Fabrication-as-a-Service: A Web-based Solution for STEM Education using Internet of Things

Gianluca Cornetta, Abdellah Touhafi, Mohammed Amine Togou and
Gabriel-Miro Muntean, *IEEE Senior Member*

***Abstract*—Recently, Fabrication Laboratories (Fab Lab) have been shown to have a great impact on learners' academic and personal progress. As a result, an increasing effort is being put to integrate Fab Labs into schools' curricula. Yet, owing to the high cost of setting-up and maintaining Fab Labs as well as the lack of sufficient funding for most schools and universities, only a limited number of institutions can afford them. In this paper, we propose a new concept called Fabrication-as-a-Service (FaaS) that uses Internet of Things to democratize access to Fab Labs via enabling a wide learning community to remotely access these computer-controlled tools and equipment over the Internet. It employs a two-tier architecture consisting of a hub, deployed in the cloud, and a network of distributed Fab Labs. Each Fab Lab interacts with the hub and other digital labs via a Fab Lab Gateway. This is to support scalability and high availability of fabrication services as well as ensure system's security. FaaS also adopts an innovative master-slave approach that uses inexpensive external hardware to monitor and control the activity of expensive fabrication equipment. The paper also describes FaaS deployment in the context of the European Union Horizon 2020 NEWTON project. Multiple scenarios have been deployed to fully illustrate the benefits of FaaS architecture and to assess the performance of its communication protocol stack.**

***Index Terms*—Fabrication-as-a-Service, Remote education, Internet of Things, Machine to Machine Communication, Fabrication Labs.**

I. Introduction

A Fab Lab is a small-scale workshop that offers personalized digital fabrication using a set of flexible computer-controlled tools and machines (*e.g*. 3D printers, laser cutters, computer numerically-controlled (CNC) machines, printed circuit board millers and other basic fabrication tools). It endorses a new learning approach that considers technology as a key material to promote concepts such as *learning by doing* and *enjoying while learning* [1], [2].

Numerous studies [3]–[12] have demonstrated the great impact of Fab Lab technologies on students' academic and personal growth. These studies have contributed to the global increase in the number of Fab Labs, estimated at 1186 in 2017. For example, SCOPES-DF [13] is a project developed by FAB Foundation[1] to promote the use of digital fabrication in Science, Technology, Engineering and Mathematics (STEM). It consists of a network of Fab Labs, approximately 1000, located in more than 78 countries. They provide various resources that are used to develop lesson plans aligned with the U.S. national education standards. AuLAB [14] is a Spanish project developed by LABoral[2] Art Center in collaboration with the Ministry of Education of Asturias. It targets primary, secondary and vocational education students. It aims at adapting education to the needs and peculiarities of each educational level. In Ireland, two projects are worth mentioning: Fab Lab Limerick [15] and WeCreate [16]. Both offer cultural and educational programmes for students, designers, crafters and entrepreneurs, bridging the gap between these technologies and creatives from all disciplines.

Yet, owing to the high costs of setting-up and maintaining Fab Labs, most public schools and universities worldwide are unable to afford them. Though, providing Fab Labs with ubiquitous access and implementing resource sharing mechanisms would eventually contribute to overcome the price barrier. This has motivated us to propose Fabrication-as-a-Service (FaaS), an innovative approach that enables remote access to Fab Labs as a Web-based service. It endorses the use of subtle concepts such as Internet of Things (IoT) and Industry 4.0 in the context of Fab Labs. The novelty of FaaS is the fact that it enables various schools and universities to access remotely existing Fab Labs, deployed in different geographical locations, instead of investing in expensive digital fabrication equipment. To the best of the authors' knowledge, there is no work in the literature that has proposed something similar. We believe that FaaS is a necessary evolution of Fab Labs, allowing them to become available to a wider community over the Internet and not to be exclusive to elite schools and universities. We also believe that FaaS can be an integral part of the twenty-first century teaching and learning paradigm as it is dynamic and user-centric.

