2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing

Event Handling for Distributed Real-Time Cyber-Physical Systems

Jaipal Singh[1], Omar Hussain[2], Elizabeth Chang[2]
[1]Dept. of Electrical and Computer Engineering
[2]Dept. of Information Systems
Curtin University
Perth, Australia
e-mail: j.singh@curtin.edu.au, {o.hussain, Elizabeth chang}@cbs.curtin.edu.au

Tharam Dillon
Digital Ecosystems and Business Intelligence Institute
Australia
www.debii.org
e-mail: tharam.dillon7@gmail.com

***Abstract*—Cyber-Physical Systems (CPS) provides a smart infrastructure connecting abstract computational artifacts with the physical world. This paper presents some challenges for developing distributed real-time Cyber-Physical Systems. The focus is on one particular challenge, namely event modelling in distributed real-time CPS. A Web-of-Things based CPS framework for event handling and processing is proposed. To illustrate the application of the proposed framework, a case study for achieving demand response in a smart home is provided.**

Keywords-cyber-physical systems; event models; distributed real-time computing.

I. INTRODUCTION

Real-time systems are defined as a system whose response time is an important determinant of correct functioning [2]. They enable the timely delivery of services. As task complexity increase, real-time systems have become distributed. The basic features by which a real-time system aims to meet the timely delivery of services are: (a) timeliness, (b) availability of the required resources, (c) reactiviness to different events, (d) concurrent execution capability, (e) dependability, and (f) defensive approach to event handling. By using these features, real-time systems have been applied in various applications, such as car engine control systems and pacemakers for heart patients.

However, real-time systems traditionally have been isolated and have been working on closed networks. The recent growth of embedded technologies and Internet/web usage in day-to-day activities requires a change on how real-time systems are connected to users and other systems. These days, real-time systems must be connected to the Internet and use IP for communication and control. This will make it easier to deploy, control and monitor real-time systems in remote locations such as offshore oil rigs and mining sites.

Cyber-Physical Systems (CPS) is a new research field that seeks to integrate embedded real-time systems into the Internet. However, CPS goes beyond traditional embedded and distributed systems. CPSs are often long lasting, with 24x7 operation and must evolve without losing stability [3].

Cyber-Physical Systems (CPSs) are integrations of computation, communication, and control with the physical world [4]. CPS is envisioned to be a heterogeneous system of systems, consisting of computing devices, embedded systems, sensors and actuators. All of these components are inter-connected allowing them to execute tasks that will link the cyber world and the physical world [5].

Our definition of Cyber-Physical Systems (CPS) is a system that interconnects the physical world with the cyber world, allowing for robust and flexible systems with multi-scale dynamics and integrated wired and wireless networking for managing the flows of mass, energy, and information in a coherent way through integration of computing and communication capabilities with the monitoring and/or control of entities in the physical world in a dependable, safe, secure, efficient and real-time fashion.

As the cyber systems and physical systems are largely isolated from each other and investigated from different domains, there is very little theoretical knowledge on developing real-time systems and real-time Cyber-Physical Systems. As CPS interact with the real-world in real-time and utilise real energy, the current approaches used in distributed real-time systems might not be suitable to be applied in this domain. Several approaches try to develop a new model for CPS but usually have success in modelling either the physical phenomenon [6] or the cyber entities [7]. No approach has successfully developed a comprehensive distributed real-time Cyber-Physical System that takes into account both cyber and physical aspects of the systems.

This paper aims to investigate the challenges in developing a distributed real-time Cyber-Physical System and propose a Web-of-Things-based framework to address this. This paper is organised as follows. In section II, we list the challenges in developing distributed real-time Cyber-Physical Systems. We provide information about event models for distributed real-time Cyber-Physical Systems in section III. Section IV details our proposed framework. In section V, we present a case study to illustrate the mechanics of the proposed framework. We conclude this paper in section VI.

II. CHALLENGES IN DEVELOPING DISTRIBUTED REAL-TIME CYBER-PHYSICAL SYSTEMS

Before we list the challenges in developing distributed real-time CPS, we first look at the challenges in developing traditional distributed real-time systems. A real time system consists of many devices that are built for a special purpose [8]. In an open network environment, communication between various heterogeneous distributed systems is critical. The challenge for interoperability becomes greater when a wide range of tasks must be handled by the application including safety critical and on-demand requests [9]. These challenges are described as follows [8]:

- The synchronization and concurrency of parallel activities of active objects in a textual representation.
- The diversity and rapid growth of amount in data.
- To discover the design anomalies particularly with the respect to timeliness and dependability.
- The temporal correctness of activities in a schedule.
- Many middleware solutions for distributed application integration are targeted at the typical business model, complex services like real time systems with the support of QoS is not generally available.

