# Mobile Distributed Complex Event Processing - Ubi Sumus? Quo Vadimus?

Fabrice Starks, Vera Goebel, Stein Kristiansen and Thomas Plagemann

**Abstract** One important class of applications for the Internet of Things is related to the need to gain timely and continuous situational awareness, like smart cities, automated traffic control, or emergency and rescue operations. Events happening in the real-world need to be detected in real-time based on sensor data and other data sources. Complex Event Processing (CEP) is a technology to detect complex (or composite) events in data streams and has been successfully applied in high volume and high velocity applications like stock market analysis. However, these application domains faced only the challenge of high performance, while the Internet of Things and Mobile Big Data introduce a new set of challenges caused by mobility. This chapter aims to explain these challenges and to give an overview on how they are solved respectively how far state-of-the-art research has advanced to be useful to solve Mobile Big Data problems. At the infrastructure level the main challenge is to trade performance against resource consumption and energy efficiency and operator placement is the most dominant mechanism to address these problems. At the application and consumer level, mobile queries pose a new set of challenges for CEP related to continuously changing positions of consumers and data sources, and the need to adapt the query processing to these changes. Finally, proper methods and tools for systematical testing and reproducible performance evaluation for mobile distributed CEP are needed but not yet available.

Fabrice Starks
University of Oslo, Oslo, e-mail: fabriceb@ifi.uio.no

Vera Goebel
University of Oslo, Oslo e-mail: goebel@ifi.uio.no

Stein Kristiansen
University of Oslo, Oslo e-mail: steikr@ifi.uio.no

Thomas Plagemann
University of Oslo, Oslo e-mail: plageman@ifi.uio.no

## 1.1 Introduction and Motivation

The Internet of Things means pervasive deployment of stationary and mobile sensors, which produce high velocity and high volume data in the form of data streams. A data stream is conceptually an infinite sequence of data tuples comprising typically the digital values of a signal measured by a sensor in the real world and a timestamp denoting when a sensor has generated the value. Real-time analysis of data streams is in Mobile Big Data important for two reasons: (1) the sheer amount of data can make it infeasible to store all data on secondary store and index it before the analysis and (2) many application domains, like smart city, automated traffic control, environmental monitoring, or emergency and rescue operations aim to maintain continuous situational awareness and if certain events happen to react to as fast as possible to them.

One promising technology to achieve situational awareness and detect events of interest in real-time is Complex Event Processing (CEP). The core idea of CEP is to regard the tuples in data streams that are generated by sources like sensors as primitive (also called atomic) events and to extract new knowledge out of the primitive events and represent it as composite events. CEP systems have become rather popular due to the powerful event paradigm and the fact that consumers can describe the composite events they are interested in the form of declarative statements or queries. Originally, the need for real-time processing of data streams, for example in stock trading, triggered the development of CEP systems and a lot of emphasis has been put onto efficiency and scalability of these systems. Naturally, CEP systems have evolved from centralized solutions to distributed solutions to be able to process larger amounts of data in real-time. Most of the DCEP research results and systems target high performance systems with stable infrastructures.

One important challenge for DCEP in Mobile Big Data and the Internet of Things is the fact that one cannot always rely on a stable and high performance infrastructure. Mobility implies the use of wireless networking technologies with potential bandwidth limitations, dependency on battery lifetime in mobile devices, and a dynamic network topology. These challenges are especially severe if infrastructure is not available, e.g., in disaster areas, and multi-hop wireless networks are established for communication. Thus, to use CEP for Mobile Big Data these infrastructure challenges have to be addressed. On the other hand properly designed DCEP can be well suited to address these challenges. For example, source filtering and data aggregation as close as possible to the data sources saves scarce resources, like bandwidth and energy of mobile devices. Furthermore, data aggregation at the network edge has the potential to improve privacy protection. The most important mechanism to address these challenges in DCEP is *operator placement* to determine which data processing tasks should be performed on which node.

Mobile consumers and/or mobile data sources introduce another important challenge for DCEP. In such scenarios, so-called *mobile queries* are traditionally processed in spatio-temporal databases to support for example location aware services, e.g., to provide a car driver continuously updated information about congestions in

the range of 1 km of the drivers current position. Handling properly such spatio-temporal data in CEP systems is a rather new, but important research topic.

Finally, we need to point out that there has been so far no systematic attempt for methods and approaches to evaluate the performance of mobile DCEP in such a way that evaluation results are (easily) reproducible by peer researchers.

It is the aim of this chapter to enable the reader to understand the potential of DCEP for Mobile Big Data and the particular challenges that are introduced by Mobile Big Data. Based on a survey of the state-of-the-art in DCEP we analyze to which extent DCEP is ready for such mobile environments. Finally, we provide the reader with an insight into the main unsolved technical issues and future research directions in the area. Several papers have captured the state-of-the-art in the area of Data Stream Managements Systems and CEP, but to the best of our knowledge there are no surveys on DCEP and especially not on DCEP in the mobile context.

The reminder of this book chapter is structured as follows. In Section 1.2 we provide some background information on CEP and DCEP, followed by an analysis of the main challenges for DCEP in Mobile Big Data. Section 1.4 presents the operator placement problem and classifies existing solutions and Section 1.5 focuses on the challenges introduced by mobile consumer, mobile data sources, and mobile queries to handle spatio-temporal data; and Section 1.6 discusses the needs for proper testing and performance evaluation methods and approaches. The conclusions in Section 1.7 summarize the current status of mobile DCEP research and yet unsolved challenges.

## 1.2 Complex Event Processing Background

Traditionally, database systems have been used to manage large amounts of data, typically by materializing it on secondary storage, e.g., storing it on disks, indexing the data, and providing a declarative Application Programming Interface (API) like SQL for asynchronous data processing on demand. However, the emergence of new applications for sensor networks, Internet traffic analysis, financial tickers, online auctions and analysis of transactional logs from web usage and telephone records introduced in the beginning of this century the need for new software solutions to be able to analyze data streams in real-time [20]. These new software solutions, called Data Stream Management Systems (DSMS), introduced the concept of data streams. A *data stream* is basically a continuous, ordered sequence of data tuples. Conceptually, data streams are similar to classical database tables. Furthermore, the concept of classical database queries has been adopted to run continuous queries over data streams to return continuously new results as new data tuples arrive. The query languages for DSMS, called Continuous Query Language (CQL), are very similar to SQL. The main difference between CQL and SQL is the need to use windows over the data stream for processing. Blocking operators, like aggregations or joins, introduce this need because they can only be used with the entire data set to produce a resultd. It is in most cases not feasible to wait until the data stream

finishes, which means the entire data set is available. Therefore, windows are used to process subsets of the data, which in turn is a number of sequential data tuples from the data stream. The size of a window is either defined by time or by number of data tuples that should be processed from a data stream at a time (per query result). Once the set of samples in a window is processed, the result for this window is returned and the window is forwarded over the data stream and processed again with the new sample(s). DSMS are capable of querying several streaming sources at once, and additionally joining and correlating them in real-time. Large queries can be split into smaller queries and easily processed in a distributed manner, since a query usually results in another stream that can be sent to another query for further analysis. Examples of DSMS include SQLstream [1], STREAM [7], AURORA [4], StreamGlobe [46] and Esper [2].

Esper is also a good example how new achievements in data stream processing lead to a new class of systems, called Complex Event Processing (CEP) systems, with even stronger abstractions and stronger stream processing capabilities. These innovations are based on the concept of events. An event can intuitively be defined as *something that happend* and is either an atomic event or a composite event (also called complex event). In probability theory, an atomic event (also called elementary event or simple event) is a subset of the sample space that only contains a single outcome. In computer science an atomic event is often understood as an event that can be detected by a system within a minimum time period and cannot be divided into other events. The authors understand an atomic event as a single sample from a sensor measuring a signal in the real world, or it is a transformation of an atomic event. For example, a sample from a sensor measuring temperature in degrees of Celsius is an atomic event, as well as a later transformation of this sample into a corresponding value in degrees of Fahrenheit. A composite event is the result of processing a set of events that are combined with operators, like statistical, logical, temporal, or spatial operators. The basic idea is that application programmers define the event they are interested in and the CEP system is analyzing in real-time the incoming event stream(s) and informs the application as soon as it detected the event of interest. Examples of existing CEP systems are SQLstream [1], StreamInsight [27], EVAM [3], or Esper [2].
DSMS [8] and CEP [32] have common goals, but the systems differ in many aspects: architecture, data models, rule languages, and processing mechanisms [16]. Furthermore, DSMS and CEP have their roots in different research communities: DSMS have their roots in the data base systems community, whereas CEP has evolved from Publish/Subscribe systems [19].

The main difference between DSMS and CEP is according to [16] that data items are considered as streams of data versus notifications of events. This means that DSMS handle the Information Flow Processing problem as processing streams of data, which originate from different sources in order to produce new data streams as output. DSMS deal with transient data that is continuously updated executing continuous (standing) queries over the stream items.

In contrast, CEP considers data items as notifications of events. Events are happening in the physical world, which have to be filtered and combined to understand

what is happening in terms of higher-level events. The focus of CEP is to detect occurrences of particular patterns of (low-level) events that represent the higher-level events. The occurrence of higher-level events has to be notified to consumers that have subscribed to these events, typically by registering a continuous query to the system that describes the patterns of events. This relationship between the CEP system and consumers is inherited from the simpler form in Publish/Subscribe systems. Traditional Publish/Subscribe systems consider each event separately from the others, and support topic or content filtering to determine whether a notification should be send to a subscriber. CEP systems have much more expressive subscription languages, e.g., CQL, to describe composite event patterns. Typically, mathematical, logical temporal and spatial relationships can be used to describe these composite event patterns. If A and B are two different events (for example a tuple in a data stream has the value A respectively B) the following composite event patterns could be described:

- $A \wedge B$: the logical $\wedge$ operator can be combined with a time window during which A and B must happen.
- $A \vee B$ : the logical $\vee$ can be combined with a time window during which either A or B must happen.
- $A \rightarrow B$: the temporal operator $\rightarrow$ defines that A must happen before B. This operator can also be combined with a time window.
- $A <> B$: a spatial location operator which defines that the location where A happens and the location where B happens overlap.

