Efficient State Update Exchange in a CPS Environment for Linked Data-based Digital Twins

Andrii Berezovskyi, Rafia Inam, Jad El-Khoury, Martin Törngren, Elena Fersman

Efficient State Update Exchange in a CPS Environment for Linked Data-based Digital Twins

Andrii Berezovskyi Rafia Inam Jad El-khoury Martin Törngren
Elena Fersman
KTH Royal Institute of Technology, Mechatronics dept. [†]

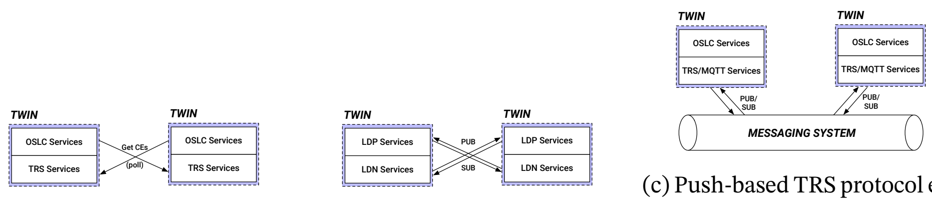
Abstract

This paper addresses the problem of reducing the number of messages needed to exchange state updates between the Cyber-Physical System (CPS) components that integrate with the rest of the CPS through Digital Twins in order to maintain uniform communication interface and carry out their tasks correctly and safely. The main contribution is a proposed architecture and the discussion of its suitability to support correct execution of complex tasks across the CPS. A new State Event Filtering component is presented to provide event-based communication among Digital Twins that are based on the Linked Data principles while keeping the fan-out limited to ensure the scalability of the architecture.

1 Introduction

The deployment of the autonomous Cyber-Physical System (CPS) is increasing day by day in commercial environments. To achieve this, the CPS shall be able to communicate and coordinate its actions with many surrounding systems in addition to the well-known tasks of localisation, planning, manipulation, object recognition etc. Such communication needs to scale for a variety use-cases, accommodate low-powered wireless devices and allow more centralised cloud-based approaches as well as multi-agent system setups.

SCOTT project (https://scottproject.eu/) has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway.



(a) Polling-based TRS protocol. (b) Push-based LDN protocol. (c) Push-based TRS protocol extension.

Figure 1: Different mechanisms for distributing State Change Events.

Additionally, CPS systems in the industrial contexts need to communicate with systems from other vendors and systems based on legacy or non-standard protocols.

Over the past decades, Hypertext Transfer Protocol (HTTP) [8] has been extensively used to communicate between enterprise applications and HTTP based Application Programming Interfaces (APIs) have become the foundational building block of the cloud-based systems. At the same time, CPS systems in the industrial settings frequently rely on Fieldbus technologies [24] that are generally incompatible with the TCP/IP stack [37, 38] and cannot be directly connected with the systems deployed in the cloud. Integrating both of these "worlds" inevitably requires bridging this protocol gap.

In addition to bridging the communication protocols, the semantics also need to be matched at the domain level. Each component within the CPS commonly relies on domain objects with different vocabularies, schemas, and semantics. Ontologies [22] provide ways to solve this problem while allowing a multitude of different vocabularies to co-exist, which is important for the deployment of heterogeneous CPS. Linked Data (LD) [10] and Semantic Web Services (SWSs) [17] offer ways to build web-based APIs that rely on well-defined vocabularies and ontologies to communicate with each other.

Digital Twins [21] are becoming increasingly important components in the CPS. Furthermore, they are best positioned to bridge the gap between the physical CPS components and the rest of the services in the CPS. In this paper, we consider the tasks of Digital Twins of keeping a copy of the physical component state up-to-date and sharing the state with the rest of the components accordingly. More importantly, we use Digital Twins to provide a way for the CPS services to command the CPS components and to perform other kinds of **read-write interactions**. When Digital Twins form the interface boundary for every component, the need to exchange state updates arises. As the number of the components in the CPS grows, the number of updates exchanged between all components is expected to grow quadratically.