In the next subsections, we will summarise some of the challenges in developing real-time Cyber-Physical Systems.

A. *Development of real-time Cyber-Physical Systems*

The diversity in system architecture and component has resulted in heterogeneity. The key of success for the collaboration between distributed computing architecture relies on consistent definition of interface and data definition/access. It is important to provide a consistent standard to ensure a smooth implementation of service collaboration, since data contains the abstract from the logic of the domain [10]. Some important features that characterise information dissemination in distributed systems are:

1. Collection and transmission of data from sensors to data concentrators. An example would be the implementation of communication between segments of networks with the use of management agent, where information is exchanged by agent on behalf of network stations.
2. Storage of data at several levels and different databases that must interoperate with each other. For example, some advanced systems which run in power plants requires the integration of real time data with historic data to make intelligent analysis and allow the system to run predictably.

With the development of Cyber-Physical Systems, more sensors and actuators are being connected to the Internet. However, in order to realise the benefit from them, it is not enough to just connect them. We need appropriate techniques by which these sensors and actuators can be found and invoked remotely. For that we need standardised interfaces that assist in maintaining interoperability of data between various real-time CPSs. This needs the following consideration:

- A format that is widely accepted, platform independent which facilitate data exchange among different data sources.
- A suitable mechanism for storing 'annotated' data that has clearly defined semantics.
- Locating these sensors on the web and invoking them to provide more complex information and functionality.
- Defining an architectural framework for the distributed real-time CPS.
- Abstractions for sensors/actuators and event representation.
- Composition of sensors/actuators and events to address complex situations.
- Semantics for sensors/actuators and events.

Based on these challenges, this paper will focus on how events are processed in a distributed real-time Cyber-Physical System. In the next section, we discuss some of the key challenges to be addressed in event representation in distributed real-time CPS.

III. Event Representation in Distributed Real-Time CPS

Events are one of the key initiators of activities in Cyber-Physical Systems. They provide a natural way to specify components of open systems in terms of interfaces and observable behaviour. Events also can form the basis for specifying coordination and composition of components [3].

Due to the wide variety of events ranging from lower-level physical signals to higher-level abstract events, CPS applications and users may be interested in certain conditions of interest in the physical world, according to which certain predefined operations are executed by the CPS. Detection of these conditions of interest (events) will lead to the desired predefined operations. As a result, any CPS task can be represented as an "Event-Action" relation [5].

Some of the challenges involved in the representation of events in dynamic real-time CPS are [11]:

A. *Abstractions for sensors and event representations*

Sensor data requires processing to generate information which can then be used for event identification and representation. Abstraction of sensor instances and event instances are required for representing sensors and events respectively. However, this is a challenging task that requires a framework that can deal with this problem in its entirety. Sensors can be represented as a resource with a unique identifier that allows them to be located through the Internet. In addition, it is important to develop the notion of

- Sensors classes which represent collections of sensors instances with particular properties. e.g. a temperature sensors monitor and alarm class. This class could consist of several instances. To allow for mobility, this class would have both location and time as two amongst other properties.
- Event classes which represent collections of event instances with particular properties. e.g. an intrusion event detection class which could consist of several event instances. Event classes too would have location and time (or at least position in a sequence in time) as two amongst other properties.

There will be relationships between these classes, which will allow for representation of generalization/specialization, composition and associations. As mentioned in section II, interoperability is an important challenge to be considered for the seamless integration of different high level applications. Successful implementation of this allows us to meet the following challenges:

- Management of raw sensor data that is kept, maintained and exported by disparate sources.
- Interpreting events associated with particular sensor configurations and outputs.
- Transforming system level knowledge from distinct sensors into higher-level management applications.
- Upgrade of existing sensors with new advanced sensors using standardised interfaces by using the same abstract level representation.

B. Compositions of sensors and events to address complex requirements

Depending upon specific application requirements, composition of event data from multiple sensors may be required. This is a challenging task that requires a dedicated framework on how information from multiple sensors is composed and correlated for meeting the QoS requirements of the specific application scenario.

A decomposition technique is needed for decomposing complex functionality into lower level resources which can then be met by specific sensors or events.

This decomposition requires the specification of items in the aggregation and their dynamic sequence of execution or arrangement. The dynamics in the case of services can be modelled by using workflows specified in a language such as BPEL and in the case of resources by using Mashups which allow easier configuration of these workflows.

The composition of events remains a challenging issue with a focus on sequencing them to produce a composite event.

