An Automated Health Monitoring Solution for Future Internet Infrastructure Marketplaces

Yahya Al-Hazmi*, Alexander Willner*, Ozan O. Özpehlivan*, Daniel Nehls*, Stefan Covaci*, and Thomas Magedanz†
*Chair of Next Generation Networks, Technical University Berlin, Berlin, Germany
*Email: {yahya.al-hazmi | alexander.willner | ozan.o.oezpehlivan | daniel.nehls | stefan.covaci}@tu-berlin.de
†Next Generation Network Infrastructures, Fraunhofer FOKUS, Berlin, Germany
†Email: thomas.magedanz@fokus.fraunhofer.de

***Abstract*—Worldwide a large number of Future Internet and Smart City infrastructures exist. To provide a global view on these infrastructures in Europe, the Infinity Project has developed an on-line catalog called the XiPi portal. Its main objective is to facilitate the construction of a sustainable market for infrastructure providers to advertise their capabilities and capacities for end-users. In this context, the requirement to provide up-to-date availability status information of individual infrastructures was raised. We introduce an architecture to provide these high-level monitoring information about the health of the involved infrastructures and their services by adopting existing Future Internet Research and Experimentation (FIRE) technologies. The approach has been integrated as an extension into the portal and selected infrastructures advertise their availability.**

***Index Terms*—Health Monitoring, Future Internet Infrastructures, Experimental Facilities**

I. Introduction

Emerging communication and networking paradigms and technologies such as Cloud Computing, Software Defined Networking (SDN), Network Function Virtualization (NFV), Machine-To-Machine Communication (M2M) and Internet of Things (IoT) are changing the current IT and Telecom domains. They bring forth new business models and opportunities towards building Smart Cities, eHealth, eGovernments, eLiving, and other services and applications.

For such new services and applications to be trialed and evaluated at scale, convenient experimental environments that support, or are enabled with, the aforementioned paradigms and technologies are required, as the transition from theoretical or simulation based research into production is not always the optimal strategy; especially if the new developed services or protocols will be deployed in large-scale or across heterogeneous networks. It is therefore foreseen, that experimentally driven research conducted on large-scale and real-world facilities is essential for Future Internet research and development. This will open the door for researchers, applications developers and Small and Medium Enterprises (SMEs) to study and test their new ideas and products in controllable and cost-effective environments.

Currently, there are a large number of testbed infrastructures worldwide built for experimenting and prototyping Smart Cities and Future Internet applications and services. To provide a global view on the existing Future Internet infrastructures in Europe, the European FP7 INfrastructures for the Future INternet CommunITY[1] (INFINITY) project has developed an on-line catalog of infrastructures called the XiPi portal[2]. Its main intention is to facilitate building a sustainable market for the infrastructure providers and operators to advertise their capabilities and capacities for end-users. They can then browse through the portal and find suitable infrastructures based on different criteria.

It was desired to provide information about the availability status of the involved infrastructures through the XiPi portal.

However, in order to provide an attraction solution with minimum efforts needed from the infrastructures, some considerations should be taken into account. First, the monitoring tools that are already in place at the infrastructures will maintain and be used for executing the measurements. Second, monitoring information about the infrastructure availability status across many infrastructures should be provided in a common data format. In order to achieve this, there is a need for a suitable solution as this is not possible using any of the existing monitoring tools as stand-alone solution.

In this paper we introduce our extension to the XiPi portal by bringing a new monitoring feature that is in charge of providing high level monitoring information about the health and availability status of the involved infrastructures and their services in a common manner. Our approach is based on providing the monitoring information through a common interface that allows all the involved infrastructures to provide their data in one single format. This overcomes the possible misinterpretation of the collected data that are otherwise provided in different data format. Our design and implementation of this feature is discussed in this paper as its main contribution.

The remainder of the paper is structured as follows. We give a brief overview of the design requirements and the architecture in Sec. II. In Sec. III, the implementation is presented. Finally, we close giving some conclusions and considerations and describe future work in Sec. IV.