The rest of this paper is organized as follows. Section II reviews some related work regarding the impact of Fab Lab-based learning on students' performance. Section III describes the proposed FaaS architecture. Section IV presents the communication protocol stack. Section V describes FaaS use cases. Section VI outlines FaaS real-time deployment as part of the NEWTON project and includes the results of FaaS Fab Lab performance testing. Finally, Section VII concludes the paper.

Assoc. Prof. G. Cornetta is with CEU San Pablo University, Madrid, Spain email:gcornetta.eps@ceu.es

Prof. A. Touhafi is with Vrije University, Bruxelles, Belgium e-mail:abdellah.touhafi@vub.ac.be

Dr. M. A. Togou and Assoc. Prof. G-M. Muntean are with the Performance and Engineering Laboratory (PEL) Dublin City University, Dublin, Ireland e-mail: {mohammedamine.togou, gabriel.muntean}@dcu.ie

[1]https://www.fabfoundation.org/

[2]http://www.laboralcentrodearte.org/en

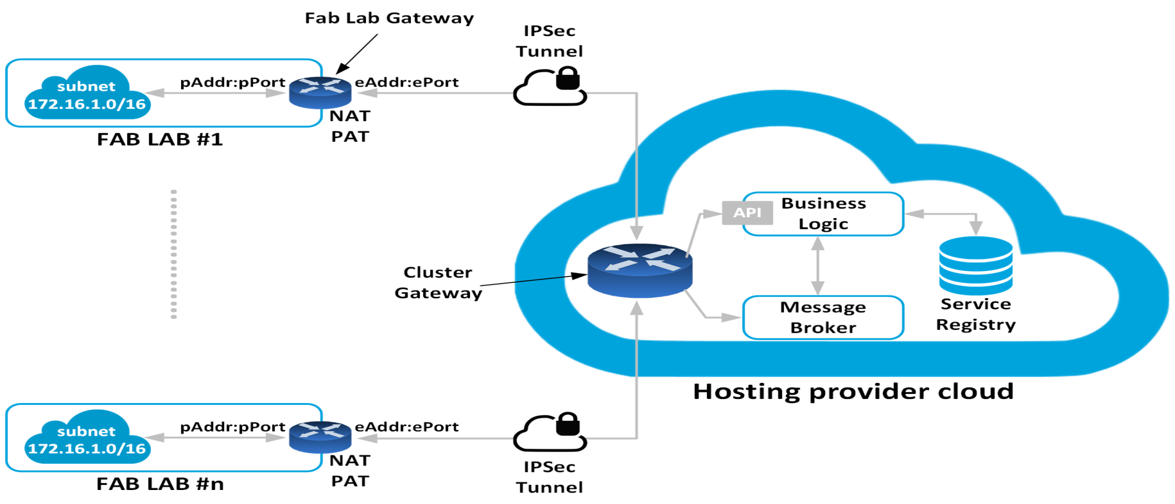


Fig. 1. Fabrication-as-a-Service (FaaS) architecture with cloud inter-networking

II. Related Works

All the research efforts put to date in the digital fabrication area have been aimed at demonstrating the effectiveness of Fab Lab-based learning on students' performance. For instance, Berry *et al*. [3] suggested that teaching Mathematics, Science and Engineering via tasks that make use of digital fabrication can help students learn faster and be more engaged. Tesconi [14] showed that by using digital fabrication, students can develop their critical thinking skills used when dealing with problems and making decisions. Johnson *et al*. [4] highlighted that having Fab Lab-based learning in primary and secondary schools can decrease school absenteeism. It can also improve students' performance in subjects such as Mathematics and Science. Angello *et al*. [5] reported that using Fab Lab-based learning in primary schools can enhance students' experience by adding *fun while learning*. Chu *et al*. [6] also showed that integrating Fab Labs in primary schools' curricula can have a great impact on students' self-efficacy and self-identification. Bang-Hee *et al*. [7] found that incorporating Fab Labs in high school curriculum helps students develop skills such as creativity, problem-solving, collaboration and communication. Harron *et al*. [8] reported that digital fabrication can prepare students, including those with special needs, for their future carriers by increasing their confidence through hands-on accessible activities. Finally, Eversmann [9] reported that the realization of large-scale prototypes in architecture using digital fabrication technologies made university students keener to self-learning.