C. Semantics for compositions of sensors and events

Semantics need to be provided for automatic sensor discovery, selection and composition. Semantics here is the study of the meaning of Things (resources that represent sensors) and events. It represents the definitions of the meaning of elements, as opposed to the rules for encoding or representation. Semantics describe the relationship between the syntactical elements and the model of computation. It is used to define what entities mean with respect to their roles in a system. This includes capabilities, or features that are available within a system.

Two approaches can be used to represent semantics. They are (1) ontologies and (2) lightweight semantic annotations. Ontologies are a formal, explicit specifications of a shared semantic conceptualization that are machine-understandable and abstract models of consensual knowledge, can be used. Using an ontology, it is possible to define concepts through uniquely identifying their specifications and their dynamic and static properties. Concepts, their details and their interconnections are defined as an ontology specification. Ontology compositions are typically formed from many interconnected ontology artifacts that are constructed in an iteratively layered and hierarchical manner. It is necessary to use a graphical representation of ontologies at a higher level of abstraction to make it easier for the domain experts to capture the semantic richness of the defined ontology and understand and critique each model.

Ontologies can be used to define the following system properties:

- **Information and Communication** - refers to the basic ability to gather and exchange information between the parties involved.
- **Integrability** - relates to the ability of sensors and devices from different sources to mesh relatively seamlessly and the ability to integrate these sensors to build the final solution in a straightforward way.
- **Coordination** - focuses on scheduling and ordering tasks performed by these parties.
- **Awareness and Cooperation** - refer to the implicit knowledge of the operation process that is being performed and the state of the system.

For representing the knowledge in a given domain ontologies and for adding semantics to individual resources annotation may suffice. A special challenge here is developing ontologies for events.

D. Event Models for CPS

Approaches have been proposed in the literature that model events in CPS. They classify events as temporal or spatial [5]. They further define different dimensions such as (a) punctual or interval, (b) single or stream, (c) action or observation, (d) point or field, and (e) causal, discrete, or continuous [3, 5].

In a CPS, the events can be further categorised as follows [5]:

- Physical-events: Physical event models the occurrence of the end-user interest in the physical world and can be any change in attribute, temporal or spatial status of one or more physical objects or physical phenomena. These events are captured through physical observation, which is a snapshot of attribute, temporal, or spatial status of the target physical event.
- Cyber-Physical events: The physical event captured using sensors collect the sensor event instances from other sensor motes as input observations and generate cyber-physical event instances based on the cyber-physical event conditions.
- Cyber-events: The top level CPS control unit serves as the highest level of observer in CPS event model. It may combine cyber-physical event instances from other CPS components (sensors) and other control units as input observations to generate the cyber event.

Some of these approaches utilise a Spatial-Temporal event models for CPS in 2-dimensions [5] and in another uses 3-dimensions [12]. Another approach provides

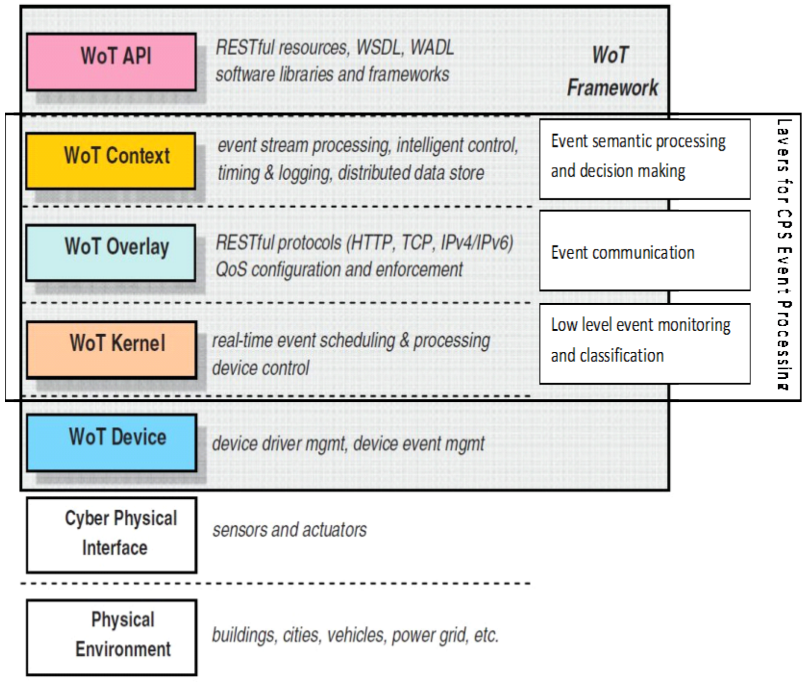
WoT API
RESTful resources, WSDL, WADL
software libraries and frameworks
WoT Framework
WoT Context
event stream processing, intelligent control, timing & logging, distributed data store
Event semantic processing and decision making
WoT Overlay
RESTful protocols (HTTP, TCP, IPv4/IPv6)
QoS configuration and enforcement
Event communication
WoT Kernel
real-time event scheduling & processing
device control
Low level event monitoring and classification
Layers for CPS Event Processing
WoT Device
device driver mgmt, device event mgmt
Cyber Physical Interface
sensors and actuators
Physical Environment
buildings, cities, vehicles, power grid, etc.