This paper builds on our previously published results [9] where we have analysed how an architecture based on the Linked Data (LD) principles can be used to achieve interoperability between CPS components and described a reference architecture based on Linked Data Services (LDSs) and a publish-subscribe mechanism. There, we presented modifications to the Open Services for Lifecycle Collaboration (OSLC) Tracked Resource

Set (TRS) protocol [1] designed to share updates between LDSs to ensure its suitability for the CPS architectures. In particular, we argued for stricter ordering guarantees of the TRS Change Events (CEs) and for allowing the updated resources to be inlined with the CEs to ensure consistency of the state in the CPS. This, however, does not affect the total number of messages that have to be processed: every subscriber has to read the complete log of the service that publishes its updates before it can filter the relevant ones. In this case, message fan-out is $O(n^2)$-bound, which make the mechanism not applicable to large-scale industrial systems.

The main contribution of this paper is an architecture relying on Linked Data Applications (LDAs) for communication in order to reduce message fan-out in a CPS. Such reduction allows us to keep the system scalable. We introduce State Event Filtering (SEF) to provide a versatile event filtering capability. The architecture of SEF allows it to be used with many popular protocols for delivering Linked Data resource update events such as TRS [1] and Linked Data Notifications (LDN) [13] as well as to perform filtering of differing levels of complexity (metadata-only, payload-based, Knowledge Base (KB)-wide). Furthermore, the SEF can be implemented per the best cloud practices and be run either as a microservice itself or to be deployed on a dataflow processing framework such as Spark [46] or Flink [14].

The rest of the paper is structured as follows. In Section 3 we describe the background of the Digital Twin communication and publish-subscribe approaches applicable in the context of this publication. In Section 4 we present a warehouse logistics use-case that will be used to motivate the problem, and the architecture of the proposed solution. In Section 5, we evaluate the proposed solution by analysing the message fan-out in a motivating use-case. We give an overview of the related work in Section 2 and, finally, we conclude the paper and present future work in Section 6.

2 Related work

Digital Twins Research into the communication *between* Digital Twins is limited. Only recent research [3, 40] discusses *bidirectional* Digital Twin communication between the physical and cyber parts through the Digital Twins. Our publication deals with inter-Twin communication.

Event-Driven Architecture Event-Driven Architecture (EDA) [29] has been applied to similar problems before. Cugola et al. in 2001 presented JEDI [16] for supporting *process-based activities*, namely performing a “coordinated set of activities involving both humans and computerized tools”. There are multiple *overlay architectures* for the EDA [5] but the broker-based one is the most widely used and is currently the basis for developing cloud-based (“cloud native”) applications.

The subscriptions can be commonly made on the basis of topics, types but also based on contents. While academic surveys present them as orthogonal approaches [5], accepted industrial approaches summarised through Enterprise Integration Patterns [23] allow many of the subscription methods to be emulated on top of topic-based approaches that are widely supported by available message brokers: a *Datatype Channel* pattern can be used to make type-based subscriptions, content-based subscriptions can be made through the *Pipes and Filters* architectural style and a number of *Message Routing* and *Message Transformation* patterns. *Location-based filtering* has been addressed in a huge number of topics and systems; we will limit the discussion here to the systems that rely on a publish-subscribe mechanism. Examples include GREEN [42], Steam [32], works by Fiege et al. [18] and Cugola et al. [15].

Event filtering In a publish-subscribe system, the total amount of messages exchanged is equal to a product of messages generated times their fan-out. Content-based publish-subscribe (CBPS) research is therefore concerned with reducing the fan-out for each broker depending on the existing subscriptions and is the main focus of this paper. When subscriptions (and frequently, processing needs) cannot be defined *a priori,* events can be filtered by the consumers rather than by the producer and the infrastructure (brokers). The typical scenario for this is a distributed dataflow processing framework (commonly referred to as a Big Data framework) that processes a sequence of events from a log or another source [44]. Another aspect of filtering is that it can be generalised to running a query over a stream of data, the filtering being the simplest one (strictly speaking, a monotonic query that only relies on stream-to-stream operators [27]). This conversation is part of a larger discussion on how stream processing related to *complex event processing (CEP)* [31].

Linked Data is not the only way There exist approaches for integration other than those based on Linked Data. OPC Unified Architecture (OPC-UA) [28] and data-distribution service (DDS) [39] have been extensively used to connect heterogeneous systems. However, while OPC-UA provides an ability to unify the communication among CPS systems, it still requires adaptors or custom clients on the cloud service side of the systems because OPC-UA is commonly not used there. Linked Data, however, uses common web technologies for communication but also mandates the use of the RDF model that allows using shared ontologies. While all aforementioned technologies can be made to use RDF too, they often come with the default or recommended modelling approaches. Wang et al. [45] have demonstrated how OPC-UA information model can be mapped to an OWL ontology that is later used to support reasoning with real-time updates; however, real-time updates are not defined as events and are simply inserted into a central triplestore which limits the scalability of the approach.

