USC ANRG Technical Report, ANRG 2017-04, 2017

The Publish-Process-Subscribe Paradigm for the Internet of Things

Bhaskar Krishnamachari, Kwame Wright

July 2017

Abstract

We advocate an extension of the traditional publish-subscribe middleware for the Internet of Things that incorporates computation that we refer to as *publish-process-subscribe* paradigm. This paradigm allows for greater privacy, aggregation and sensor fusion to improve resource utilization while delivering meaningful data to end points, as well as the necessary processing capability for controlling actuators. We give examples of such publish-process-subscribe systems suitable for IoT systems.

Introduction

The emerging Internet of Things (IoT) poses significant challenges including the need for greater computation for data processing, greater resource constraints, resource heterogeneity, and dynamics in resource availability, ensuring privacy for individual users sharing common resources, and need to allow rapid development of diverse sensing and control applications that incorporate data analytics. Correspondingly the challenges for application developers are also higher as they must deal with much greater complexity and heterogeneity of resources and uses than on the traditional Internet.

Besides work on REST implementations on constrained devices (such as CoAP [1,2]), an approach to enable IoT applications that has been growing in popularity is the traditional publish-subscribe paradigm [3]. This is evidenced for instance, by the increasing adoption of open protocols such as MQTT (mqtt.org), MQTT-S [4] and and proprietary platforms such as PubNub (pubnub.com). The benefit of the publish-subscribe approach is that it allows for fast and robust implementation of real-time many-to-many communications. It allows for asynchrony and is forgiving when faced with lossy and dynamic connectivity. The state of the art in these systems is the use of (typically centralized, though there are also some distributed approaches such as ZeroMQ, zeromq.org) brokers to which all published sensor data streams are sent and from which the data reaches subscribers.

To meet some of these major challenges associated with IoT systems, we advocate an extension of the traditional publish-subscribe approach that we refer to as **publish-process-subscribe**, which allows for the *en route* processing of data. In its simplest form, it provides for computation to be done at a more capable, broker so that a client can subscribe to a processed version of published raw data from one or more publishing devices or topics. In a more sophisticated version, the computation may be distributed and take place over a heterogeneous collection of compute points.

There are several advantages of the publish-process-subscribe paradigm:

Privacy: A raw data stream from one publisher could be processed through an anonymizing filter. Now, access controls could be set up via the broker so that certain authorized subscribers can have access to the raw data, while others are provided access only to the anonymized version.

Sensor Fusion and Aggregation: By allowing for en-route computation, data flowing from multiple subscribers can be combined and processed together to provide a more meaningful stream of refined data for a subscriber. This would also reduce bandwidth needs of the subscribing devices.

Computation for Automated Control**:** Computation specified over streaming published data could also be used to generate processed streams that intended to control particular actuators. This could be useful when there is limited computation at the client side.