Figure 1. Layers for event processing in Web-of-Things Cyber-Physical Systems framework

semantics to the events detected by using first order logic, such as an adaptive discrete event calculus [13].

In the next section, we propose a Web-of-Things based CPS framework for event processing.

IV. Event Processing using Web-of-Things Framework for Cyber-Physical System

In our previous work, we presented a Web-of-Things (WoT) conceptual framework and reference architecture specifically for CPS. For the sake of completeness we briefly discuss the framework in this section. For further details, readers are encouraged to refer to [1].

The proposed framework not only integrates the cyber world and physical work through the web, but it also caters for bi-directional, real-time processing between the cyber world and the physical world with end-to-end QoS determinism and predictability. This section will propose an architecture for event handling and processing for CPS.

As shown in Fig. 1, the proposed WoT based CPS framework consists of five layers:

1. WoT Device: This layer constitute the cyber-physical interface of the system. It is a resource-oriented abstraction that unifies the management of various devices. It states the device semantics in terms of RESTful protocol.
2. WoT Kernel: This layer provides low level run-time for communication, scheduling, and WoT resources management. It identifies events and allocates the required resources, i.e. network bandwidth, processing power and storage capacity for dealing with a large amount of data from the WoT Device layer.
3. WoT Overlay: This layer is an application-driven, network-aware logical abstraction atop the current Internet infrastructure. It will manage volatile network behavior such as latency, data loss, jitter and bandwidth by allowing nodes to select paths with better and more predictable performance.
4. WoT Context: This layer provides semantics for events captured by the lower layers of WoT framework. This layer is also responsible for decision making and controlling the behaviour of the CPS applications.
5. WoT API: This layer provides abstraction in the form of interfaces that allow application developers to interact with the WoT framework.

Fig. 2 represents the CPS reference architecture by using the proposed WoT CPS framework. It is comprised of a CPS Fabric and CPS Nodes, each of which implement 3 layers of the WoT CPS framework shown in Fig. 1.

A CPS Node is a WoT framework embedded computing unit that is connected to one or more devices, i.e. sensors and actuators, through the Device layer. Physical events will be detected by the CPS Node where it will be classified and prioritised at the Kernel layer. Each device collects specific information from a given physical environment and sends them back to the CPS Node. The CPS Node may transform this information into CPS events processed at the WoT Kernel layer, which may then liaise with other CPS Nodes through the CPS Fabric. These events will be communicated to the CPS Fabric through the Overlay layer for further event processing.

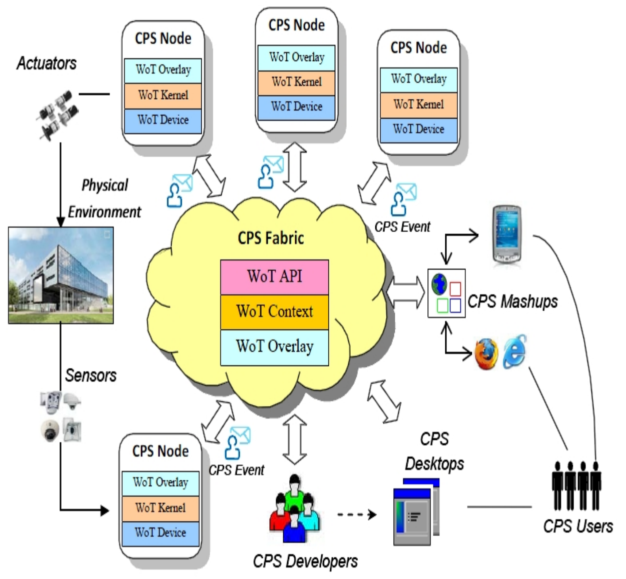

Figure 2. WoT CPS Reference Architecture (reproduced from [1])

The CPS Fabric consists of in-network system functions to facilitate communication between the different CPS nodes. It uses the WoT Context layer to (1) derive passive context from processing event streams and (2) perform intelligent control based on active context. Users and applications use WoT API to register application-specific contextual information, infuse additional data from external sources, and integrate CPS Fabric with existing decision-making modules and other related computational functions [1].