Recently, partitioned log solutions such as Apache Kafka became popular because they offer unparalleled scaling *and* durability compared to other solutions. Linked Data offers systems based on different architectures, and TRS naturally maps to the log-based approach. Change Data Capture (CDC) can also be performed on SQL-based relational database management systems (RDBMS).

Semantic Web Similar problems have been investigated in the Semantic Web community, focusing on knowledge-based coordination, knowledge-based decision making and data stream processing [33]. While many approaches allow doing reasoning while processing the change, their granularity is usually too low for a real-world application context (a single triple or a single resource change) and are done at a very low level (triplestore, an equivalent of a Data-tier in the 3-tiered application architecture) [33, 36]. This limits the scalability of the system and conflicts with the way "cloud-native" applications are to be built. Interoperability of event sources has also been studied in the Semantic Sensor Web Project [4, 41]. Semantic web community has also considered the need to remove the need for the clients to do the work of processing all the changes [34]. This work, however, is again done at the level of the triples and within the triplestore with a reasoning profile. This does not solve the problem of actually processing the events but generating secondary events that result from reasoning on the updated KB.

Evaluation and Frameworks A variety of benchmarks exist but none to our knowledge target *communication between Digital Twins.* One example in the Semantic Web area is the SEP-BM benchmark due to Murth et al. [35].

3 Background

Digital Twins are becoming frequently discussed in the CPS domain as a way to establish a "digital thread" of the CPS from design, in manufacturing and throughout the lifecycle. Simulation aspect of Digital Twins is well-researched [11]. However, as CPS grows to thousands of components, *communication* between the components becomes an important issue in the operational phase of CPS. Digital Twins are perfectly suited to represent physical CPS components to the rest of the CPS and to become components' "cloud interface".

Linked Data is built on four principles [10] that are highly suited to dynamic and reconfigurable CPS systems. Linked Data relies on Resource Description Framework (RDF) model [30] to represent the information in a graph-based manner. Linked Data Platform (LDP) [43] further builds on the Linked Data principles and defines the basic building blocks for developing LDSs. OSLC specifications [2] build upon the World Wide Web Consortium (W3C) LDP recommendation to allow developing usable LDSs that can *interoperate.*

Whenever a state of a component in a CPS changes, the change might need to be observed in the rest of the system. Common methods to distribute State Change Events (SCEs) in LDS-based systems include W3C LDN [13], OSLC TRS and others. Broadly, those methods can be split into two categories: push-based (such as LDN) and polling-based (such as TRS).

The Fig. 1a illustrates the communication pattern in the polling-based approach. In the OSLC TRS protocol, Change Events from the corresponding OSLC Server are appended to the TRS Change Log. A single TRS Change Log is published per OSLC Server. Change Events contain only event metadata and among them is a link to the RDF resource impacted by the Change Event. In LDN, every subscriber is equipped with a *mailbox* (represented by a simple LDP Container). The LDP server where an event occurs issues an LDN Notification, and it is sent individually to the mailbox of every subscriber (see Fig. 1b). The payload of an LDP notification can be any valid RDF graph serialised into JSON-LD [26]. This opens up possibilities to either use lightweight notifications (e.g. using `as:Announce` class) or inline the payload fully.

In our recent publication [9], we have extended TRS protocol to *push* Change Events over the Message Queuing Telemetry Transport (MQTT) [6]. Additionally, the updated resource was inlined with the Change Event to maintain state consistency. The communication pattern, depicted in Fig. 1c, is similar to the one for the LDN previously shown in the Fig. 1b. Similarly, the LDN protocol can be converted into a polling-based if every notification is appended to a log like in TRS. In either the push-based or the polling-based protocols, the message processing volume remains unchanged because the subscribers need to process the SCEs before filtering them out.