Another important difference between DSMS and CEP is the fact that CEP is stateful and DSMS stateless. DSMS use windows to enable the use of blocking operators. A window determines one particular sequence of data tuples. Once all tuples in a window are processed, the result is forwarded as output in DSMS. Therefore, DSMS are not able to detect specific sequences of events in an event stream. To be able to detect specific event sequences in CEP, they are typically built on a state machine and use so called *selection* and *consumption policies* to determine which events to consider for processing. Events that are part of a given event sequence trigger a state transition in this state machine until the final state is reached and the event pattern is detected. Selection policies determine which incoming events are used during processing and consumption policies determine what to do with events that have been processed, e.g., whether to evict an event from the memory or to re-use it in the next processing iteration.

The step from centralized CEP to distributed CEP (DCEP) has two main reasons: (1) parallelizing CEP engines to scale the performance of CEP and beeing able to process more data in real-time, and (2) the fact that recent CEP application domains like environmental monitoring or smart cities comprise a large number of distributed information sources, e.g., sensors and information sinks, human consumers or control systems [16]. Etzion et al. [18] structure channel based event distribution into event producers, event channels, and consumers. Event channels can be an intermediary service that is often called broker. As such, a DCEP engine can be seen as

a set of event brokers that are connected in an overlay network, also called *event processing network* [16].

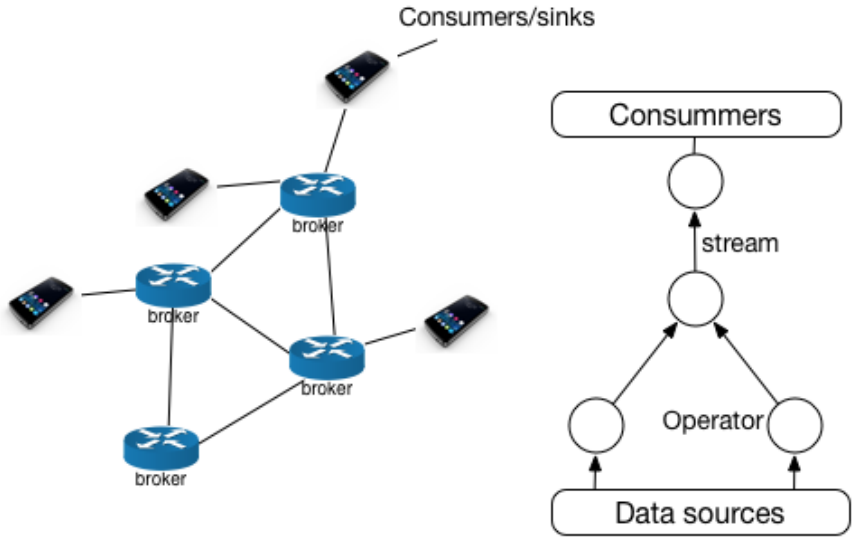


**Fig. 1.1** System model and CEP operator tree (according to [28]).

Koldehofe et al. [28] model the operation of a DCEP system by an operator graph (see Figure 1.1). The operator graph is a directed graph with three different types of nodes: operators, sources, and consumers; and the links are event streams between operators, sources, and consumers. Each operator is hosted by some broker and implements a correlation function which defines the mapping of input events of the operator to outgoing event stream. Operator placement is the task to assign each operator to a broker in the event processing network (or operator network). The event processing network implements specialized routing and forwarding and aims at high scalability for high performance CEP. Therefore, a lot of DCEP research has aimed to optimize bandwidth utilization and end-to-end latency, which are usually ignored in DSMS [16]. For mobile DCEP many more optimization parameters are important, like energy consumption or security constraints, and are discussed in detail in Section 1.4.

## 1.3 Requirements for Mobile DCEP

It is well known that the main challenges of Big Data are caused by the volume, velocity, variety, and veracity of the data. Mobility in Mobile Big Data adds another

dimension to this problem domain. In order to understand the challenges mobility introduces for DCEP we consider mobility from two viewpoints: (1) the impact of mobility on the computing infrastructure and (2) from the applications respectively consumers point of view. We assume, without loss of generality, that the goal of CEP applications is to provide timely situational awareness to consumers.

Mobile devices, like sensors, smart phones, tablets, laptops, and other computing devices obviously require the use of wireless networking technology and need to be battery driven. There are two basic classes of networking approaches that are used to connect mobile devices, which are often called *infrastructure-based* and *infrastructure-less*. In infrastructure-based approaches only the edge of the network to which the mobile devices connect is wireless, typically a cellular network (e.g., 3G, 4G, and the future 5G), or a WiFi network. These wireless edge networks are connected with the Internet by a wired network infrastructure. In infrastructure-less networks, computing devices form with their wireless networking interfaces in promiscuous mode a multi-hop wireless network like a Mobile Ad-Hoc Networks (MANET), Wireless Sensor Networks (WSN), or Vehicle Area Networks (VANET). Obviously, there are many combinations of these two classes of networking possible, but these two are sufficient to identify the challenges caused by mobile devices. The fundamental mechanism in DCEP to address these challenges is operator placement. Section 1.4 gives an explanation and definition of the operator placement problem, as well as a classification of state-of-the-art solutions of operator placement for mobile DCEP.

Before discussing these infrastructure-related challenges we first aim to give the reader an intuitive understanding of the issues caused by consumer and application needs in mobile settings due to the spatio-temporal nature of data that needs to be handled. Spatio-temporal means for example that objects have a location, i.e. the spatial property of an object, and that moving objects change their location over time, which in turn is a spatio-temporal aspect of moving objects. Moving objects can have the role of data sources, e.g., a car that continuously reports its location and other sensor data collected by the car; or moving objects can have the role of consumers. Consider for example a service that is using data from road and parking lot sensors to give the moving consumer in real-time information on free parking lots in the vicinity of the consumers. To provide the consumer with this information only data from sensors in the vicinity of the user need to be analyzed. The query to produce the information for the service is continuously running while the location of the mobile user is changing. Due to the change in user location, the set of sensors that are in the vicinity of the user is changing. Such a *mobile query* requires to continuously adapt the set of sensors to be used. Additionally, many users typically use such a service at the same time; some might be at very distant locations and some closer to each other. In the latter case, the sets of relevant sensors for the users that are currently close to each other overlap. Researchers face the problem of how to avoid that the common subset of sensor data is transferred and processed multiple times, because redundancy reduction means resource savings, in terms of bandwidth, computational capacity, or energy. The fact that these users have different mobility patterns makes this kind of redundancy reduction harder since the

common subset of relevant sensors is continuously changing. In Section 1.5, we explain in more detail the issues DCEP needs to address to support mobile queries, and to properly and efficiently handle spatio-temporal data.

The need of efficient data handling and careful resource consumption is directly implied by the use of mobile devices. Mobile devices are battery driven and one important research goal in mobile wireless networks in general is to use the limited amount of energy in the battery as good as possible. Recharging of batteries or changing of batteries (like in wireless sensors) is in in the best case cumbersome and in the case of WSN potentially very expensive. Therefore, energy efficiency is the ultimate goal in WSN, rather important in MANETs since it directly relates to the lifetime of the MANET, and of less importance in VANETs since the engine of a car can continuously charge the battery. In case of infrastructure-based networks the device owner is confronted with the consequences of battery lifetime and the need for recharging.

The fact that transmit and receive operations of mobile devices are substantially contributing to their energy consumption implies directly to design solutions that carefully handle networking resources, both in terms of bytes per second transmitted (i.e., bandwidth consumption) and packets per second. Another reason to consider bandwidth consumption is the fact that wireless networks are based on a shared medium with a limited amount of bandwidth. Wireless networks can also be affected by noise, high rates of packet collisions and unstable connectivity due to (too) long distances between sender and receiver, which in turn can result in higher packet loss rate and lower bandwidth.

This situation results in a rather large set of conflicting requirements for design, implementation, and deployment of mobile DCEP.

- Low event delivery delay is important to enable situational awareness for the consumer and to initiate immediately certain actions to react to detected events of interest. The potentially large amounts of data that need to be handled increase this challenge.
- Complete and consistent results are needed to achieve correct situational awareness. This means for example that a DCEP system needs to guarantee a high event delivery ratio with a high Quality of Information.
- Efficient resource consumption in terms of computational costs, network utilization, and even monetary costs (if it is necessary to buy resources) is important for several reasons: to achieve low cost services for consumers, saving energy consumed for computational and networking tasks, and being able to handle as good as possible high volume and high velocity data.
- Enable scalable solutions to handle the ever increasing amount of data sources, different consumer interests and volume of data. On the architectural level, distributed and parallel processing needs to be supported. On the application level flexible concepts and corresponding support for Quality of Information need to be supported such that in case of too high system load the Quality of Information can be degraded to a certain level and still acceptable results can be produced, e.g., through load shedding.

- Reliability and fault-tolerance is important especially if the situation awareness is be used for crucial tasks, like traffic control or industrial control systems. Infrastructure components can be prone to hardware and software failures, packets can be lost in wireless networks due to noise, mobile devices might be turned off due to empty batteries, connections might be lost, or even networks might be partitioned.

Operator placement is a rather powerful mechanism in DCEP and can be used to address several of the above-mentioned requirements.