The event modelling is implemented at the WoT Context layer. CPS applications needs to handle two types of event or contexts - active context that controls the behaviour of the applications and passive context that provides meaning to applications and/or users [1].

The **passive context** provides meaning for users to carry out reasoning, diagnosis and decision making from event data. Techniques will be developed to restore, discover and construct contextual information from event stream data sent by the WoT Overlay layer. This will be done by processing large scale complex event streams with stringent timeliness requirements. Large sets of discrete event instances will be organised into a number of contiguous regions bounded by temporal, spatial, and other semantic constraints (e.g. ordering, negation, etc.) so that sequences of occurring events will be handled as a whole rather than individual events. CPS applications will use a declarative language (e.g. RDF) to express its requirements as context discovery statements that will be continuously evaluated by the WoT Context layer against event streams. This approach shields high level applications from dealing with low-level event processing such as filtering, correlating, transformation, indexing, querying, and composition.

The **active context** (e.g. mission-related messages, environmental configuration, parameter settings, etc.) is a mediator to generate tasks (e.g. inbound events) which then computes control signals to CPS devices. CPS applications will use a declarative language combined with context models (e.g. RDF triples) as context execution statements. The proposed framework will provide active context handling through 5 phases - context scanning, context recognition, context enforcement, task selection and task validation [1]. The developed active context modelling will cater for distributed context handling through decomposition, particularly for geographically distributed CPS devices. The developed model will handle both human user produced active context as well as active context caused by changes to the physical environment (states). For state changes, the model will transform the passive context derived from the underlying event stream into active context in order to influence the CPS application.

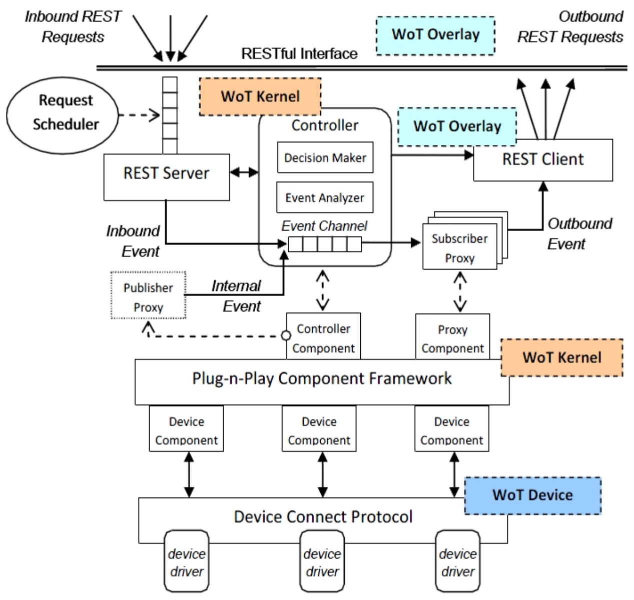

Figure 3. Internal architecture of a CPS Node (reproduced from [1])

The internal architecture of a CPS node is shown in Fig. 3. As mentioned earlier, the WoT Kernel layer is used to process events. It is comprised of five key architectural components:

- OSGi Component Framework: A loose-coupled component framework with certain fault-tolerance characteristics comprising a Device Component (DC) for directly invoking the Controller Component (CC) for publishing an Internal Event and the CC for calling DCs's component interfaces with detailed control information for implementation of the physical device through the device driver.
- Controller: Component for real-time event collection processing (such as filtering, correlating, and aggregating), scheduling (reordering, prioritizing, etc.), and dispatching. It can be configured to support different scheduling and dispatching strategies based on the CPS application requirements. This component will

analyse the events (Internal Event and Inbound external Events) with diversified end-to-end QoS requirements and make controlling decisions.

- REST Server: Accepts incoming messages from other CPS Nodes (event messages) or from the CPS fabric (control messages to initiate action).
- REST Client: Used by the controller component to directly send a notification or a state inquiry message to other CPS nodes in response to an urgent event.
- Request Scheduler: Used for scheduling incoming event messages based on level of importance (e.g. mission critical real-time events will be forwarded in the queue before other messages).

