Towards Pervasive Web API-based Systems

Felix Leif Keppmann, felix.leif.keppmann@kit.edu, Karlsruhe Institute of Technology
Maria Maleshkova, maria.maleshkova@kit.edu, Karlsruhe Institute of Technology

Recent technology developments in the area of services on the Web are marked by the proliferation of Web applications and APIs. This development, accompanied by the growing use of sensors and mobile devices raises the issue of having to consider not only static but also dynamic data accessible via Web APIs. New implications on the communication in such networks can be derived, i.e., the active role of pulling or pushing data is no longer exclusively assigned to specific roles, e.g. services, or clients. Furthermore, the establishing of data flow between devices in such a network may be initiated by any of the participating devices. In this paper we present a general approach for the modelling of a service-based systems, which takes into consideration the providing/consuming of dynamic data sources. In particular, we develop a formal model for capturing the communication between Web APIs and client application, by including dynamic data producing services and data consuming client applications. We complement our model with a decision function for optimising the pull/push communication direction and optimise the amount of redundant transferred data (i.e. data that is pushed but cannot be processed or data that is pulled but is not yet updated). The presented work lays the foundation for creating intelligent Web API-based systems and client application, which can automatically optimise the data exchange.

1 Introduction

Web services provide means for the development of open distributed systems, based on decoupled components, by overcoming heterogeneity and enabling the publishing and consuming of data and functionalities between applications. Recently the world around services on the Web, thus far limited to “classical” Web services based on Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL), has been enriched by the proliferation of REST services, when their Web Application Programming Interfaces (APIs) conform to the Representational State Transfer (REST) architectural principles (Richardson & Ruby, 2007). These Web APIs are characterised by their relative simplicity and their natural suitability for the Web, relying directly on the use of Uniform Resource Identifier (URI), for resource identification and Hypertext Transfer Protocol (HTTP) for service interaction and message transmission.

At the same time, emerging complex and distributed systems composed of nodes with heterogeneous hard- and software characteristics face new challenges. For example, the wider use of sensors and mobile devices raises the issue of having to consider not only static but also dynamic data. However, traditional client-server based architectures are not optimal for supporting the consumption and provision of static and dynamic data alike. Furthermore, devices seen as nodes in a network, cannot be directly identified as data consumers or provides based solely on their role as server or client application. This has implications on the overall communication between the nodes. As a result, mapping the data flow in a distributed complex scenario to the Web pull-push-based communication becomes a challenge. While both, push and pull, enable the data flow between nodes, one may be less efficient in terms of transmitting redundant data, with impact on the overall system.

In this work we address some of the issues appearing in these scenarios. In particular, we provide: 1) a model to capture a network of data producing and consuming nodes with their relevant properties, 2) a decision function to optimise the communication between services and clients and thus determine, which nodes in the network should be actively initiating the communication.
2 Motivation

In order to establish a more specific notion of the problem we use the scenario of a house monitoring system. In this example different sensors serve as services, e.g. as Web API, and provide their sensor data on the network. Client applications monitor the sensors and allow end users to access and visualise the sensor data. Due to the nature of the sensors, the rate, with which their services register and provide new data, i.e. the sensor’s update frequencies, differs. Similarly, the characteristics of the monitoring clients differ in their update rates, for instance in the type of visualisation (e.g., highly frequent display or rarely updated web-based map). Some of the services and clients are connected by a data flow, thus the client requires data from the service to provide a certain functionality.

We introduce our monitoring system scenario by looking at the communication and data flow in terms of a network. This network incorporates services, which can produce or simply provide access to data, and clients, which consume data. A service may be perceived as static (e.g., a blueprint of a house) or as dynamic (e.g., a video or a temperature sensor). In addition, a service has an update frequency (over a certain time span), which may be zero in the case of static data, or several times per seconds, in case of dynamic data. Clients can consume data from a data source very rarely (e.g., only once to create a map) or up to several times in short intervals (e.g., multiple times per second to update a visualisation). Thus clients are also characterised by a specific update frequency, depending on the functionality that they provide.

The result is a network of communicating services and clients as nodes, characterised by their update frequencies, and the connections between the nodes, based on data production and consumption, visualised in Figure 1. In case of a data flow a connection between a service and a client is placed. Furthermore, some nodes expose data at a constant update frequency, while others, such as the temperature sensor, expose a minimum, maximum and average update frequency.

