BRACELET: Edge-Cloud Microservice Infrastructure for Aging Scientific Instruments

Phuong Nguyen, Tarek Elgamal, Steven Konstanty, Todd Nicholson, Stuart Turner, Patrick Su, Klara Nahrstedt,
Timothy Spila, Roy H. Campbell, John Dallesasse, Michael Chan, Kenton McHenry
University of Illinois at Urbana-Champaign

***Abstract*—Recent advances in cyber-infrastructure have enabled digital data sharing and ubiquitous network connectivity between scientific instruments and cloud-based storage infrastructure for uploading, storing, curating, and correlating of large amounts of materials and semiconductor fabrication data and metadata. However, there is still a significant number of scientific instruments running on old operating systems that are taken offline and cannot connect to the cloud infrastructure, due to security and network performance concerns. In this paper, we propose BRACELET - an edge-cloud infrastructure that augments the existing cloud-based infrastructure with edge devices and helps to tackle the unique performance & security challenges that scientific instruments face when they are connected to the cloud through public network. With BRACELET, we put a networked edge device, called cloudlet, in between the scientific instruments and the cloud as the middle tier of a three-tier hierarchy. The cloudlet will shape and protect the data traffic from scientific instruments to the cloud, and will play a foundational role in keeping the instruments connected throughout its lifetime, and continuously providing the otherwise missing performance and security features for the instrument as its operating system ages.**

I. INTRODUCTION

With the proliferation of digital technologies, instrumentation, and pervasive networks for data collection, sharing, and analysis, there are increasing needs for advanced cyber-infrastructure to support data-driven and interdisciplinary scientific research. However, related efforts [1] mainly focus on homogenous, well-organized data in an offline or batch manner (e.g., in astronomy and high energy physics), and much less effort has been on long-tail data, i.e., data of small or medium sizes collected during day-to-day research, or "dark data", i.e., unpublished data including results from failed experiments and records viewed as ancillary to published studies. Therefore, in material sciences for example, it often takes a long time from the discovery of new materials to fabrication of new and next-generation devices based on the new materials [2].

In order to speed up new discoveries, there have been recent efforts [3, 4] that focus on enabling digital data sharing and ubiquitous network connectivity between scientific instruments (e.g., SEMs, TEMs, AFMs) and cloud-based storage infrastructure for uploading, storage, curation, and correlation of large amounts of materials and semiconductor fabrication data and metadata. However, there is still a significant number of scientific instruments that run their scientific software tools on old operating systems (e.g., Windows XP, Windows NT, Windows 2000). Since these OSes cannot operate at the network speed of a powerful cloud and are not patched with the latest security patches, the instruments are taken offline and cannot connect to the cloud infrastructure. This is because if these instruments were put on the network, they would be destroyed by viruses and might represent major security threats and performance bottlenecks to the very expensive instruments and the overall network infrastructure. Furthermore, this situation will not go away, since instrument companies do not upgrade their instrument software at the same frequency with which the computing companies upgrade their OSes[1]. Even more recent OSs, such as Windows 7, will become obsolete in the near future, and scientific instruments running on Windows 7 will eventually join the group of offline instruments. As a result, the current networked solution for scientific instruments is not evolvable and represents a major barrier to accelerating the pace of discovery and deployment of advanced cyber-infrastructure.

In order to bridge the security and performance gaps between disconnected scientific instruments and cloud-based cyberinfrastructure, in this paper, we propose BRACELET, an edge-cloud infrastructure that introduces networked edge device, called *cloudlet*, in between scientific instruments and cloud as the middle tier of a three-tier hierarchy. The introduction of cloudlets poses several challenges. *First*, cloudlets need to be integrated seamlessly with the existing cloud-based cyberinfrastructure to securely offloading scientific data across edge and cloud. *Second*, recent survey [5] has shown highly dynamic characteristic of data being uploaded from scientific instruments. Hence, it is important for the hierarchical edge-cloud infrastructure to allow data processing across edge and cloud to provide better load balancing at peak workloads. This becomes more challenging considering that the scientific data workloads are often represented by a workflow model with complex interactions and dependencies between tasks.

