2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop

Data Prefetching in Smart Systems

Sabine Weninger and Michael Borkowski
Distributed Systems Group
TU Wien
Vienna, Austria
{s.weninger, m.borkowski}@infosys.tuwien.ac.at

***Abstract*—In smart system scenarios, such as the Internet of Things (IoT), managing data traffic remains a crucial challenge, with high data volumes and low data transmission rates significantly hindering user experience. In this light, a promising technique is data prefetching, which involves the fetching of data on specific devices before it is required by the user, in order to reduce the response time and improve the application's user-perceived Quality of Experience (QoE).**

In this paper, we present an implementation of such data prefetching in an IoT scenario. We evaluate our solution using a real-world testbed consisting of Raspberry Pi computers and an Android smartphone. Our results show that data prefetching is a viable method for reducing response time in IoT scenarios, and subsequently, improving the user-perceived QoE.

***Index Terms*—prefetching, smart systems, IoT**

I. Introduction and Motivation

Smart systems, i.e., systems displaying "smart behavior", consist of sensors, which detect changes in the environment, and actuators, which adapt the system to these environmental changes [1, 7]. Examples for such smart systems include smart cities and smart factories [16]. For instance, the public transport infrastructure in a city can be equipped with sensors, collecting data such as the current position of trams, buses, and subway trains. A user traveling through the city is then able to gather this live information on a mobile phone, take into account possible delays, and select the ideal route. Other scenarios include parking sensors installed in a city, where parking spots are made visible for the driver of a car, reducing the time required for finding a parking spot, which in turn leads to reduced CO_2 emissions and less traffic congestion [16].

Internet of Things (IoT) devices can act as sensors and actuators, and can be used to implement such a smart environment [2, 7]. Technologies used for IoT communication include Radio Frequency Identification (RFID) [6, 7] and Wireless Sensor Networks (WSN) [15].

An important factor in the advancement of smart systems is mobile computing. Smartphones are now omnipresent and mobile traffic has grown 18-fold over the past 5 years [5]. This trend will continue, with estimations predicting a 7-fold increase of mobile traffic and a 3-fold increase of connection speeds between 2016 and 2021 [5]. Mobile computing provides the technology necessary for connecting devices over large distances, creating global communication networks [7].

Having smart technology incorporated into everyday objects and connecting them through the Internet poses several challenges, a key one being the location of storing and processing data. The current paradigm shift towards cloud computing [10] provides a possible solution to this challenge. For instance, the aforementioned IoT sensors can send their data to a cloud-based system, to which users connect to query the data. However, this poses the disadvantage of large amounts of data traffic, including both upstream communication from the sensors to the cloud, and downstream communication back to the mobile device, for instance, the user's car navigation system.

We investigate a solution to this problem, which works by prefetching data that the user will need, to a location which the user is going to pass in the foreseeable future. Prefetching is the process of transferring data ahead of the time of requirement, in order to have the data available and ready to send upon the user's request [4, 8]. In the presented context of smart cities, live public transport data is sent to the IoT devices along the user's path of navigation. Once the user is within range of these IoT devices, the data can be transferred directly, avoiding time-consuming request-response communication between the user's device and a central server. This helps to improve the Quality of Experience (QoE) [4, 9].

This paper presents a demonstration of a data prefetching solution in an IoT environment. Existing literature already includes various concrete algorithms for finding decisions on when and where to prefetch which data items, and these algorithms cover many different use cases [4, 8, 12]. We therefore do not discuss another such algorithm. Instead, we provide a reference solution for data prefetching on IoT and address possible issues that can arise. The solution is designed in a way that enables its application to various use cases, including smart cities, smart factories, and mobility scenarios in general.

The remainder of this paper is structured as follows: In Section II, we provide an overview of the system architecture of our approach. In Section III, we discuss our implementation, and evaluate its functionality. Finally, in Section IV, we conclude our work, discuss limitations, and provide additional outlook and discussion.