We use the scenario of a house monitoring system as the basis for deriving requirements for a formal model that is capable of capturing services and client applications in terms of a data-driven communication network. In a scenario, as the one described above, services and clients communicate to transfer data from a service to a client. Thereby, two basic communication directions can appear – the client requests data from the service, i.e. pull, or the service actively sends data to the client, i.e. push. In both cases the data, which is transferred in the messages, may be redundant. On the one hand, a message contains the same data if a client requests data from a service, which has not been updated since the last request. On the other hand, data is discarded if a service sends data to a client, which at that time cannot be processed by the client. How often the data of services is updated and how often clients are able to process data are represented by their update frequencies.

Based on the scenario and the described characteristics of the network nodes, we can derive requirements for defining a model that supports the optimisation of the data-driven communication between data services and clients. In particular, the model should enable: 1) the minimisation of redundant transferred data contained in messages. Moreover, it should 2) respect the flow of data in the network of services and clients, and 3) optimise the direction of active communication between nodes in the network (services that actively provide data, i.e. push, vs. clients that actively request data, i.e. pull). Since they express the capabilities to provide and request data, 4) the frequencies of services and clients must be taken into account.
3 Communication Model

In this section we introduce our communication model, which captures all nodes, the data flow in the network, as well as the different update frequencies of services and clients. Based on both, the data flow and the frequencies, a decision function is introduced, which support the optimisation of the pull-push directions in the network. The optimised pull-push directions result in minimising the volume of redundant data that is being transferred, therefore, reducing the overall data exchange volumes. In the following we first elaborate on important preliminaries and definitions, second we introduce the communication model and finally, we describe the decision function.

The following preliminaries apply, in order to keep the function and the model of the basic approach simple. Nodes in the network may play the role of a service, client, or both service and client and expose one frequency, i.e. data consumption and/or data provisioning update frequency. This frequency is, in case of a service, the update frequency of new data provided on the network. In case of a client, it is the frequency, with which the client requires data coming from a service. Nodes acting as a service and a client at the same time have only one frequency for both.

The following definitions apply. The number of events per unit time is called frequency. In this context we use the unit Hertz (Hz) for frequency, defined as number of events (e.g. new data provided by a service) per second \( f = \frac{k}{t} \). While \( k \) is the number of events and \( t \) is the time in seconds. It can be measured by counting the number of occurrences of an event for a given period of time. All nodes in a network are numbered by \( 1, \ldots, n \) with \( n \in \mathbb{N} \). Each node in the network may play the role of a service, denoted by \( S \), or client, denoted by \( C \). A node in a specific role is denoted by \( S_i \) or \( C_j \) with \( 1 \leq i, j \leq n, n \in \mathbb{N} \). Each node in the network exposes its frequency with a minimum, average and maximum. A constant frequency is denoted as \( f_i = f_{i}^{\text{min}} = f_{i}^{\text{avg}} = f_{i}^{\text{max}} \) with \( 1 \leq i \leq n, n \in \mathbb{N} \). A variable frequency it is denoted as \( f_{i}^{\text{min}} < f_{i}^{\text{avg}} < f_{i}^{\text{max}}; 1 \leq i \leq n, n \in \mathbb{N} \). For convenience and readability, the frequencies of nodes with a particular role are denoted as \( f_{S_i} \) and \( f_{S_i}^{\text{min,avg,max}} \) or \( f_{C_j} \) and \( f_{C_j}^{\text{min,avg,max}} \) with \( 1 \leq i, j \leq n, n \in \mathbb{N} \).

### 3.1 Model

The model consist of a data flow graph \( D \) and the minimal \( N \), average \( G \) and maximal \( X \) frequencies of all involved network nodes. It allows to determine a communication graph \( C \), representing the nodes in the network which actively communicate. Data flow and communication graph combined make a point, on which nodes pull or push data in the network to establish the data flow. A decision function, described in Section 3.2 determines the communication graph \( C \) based on \( D, N, G \) and \( X \).