Yet, to the best of authors' knowledge, no studies could be found in the literature that propose to enhance the Fab Lab functionality by providing support for pervasive and ubiquitous Internet access. This has motivated the authors to propose FaaS, a new fabrication methodology inspired by Industry 4.0 and IoT concepts to provide users with the possibility of remotely accessing the digital fabrication equipment to control fabrication activities. The main goal of FaaS is to ensure a democratic access to these workshops, enabling a wide community to benefit from them.

III. FaaS Architecture

FaaS architecture provides Fab Labs with an abstraction layer that wraps the underlying hardware infrastructure into a programmable interface. This latter consists of a set of Application Programming Interfaces (APIs) that enable third-party applications to access Fab Labs as a Web service. These APIs perform the following functions:

1) Remotely controlling and configuring Fab Lab equipment;
2) Ensuring Inter and Intra-Fab Lab communication along with task synchronization.

Using these APIs, remote monitoring and automatic synchronization of the machines involved in a fabrication batch can be done with minimum human intervention. They also enable the partitioning of complex designs into small pieces that will be dispatched to different networked Fab Labs. Furthermore, they facilitate the implementation of Education-to-Education (Ed-to-Ed) scenarios in which partner institutions can share expensive fabrication equipment. Connecting multiple Fab Labs is very challenging. Indeed, several issues have to be addressed, including system interfacing, scalability, security, quality of service as well as real-time and non-blocking communications. In addition, the possibility to manage collaborative and distributed fabrication batches with little or no human supervision can be a potential cause of equipment's failure. This in turn entails the development of a monitoring software and hardware infrastructure to guarantee safe equipment operation and fault tolerance. In the following subsections, we describe how all of the aforementioned issues are addressed by FaaS.

A. Design Goals and Problem Statement

The major design goals are as follows:

1) Develop a distributed and modular infrastructure to enable interconnection and communication of several Fab Labs spread over a wide geographical area.
2) Create software and hardware wrappers to expose the fabrication machines to the Internet as web services through a set of REST (REpresentational State Transfer) APIs.

3) Design the communication protocol that enables remote access, monitoring, and resource sharing among Fab Labs.

These objectives must be accomplished while:

1) Complying with the security restrictions imposed by the network administrators of the institutions in which the Fab Lab infrastructure is deployed.
2) Limiting the deployment costs by leveraging a minimal cloud infrastructure and inexpensive off-the-shelf micro-controllers to implement machine and Fab Lab wrappers.
3) Limiting the number of vendor-specific cloud services in order to ease the migration to different Infrastructure as a Service (IaaS) providers and to enable the implementation of multi-cloud architectures.
4) Providing software interfaces to easily plug the Fab Lab infrastructure in third-party infrastructure or applications.

The problem of infrastructure costs and scaling has been thoroughly addressed in [17]. A minimal deployment on Amazon AWS public cloud requires (beside virtual machines and storage) just the following services:

1) The DNS (Domain Name System) service to translate logical names into IP addresses;
2) A distributed storage system to implement the blob-store[3].

In Amazon AWS, the DNS service is called *Route53*, whereas the distribute storage system is called *S3* (Simple Storage Service). Thus, the Fab Lab Cloud Hub application has no vendor lock-ins since it does not directly rely on specific services managed by the cloud provider and can be easily migrated to any private or public cloud.

Because of its significant benefits compared to a point-to-point interconnection, the spoke-hub architecture, illustrated in Fig. 1, was adopted to interconnect the distributed Fab Labs. In a point-to-point architecture, Fab Labs could directly communicate with each other. This means that the communication overhead and costs increase exponentially according to $\binom{n}{2} = \frac{n(n-1)}{2}$, where n is the number of interconnected Fab Labs. Thus, 4 interconnected Fab Labs would require 6 connections, 8 Fab Labs would require 28 connections, and so on. Conversely, a spoke-hub model scales better while maintaining a consistent architecture and is more affordable in long term.