4 System Design

4.1 Use-case

The use-case describes warehouse logistics operations, where the functionality is highly automated. The high-level goal statement is provided to the system, which is first translated by the system into different actions. In the system, actions are performed by heterogeneous devices. For example, robots are performing pick-and-place operations for the products and moving them between shelves and conveyor belts in the area with human operators not restricted from moving around freely. As shown in Fig. 2, physical CPS components, such as robots are paired with a Digital Twin, legacy services that are part of the CPS are accessed through Adaptors, for example, an Enterprise Resource Planning (ERP) system, while new services are developed with a native LD interface. Digital Twins reflect the state of the physical CPS components. The interface to all components throughout the system is uniform and is based on LD. The operations in the system are controlled through the Warehouse Controller and planned through the Planner Service that relies on Planning Domain Definition Language (PDDL) [19] to create correct plans for the components. If CPS components rely on other components during operation, they need to subscribe to the updates relevant for ensuring the goal is reached.

In the system, a SLS is introduced in order to perform filtering of the SCE messages and to reduce the number of messages to be processed by the individual Twin and Adaptor components in the CPS. It relies on the State Event Filtering (SEF) service, which is described in the Section 4.3.

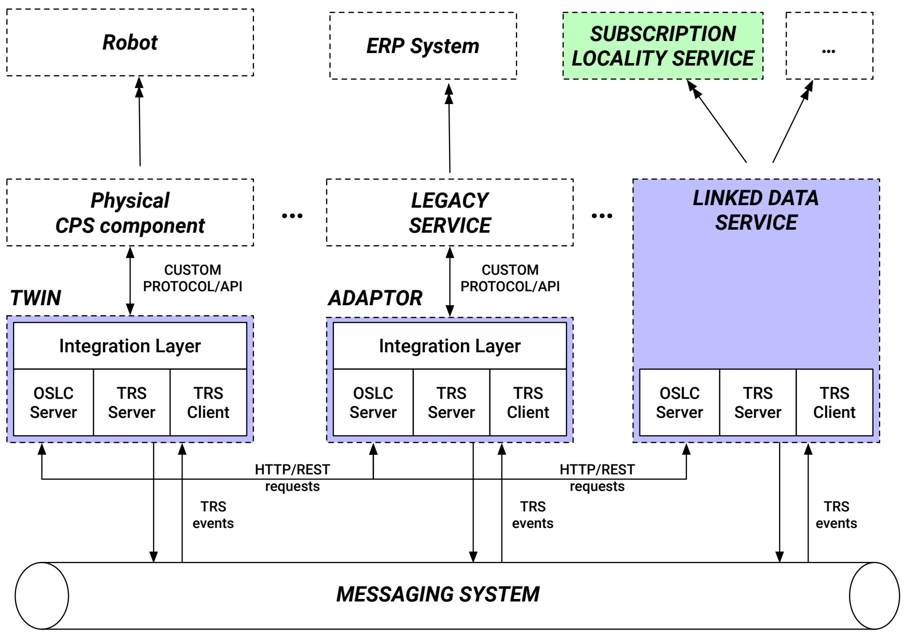


Figure 2: The use-case architecture with the newly introduced Subscription Locality service (SLS) component.

For analysis, we assume the warehouse to have $l \times l$ waypoints along which robots can travel. Furthermore, we assume each robot maintains 2 risk zones (for details regarding the use of risk zones for risk assessment see [25]) within which the actions of other components have to be monitored closely to avoid collision (yellow zone, z_y) or where corrective action is needed (red zone, z_r), respectively.

4.2 Assumptions

In this publication, we assume the scenario where all possible communication needs cannot be foreseen in advance and be clearly automated. Instead, we rely on an EDA that allows the subscribers to decide how to react to the incoming information rather than the publishing node ordering all (known) nodes to perform predefined actions. Internal state of the CPS components is encapsulated by exposing only the *observable state* subset to the subscribers of its Digital Twin. Further, we have a set of assumptions regarding the CPS and the environment it is used in.

1. The set of physical CPS components in the system is complete: $R = \{A, B, ...\}, \nexists r : \text{robot}(r) \;\; r \notin R$. Thus, each component can know about the total number of other components.

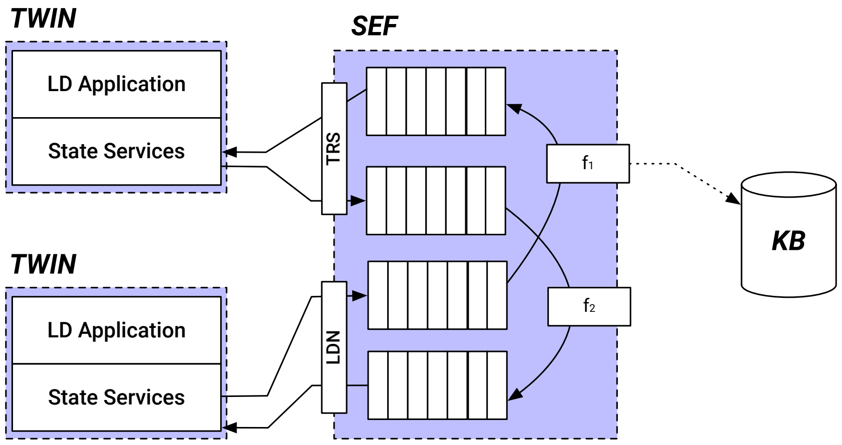


Figure 3: Architecture of the State Event Filtering service.