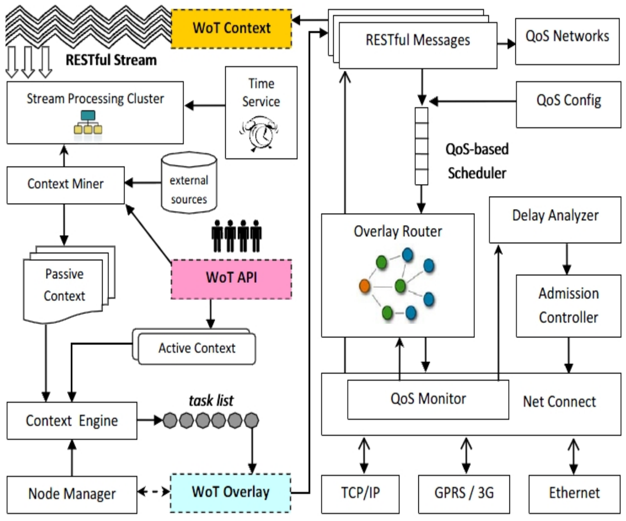

Figure 4. Internal architecture of a CPS Fabric (reproduced from [1])

The CPS Fabric architecture as shown in Fig. 4 consists of (a) WoT Overlay layer that facilitates communication amongst CPS Nodes (right side of figure) and (b) WoT Context layer for (1) deriving passive context from processing event streams, and (2) performing intelligent control based on active context, and the WoT API layers for registering application specific contextual information, infusing data from external sources, and integrating CPS Fabric with existing decision making modules and other related computational functions (left side of figure).

The WoT layers are implemented in the CPS fabric through the following components:

- Overlay Router: Used instead of underlying network router to ensure on-time delivery for time-critical RESTful messages, especially event messages from CPS Node. It uses multiple path delivery to achieve this.
- Delay Analyser: Monitors communication to ensure that all components within the CPS meets QoS requirements.
- QoS Config: Tags RESTful messages (especially control messages for actuators that require immediate action) with priority labels given application-specific requirements.
- QoS-based Scheduler: Schedule incoming RESTful messages through a priority queue based on the priority tags (high priority event messages from CPS Nodes).
- Stream Processing Cluster (SPC): Provides efficient stream query and storage for CPS applications.
- Time Service: Provides global time reference to ensure synchronicity to all components of the CPS and that the temporal properties of CPS events are accurately used by CPS nodes.
- Context Miner: Executes event queries to discover application-specific Passive Context that is of particular interest to CPS users.
- Context Engine: Analyse all Passive and Active Context and generates a list of tasks to be allocated to different CPS nodes for further processing.
- Node Manager: Critical for SPC management and task delivery as it updates the conditions of each CPS node.

The WoT as a framework empowers its use in a broader context including a number of existing application areas such as oil, gas, resources and manufacturing industries, vehicle, road, traffic and transportation, smart homes, human space technologies, smart appliances and wearable devices, social networking, and convergence. In our previous work, we provided a detailed case study on how the WoT CPS framework can be applied in an Intelligent Transportation System application for pre-trip planning, on-trip re-planning and accident mitigation [1]. In the next section, we show how this WoT CPS framework can be applied in developing a CPS-based electricity smart grid.

V. CASE STUDY – CPS-BASED FRAEMWORK FOR ACHEIVING DEMAND RESPONSE (DR)

Energy demand from users to meet their day-to-day needs is constantly increasing. Demand response (DR) activities are defined as “actions voluntarily taken by a consumer to adjust the amount or timing of his energy consumption. Actions are generally in response to an economic signal (e.g. energy price, or government and/or utility incentive) [14]. Demand response is a reduction in demand designed to reduce peak demand or avoid system emergencies.

Our proposed architecture for achieving DR through CPS consists of smart homes, various influencing units like temperature sensing devices, weather forecast devices etc and utility company as shown in Fig. 5. Smart homes consist of various energy consuming devices, smart storage etc, each equipped with a Demand Response (DR) Node and a local DR Fabric. The primary aim of the DR Node is to:

(a) manage the sensors attached to the device, capture the information from the device, send it to the local DR Fabric, and
(b) receive and act on the appropriate instructions from it.

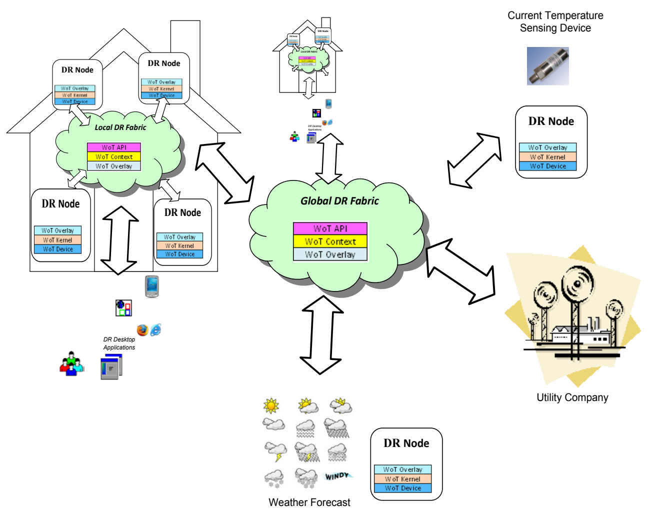

Figure 5. Achieving demand response through distributed real-time CPS.

