Adaptive Complex Event Processing over Fog-Cloud Infrastructure Supporting Transitions

Manisha Luthra
Technical University of Darmstadt
Multimedia Communication Lab (KOM)
Darmstadt, Germany
manisha.luthra@KOM.tu-darmstadt.de

Boris Koldehofe
Technical University of Darmstadt
Multimedia Communication Lab (KOM)
Darmstadt, Germany
boris.koldehofe@KOM.tu-darmstadt.de

Ralf Steinmetz
Technical University of Darmstadt
Multimedia Communication Lab (KOM)
Darmstadt, Germany
ralf.steinmetz@KOM.tu-darmstadt.de

***Abstract*—Fog Computing is an emerging trend that can enable profound applications in the Internet of Things (IoT) arena. The IoT applications typically deliver vital information from multiple sources to the end-users in the form of notifications e.g., heart status in health monitoring. Complex Event Processing (CEP) is a powerful paradigm that bridges this gap from raw sensor data to meaningful information. But, IoT applications involves a wide distribution of heterogeneous devices that are highly dynamic (e.g. mobile nodes). This poses a strict need for a highly adaptive system. Consequently, first we propose a TCEP system that allows in-network processing of CEP operator graph on the fog-cloud infrastructure. Secondly, we describe how such a system can benefit from *transitions* between different CEP algorithms (mechanisms) to overcome the heterogeneity and dynamics of the fog-cloud infrastructure. This leads us to important research questions related to transition that are presented and addressed in this research work.**

***Index Terms*—Fog Computing, Internet of Things (IoT), Complex Event Processing (CEP)**

I. Introduction

Nowadays, there is a growing surge of Internet of Things (IoT) applications like smart cities, health, industry (Industrie 5.0) etc. [1] The IoT applications connect a multitude of devices over an interoperable communication network, exchanging enormous amount of information about themselves and their surroundings. Complex Event Processing (CEP) is an important paradigm to extract meaningful information from disparate sources in real-time. Typically, a CEP system consists of *event producer(s)* generating low-level data streams e.g., sensor data that are to be processed and correlated. The *event consumer(s)* or the IoT application *end-users* can specify the events of interest as a continuous query with the system. The query is transformed into an *operator graph*, where *operators* are the semantic units of query e.g., joins, filter, windows etc. The *event broker(s)* perform distributed in-network query (operator graph) processing, in order to determine the events of interest. The CEP system then notifies the events of interest to the event consumer(s). In this way, CEP provides a powerful way to deliver meaningful information to the end user for IoT applications.

However, the emerging wave of IoT applications e.g., in *Smart City* environment are demanding in one or more performance objectives, stringent latency requirement, network bandwidth constraints, mobility support and location-awareness. *Fog-computing* [2] is an emerging platform that provides some of the aforementioned characteristics. It extends the cloud-computing paradigm to bring computation towards the edge near to the end-users (consumers). Big cloud providers like Amazon and Google have recently launched fog locations, Amazon CloudFront[1] and Google Cloud CDN[2] respectively, that enable new breed of such applications in gaming, e-commerce, social media etc.

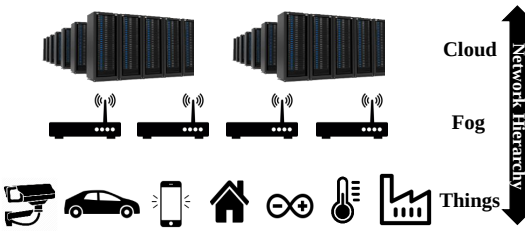

Fig. 1. Fog-Cloud infrastructure for deployment of IoT applications.

It is often pointed out that a federation of cloud and fog can allow wider range of applications where latency sensitive operators can be placed at the fog, while compute intensive operators at the cloud. An example of such a network hierarchy is illustrated in Fig. 1 [3].