2. While the CPS system is complete, its state is not fully observable, and state changes can happen outside plan execution (through Action Effects, e.g. in PDDL).
3. A robot does not know about the location of a human.
4. A robot does not know the exact location of every other robot.

In this paper, we also assume that creating a single plan for all components in the system is not practical because the system may involve proprietary components with the "black box" behaviour as well as other entities (e.g. humans) that may change the CPS state in a way that cannot be predicted. Furthermore, classical planning is a PSPACE-hard problem (once we allow for negative preconditions and effects, [20]). For these reasons, we focus on producing plans with a domain and problem limited to a task at hand. The runtime component then has to ensure the ability to execute planned actions under the current system state.

4.3 Resource update architecture

As described in the introduction, the key problem in the event-based Linked Data system is the reduction of messages needed to convey relevant state updates to a limited number of subscribers. Each Digital Twin holds state in a set of RDF resources [30] it manages. We represent all SCEs for a given Digital Twin $t \in T$ in a totally ordered set L_t. When we consider resources across all Digital Twins in the CPS to be a part of a Knowledge Base (KB), the combined SCE log represents a partially ordered set $\mathcal{L} = \bigcup_{t \in T} L_t$.

In order to reduce the number of messages processed by any given Digital Twin, we need to apply a filtering function $f : \mathcal{L} \to \mathcal{L}$ (with its image being a subset of the preimage). Depending on how filtering can be done, we define **three classes of filtering methods for Linked Data events:**

1. Metadata-based approach. In this case, access to the resource in question is not necessary, and filtering can be done based on a set of limited metadata properties in the event RDF resource $f_m : \mathcal{L} \to \mathcal{L}$.

2. Resource-based approach. In this case, information in the updated resource $r \in R$ is required to do filtering; the resource has to be inlined with the event or retrieved separately. The filtering function in this case can be described as $f_r : R \times \mathcal{L} \to \mathcal{L}$.

3. KB-based approach. In this case, the information in the updated resource alone is not sufficient to determine the relevance of its update. As part of the filtering, the KB has to be queried.

   a. All of the KB needs to be available for the query. It is represented as a powerset of all resources in the domain of the filtering function $f_K : \mathcal{P}(R) \times R \times \mathcal{L} \to \mathcal{L}$.

   b. The KB query can be limited to a certain window $W \in \mathcal{P}(R)$, which would allow resolving the query via a stream reasoning framework [7,12].

If we look at the outlined filtering classes from the querying perspective, both metadata-based and resource-based approaches are cases of monotonic query that only relies on stream-to-stream operators [27].

The State Event Filtering (SEF) service (see Fig. 3) is proposed to process all of the State Change Events and to perform detection of the relevant events on behalf of the individual LDAs. The adaptor-like interface of the SEF service allows it to be designed using the same principles as any other cloud service, hiding its internal implementation. Various implementations are possible, but the one that can be easily set up using open-source components and scaled well is a combination of a persistent log (e.g. Apache Kafka) and a dataflow streaming engine (e.g. Apache Flink). Thus, SEF allows the LD **event protocol interfaces to be decoupled from event processing** that can be done in a technology-agnostic way.

The other characteristic of the SEF service is its ability to substitute the original event source. As shown in Fig. 3, the SEF service can expose TRS and LDN interfaces itself, making event consumption easy even for the components that are not aware of the SEF service existence in the architecture. Typically, SCE consumers would filter events internally as part of their processing and would not publish the intermediary result of filtered events. By publishing the filtered log of SCEs, the SEF service allows similar filtering to be done only once for multiple subscribers. This also allows the **SEF service to be used to integrate multiple LD event protocols,** including legacy ones without changing the components implementing those protocols.