The local DR Fabric is responsible for management of the DR nodes within a smart home and to communicate with the utility company. A DR Node has three layers as shown in Fig. 5. The WoT Device layer communicates with the sensors attached to the device. It also forms a communication link to the DR Nodes of other devices like the smart meter by using the event subscription mechanism. The information sent back from the local DR Fabric to the DR nodes is transformed into events by the WoT Kernel and communicated to the device through the WoT Device layer. The WoT Overlay provides an application-driven network-aware logical abstraction atop the current internet infrastructure such as TCP/IP for communication between the DR nodes and local DR Fabric. The local DR fabric has three layers. The WoT Overlay layer that communicates with the various DR nodes, is followed by a DR Context layer that parses the users requirements, computes the processes according to the inputs received from the utility company and identifies which relevant DR nodes need to be contacted. The WoT API layer provides abstraction in the form of interfaces that allows smart home users to interact with the WoT framework, in terms of inputting their preferences and requirements. The DR node communicates DR events to the local DR fabric through the CPS Event Channel. The Event Channel includes event collection processing (such as filtering, correlating, and aggregating), scheduling (reordering, prioritizing of different events etc.), and dispatching to different CPS nodes. The Event Channel can be configured to support different scheduling and dispatching strategies based on the DR application requirements.

A. *Achieving Demand Response in a Smart Home*

Unlike in traditional utility infrastructure, where the users do not know about the energy they have utilized in terms of financial amounts until the end of their billing cycle, smart grids allows them to achieve demand response by:

(a) keeping an eye on their consumption on a real-time basis, and

(b) take appropriate decisions in categorizing and choosing which devices to use according to its preferences in response to the quoted price.

To explain how this can be achieved by using the CPS system proposed in Fig. 5, let us consider that a user initially sets up the various parameters that relate to his preferences. Those parameters may relate to various factors such as the comfort levels that he wants his to-do tasks for the day like operating the washing machine or the dish washer etc. These preferences (events) are set by the user by using either the web-based browser application or though a smart phone via the WoT API layer of the local DR fabric. The desired preferences of the user will be parsed by the WoT context layer and the appropriate functions to be called to the different relevant devices determined. For example, if the user wants his comfort levels to be high, then the appropriate applications like the air conditioner and its specific functions like 'maintain temperature at 25^{o}C', 'windows blinds down' etc will be identified. These functions will be communicated to the appropriate DR node of the devices through the WoT overlay layer. These functions will be interpreted by the WoT Kernel layer of the DR node and they tune the device to work accordingly. This instruction is sent to the device through the WoT Device layer, and its operation is constantly analysed by using sensors to ensure that the device works as desired by the user. The sensors also capture and communicate back the amount of energy being consumed by the device to the CPS node which then communicates it to the smart meter node and local DR Fabric. The local DR fabric will send the amount of energy required to the global DR fabric through the WoT overlay layer by using the RESTful protocol. The utility company will have pre-determined the appropriate forecasted price for that time instant [15]. The amount to be charged to the user will be according to his consumption and will be communicated back to the local DR Fabric through the WoT overlay layer. The WoT context layer of the local DR fabric will keep a track of the amount of energy being consumed (in terms of financial amount) based on the consumption reported back from the DR nodes of various devices.

The proposed CPS system can perform various intervention strategies to achieve demand response. For example, let us consider the scenario where a user sets his parameters such as 'comfort level high between 3:00 – 5:00 PM' and 'do not consume more than $15 dollars worth of energy from the grid in a day' through the local DR fabric WoT API layer. These preferences get translated through the WoT Context layer, the appropriate functions invoked and communicated to the appropriate DR nodes. Let us assume that an event where the price which the local DR fabric gets from the utility company is such that the consumption of the user exceeds the set target of $15 for the day. In such scenarios, the local DR fabric may contact the DR node of smart storage to see the possibility of offloading the energy consumption for that particular device from the grid to the stored energy. These communications are done through the WoT overlay layer and the computations performed in the WoT context layer of the local DR fabric and then communicated to the DR nodes where they are implemented.