Such a diffused infrastructure is suitable to support large-scale distributed CEP systems enabling deployment of IoT applications. For instance, to perform in-network continuous *query* processing over the fog-cloud infrastructure. However, there are two main challenges in this: 1) presence of heterogeneous infrastructure as illustrated in Fig. 1 by the three layers (things, fog and cloud) ranging from smartphones, vehicles to switches and routers to data center, 2) dynamic streaming environment consisting of: a) *data streams* produced at varying data rates, b) *devices* that can be mobile and c) fluctuating properties of the *communication network* e.g., bandwidth. Most importantly, the dynamic environment influences performance objectives of a large number of users. The distributed CEP system must take into account these challenges to deliver the desired performance e.g., low latency to an IoT application. Yet, state of the art distributed CEP (DCEP) systems fall short in their support towards highly

[1]https://aws.amazon.com/cloudfront/ [Accessed January 2018]
[2]https://cloud.google.com/cdn/docs/locations [Accessed January 2018]

dynamic nature of event consumers, producers and brokers. Considering mobility, there exists some approaches [3] that allow producers and consumers to be mobile, but connected to a fixed or quasi-stationary broker network. Additionally, they are restricted in their flexibility of providing mechanisms at run-time, that could deal with the dynamic nature of the environment. Some authors proposed elasticity in DCEP systems to deal with varying workload, but lack support for higher mobility [4].

To this end, we propose a concept for a highly adaptive DCEP system over a fog-cloud infrastructure that supports strong dynamics of streaming environment (viz. producers, consumers and brokers) for IoT applications. This is accomplished by runtime adaptation, aka. *transition* [5] between distributed CEP mechanisms. The transition to a *new* DCEP mechanism must fulfil varying and conflicting performance objectives from a large number of users. The ultimate goal is to provide methods to enable adaptation between DCEP mechanisms to deal with a dynamic streaming environment and performance objectives. Towards this, we investigate the following research questions for a Transitive-CEP (TCEP) system:

1) What are the potential mechanisms in a DCEP system that can benefit from transitions?
2) How can we specify transitions in a DCEP system?
3) How, when and who triggers the transitions in a TCEP system?
4) How to enable uninterrupted user experience using TCEP system?

We elaborate further on the problem based on a case study on Smart City Environment in Section II, where flexibility in performance objectives is desired. Afterwards, we enumerate important challenges identified in the case study, that influences the execution of an CEP *operator graph*, in Section III. Next, we identify key transitions in DCEP mechanisms for operator graph and overlay network in Section IV. Finally, we conclude and give next steps for our fog-cloud assisted Transitive-CEP (TCEP) system in Section V.

II. Case Study: Smart City Environment

According to Cisco[3], the *Smart-city* market is estimated to generate a revenue of hundred billion dollars by 2025. Running projects on the smart cities like European Smart Cities[4], highlight the increasing significance of key industry and service sectors in this domain including Smart Mobility, Smart Health, Smart Home, and other value added services. While the IoT space offers an end-to-end solution, our focus in this work is to enable real-time processing of IoT data using DCEP.

Current IoT architectures rely on either of the two extremes for data processing – cloud, or fog [2]. On the one hand, sending all the data to the cloud for high capacity storage and computation, e.g., video surveillance for traffic monitoring. On the other hand, time-critical applications like autonomous driving rely on local computation (on the sensors), particularly the Things layer in Figure 1. Still, IoT applications e.g., in the Smart City scenario pose challenges that are not solved by the two extreme architectures. For instance, routing data to the cloud can take several hundred milliseconds to react, that can lead to life critical situation, e.g., for future autonomous car to car communication. On the other side, some IoT devices are resource constrained, hence processing data locally can also be complex as well as costly. For instance, widely used Raspberry Pi devices used for multiple purposes comes with 1.2 GHz processor and 1 GB RAM. Such devices can only be used for pre-processing primitive events. Another concern is that sending data over a communication network to a cloud server consumes much energy and bandwidth. This implies a complex trade-off, particularly if local nodes are battery-powered. In this paper, we address these limitations of current IoT architectures by flexible and adaptive DCEP enabling in-network *query* processing guided by transitions (cf. Section IV). Therefore, we focus on three important questions: 1) *where*, 2) *how* to perform processing and 3) how placement of processing operators *impact performance objectives* of IoT end users.