## 1.4 Operator Placement

The classical approach for data mining is to send all data to a central server and to process it on the server. However, in Internet of Things applications that establish situational awareness typically only a particular subset of the data is of interest for the application. Sending irrelevant data to the server is obviously a waste of resources. A simple but efficient approach to reduce resource consumption and enable scalable mobile DCEP systems is to filter events as close as possible to their data source, in the best case, directly at the source, i.e., source filtering. Source filtering is the first step to minimize the consumption of shared resources by eliminating irrelevant events at their sources. The next step is to perform the aggregation and matching of the events in the vicinities of their sources [50]. Processing events near their sources, referred to as *in-network processing*, filters out events that are not of interest and eliminates duplicates early, which in turn reduces system bandwidth and energy consumption, which is especially important in wireless networks. In-network processing takes advantage of increasingly powerful fixed and mobile devices in wireless edge networks. However, the heterogeneity, resource limitations, privacy, security and other challenges related to these edge networks makes in-network processing intricate to implement.

The basic idea behind in-network processing in CEP is that queries are transformed into an operator tree, and that the operators in the tree structure can be processed independently on event brokers. The operators are assigned to brokers in such a way that the performance goals of the system are achieved. Once placed on the brokers, the operators are processed in a CEP overlay called *operator network* [43].

An operator network is a class of overlay networks used for data stream and event stream in-network processing. Operators assigned to physical hosts form an overlay network, which process data from distributed data sources. Results from the operator network data processing are delivered to user applications which are hosted to physical host(s) called sinks.

In large scale operator networks, the physical hosts to which operators are assigned have a significant and direct impact on the performance of the entire system [42]. The operator placement mechanism is responsible for building and maintaining the operator network through operator placement and adaptation and has, still, an important role in the optimization of the system performance.

Due to its importance in DCEP, operator placement is the most investigated mechanism in DCEP related research and its description is correspondingly prominent in this book chapter. In the following, we first give a more in-depth, but informal explanation of operator placement before we formulate the operator placement problem as multi-dimensional optimization problem in Section 1.4.2. The formal problem definition represents also the foundation for a classification of existing operator placement research. The structure of existing operator placement mechanisms, i.e., centralized and decentralized operator placement is explained in Section 1.4.3 and operator placement adaptation is explained and classified in Section 1.4.4.

### *1.4.1 General Idea*

An operator placement mechanism is concerned with how to optimally assign a set of operators to brokers. The goal of an operator placement mechanism is to build an operator network, which optimizes resource consumption and achieves the performance targets of the system. As an example, in Mobile DCEP systems, the operator network would need to optimize the consumption of shared and scarce system resources such as bandwidth while ensuring the performance in terms of low latency.

To achieve its goal, the placement mechanism is provided with the following information:

- an operator tree,
- a set of physical hosts with stream processing capability, i.e., a set of brokers,
- resource availability and demand profiles, and
- a set of constraints.

The operator tree is an internal system representation of a CEP query ready to be assigned to brokers. Figure 1.2 shows a simple operator network to process $(A \wedge B) \vee (C \wedge D)$. Some operators in the operator tree are intuitively pre-assigned to specific brokers. The leaves of the operator tree are typically placed on their respective data sources. In Figure 1.2 the filters for atomic events A, B , C and D are pre-assigned to the sources of the data streams they are supposed to filter, i.e., atomic events that match A, B, C, or D. As an example, the temperature sampling operator should only be placed on nodes with temperature sensors. The output of the root in the operator tree is forwarded directly to the node hosting the CEP application. In Figure 1.2 the root operator $\vee$ placed on Broker 1 forwards its output to the application node: the Command and Control Center (CCC) in this case. Other operators can be bound to specific brokers for monetary, privacy or security reasons. We refer to the operators with predefined placement assignment as *pinned operators*. The remaining operators are referred to as *unpinned* and are assigned to brokers by the placement algorithm.

It is typical to differentiate between physical hosts in the operator network based on their role, i.e., data source nodes, brokers, and sink(s).

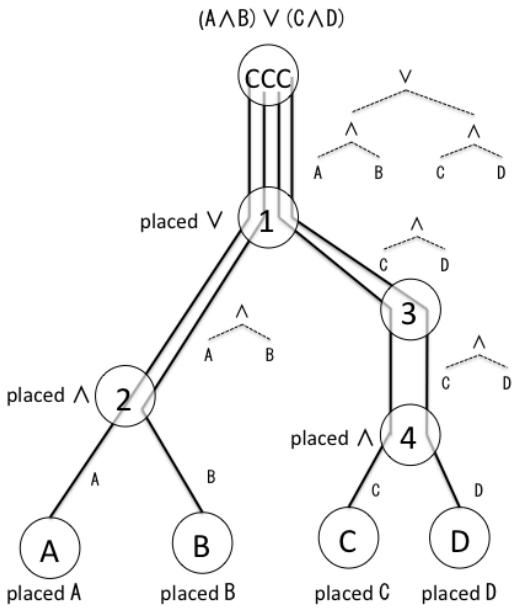


**Fig. 1.2** Operator network in a MANET for an Emergency and Rescue Mission: The circles represent physical hosts. the sink (ccc), the event brokers (1, 2, 3, 4) and data sources labeled with their corresponding pinned operators (A, B, C, D). The unpinned operators are placed besides their processors/ event brokers. Events traversing the operator network edges are results from sub-trees in the operator tree. The edges in the operator tree are labeled with corresponding sub-trees

The data source nodes generate atomic events for the DCEP system and therefore are pre-assigned the leaves of the operator tree. In Figure 1.2, the nodes A, B, C and D are data sources for the corresponding leaves in the operator tree. Data source nodes can also process other operators in the operator graph. The brokers are those which are eligible to process unpinned operators (nodes 1, 2, 3, 4 in Figure 1.2) and sink(s) (node CCC in Figure 1.2) are nodes which have a direct connection to CEP application(s) and are responsible for submitting queries to the DCEP system. A sink is a typical location to place the root an operator tree. Notice that It is possible for a single node to process several operators of an operator graph.

The output of an operator placement mechanism is an operator placement scheme which is a blueprint for an operator network. An operator placement algorithm can build the operator network in a centralized or decentralized manner. Furthermore, most operator placement algorithms implement an adaptation strategy in order to maintain the desired performance as the system and its environment change. The underlying problem of assigning a set of operators to a set of processing nodes has been found to be NP-Complete. However, heuristics based algorithms can be used to find placement solutions in large scale scenarios [10].

In essence, the operator placement problem is an optimization problem. It aims to find an optimal query processing scheme which yields an optimal system performance and resource consumption within certain system or application constraints. While the performance of CEP applications is a priority, the operator placement mechanism needs to find an optimal resource consumption scheme in order to ensure the scalability of the system. More so, in some systems, the consumption of a system's resources such as energy, has a direct impact on how long the system remains operational.

More so, in some environments, the consumption of system resource determines how long it can remain operational.

The optimal placement assignment scheme is found within the predefined constraints provided to the placement mechanism [29,41]. An example of an application-defined constraint is the maximum allowed end-to-end latency. Such a constraint defines the solution space for the placement mechanism and the latter typically use an objective function to find the optimal placement assignment solution.

Finding the optimal operator placement assignment for DCEP system involves two main activities. The first activity is concerned with defining the main optimization metrics for a system and formulating a constrained or unconstrained optimization function. The second activity is concerned with creating an algorithm that effectively solves the optimization function and finds an optimal placement for an operator graph.

In the next section, we formally define the operator placement problem and explore the main optimization goals addressed in existing research along with examples for illustration. Afterwards, we investigate existing placement algorithm design characteristics and adaptation approaches.

### *1.4.2 Problem Formulation*

The operator placement problem is an optimization problem similar to the task assignment problem. Given a set of operators and nodes on which they can be processed, the optimal assignment that yields the best system performance should be determined. Existing operator placement algorithms try to solve either a constrained or unconstrained placement optimization problem. Constrained placement algorithms consider the optimization constraints as a means to ensure some QoS for the application-perceived performance [41]. However, it is also possible to apply resource consumption-related constraints to the placement optimization problem. This ensures an efficient usage of the system's shared resources in order to achieve high scalability and longer lifespan (when applicable). Other placement algorithms solve an unconstrained optimization problem with just an objective function to optimize.

An objective function, which captures relevant system performance and resource consumption metrics, is used to determine the optimal solution. The objective function typically defines critical resources and performance metrics to optimize.

For example, given $N$ processing nodes available for processing $O$ operators, the cost of processing an operator $o$ on a node $n$ is: $C_{(o,n)}$ for $o = 1,...,O$ and $n = 1,....,N$. If we consider $P_{(o,n)}$ as the assignment of operator $o$ to node $n$, the objective function is defined as follows:

$$min \sum_{o=1}^{O} \sum_{n=1}^{N} C_{on} P_{on} \quad : \quad P_{on} \in \{0,1\} \tag{1.1}$$

where $P_{on} = 1$ when operator $o$ is placed on node $n$, and $P_{on} = 0$ otherwise.
In this particular case, the objective is to minimize the overall cost of processing all operators from an operator graph. Other examples of objectives are end-to-end-latency, energy consumption, etc.

The optimization objective and constraints are used to model the targeted system performance. Consequently, they reflect aspects of the challenges faced by the system and its overall performance goals.

In particular, the constraints are used to define boundaries for allowed resource consumption and application performance schemes. They determine the placement assignment solution space from which the optimal solution is to be selected. As an example, for real time data stream systems, timeliness is a pre-requisite to function appropriately. End-to-end latency constraints can be applied on the optimization problem in order to ensure a maximum end-to-end delay.

Using information about system resource availability and application demands for such resources, constraints for the placement assignment problem can be defined to ensure a certain degree of application performance while containing the consumption of system resources within acceptable levels for the scalability of the system. It is also possible to define constraints that enforce policies related to privacy, security, etc. For example, Cipriano et al. [15] consider security as a deployment constraint, which requires that only physical nodes that hold a certain certificate can serve as brokers.

