Dwarakanath, R.; Koldehofe, B.; Steinmetz, R.: Operator Migration for Distributed Complex Event Processing in Device-to-Device Based Networks. In ACM M4IoT, 2016, pp. 13-18, ISBN: 978-1-4503-4663

Operator Migration for Distributed Complex Event Processing in Device-to-Device Based Networks

Rahul Dwarakanath
Multimedia Communications Lab (KOM), TU Darmstadt
Darmstadt, Germany
dwarakan@KOM.tu-darmstadt.de

Boris Koldehofe
Multimedia Communications Lab (KOM), TU Darmstadt
Darmstadt, Germany
koldehofe@KOM.tu-darmstadt.de

Ralf Steinmetz
Multimedia Communications Lab (KOM), TU Darmstadt
Darmstadt, Germany
steinmetz@KOM.tu-darmstadt.de

ABSTRACT

Recent times have seen a surge in the number of context-aware systems, given the proliferation of sensors and sensor-based devices, especially with the advent of new paradigms such as the Internet of Things (IoT) and device-to-device (D2D) communication. Complex event processing (CEP) provides a cogent means to obtain higher-level context information from low-level sensor data streams through *operator graphs*, which dictate the order of event processing steps. However, the reliable execution of these operator graphs in a distributed manner becomes increasingly challenging in dynamic D2D environments, especially when device availability fluctuates, necessitating efficient operator migration strategies. In this paper, we first analyze the existing efforts towards operator migration for their applicability in D2D-based networks. Subsequently, we propose our initial approach for reliable operator migration by exploiting the intermediate states of the CEP operators.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; C.4 [**Performance of Systems**]: Reliability, Availability, and Servicability

Keywords

Complex Event Processing, Operator Migration, D2D, Reliability

1. INTRODUCTION

Over the recent past, there has been a rapid increase in the number of context-aware user-centric applications, given the steady proliferation of sensor-based devices (especially handheld devices like smartphones, tablets, etc.) and other sensors in the environment. The advent of the Internet of Things (IoT) paradigm promises an increased availability of information about the user environment. These applications monitor the context of the users and adapt their services to suit the situations experienced by the user at different points in time [5]. Examples of such applications include location-based services (e.g. Waze[1]), personal finances (e.g. Mint[2]), and context-aware communication [6], among others.

Complex event processing (CEP) provides the means to detect the occurrence of inherent patterns in low-level data streams (e.g. sensor data), thereby allowing for an efficient recognition of higher-level situational context [4]. The required information about a given user and the prevailing situation is generally obtained by fusing data from sensors and inferring them in an appropriate manner. Principally, CEP involves the sequencing and reordering of incoming data streams so that they may be filtered, correlated, and aggregated to obtain the required results. Each of these fundamental computing modules are termed operators. The processing logic to obtain the desired higher-level events can be organized in the form of operator graphs, which dictate the order in which the operators (graph vertices) need to be executed. The sources of low-level event data, the individual CEP operators, and the consumers of the processed data are connected by event streams to an operator graph [18].

The CEP operator graphs can be executed either individually (e.g. on smartphones), or in collaboration with the available processing devices, where an efficient placement of the operators among the available devices becomes crucial. Generally, the processing of user context entails the analysis of large data volumes coming from heterogeneous sources, and calls for high requirements in terms of resource usage (e.g. battery power, memory space, etc.) and latency [18,20]. Existing literature addresses this issue by employing a distributed processing approach and focusing on assuring either energy-efficiency or low processing latency [18, 21, 24].