A. Traffic Control

According to INRIX 2017 Global Traffic Scorecard[5], European drivers spent over 91 hours in congestion last year. The traffic congestion continues to rise, if it is left unchecked. A multitude of sensors in smart cities such as smart cameras, environmental sensors like audio, radars, induction loops and GPS sensors on smart cars can be used to derive insightful information about traffic. For example, notifications can be delivered to the drivers regarding the traffic flow, congested roads, unobstructed roads, or warning about road condition and accidents. DCEP allows this correlation of sensor data from the multiple data sources to derive higher level information such as the status of traffic congestion. This is performed by processing the information inside the network at multiple devices (e.g., fog or cloud nodes). We look into three possibilities of distributed CEP: 1) distributed local/fog CEP, 2) distributed cloud CEP or 3) distributed fog-cloud CEP.

Although, devices such as smartphones operate on high-speed processors with clock frequencies upto 1 GHz, processing big video streams from traffic monitoring cameras, locally on these low-powered sources is not resource efficient. A typical traffic monitoring camera captures at a resolution $>= \ 320 \times 240$ pixels and frame rate $10fps$ i.e. 768,000 pixels/data points per second, approximately. Sending all of this data to a cloud server imposes a significant cost. On the other side, processing lower level sensor data from vehicular sensors can be performed at local/fog nodes. Thus, trade off between performing DCEP locally or at cloud must be considered.

[3]The city of the future: Smart and connected, from http://www.cisco.com/web/tomorrow-starts-here/connectedcities/preview.html [Accessed January 2018]

[4]http://www.smart-cities.eu/ [Accessed January 2018]

[5]http://inrix.com/scorecard/ [Accessed January 2018]

The detection of complex event like traffic congestion is time and location dependent. It is not mindful to send a notification for traffic congestion when it has been cleared (time), or for a road where the user is not travelling (location). Thus, for an accurate and efficient decision, the system must be time and location-aware, but also be aware of other environmental factors (context-aware) that are significant for controlling traffic congestion. For example, it is important to note that the traffic conditions are not the same the entire day, as they are during rush hours. Therefore, it is very important to reconfigure DCEP mechanisms at runtime in accord with the traffic conditions and the need of end-users. For instance, for emergency services the notifications must be delivered undelayed, in contrast to a normal user.

B. Smart Health Monitoring

With the commence of IoT, there has been a growing number of devices that allows efficient health monitoring, including fitbit, body cardio scale, blood pressure (BP) monitor, Kito+ and many more. DCEP allows to collaboratively process the information from this variety of sensors to gain meaningful insights on one's health, e.g., heart status. However, to make full use of these vital sensors, information must be processed quickly and efficiently. The power of cloud can be used to process the information quickly, but transferring data to cloud for processing is time consuming, and the delays caused might lead to life critical situations like a heart stroke. Besides, cloud computing can also be a source of data privacy concerns. The privacy can be protected if the data is processed locally, either within the sensor, the body or the house/hospital where it was produced. Since user tend to be mobile, once the data leaves the private sphere, sophisticated DCEP mechanisms must be deployed to guarantee protection of privacy. On the other hand, prevalent insights can be obtained by data collection and batch processing offered by cloud, e.g, heart status over an year.

Vital signs from various sensors can be used to predict individual's health status e.g., by means of tools like Early Warning Score (EWS) system. It is a manual tool widely used in hospitals to track the condition of patients. It involves measuring five physiological parameters namely heart rate, systolic BP, breath rate, SPO_2 and body temperature and assigning them score between 0 and 3, with a lower score meaning a better condition [2]. IoT-enabled wearables for health monitoring can be used to empower systems like EWS, to continuously track and predict individual's health in an automated way. However, the system alone faces open issues that must be addressed [2]. Environmental factors influences the vital signs like heart rate etc., which must be considered to reach to a more realistic solution. For instance, a heart rate of 120 beats per minute would be an alarming sign for a patient who is sleeping, while it can be completely normal for who is exercising. If not considered, such characteristics can drastically decrease the performance of the system in terms of accuracy and precision. Corresponding DCEP mechanisms are needed to adjust the scores in order to avoid false alarms as well as guaranteeing protection of privacy in such systems. To summarize, there is a strong need to reconsider DCEP mechanisms to cater the needs of different end users of IoT applications in accord with the environmental context.