Furthermore, because filtering inside of an SEF service can be captured by a function $f : \mathcal{L} \to \mathcal{L}$, filters can be *composed* so that given another filter g (possibly running on another SEF service instance) the combined filter can be described as $f_c = g \circ f$. Composition at the service level is an important property that allows both for the separation of concerns as well as robust scaling.

Internally, the SEF service can enrich SCEs with the information about the resources in question or with the resources queried over the KB. As shown in Fig. 3, the filtering function f_1 relies on the KB query while f_2 does not.

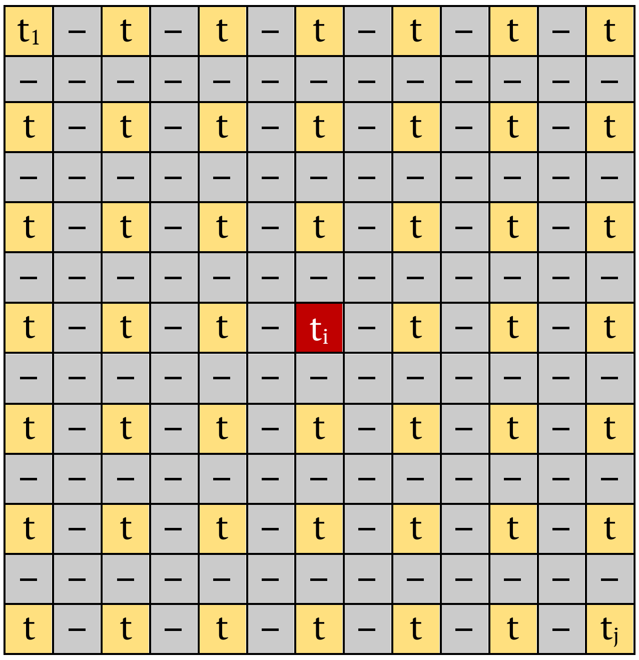


Figure 4: Physical CPS component t_i surrounded by j components; $z_y = 5$, $z_r = 1$.

5 System evaluation

The resulting system allows us to achieve a significant reduction in message volume given a few basic assumptions hold. Assuming that during plan execution TRS Servers for each Digital Twin or service generate k CEs, the number of messages m_t that each Digital Twin t needs to process grows with linear complexity and is bound by $O(n^2)$:

$$\begin{aligned} m_1 &= (n-1)\cdot k \\ m_2 &= (n-1)\cdot k \\ &\dots \\ m_n &= (n-1)\cdot k \\ \Sigma &= n\cdot(n-1)\cdot k \\ O(\Sigma) &= n^2 \end{aligned} \tag{1}$$

It is worth noting that the rate at which CEs are published is determined by the CPS components themselves. The subscribers can only affect the number of the messages processed by employing filters described below. In particular, they will not have to process any location update CEs from the components located outside their *yellow zone*.

After introducing the Subscription Locality service (instantiation of the SEF service) to filtering, the amount of messages that need to be processed in the system is equal to sum of the number of messages m_f to be processed (filtered) by the SLS itself plus the number of the messages m_t forwarded to each Digital Twin t after applying the filtering function flt_t (assuming only one filtering function):

$$\begin{aligned} m_f &= n\cdot k \\ m_1 &= |\{e_i \in \mathcal{L} | \mathsf{flt}_1(e_i)\}| \\ m_2 &= |\{e_i \in \mathcal{L} | \mathsf{flt}_2(e_i)\}| \\ &\dots \\ m_n &= |\{e_i \in \mathcal{L} | \mathsf{flt}_n(e_i)\}| \end{aligned} \tag{2}$$

While it is not possible to estimate the total number of messages Σ^f for a general case without knowing properties of filtering functions (in the worst case bound by $O(n^2)$ if $m_t = |\mathcal{L}|$, i.e. the filtering function does not filter out any events), the filtering functions used in our use case are based on the proximity of the physical CPS components and can be used for this analysis. For the analysis below will we will assume the risk zones to be fixed in size and non-overlapping.

Given that no CPS components shall be within the red zones of other components, the maximum number of components to be monitored shall not exceed $\lceil \frac{2\cdot(z_y+z_r)+1}{z_r+1} \rceil^2 - 1$, which is bounded by $O(1)$ for a large l. Given that message processing by the SLS itself is bounded by $O(n)$, total processing event processin\$\$g volume is, therefore, $O(n)$-bound. Given the warehouse size $l = 1000$ with $1000 \times 1000 = 10^6$ waypoints and the sizes of the risk zones $z_r = 1$ and $z_y = 5$ waypoints (as illustrated in Figure 4), there are no more than 48 components to be monitored at any point in time for a given component. This is in line with our assumption that the outermost risk zone is much less than the size of the warehouse $z_r < z_y \ll l$.