The data flow is represented by a directed unweighted graph and is encoded in the adjacency matrix \( D \). Nodes playing the role of a service are indexed by \( m \), of a client by \( n \) and the size \( m \times n, m = n \) of the adjacency matrix is determined by the number of nodes in the network. Each connection \( d_{m,n} \) between a service and a client is encoded by \( 1 \) in the data flow adjacency matrix, in direction from service to client. The diagonal entries are set to \( 0 \) to avoid loops from a node to itself in the graph. Each entry in the matrix specifies the direction, in which data is transferred.

\[
D_{m,n} = \begin{pmatrix}
0 & d_{1,2} & \cdots & d_{1,n} \\
d_{2,1} & 0 & \cdots & d_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
d_{m,1} & d_{m,2} & \cdots & 0
\end{pmatrix}
\]

\[d_{m,n} \in \{0,1\}, \quad m = n, \quad m, n \in \mathbb{N} \tag{1}\]

The communication graph \( C \) is a graph with the nodes \( N \), and the edges \( X \). The data flow is represented by a directed unweighted graph and is encoded in the adjacency matrix \( D \). The communication graph combined make a point, on which nodes pull or push data in the network to establish the data flow. A decision function, described in Section 3.2 determines the communication graph \( C \) based on \( D, N, G \) and \( X \).

\[
N_n = \begin{pmatrix}
f_{1}^{\text{min}} \\
f_{2}^{\text{min}} \\
\vdots \\
f_{n}^{\text{min}}
\end{pmatrix}
\quad G_n = \begin{pmatrix}
f_{1}^{\text{avg}} \\
f_{2}^{\text{avg}} \\
\vdots \\
f_{n}^{\text{avg}}
\end{pmatrix}
\quad X_n = \begin{pmatrix}
f_{1}^{\text{max}} \\
f_{2}^{\text{max}} \\
\vdots \\
f_{n}^{\text{max}}
\end{pmatrix}
\]

\[
f_{n}^{\text{min,avg,max}} \in \mathbb{R}, n \in \mathbb{N} \tag{2}\]

The data flow is represented by a directed unweighted graph and is encoded in the adjacency matrix \( D \). Nodes playing the role of a service are indexed by \( m \), of a client by \( n \) and the size \( m \times n, m = n \) of the adjacency matrix is determined by the number of nodes in the network. Each connection \( d_{m,n} \) between a service and a client is encoded by \( 1 \) in the data flow adjacency matrix, in direction from service to client. The diagonal entries are set to \( 0 \) to avoid loops from a node to itself in the graph. Each entry in the matrix specifies the direction, in which data is transferred.

\[
C_{m,n} = \begin{pmatrix}
0 & c_{1,2} & \cdots & c_{1,n} \\
c_{2,1} & 0 & \cdots & c_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
c_{m,1} & c_{m,2} & \cdots & 0
\end{pmatrix}
\]

\[c_{m,n} \in \mathbb{R}, \quad m = n, \quad m, n \in \mathbb{N} \tag{3}\]
between a particular service and client.

The minimum, average and maximum of the frequencies of all nodes in the network are encoded as vectors. They are indexed by the index of the respective node and the size of the vector is determined by the number of nodes.

The determined communication graph is encoded in a similar way as the data flow graph by a directed finite simple graph with weights, encoded in the adjacency matrix $C$. In contrast to the data flow adjacency matrix, the range of the communication adjacency matrix is real numbers. Each entry in the matrix specifies the frequency and direction, in which a particular service has to push, or a particular client has to pull data in order to establish the data flow encoded in the data flow adjacency matrix.

### 3.2 Decision Function

The entries of the communication adjacency matrix $C$ are derived from $D$, $N$, $G$ and $X$. Each entry $c_{i,j}$ of the matrix is determined by the decision function.

We distinguish four cases of the function, which differ in the constant or inconstant frequencies of services and clients. By default $c_{i,j} = 0$ is set. In the first case, both the service and the client have constant frequencies. In the second case, the service has an inconstant frequency and the client has a constant frequency. The border cases $f_{S_i}^\text{max} = f_{C_j}$ and $f_{C_j}^\text{min} = f_{C_j}$ are included in the cases $f_{S_i}^\text{max} \leq f_{C_j}$ and $f_{S_i}^\text{min} \geq f_{C_j}$ correspondingly. A minimal frequency $f_{S_i}^\text{min}$ of the service $S_i$ that equals the constant frequency $f_{C_j}$ of the client $C_j$ means in average a higher frequency of the service compared to the client, which has at least the frequency of the client. The second border case is handled analogically. In the third case, the service has a constant frequency and the client has an inconstant frequency. Finally, in the fourth case both service and client have inconstant frequencies.