[1] http://fi-infinity.eu
[2] http://xipi.eu

978-0-9836283-9-2

II. Architectural Design and Requirements

The XiPi portal offers an online catalog for multiple stakeholders. On the one hand, it allows users to search for the most suitable infrastructures that fit their requirements. On the other hand, it is of added-value to the infrastructure providers, since it enables to advertise their offerings and to increase the visibility to various Future Internet communities. In order to build a basis for trustworthiness between users and infrastructure owners, advertising health information through the portal is envisioned. In order to implement such an automated information update, several requirements can be identified and multiple approaches can be considered.

A. Underlying Design Decision

Infrastructures may use different monitoring tools internally that might use different databases and Application Programmers Interfaces (APIs) as well as various data formats. In order to efficiently collect information data across the involved infrastructures, the data should be provided following one single format. To update the availability statuses of different infrastructures, we consider three different methods for infrastructures to expose their status. These are based on widely-used and well-recognized protocols and technologies in both Future Internet Research and Experimentation [1][3] (FIRE) and Future Internet Public Private Partnership [2][4] (FI-PPP) programs.

1) OMSP based push method: Each infrastructure can provide information about its key components regularly by following a push method to a central collection point of the XiPi portal. This architectural decision excludes additional responsibility, and possible complexity, of the portal. The OML Measurement Stream Protocol[5] (OMSP) can be used for this purpose and has already been used within Federation for FIRE [3] (Fed4FIRE) for these purposes. Thus, each infrastructure has to be able to provide monitoring data pushed regularly as OMSP streams with the help of existing ORBIT Measurement Library [4] (OML) implementations to a central OMSP server that is offered at the portal level.

2) SFA based pull method: The status of the key components of an infrastructure can also be requested by the portal in a pull manner. This can be achieved by Slice-based Federation Architecture [5] (SFA) enabled infrastructures by extending the implementation of either the `getVersion()` or the `listResources()` method calls on the infrastructure side. The XiPi portal has to be aware of the Uniform Resource Locator (URL) of the SFA Aggregate Manager (AM) of each infrastructure and, depending on the implementation, also has to have valid credentials for each infrastructure. The portal has then to invoke the according method that includes in its response the statuses of the key components of the respective infrastructure.

[3]http://ict-fire.eu
[4]http://fi-ppp.eu
[5]http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html

3) FI-WARE Monitoring Generic Enabler method: Infrastructures could publish the status of their key components by offering a Future Internet Core Platform [6] (FI-WARE) Monitoring Generic Enabler (GE) compatible interface. The XiPi portal has to be aware of this interface and then either call these to check the status of the components or registers a callback URL.

Although any of these three options could be conceptually used, we have decided to implement an OMSP based interface for the XiPi portal. Our decision is made based on three considerations:

- Given the fact that most of the FIRE facilitie shave already adopted the OMSP based approach to provide monitoring information in a common way across infrastructures, this was a suitable starting point for us to accelerate having many infrastructures in board in a short time with a very limited efforts from the infrastructures to be compliant with our solution.
- The push method has its advantages. It allows infrastructure owners to advertise and provide information about their key services that might only be available at their infrastructures and they want them to be visible to wider communities. They can even dynamically reduce or expand their advertisements. In contrast, using any of the other two options that follow a pull mechanism, we might go for retrieval a static set of data across all infrastructures. Thus, the solution at portal level has to fetch the status information of pre-defined set of components or services from the individual infrastructures.
- Using a pull mechanism adds complexity including responsibility to the portal. The solution at portal level should know and regularly contact many URLs of data sources at infrastructure level (at least one per infrastructure or possibly one per component) to retrieve the concerned data. However, in the push method, all infrastructures need only to know one single URL of the monitoring collection server at portal level.

B. Overall Architecture

Following the centralized model of the XiPi portal, the proposed health monitoring solution is adopting the same approach. Fig. 1 illustrates an overview of the selected architecture. An infrastructure together with its capabilities and services is registered by its owner manually through the portal (cf. Fig. 2). Each infrastructure will then provide monitoring information about the health of the infrastructure in a regular basis as OMSP streams (cf. Fig. 3). These data are received and processed by a monitoring module through its southbound OMSP interface. It then stores both the status of the individual components and services as well as the overall infrastructure status in a database. Finally, users can monitor the status of any infrastructure through a Graphical User Interface (GUI) that retrieves all data from the database through a northbound REST interface offered by the monitoring module.