6 Conclusions and Future Work

In this paper, we have presented an architecture for exchanging State Change Events case within the Cyber-Physical System. Each physical CPS component is paired with a Digital Twin that has a uniform Linked Data interface for achieving interoperability between the components. Whenever the state of a component changes, its Digital Twin publishes a corresponding State Change Event, which is then processed by the subscribing components or their Digital Twins.

The key challenge addressed is the reduction of the SCEs to be processed as the CPS grows. Three kinds of filtering were identified: based on SCE metadata only; based on SCEs and changed resources; based on SCEs, changed resources, and the state of the KB with the special case of stream windows. A State Event Filtering service was proposed to enable filtering of the SCEs. This service can be scaled without forming a star topology in all cases except when access to the full state of the KB is necessary (class 3.(a)).

The architecture was applied to the warehouse logistics use-case. A Subscription Locality service was used to process location-related SCEs filtered by the SEF and to forward only the SCEs from the components in the vicinity of a given component. Analysis has shown that this processing strategy is $O(n)$-bound as opposed to direct processing of the SCEs by each Digital Twin alone, which is $O(n^2)$-bound. The existence of such bound allows to plan the network capacity and prevent network overload.

On a conceptual level, this architecture allows implementing a Content-Based Publish-Subscribe (CBPS) system in a CPS that is based on Linked Data while allowing non-content-based protocols (LDN, TRS, and others) and messaging systems (Kafka, MQTT, and others) to be used. In the future, improvements in CBPS can be used to develop systems with more advanced forwarding capabilities but they will likely require major changes to have both SCE protocols and messaging systems to be content-based as opposed to topic-based. Future directions include development of basic building blocks for typical filtering needs in the CPS such as based on location, time, current or planned operations as well as performance evaluation of the developed prototype.

References

[1] OSLC Tracked Resource Set specification. `https://raw.githack.com/oasis-tcs/oslc-core/master/specs/trs/tracked-resource-set.html`, (Accessed on 2019-03-29)

[2] OSLC Core version 3.0. OASIS Committee Specification (2017), edited by Jim Amsden. 04 April 2017

[3] Al Sunny, S.N., Liu, X.F., Shahriar, M.R.: Mtcomm: A semantic ontology based internet scale communication method of manufacturing services in a cyber-physical

manufacturing cloud. In: 2017 IEEE International Congress on Internet of Things (ICIOT). IEEE (2017)

[4] Balazinska, M., Deshpande, A., Franklin, M.J., Gibbons, P.B., Gray, J., Hansen, M.H., Liebhold, M., Nath, S., Szalay, A.S., Tao, V.: Data management in the world-wide sensor web. IEEE Pervasive Computing 6 (2007)

[5] Baldoni, R., Virgillito, A.: Distributed event routing in publish/subscribe communication systems: a survey. DIS, Universita di Roma La Sapienza, Tech. Rep 5 (2005)

[6] Banks, A., Gupta, R.: MQTT version 3.1.1. OASIS standard 29 (2014)

[7] Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying RDF streams with C-SPARQL. SIGMOD Record 39, 20–26 (2010)

[8] Belshe, M., Peon, R., Thomson, M.: RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2). Standard, Internet Engineering Task Force (2015)

[9] Berezovskyi, A., El-khoury, J., Fersman, E.: Linked data architecture for plan execution in distributed cps. In: IEEE ICIT (2019)

[10] Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5, 1–22 (2009)

[11] Boschert, S., Rosen, R.: Digital Twin—The Simulation Aspect, pp. 59–74. Springer International Publishing, Cham (2016), https://doi.org/10.1007/978-3-319-32156-1_5

[12] Calbimonte, J.P.: Linked Data Notifications for RDF Streams. In: WSP/WOMoCoE@ISWC (2017)

[13] Capadisli, S., Guy, A., Lange, C., Auer, S., Sambra, A., Berners-Lee, T.: Linked data notifications: a resource-centric communication protocol. In: European Semantic Web Conference. pp. 537–553. Springer (2017)

[14] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. IEEE Data Eng. Bull. 38, 28–38 (2015)

