Architecture Requirements for Open Inference Networks

Priscilla Naa Dedei Hammond[a], Fritz Solms[b] and Bruce W. Watson[c]
Department of Information Science, Stellenbosch University, Stellenbosch WC, South Africa

Keywords: Open Systems, Internet of Things, Complex Event Processing, Pattern Detection, Distributed Architecture Systems.

Abstract: Real-time distributed Internet of Things (IoT) systems are increasingly using complex event processing to make inferences about the environment. This mode of operation is able to reduce communication requirements, improve robustness and scalability, and avoid the need for big data storage and processing. With systems making many inferences about the environment, there is no provision for general access to these inferences as well as the ability to make further inferences. Most IoT systems are closed or very limited in their openness and discoverability because they are exist for commercial purposes in which they control all the elements of the system. To this end, we propose the concept of an Open Inference Network (OIN): a novel open architecture for detecting and publishing complex events at various abstraction levels in an event cloud, i.e. the set of events consumed and produced by this system. Such systems contain three types of nodes: basic event source, inference, and activity nodes. Inference nodes detect event patterns that may encode some meaning and inject corresponding higher-level events into the event cloud. Activity nodes respond to an event by prompting an external system to perform some action; such an action may result in outputs that appear as new events. We consider the architecture requirements for OIN by assessing the required architectural elements against current IoT standards. These requirements mainly consist of event description and discoverability, and security, which together enable developers to collaboratively grow and evolve OINs. This is an intermediate study which does not include an empirical study.

1 INTRODUCTION

In order to improve scalability, robustness, privacy, performance and environmental impact within Internet of Things (IoT) based systems computing load is increasingly moved from server-based cloud computing to distributed computing done near the event sources. In the case of Edge Computing, the processing is pushed onto the devices which communicate events to the network (Choochotkaew et al., 2017). In the case of fog computing, the processing is done at various levels around the event sources including the devices which feed base events into the event cloud, the Local Area Network (LAN) to which the devices whose information is being processed are connected, on mobile devices, and anywhere on the Internet (e.g. on cloud servers) (Perera et al., 2017).

In addition to single events conveying information, one commonly finds that meaningful information can be inferred from correlations of events (Luckham, 2002). Identification of event patterns produced by event correlations is commonly done via Complex Event Processing (CEP) (Luckham and Frasca, 1998). These event correlations commonly involve correlations across event streams and time (Choochotkaew et al., 2017; Cugola and Margara, 2012).

Event-driven Architectures (EDAs) where decisions and actions are triggered by events are used increasingly to facilitate loosely-coupled asynchronous processing within a sensory world (Dunkel, 2009; Chandy, 2016).

The vision of ambient computing in IoT networks is to primarily populate environments with self-organizing devices which use information they receive from sensors to make inferences which form the basis for control decisions (Ali et al., 2022). Such systems would benefit from an open reference architecture based on a set of public standards. Even though a lot of research has gone into platforms and standards supporting pervasive computing, a complete and open reference architecture enabling different parties to integrate their self-organizing devices within an envi-

[a] https://orcid.org/0000-0002-7353-1844
[b] https://orcid.org/0000-0003-2845-1623
[c] https://orcid.org/0000-0003-0511-1837

Hammond, P., Solms, F. and Watson, B.
Architecture Requirements for Open Inference Networks.
DOI: 10.5220/0011088800003176
In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2022)*, pages 510-518
ISBN: 978-989-758-568-5; ISSN: 2184-4895

ronment of ubiquitous and ambient intelligence does not currently exist.

In this paper, we present an intermediate study which does not include an empirical study where we consider the architectural requirements for an OIN which is an open EDA facilitating incremental, cooperative construction of a system composed of three types of sensor, inference and activity nodes which can be integrated external systems executing business processes.

The system of loosely-coupled nodes is envisaged to continuously evolve and be grown by independent entities. The construction of such a system could be viewed as society continuously and cooperatively evolving an open nervous system for itself. We will call such a system an Open Inference Network (OIN).