Moreover, the software architecture is organized as a set of loosely-coupled microservices accessible through a set of REST APIs. Each microservice can run and scale independently; this makes the implementation of the use-case scenarios very easy. Given that our design is cost-constrained, we tried to find the right balance between affordable costs and acceptable performance. To date, a simulator of our cloud infrastructure [18] was implemented to investigate the cost-performance trade-offs and to identify potential system bottlenecks as the infrastructure scales.

[3]A blobstore is a data store for large files in BLOB (Binary Large OBject) format. In the context of FaaS architecture, it is used to keep backups of the cluster deployed on cloud premises.

B. Cloud-based Architecture

The overall FaaS architecture is illustrated in Fig. 1. FaaS employs a two-tier architecture consisting of a hub and a network of distributed Fab Labs. Each Fab Lab interacts with the hub, deployed in the cloud, and other labs via a Fab Lab Gateway. Instead of the devices processing the data themselves as in [19], we opted for a standard approach that processes all the data in the cloud. This is because the amount of data to be processed (i.e., the Fab Lab status) is not large enough to justify the deployment of a middleware layer between Fab Labs and the Cloud hub. The Service Oriented Architecture (SOA) guarantees the inter-operability of the different system components, regardless of the used implementation technology, while supporting a smooth system scalability. The service registry acts as a centralized communication hub among service providers and subscribers.

FaaS enables two types of communication: *Inter-Fab Lab communications* and *Intra-Fab Lab communications*. The former is managed by the centralized broker on the cloud premises and denotes the outbound traffic of the Fab Lab network. The latter is managed by the Fab Lab Gateway in the Fab Lab VPN and designates the local network inbound traffic. This architecture reduces the load on the centralized broker considerably, whose task is simply relying short high-level commands from the source to the destination gateway. The gateway acts as a relay for the Fab Lab inbound traffic, routing the incoming command to the target machine according to specific policies (*e.g.* machine availability, type and complexity of the fabrication batch, etc.).

Each Fab Lab has at least one public IP address *eAddr:ePort* and can only be accessed through a Fab Lab Gateway. This latter maps the inbound traffic into a private address *pAddr:pPort* by means of a Network Address Table (NAT) and a Port Address Table (PAT). The Fab Lab Gateway performs the same task on the outbound traffic The message flow between the cloud application and the networked Fab Labs is managed by a cloud-deployed Message Queue Telemetry Transport (MQTT) broker [20]. The employed publish-subscribe messaging protocol requires the message broker to distribute messages to only clients who are subscribed to a certain topic.

C. IoT Wrapper and Command Interface

FaaS employs an innovative master-slave approach (*i.e.* see Fig. 2) that uses inexpensive external hardware to monitor and control the activity of expensive fabrication equipment. The *master unit* (MU) is basically an off-the-shelf micro-controller unit (MCU) with basic Ethernet and wireless connectivity along with a USB and a General Purpose Input Output (GPIO) port. The USB port is used to communicate with the digital fabrication machine (*slave*) while the wireless interface is used to communicate with other Fab Lab networked equipment. The status of the fabrication machine (*i.e.* switched-off, idle or busy) is monitored by a non-invasive current transformer (CT) sensor that measures the AC current drawn by the equipment. To do so, the CT sensor is linked to the GPIO digital interface via a high-resolution analog-to-digital converter (ADC) and a simple signal conditioning circuit.