The energy consumption of the device is closely monitored through the sensors to ensure that the user's preferences are met. The local DR fabric, by using the WoT context layer can also re-prioritize the preferences of the users to meet the desired conditions of achieving demand response. For example, if the energy price is high, then it can shift the scheduled operation time of certain devices like dish washer, which the user does not want at this stage to another time. The local DR fabric within a home can also communicate with the external DR nodes like temperature sensors or weather forecast sensors through the WoT overlay layers to monitor external events and considers their inputs while performing computations in the WoT context layer for achieving demand response. For example, if the user wants to increase the amount of energy stored in his smart storage and the weather forecast sensor communicates that the next 2 days are going to be rainy, then the user can input his preference through the WoT API of the local DR Fabric 'do not use energy from smart storage'. This information will be processed by the WoT context layer and communicated across to the appropriate DR node to make them aware. Alternatively, if the user wants to maintain the home temperature at 25^{o}C between 3:00 – 5:00 PM and the temperature sensor communicates a drop in temperature during that time, then based on the communication between the temperature sensor DR node and local DR fabric and the computations performed at WoT context layer and communicated to the appropriate DR node, either the level of operation of the air condition can be reduced or it can be switched off for saving temperature.

So by integrating the CPS framework with WoT architecture, a robust, dynamic and two way communication IT infrastructure is achieved for realizing the concept of smart grids. As compared to the existing architecture for Demand Response, the CPS-WoT enabled architecture not only provides the users with various options that responds to various external and internal events but also allows them to control the various devices according to his preferences.

VI. Conclusion

In order to realise the full benefits of Cyber-Physical Systems, we need to have a framework that integrates various devices across the physical and digital domain on a real-time basis. In this paper, we discuss the various challenges that need to be solved in order to realise such a framework. This work focused primarily on one such challenge, namely event handling and processing. We proposed a Web-of-Things based framework for realising the real-time identification and processing of events in a distributed real-time Cyber-Physical System. In our future work, we aim to address the open questions identified in this paper.

References

[1] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber–physical systems," *Concurrency and Computation: Practice and Experience,* vol. 23, pp. 905-923, 2011.

[2] C. M. Krishna, "Real-Time Systems," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, ed: Wiley Online Library, 1999, pp. 262-270.

[3] C. Talcott, "Cyber-Physical Systems and Events," in *Software-Intensive Systems and New Computing Paradigms*. vol. 5380, M. Wirsing, J.-P. Banâtre, M. Hölzl, and A. Rauschmayer, Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 101-115.

[4] E. Lee, "Cyber physical systems: Design challenges," in *IEEE Object Oriented Real-Time Distributed Computing*, 2008, pp. 363-369.

[5] Y. Tan, M. C. Vuran, and S. Goddard, "Spatio-temporal event model for cyber-physical systems," in *29th IEEE International Conference on Distributed Computing Systems Workshops*, 2009, pp. 44-50.

[6] G. Quan, "An Integrated Simulation Environment for Cyber-Physical System Co-simulation," in *National Workshop on High-Confidence Automotive Cyber-Physical Systems*, USA, 2008.

[7] T. Hnat, T. Sookoor, P. Hooimeijer, W. Weimer, and K. Whitehouse, "Macrolab: a vector-based macroprogramming framework for cyber-physical systems," in *The ACM Conference on Embedded Networked Sensor Systems*, 2008, pp. 225-238.

[8] P. M. Poon, T. S. Dillon, E. Chang, and F. Ling, "XML Profile for Distributed Real Time Systems," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, 2008, pp. 570-576.

[9] D. Sharp, "Real-Time Distributed Object Computing: Ready For Mission-Critical Embedded System Applications," in *3rd International Symposium on Distributed Objects & Applications*, Italy, 2001.

[10] C. M. Weiszmann, S. (2005). *The Impact of Service-Oriented Architectures On Data Access and Integration*. Available: http://www.sdtimes.com/article/special-20050501-01.html

[11] T. Dillon, A. Talevski, V. Potdar, and E. Chang, "Web of Things as a Framework for Ubiquitous Intelligence and Computing," in *Ubiquitous Intelligence and Computing*. vol. 5585, D. Zhang, M. Portmann, A.-H. Tan, and J. Indulska, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 2-13.

[12] Y. Tan, M. C. Vuran, S. Goddard, Y. Yu, M. Song, and S. Ren, "A concept lattice-based event model for Cyber-Physical Systems," presented at the Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, Stockholm, Sweden, 2010.

[13] K. Yue, L. Wang, S. Ren, X. Mao, and X. Li, "An Adaptive Discrete Event Model for Cyber-Physical System," in *Analytic Virtual Integration of Cyber-Physical Systems Workshop*, USA, 2010, pp. 9-15.

[14] IndEco, "Demand side management and demand response in municipalities," IndEco Strategic Consulting Inc., Toronto2004.

[15] B. R. Szkuta, L. A. Sanabria, and T. S. Dillon, "Electricity price short-term forecasting using Artificial Neural Networks," *IEEE transactions on power systems,* vol. 14, pp. 851-857, 1999.