III. Challenges

In this section, we summarize the challenges brought by IoT applications studied in the aforementioned case study examples of traffic control and health monitoring.

Dynamic streaming environment. a) Data streams produced by IoT devices arrive at varying rates and volume, but also with a constantly changing quality (or certainty). E.g., sensor data produced by some devices can be noisy and erroneous, while by some devices can be more accurate. Besides, b) the streamed data has to be processed by user devices that perhaps can be resource and memory constrained (cf. Section II-A). In addition, due to mobility some devices might become unavailable for processing. Furthermore, c) the communication network that connects the IoT devices, possess highly fluctuating properties e.g., time varying latency, bandwidth, etc., that might have an influence on performance.

Varying performance objectives. As pointed out earlier in our case study, the dynamic streaming environment causes changes in the performance objectives of the IoT application. For instance, latency requirement of an emergency service (ambulance) for traffic notification will be significantly urgent than of a normal user. Similarly, in health monitoring the environment or user context (like location and activity) impact the performance perceived by the end user – privacy and accuracy, respectively.

Heterogeneity of infrastructure. The diffused infrastructure of fog-cloud itself is a challenge because of network and system heterogeneity. Some nodes might be geographically located far away, while some could be near. Because of this, there can be huge latencies between some nodes. Thus, TCEP system must prepare for proper coordination and planning of execution, i.e. where to process an operator – on cloud or fog node, split an operator to fog and/or cloud or perform parallel processing at both of them. The decision varies based on the adaptive selection of CEP mechanisms.

IV. Mechanism Transition in DCEP

A. Need for Transition in DCEP mechanisms

In the view of the aforementioned challenges, there is a strong need of run-time adaptation of the underlying DCEP mechanisms. Current systems neglect that the DCEP mechanism performs well only under the given environmental conditions, the respective assumptions and performance objectives. However, if dynamics are introduced into the environment, the system's performance objective might not be met. For instance, higher workload might render system unreliable and inefficient, as observed in the case study presented in Section II. Besides, performance needs of large number of users might vary significantly e.g., for emergency service in traffic control and exercising habits of different users in health monitoring. Thus, a DCEP system must adapt its mechanisms at runtime subject to the performance objectives of the end-users. In our

existing work [5], we show that in this case a *transition* is well suited. We extend this work, to show our hypothesis that transition in CEP mechanisms e.g., operator placement (cf. Section IV-B) could aid us in run-time fulfilment of performance objectives for dynamic user environment.

B. Mechanism: Overlay and operator graph transitions

DCEP system detects events of interest by distributed in-network processing of *operator graph* over the fog-cloud infrastructure. To do this, an appropriate selection of a node to deploy an CEP operator is performed, i.e. well known as *operator placement* problem. The operators are assigned to the nodes such that the performance objectives specified by the system are achieved. The performance objective are such that, they best satisfy the end-user's requirements. Once the operators are placed on the nodes, the *operator graph* is processed in a CEP *overlay* called an *operator network* [3]. It has been proven in the literature, that an optimal assignment of an operator to a node is a NP-complete problem. Thus, many heuristics and approximation algorithms are proposed for operator placement [6]. Although, each algorithm provides a solution to optimally place an operator to a node, they make different assumptions for the respective problem. The design characteristics of the placement are based on these assumptions and the performance objective of the application. The dynamics in the environment as discussed before (cf. Section III) influences the assumptions, performance objectives and hence choice of the placement mechanism.

