XTream: Personal Data Streams*

M. Duller, R. Tamosevicius, G. Alonso, D. Kossmann
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
{michael.duller, rokas, alonso, kossmann}@inf.ethz.ch

ABSTRACT

The real usability of data stream systems depends on the practical aspect of building applications on data streams. In this demo we show two possible applications on data streams implemented on our prototype platform *XTream*. One application integrates VoIP and E-Mail, the other one incorporates streams in a Smart Home setting. Using these applications we try to identify and discuss the functionality that data stream management systems should provide. Those attending the demo will be able to compose their own applications.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: User/Machine Systems—*Human information processing*; H.2 [**Database Management**]: Systems—*Distributed Databases*; H.5 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Design, Management

Keywords

XTream, Data Streams, Distribution, Information System, Personalization, OSGi, R-OSGi

1. INTRODUCTION

There is an important and rapidly growing body of work on processing data streams using queries. However, comparatively little work has been done on the more practical, but not less relevant, aspect of how to build and deploy complex and distributed applications over data streams. In this demo we make the point that this aspect is the one that determines the real usability of a data stream system. The demo inverts the terms in which most demos are performed. Rather than showing what our system can do, it intends to identify what functionality any data stream management system should provide. We then demonstrate how the prototype platform *XTream* implements such functionality. For this purpose we focus on personal data streams, i.e., quasi-real-time data streams relevant to people, and two applications built on those streams. The data streams we consider include, among others, E-Mails, information on phone calls, SMS, information from the digital home (e.g., status of home appliances, events in the home or office), and data from sensor networks. We use these streams and the experience gathered from building the applications to show that although one can cast the problem in terms of continuous queries and triggers, in practice querying ends up not being the biggest problem. Instead, we show that acquisition, distributed processing, personalization, and the integration with a wide variety of devices are the real challenges. Furthermore, flexible deployment and automatic software configuration, typically outside the realm of database centric data streaming systems, are mandatory properties. The purpose of this demo is to explore these issues, discuss some of the tools needed to solve them, and demonstrate a prototype of a system that implements some of these tools.

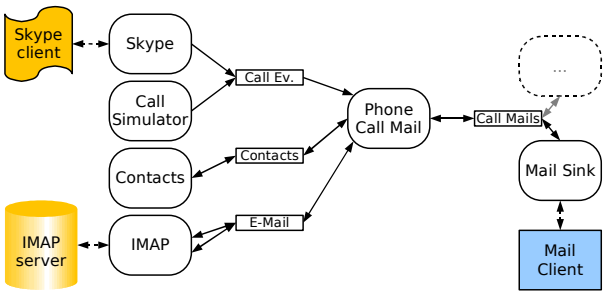

Figure 1: SkypeMail application setup.

2. APPLICATIONS

2.1 Integration of VoIP and E-Mail

Personal data streams must be user-centric. The user knows best what information to integrate and what is relevant. Accordingly, our goal is to provide users with the tools to do so by approaching the problem as a user-driven, dynamic data stream integration problem. To illustrate the types of interaction that such an integration implies, we have built a data stream application that combines Skype with IMAP to provide context to a phone call (Figure 1). Upon receiving a Skype call, the name of the caller is extracted

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number 5005-67322.

SIGMOD'07, June 12–14, 2007, Beijing, China.
ACM 978-1-59593-686-8/07/0006.

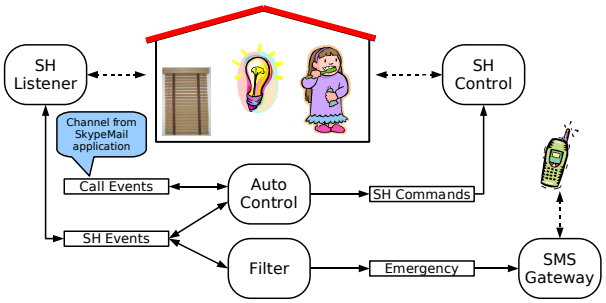

Figure 2: Smart Home setup.

from the meta-data associated to the call. Then the name is looked up in a contact database and translated into an E-Mail address. This E-Mail address is used to query an IMAP server to retrieve the last ten E-Mails from and to that caller. The E-Mails are displayed and directly available during the call. Furthermore, new E-Mail from and to the caller will be displayed immediately. The application is easily extensible and its behavior can be dynamically changed (e.g., display E-Mail from other sources, different criteria can be used to select the E-Mails, list past Skype calls with all the persons addressed in the E-Mails displayed, etc.).