En route to 5G networks, one of the paradigms expected to play a key role towards facilitating efficient network coverage is device-to-device (D2D) communication [22]. This is particularly pivotal in the IoT, where devices such as user smartphones and environmental sensors communicate amongst themselves to share and process information collaboratively, without the involvement of any centralized entity [1, 17]. Such D2D networks are known to be dynamic in nature, given the random movement of the involved devices as well as their resource-constrained nature, increasing the network's susceptibility to failure. Thus, besides an efficient operator placement mechanism, the reliable execution

M4IoT 2016, December 12-16, 2016, Trento, Italy

DOI: http://dx.doi.org/10.1145/3008631.3008634

[1] https://www.waze.com/en/
[2] https://www.mint.com/

of CEP operator graphs in D2D networks requires mechanisms for their migration, as well. Operator migration is paramount to combat the dynamic nature of such networks, so that node departures or failures do not adversely affect the overall functionality of the CEP system.

Typically, the principle behind operator migration involves one of two approaches—active replication [18,19] and rollback-recovery [7, 13, 20]. Active replication of certain operators on backup nodes accounts for a timely recovery of the system upon operator migration but at the cost of increased processing power and communication overhead. In rollback-recovery, certain checkpoints are maintained in the form of persistant storage to facilitate system restore opportunities. This, however, generally leads to increased delay penalties (loss of events and system downtime) and/or increased usage of buffer space.

In this paper, we summarize the main requirements for efficient operator migration in D2D-based networks, by analyzing the internal workings of CEP operators. We identify the key shortcomings of related work for distributed CEP in one such dynamic environment. In order to improve operator migration, we exploit the intermediate state information of the different operators to reduce the amount of information transferred and buffered on the devices. In doing so, we propose an initial approach to compromise between an active replication of all event streams and increased delay penalty through rollback-recovery.

In the following, Section 2 examines the related work towards operator migration in event processing systems, in general. Section 3 introduces the system model and presents the internals of a CEP operator. Our proposed approach for operator migration is presented in Section 4, followed by a discussion of its advantages in Section 5. Finally, Section 6 concludes the paper.

2. OPERATOR MIGRATION SO FAR

By and large, the existing approaches on distributed CEP focus on the optimization and flexible execution of the operator graphs, and in particular, the optimal placement of the operators on the available devices. Latency is considered as an important metric while placing the operators among the available processing devices, as shown by a survey by Lakshmanan et al. [14]. Other works consider the case of latency and network traffic for the placement of CEP operators in different application settings [4, 10]. Starks et al. present an energy-efficient mechanism for distributed CEP in MANETs [21], exploiting the advantages of decentralized operator placement. They aim to minimize the energy consumption in resource-constrained devices by limiting the data transmission costs and achieving near-optimal operator placement, compared to a centralized approach.

However, all of these approaches deal with the initial placement of CEP operators, but do not address the problem of operator graph maintenance. Operator migration is a key requirement in systems where nodes can fail or disappear, such that the operator graph can be restored. Most approaches towards reliability and fault-tolerance in event processing can be put into two main categories—(active) replication mechanisms [19] and rollback-recovery mechanisms [7].

Active replication mechanisms were introduced first by Schneider [19] towards fault-tolerant systems. Such systems minimize the delay penalty (loss of events) upon migration, such that it takes less time to restore the processing chain. However, this entails a high amount of communication overhead and processor utilization, since all the event streams must be replicated on backup nodes. Völz et al. [23] propose an active replication scheme for CEP systems, particularly focusing on coordinating the replicas for each operator. Ottenwälder et al. [18] propose a probabilistic approach for migration in mobile CEP (MCEP) with respect to mobile situation awareness applications. They provide mechanisms to optimize system performance through reduced latency and improved bandwidth utilization in dynamic situations.

Rollback recovery is generally performed by replaying stored backups of previous checkpoints during the execution of an operator graph. This method (also, upstream backup) is used frequently in large event processing systems, given the flexibility in determining when and which part of the event streams are stored. Operators maintain events in their output buffer until they receive acknowledgments for those events from the subsequent operator(s). Hwang et al. [11] first introduced this approach for stream processing applications. Gu et al. [8] focus on improving checkpointing by reducing the overhead through cumulative acknowledgments.