To address the above challenges, we design BRACELET by extending the state-of-the-art cloud-based microservice cyberinfrastructure [3, 4] to the edges. Specifically, we design an edge-cloud microservice execution and coordination mechanism to support: (1) securely offloading scientific data to the edge and to the cloud, and (2) seamlessy executing scientific workflows across edge and cloud. We have validated the design of BRACELET on real edge-cloud environment of campus cyberinfrastructure and we summarize our contributions as follows:

- We present *the first* edge-cloud microservice cyberinfrastructure that tackles both performance and security challenges of scientific instruments' lifetime connectivity.

[1]It is often that the instrument companies (e.g., GE, Siemens) stop augmenting/updating their scientific softwares when OSes are upgraded to newer versions or when new OS patches come up. Hence, to make use of the instruments, scientific users have to run the instruments on older OSes.

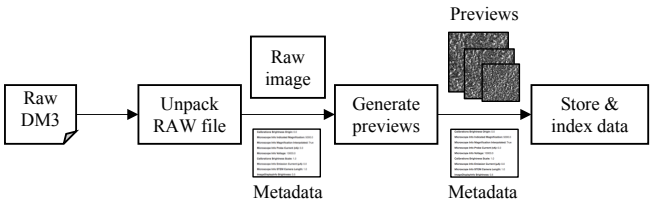

Fig. 1: A sample of simple workflow to process raw DM3 file generated from digital microscope.

- We present a security design to ensure that BRACELET is able to protect vulnerable scientific instruments from external threats.

The remaining of the paper is organized as follows. We first provide some background information about our targeted environment in Section II. In Section III, we give an overview of BRACELET's architecture and we describe in details both the edge-cloud execution model and the security design. After that, in Section IV, we present in details our system implementation. We then present our evaluation results in Section V and in Section VI, we conlude the paper.

II. BACKGROUND

To better understand the target environment of material-related instruments, we provide some background information on the current state of cyberinfrastructure in materials-related environment, types of data generated from the instruments, and example of workflows used to process that kind of data.

The state-of-the-art cyberinfrastructure in materials-related environment [3] uses a two-tier architecture that connects modern scientific instruments directly to a cloud-based infrastructure to support capturing data from instrument, transferring, and processing the digital data in real-time and in trusted manner before archiving, further analysis, visualization for more efficient interpretation and sharing of the experimental results. Data generated from instrument is often in raw format that it requires additional data processing to extract useful information. With materials-related data (and scientific data in general), the data processing step often involves executing a workflow (i.e., a form of a directed acyclic graph) of tasks on the raw data, where, each task performs a specific processing on the data. Figure 1 shows an example of a data processing workflow of three tasks that involves processing of raw DM3 file generated from a digital microscope. In the first task, the raw file is unpacked into image part and metadata. Next, the raw image is processed and multiple previews of different sizes are generated. In the last task, the final data is stored in the database and metadata is indexed to make it searchable. All tasks are performed on the cloud-based infrastructure.

To support processing heterogeneous types of data uploaded from instruments, latest data cyberinfrastructures [3, 4] leverage a microservice-based design, in which each data processing task is modeled as a microservice with independent request queue and a set of task consumers to handle requests (more on this in Section III). This microservice-based approach has demonstrated its effectiveness in handling complex workflows under dynamic workfload situations.

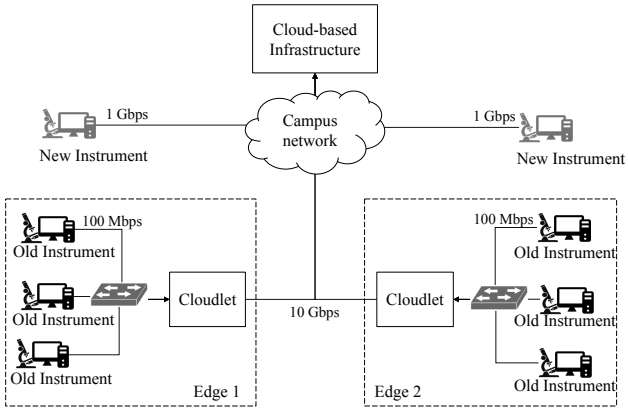

Fig. 2: Overview of BRACELET system.