For an OIN to be feasible, there are a number of critical architectural requirements covering the ability to (a) describe the data communicated with events including structure and relationships to its environment, (b) discover event sources which produce events one requires for to trigger activities or make higher-level inferences, and (c) assess the trust level of event sources and event source registries (d) whether an event consumer is entitled to the information provided by an event source.

The contributions of this paper are to:

- propose the concept of a fog-based software architecture enabling the collaborative development an OIN,
- to identify the standards required to facilitate an OIN and
- to assess which standards requirements can be met by standards specifications currently available and which standards need to still be specified in order to facilitate an OIN.

2 BACKGROUND

The proposed OIN would be integrated into the IoT, would have to be able to discover the meaning and structure of events, and would have to be able to identify event patterns resulting from meaningful event correlations. To this end we provide an overview of some of the concepts around IoT, ontologies, fog computing and CEP.

IoT is a fast growing technology providing an integration infrastructure between heterogeneous objects (such as smart devices, smart objects, sensors, actuators, Radio-frequency identification (RFID), embedded computers, etc.) as well as bi-directional interaction between the objects and people. Objects and environments (such as smart homes, smart cities, etc.) have become smarter via IoT, meaning they have been enhanced for interactions with systems and/or people and have been computerized and equipped with network interfaces (Rajguru et al., 2015). Physical devices embedded with sensors and actuators interconnected over a network continuously gather and exchange data, make decisions and initiate actions in order to enrich and improve the environment. These decisions are based on both real-time data and historical data (Augusto and McCullagh, 2007).

CEP is a technique used for real-time processing of event streams in order to reduce event load and to detect event patterns from which meaningful higher level events are inferred. Event patterns are specified declaratively using an Event Processing Language (EPL) and queries are continuously executed by an event processing or rules engine against event streams. Event patterns may specify correlations across event streams and time. This generally requires that CEP rules engines trigger state machines and that they make use of event caching. There exist, a variety of EPLs as well as a number of proprietary and open source implementations of CEP rules engines.

Fog computing is defined as a highly virtualized platform providing computing, storage, and networking services between end devices and traditional Cloud Computing Data Centers (Bonomi et al., 2012). It bridges between cloud servers and data devices. The key aspects of fog computing is that intelligence and computing power is placed on the Local Area Network (LAN). Data is transmitted from data devices to a fog gateway and then to cloud servers. Fog computing addresses the needs of IoT environments with the high volumes of data emitted from sensors and devices improving the overall responsiveness and reduce network latency. Fog computing also acts as an agent for lightweight IoT devices by keeping the security credentials of these devices updated.

An ontology defines objects, properties, abstractions (classes) and relationships between these. Ontologies are formally specified to standardize domain terminology and facilitate the use of automated reasoners to compute inferences.

Ontologies can be specified using either the Resource Description Framework (RDF) or the richer but potentially more restrictive Web Ontology Language (OWL). RDF is widely used to describe and exchange information on the Web. Information is provided in the form of subject-predicate-object statements called triplets (e.g. “My tea is hot”). These are commonly stored either relational databases or triple-stores which are optimized for the task of stor-

ing triples. Each of the concepts ("my tea", "is", and "hot") would be identified through a Uniform resource Identifier (URI) and the database entry for a statement would have 3 columns each containing a URI.

RDF Schema (RDF-S) is an RDF ontology introducing the concepts of literals, data types, classes and properties as well as specialization relationships in the form of sub-classes and sub-properties. In addition to RDF-S, one can use the Web Ontology Language (OWL) to specify an ontology. OWL is a richer language which restricts the statements one can make in order to facilitate more rigorous and more efficient automated reasoning. It comes in flavours (OWL-Lite, OWL-DL and OWL-Full) of increasing expressiveness but reduced limitations. A valid RDF document is not necessarily valid in OWL-Lite or OWL-DL but is valid in OWL-Full. On the other hand, OWL-DL guarantees computability.

3 RELATED WORK

Various studies have shown that partitioning CEP queries and distributing complex event processing across IoT edge devices and cloud virtual machines reduced communication volumes and improved scalability, performance, robustness, flexibility and cost (Choochotkaew et al., 2017; Ghosh and Simmhan, 2018). (Lan et al., 2019) proposed improving efficiency by decomposing CEP queries into subtasks distributed across fog devices. This reduces complexity the complexity of individual event processing nodes.