978-0-9836283-9-2

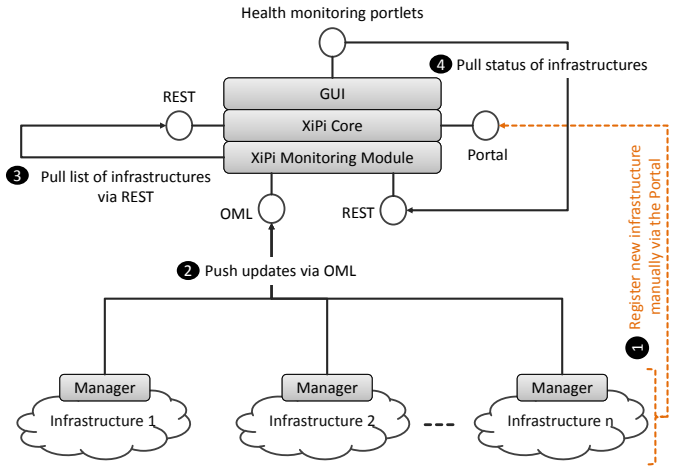

Fig. 1. Architecture Overview

C. Infrastructure Identification

Currently the XiPi portal includes static information about the natures and capabilities of over than 200 Future Internet infrastructures that have been already registered via a form-based manual process. This information is visible through the XiPi portal for public visitors. However, the discussed monitoring integration first has to identify a list of infrastructures that are able to provide their availability status information. This is achieved through invoking the XiPi database that includes names of infrastructures with further information through its XiPi Representational State Transfer (REST) interface. Fig. 2 gives an overview of how the list of infrastructures is added to the XiPi database, how this list through a database client is retrieved by the monitoring module that then updates its own database.

D. Information Publication

In order to update information of the monitoring module, the infrastructure has to push the according information. Fig. 3 illustrates a sequence diagram for pushing these data and how this information is provided to the users.

E. Information Processing

Monitoring information is received at the portal level by a monitoring module. It is in charge of processing the data, calculating the status of the infrastructure as a whole and then making its status as well as statuses of its individual components available to be shown through the health monitoring web fronted.

F. Data Storage

The objective of the facility monitoring feature is to expose the availability status of the infrastructures. Therefore, only the last updated statuses of the key components of each infrastructure along with their last check times are of importance. To this end, there is no need to store all data records (received updates), but rather the latest data (latest received update).

III. Implementation in XiPi

Based on the presented architecture, an implementation of the functional elements at both the infrastructure and the XiPi portal side has been deployed.

A. Data Format

In Listing 1, an example of a valid OMSP stream is depicted. The Lines 1 to 11 are header information and define meta information about the monitoring data that start in Line 12.

Listing 1. Monitoring Data as OMSP Stream

```
1  protocol: 4
2  domain: FOKUS FUSECO Playground
3  start-time: 0
4  sender-id: fuseco.fokus.fraunhofer.de
5  app-name: fiteagle
6  schema: 0 _experiment_metadata subject:string key:
       string value:string
7  schema: 1 epc_client statusMessage:string up:double
       last_check:string
8  schema: 2 wifi statusMessage:string up:double
       last_check:string
9  schema: 3 fiteagle statusMessage:string up:double
       last_check:string
10 content: text
11
12 0.674263000488 1 0 fine 1 2013-03-14T12
       :34:34.102734+02:00
13 0.674374103546 2 0 up and running 1 2013-11-08T10
       :29:57.273166+01:00
14 0.674427986145 3 0 executing server update 0 2013-11-08
       T10:29:57.273166+01:00
15 ...
```

In order to be able to map these data with the information of the XiPi database, the domain (see Line 2) must correlate with the infrastructure name within the XiPi portal database the according database entry. Furthermore, the following schema definition is required (see Lines 7 to 9 and Lines 12 to 14):

`schema: <id> <identifier> <value 1> <value 2> ... <value n>`

Where,

- `id`: schema identifier
- `identifier`: must equal the component name in the monitoring database
- `value 1`: a text message to provide further information
- `value 2`: must be `1` if the component is up and `0` if the component is down
- `value 3`: must be a text message

Given the heterogeneity and diversity of the involved testbeds, no static schema or semantic annotations are prespecified for the different resources. They can be defined by each involved facility.