However, as motivated in Section I, since a large number of scientific instruments are still offline, it is intuitive to introduce an intermediate component (entity), or an edge device, to help to connect these instruments to the cloud-based infrastructure. The introduction of the edge devices not only helps to protect vulnerable instruments from security threats but also opens opportunities for off-loading computation from the cloud-based cyberinfrastructure, to the edges. In addition, as recently shown in [6], the hierarchical edge-cloud architecture is also more efficient in handling peak workloads, compared to a flat edge-cloud architecture that does not take into account the hierarchy structure. In Section III, we present our solution to tackle both computation and security challenges of microservices over hierarchical edge-cloud architecture.

III. BRACELET ARCHITECTURE

An overview of BRACELET's 3-tier architecture is presented in Figure 2. In particular, the first tier, i.e., *instrument tier*, includes scientific instruments attached to computers running old operating systems that could not directly connect to the cloud (the new instruments that run with more advanced operating systems can connect directly to the cloud in the existing 2-tier architecture). On each instrument's computer, users use a uploader client to upload experiment data upstream. The second tier, i.e., the *edge* or *cloudlet* tier[2], includes edge-based devices, or cloudlets, that consist of two network interfaces: one connects to instruments' VLAN and another connects to the cloud via public network. Lastly, on the third tier, i.e., *cloud tier*, we deploy a cloud-based infrastructure that connects to the public network. The cloud-based tier supports data processing, curation, storage, correlation, and search of scientific experiment data uploaded from instruments via cloudlets.

A. BRACELET's Micro-service Architecture

Figure 3 shows the detailed microservice architecture of BRACELET and its performance components. To enable seamless integration of cloudlets to the existing 2-tier cloud-based infrastructure, we design BRACELET by extending the cloud-based microservice architecture [3, 4] to the edges. In particular, while the cloud-based infrastructure operates on the full 5-layer architecture, the cloudlets operates on three

[2]From now, we will use *edge*, *edge device* and *cloudlet* interchangably.

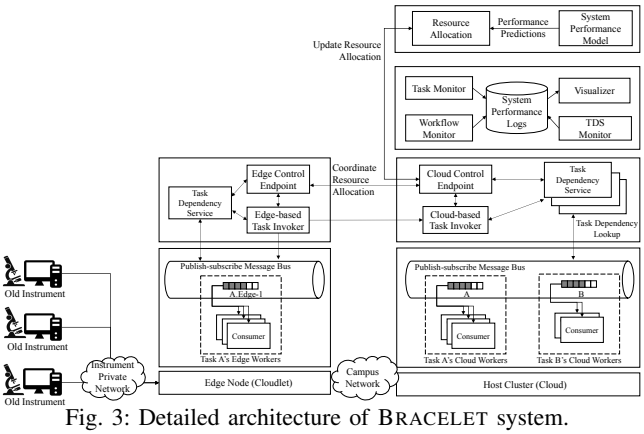

Fig. 3: Detailed architecture of BRACELET system.

layers to enable computational offloading of tasks to the edges and seamless communication between edge- and cloud-based components. In the following, we describe all the layers in details.

1) Infrastructure Layer: Infrastructure layer provides a level of abstraction and virtualization of all computation and storage resources across cloud and edges. We leverage container technology for virtualization and use a container orchestration engine to manage the container allocation across edge-cloud infrastructure.

2) Execution Layer: We design execution layer using a microservice workflow execution model across cloud and edges. Experiment data uploaded from instruments will be handled by a specific type of *data processing workflow*, each workflow type corresponds to a directed acyclic graph (DAG) of a subset (or all) types of *data processing tasks* that system supports. We model each task as a *microservice*[3] with its own *request queue* that stores the task's requests, and a set of *task consumers* that subscribe to the request queue to perform actual processing of the task's requests.

The communication between dependent tasks in a workflow follows the *publish-subscribe mechanism*. When a task request arrives at the queue, a task consumer subscribing to the queue will pick up the request to process it. After processing the request, the consumer asks the coordination layer (to be described shortly) about the subsequent tasks of the workflow and publish the request to the corresponding queues of the subsequent tasks. We assume that all workflow data and intermediate results between tasks are stored in a shared storage system that can be accessed by all microservices across cloud and edges.

A microservice can be deployed on a cloudlet (or multiple cloudlets), on cloud, or on both cloud and cloudlets. The publish-subscribe message bus is available across cloud and edges to enable seamless communication between edge- and cloud-based microservices.