Fernandez et al. [3] propose a scale out and fault-tolerance mechanism for stream processing systems by managing operator state. They introduce an approach to scale out bottleneck operators as well as restore the system operation after node failures by exploiting the processing, buffer, and routing states of each operator. Koldehofe et al. [13] approached the same issue by proposing an adaptive acknowledgment-based scheme to maintain certain *savepoints* that can be used to restore an event processing system after failure.

Certain other approaches focus on combining the two approaches, exploiting the trade-off between the two. Hwang et al. [12] propose an approach that choses the best scheme between upstream backup, passive standby, and active replication, based on the resource consumption and recovery time of the operators. Heinze et al. [9] propose an adaptive approach for fault-tolerance by optimizing the number of operators running in parallel (replicas). They use recovery time estimates for each operator in order to determine the operators for replication.

In our work, we address the issue of operator migration in dynamic mobile environments with D2D communication, where the availability of the devices for sensing and processing varies with time. We propose an approach that makes use of both active replication as well as rollback recovery, wherein we exploit the state information of the CEP operators in order to improve feasibility of operator migration in such scenarios.

3. SYSTEM AND CEP MODEL

In this section, we establish the fundamentals as well the assumptions behind our approach. We describe our system model as well as the details of an operator graph. We also delve into the internal state of an operator, which plays an important role in our approach.

3.1 System Model

Our system model comprises user smartphones and other sensor devices that provide information about the user environment in terms of the location, movement, sound levels, temperature, humidity, etc. As mentioned at the beginning of the paper, the main application scenario addressed by our work is the detection and usage of user contextual in-

formation in order to improve the services provided to the users, accordingly. We focus on collaborative scenarios with decentralized sources of information, where the CEP operators can be executed in a distributed manner on the available devices, such that the system scales with an increase in the number of sources and devices. In this paper, we particularly focus on quasi-stationary environments, where the participation of the devices varies from time to time (low rates of device churn).

3.2 CEP Operator Graph Fundamentals

In general, a CEP system can be described as a combination of producers, who are the sources of the atomic events in the system, as well as the consumers, who are the sinks interested in the higher-level events generated by the system. A set of correlation operators Ω analyze the inherent patterns in the lower-level (atomic) events and generate the higher-level events that the consumers are interested in. Mathematically, we can model the functionality of a (distributed) CEP system by an operator graph $G(\Omega \cup P \cup C, E)$ that connects the producer set P with the consumer set C through event streams $E \subseteq (P \cup \Omega) \times (\Omega \cup C)$.

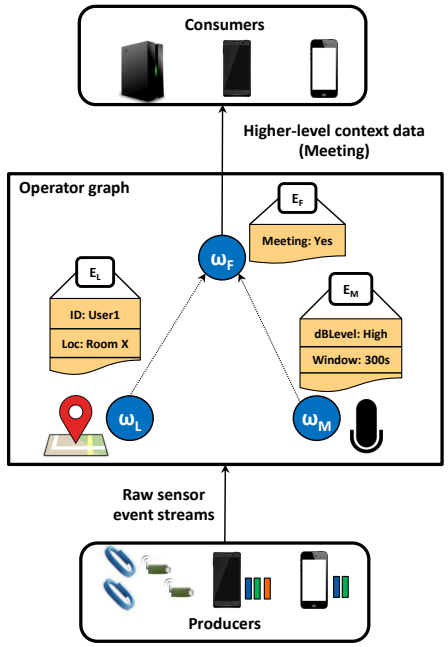

Figure 1: Illustration of a CEP Operator Graph and System Model

Let us consider an example to understand the principle behind the execution of an operator graph. Figure 1 shows the system model and a sample operator graph for the detection of a meeting scenario, based on existing models in related literature [2,16]. The sensor devices (e.g. smartphones, room sensors) providing the GPS, microphone, and accelerometer readings are the producers $p \in P$. The devices obtaining the higher-level events (here, meeting) are the consumers $c \in C$. Here, operator ω_M interprets the decibel levels in the environment based on the incoming microphone readings. Operator ω_L analyzes the incoming GPS data against an open street map to determine the exact location and filters out the data that do not fall in the area under question. (This can also be extended to indoor-positioning systems with the help of WiFi or Bluetooth [15]). Finally, operator ω_F then compares the decibel values and the location information of the user(s) to infer if the scenario is a meeting or not.