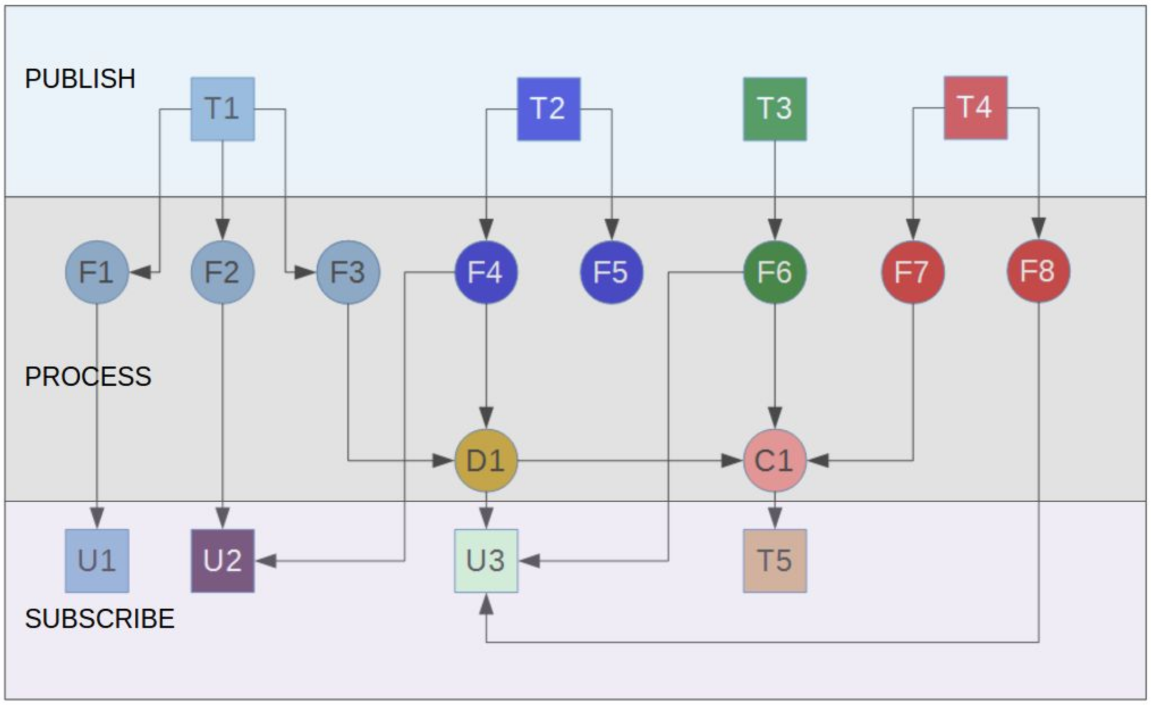

Figure 1: Illustration of Publish-Process-Subscribe: Things T1, T2, T3, and T4 are publishers of data streams. F1 through F8 represent privacy filters. User U1 is subscribed to the stream from T1 filtered through F1 (say representing a high quality, low-privacy stream). Another user U2 is subscribed to T1's stream through filter F2 (perhaps representing an intermediate level of privacy) as well as T2's stream through filter F4 (perhaps a high quality, low privacy level filter). D1 represents a data analysis or sensor fusion process node which combines the output of T1 filtered through F3, and the output of T2 filtered through F4. User U3 is subscribed to 3 streams: the output of D1, of T3 filtered through F6, and the output of T4 filtered through F8. Finally, C1 represents an automatic control algorithm that combines the output of D1 and the output of T3 filtered through F6 and the output of T4 filtered through F7 to send control signals to actuate another Thing T5. Here the filter blocks, data analysis/sensor fusion blocks and control

blocks represent processing, users are subscribers, and things are both publishers and subscribers depending on whether they are sensors (T1-T4) or actuators (T5).

Case Study 1: Noctua

Noctua is a macroprogramming framework that we are currently developing for the Internet of Things (IoT). It is designed to address three fundamental challenges in IoT systems: optimizing communication, distributing computation, and protecting privacy. Noctua is based on MQTT [5], a lightweight publish-subscribe protocol, an industry standard for IoT systems. Noctua allows edge or cloud devices to perform computation, allowing for aggregation to occur at arbitrary points in a network, which reduces congestion downstream. Also, by offloading computation, Noctua has the added benefit of improving the lifetime of battery-powered devices. Noctua is capable of enforcing access constraints on data streams, a standard way of protecting information. However, when combined with it's offloading capabilities, Noctua can provided access to aggregated forms of data, such that the privacy of individual values are maintained but useful observations can still be made. The key idea behind Noctua is that it provides subscribers a way to specify computations over published streams and receive the processed streams. More details on Noctua will be available in a future publication from ANRG.

Case Study 2: PubNub Blocks

There is also a state of the art commercial system that has implemented a version of publish-process-subscribe functionality. PubNub has recently introducedFunctions, a function as a service platform. PubNub Blocks offer the ability to modify and apply computations on data in flight with user-specified functions called Blocks. For more details see https://www.pubnub.com/docs/blocks/introduction

References

[1] Shelby, Z., *et al.* "Constrained Application Protocol (CoAP), draft-ietf-core-coap-13." *Orlando: The Internet Engineering Task Force–IETF, Dec* (2012).

[2] Colitti, W., K. Steenhaut, and N. De Caro. "Integrating wireless sensor networks with the web." *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)* (2011).

[3] Eugster, P. T., *et al.* "The many faces of publish/subscribe." *ACM Computing Surveys (CSUR)* 35.2 (2003): 114-131.

[4] Hunkeler, U., H. L. Truong, and A. Stanford-Clark. "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks." *Communication Systems Software and Middleware and Workshops, 3rd International Conference on*. IEEE, 2008.