This apparently simple application involves a rather complex set of functionalities: event processing triggered from a push stream (Skype producing an asynchronous data stream announcing calls), data filtering to extract the user name, ability to run a query on a database to retrieve the E-Mail address, dynamic generation of a pull stream from the IMAP server, and the conversion of this stream into a push stream to keep the E-Mail list up-to-date. In addition, the ability to define workflows with all these operations (similar to a business process) and provide the right abstractions for the interactions is crucial to implement such an application. Many of these operations are not supported by existing systems without ad-hoc programming. Nonetheless, this functionality is very common in personal data stream applications and should be supported from the start by the underlying data stream management system.

2.2 Streams in the Smart Home

The second scenario involves a digital home setting. The scenario is part of the *Smart Home* project, a cooperation between ETH Zurich and Siemens AG, and part of an initiative led by Siemens and German Telekom with a first house in Berlin (www.t-com-haus.de) and several others currently under construction in Dubai. The goal of the Smart Home project is to integrate technologies for security, lifestyle, and communication and make them available in the home. The demonstration will illustrate the importance of rapid development, integration, and deployment of personalized data stream services across a wide variety of platforms and devices. Figure 2 illustrates this application. Both the events generated by the Smart Home and the interactions with the appliances are realized with data streams. For example, users can subscribe to certain events and receive a notification on their mobile phone. Furthermore, control of the smart home can be automated depending on the users' preferences. We will demonstrate this automation by integrating the call information as it is used in the Skype/E-Mail application described above. The goal is for the user to set up different controls in the house that will react differently depending on who is calling (e.g., the so called *mood* controls determining light levels, opening or closing of blinds, etc.). The Smart Home scenarios explore in more detail the problem of event management and the programming and deployment flexibility needed to interact with hardware appliances. For instance, it encompasses user interfaces on different devices, implying the need for support for distribution from the start. In addition, and because there are several ways to interact with the system, these interfaces must be synchronized in order to avoid the user carrying out the same action more than once. The application must be able to not only consume streams, but also to react to incoming data streams by generating additional streams, some of them with commands for devices. Finally, the proliferation of devices to control and be controlled opens up interesting opportunities for optimizing the deployment of the different parts of the processing chain. All these aspects are considered to be orthogonal or even irrelevant to data stream processing but, the demonstration will show, are crucial components of the underlying streaming platform.

3. DATA STREAM PLATFORM

These applications raise a number of non-trivial issues. One of the main challenges is the lack of abstractions for the operations and interactions that take place. In addition to handling data streams (e.g., filtering), these applications must, e.g., communicate with external data sources, manage both push and pull interactions, integrate a number of different technologies, be able to evolve in unpredictable ways, run seamlessly on different mobile and unreliable devices as well as on servers, etc. The applications that will be demonstrated show that, far from being orthogonal, these issues are the ones defining the capabilities of a stream processing system. In what follows we briefly present the XTream platform used to build the applications described above. XTream is an early prototype intended to test and illustrate different technologies.

3.1 Processing Model

We use a data flow model similar to that of Ptolemy [2] and also used by Aurora [1] or Telegraph [3]. The architecture is based on *SLETs* and *channels*. An SLET (short for *streamlet*) is a generic program with well defined interfaces (including Web service-based ones) that consumes and produces data streams. This allows users not only to implement their own operators in the shape of SLETs but to implement SLETs with arbitrary processing logic and use them seamlessly in the system — one of the key differences between XTream and conventional stream processing systems with a predefined set of data manipulation operators.

SLETs are oblivious of each other. They communicate through channels. Channels are abstract constructs mapped to different concrete implementations (views over databases, publish/subscribe mechanisms, gateways across nodes in different locations, repositories if the channel is persistent, etc.). Developers and users only need to define the desired properties and the underlying platform will choose the best implementation. Channels can receive data from several SLETs and several SLETs can read data from the same channel. Channels can also be push or pull and support callback operations that are forwarded back to the SLETs

Figure 3: Smart Home simulator and PDA running home control.

depositing data on the channel. Since SLETs and channels are loosely coupled they can be dynamically connected and disconnected. Furthermore, different properties can be assigned to SLETs and channels. For instance, two SLETs can be synchronized so that if one SLET consumes a data item from a channel, its companion SLET will not consume this data item again. Providing these properties and having the possibility of push, pull, and callback operations back to the SLETs where data originated from are further differences between XTream and conventional data stream processing systems.

