>
 <owl:ObjectProperty rdf:ID="$PathS_{high}$">
  <rdfs:domain rdf:resource="#LocalPathUsage">
  <rdf:range rdf:resource="xsd:string">
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="$PathS_{low}$">
  <rdfs:domain rdf:resource="#LocalPathUsage">
  <rdf:range rdf:resource="xsd:string">
 </owl:ObjectProperty>
</owl:Class>
 <owl:Class rdf:ID="GlobalContext">
  <owl:Class rdf:ID="GlobalPathUsage">
   <rdfs:subClassOf rdf:resource="#GlobalContext"/ >
</owl:Class>
 <owl:ObjectProperty rdf:ID="$xPathS_{high}$">
  <rdfs:domain rdf:resource="# GlobalPathUsage">
  <rdf:range rdf:resource="xsd:string">
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID=" $xPathS_{low}$">
  <rdfs:domain rdf:resource="# GlobalPathUsage">
```

```
        <rdf:range rdf:resource="xsd:string">
     </owl:ObjectProperty>
    </owl:Class>
  </owl>
```

The above use cases have illustrated how the proposed ontology-based context model can be applied to contextualize the mostly numeric data in the network, and how the contextualized data can be used beyond their sources by facilitating context sharing between network entities, all with the goal of enabling context-aware management of WSNs, which can also harness the rich context knowledge of the IoT systems. In comparison with the original CAMS, the 'contextualized' CAMS presented in this paper, *i.e.*, CAMS using the proposed context ontology model, is more prepared to perform in the IoT environment, since all network entities share a common understanding of the network related information originated from heterogeneous sources, but contextualized using the same proposed model for a unified unambiguous interpretation. As mentioned, while there exist context ontology models for mitigating the complexity of systems operating in heterogeneous environments, most if not all of them are focused on modeling system or application level contexts, with very few or no ontology models proposed for network level contexts. Hence, to the best of our knowledge, the proposed model in this paper is one of the first (if not the first) for WSN management in IoT.

6. Conclusion

This paper proposed an ontology based context model for context aware WSN management in the IoT. Unlike previous models that focus mainly on contexts at application and service levels, an ontology model is proposed for the first time that focuses on representing network related situations of an IoT system as contexts which can be shared, understood, and utilized by heterogeneous nodes for context aware management of the underlying WSNs. Under the proposed model, context knowledge from system, network and node levels can be classified as local or global, and high or low level contexts according to their scope and level of aggregation. A use case of the proposed model is presented in which a context ontology is designed for and applied to an existing context-aware algorithm for cross-layer optimization in WSN.

The designed ontology models the contextual information found in common WSN scenarios, and makes them shareable between different parts of a system. Hence, a potential direction for future work is to further investigate the applications of the proposed model in particular for context-aware WSN algorithms or mechanisms that wish to harness the rich context knowledge of the IoT systems. Another promising direction is to deal with the practical challenges of realizing a small-footprint ontology store or database management system that can perform context storage and querying efficiently under the constraints of limited bandwidth, memory and processing resources of sensor devices. Furthermore, given the heterogeneous nature of IoT, multiple context ontologies may co-exist in the same system. Thus, ontology merging or alignment represents another interesting dimension for future investigation.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Rajan, M.A.; Balamuralidhar, P.; Chethan, K.P.; Swarnahpriyaah, M. A Self-Reconfigurable Sensor Network Management System for Internet of Things Paradigm. In Proceedings of the International Conference on Devices and Communications (ICDeCom), Mesra, Ranchi, India, 24–25 February 2011.
2. Wang, Q.; Jäntti, R.; Ali, Y. On Network Management for the Internet of Things. In Proceedings of the 8th Swedish National Computer Networking Workshop (SNCNW), Stockholm, Sweden, 7–8 June 2012.
3. Hu, P.; Portmann, M.; Robinson, R.; Indulska, J. Context-Aware Routing in Wireless Mesh Networks. In Proceedings of the 2nd ACM International Conference on Context-Awareness for Self-Managing Systems (CASEMANS), Sydney, Australia, 19 May 2008.
4. Oteafy, S.; Hassanein, H.S. Resource re-use in wireless sensor networks: Realizing a synergetic internet of things. *J. Commun*. **2012**, *7*, 484–493.
5. Zhang, J.; Sun, Y. Managing Resources in Internet of Things with Semantic Hyper-Network Model. In Proceedings of the IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Toulouse, France, 25–27 June 2012.
6. Dey, A.K. Understanding and using context. *Pers. Ubiquitous Comput*. **2001**, *5*, 4–7.
7. Kim, K.T.; Choi, W.J.; Youn, H.Y. CA-MAC: Context Adaptive MAC Protocol for Wireless Sensor Networks. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Budapest, Hungary, 29–31 April 2009.
8. Zhou, H.; Hou, K. CIVIC: An Power- and Context-Aware Routing Protocol for Wireless Sensor Networks. In Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), Shanghai, China, 21–25 September 2007.