Both (Ren et al., 2019) and (Li et al., 2018) conducted comprehensive surveys architecture design of on emerging distributed fog/edge computing paradigms in IoT networks. The studies did not the event processing aspect.

(Cugola and Margara, 2012) introduced an application domain that demands real-time processing into data flows based on the data content and the relationship between the data in a distributed manner. They also touched on possibly creating a rule language with both data processing and event detection capabilities to enhance high expressiveness in catering for uncertain data and rules.

Since events are the primary concept of EDA, they need to be described by a formal structural event model based on a precise formalism/ontology (Schaaf et al., 2012). The key issue of a Structural Event Models is the layered hierarchy of all event types covering the incremental enrichment of raw sensor events into more abstract and sophisticated domain events.

(Yemson et al., 2019; Schaaf et al., 2012; Teymourian and Paschke, 2009) aimed to improve inferability through semantic complex event processing. However, inference based on rule definitions with rigid expressiveness limits CEP across overlapping ontologies defined for different domains.

(Shaw et al., 2009) studied an architecture where event records are analyzed as linked data from which event characteristics and links are inferred which are published in RDFS+OWL descriptions.

(Valle et al., 2021) identified common issues surrounding the lack of understanding in the integration of independent, heterogeneous and distributed systems but focused a high-level topological and architectural strategies for multi-system integration.

Cognitive Event Processing (CoEP) uses artificial neural networks to learn event patterns (Yang et al., 2015). The authors incorporated a natural language event constructor to increase expressiveness in event pattern definition.

4 OIN

OIN is an open network of event publishing nodes conveying either very fine grained information or higher-level inferred information to the network for consumption and higher level inference by other nodes. Both, concrete (root) events as well as inferred abstract (higher-level) events are thus available for higher-level event detection. To this end three types of nodes are published to an OIN:

- **Sensor Nodes** feed new information about the environment into the OIN. Examples include sensory or device nodes which provide event streams of sensed information and nodes which provide information about activities in external systems.
- **Inference Nodes** use complex event processing (CEP) to infer higher-level events from finer grained events. Complex events produced by inference nodes can be used in higher level pattern matching rules of yet higher-level nodes.
- **Action Nodes** are special types of inference nodes which use detected events to trigger some action in some external system. They may optionally publish either the results of the action as events or incorporate information received from external systems within generated events which may be consumed by other nodes to either make higher-level inferences or trigger subsequent actions (as in a sequential process).

For an OIN to be able to evolve within an open and dynamic world, an OIN node must be able to config-

ure and publish itself within an OIN. To this end inference and activity nodes deployed in OIN discover event sources which they require for their rule based inference. This requires the availability of semantic descriptions of produced events as well as the discoverability of event sources producing events communicating information a node requires for its decision making. To this end they must be able to describe the events they need for their inference as well as the events they produce. A semantic event description must include:

- the semantic meaning of the (possibly inferred) event itself (e.g. a wildfire),
- the meaning of complex (structured) components of the event,
- the meaning and value of each leaf element of the event,
- the relation of the event to other events (e.g. in the context of CEP pattern matching - there was an eruption), and
- for complex events, the pattern matching rule on a semantic level.

Being an open network where different, unknown parties may produce and consume events requires that nodes to assess whether they can trust a node which either produces or consumes events, i.e. neither will a node generally be willing to blindly trust an unknown event source that the events it produces are correct, nor is the information communicated with produced events meant to be consumable by any node owned by some unknown party. Event sources thus need to publish:

- their location on the internet,
- the semantics and structure of the information contained in the events they produce,
- the technology used to encode the event (e.g. JSON or XML),
- the physical communication channels over which the events are made available,
- information which enables nodes to assess whether they are willing to trust the event source and
- optionally restrictions on who may consume the events it produces.

4.1 OIN Application

Different entities can publish new event sources, inference and action nodes to the network producing events which may be consumed by any interested parties for their purposes. A software application, on the other hand, is meant to address some specific task. An OIN is expected to facilitate many applications and nodes can be shared across these. A software application leveraging OIN would contain elements which are not part of the OIN itself like external services, databases, user interfaces and so on.