We can describe an event e as a tuple of attribute-value pairs such that $e = (att_1, val_1), (att_2, val_2), \ldots, (att_n, val_n)$. Between any two nodes in the system, say origin o and destination d, we consider an *event stream* $(o, d) \in E$ as part of the operator graph, directed from the origin to the destination. Consequently, we term the event stream emerging from an origin node as its *outgoing event stream*, and the event stream going into a destination node as its *incoming event stream*. Each event is assumed to carry a timestamp and a source ID, such that the events are attributed by their origin node(s) as well as their time of occurence.

An operator graph consists of a set of operators $\omega \in \Omega$ that act upon the incoming event streams. For each given operator ω, we describe the set of incoming event streams as $I_\omega \in E$. Thereupon, each operator ω performs a set of correlation operations on I_ω, depending on the prevailing rule model. Finally, the output (complex events) of the correlation operations are inserted into the outgoing event streams $O_\omega \in E$ in temporal order. The mapping function $f_\omega : I_\omega \rightarrow O_\omega$ represents the internal logic of an operator ω.

3.3 CEP Operator State Model

Figure 2 depicts the internal logic of a typical operator used in CEP. The incoming event streams I_ω are stored in an input buffer B_I, until they are processed by the internal correlation function f_ω. Similarly, O_ω are stored in the output buffer B_O, until they are acknowledged by the subsequent operators receiving these events.

The intermediate steps of an operator can be split into three parts—one or more selectors, the internal correlation function, and a sequencer. The selectors analyze I_ω in B_I and determine the set of events σ that satisfy the required pattern for the correlation function f_ω. The events *consumed* by the selectors are then removed from B_I, to make room for the next I_ω. There can be more than one selector depending on the processing logic for a given operator, e.g. filters, aggregators, negators, etc. The events $(ce_1, ce_2, \ldots, ce_n) \subseteq CE \in E$ produced by f_ω are then appended with a sequence number and a timestamp by the sequencer and sent to B_O. We can now describe the correlation function as $f_\omega : \sigma \rightarrow (ce_1, ce_2, \ldots, ce_n)$.

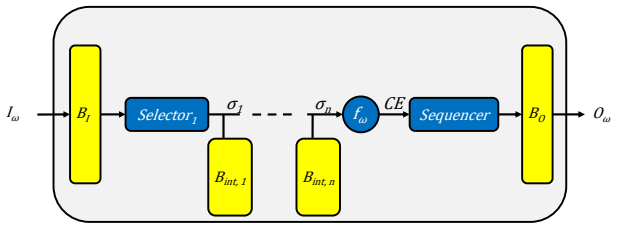

Figure 2: Illustration of a CEP Operator State Model

As a result, the state of an operator ω at time T, $\Psi_\omega(T)$ comprises the states of B_I and B_O, the selectors, f_ω, and the sequencer. In addition to the input and output buffers, we also consider additional *intermediate* buffers B_{int} after each selector, depending on the type of operator and processing

logic behind it. We shall delve into the use of an intermediate buffer in the next section, when we discuss the consequences of operator migration for different operator types.

Basically, any operator migration approach for operator recovery requires the replication of operator state between the active and the backup node(s). The crux of the problem lies in the amount of state information that needs to be replicated and the amount of state information that can be stored and replayed for migration. Different operator types possess different levels of state, making it necessary to analyze operator migration from the perspective of the internal working of an operator.