II. Overview

In our scenario, a user is moving along a given route and using a *mobile device* to access certain data, e.g., information about the route of his trip or the amount of traffic. Along the route, *IoT devices* are installed at various geographic (latitude/longitude) locations. These IoT devices can be sensors

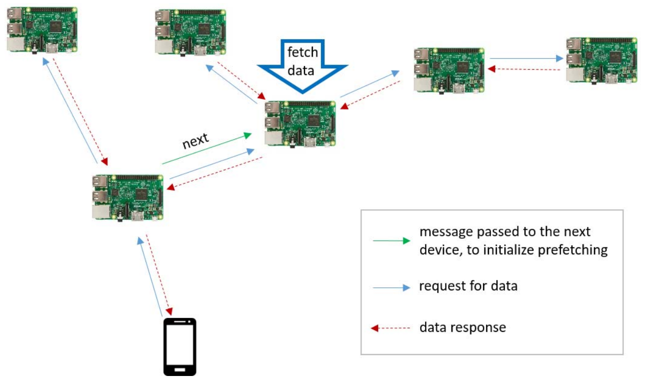
fetch data
next
message passed to the next device, to initialize prefetching
request for data
data response

Fig. 1. Architecture of our prefetching solution

such as traffic or parking sensors, but can also be small computing devices, comparable to Raspberry Pi computers. The user's mobile device is connecting to various IoT devices along its route. Furthermore, the mobile device is aware of its future route, i.e., the geographic target which the user is approaching. This can stem from sources like a turn-by-turn navigation system, or by employing user mobility prediction [11]. This implies that it is possible to determine the IoT device to which the mobile device will connect next. The IoT devices communicate with each other on a peer-to-peer basis, i.e., a central controller is not required in general. Furthermore, not all IoT devices hold all IoT data. Instead, each IoT device is responsible for a separate subset of data, but can query other IoT devices if additional data is required. Therefore, the IoT device to which the mobile device is currently connected can fetch data required by the mobile device from other (neighboring) IoT devices. An overview of the overall architecture of our solution is shown in Figure 1.

We demonstrate our approach by providing a system for querying public transport traffic information (e.g., departure time of next bus) from a major public transport network operator in Vienna, *Wiener Linien*. In our scenario, the user moves through a landscape of IoT devices, and can submit queries for this data. Using prefetching, our solution ensures that data required in the foreseeable future, i.e., data about route parts which will be required soon according to the planned transit path, are prefetched by the IoT devices along the path.

In the following, we describe the various components of our solution. Note that only the mobile device application itself is executed on the mobile device. All other components are running on the IoT devices, which reduces workload for the mobile device, avoiding severe reduction of battery life.

A. Mobile Device Application

The application running on the mobile device is responsible for providing the user with required information. In our case, this "payload" is live traffic data from the Wiener Linien network. The mobile device is connected to one given IoT device at a time, and can request information about live public transport data. Upon user request, the mobile device application queries the currently connected IoT device for this information, which may either directly respond to the query (if the data is present on the given IoT device), or forward the request to another IoT device (e.g., if data concerning a platform is queried for which the current IoT device does not have any information).

B. IoT Device Application

On the IoT devices themselves, an application consisting of multiple services is deployed.

Data Prediction Service This service is responsible for predicting which data is most likely to be required by the user. As input, this service takes data about the current location and planned destination of the user together with the current speed, and it returns a list of possibly relevant data items.

User Mobility Prediction Service The user mobility prediction service forecasts the geographic path which the user is expected to take. For each IoT device, provided with the same information as the data prediction service, this method returns the point in time at which the user will come within range, and how long the user is expected to stay in range.

Time Prediction Service Finally, this service provides information about when to start prefetching data, and which data items should be prefetched by which IoT device. This service uses the output of the preceding services to determine how much data can be transferred to the user during the time of connectivity.

In our current implementation, these three services are realized using relatively simple methods. We use linear extrapolation to determine the user's future path and speed, and simply use the size of data items to estimate which data items to transfer. As we will show in Section III, we only use a linear sequence of IoT devices, which significantly simplifies the functionality of these services.

In a real-world IoT scenario, more sophisticated techniques can be used for these predictions. Our modular solution allows replacing these implementations with suitable techniques. For instance, our earlier work proposes using geographic coordinates together with Kalman filters to create a more suitable approximation as a means of smoothing signals [3].

III. Implementation and Evaluation

In this section, we describe the implementation of our solution. Furthermore, we evaluate the solution in a real-world testbed.