[15] Cugola, G., Munoz de Cote, E.: On introducing location awareness in publish-subscribe middleware. pp. 377–382 (01 2005)

[16] Cugola, G., Nitto, E.D., Fuggetta, A.: The jedi event-based infrastructure and its application to the development of the opss wfms. IEEE Trans. Software Eng. 27, 827–850 (2001)

[17] Domingue, J., Rowlatt, M., Gutiérrez, L.A., Cabral, L.: Semantic web services. In: Springer Berlin Heidelberg (2012)

[18] Fiege, L., Freiling, F.C., Kasten, O., Zeidler, A.: Supporting mobility in content-based publish/subscribe middleware. In: Middleware (2003)

[19] Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. J. Artif. Intell. Res. 20, 61–124 (2003)

[20] Ghallab, M., Nau, D., Traverso, P.: Automated Planning: theory and practice. Elsevier (2004)

[21] Glaessgen, E.H., Stargel, D.: The digital twin paradigm for future NASA and US Air Force vehicles

[22] Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Handbook on ontologies, pp. 1–17. Springer (2009)

[23] Hohpe, G., Woolf, B., et al.: Enterprise integration patterns (2003)

[24] Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series. Standard, International Electrotechnical Commission, Geneve, CH (2014)

[25] Inam, R., Raizer, K., Hata, A.Y., Souza, R.S., Fersman, E., Cao, E., Wang, S.: Risk assessment for human-robot collaboration in an automated warehouse scenario. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) 1, 743–751 (2018)

[26] Lanthaler, M., Gütl, C.: On using JSON-LD to create evolvable RESTful services. In: WS-REST (2012)

[27] Le-Phuoc, D., Parreira, J.X., Hauswirth, M.: Linked stream data processing. In: Reasoning Web International Summer School. pp. 245–289. Springer (2012)

[28] Leitner, S.H., Mahnke, W.: Opc ua - service-oriented architecture for industrial applications. Softwaretechnik-Trends 26 (2006)

[29] Luckham, D.: The power of events: An introduction to complex event processing in distributed enterprise systems. In: RuleML (2002)

[30] Manola, F., Miller, E., McBride, B.: RDF 1.1 primer. W3C Working Group Note, February 25 (2014)

[31] Margara, A., Cugola, G.: Processing flows of information: from data stream to complex event processing. In: DEBS (2011)

[32] Meier, R., Cahill, V.: Steam: Event-based middleware for wireless ad hoc network. In: ICDCS Workshops (2002)

[33] Murth, M., eva Kühn: A semantic event notification service for knowledge-driven coordination (2008)

[34] Murth, M., eva Kühn: A heuristics framework for semantic subscription processing. In: ESWC (2009)

[35] Murth, M., Winkler, D., Biffl, S., Kuehn, E., Moser, T.: Performance testing of semantic publish/subscribe systems. pp. 45–46 (10 2010)

[36] Phuoc, D.L., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: International Semantic Web Conference (2011)

[37] Postel, J., et al.: RFC 791: Internet protocol. Standard, Internet Engineering Task Force (1981)

[38] Postel, J., et al.: RFC 793: Transmission control protocol. Standard, Internet Engineering Task Force (1981)

[39] Schlesselman, J.M., Pardo-Castellote, G., Farabaugh, B.: Omg data-distribution service (dds): architectural update. IEEE MILCOM 2004. Military Communications Conference, 2004. (2004)

[40] Shahriar, M.R., Sunny, S.M.N.A., Liu, X.F., Leu, M.C., Hu, L., Nguyen, N.T.: Mt-comm based virtualization and integration of physical machine operations with digital-twins in cyber-physical manufacturing cloud. 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom) (2018)

[41] Sheth, A.P., Henson, C.A., Sahoo, S.S.: Semantic sensor web. IEEE Internet Computing 12 (2008)

[42] Sivaharan, T., Blair, G.S., Coulson, G.: Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing. In: OTM Conferences (2005)

[43] Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0. W3C Recommendation, February 26 (2015)

[44] To, Q.C., Soto, J., Markl, V.: A survey of state management in big data processing systems. The VLDB Journal 27, 847–872 (2018)

[45] Wang, S., Wan, J., Li, D., Liu, C.: Knowledge reasoning with semantic data for real-time data processing in smart factory. In: Sensors (2018)

[46] Zaharia, M.A., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. Annals of emergency medicine 39 6, 691–2 (2002)