In the header, the `domain` tag (Line 2) contains the infrastructure name. In Lines 7-9, the structure/schema of the infrastructure component monitoring information is defined. In Line 7 for instance:

- `1` is the schema number,
- `epc_client` is the component name,
- `statusMessage` is defined to provide human readable information,
- `up` hold the information about the component status (0=down, 1=up), and
- `last check` indicates when the components status was checked.

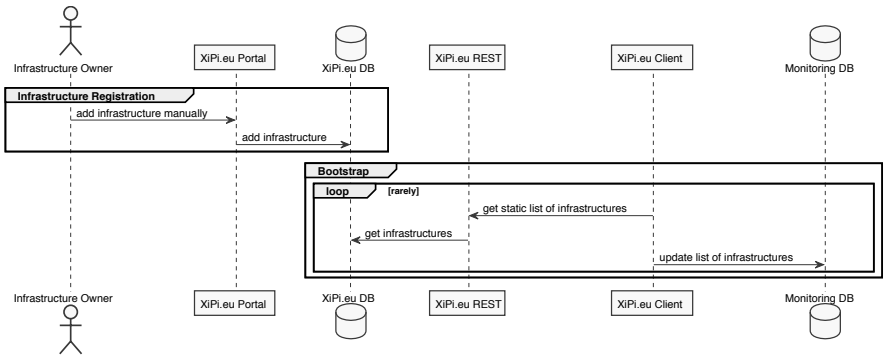

Fig. 2. Overall Sequence Diagram for Adding Static Information

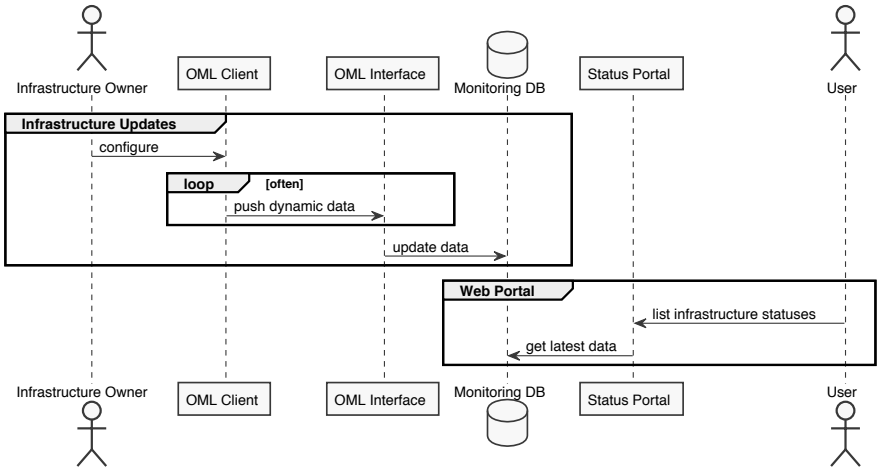

Fig. 3. Sequence Diagram for Pushing Dynamic Information

This structure must be kept while pushing the information. The schema number and the schema name must be modified for more components as it is the case in Lines 8 and 9.

The second part (Lines 12-15) is the actually pushed monitoring data, which has the structure defined in the `schema`. For example Line 14 belongs to the component with the schema number 3, the resource is down, the status message says `executing server update`, and the information was generated at `2013-11-08T10:29:57.273166+01:00`. In Line 12, the resource (with the schema name `epc_client` and schema number `1`) is up and running, but the last checked date can be considered as not up-to-date.

B. Implementation at the Infrastructure

The solution has been designed with a minimum implementation effort at the infrastructure level. Infrastructures maintain using their local monitoring systems but they should provide the data through common interface and data model.

Each infrastructure has to push the respective monitoring information regularly as OMSP streams. However, this supposes that the infrastructure has the ability to provide data in this format. Fortunately, existing OML implementations can be reused that have binding to different languages such as C, C#, Ruby, Python and Java. All what an infrastructure needs to do is to deploy any of these OML client libraries and to write a simple script (compliant with the selected library) acting as a wrapper that fetches the concerned monitoring data from the local monitoring tools, which is used anyway for internal administrative purposes, and forwards it to the selected OML library that encapsulate the data in OML streams towards the central collection server at portal level.