A. Implementation

In order to evaluate our approach, we use a testbed consisting of five Raspberry Pi computers representing the IoT devices. These IoT devices have IDs numbered 1 through 5, and are placed on two separate floors of the institute building, separated by distances between 10 and 30 m. Naturally, in a real IoT setting, these devices are further apart, and are only

TABLE I
TEST RUNS

Test Run	Start Device	Device Direction	Link Speed	Prefetching Enabled
1	1	ascending	1 kB/s	yes
2	2	ascending	1 kB/s	yes
3	1	ascending	1 kB/s	yes
4	5	descending	1 kB/s	yes
5	1	ascending	1 kB/s	no
6	2	ascending	1 kB/s	no
7	1	ascending	1 kB/s	no
8	5	descending	1 kB/s	no
9	1	ascending	5 kB/s	yes
10	2	ascending	5 kB/s	yes
11	1	ascending	5 kB/s	yes
12	5	descending	5 kB/s	yes
13	1	ascending	5 kB/s	no
14	2	ascending	5 kB/s	no
15	1	ascending	5 kB/s	no
16	5	descending	5 kB/s	no

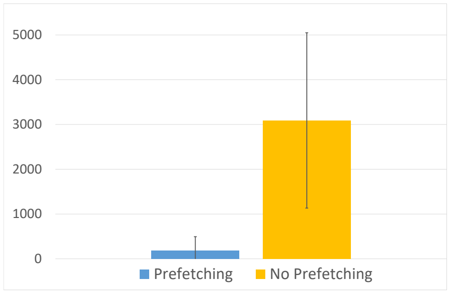

Fig. 2. Average response time, link speed of 1 kB/s (error bars denote σ)

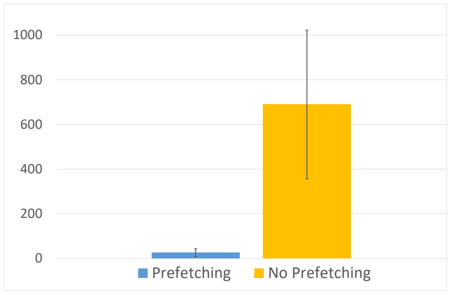

Fig. 3. Average response time, link speed of 5 kB/s (error bars denote σ)

connected by a slow mobile connection. We therefore simulate this slow mobile connection between the IoT devices using the SpiceJ library [14].

While our approach does not depend on a given communication method or protocol, and in theory, any kind of connectivity can be used, we use Wi-Fi in our scenario. Each IoT device creates a Wi-Fi network, to which the mobile device can connect. In our test setup, the mobile device travels along the IoT devices either in ascending order, i.e., from the lowest to the highest IoT device ID, or in descending order, depending on the test run configuration, as discussed in Section III-B. We perform some runs in descending order to achieve a reference on whether the ordering influences the results. Additionally, the mobile device queries the IoT devices for live traffic data provided by the Wiener Linien API[1]. This API represents platforms using identifiers, which are called *RBL* numbers[2], and we use these RBL numbers to distinguish platforms. Each RBL number is assigned a geographic position.

In order to create a realistic setup, we inhibit the available data of each IoT device to only read certain subsets of information. To this end, we divide the city area of Vienna into six rectangles based on geographic coordinates, and based on its location, each IoT device is assigned one of these areas. If a user requests information about RBL numbers (platforms) within a different area, the IoT device will ask the neighboring devices for this information. Otherwise, if all IoT devices had access to all data, prefetching would be redundant. To this end, the queries used in this experiment are selected in a way that ensures that multiple IoT devices must be considered to respond to the query (i.e., data stemming not only from the currently connected IoT device is requested).

In our testbed, the mobile device is implemented using an Android smartphone, and the software running on this device is an Android app written specifically for this evaluation. The mobile device application allows the user to query the aforementioned Wiener Linien live data. The IoT devices are running a Java application consisting of the three services described in Section II. The mobile devices are configured to connect to the Wi-Fi network with the strongest signal, and perform roaming upon insufficient network quality with increased roaming sensibility, enabling a smooth hand-over and avoiding "blind spots" where no Wi-Fi network is available.

B. Evaluation

In order to show that the proposed prefetching reduces the overall time until a data item becomes available, we perform 16 test runs with various configurations, and record the resulting response time. Out of these 16 test runs, 8 are performed with prefetching, and 8 without. Furthermore, 12 test runs are performed in ascending order, and 4 test runs are performed in descending order. In order to analyze the impact of connection speed between the mobile device and the IoT devices, we use two different link speeds (1 kB/s and 5 kB/s). Table I gives an overview of the test run configurations.

Using a link speed of 1 kB/s and no prefetching, it takes an average of 3091 ms ($\sigma = 1958$) per data item to receive a response. With prefetching enabled, this number is reduced to only 180 ms ($\sigma = 314$). Similarly, for a link speed of 5 kB/s and no prefetching, an average response time of

[1]API endpoint: http://www.wienerlinien.at/ogd_realtime/monitor.