Consider, for example, the view onto a subset of a potential OIN depicted in Figure 1. Part of the depicted OIN is used within a wildfire management application. Wildfires are common, particularly in the Western Cape, South Africa, where huge areas are regularly destroyed by fire leading to massive environmental, health and safety as well as cost implications. Early detection of fires is critical to reduce damage and cost.

In areas which have high wild-fire risk it is common to have a network of sensors measuring temperature, air and soil humidity, carbon monoxide, wind speed and wind-direction and light intensity (Natarajan and Manu, 2017).

The base layer is a sensor (diamonds) layer which includes from left to right light-intensity, temperature, soil-humidity, carbon monoxide, sound, rain, wind (speed and direction), and industrial emission sensors. The ovals represent inference nodes which use CEP to infer higher level events from event streams represented by dashed lines with arrows indicating the direction of event flow. For example, high-fire-risk events are inferred from the fire-danger CEP node (fire in a warning triangle) from light intensity, air temperature and soil humidity events. On the other hand the presence, intensity and area of a fire are inferred by fire detection nodes from event streams originating from the same temperature sensors as well as from carbon-monoxide and sound sensors. The inferred fire description events together with wind, soil humidity and rain events are used in conjunction with an external fire-spread prediction service (which has access to maps reporting combustible materials such as vegetation, buildings, among others) to generate fire-spread prediction events. Various warning systems could subscribe to this inferred event stream to issue fire warnings to parties which are at health, safety or damage risk. Furthermore, the fire-spread prediction event stream together with the fire risk and fire description event streams are used by an external fire combating service to initiate and optimize the effectiveness of fire combating activities.

The concept of an application is a little arbitrary within such an architecture. It typically involves grouping of nodes and external services based on joint purpose and ownership. For example the fire-related

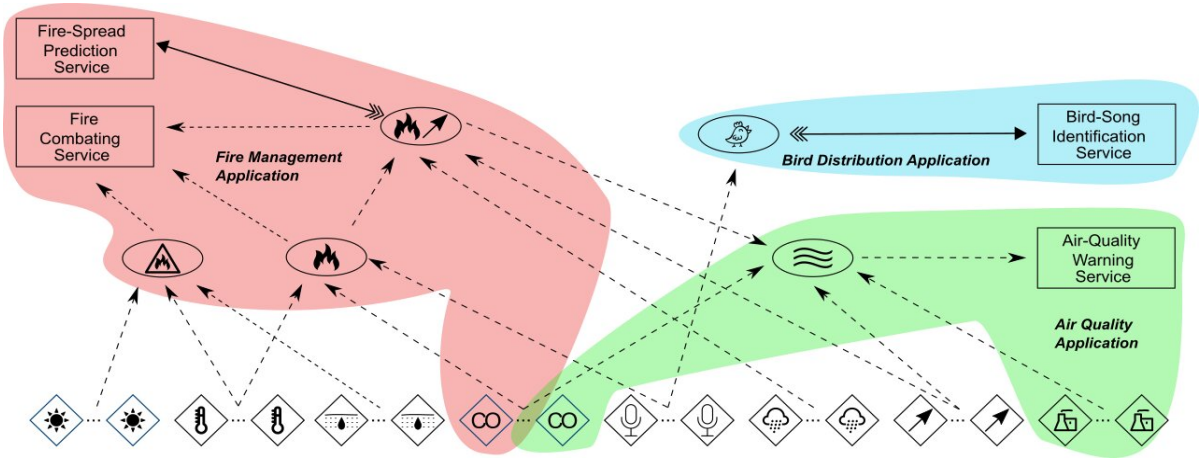

Figure 1: A view onto part of an OIN showing sensor nodes, CEP inference nodes, activity nodes, external services and grouping into applications.

services and nodes as well as some of the carbon-monoxide sensors are here grouped into a fire management application. The application does, however, feed off areas of the OIN which are not part of the application itself (light-intensity, temperature, soil humidity, sound, rain and wind sensors) and generates event streams which can be consumed by other applications built around the OIN by potentially other parties. For example, the inferred fire-spread prediction events from the fire management application are used in conjunction with event streams from CO, wind and industrial emission sensors to generate air quality prediction events which can be consumed by an air-quality warning service. The sound event streams are not only used to detect the sounds of a fire, but also by bird song inference nodes which use an external bird song identification service to generate bird spotting events.