4. OPERATOR MIGRATION IN D2D BASED NETWORKS

The main goal of our work is to optimize the migration of operators for distributed CEP in a dynamic and resource-constrained environment. To this end, we propose a new approach that accounts for the different intermediate states for the operators in CEP and exploits the same to efficiently facilitate reliable event processing. In the following sections, we discuss the additional challenges that arise in a D2D environment, and present our initial approach to combat these challenges towards efficient operator migration.

4.1 Challenges in D2D Environments

D2D communication is proposed to represent a new class of self-configuring and cooperative communications in environments such as vehicular communication, disaster scenarios, and opportunisitic sensing and networking [22]. They allow for the exchange of information in a local environment, without the need for a centralized server to overlook the operations and maintain resources. In the context of our work, we assume that user smartphones and the neighbouring sensor devices form the foundation of the D2D-based network, such that the CEP operations are primarily executed on the smartphones. To this end, we first address the main challenges posed in D2D communication, especially in the context of operator migration for distributed CEP.

Dynamism. A user environment is inherently quite dynamic and vulnerable to changes, given the movement of users and variance in device availability. The execution of CEP operator graphs is considerably affected by changes in the environment, in terms of the available sensors as well as the processing devices. The operator migration mechanism has to adapt quickly to these changes and restore the processing chain as fast as possible.

Resource Constraints. It is inherent to the nature of mobile devices to have minimal amount of resources. A naïve replication of operator states leads to unnecessary usage of backup devices, especially in static scenarios. On the other hand, a complete rollback recovery requires larger buffer spaces in the order of GB [20] and a larger delay penalty (stalling of the operator graph) upon migration, leading to a higher number of false positives or false negatives [23]. Therefore, the migration of operators has to be adjusted in such a way that unnecessary resource consumption and event loss are minimized.

Constrained System Overview. This challenge results from the decentralized nature of execution in the network. This leads to constrained vision over the connectivity and the performance of the processing devices. This can be overcome either by improving the coordination between the devices, leading to larger number of control messages, or by making use of a coordinator node that has global knowledge of the network [13]. Many D2D scenarios also consider base stations for control messages to the individual devices [22].

In this paper, we primarily address the first two challenges related to the dynamic and resource-constrained nature of the user environment. Given that the existent approaches on operator migration are not suitable for D2D environments, we propose a feasible solution that draws on the concepts of active replication and rollback-recovery by exploiting different parts of the operator state during migration.

4.2 Operator State Migration Approach

As shown in Figure 2, an operator typically consists of an input and output buffer B_I and B_O, selectors, a sequencer, and the actual correlation function f_ω. In most related efforts towards operator migration, the individual parts of an operator and their state information are not considered, leading to unnecessary data transfer and increased buffer maintenance. In our solution, we propose the partial transfer of events and state information based on the operator at hand as well as processing capacity of the nodes involved.

In particular, we exploit the intermediate buffer(s) B_{int} of an operator, introduced in Section 3.3. The selectors analyze the buffer I_ω and extract the events σ as per the processing logic. The selected events σ can include events that contain the required attributes and/or fall in the value range for the given processing logic. The question however remains as to how much state information should be migrated on to the backup node (and how often). Note that we do not focus on the choice of the backup node(s) and the optimal placement of the operators. Interested readers may consult related work on operator placement in distributed networks [14, 21].

Consider Figure 3 which illustrates the transfer of data synchronization messages between the active and backup nodes in the network, as well as the replication of internal operator state at regular intervals. Nodes X (running operator ω_1) and Y (running operator ω_2) are the active nodes that are part of the operator graph, whereas Z acts as the backup node. Events $(X,\ Y) \in E$ flow from the output buffer $B_{O,1}$ of ω_1 to the input buffer $B_{I,2}$ of ω_2. Node Z also maintains an input buffer of an uninstantiated operator for restoration purposes.