In this perspective, a wrapper script along with any OML library acts as adaptation mechanism to convert the data from native data formats at infrastructures into the common OMSP format. Within this project, an example script has been provided to be used with the Zabbix monitoring tool that is used at many infrastructures for local monitoring.

OML streams are pushed in a regular basis, and the provided data are published with its original timestamp. The higher the update rate, the more accurate the provided data are. Therefore, in order to keep publishing the data in an up-to-date basis, the infrastructure should provide the data in a faster and reasonable update rate. We recommended using an update rate of maximum 60 seconds. Thus, partial data might be provided after up to maximum one minute from its production time. Yet, the status information published via the portal are based on the actual data together with their original timestamps. Nevertheless, we could increase the accuracy by reducing the update rate.

C. Implementation at the Portal

This section represents the implementation of the two main components of the health monitoring service at the XiPi portal level: the monitoring module and the monitoring frontend.

1) Monitoring Module: Following the presented architecture, the monitoring module queries the XiPi REST interface to derive information about infrastructures from the XiPi database that includes a list of all registered infrastructures. It offers two interfaces to interact with the outside world. A southbound interface towards the infrastructures is implemented as an OMSP interface for receiving monitoring information pushed by infrastructures. Monitoring streams are processed within this module. A northbound REST interface is used by the health monitoring frontend to retrieve status information of the all infrastructures and their individual components and services. This interface can also be offered to the external application developers.

The XiPi monitoring module checks internally the status of the components regularly. If the last checked date of a component is `too old`, it will be deleted from the components list. If the last checked date of a component is `old`, but not `too old` and the status information pushed as `up`, it will be marked gray in the component list. Amongst others these are also checked periodically and the components list is updated accordingly. For the components, the setting of the time to be `too old` or `old` and also the time period of the internal checks can be done over java preferences.

2) Health Monitoring Frontend: The health monitoring service has been integrated with the environment used by the XiPi portal (Liferay). It is implemented as two integrated portlets and a web service. The layout of the GUI is embedded into the existing portal and in the main page all infrastructures stored in the monitoring database are listed (cf. Fig. 4). They are ordered by their status and name.

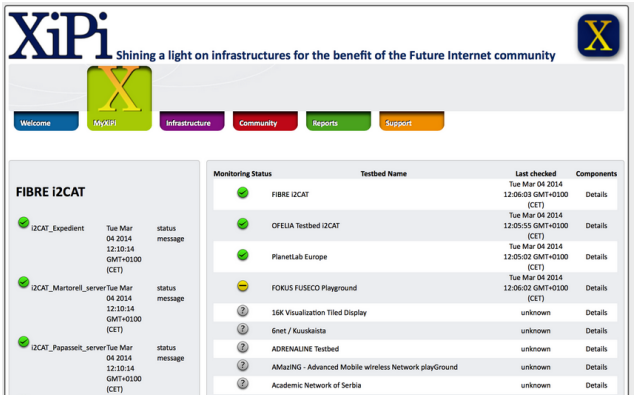

Fig. 4. XiPi Monitoring GUI

We distinguish between four different statuses that are described in Table I. An infrastructure status depends on the statuses of the containing components. By clicking the `details` button, the statuses of the components are derived from the REST interface of the XiPi monitoring module. The details are shown on the left side. The statuses of the individual components are shown as icons and their last check dates are listed as well. Also the calculated overall status and the oldest last checked value are shown for the infrastructure. By clicking the `status message` button on the component details, the reason of the current status (which is pushed by the infrastructure) is shown.

Tab. I
POSSIBLE STATUSES OF AN INFRASTRUCTURE

Short	Icon	Description
Up		All components are up and running.
Outdated		At least one component was not updated within a given threshold.
Partial		At least on component is down.
Down		All components are down.
Unknown		No monitoring information are available for the given testbed.

The overall status of the infrastructure is calculated following an algorithm that is characterized in Algorithm 1 and the corresponding Fig. 5.