4.2 Limitations

Since OIN is an open community-driven network where different parties independently deploy and remove different types of nodes, nodes cannot be hard-wired to other nodes. Instead, nodes auto-configure and auto-evolve their connectivity based on the availability of nodes which can provide streams of required events.

IoT systems are naturally event-based systems. EDAs have been shown to be well-suited for complex and continuously evolving systems (Chandy, 2016). Yet the software architectures of IoT systems are not commonly modeled using only a subset of concepts from EDAs.

There are a number of aspects which can be accommodated within the proposed software architecture but for which the architecture does not make explicit provision - i.e. it is left to either the infrastructure which deploys components into the open CEP IoT architecture or to the logic of the nodes themselves. Examples include:

- the deployment of nodes may be automated and optimized by client software feeding nodes into the architecture (Cugola and Margara, 2012)
- Nodes may use learning algorithms to identify meaningful event patterns and publish themselves as event sources for such meaningful events (Mehdiyev et al., 2015; Janjua et al., 2019)

5 APPROACH

In view of supporting the open, community-based aspect of an OIN, the core architectural concerns are

1. the ability to discover event sources which are in a position to provide the information required for certain inferences or activities and
2. the ability to assess the trustworthiness of a discovered event source.

The assessment of standards support for an OIN will hence focus on the *discoverabilty* and *security* with.

In this section, we discuss the methods used to identify the architectural and standards requirements around these quality attributes and to assess the requirements coverage of currently available public standards. This was done using the following approach.

1. Refining the architectural requirements for the two quality requirements.

   This was done by explicitly defining the objectives and sub-objectives of an OIN architecture in

relation to its key quality attributes. With each objective, its purpose, the motivation for its existence, its high-level scope of its functionality, and its benefits, opportunities and limitations were also defined.

2. Identifying aspects of these architectural requirements which need to be supported by public standards.

   The aspects were identified by breaking the OIN architecture into domain areas for which standards are required: Ontology specification, rules, data encoding, trust assessment, access control, conceptual data structure specification and event source registry service (ESRS).

3. Identifying sources from which standards candidates which may meet aspects of the the standards requirements can be discovered.

   This was done by investigating public standards documentations published by standard organisations and conducting literature search on academic papers for the identified standard domains as summarized in Table 1.

4. Associate with each standard requirement potential standard candidates which may meet the standard requirement either fully or partially.

   This was done by identifying for each standard requirement its domain and identifying those standards which target that domain.

5. Assessing the extend to which these standards candidates meet the requirements.

   By spanning and mapping the capabilities of existing tools that use these standards against the requirements scope for each standard domain and indicate what they can cater for the corresponding requirement. This also includes identifying the requirements for which standards still need to be developed.

6 STANDARDS REQUIREMENTS FOR AN OIN ARCHITECTURE

The architecture of an OIN is based on the IoT and will thus rely on the standards developed for the Internet itself (TCP/IP, DNS, among others) as well as standards used and being developed for the IoT. These include a range of established and emerging standards for light-weight and secure communication, event/message publication and streaming (MQTT, AMQP, among others), request-response messaging as well as developing (COAP), deploying or managing nodes (TR-069 and OMA-DM) and distributing tasks across node. For an OIN to function across tool suites and across nodes developed and maintained by different organizations, a set of additional standards particularly to facilitate discoverability. This includes standards for (a) describing primitive and complex events, (b) encoding events, (c) assessing whether an entity (event producer or event consumer) can be trusted, and (d) publishing and discovering trusted event producers (event sources or inference nodes) which produce required events encoded in.

This section only considers standards requirements for an OIN. In section 7, we assess the standards availability, i.e. which standards requirements can be met with available standards and standards which still need to be developed.

