Science of Computer Programming 105 (2015) 124–144

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Computing end-to-end delays in stream query processing

Vasvi Kakkad *, Andrew E. Santosa, Alan Fekete, Bernhard Scholz

School of Information Technologies, The University of Sydney, NSW 2006, Australia

ARTICLE INFO

Article history:
Received 17 December 2013
Received in revised form 5 March 2015
Accepted 7 April 2015
Available online 20 April 2015

Keywords:
Stream data processing
Distributed environment
End-to-end delay

ABSTRACT

Real-time data processing is essential in many stream-based applications including disaster area monitoring, health monitoring, and intrusion detection. In this work, we propose an approach that measures time delays in stream query processing. We represent a stream query as a graph consisting of operators that process data and channels that transport data tokens between operators. Our model establishes a causality relationship between consumed and produced data tokens at each operator and their corresponding occurrence times. The total time taken for the computation from the input to the output of a query, i.e., *end-to-end delay*, is computed by the causality relationships and periodic schedules for stream queries. Experiments are conducted to validate the proposed technique.

1. Introduction

The rapid advancement in wireless sensor networks (WSNs) enable the development of complex applications such as health monitoring [1], disaster area monitoring [2], and intrusion detection [3]. In such applications, concerned health and environmental data is captured using a multitude of sensors such as temperature, pressure, and motion readings. A wide range of sensors are deployed at distant locations to collect a stream of data like temperature rise near disaster-prone area and intrusion at highly secured locations. The data collected at sensors could be numerical, digital or discrete and to extract complex information, the sensed data need to be filtered, transformed, and merged. Some of the existing systems including Aurora [4], Medusa [5], Borealis [6], Mad-WiSe [7], and Curracurrong [8] express WSN queries with stream data processing. The aforementioned existing systems emphasise on providing energy-efficiency and flexibility in WSN applications. However, apart from those features, it is important that sensed data reaches the base station reliably and timely for time-critical applications such as health and disaster area monitoring. The ad-hoc infrastructure and resource constraints in WSN increase the uncertainty of successful and real-time data transmission. Approaches to overcome such challenges in WSN have been studied for a decade [9]. Recent works have analysed the end-to-end timeliness in terms of probability distribution [10,11], and first-order statistics [12]. For unreliable networks, work on real-time queueing theory provides stochastic models [13]. Our goal is to provide a comprehensive approach for determining delays in stream query processing using event causality concepts. In this article, based on event causality we introduce the notion of timeliness into the semantics for stream processing and an algorithm to measure end-to-end delay that in the processing.

Consider a disaster area monitoring application in the Curracurrong system [8], which deploys several proximate sensors and measures temperature change. The query collects the average temperature reading from the sensors placed at distant locations and checks whether the reading goes beyond a certain threshold. Fig. 1(a) shows a stream graph for Curracurrong query of the given application, where the result is recorded at the base station. The stream graph is composed of: two sense operations `Sense-1` and `Sense-2`; two filters `Average-1` and `Average-2` to compute average temperature

* Corresponding author.
E-mail address: vasvibhatt@gmail.com (V. Kakkad).

http://dx.doi.org/10.1016/j.scico.2015.04.003
0167-6423/

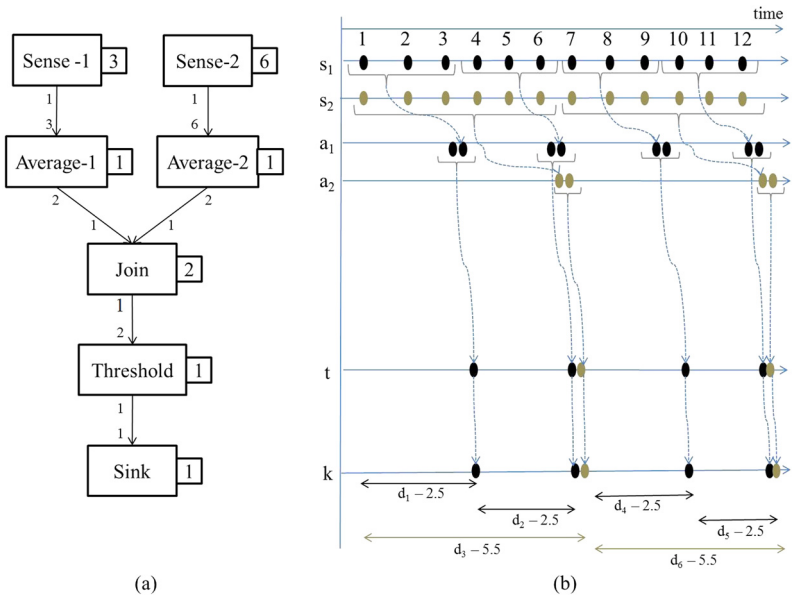

Fig. 1. Stream graph and event causality with synchronised sensors.

from `Sense-1` and `Sense-2`, respectively; a join operation `Join` to merge data from two filters; a filter `Threshold` that checks whether the average temperature readings are above threshold; and a `Sink` operator, where the final data is recorded. All operators are connected with uni-directional communication channels.

In the example, sensor nodes are deployed at distant locations and operators are placed in the intermediate sensor nodes before the data reach the sink (at the base station). It is important that temperature-related data collected at sensors reach the sink on time so that the user can take required action. Determining the freshness of the data relies on knowing how long it takes to propagate data from a sensor to the sink. Various operations on the sensed data insert certain delays in information generation at the sink (Fig. 1(b)). The figure shows that both sensors, represented as s_1 and s_2, generate data with uniform frequency and propagate the sensed data to consecutive operators until they reach the sink operator. During data propagation, the intermediate operators, such as average and threshold, consume more than one data token and take time for computation, both of which ultimately result in delay. Sense operators continuously generate data at every second and propagate them to averaging operators. Filter `Average-1`, represented as a_1, has window size 3, and therefore waits 3 s for data availability and computes the average, adding a few milliseconds delay. Likewise, filter `Average-2`, represented as a_2, waits for 6 s and inserts a few milliseconds delay during computation. The `Join` operator merges the data from two average filters and forwards them to `Threshold` (t) without any delay. The `Threshold` operator inserts a few milliseconds delay before the data token finally reaches the sink, k. The first data token generated at the sense reaches the sink with 2.5 s delay. The delays are not the same for each token reaching the sink, for two reasons: the different data rates of each operator; and sensor periodicity. The challenge is thus how to compute precise end-to-end delays in a stream graph, as shown in Fig. 1(b).

We used compositional stream graphs to establish a causality relationship for tokens and a notion of a time steady state to compute end-to-end delays of stream queries. Our approach was built on top of Curracurrong framework [8]. In summary, the paper makes three contributions:

1. a denotational semantics for stream data processing that explains time information propagation,
2. an algorithm to measure the end-to-end delays in a stream graph, and
3. an experimental evaluation to show the efficiency and effectiveness of our approach in determining end-to-end delays.

The remainder of the paper is organised as follows. Section 2 defines our model and formally describes problem definition statement. Section 3 defines compositional stream graphs and semantics. Section 4 shows an algorithm and defines abstraction of concrete semantics to determine delays. We present experimental evaluations that confirm the efficacy of our approach, together with periodicity scaling in Section 5. In Section 6 we survey the related work and conclude this article in Section 7.

2. System model and problem definition

As in the data flow model [14], we represent a WSN query as a *stream graph* $G = (V, E)$, whose vertices V are called *operators* and whose edges $E \subseteq V \times V$ are called *channels*. The *source* of an edge (u, v), denoted by $src(u, v)$, is u and the