In summary, the decision function determines, which node, given two nodes that are exchanging data (i.e. a services and a client), has to be the active one. Combined with the communication model, the decision function supports the formal capturing of a network of nodes, based on data-driven communication, and prescribes the optimal set of pushing and pulling nodes, in order to minimise the transfer of redundant data.
4 Evaluation

In this section we provide two sets of preliminary evaluation results. First, we apply the communication model and the decision function on actual nodes, with specific frequencies, and taking into consideration available connections. Second, we check the conformity of the model to the derived requirements. An in-depth experimental evaluation is part of future work.

We apply the model on the motivation scenario s (see Section 2) and, subsequently, calculate the communication matrix. We construct the data flow matrix $D_s$ of scenario $s$ based on the description of the scenario and as shown by the arrows for data flow in Figure 2 (solid arrows). Each non-zero entry in the matrix represents one data flow between nodes in the scenario (e.g., from node 1 as service ($S_1$) to node 3 as client ($C_3$)).

The frequencies of participating nodes in the network are given by the specification of the devices and applications, also shown in Figure 2. We construct all three vectors for minimal $N_s$, average $G_s$ and maximal $X_s$ frequencies, apply the decision function on the given arguments and derive the communication matrix $C_s$.

The derived set of active nodes within the network are visualised in Figure 2 as dashed arrows. For example, for nodes 1 and 4, 1 is the service and 4 the client (solid arrow), however, based on the update frequencies, 4 should be the active node and request or pull data from 1.

We also evaluate the communication model and the decision function based on its conformity to the requirements, which were derived in Section 2. The decision function is based on optimising (minimising) the redundancy in terms of transferred data, thus minimising the volume of exchanged data. It uses the update frequencies in order to determine, which node should be the active in terms of producing or consuming the data, thus reducing the data, which was not updated for pull or cannot be processed for push (Requirement 1). Furthermore, the model and the decision function do not require that the function of a node is reassigned (services remain services and clients remain clients) or that the data flow direction is changed (Requirement 2), they simply determine, which nodes should be active (Requirement 3), based on taking the update frequencies into consideration (Requirement 4).
5 Related Work

Related work has already been conducted in different domains, including economics and business administration, or, taken as excerpt, the field of sensors and sensor networks. The authors in (Cheng, Perillo, & Heinzelman, 2008) discuss different deployment strategies to avoid the decrease of sensor network lifetimes caused by high energy consumption of specific important network nodes. They propose a general model to compare these strategies under certain restrictions as well as calculate the costs caused by the deployment. For the scenario of a sensor network deployed over an area for surveillance, (Mhatre, Rosenberg, Kofman, Mazumdar, & Shroff, 2005) propose a cost model approach to determine an optimal distribution of sensing nodes and cluster head nodes. For large wireless ad-hoc networks (Liu, Huang, & Zhang, 2004) propose a combination of pull- and pushed-based strategies to optimise the routing for specific information needs. Thereby, the query frequencies are taken into account. Compared to our work these approaches focus more on including factors specific to the deployment of sensors, e.g. power consumption, wireless strength or equipment costs, while we focus more on the optimisation of general issues in a network of data producing and consuming nodes.

6 Conclusion

Current developments of the Web are marked by the growing adoption and use of Web APIs. This trend, in combination with the wider use of sensors and mobile devices, raises new unaddressed challenges. In particular, there is the need for a general solution that enables the modelling of service-based systems, which is also capable of handling static as well as dynamic data exposed via Web APIs. In this paper we have presented a modelling approach that captures Web APIs and client applications, as data providers and consumers, which are characterised by their update frequencies. We use the formal model as the basis for applying a decision function for automatically determining, which communicating party needs to be active, by sending or requesting the data, in order to optimise the communication in terms of minimal data redundancy. The presented work lays the foundation for creating intelligent Web API-based systems and client application, which can automatically optimise the data exchange. As part of future work, we aim to integrate further influential factors. Currently, the proposed model allows only one type of frequency per network node. However, a node can actually serve as a service and client at the same time and would have an input and an output frequency, which may be equal or, e.g. in an aggregating node, differ. Similarly, latency and bandwidth are not considered in the current model. Latency would influence both the communication model and, especially, the decision function, which would have to take into account not only frequencies but also the possible latencies. The addition of bandwidth limitations would help to optimise data-driven communication. On the one hand, some factors could exclude others (i.e. bandwidth limitations overrule update frequencies). On the other hand, in combination with latency, the model could show the trade-off between lower latency and higher bandwidth.

References