6.1 Event Description

An event may contain event metadata common to all events (e.g. address of the source node, the time stamp of the event), data which is specific to the type of event and (e.g. sensed temperature) as well as inferred meaning generated by inference nodes (e.g. the presence and attributes of a fire inferred from sensed events). Since event types are to be published by very different entities, event consumers must be able to discover event data structures including the meaning and relationships between data items.

Note that schema specification standards like eXtensible Markup Language (XML) Schema are not suited for an open world where the common semantics could be encoded by different entities within different data structures. On the other hand, ontologies are specifically designed for an open world scenario. They are used to share and communicate a common understanding for different domains. OIN require standards for specifying and querying and relating ontologies. The latter includes related concepts across ontologies including the ability to specify that two concepts in two ontologies are equivalent. The latter is required as different entities may work within their own ontologies which may have semantic overlaps with ontologies used by other, independent entities.

Additionally, an OIN will require the ability to register ontologies used by event producers contributing sensed or inferred events to the OIN as well as efficient querying of a distributed set of related ontologies. The latter will most certainly require distributed processing of queries (Sakr et al., 2018).

Table 1: The standard domains for OIN with corresponding public standard documentations.

Standard domain	Resources
Ontology specification	– Resource Description Format (RDF) (Gibbins and Shadbolt, 2009) – Web Ontology Language (OWL) (Bechhofer et al., 2004a)
Rules	– Rule Interchange Format (W3C RIF) (Kifer, 2008) – Reaction RuleML (Paschke, 2014; Paschke, 2006)
Conceptual data structure spec	– Web Ontology Language (OWL) (Bechhofer et al., 2004a)
Data encoding	– Concise Binary Object Representation (CBOR) (Bormann and Hoffman, 2013) – Javascript Object Notation (JSON) (Crockford and Morningstar, 2017) – eXtended Markup Language (XML) (Bechhofer et al., 2004b)
Trust assessment & access control	– Internet X.509 Certificate(RFC5280) (Boeyen et al., 2008) – ISO std for information security (ISO27001) (Watkins and Safari, 2013)
Event source registry	– Standard will need to be developed

6.2 Discoverability

Since an OIN is an open inference network with nodes being added, removed and modified continuously, a node must be able to query event producers (sensor, inference or activity nodes) which are both trusted and generate events containing the information required to either make some higher level inference or trigger some activity. To this end, nodes must be able to publish (a) its location on the Internet, (b) the physical event stream(s) it makes available, (c) either an embedded description or a link to a description of the events published on these event streams including – the variable data (e.g. sensed or inferred values), and – (c) fixed data which does not change from event to event (e.g. the physical location of a stationary node or the units within which a measured quantity is published), (d) the encoding used for the events, and (e) credentials which can be used to assess whether the producer is trusted by the consumer.

A query for suitable event producers would need to specify a description of the information required with the event and a set of constraints for the events the consumer intends to receive. Event constraints may be (a) constraints around the required data fields, (b) constraints on fixed data for an event stream (e.g. want to only receive temperature reading from nodes located within a specific physical region), and (d) constraints on the entity vouching for the data (trustability).

Furthermore, since an OIN is an open and potentially global infrastructure within which a new form of applications is being developed, the lookup service needs to be a hierarchical and decentralized event producer lookup service propagating information around event sources in a way similar to what Domain Name Systems (DNS) currently propagate domain name information across a network of hierarchical domain name servers. An OIN thus requires a set of standards facilitating the interoperability of such an Event Source Registry System (ESRS).

Furthermore, as event sources are modified, removed and added nodes requiring events may have to regularly update the event channels they observe making further queries to the ESRS.

6.3 Security

Even though an OIN is an open network, aspects of security are very important. In particular, an OIN faces threats of (a) access to event source registry, (b) access to event streams, (c) encryption of event data, and (d) certificate-chain based vouching of info (events).

7 ASSESSMENT OF AVAILABLE STANDARDS

In this section we consider the standards requirements from the previous section and identify currently available standards which meet aspects of these requirements as well aspects of the standards requirements which no standards could be identified. Table 2 provides a summary of the domains for which an OIN requires standards and, when available, standard candidates which address these requirements.