Objective functions can be used with or without constraints. The definition of the objective function is the first step in the process towards creating an efficient and effective operator network, because the parameters in the objective function reflect the critical resources or performance metrics that should be optimized. The objective is a quantitative measure of the performance targets of the system that needs to be maximized or minimized. Obviously, different systems have different performance targets, which are determined by either the application performance requirements or the scarcity of certain system resources. For example, the performance goal of the system might be to minimize end-to-end latency in cases with real-time applications such as CEP. In other cases, the main goal might be to minimize the consumption of scarce resource in order to ensure the scalability of the system.

In the following subsections we present how the most important parameters, i.e., energy consumption and network usage, are included in objective functions and considered in constrained placement, before we use the parameters to classify existing placement solutions.

#### 1.4.2.1 Energy Consumption

The first group of research works focuses on the issue of energy scarcity in WSN. Energy consumption optimization stands out as a critical part of the design, deployment and operation of WSNs [6, 31]. Data transmission has been found to be the biggest energy consumer, therefore, most research papers on operator placement in WSNs focus on minimizing data transmission over the network with in-network processing [9, 11–13, 31, 49]. As such the cost function to minimize is defined as:

$$\sum_{a \in A} d_a C_{a_t}^{a_h} \tag{1.2}$$

where $A$ is the set of all links in the operator network and $d_a$ is the data rate on link $a \quad \forall a \in A$. $C_{a_t}^{a_h}$ is the communication cost on the link $a$ where $a_h$ and $a_t$ are the ingress and egress operators of the link. Given an operator network link between two operators, the placement mechanism should place the two operators such that the communication cost is minimized, especially if the data rate between the two operators is high. Please take note that the operator network link might comprise at the physical network level several nodes and the links between them.

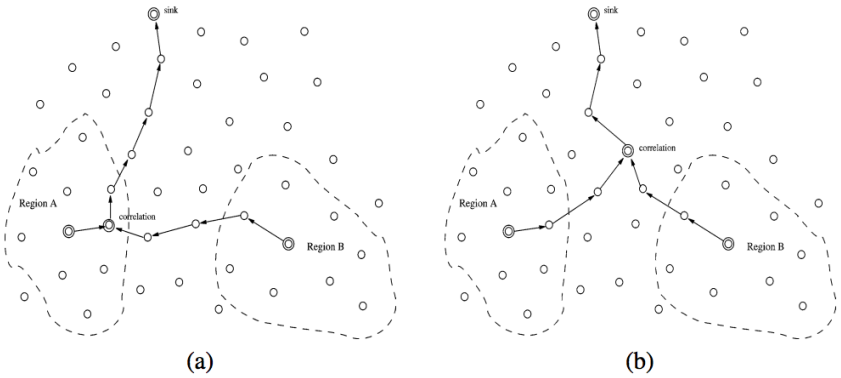


**Fig. 1.3** optimal operator placement examples for different data rate scenarios: (a) high data rate from region A , (b) more or less similar data rates from the two regions [9]

One example where the problem of operator placement for energy optimization is addressed is the research work of Bonfils et al. [9]. Their optimization goal is to minimize the amount of data transferred over the network in order to minimize network energy consumption. For example, a user wants to be notified when two related events are detected in two distinct regions of the network within a predefined time window. This is expressed using a correlation operator, which consumes the related events from the two regions. In this scenario, the data sources reside in the two regions and the sink consumes events produced by the correlation operator. A

correlation operator is very selective, which means that it produces a significantly lower amount of data compared to its data input. As such, its placement is crucial for the amount of data transmitted in the network. Ideally, the correlation operator should be pushed close to the data sources in order to eliminate duplicates as soon as possible. Figure 1.3 [9] shows two cases with to different placements of the correlation operator that each minimize data transmission, depending on the data rate from the data sources. In Figure 1.3-(a), one of the regions is generating significantly large amount of data compared to the other region. Therefore, it make sense to place the correlation operator close to the data source in the high data rate region, to minimize the overall network data transmission. In 1.3-(b), both regions are producing approximately the same amount of data, therefore, the path length between the data sources is considered instead. Thus, the placement of the correlation operator depends on: (1) the data rate of both the operator and the data sources, and (2) the path length between the data sources, the correlation operator and the sink.

Consequently, the placement optimization problem in [9] captures the data rate and path length between an operator and each one of its children as optimization goals. They consider a sensor network as a directed graph where vertices represent sensor nodes and where edges represent communication links, and a query is a operator graph with a tree structure [9]. The placement problem is modeled as the assignment of operators onto nodes that minimizes the global cost:

$$min \sum_{(i,j)\in\lambda} x_{ip} x_{jq} S_{pq}(d_{ij}) \tag{1.3}$$

subject to

$$\sum_{p\in\pi} x_{ip} = 1, \quad \forall i \in \eta, \forall p \in \pi : x_{ip} \in \{0,1\} \tag{1.4}$$

$\lambda$ is the set of all edges in the operator tree. $S_{pq}(d_{ij})$ is the data rate between nodes $p$ and $q$ processing operators $i$ and $j$ respectively. $x_{ip} = 1$ if operator $i$ is placed on node $p$, and $x_{ip} = 0$ otherwise. $\eta$ is the set of all operators in the operator tree and $\pi$ is the set of all physical hosts.

#### 1.4.2.2 Network Usage

Most CEP systems are either real time or near real time, which means that low latency is an important metric that should be part of the objective to optimize. Another characteristic of mobile DCEP is the high amount of data, which requires significant system resources. In particular, the shared bandwidth in mobile systems becomes a scarce resource, and its consumption must be optimized to achieve the expected degree of scalability.

As such, the placement mechanism needs to find an optimal placement assignment that achieves the right balance between optimal bandwidth consumption for

scalability and minimal end-to-end latency. Some research papers use the bandwidth delay product as the objective to minimize in order to find a resource efficient and low latency query processing scheme [38, 40, 41, 43]. The bandwidth delay product is referred to as the network usage and defined as follows:

$$\sum_{l \in L} dr(l) Lat(l) \tag{1.5}$$

Where $L$ is the set of all links in the operator network, $dr(l)$ is the data rate on link $l$ in the operator network, and $Lat(l)$ is the delay on link $l$. The objective above includes both the scarce resource and the main performance metrics to optimize. By minimizing such an objective, it is possible to find an optimal solution both in terms of bandwidth consumption and end-to-end latency.

Pietzuch et al. [40] present a placement algorithm that aims to minimize the network usage of a query processing scheme while maintaining low latency. The goal for this algorithm is twofold: on the one hand it should achieve good stream application perceived performance such as low delay, and on the other hand it should at the same time optimize the consumption of scarce resources by minimizing the bandwidth consumption in order to support a large number of streams.

Satisfying the application performance needs while minimizing the overall bandwidth usage is particularly challenging as techniques to optimize one can produce sub-optimal performance for the other. In particular, a technique to minimize the application perceived delay would choose the shortest paths between data sources and consumers and use them to transfer all data between them. One technique to optimize bandwidth consumption is to balance bandwidth usage in the network by routing data through potentially longer routes in order to distribute the network load. When choosing only the shortest paths, certain links in the network will quickly be overloaded with data and either fail (node failure due to lack of battery energy) or start dropping all data.

The bandwidth delay product (network usage) metric is used in [40] to model an objective function to calculate the optimal solution in terms of bandwidth utilization and end-to-end latency. The network usage $u(q)$ to minimize is modeled as in Equation 1.5.

#### 1.4.2.3 Constrained Optimization

Another way to consider network related parameters like latency and bandwidth consumption and other parameters in operator placement is to use these parameters as constraints. For example, a maximum allowed latency can be expressed as a constraint for the operator placement mechanism [41]. The constraint defines a maximum allowed end-to-end latency which effectively reduces the set of eligible placement assignment solutions. Only those placement assignment solutions with an end-to-end delay below the predefined maximum are eligible for the optimal solution. This works well with the network usage objective function, as it eliminates solutions with poor end-to-end latency no matter how efficient they might be in

terms of network usage. As such, the solution to the objective expressed in Equation 1.5 is found from placement assignments that meet the following latency restriction:

$$L(G) \leq R \tag{1.6}$$

$L(G)$ is the end-to-end latency experienced by the application and $R$ is the maximum end-to-end latency.

Rizou et al. [41] optimize the network usage within a predefined end-to-end delay constraint. A maximum allowed end-to-end delay is important for real time and near real time applications. The placement problem is addressed in a two-stage approach. In the first stage, an optimal solution is found based on the objective alone. In the second stage, the optimal solution found in the first stage is modified to satisfy the latency constraints while ensuring that the initial network usage is only slightly increased. The unconstrained optimization phase is performed in a centralized manner, while the constrained optimization phase is performed in a distributed manner.

**Table 1.1** A classification of placement mechanisms based on their optimization goals

| Placement mechanism algorithms | Energy optimization | Network usage | Constrained optimization |
|---|---|---|---|
| Lu et al [30] | X | | |
| Ying et al [49] | X | | |
| S. Rizou et al [41] | | X | X |
| Rizou et al [43] | | X | |
| Ottenwälder et al [38] | | X | |
| Pietzuch et al [40] | | X | |
| Bonfils et al [9] | X | | |
| Chatzimilioudis et al [11] | X | | |
| Chatzimilioudis et al [13] | X | | |
| Chatzimilioudis et al [12] | X | | |
| F. Starks et al [45] | X | | |

#### 1.4.2.4 Classification of Placement Mechanisms

The optimization goals are a good foundation for a classification of the most prominent operator placement approaches for Mobile DCEP. Table 1.1 shows the resulting classification. It can be clearly seen in Table 1.1 that most approaches target energy consumption. Energy consumption is considered as the most important optimization goal in WSNs and its optimization is a means to prolong the lifetime of the mobile DCEP systems that are deployed in networks with limited energy.