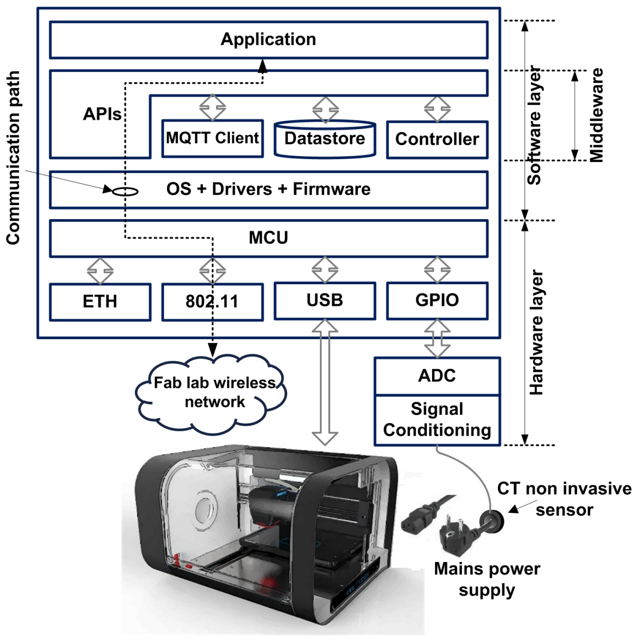


Fig. 2. Fabrication equipment interfacing

The MU is responsible for decoding and translating an incoming message into a set of commands comprehensible to the slave machine. These operations are performed by the set of software modules depicted in Fig. 3. While module communications rely on asynchronous event-driven callbacks that are managed by an event loop, commands dispatched to the *slave* machine are handled synchronously. Command execution is supervised by a controller, whose state diagram is presented in Fig. 3. In each state, an event is generated. The generated events are captured by other software modules, triggering the execution of specific callback functions. For instance, the command Interpreter integrates an event listener that captures the *new_command* event, emitted by the command Queue, and the *status* event, emitted by the Controller. It then schedules the execution of several event-dependent callbacks, as illustrated in the following pseudo-code snippet:

```
var queue = require ('queue');
var controller = require ('controller');
var cq = queue.connect;
var ctrl = controller.connect;
cq.on('new_command', function decode(){
     //on new_command event start decoding
     });
ctrl.on('error', function abort(){
     //on error event abort command
     });
...
```

In case an *error* event is emitted by the Controller, a callback that aborts decoding is executed. When a new command is executed, the Controller starts monitoring the current drawn by the equipment, building a current pattern for every executed command and saving it to the local data store.

D. Scalability

To support scalability and high availability of the fabrication services, FaaS deploys a two-level parent-child broker architecture (see Fig. 6), individual command pooling and a cluster of brokers. The parent (on the cloud server) and the child broker (on the Fab Lab Gateway) share the responsibility of routing the end-to-end traffic between clients and remote machines. This is beneficial to the cloud server as the number of publishing nodes will be limited to Fab Lab Gateways only. Moreover, the use of a pool of gateway-specific message queues instead of a unique centralized queue allows for the deployment of smart message passing protocols. These protocols are capable of routing traffic towards specific subscribers only (*e.g.* according to geographic proximity) instead of continuously broadcasting incoming messages to all subscribers. Finally, clustering the parent broker, by replicating the number of broker instances, will provide high availability and better fault tolerance.

E. Security

The proposed architecture in Fig. 1 implements a spoke-hub set-up in which the networked Fab Labs communicate through a centralized hub on the cloud premises. In this scenario, the security concerns are related to:

1) Fab Lab to Hub communications.
2) Intra-Fab lab (i.e. machine to gateway) communications

Therefore, security is reinforced at three different levels:

1) *Network level*: by encapsulating brokers and clients into Virtual Private Network (VPNs) to provide a trustworthy connectivity.
2) *Transport level*: by deploying encryption-based protocols such as Secure Socket Layer (SSL) or Transport Layer Security (TLS) to provide confidentiality.
3) *Application level*: by employing the MQTT protocol that provides a client identifier and credentials which can be used to authenticate devices. These properties can be used by the broker to define authorization levels for each connected device/application.