While SLETs have an input part, an output part, and a processing core interacting with both parts, data sources and sinks are wrapped with "half SLETs". Their core interacts with the external source or sink and the output or input part interacts with the channels in XTream. Processing of streams is done through data pipelines that alternate SLETs and channels. Users build such workflows or pipelines by connecting the inputs and outputs of channels and SLETs.

3.2 Implementation

The current prototype implementation of XTream covers the main tools to address the requirements posed by the applications described. Rather than extending a database engine, we have opted for using an efficient processing platform that also supports distribution. We find it easier to add querying capabilities to that platform than to add flexibility and software management functionality to a querying engine. The implementation of the applications using our prototype prove this to be the case but we expect this to be one of the controversial points of the demonstration.

The core of XTream is implemented as OSGi [4] *bundles*. An OSGi bundle contains classes and other resources as well as a list of properties. It can be installed and removed at runtime, thus providing support for dynamically adding and removing components. A bundle can register services that, in turn, can be retrieved and used by other bundles. A service is an instance of a class registered with the framework's service registry under one or more interfaces and with a set of properties which can be used to locate a specific service.

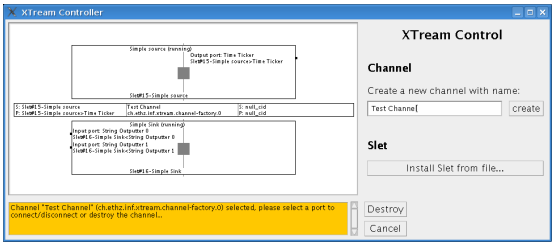

Figure 4: GUI for managing an XTream system.

SLETs and channels are both implemented as bundles and services, especially facilitating their lifecycle management as well as their interaction with each other and with the XTream platform. By implementing the XTream platform itself as OSGi bundles as well, the platform is modular, and features (e.g., a monitoring module) can be added and removed at runtime. Furthermore, we use some of the services specified by OSGi. The current version of XTream uses Concierge [5], our own OSGi implementation, and was tested with Knopflerfish OSGi as well. Due to the nature of OSGi, the current prototype is limited to interacting with channels and SLETs that exist in the same virtual machine. In order to implement distribution over multiple virtual—and thus also physical—machines, we will use R-OSGi [6], a system for transparent access to remote OSGi services, which allows us to distribute the SLETs and channels among multiple instances of an OSGi framework.

4. THE DEMONSTRATION

The demonstration will involve the applications Skype/E-Mail integration and Smart Home events, control, and integration. The applications will be implemented as a workflow of SLETs and channels on the XTream prototype platform. They will be deployed in several laptops and mobile devices (PDA, mobile phone). We will show how to quickly develop such an application, how to evolve it, and how to deploy it in a distributed setting. We invite all those attending the demo to interact with the applications, customize them, and build their own applications.

The Smart Home application runs on top of a simulator (Figure 3). It will be used to gather and distribute events related to changes in the home and to return commands to the home either at the user's request or as a result of event processing implemented within the platform. We will also show how to implement simple workflows as a way to illustrate how users may eventually interact with such a system. During the demo we will use a simple GUI to generate channels, deploy SLETs, connect channels to SLETs, and move SLETs to different devices (Figure 4).

5. REFERENCES

[1] D. J. Abadi et al. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.
[2] J. Buck et al. Ptolemy: A Framework for Simulating and Prototyping Heterogenous Systems. *Int. Journal in Computer Simulation*, 4(2), 1994.
[3] S. Chandrasekaran et al. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *CIDR*, 2003.
[4] OSGi service platform. `http://www.osgi.org/`.
[5] J. S. Rellermeyer and G. Alonso. Concierge: A Service Platform for Resource-Constrained Devices. In *EuroSys*, 2007.
[6] J. S. Rellermeyer and G. Alonso. Services everywhere: OSGi in Distributed Environments. In *EclipseCon*, 2007.