Network usage is the second most important and natural optimization metric addressed by a significant number of operator placement mechanisms. Its popularity is due to its ability to capture both the limited bandwidth resource and application latency requirements. Furthermore, most research works in this category target mobile networks where energy consumption is not as crucial as in WSN, for example

because it is easier to recharge the battery of a smart phone compared to sensors deployed at remote locations.

Few research papers have yet addressed the placement problem as a constrained optimization problem. This is however a natural next step towards optimal placement schemes to effectively address both the need for efficient resource consumption and low latency in mobile DCEP systems. Furthermore, constraints provide an easy means to implement triggers for placement adaptation (see Section 1.4.4).

### *1.4.3 Algorithm Design*

The main goal of a placement mechanism is to find a placement assignment of an operator graph to networked nodes which optimally satisfies a predefined objective function subject to one or more constraints [29]. The information used to achieve this goal varies in terms of scope and variability. On the one hand, some placement algorithms have access to the entire network topology in addition to workload and resource availability information. This makes it possible to perform placement assignment in a centralized manner [12, 23, 39, 44]. In some edge networks such as MANETs, it has been shown that certain routing protocols such as OLSR are able to maintain a rather complete view of the network topology on each node [17]. This information is stored in the routing table and would be available to a placement mechnism for free, i.e., no extra messages need to be exchanged to use this information. On the other hand, some placement algorithms cannot assume knowledge of the entire network state and resource availability due to various reasons, like the costs are too high to maintain this information, or in delay tolerant networks it might take quite some time to get information from another network partition. These algorithms must perform placement assignment in a decentralized manner based on local information [**?**, 9, 11–13, 31, 38, 40, 41, 43, 49] . As such, the scope of the input data provided to the placement mechanism has a direct impact on its inherent structure. Centralized placement mechanisms rely on a single node with global information about the system to perform placement, while distributed placement algorithms rely on local information to gradually find an optimal placement for the operator graph. Parts of the input data for the placement mechanism are subject to change across time due to mobility and other reasons. Consequently, it is important for a placement mechanism to include an adaptation strategy in order to maintain the target system performance (see Section 1.4.4).

#### 1.4.3.1 Centralized Placement Algorithm

Centralized placement mechanisms perform placement assignment of the entire operator graph on a single node, which is typically the sink [12, 23, 39, 44]. Consequently, the cost of query dissemination is considered insignificant and therefore ignored [12].

It is relatively easy and straightforward for a centralized placement approach to find a global optimal placement assignment [23]. However, such approaches do not scale well in large-scale scenarios even if global resource information is available. Therefore, in cases where network resources availability changes over time, the centralized approach can incur substantial communication overhead and delay and lead to the deterioration of the overall system performance. Consequently, some works apply a two step operator placement approach where the initial centralized placement assignment is iteratively updated towards a good respectively the optimal scheme [23].

To exemplify centralized placement mechanisms, we briefly present the core idea of two centralized placement algorithms introduced by Chatzimilioudis et al. [12]. The first algorithm basically analyses the entire search space for the optimal solution, which is guaranteed to be found. The algorithm uses dynamic programming to build a matrix of operators and all nodes in the network and systematically considers all possible placement assignments. This solution is obviously computationally demanding and inapplicable for large problems. To combat this scalability issue, Chatzimilioudis et al. propose a heuristic-based algorithm which is able to find a near optimal solution. The algorithm has a two stage approach, where the first stage is performed in a centralized manner and the second decentralized. In the first stage, an operator tree is built and used as input to the second stage. In the second stage, the placement of the operators in the evaluation tree is iteratively optimized in a top down manner. This algorithm assumes that each node performing operator placement has knowledge of the entire network. Their evaluation shows an improvement in total query processing cost of 10% to 95% compared to the naive approach. This is due to both the reduced communication cost and near optimal placement assignment from the heuristic based algorithm.

#### 1.4.3.2 Decentralized Placement Algorithm

In a decentralized placement mechanism scheme, the placement assignment is performed based on local information shared between neighbor nodes. The scope of the local information varies from neighboring nodes (one hop neighbors for example) to an entire network cluster.

Some decentralized placement mechanisms start with an initial processing cost exchange between neighbors before proceeding with the actual placement assignment [**?**,49]. In such schemes, all nodes in the network are participating in the cost information exchange. Additionally, approaches such as [**?**,13] allow any node to directly broadcast their cost information in case the local resource availability changes or they can re-broadcast overheard cost information from neighbor node(s). Due to their reliance on flooding techniques, the communication cost for these approaches can quickly dwarf the incentives of in-network processing especially when the rate of change is too high due to mobility or other reasons. One approach which reduces the message overhead related to operator placement is presented in [31]. The proposed placement mechanism uses an area-restricted flooding mechanism in order

to limit the number of network nodes involved in operator placement and therefore reduces the inherent message overhead. However, in a highly dynamic network environment, the need to synchronize cost information between neighbors in order to perform an optimal assignment can quickly incur a high message cost and even fail to converge.

Another approach suggested in [45] uses the location of the data sources that will host the leaves of the operator graph, to direct the distributed placement scheme. Only relevant candidates for processing a part of the operator graph participate in the placement scheme. Relevant candidates are those nodes that are part of the routes from the data sources to the sink(s). Moreover, the decision to place an operator ton a specific network node does not require any synchronization between neighbor nodes. The proposed placement mechanism assumes network knowledge, but it can easily be extended to support only local network information. The main goal of this algorithm is to incur as low overhead for the operator placement as possible and to be able to choose a good placement scheme. However, it does not need to be the optimal placement, because mobility will probably change (and mostly reduce) the performance of a selected operator placement scheme. Therefore, it is more important to have a light-weight operator placement algorithm, which in turn allows to perform a new operator placement with low costs, than to spend a lot of resource to find the optimal placement, which might be sub-optimal after a short time due to mobility.

The approach presented in [13] also aims to reduce the communication cost related to initial placement. The distributed techniques for operator placement achieve this aim by:

- identifying special cases where no flooding is needed to perform placement,
- limiting the size (number of nodes) of the neighborhood to be flooded

The core idea behind the algorithm is the concept of candidate nodes, i.e., physical host in the network, which are better suited to host a given operator. The candidate nodes are elected from a set of neighboring nodes in the network. The set of candidate nodes for a given operator is kept to the minimum (using a cost threshold) in order to limit the number of message exchanged of the network during placement information exchange between them. This effectively reduces the communication cost related to the placement of the operator.

The set of candidate nodes for an operator is created in a centralized manner without network communication, this allows the algorithm to detect special cases where there is no candidate node which is better suited to host the given operator. In this particular case (according to their experiments, 56%-85% of the time, there is no candidate node which is better suited to host the given operator), there is no need to initiate the distributed operator host election algorithm. The radius for flooding during initial neighbor discovery is also limited, and it ensures that the optimal physical host for an operator can be found in the set of nodes that are part of the limited flooding. Results from experiments show a 50% to 100% reduction in the communication cost compared to naive flooding techniques.

### *1.4.4 Placement Adaptation*

Mobile systems are inherently dynamic and changes of all kind can occur, like number of nodes, availability of links, resource availability, data rates, and many more. These changes are classified by [29] in three categories: changes concerning network infrastructures, changes concerning data characteristics, and changes concerning operator tree information. Any of these changes can have a negative impact on the performance of an operator network. A placement adaptation strategy aims to adjust the operator network after a change such that it fulfills again the application requirements.

Changes concerning the network infrastructure represent scenarios where the network topology changes due to node failure, mobile nodes, link failure (due to network congestion or node failure), or new node(s) joining the network [13, 38, 40]. There might also be changes in the local resources for a network node, e.g, the battery might be drained, or other computationally intensive software implies a high workload for a node [5, 13].

Changes conerning the network load happen for example when the data rates from sensors or source filters change, or other background traffic increases. For example, the increase in data rate at the input of one or more operators in the network might result in an increase in the total cost of in-network processing if bandwidth consumption is part of the objective function [9, 11]. The network load can also change due to new application traffic in the network or a change in data rate for other applications using the same network infrastructure.

Changes concerning the operator tree occur when the number of operators changes due to new queries submitted or previous ones are terminated. Additionally, the operator tree might be updated due to changes in the user's interest (location) requiring the adaptation of the corresponding operator network (see Section 1.5).

As the query processing scheme performance deteriorates, the placement adaptation strategy consists in picking new hosting node(s) for one or more operators in the flow graph. Two main approaches are identified in [13]: operator migration and placement update.

With operator migration the placement adaptation for an operator is performed by moving it from one node to another until an optimal placement assignment is found [9,11,35,38,40]. During operator migration, every node involved in the process uses local information exchanged between neighbors to determine which one of them is better suited to host the current operator. The limited scope of the information used makes the approach relatively easy. However, in a highly dynamic environment, it could be difficult for the migration process to converge towards an optimal or even good sub-optimal placement.

The placement update approach aims to find the best host for an operator immediately. This can be done in a centralized manner as in [40], or decentralized [5, 13] manner by reusing initial placement techniques for the single operator instance.

Different approaches are used to determine when to trigger the operator migration or placement update. One approach is to monitor the processing cost related to each operator and exchange this information between neighbors. When a

predefined threshold is reached for a given operator, its migration process is triggered [5, 9, 11, 13, 35, 40]. Other approaches monitor constraints violations in addition to a predefined performance threshold based on the applied objective [38]. Another approach is to periodically trigger the placement adaptation of the entire operator graph based on a predefined time interval. In all cases, an operator migration or placement update will potentially trigger subsequent operator migration(s) or placement updates.