Note that we address secure Fab Lab to Hub communications at the network level. This is the standard approach for applications where, like in our case, a gateway is connected to fabrication machines on one side and to the cloud communication hub on the other. Indeed, the Spoke and the hub nodes form a Virtual Private Network (VPN) in which the Fab Lab gateway and the virtual machine instances, on cloud premises, communicate securely over the Internet using private IP addresses over an IPSec (IP Secure) tunnel. Within a Fab Lab, security can be enforced both at transport (through SSL/TLS transport encryption to ensure confidentiality) and application level (through device credentials that allow to implement authorization and authentication policies). However, since the Fab Lab network is a private and trusted network accessible only through the VPN tunnel, implementing these security levels is not strictly necessary. It is noteworthy to mention that since the transport level encryption may add a significant overhead, payload encryption at the application level can be a valid alternative.

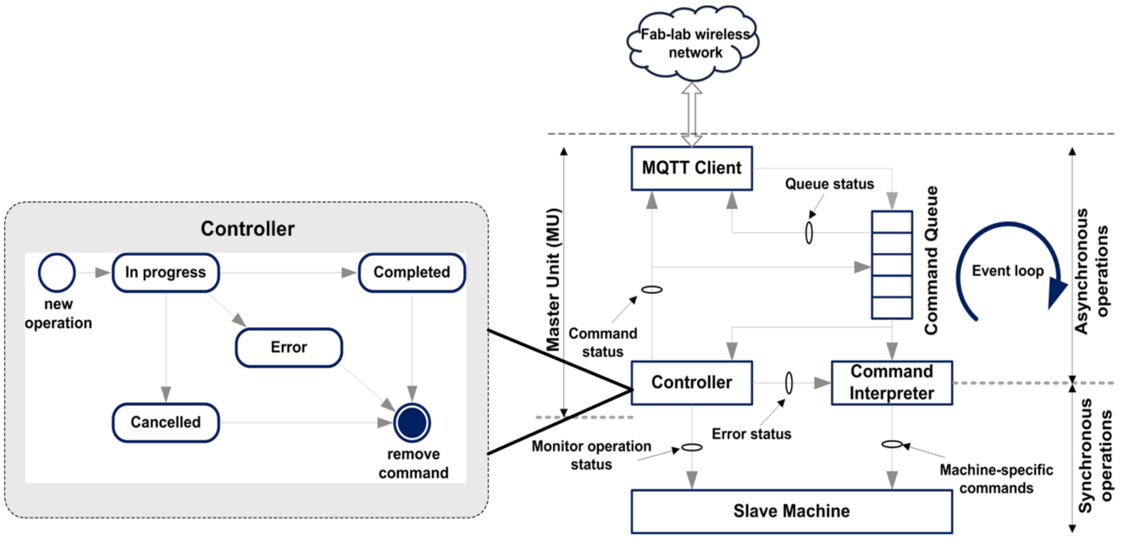
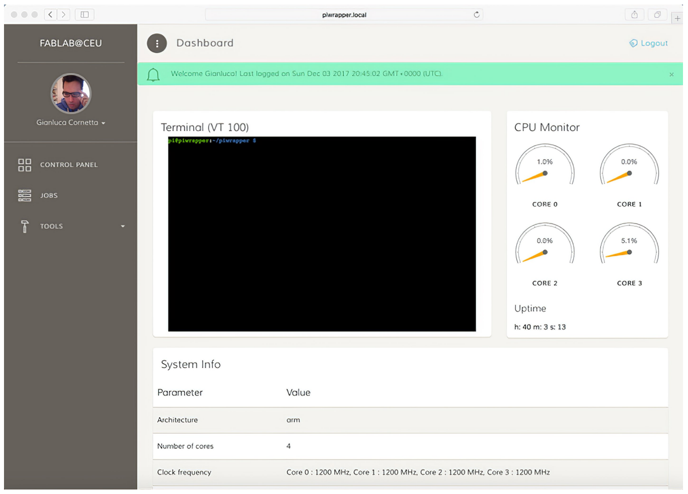


Fig. 3. Machine wrapper: software architecture and machine state controller



Welcome Gianluca! Last logged on Sun Dec 03 2017 20:45:02 GMT+0000 (UTC).