We consider the input and output buffers of the nodes in order to adjust the number of event streams to migrate on to the backup node(s). Each event in $B_{O,1}$ is removed as and when its receipt is acknowledged by ω_2. For the purpose of recovery, we maintain an additional buffer $B_{restore}$ on each node, wherein the events that are acknowledged (but not yet consumed) by the subsequent operator (here, ω_2) are stored. The first half of $B_{restore,X}$ is transferred to the input buffer of Z, while the remaining half is retained at X.

The backup buffer $B_{restore,X}$ is updated depending on the processing speed of ω_2. Accordingly, synchronization messages are exchanged among the nodes, in order to discard unnecessary events (that are already processed). The state of the intermediate buffers B_{int} is replicated at Z at regular intervals depending on the processing logic of the corresponding operator, as will be explained below. The structure of the control messages corresponds to the state information that needs to be transferred between devices, as well as the

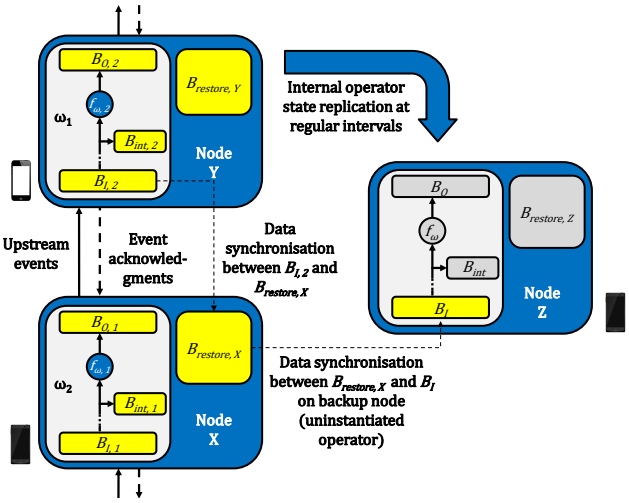

Figure 3: Illustration of Message Exchange and Buffer Maintenance

acknowledgment messages.

4.3 Exploiting the Intermediate Buffer

The above algorithm provides a general overview over the proposed measures in our approach for efficient operator migration in dynamic D2D environments. As mentioned earlier, the particularities of partial state transfer varies among the different CEP operators in use nowadays. In the ensuing discussion, we shall consider some examples of common CEP operations—filtering, (sliding) window-based aggregation, and (batch) window-based sequencing—and discuss the vital state information in each case.

Filtering. Typically, in a filter operator (part of single-item operators), for all incoming event streams I_ω in B_I, one or more attributes are compared against a certain value or a value range to check if they satisfy a condition. The selected events are then put into B_O, so that they may be forwarded to the subsequent operators and/or consumers. In this case, the selector(s) and the sequencer are effectively stateless, given that the correlation function f_ω performs a simple task of sorting out events. Consequently, the internal parts of the operator are completely stateless, and only B_I and B_O will be considered for migration.

(Sliding) Window-based Aggregation. Any window-based operator needs to select only those events in B_I for further processing that fall in a specific time window or lie within a given number of events (count-based). For a fixed window-based aggregation or derivation, the selector compares the timestamps of each event e in B_I and stores those (temporarily) in B_{int}, before forwarding the selected events σ to f_ω, for aggregation or deriving higher-level information. For example, deriving the information that a person is speaking requires microphone readings over a short period of time (say, 30–60 seconds). This example can also be applied for sliding window-based aggregation/derivation, wherein the time window under consideration is moved as and when new events enter B_I. A window-based operator can be further expanded with another selector that compares the attribute values and sorts out events in the form of a filter.