3) Coordination layer: On top of the execution layer is the coordination layer that consists of a *task dependency service*, or TDS, that maintains the dependencies between tasks of a workflow (i.e., task dependencies are essentially the directed edges of workflow's task graph) and responds to task dependency lookups from the execution layer.

The separation of task coordination from the execution of tasks enables more flexible and scalable workflow composition (i.e., we can support new workflow types by simply creating new set of task dependencies between the existing tasks). To offer high availability and high performance, we designed TDS as an ensemble of multiple TDS instances running on both cloud and edges and maintain a replica of task dependency data on each instance.

To coordinate resource allocation across cloud and edges, coordination layer maintains a *control endpoint* on each cloud and edge side. The *cloud control endpoint* is the centralized entity that receives new resource allocation from the adaptation layer (to be described shortly) and informs other *edge control endpoints* to implement new allocation.

4) Monitoring layer: Monitoring layer captures performance metrics of workflows, microservices, and TDS. These metrics are stored in a performance logs database. Performance data is used by adaptation layer (to be described shortly) to make resource allocation decisions. Although monitoring services are running on the cloud, they still can seamlessly communicate with components running on edges to collect the performance metrics, thanks to the deployment of coordination and execution layers across cloud and edges.

5) Adaptation layer: Adaptation layer is the brain of BRACELET system. This layer consists of a *system performance model* that is trained on the performance logs collected by monitoring layer and provides near future performance predictions to help *resource allocation* module to dynamically allocate resources for microservices across cloud and edges.

B. Edge Cloud Microservice Execution Model

Since data can be uploaded to the cloud either via a cloudlet or directly from advanced instruments (which are able to connect directly to the cloud without cloudlet), all microservices have to be deployed on the cloud, so that they can be ready to support processing all types of workflows. For each cloudlet, depending on the types of data that is uploaded from instruments to the cloudlet, microservices of the corresponding data processing workflows have to be deployed on the cloudlet. Therefore, the initial deployment of microservices on cloud and cloudlets can be decided in advance with knowledge of the types of uploaded data[4]. For example, if the system supports all types of workflows, then microservices of all tasks `A`, `B`, `C`, `D` are deployed on the cloud. If only data corresponding to workflow `Wf1` is uploaded through an edge named `E1`, then initial deployment on `E1` will include microservices of the tasks in `Wf1`, namely `A.E1`, `B.E1`, and `C.E1` (i.e., the edge-specific suffix is used to differentiate with cloud-based deployments of `A`, `B`, and `C`).

With the above initial deployment, the execution of a workflow across cloud and edge can be conveniently handled by the cloud control endpoint via dynamic configuration of task dependencies on TDS. For example, to execute a workflow `Wf1` across edge `E1` and cloud (e.g., processing requests of task `A` and `B` on `E1`, and of task `C` on the cloud), the cloud control

[3]From now, we will use *task* and *microservice* interchangably.

[4]This is a reasonable assumption since the type of uploaded data is specific to the type of instrument, which is known information.

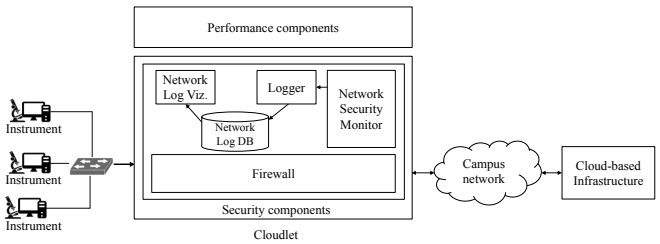

Fig. 4: BRACELET' security component design.

TABLE I: Example dependency table of a cloud workflow `Wf1`.

Job type	From	To
...	...	...
Wf1	Start	A
Wf1	A	B
Wf1	B	C
Wf1	C	End
...	...	...

TABLE II: Example dependency table of edge workflow `Wf1`.