[2]RBL stems from the German word Rechnergesteuertes Betriebs-Leitsystem (computer-aided operation management system), an ICT system for management of public transport fleets.

689 ms ($\sigma = 333$) is achieved. With prefetching enabled, the average response time is 26 ms ($\sigma = 18$). Therefore, prefetching enables a reduction in response time per data item by 2911 ms and 663 ms for link speeds of 1 kB/s and 5 kB/s, respectively. A Student's t-test shows the significance of both reductions, yielding p-values of < 0.001 for both link speeds. The order of evaluation (ascending or descending) did not significantly influence the results.

IV. Discussion

This work shows that data prefetching significantly reduces response time and therefore can lead to an improved user-perceived QoE. We have implemented a relatively simple solution, with a sequential alignment of IoT devices, which naturally does not correspond to the network topology of a large-scale IoT scenario. Furthermore, the computational capability of actual IoT devices may vary. In our scenario, we use Raspberry Pi computers, which, compared to battery-operated low-power sensors, are relatively powerful. Additional optimization and reduction of requirements for (at least some) IoT devices may be necessary to deploy our solution in a large-scale IoT scenario.

Furthermore, another limitation is that while fetching data items from neighbors, IoT devices currently wait for the complete response before forwarding these data items to the mobile device. This could be performed in a stream-based, byte-by-byte manner, which would further accelerate data transmission. Another possibility for improvement lies within the three services in the software running on the IoT devices (data prediction service, user mobility prediction service, and time prediction service). These services were implemented using simple algorithms which were suitable for our evaluation testbed, but better techniques are needed for the prediction of these variables in a real-world scenario. Finally, the network topology of the IoT devices was not covered in our work, and instead of a pure peer-to-peer layout, Fog computing [13] can be used for coordination between IoT devices.

Nevertheless, we have shown the feasibility of prefetching in a scenario with a mobile device fetching data from IoT devices. Our results show that prefetching has the potential to significantly reduce the response time, improving user experience. We have also provided a reference solution, highlighting important components and subsystems in our implementation.

Acknowledgment

Research presented within this work is partially funded by COMET K1, FFG - Austrian Research Promotion Agency, within the Austrian Center for Digital Production.

References

[1] G. Akhras. "Smart materials and smart systems for the future". In: *Canadian Military Journal* 1.3 (2000), pp. 25–31.

[2] L. Atzori, A. Iera, and G. Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[3] M. Borkowski, C. Hochreiner, and S. Schulte. "Moderated Resource Elasticity for Stream Processing Applications". In: *Euro-Par 2017: Parallel Processing Workshops*. 2017, pp. 5–16.

[4] M. Borkowski, O. Skarlat, S. Schulte, and S. Dustdar. "Prediction-Based Prefetch Scheduling in Mobile Service Applications". In: *IEEE International Conference on Mobile Services (MS)*. IEEE. 2016, pp. 41–48.

[5] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021*. 2017.

[6] K. Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[8] W. Hummer, S. Schulte, P. Hoenisch, and S. Dustdar. "Context-aware data prefetching in mobile service environments". In: *IEEE International Conference on Big Data and Cloud Computing (BdCloud)*. IEEE. 2014, pp. 214–221.

[9] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. K. Dey. "Factors Influencing Quality of Experience of Commonly Used Mobile Applications". In: *IEEE Communications Magazine* 50 (4 2012), pp. 48–56.

[10] P. Mell and T. Grance. "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology". In: *National Institute of Standards and Technology, Information Technology Laboratory* 145 (2011), p. 7.

[11] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. "Mining user mobility features for next place prediction in location-based services". In: *IEEE International Conference on Data Mining (ICDM)*. IEEE. 2012, pp. 1038–1043.

[12] V. A. Siris and D. Kalyvas. "Enhancing mobile data offloading with mobility prediction and prefetching". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 17.1 (2013), pp. 22–29.

[13] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. "Optimized IoT service placement in the fog". In: *Service Oriented Computing and Applications* 11.4 (2017), pp. 427–443.

[14] *SpiceJ*. https://github.com/michael-borkowski/spiceJ. [Online; accessed 20-June-2018].

[15] S. Yinbiao, K. Lee, P. Lanctot, F. Jianbin, H. Hao, B. Chow, and J. Desbenoit. "Internet of things: wireless sensor networks". In: *White Paper, International Electrotechnical Commission* (2014).

[16] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. "Internet of things for smart cities". In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32.