9. Koo, B.; Won, J.; Park, S.; Eom, H. PAAR: A Routing Protocol for Context-Aware Services in Wireless Sensor-Actuator Networks. In Proceedings of the First Asian Himalayas International Conference on Internet, Kathmandu, Nepal, 3–5 November 2009.
10. Haque, M.E.; Matsumoto, N.; Yoshida, N. Utilizing multilayer hierarchical structure in context aware routing protocol for wireless sensor networks. *Int. J. Comput. Sci*. **2010**, *4*, 23–37.
11. ElGammal, M.; Eltoweissy, M. Distributed Context-Aware Affinity Propagation Clustering in Wireless Sensor Networks. In Proceedings of the 6th International Conference on Collaborative Computing (CollaborateCom), Chicago, IL, USA, 9–12 October 2010.
12. Shu, L.; Zhang, Y.; Yu, Z.; Yang, L.T.; Hauswirth, M.; Xiong, N. Context-aware cross-layer optimized video streaming in wireless multimedia sensor networks. *J. Supercomput*. **2010**, *54*, 94–121.
13. Wang, C.; Liu, J.; Kuang, J.; Xiang, H. A Context-Aware Architecture for Wireless Sensor Networks. In *Recent Advances in Computer Science and Information Engineering*; Qian, Z., Cao, L., Su, W., Wang, T., Yang, H., Eds.; Springer: Heidelberg, Germany, 2012; pp. 199–205.
14. Gladisch, A.; Daher, R.; Lehsten, P.; Tavangarian, D. Context-Aware Energy Management for Energy-Self-Sufficient Network Nodes in Wireless Mesh Networks. In Proceedings of the 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Budapest, Hungary, 5–7 October 2011.

15. Kim, H.; Park, J.; Seong, D.; Yoo, J. A Context Aware Data-Centric Storage Scheme in Wireless Sensor Networks. In *Multimedia, Computer Graphics and Broadcasting*; Kim, T., Adeli, H., Grosky, W.I., Pissinou, N., Shih, T.K., Rothwell, E.J., Kang, B.-H., Shin, S.-J., Eds.; Springer: Berlin, Germany, 2011; pp. 326–330.
16. Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A.; Riboni, D. A survey of context modelling and reasoning techniques. *Pervas. Mobile Comput.* **2010**, *6*, 161–180.
17. Strang, T.; Linnhoff-Popien, C. A Context Modeling Survey. In Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp), Nottingham, UK, 7 September 2004.
18. Serrano, J.M.; Serrat, J.; Strassner, J. Ontology-Based Reasoning for Supporting Context-Aware Services on Autonomic Networks. In Proceedings of the IEEE International Conference on Communications (ICC), Glasgow, UK, 24–28 June 2007.
19. Coronato, A.; de Pietro, G.; Esposito, M. A Semantic Context Service for Smart Offices. In Proceedings of International Conference on Hybrid Information Technology (ICHIT), Cheju Island, Korea, 9–11 November 2006.
20. Lee, K.W.; Cha, S.H. Ontology-Based Context-Aware Management for Wireless Sensor Networks. In *Advances in Computer Science, Environment, Ecoinformatics, and Education*; Lin, S., Huang, X., Eds.; Springer: Berlin, Germany, 2011; pp. 353–358.
21. Verstichel, S.; Volckaert, B.; Dhoedt, B.; Demeester, P.; de Turck, F. Context-aware scheduling of distributed DL-reasoning tasks in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2011**, *2011*, 1–24.
22. Anastasi, G.; Conti, M.; di Francesco, M.; Passarella, A. How to Prolong the Lifetime of Wireless Sensor Networks. In *Handbook on Mobile Ad Hoc and Pervasive Communications*; Denko, M., Yang, L.T., Eds.; American Scientific Publishers: Valencia, CA, USA, 2006; pp. 1–26.
23. Hu, P.; Zhou, Z.; Liu, Q.; Li, F. The HMM-Based Modeling for the Energy Level Prediction in Wireless Sensor Networks. In Proceedings of the IEEE 2nd Conference on Industrial Electronics and Applications (ICIEA), Harbin, China, 23–25 May 2007.
24. Goudarzi, R.; Jedari, B.; Sabaei, M. An Efficient Clustering Algorithm Using Evolutionary HMM in Wireless Sensor Networks. In Proceedings of the IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), Melbourne, Australia, 11–13 December 2010.
25. Shanthini, J.; Vijayakumar, S. Modified simple network management protocol for 6Lowpan. *Proced. Eng.* **2012**, *38*, 1024–1029.
26. *The Description Logic Handbook: Theory, Implementation and Applications*; Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., Eds.; Cambridge University Press: Cambridge, UK, 2003.
27. Iglesias, J.; Bernardos, A.M.; Tarrio, P.; Casar, J.R.; Martin, H. Design and validation of a light inference system to support embedded context reasoning. *Pers. Ubiquitous Comput.* **2012**, *16*, 781–797.

28. Hasemann, H.; Kroller, A.; Pagel, M. RDF Provisioning for the Internet of Things. In Proceedings of the 3rd International Conference on the Internet of Things (IoT), Wuxi, China, 24–26 October 2012.

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).