Job type	From	To
...	...	...
Wf1.E1	Start	A.E1
Wf1.E1	A.E1	B.E1
Wf1.E1	B.E1	C
Wf1.E1	C	End
...	...	...

endpoint simply creates a new edge-based workflow type on TDS, namely `Wf1.E1` that directs requests of task `A` and `B` to their `E1`-based microservice deployments, namely `A.E1` and `B.E1` (task `C` is still handled by its cloud-based microservice deployment). Table I shows an example of the dependency table if the tasks of workflow $Wf1$ are placed on the cloud and Table II shows the dependency table if $Wf1$ is placed on the edge. After creating the new workflow type `Wf1.E1`, cloud control endpoint will inform edge control endpoint at `E1` to use `Wf1.E1` as the workflow type to process all requests for `Wf1` of data being uploaded via `E1` (instead of the initial cloud-only version of workflow `Wf1` as shown in Table I).

C. BRACELET's Security Design

BRACELET' security components (Figure 4) are designed to help protect vulnerable scientific instruments once they are connected to the edge-cloud cyberinfrastructure. They consist of a software firewall that is configured with *whitelisting rules* to enable only data traffic from instruments to the cloud and certain control traffic from the cloud to the cloudlet. Furthermore, each cloudlet also includes a network security monitor component to listen to and capture meta-data of all network traffic in and out of the cloudlet. The security monitor component is also capable of applying customizable scripts to filter and analyze network traffic to detect and alert of potential attacks. All network monitoring logs are collected, parsed, and transformed by a *logger* component, and stored into a network logs database. Real-time network traffics and statistics can be queried and visualized to BRACELET's admin by the *network log visualization* component. In our implementation, we use Bro [7] as the network security monitor at cloudlet and use ELK stack [8] (i.e., Elastic-Logstash-Kibana) for logging, storing, and visualizing the collected network security logs.

In addition to data-driven security monitoring and detection at cloudlet, all vulnerable scientific instruments are connected to private instrument network via a managed switch so that instrument's MAC layer address is checked to ensure that the instrument can only talk to cloudlet and not to other peer instruments. At application level, users are required to login on each instrument in order to upload data, and the login sessions are additionally verified with *instrument reservation database* as part of the two-factor authentication process.

IV. SYSTEM IMPLEMENTATION

We implement BRACELET by extending the implementation of the existing cloud-based micro-service infrastructure [3, 4] to the edges. The whole edge-cloud infrastructure cluster is managed by a single Kubernetes [9] container orchestration engine. Specifically, the cloud-based system is deployed on a cluster of two nodes, each node is equipped with an Intel Xeon quad core processor, 1.2Ghz per core, and 16GB of RAM. Two cloudlets, each cloudlet is equipped with Intel Core i7 CPU 3.4Ghz and 8GB of RAM, are connected to the cloud-based system as remote nodes in the Kubernetes cluster. Each cloudlet has its own locality tagging to differentiate it from other cloudlet and cloud-based nodes. This locality tag is used by cloud controller to place computation specifically to the microservices running on the edge. The main reasons why we deploy edge- and cloud-based components in a single Kubernetes cluster are two fold: i) to simplify service discovery between microservice running on edges and cloud (i.e., microservices run on the same overlay network managed by the Kubernetes cluster), and ii) to provide a single, global view of resources for the cloud controller.

Each microservice is implemented with a RabbitMQ [10] request queue and a set of Docker container-based task consumers that are deployed as a ReplicationController[5] set on Kubernetes. TDS service is based on Apache Zookeeper [11] coordination system to ensure strong consistency and high availability. We configure Zookeeper and RabbitMQ using ensemble and cluster mode respectively so that we have a Zookeeper and RabbitMQ endpoint on each edge and cloud side (i.e., to improve availability and enable seamless communication between microservices). Monitoring layer's implementation is similar to the one in [4], and we use Tensorflow [12] to build microservice performance model used in the adaptation layer.

V. EVALUATION

To evaluare BRACELET, we use the MDP workflow ensemble [3, 4] that supports processing experimental data generated by digital microscopes. MDP consists of three types of workflows and four types of tasks. We implemented various microservice computation placement strategies in BRACELET's to show how the cloudlet can help reduce the average delay of processing sudden spikes in the incoming requests. We compare the following placement strategies:

- *Bandwidth-optimized*: Initially, all requests are handled by cloud-based micro-services. When performance guarantee is violated, the processing of requests that arrive from an edge is offloaded to the edge-based microservices. As shown in [6], this approach helps to improve the efficiency of cloud resource utilization when serving the peak workload.