The cost of the actual migration or placement update should be worth the operator placement adaptation, which means that the increase performance of an adapted operator network gives higher benefits that the adaptation costs. This is not always the case as the placement adaptation process requires transferring the state information of all operators that are hosted on a new broker. This state information can be as large as several GBs [38]. As such, in certain scenarios, the migration of placement update for an operator might incur a significant cost, especially in terms of network usage. In some cases, however, the migration or placement update for an operator might be unavoidable, e.g., the battery of the hosting node will soon be depleted. It is also possible to experience a sort of freeze period during placement adaptation or operator migration. The freeze period occurs as the operator and its state are in transit from their previous host towards their new host.

**Table 1.2** Classification table for different adaptation scheme

| Adaptation schemes | Monitored change | | | Adaptation techniques | | Adaptation trigger | |
|---|---|---|---|---|---|---|---|
| | network topology | data rate | logical graph | operator migration | placement update | performance threshold | Constraints violation |
| Oikonomou et al [35] | X | | | X | | X | |
| Bonfils et al [9] | | | X | X | | X | |
| Chatzimilioudis et al [11] | | | X | X | | X | |
| Pietzuch et al [40] | | X | | | X | X | |
| Chatzimilioudis et al [13] | X | X | | | X | X | |
| Z. Abrams et al [5] | | X | | | X | X | |
| Ottenwälder et al [38] | X | | | X | | X | X |

To exemplify operator placement adaptation, we refer to the work by Pietzuch et al. [40]. In this work, a placement update is used to regularly solve the placement optimization problem for each unpinned operator. In particular, every network host regularly attempts to find a better placement assignment for each unpinned operator using local cost information exchanged between neighbor operator host in the operator network.

Predefined threshold(s) are used to determine whether an operator placement update should take place or not. One threshold determines when the difference in performance between the newly found optimal placement and the previous one is high enough to incentivize the placement update. Another threshold determines whether the cost of the placement update is low enough given the expected gains from the new placement assignment. Additionally, the longevity of the query to which the operator belongs is taken into consideration by the latter threshold in order to make sure the query will run long enough to amortize the cost endured by the operator placement update process. The cost thresholds are used to ensure that both the network resources consumed and the operator placement update delay do not cripple the overall system performance. If the placement of operators is updated frequently, the adaptation cost might grow higher then performance gains. Additionally, if the placement of operators is updated for insignificant gains, the overall performance of the system might be degraded.

To evaluate the performance of the placement update scheme, 24 queries are created. The performance of the operator network for each query is evaluated two times, i.e., with adaptation enabled and and without adaptation. Overall results show a 75% decrease in network usage ( see section 1.4.2.2 ) when operator adaptation is enabled. Finally, the aggregated query delay is reduced by 10,5% through adaptation.

While the results show clear gains in terms of both the application perceived performance and system resource consumption, the evaluation system model considered is rather simplistic compared to typical scenarios with Mobile Big Data.

The operator graph comprises only 3 nodes, which introduces some uncertainty whether the results are representative for large scale scenarios for Mobile Big Data. The migration rate experienced in the experiments is in average 3.5 adaptations per query, which indicates that the results are probably not representative for higly dynamic Mobile Big Data systems. However, evaluation of Mobile DCEP with placement adaptation is rather hard, because appropriate methodologies and tools are missing (see Section 1.6).

## 1.5 Mobile Queries

The topic of spatio-temporal data and mobile range queries has been extensively studied in the database community. The overall goal is to provide continuously updated information, typically to a mobile consumer, e.g., the five closest bus stops to the current location of the consumer. The survey by Ilarri et al. [22] gives an excellent overview of challenges and approaches to enable location-dependent query processing in traditional database settings, i.e., the data is materialized on secondary storage before processing. Traditional approaches to store, query, or index spatio-temporal data are insufficient to handle the high data rates and potentially very large data sizes in Mobile Big Data [33]. This insigth motivated researchers to combine the two worlds of traditional spation-temporal data management and Data Stream

Management Systems. The systems [48] and [33] are to the best of our knowledge the first published DSMS with support for moving range queries. Research on CEP support for mobile range queries is still in its infancy and pioneering work is recently published in [36], [37], [38], [21], and [28]; and to a larger part summarized in the Thesis presented by Ottenwalder [38]. Therefore, we base our description of challenges in mobile DCEP introduced by spatio-temporal data issues and new solutions on the terminology and model of [36, 38]. The overall goal of this work is to enable location based situational awareness for consumers in a mobile setting.

To achieve this situational awareness mobile CEP queries, called *MCEP queries*, they need to be registered at the MCEP system. A MCEP query $Q$ has the following structure:

$$Q = \{G, fo, R, , \delta, PoI\} \tag{1.7}$$

$G$ represents an operator graph, $fo$ is focal object of the consumer, $R$ a function to calculate the spatial interest based on $fo$, $\delta$ a lifetime parameter, and $PoI$ the delivery semantics. That means that in case of a mobile focal object $fo$ the function $R$ needs to be recalculated if $fo$ has a new position to adapt the spatial interest, i.e., the region of interest. The function $R$ is by purpose not defined in this model in order to enable regions of interest with arbitrary shapes. As such, a sequence on location updates from $fo$, i.e., $(l_1, l_2, l_3, l_4, l_5, ...)$ results in a sequence of changing spatial interests $(R_1, R_2, R_3, R_4, R_5, ....)$ where $R_i = R(l_i)$ for each $i \in \mathbb{N}$

Ottenwaelder et al. [36] use the example of a traffic awareness in which a consumer is driving a car and aims to avoid traffic jams. As such the consumer is interested in all accidents that happened within the last 30 minutes within 500 meters of the consumers current location. In this case, the consumer or the consumer's car is the focal object $fo$ which continuously reports location updates. The function $R$ calculates each $l_i$ a circle with a radius of 500 meters and $l_I$ as the center. The parameter $\delta$ in the query has the value 30 minutes.

A change of spatial interest from $R_i$ to $R_{i+1}$ requires to update the operator graph $G$ accordingly since the set of sensors that are deployed in $R_i$ and $R_{i+1}$ is typically not equal and the sensors are represented as leaves in $G$. The update of the spatial interest and the following switch to a new operator graph introduces new challenges:

- For traditional CEP systems, the temporal order of events can be for many operators crucial to perform correctly. A change in spatial interest with a swicth of the operator graph implies that in mobile CEP with spatio-temporal data also the spatial order and spatio-temporal order is important. To achieve a spatially ordered event stream all events from $R_i$ need to be delivered before events from $R_{i+1}$ are delivered. A spatio-temporal order requires spatial event order and temporal event order for events from each $R_i$.
- The concepts of consistency and completeness need to be extended for MCEP. Spatial consistency ensures that all nodes in one operator graph process only input data that is based on one region of interest. This can be atomic events stemming from one particular region of interest or composite events that are based on these atomic events. Temporal completeness requires that situational information

for one region of interest is delivered in spatio-temporal ordering for the temporal interest $\delta$. Thus temporal completeness with a large $\delta$ leads to large latency.

- CEP operators can, in contrast to DSMS, be stateful. As such an operator cannot just proceed to process the incoming data after an operator switch. Instead, the operator state that was established when processing input data for $R_i$ needs to be deleted, respectively the operator needs to be restarted.
- To detect events of interests in the new region $R_{i+1}$, historical events, i.e., those that happened before the operator graph switch are useful for two reasons: (1) a window over the input data needs to be filled up before the operator can start processing, which obviously introduces a start-up latency. If historical data is available, the window can be filled up much faster, which in turn reduces the start-up latency. (2) Historical events are useful for the consumer. In the traffic awareness example accidents that happened in the new region of interest $R_{i+1}$ before the switch to $G_{i+1}$ are useful for the consumer, because roads will be congested for some time after the accident.

Any CEP system supporting mobile queries needs to know the location of data sources, consumers, and brokers. The MCEP system [36] comprises a location and performance monitor that continuously monitors the location of data sources and consumers. A location update of a consumer from $l_i$ to $l_{i+1}$ triggers a query configurator which initiates a switch to a new operator graph based on the new region of interest $R_{i+1}$. The data sources in $R_i$ are instructed to stop streaming atomic events, and the set of data sources in $R_{i+1}$ is identified and these data sources are instructed to start streaming atomic events. Please note that there is a high probability that the sets of data sources in $R_i$ and $R_{i+1}$ overlap. To achieve spatial consistency and spatio-temporally ordered results, so-called *markers* are inserted in the event streams. Markers are special messages that separate in each atomic event stream from the data sources that are in $R_i$ and $R_{i+1}$ the atomic events that are relevant for $R_i$ and $R_{i+1}$. The arrival of a marker at an operator implies that now atomic events from a new region arrive. It is possible that the operator is still waiting for atomic events from the old region of interest to achieve completeness. Before processing the atomic events from the new region of interest, the operator is reset to avoid spatial inconsistencies. A marker is inserted in the operators' outgoing event stream before the first event from $R_{i+1}$ is inserted. In this way, markers are inserted by each operator in the operator graph and enable spatial consistency and spatio-temporally ordered results for all operators. Several optimization techniques are leveraged in MCEP to produce timely results. In the proactive version of an operator switch, the future location of the focal object is predicted and the system starts to process historical results for a future region of interest. Once the focal object is in the predicted region of interest, all historical events are already available and processed. Another optimization is related to the overlapping sets of data sources of subsequent regions of interest and reuses the events for processing. The delivery semantics *PoI* are used to specify how to trade Quality of Information against streaming and processing costs.