Data: Collection C of component statuses of an infrastructure.
Result: Status T of the infrastructure.
begin
  $T \longleftarrow UNKNOWN$
  forall the *components in* C *as* S **do**
    if S == *UP* **then**
      if T == *DOWN* **then**
        T = Partially
      end
      if T == *UNKNOWN* **then**
        T = UP
      end
    end
    if S == *OUTDATED* **then**
      if T == *DOWN* **then**
        T = PARTIAL
      end
      if T == *UNKNOWN* **then**
        T = OUTDATED
      end
      if T == *UP* **then**
        T = OUTDATED
      end
    end
    if S == *DOWN* **then**
      if T == *UNKNOWN OR* T == *DOWN* **then**
        T = DOWN
      else
        T = PARTIAL
      end
    end
    if S == *UNKNOWN* **then**
      if T == *UNKNOWN* **then**
        T = UNKNOWN
      else
        if T == *DOWN* **then**
          T = DOWN
        else
          T = PARTIAL
        end
      end
    end
  end
end

Algorithm 1: Status Calculation of an Infrastructure

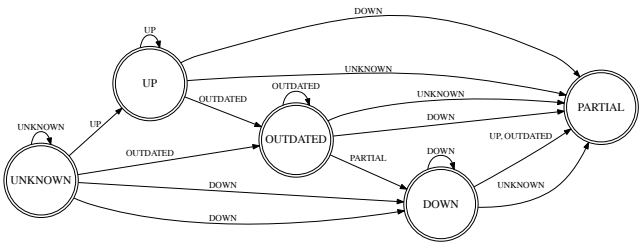

Fig. 5. Deterministic Finite Automaton DFA of the Infrastructure Status Algorithm

IV. Conclusions and Future Work

We have presented an architecture to provide monitoring information about the health and availability status of Future Internet infrastructures by using a common interface to publish related in a single format. It has been shown that FIRE related technologies, namely OMSP compliant implementations, can be used to publish these data to and integrate them into the XiPi portal. Furthermore, guidelines for infrastructures on how to be compliant with the XiPi monitoring solution have been discussed.

Following this approach, the trustworthiness between users and facility owners is being increased. Users can observe the health of monitored infrastructures and their owners can transparently communicate the robustness of the facility. However, only XiPi's registered users can benefit from this service, this decision made by the XiPi as one of its future and sustainability strategies. Yet, the registration is open to any community. Even though the solution presented in this paper is described specific to its implementation and validation within the XiPi portal, the solution is in principal also applicable to other similar environments, where multiple infrastructures cooperate in a federated manner.

Further research is currently being conducted to define a semantic information model that will facilitate the automatic mapping between monitored and advertised resources of an infrastructure. Additionally, involving more infrastructures as well as improving the performance of the solution remains for future work.

Acknowledgments

Research for this paper was partially financed by the EU FP7 project INFINITY (grant agreement no. 285192) and the EU FP7 project Fed4FIRE (grant agreement no. 318389). We thank our project partners for their contributions and their collaboration to this research work.

References

[1] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future Internet Research and Experimentation: The FIRE Initiative," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 89–92, 2007. [Online]. Available: http://doi.acm.org/10.1145/1273445.1273460

[2] D. Havlik, S. Schade, Z. Sabeur, P. Mazzetti, K. Watson, A. J. Berre, and J. L. Mon, "From Sensor to Observation Web with environmental enablers in the Future Internet," *Sensors*, vol. 11, no. 4, pp. 3874–907, Jan. 2011. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3231333&tool=pmcentrez&rendertype=abstract

[3] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, A. Quereilhac, Y. Al-hazmi, F. Lobillo, C. Velayos, A. Vico-oton, and G. Androulidakis, "Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities," in *Future Network and Mobile Summit*, 2013.

[4] O. Mehani, G. Jourjon, J. White, T. Rakotoarivelo, R. Boreli, and T. Ernst, "Characterisation of the Effect of a Measurement Library on the Performance of Instrumented Tools," Tech. rep. 4879. NICTA, Tech. Rep., 2011. [Online]. Available: http://olivier.mehani.name/publications/2011mehani_oml_performance.pdf

[5] L. Peterson, S. Sevinc, J. Lepreau, and R. Ricci, "Slice-based Federation architecture," GENI, Tech. Rep., 2009. [Online]. Available: http://groups.geni.net/geni/wiki/SliceFedArch

[6] A. Glikson, "FI-WARE: Core Platform for Future Internet Applications," in *Proceedings of the 4th Annual International Conference on Systems and Storage*, Haifa, 2011.