The RDF provides a basic set of elements to describe ontologies, whilst the OWL provides a much more extensive set of elements which also enable one to relate ontologies and to specify equivalence relationships between concepts within different ontologies. Furthermore, certain constrained OWL versions (e.g. OWL-DL) enforce computational completeness and decidability facilitating practical reasoning algorithms including the ability to reason in the context of ontologies containing contradictions ei-

Table 2: The domains for which OIN requires standards and available standards meeting these standards requirements.

Standard domain	Standard candidates
Ontology specification	– Resource Description Format (RDF) – Web Ontology Language (OWL)
Rules	– Rule Interchange Format (W3C RIF) – Reaction RuleML
Conceptual data structure spec	– OWL, OWL-DL – Web Ontology Language (OWL)
Data encoding	– Concise Binary Object Representation (CBOR) – Javascript Object Notation (JSON) – eXtended Markup Language (XML)
Mapping to physical	CBOR/RDF-JSON/XML serialization
Event messaging	MQTT, AMQP
Event streaming	Only tool-specific: Apache Kafka, IBM Event Stream, ...
Standards facilitating ESRS	Currently no candidates available
Trust assessment	SSL/TLS with X.509 certificates
Access control	SSL/TLS with X.509 certificates

ther within an ontology or across ontologies. Reasoning algorithms facilitating distributed processing do exist. Both OWL and RDF are standards maintained by the World Wide Web Consortium (W3C).

Furthermore, both OWL and RDFS provide the facility to specify classes representing data structures. These need to be encoded in reasonably efficient forms for the OIN. The Concise Binary Object Representation (CBOR) provides a standard for efficient encoding of data structures. Other, less efficient alternatives are JSON and XML schema.

The standards required for a Event Source Registry System (ESRS) as described in section 6.2 are currently not available and will need to be specified. Finally, trust assessment can be done using SSL/TLS with X.509 standards around maintaining, storing and querying certificate chains.

8 CONCLUSION

We proposed a novel fog-based software architecture and development paradigm for collaboratively developing systems within an open inference network using sensor, inference and activity nodes. Levels of inference nodes incrementally make inferences at different levels of abstraction and make this higher-level intelligence available to be used for further inference or to trigger activities within systems. An OIN requires standards for describing ontologies and data structures, reasoning, discovering suitable event sources through a network of hierarchical event source registries (an event source registry system, ESRS), publishing and streaming events and assessing trustability of event sources and consumers. We have identified standards addressing most of these requirements. Areas which do not have current standards support are the standards required around the ESRS and event streaming.

The future work will focus on including the specification of standards required for event streaming and ESRS, and the development of a reference implementation of an OIN which will represent the proof of concept implementation for empirical studies.

ACKNOWLEDGEMENTS

This work is based on research supported in part by the National Research Foundation of South Africa (NRF) (Grant number: 106028)

REFERENCES

Ali, O., Ishak, M. K., Bhatti, M. K. L., Khan, I., and Kim, K.-I. (2022). A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface. *Sensors*, 22(3).

Augusto, J. C. and McCullagh, P. (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems*, 4(1):1–27.

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., Mcguinness, D., Patel-Schneider, P., and Stein, L. (2004a). Owl web ontology language reference.

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., Mcguinness, D., Patel-Schneider, P., and Stein, L. (2004b). Owl web ontology language reference.

Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., and Cooper, D. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA. Association for Computing Machinery.

Bormann, C. and Hoffman, P. E. (2013). Concise Binary Object Representation (CBOR). RFC 7049.

Chandy, K. M. (2016). *Event Driven Architecture*, page 1–5. Springer New York, New York, NY.

Choochotkaew, S., Yamaguchi, H., Higashino, T., Shibuya, M., and Hasegawa, T. (2017). Edgecep: Fully-distributed complex event processing on iot edges. page 121–129.

Crockford, D. and Morningstar, C. (2017). Standard ecma-404 the json data interchange syntax.