Earlier work on the SOLE system [33] considers so-called regions of uncertainty which is caused by the fact that the system does not know about all data sources

in a region of interest. The reasons for this uncertainty can be that new queries are installed and no historical data is available, as such it takes some time until the windows are filled and results can be produced. Furthermore, moving queries imply continuously changing regions of interest, which in turn leads to new data sources. This is similar for mobile data sources that move into a region of interest. To handle the last two cases of uncertainty SOLE applies a caching strategy for all moving objects that are predicted to be at some point in time in the region of interest. As such this solution is similar to the proactive approach in MCEP.

Query optimization is a classical research problem in databases and also investigated in spatio-temporal databases. Mobile queries have a huge potential for optimization. Consider the application for a car driver to get continuous information about traffic congestions in the vicinity of the driver. The fact that there is not a single car driver on the road, but instead a large amount of drivers introduces severe scalability issues. Each driver represents one focal object $fo_i$ with its unique location $l_i$. Therefore, $i$ mobile queries need to be executed and $i$ can be very large considering the number of cars that are travelling on roads in major cities during rush hour. Two mechanisms are presented in [38] to address this scalability issue:

- Reuse of processing results: If all car drivers use the same or rather similar mobile queries to achieve situational awareness there is a substantial overlap of the operator graphs that are used to process these queries. The regions of interest of focal objects that are close to each other will also overlap substantially and as such the operators in the different operator graphs will process to a certain degree the same input events.
- Relax the requirement for a fully accurate set of input events to calculate situational awareness. This idea is similar to the well-known technique of load shedding in CEP. By relaxing the need for accurate input data it is possible to increase the number of operator graphs for different focal objects that can share the same set of input events.

The solution presented in [38] is based on a reuse-aware operator graph in which some operator graphs process input events on behalf of other operators and a system component called *selection manager*. The selection manager analyzes the degree to which input events of operators overlap. Given that the overlap is large enough with respect to a predetermined quality metric, the selection needs to be processed only once and can be reused for the other operator graphs.

Combining this kind of query optimization with operator placement could allow for more improvements if the computational complexity and the deployment costs could be kept low enough. However, this challenge is subject to future work.

## 1.6 Mobile DCEP Evaluation

The usefulness of mobile DCEP systems depends on their performance. Due to their large scale and complexity, their performance evaluation constitutes a challenge on

its own, requiring the use of well-established, systematic methodologies. This section introduces the most commonly used techniques for performance evaluation of distributed systems, then summarize how these are used to evaluate mobile DCEP systems.

### *1.6.1 Basic Requirements and Approaches*

We very briefly summarize the common approaches for performance evaluation of distributed systems. Consult [24] for a more elaborate treatment of the subject.

Before we describe the approaches, we explain the desirable properties of the evaluation approach and results. A basis for performance evaluation is sufficiently *representative, accurate and understandable* data that enables a proper analysis of the system under test. With experimental approaches, it is important that the experiments are *repeatable*, e.g., to enable third party verification of the results or to investigate the results of incremental system improvements. Results obtained from systems with similar purposes should furthermore be *comparable*, e.g., by applying similar models and/or (values of) parameters and metrics. To facilitate acquiring results with all these properties, the evaluation techniques and tools should require a *low effort and cost*.

The most representative results are obtained from real world experiments using workloads, configurations and environments similar to that expected during the final deployment. For large-scale and/or complex systems, this approach is often too expensive and time consuming. Instead, evaluation is performed using abstract mathematical or simulation models. The quality of model-based evaluation rests on how well the used model captures the characteristics of the system under test that determine its performance. When this cannot be achieved to a satisfactory degree with mathematical formula, due to system complexity or scale, simulation and emulation provides a popular alternative. Simulation is by far the most common evaluation approach in computer systems' evaluation due to its low cost, support for abstractions that facilitate understanding and a controllable experimentation environment. Emulation combines real and simulated components, e.g., running real applications on virtual network nodes, and thereby benefits from the advantages of both real and simulation experimentation. However, since emulation experiments involve real components, they also suffer from scalability limitations.

### *1.6.2 Performance Evaluation for Mobile DCEP*

This section summarizes 19 performance evaluation reports presented in 13 key publications on mobile DCEP [9–13, 26, 28, 31, 40, 41, 43, 45, 49] in terms of the applied approach, tools, parameters and metrics.

All publications include at least one simulation study, except for two publications based on emulation that employ both simulated and real components [41, 45]. As a result, 14 of 19 evaluation reports are based on either simulation (11 reports) or emulation (three reports). Of the remaining five reports, three are based on real world experiments (in [10, 40]), e.g., with the well-known PlanetLab test bed [14], and two are based on mathematical analysis (in [28, 31]). The fact that simulation is the most popular approach is not surprising since a proper evaluation of placement algorithms typically requires networks with several hundred nodes, making real-world experiments unfeasible. This problem is further exacerbated for wireless networks where the shared medium and node mobility implies a high degree of network dynamicity, making it exceedingly difficult to conduct controlled and repeatable experiments. We find that seven of the simulation experiments are performed with simulators that are created for the specific experiments at hand, and that the remaining four experiments are performed with the popular network simulators J-Sim [25], OMNeT++ [47] and PeerSim [34] using real world or generated topologies, mobility patterns and network traffic as input. Our survey indicates that there exists no common simulation platform to enable the evaluation of mobile DCEP systems in general.

Some reports involve parameters and metrics that cannot readily be found in other reports, e.g., model-specific paramters like the $\alpha$ in [31] and metrics like the *stretch factor* in [43]. For results from such reports to be comparable with those for other similar systems, such parameters and metrics must first be translated. This might be cumbersome or even impossible, limiting the comparability of results. There are, however, metrics that are widely used within a subset of the reports. For instance, solutions for WSN [9, 11–13, 26, 31, 49] address the common challenge of resource constraints on nodes and are thus typically evaluated in terms of energy consumption and/or processing cost. Comparability is however somewhat limited by the fact that the metrics can be defined slightly different between works, or because the applied parameter types and values differ significantly from study to study. For example, the number of nodes in the simulated networks ranges from less than 10 [26] to several hundred [11, 13]. The most common metric for the remaining six works [10,28,40,41,43,45] is *network usage* based on the bandwidth-delay product presented in Section 1.4.2.2. This is nevertheless only used in three of these six works [10, 28, 40] and can therefore hardly be considered as a common ground for comparison. In general, we cannot identify any clear consensus in terms of metrics and parameters that facilitate comparability among different mobile DCEP systems.

### *1.6.3 Future Work on the Evaluation of Mobile DCEP*

Due to factors such as cost and effort, the de-facto standard evaluation approach for mobile DCEP is simulation, mostly with simulators created for the paper at hand. There are several disadvantages of this extensive use of custom simulators. First, their models are not subjected to the rigorous validation that models in more general

purpose simulators are subjected to. Examples of such general purpose simulators include the widely-used, de-facto standard simulators used in the netorking community, i.e., Ns-2, Ns-3 and OMNeT++. These popular simulators have been available for decades during which their models have been subjected to continuous validation to assess realism and comparability. Second, for the experiments to be repeatable, the custom simulators need to be very simple to allow sufficient, yet brief description in an evaluation report. This problem is exacerbated by the fact that the simulators are rarely made available for download online. To accomplish this, the simulations are often based on overly simplified assumptions, e.g., not accounting for complex network phenomena like link interference and bit-error rates, which in turn affects the credibility of the results. Comparability is also compromised since different simulators are based on different models, parameters and metrics that produce results that cannot readily be compared. In contrast, the above mentioned network simulators Ns-2, Ns-3 and OMNeT++ are freely available for download online, and are maintained by a well-established, code review-based developer community that provides a channel through which researchers can contribute and distribute their models world-wide. The mobile DCEP community is a relatively new one compared to the networking community. This might be the reason behind the lack of a corresponding de-facto standard simulator for mobile DCEP. Our findings suggest that such a generic DCEP-simulator, facilitating the evaluation of a wide variety of mobile DCEP solutions, would help improve the quality of the simulation results in terms of *repeatability* and *comparability*. Since the performance of DCEP is largely affected by the characteristics of computer networks, models for a DCEP-simulator would benefit in terms of *accuracy* and *realism* from the reuse of these computer network models. We argue that this is best approached by extending existing network simulators with DCEP-models, rather than vice-versa, in order to benefit from the large base of models, knowledge and support available in the network simulation community.

## 1.7 Summary and Conclusion

CEP is a promising technology to enable situational awareness in real-time in the Internet of Things, because it provides a declarative interface to mobile DCEP application programmers, abstracting away data processing intricacies in the distributed environment. Therefore, we are convinced that mobile DCEP can play an important role in future Mobile Big Data systems.

However, mobile DCEP need to handle unstable infrastructure with limited resources, because mobility implies the use of wireless networking technologies with potential bandwidth limitations, dependency on battery lifetime in mobile devices, and a dynamic network topology. Consequently, effective data handling and efficient resource consumption is a prerequisite for such systems. The most important mechanism to handle these issues in mobile DCEP and to meet application QoS requirements is operator placement.

The classification of the state-of-the-art in operator placement in Section 1.4 shows that most works aim to minimize system resource consumption such as energy and bandwidth. Some researchers address application QoS requirements such as low latency together with efficient system resource consumption. Including constraints in the operator placement, like security concerns, is in its infancy and the potential that operator placement has for privacy protection has to the best of our knowledge not been addressed yet. A lot of decentralized placement mechanism exists, but very few address issues related to a dynamic topology in mobile DCEP. Consequently, none of the solutions proposed is applicable in highly dynamic environments. A lot of work has been done in enabling placement adaptation to deal with change in DCEP systems. The adaptation strategies vary by the monitored change, adaptation techniques applied and adaptation triggers. The network topology, data rate and change in the operator graph are the main elements monitored to determine when it is time to trigger adaptation using predefined performance threshold. Some works consider constraints violation as a means to ensure application QoS through adaptation. Two main adaptation techniques are applied: operator migration and placement adaptation. It is our belief that, to enable QoS for mobile DCEP applications, it is necessary to further study constrained violation based adaptation triggers.