Centralized vs. Decentralized Algorithms. The placement algorithms are characterized based on: 1) *how* the placement decisions are made and 2) *whether* the decisions require centralized knowledge about the environment [6]. The centralized algorithms have central knowledge about the environment e.g., on network state, workload and the resources, whereas the decentralized algorithms make decision on placement based on local knowledge. Apparently, centralized algorithms suffer from scalability issues while decentralized algorithms not. The event workload on the system in the IoT applications vary significantly, as specified in our case study. Thus, transitions between centralized and decentralized algorithms are to be explored, to ensure that the CEP overlay is not under-utilized but also not over-utilized. Scalability can be provided in a distributed CEP system in two ways: 1) vertical scalability (scale up), 2) horizontal scalability (scale out). Scaling up means to add/remove resources to/from the nodes, while scaling out means to add/remove nodes in/from the CEP overlay network. Techniques like parallel processing (scale up) or load partitioning and balancing (scale out) can be used to adapt to decentralized algorithm and thereby satisfying the workload requirements. Recently, there is an increasing interest in providing such adaptations by using auto-scaling strategies [4]. This work could be a start point to further analyze the CEP overlay transitions.

Static vs. Dynamic Operator Network. Mobility is one of the major causes of dynamics in the IoT applications. Decentralized algorithms can efficiently respond to such dynamic changes at runtime. This requirement raises important issues like efficient operator migration. It refers to efficient movement of an operator from one node to another to optimally satisfy a performance objective, in response to changes in the environment e.g., mobility. However, operator migration is costly, especially for stateful operators [3]. Thus, transition between mechanisms for static vs dynamic networks are investigated to avoid additional costs and to provide an efficient and seamless transition. For static or quasi-static networks it is recommended to avoid operator migration, as it is very expensive. On the other hand, for extremely dynamic operator network, migration is crucial. Existing work [3], [5] looks into this problem but not for mobile operator network. Additionally, methods for cost of operator migration are yet undiscovered. Moreover, existing approaches for operator placement and migration satisfy the *same* performance objective i.e. statically specified at the design time. However, there are situations in an IoT environment where the performance objective changes as illustrated in the case study. For this reason, transitions in the identified CEP mechanisms are prevalent.

V. Preliminary Conclusion and Future Work

In this paper, we presented important research questions motivating the need for transitions between different existing CEP mechanisms in a highly adaptive and context-aware CEP system. We presented a case study on Smart City environment for two different applications, traffic congestion control and smart health monitoring. Several scenarios (traffic control and health) are identified where transitions in CEP mechanisms can be beneficial with assistance of a fog-cloud architecture. Finally, important CEP mechanisms are identified for transitions corresponding to the situations in our case study. Intuitively, transitions are costly and therefore in the near future, we will look into its cost and benefits w.r.t. aforementioned scenarios. Furthermore, important research questions such as how, when and who triggers the transition are to be investigated.

Acknowledgement: This work has been co-funded by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Center (CRC) 1053 – MAKI.

References

[1] P. Samulat, *Die Digitalisierung der Welt: Wie das Industrielle Internet der Dinge aus Produkten Services macht*. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, pp. 103–124.

[2] M. A. Rahmani, L.-S. P. Preden, and A. Jantsch, *Fog Computing in the Internet of Things*. Springer International Publishing, 2018.

[3] B. Ottenwälder, B. Koldehofe, K. Rothermel, K. Hong, D. Lillethun, and U. Ramachandran, "MCEP: A Mobility-Aware Complex Event Processing System," *ACM Transactions on Internet Technology (TOIT)*, vol. 14, no. 1, pp. 1–24, 2014.

[4] T. Heinze, Z. Jerzak, G. Hackenbroich, and C. Fetzer, "Latency-aware elastic scaling for distributed data stream processing systems," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '14. ACM, 2014, pp. 13–22.

[5] P. Weisenburger, M. Luthra, B. Koldehofe, and G. Salvaneschi, "Quality-aware runtime adaptation in complex event processing," in *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS '17, 2017, pp. 140–151.

[6] G. T. Lakshmanan, Y. Li, and R. Strom, "Placement strategies for internet-scale data stream systems," *IEEE Internet Computing*, vol. 12, no. 6, pp. 50–60, 2008.