[5]ReplicationController is a concept in Kubernetes that consists of a set of replicas of a container. Kubernetes helps to ensure that a specified number of these container replicas (i.e., corresponds to $m_j(k)$ in our case) are running at any time.

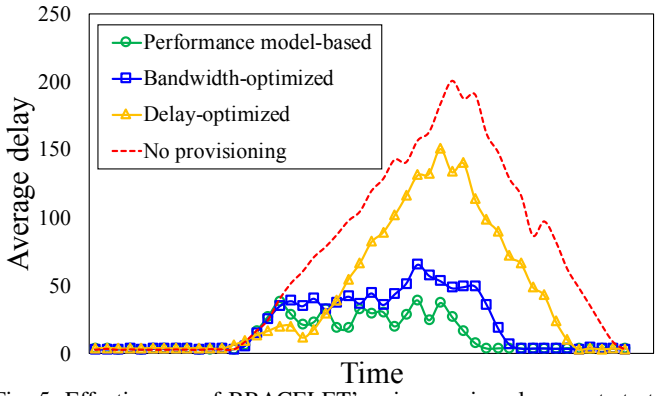

Fig. 5: Effectiveness of BRACELET's microservice placement strategy compared with others.

- *Delay-optimized*: Initially, requests arriving from an edge are handled by edge-based micro-services. When performance violation occurs, the processing of requests is offloaded to the cloud-based micro-services.
- *Performance model-based strategy:* The proposed strategy in *Bracelet* and it is a machine learning-based approach that uses historic data to learn a performance model for the average delay of tasks on both the edge and the cloud. A greedy strategy is then used to decide the placement of tasks across edge and cloud. The approach is described in details in our technical report [13].
- *No provisioning:* All requests are handled by cloud-based micro-services. Edge microservices are never provisioned in this strategy.

The result in Figure 5 shows that the three approaches that utilize the edge are able to handle the sudden spikes in a more efficient manner compared to the *no provisioning strategy*. The *Performance model-based* placement strategy outperforms other approaches by dynamically evaluating different placement options using an accurate performance model [13].

VI. Conclusions and Future Work

In conclusion, we have presented *the first* edge-cloud microservice cyber-infrastructure that tackles both performance and security challenges of scientific instruments' lifetime connectivity.We extended the microservice architecture to support seamless coordination and task across edge and cloud. We also presented a security design to protect vulnerable instruments from security threats.

Acknowledgment

This research was funded by the NSF DIBBs, award number 1443013 and NSF ACI CC, award number 1659293. The opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not reflect the view of the NSF.

References

[1] A. R. Ferguson, J. L. Nielson, M. H. Cragin, A. E. Bandrowski, and M. E. Martone, "Big data from small data: data-sharing in the'long tail'of neuroscience," *Nature neuroscience*, vol. 17, no. 11, p. 1442, 2014.

[2] J. Holdren, "Materials genome initiative for global competitiveness," *National Science and Technology Council OSTP*, 2011.

[3] P. Nguyen *et al.*, "4ceed: Real-time data acquisition and analysis framework for material-related cyber-physical environments," in *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*, 2017, pp. 11–20.

[4] P. Nguyen and K. Nahrstedt, "Monad: Self-adaptive microservice infrastructure for heterogeneous scientific workflows," in *Autonomic Computing (ICAC), 2017 IEEE International Conference on*, 2017, pp. 187–196.

[5] "User study and survey on material-related experiments," http://hdl.handle.net/2142/94738, 2016, [Online; accessed April 30, 2018].

[6] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, 2016, pp. 1–9.

[7] "Bro Network Security Monitor," https://www.bro.org/, accessed: 2018-05-10.

[8] "ELK Stack," https://www.elastic.co/elk-stack, accessed: 2018-05-10.

[9] "Kubernetes," https://kubernetes.io/, accessed: 2018-05-10.

[10] "RabbitMQ," https://www.rabbitmq.com/, accessed: 2018-05-10.

[11] "Apache Zookeeper," https://zookeeper.apache.org/, accessed: 2018-05-10.

[12] "Tensorflow," https://www.tensorflow.org/, accessed: 2018-05-10.

[13] P. Nguyen *et al.*, "Bracelet: Hierarchical edge-cloud microservice infrastructure for scientific instruments lifetime connectivity." in *http://hdl.handle.net/2142/100335*, 2018.