Window-based Sequencing. A window-based sequencing operator looks for certain event patterns in a given time frame (batch). It adheres to a certain sequence (as per the processing logic) while selecting the events for the internal correlation function. In the example in Section 3.2, say the sound events E_M are to be considered only if the location events E_L correspond to be pre-defined location (room X). For the corresponding operator ω_M, the *first* selector sorts out the events that do not fall in the given time window, and puts the selected events σ_1 into $B_{int,1}$. A *second* selector then puts the events in the sequence required (say, a subset of E_L and then a subset of E_M). However, it may so happen that the events arrive in a haphazard order, such that the second selector maintains the partially sequenced set of events in its intermediate buffer $B_{int,2}$, until the events can be reordered and sent to f_ω for processing.

In both of the above cases, from the operator state perspective, the intermediate buffers B_{int} become increasingly important for operator migration on resource-constrained devices. For example, a sliding window-based operator only needs to update the new set of events that were added to the intermediate buffer B_{int} by the corresponding selector, hence reducing the number of control messages between devices. It is not necessary to transfer the entire buffer state each time. The migration of B_{int}, along with B_I, will reduce the amount of computation intensity on the backup node, considerably. In doing so, the backup node may start processing at the same point the active node left off.

5. DISCUSSION

In this paper, we propose the partial transfer of state information by exploiting the intermediate buffers of operators to optimize the migration of operators in a distributed CEP system. In theory, we expect to obtain a better latency/bandwidth tradeoff, given the reduction in the data amount replicated as well as the delay penalty upon restarting an operator on a backup node. This particularly applies for applications where there is a continuous stream of events, such as the scenario described in Section 3.2, where any system downtime cannot be tolerated.

The control messages between the nodes is not restricted to the maintenance messages mentioned above. A new node taking over an operator also entails an adjustment of the underlying associations between the nodes. The amount of buffer space necessary in each of the nodes is also an important evaluation metric. While the resource-constrained nature of the nodes in a D2D scenario is primarily restricted to their battery life-time and their processing capacity, the amount of space available can also be a bottleneck depending on the application at hand.

Furthermore, given the possibility of dense networks in the IoT, D2D communication can be affected by many environmental factors. Devices can obstruct each other, leading to loss of messages and deterioration in system performance. It is essential to choose appropriate operator placement algorithms that take the underlying topology into account.

Generally, reliability cannot be completely guaranteed in a mobile processing environment, given that a network with n nodes can only be made reliable as long as at least k nodes, $k < n$, are available for any operations. However, it can be improved by either reducing the *vertical* dependency within an operator graph, making the operator graph more stateless, or by extending the number of *horizontal* backup nodes available for each operator.

6. CONCLUSION AND FUTURE WORK

The advent of the IoT as well as newer paradigms like D2D communication has opened new avenues for user-centric applications, especially with the proliferation of sensors and sensor-based devices. Distributed CEP provides for a prudent approach to ensure high performance and accuracy for such applications. However, the adoption of CEP in dynamic user scenarios can be problematic unless suitable measures are taken towards operator migration. In this paper, we presented some measures to operator migration in dynamic resource-constrained environments, by exploiting the internal state information of CEP operators. In future work, we shall implement the above approach in a simulation environment and compare it against existing approaches towards reliable CEP. We believe that we can achieve a balance between the bandwidth required for recovery messages and the latency of restoration of an operator graph after migration.

7. ACKNOWLEDGMENTS

This work has been [co-]funded by the Social Link Project within the Loewe Program of Excellence in Research, Hessen, Germany, and by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Centre (CRC) 1053 – MAKI.

8. REFERENCES

[1] O. Bello and S. Zeadally. Intelligent Device-to-Device Communication in the Internet of Things. *IEEE Systems Journal*, 10(3):1172–1182, 2014.

[2] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A Survey of Context Modelling and Reasoning Techniques. *Pervasive and Mobile Computing*, 6(2):161 – 180, 2010.

[3] R. Castro Fernandez, M. Migliavacca, E. Kalyvianaki, and P. Pietzuch. Integrating Scale-Out and Fault-Tolerance in Stream Processing using Operator State Management. In *ACM SIGMOD*, pages 725–736, 2013.

