Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 195 – 202

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

A Transaction Model for Executions of Compositions of Internet of Things Services

K. Vidyasankar

Department of Computer Science,Memorial University, St. John's, Newfoundland, Canada A1B 3X5

Abstract

Internet of Things (IoT) is about making "things" smart in some functionality, and connecting and enabling them to perform complex tasks by themselves. The functionality can be encapsulated as services and the task executed by composing the services. Two noteworthy functionalities of IoT services are monitoring and actuation. Monitoring implies continuous executions, and actuation is by triggering. Continuous executions typically involve stream processing. Stream input data are accumulated into batches and each batch is subjected to a sequence of computations, structured as a dataflow graph. The composition may be processing several batches simultaneously. Additionally, some non-stream OLTP transactions may also be executing concurrently. Thus, several composite transactions may be executing concurrently. This is in contrast to a typical Web services composition, where just one composite transaction is executed on each invocation. Therefore, defining transactional properties for executions of IoT service compositions is much more complex than for those of conventional Web service compositions. In this paper, we propose a transaction model and a correctness criterion for executions of IoT service compositions. Our proposal defines relaxed atomicity and isolation properties for transactions in a flexible manner and can be adapted for a variety of IoT applications.

Keywords: Internet of Things services; transactions; atomicity; isolation; stream data processing;

1. Introduction

The concept of transactions has been extremely helpful to regulate as well as ensure the correctness of concurrent executions of operations in various applications. The transaction concept was introduced first in the context of (centralized) database systems, and then adopted in various advanced database and other applications, more recently in Web services[1], electronic contracts[2], transactional memory[3] and stream processing[4]. Transactions are characterized by ACID properties: Atomicity, Consistency, Isolation and Durability. While these properties are considered very strictly for database operations and memory operations[3], they are relaxed in other applications, depending on the se-

[*] K. Vidyasankar. Tel.: +1-709-864-4369; fax: +1-709-864-2009.
E-mail address: vidya@mun.ca

doi:10.1016/j.procs.2016.04.116

mantics and constraints of the application environments. The earliest and most universally applied relaxation is with atomicity and isolation, in the definition of *saga*s[5]:

- A transaction is said to be correct and to preserve consistency, if executed completely or not at all;
- A higher level transaction can be split into, and executed by, several lower level transactions;
- Then, isolation is relaxed from the entire high level transaction to the individual lower level transactions;
- For atomicity, all the lower level transactions must be executed successfully, or none at all;
- If some of them are execuited successfully, but others cannot be executed successfully, then the earlier ones need to be compensated, to achieve overall null execution; and
- The compensation can only be logical and should take into account that other transactions might have observed and used the results of the successfully executed low level transactions.

Atomicity and isolation have been relaxed further in the various contexts mentioned above and in general nested transactions. In this paper, we study the relaxations that are applicable to compositions of Internet of Things Services.

Internet of Things (IoT) is about things connected through internet. A "thing" is any object of interest with some communication capability. Following the Service Oriented Architecture (SOA), "service" encapsulation is given to the functionality of things. These services can be employed as any other services. Basic services are combined to perform complex tasks or to build composite services. Monitoring and actuation are two essential functionalities of IoT services. Monitoring implies continuous executions, and actuation is by triggering. Continuous executions typically involve stream processing. Stream input data are subjected to a sequence of computations, structured as a dataflow graph. The computation is push-based: as soon as batches of input arrive, the computation is triggered. In addition, the stream input keeps arriving continuously and so the composition may be executing several batches concurrently. Additionally, some non-stream transactions (denoted OLTP transactions in this paper) may also be executing concurrently. In contrast, in a typical Web services composition, the execution is pull-based and we normally consider one-time execution (of an OLTP transaction) in each invocation of the composition.

In this paper, we propose a transaction model and a correctness criterion for executions of IoT services compositions. Our proposal defines relaxed atomicity and isolation properties in a flexible manner and can be adapted for a variety of IoT applications. We introduce executions of IoT services compositions in Section 2. We start with core definitions of compositions and transactions in this section. Then, we discuss the various relaxations in Section 3, and give our transaction model and a composition model in that section. We discuss related work in Section 4 and conclude in Section 5.

2. Executions

A *composition* C is $(\mathcal{P},\prec_p)$, where $\mathcal{P}$ is a set of *transaction programs* $\{P_1, P_2, \ldots, P_n\}$, simply called *programs*. and $\prec_p$ is a partial order among them. We call the (acyclic) graph representing the partial order the *composition graph* $\mathcal{GC}$. Each execution of a program yields a *transaction*. A transaction may have some stream and/or non-stream inputs, and may produce some stream and/or non-stream outputs. Stream data are sequences of tuples. They may come from outside the composition; these are from *base* streams. Others may be generated by programs in the composition; these are of *derived* streams. Transactions process stream inputs in *atomic batches* (also referred to, simply, as *batches*), the batch size being determined by applications. The stream outputs produced by transactions are also in batches, and they will constitute atomic batches for inputs to other transactions, if any.

In an execution of a composition, some of its programs will be executed, resulting in a set of transactions with a partial order $\prec_t$. We call this a *composite transaction*, denoted as $\mathcal{T} = (\{T_1, T_2, \ldots, T_m\},\prec_t)$. We denote $\{T_1, T_2, \ldots, T_m\}$ as *set*($\mathcal{T}$). The graph representing $\prec_t$ is called *transaction graph* $\mathcal{GT}$. The transaction graphs are acyclic. We note that each T_i is an execution of some program P_j. It is possible that $\mathcal{T}$ has more than one execution of some P_j (like in Meehan et al[6]). The partial order $\prec_t$ is compatible with $\prec_p$, that is, if T_i is an execution of P_j, T_k is an execution of P_l and $P_j \prec_p P_l$, then $T_i \prec_t T_k$.

Stream processing is usually push-based. As soon as the stream input tuples add up to a batch, the execution of the program for which they are input will start. OLTP transactions may be executed in pull-based fashion, like in Web services composition dealing with non-stream data. Both stream processing and OLTP transactions may trig-