Operator placement itself is also not sufficient for mobile environments, since mobility of devices, including brokers hosting operators, can render an initial and near optimal placement in short term sub-optimal or even result in a very inefficient system. Therefore, an efficient placement adaptation is required to ensure the performance and efficiency of the system.

Mobility does not only cause challenges at the infrastructure level, but also at the user and application level. For example, mobile consumers of location-based services are often interested in events in their vicinity. Due to mobility the region of interest and the sensors in these regions continuously change. To support mobile queries in CEP for this kind of applications mobile DCEP needs to properly handle dynamic data sources or producers, dynamic or mobile range queries, spatio-temporal event ordering, spatial consistency and temporal completeness, CEP operator state transition and management, and location awareness.

There is currently very limited work on mobile queries for mobile DCEP. More research needs to be done to explore all the issues introduced by mobile consumers with location based interests. Furthermore, operator placement techniques need to address challenges introduced by mobile queries in order to ensure efficient and effective event processing networks.

Mobile DCEP is mostly evaluated with either simulation or emulation. The parameters used vary across evaluation reports, making it difficult to compare them. Custom simulators are used for evaluation, making them less reliable compared to more established simulators. Furthermore, the custom simulators are rarely made available to reproduce the results from the evaluations. It appears that the DCEP research community would benefit from a generic DCEP-simulator facilitating the evaluation of a wide variety of mobile DCEP solutions. Such a simulator would enable repeatable experiments with results that are representative of the simulated

system as well as comparable with results from experiments with other systems conducted with the same simulator. Additionally, reusing already existing and mature simulation tools from the networking community would ensure accuracy and realism of mobile DCEP evaluations due to their large scale networking characteristics.

Even thought there are many open research issues in mobile DCEP, this chapter shows that results and useful systems for certain scenarios exist. Since mobility can have many forms it will probably turn out in the future that it is not possible to design one mobile DCEP system that handle all the challenges which are caused by mobility. Instead, proper solutions for cases in which only the edge network is mobile might be earlier ready than proper mobile DCEP solutions for infrastructure less networks.

## References


1. http://sqlstream.com/intro. Accessed: 2016-12-30.
2. http://www.espertech.com/esper/. Accessed: 2016-12-30.
3. http://evam.com/platform/. Accessed: 2016-12-30.
4. Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, August 2003.
5. Z. Abrams and Jie Liu. Greedy is good: On service tree placement for in-network stream processing. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 72–72, 2006.
6. Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.
7. A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R Motwani, U. Srivastava, and J. Widom. Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab, 2004.
8. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM.
9. Boris Jan Bonfils and Philippe Bonnet. Adaptive and decentralized operator placement for in-network query processing. *Telecommunication Systems*, 26(2-4):389–409, 2004.
10. Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, and Matteo Nardelli. Optimal operator placement for distributed stream processing applications. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, DEBS '16, pages 69–80, New York, NY, USA, 2016. ACM.
11. Georgios Chatzimilioudis, Alfredo Cuzzocrea, Dimitrios Gunopulos, and Nikos Mamoulis. A novel distributed framework for optimizing query routing trees in wireless sensor networks via optimal operator placement. *Journal of Computer and System Sciences*, 79(3):349–368, 2013.
12. Georgios Chatzimilioudis, Huseyin Hakkoymaz, Nikos Mamoulis, and Dimitrios Gunopulos. Operator placement for snapshot multi-predicate queries in wireless sensor networks. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 21–30. IEEE, 2009.
13. Georgios Chatzimilioudis, Nikos Mamoulis, and Dimitrios Gunopulos. A distributed technique for dynamic operator placement in wireless sensor networks. In *2010 Eleventh International Conference on Mobile Data Management*, pages 167–176. IEEE, 2010.

14. Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.
15. N. Cipriani, C. Lbbe, and A. Moosbrugger. Exploiting constraints to build a flexible and extensible data stream processing middleware. In *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, April 2010.
16. Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012.
17. O.V. Drugan, T. Plagemann, and E. Munthe-Kaas. Dynamic clustering in sparse {MANETs}. *Computer Communications*, 59:84 – 97, 2015.
18. Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.
19. Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.
20. Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, June 2003.
21. Kirak Hong, David J. Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. Opportunistic spatio-temporal event processing for mobile situation awareness. In *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*, pages 195–206, 2013.
22. Sergio Ilarri, Eduardo Mena, and Arantza Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Comput. Surv.*, 42(3):12:1–12:73, March 2010.
23. N. Jain, R. Biswas, N. Nandiraju, and D. P. Agrawal. Energy aware routing for spatio-temporal queries in sensor networks. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 3, pages 1860–1866 Vol. 3, March 2005.
24. Raj Jain. The art of computer systems performance evaluation. *Wiley Thomas Limoncelli, Christina Hogan, Strata Chaiup, The Practice of System and Network Administration. lSBN-I*, 3:978–032, 1991.
25. Jaroslav Kačer. Discrete event simulations with j-sim. In *Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002*, pages 13–18. National University of Ireland, 2002.
26. Vasvi Kakkad, Andrew E Santosa, and Bernhard Scholz. Migrating operator placement for compositional stream graphs. In *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 125–134. ACM, 2012.
27. Seyed Jalal Kazemitabar, Ugur Demiryurek, Mohamed Ali, Afsin Akdogan, and Cyrus Shahabi. Geospatial stream query processing using microsoft sql server streaminsight. *Proc. VLDB Endow.*, 3(1-2):1537–1540, September 2010.
28. Boris Koldehofe, Beate Ottenwälder, Kurt Rothermel, and Umakishore Ramachandran. Moving range queries in distributed complex event processing. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, pages 201–212, New York, NY, USA, 2012. ACM.
29. Geetika T. Lakshmanan, Ying Li, and Rob Strom. Placement strategies for internet-scale data stream systems. *IEEE Internet Computing*, 12(6):50–60, November 2008.
30. Zongqing Lu and Yonggang Wen. Distributed and asynchronous solution to operator placement in large wireless sensor networks. In *Proceedings of the 2012 8th International Conference on Mobile Ad-hoc and Sensor Networks*, MSN '12, pages 100–107, Washington, DC, USA, 2012. IEEE Computer Society.
31. Zongqing Lu, Yonggang Wen, Rui Fan, Su-Lim Tan, and Jit Biswas. Toward efficient distributed algorithms for in-network binary operator tree placement in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 31(4):743–755, 2013.
32. David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

33. Mohamed F. Mokbel and Walid G. Aref. Sole: Scalable on-line execution of continuous queries on spatio-temporal data streams. *The VLDB Journal*, 17(5):971–995, August 2008.
34. Alberto Montresor and Márk Jelasity. Peersim: A scalable p2p simulator. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 99–100. IEEE, 2009.
35. Konstantinos Oikonomou, Ioannis Stavrakakis, and Alexios Xydias. Scalable service migration in general topologies. In *Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, WOWMOM '08, pages 1–6, Washington, DC, USA, 2008. IEEE Computer Society.
36. Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, Kirak Hong, David J. Lillethun, and Umakishore Ramachandran. MCEP: A mobility-aware complex event processing system. *ACM Trans. Internet Techn.*, 14(1):6:1–6:24, 2014.
37. Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, Kirak Hong, and Umakishore Ramachandran. RECEP: selection-based reuse for distributed complex event processing. In *The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014*, pages 59–70, 2014.
38. Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, and Umakishore Ramachandran. Migcep: operator migration for mobility driven distributed complex event processing. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 183–194. ACM, 2013.
39. Animesh Pathak and Viktor K. Prasanna. Energy-efficient task mapping for data-driven sensor network macroprogramming. *IEEE Trans. Comput.*, 59(7):955–968, July 2010.
40. Peter Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh, and Margo Seltzer. Network-aware operator placement for stream-processing systems. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 49–49. IEEE, 2006.
41. S. Rizou, F. Diirr, and K. Rothermel. Fulfilling end-to-end latency constraints in large-scale streaming environments. In *30th IEEE International Performance Computing and Communications Conference*, pages 1–8, Nov 2011.
42. S. Rizou, F. Durr, and K. Rothermel. Providing qos guarantees in large-scale operator networks. In *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, pages 337–345, Sept 2010.
43. Stamatia Rizou, Frank Durr, and Kurt Rothermel. Solving the multi-operator placement problem in large-scale operator networks. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1–6. IEEE, 2010.
44. Utkarsh Srivastava, Kamesh Munagala, and Jennifer Widom. Operator placement for in-network stream query processing. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 250–258, New York, NY, USA, 2005. ACM.
45. F. Starks and T. P. Plagemann. Operator placement for efficient distributed complex event processing in manets. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 83–90, Oct 2015.
46. Bernhard Stegmaier, Richard Kuntschke, and Alfons Kemper. Streamglobe: Adaptive query processing and optimization in streaming p2p environments. In *Proceeedings of the 1st International Workshop on Data Management for Sensor Networks: In Conjunction with VLDB 2004*, DMSN '04, pages 88–97, New York, NY, USA, 2004. ACM.
47. András Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM2001)*, volume 9, page 65. sn, 2001.
48. X. Xiong, H. G. Elmongui, X. Chai, and W. G. Aref. Place: A distributed spatio-temporal data stream management system for moving objects. In *2007 International Conference on Mobile Data Management*, pages 44–51, May 2007.
49. Lei Ying, Zhen Liu, Don Towsley, and Cathy H Xia. Distributed operator placement and data caching in large-scale sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.
50. Ben Zhang, Nitesh Mor, John Kolb, Douglas S. Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The cloud is not enough: Saving iot from

the cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, Santa Clara, CA, July 2015. USENIX Association.