[4] G. Cugola and A. Margara. Deployment Strategies for Distributed Complex Event Processing. *Computing*, 95(2):129–156, 2013.

[5] A. K. Dey, G. D. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-computer interaction*, 16(2):97–166, 2001.

[6] R. Dwarakanath, D. Stingl, and R. Steinmetz. Improving Inter-user Communication: A Technical Survey on Context-aware Communication. *PIK-Praxis der Informationsverarbeitung und Kommunikation*, 38(1-2), 2015.

[7] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A Survey of Rollback-Recovery Protocols in Message-Passing Systems. *ACM Computing Surveys*, 34(3):375–408, 2002.

[8] Y. Gu, Z. Zhang, F. Ye, H. Yang, M. Kim, H. Lei, and Z. Liu. An Empirical Study of High Availability in Stream Processing Systems. In *ACM/IFIP/USENIX Middleware*, page 23, 2009.

[9] T. Heinze, M. Zia, R. Krahn, Z. Jerzak, and C. Fetzer. An Adaptive Replication Scheme for Elastic Data Stream Processing Systems. In *ACM DEBS*, pages 150–161, 2015.

[10] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. Opportunistic Spatio-Temporal Event Processing for Mobile Situation Awareness. In *ACM DEBS*, pages 195–206, 2013.

[11] J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. High-Availability Algorithms for Distributed Stream Processing. In *IEEE ICDE*, pages 779–790, 2005.

[12] J.-H. Hwang, Y. Xing, U. Cetintemel, and S. Zdonik. A Cooperative, Self-Configuring High-Availability Solution for Stream Processing. In *IEEE ICDE*, pages 176–185, 2007.

[13] B. Koldehofe, R. Mayer, U. Ramachandran, K. Rothermel, and M. Völz. Rollback-Recovery without Checkpoints in Distributed Event Processing Systems. In *ACM DEBS*, pages 27–38, 2013.

[14] G. T. Lakshmanan, Y. Li, and R. Strom. Placement Strategies for Internet-Scale Data Stream Systems. *Internet Computing, IEEE*, 12(6):50–60, 2008.

[15] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(6):1067–1080, 2007.

[16] S. W. Loke. On Representing Situations for Context-Aware Pervasive Computing: Six Ways to Tell if You Are in a Meeting. In *IEEE PerCom Workshops*, pages 5 pp.–39, 2006.

[17] L. Militano, G. Araniti, M. Condoluci, I. Farris, and A. Iera. Device-to-Device Communications for 5G Internet of Things. *EAI Endorsed Transactions on Internet of Things*, 15(1), 2015.

[18] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran. MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. In *ACM DEBS*, pages 183–194, 2013.

[19] F. B. Schneider. Implementing Fault-Tolerant Services using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[20] Z. Sebepou and K. Magoutis. CEC: Continuous Eventual Checkpointing for Data Stream Processing Operators. In *IEEE/IFIP DSN*, pages 145–156, 2011.

[21] F. Starks and T. P. Plagemann. Operator Placement for Efficient Distributed Complex Event Processing in MANETs. In *IEEE WiMob Workshop*, pages 83–90, 2015.

[22] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu. Device-to-Device Communication in 5G Cellular Networks: Challenges, Solutions, and Future Directions. *IEEE Communications Magazine*, 52(5):86–92, 2014.

[23] M. Völz, B. Koldehofe, and K. Rothermel. Supporting Strong Reliability for Distributed Complex Event Processing Systems. In *IEEE HPCC*, pages 477–486, 2011.

[24] J. Yang, T. Mo, L. Lim, K.-U. Sattler, and A. Misra. Energy-Efficient Collaborative Query Processing Framework for Mobile Sensing Services. In *IEEE MDM*, pages 147–156, 2013.