Cugola, G. and Margara, A. (2012). Low latency complex event processing on parallel hardware. *Journal of Parallel and Distributed Computing*, 72(2):205–218.

Dunkel, J. (2009). On complex event processing for sensor networks. In *2009 International Symposium on Autonomous Decentralized Systems*, pages 1–6.

Ghosh, R. and Simmhan, Y. (2018). Distributed scheduling of event analytics across edge and cloud. *ACM Trans. Cyber-Phys. Syst.*, 2(4).

Gibbins, N. and Shadbolt, N. (2009). Resource description framework (rdf).

Janjua, Z. H., Vecchio, M., Antonini, M., and Antonelli, F. (2019). Irese: An intelligent rare-event detection system using unsupervised learning on the iot edge. *Engineering Applications of Artificial Intelligence*, 84:41–50.

Kifer, M. (2008). Rule interchange format: The framework. volume 5341, pages 1–2.

Lan, L., Shi, R., Wang, B., Zhang, L., and Jiang, N. (2019). A universal complex event processing mechanism based on edge computing for internet of things real-time monitoring. *IEEE Access*, 7:101865–101878.

Li, C., Xue, Y., Wang, J., Zhang, W., and Li, T. (2018). Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Comput. Surv.*, 51(2).

Luckham, D. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley.

Luckham, D. C. and Frasca, B. (1998). Complex event processing in distributed systems. Technical report, Stanford University.

Mehdiyev, N., Krumeich, J., Enke, D., Werth, D., and Loos, P. (2015). Determination of rule patterns in complex event processing using machine learning techniques. *Procedia Computer Science*, 61(Supplement C):395–401. Complex Adaptive Systems San Jose, CA November 2-4, 2015.

Natarajan, T. and Manu, V. (2017). Integrated forest fire and wildlife monitoring system using wireless sensor network. *Journal of Embedded Systems and Processing*, 2:1–7.

Paschke, A. (2006). The reaction ruleml classification of the event / action / state processing and reasoning space. *CoRR*, abs/cs/0611047.

Paschke, A. (2014). Reaction ruleml 1.0 for rules, events and actions in semantic complex event processing.

Perera, C., Qin, Y., Estrella, J. C., Reiff-Marganiec, S., and Vasilakos, A. V. (2017). Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)*, 50(3):1–43.

Rajguru, S., Kinhekar, S., and Pati, S. (2015). Analysis of internet of things in a smart environment. *Analysis*, 4(4).

Ren, J., Zhang, D., He, S., Zhang, Y., and Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Comput. Surv.*, 52(6).

Sakr, S., Wylot, M., Mutharaju, R., and Danh Le Phuoc (Author), I. F. (2018). *Linked Data: Storing, Querying, and Reasoning*. Springer.

Schaaf, M., Grivas, S. G., Ackermann, D., Diekmann, A., Koschel, A., and Astrova, I. (2012). Semantic complex event processing. *Recent Researches in Applied Information Science*, pages 38–43.

Shaw, R., Troncy, R., and Hardman, L. (2009). Lode: Linking open descriptions of events. In Gómez-Pérez, A., Yu, Y., and Ding, Y., editors, *The Semantic Web*, pages 153–167, Berlin, Heidelberg. Springer Berlin Heidelberg.

Teymourian, K. and Paschke, A. (2009). Towards semantic event processing. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, New York, NY, USA. Association for Computing Machinery.

Valle, P. H. D., Garcés, L., and Nakagawa, E. Y. (2021). Architectural strategies for interoperability of software-intensive systems: practitioners' perspective. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1399–1408.

Watkins, S. and Safari, a. O. M. C. (2013). *An Introduction to Information Security and ISO27001:2013, A Pocket Guide, Second Edition*. IT Governance Publishing.

Yang, J., Ma, M., Wang, P., and Liu, L. (2015). From complex event processing to cognitive event processing: Approaches, challenges, and opportunities. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1432–1438. IEEE.

Yemson, R., Konur, S., and Thakker, D. (2019). A novel semantic complex event processing framework for streaming processing. In *Proceedings of the 9th International Conference on the Internet of Things*, IoT 2019, New York, NY, USA. Association for Computing Machinery.