Terminal (VT 100)

```
pi@piwrapper:~/piwrapper $
```

CPU Monitor

1.0% CORE 0	0.0% CORE 1
0.0% CORE 2	5.1% CORE 3

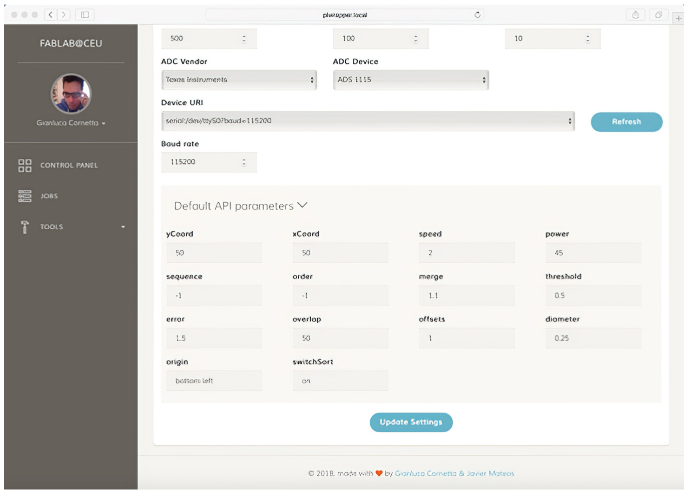
Uptime

h: 40 m: 3 s: 13

System Info

Parameter	Value
Architecture	arm
Number of cores	4
Clock frequency	Core 0 : 1200 MHz, Core 1 : 1200 MHz, Core 2 : 1200 MHz, Core 3 : 1200 MHz

(a)



500	100	10

ADC Vendor: Texas Instruments
ADC Device: ADS 1115
Device URI: serial:/dev/ttyS0?baud=115200
Baud rate: 115200

Default API parameters

yCoord	xCoord	speed	power
50	50	2	45
sequence	order	merge	threshold
-1	-1	1.1	0.5
error	overlap	offsets	diameter
1.5	50	1	0.25
origin	switchSort		
bottom left	on		

© 2018, made with ❤ by Gianluca Cornetta & Javier Mateos

(b)

Fig. 4. Machine wrapper web interface: (a) dashboard and b) machine parameters panel

F. User Interfaces

As already stated, the system software architecture relies on a loosely-coupled set of microservices that implement the machine wrapper, Fab Lab gateway and cloud hub routing logic. Each microservice provides a set of APIs which are fully compliant with Swagger 2.0 (now OpenAPI) specifications. The Swagger UI has been integrated into the microservices to provide end users with web interfaces to test the APIs and program the system. In addition, the Fab Lab Gateway and the machine wrapper also provide additional user interfaces. Indeed, the Fab Lab Gateway has a command line interface that allows gateway configuration. This includes setting up the API rate limiting policies, managing the certificates, configuring alarms and accessing the geolocation API. The machine wrapper has a web administration interface that provides full access and control of the underlying hardware, as shown in Fig. 4.

The dashboard provides an interface to directly access the machine wrapper as depicted in Fig. 4(a). It has three main panels: a fully-featured VT-100 terminal that allows access to all resources of the machine wrapper, a panel with real-time gauge charts that show the CPU load of each individual processor core as well as the server up-time, and a panel with the general wrapper system information (i.e. architecture, operating system, number of cores, core frequency, etc.). These panels enable editing machine configuration and settings, including machine logical name, type and vendor. Moreover, it is also possible to set up the threshold current to indicate that the machine is working/busy and to configure the ADC device connected to the machine wrapper GPIO port. Indeed, the user can select a specific ADC device (only Texas Instruments ADS1x15 devices are supported so far), program its sampling rate, and define the time window of a software low-pass filter that averages the measured samples to provide some noise immunity. The user can also configure the baud rate of the communication channel between the machine wrapper and the connected digital fabrication machine. Finally, this panel also allows the possibility to edit the default fabrication parameters of the connected machine as illustrated in Fig. 4(b).

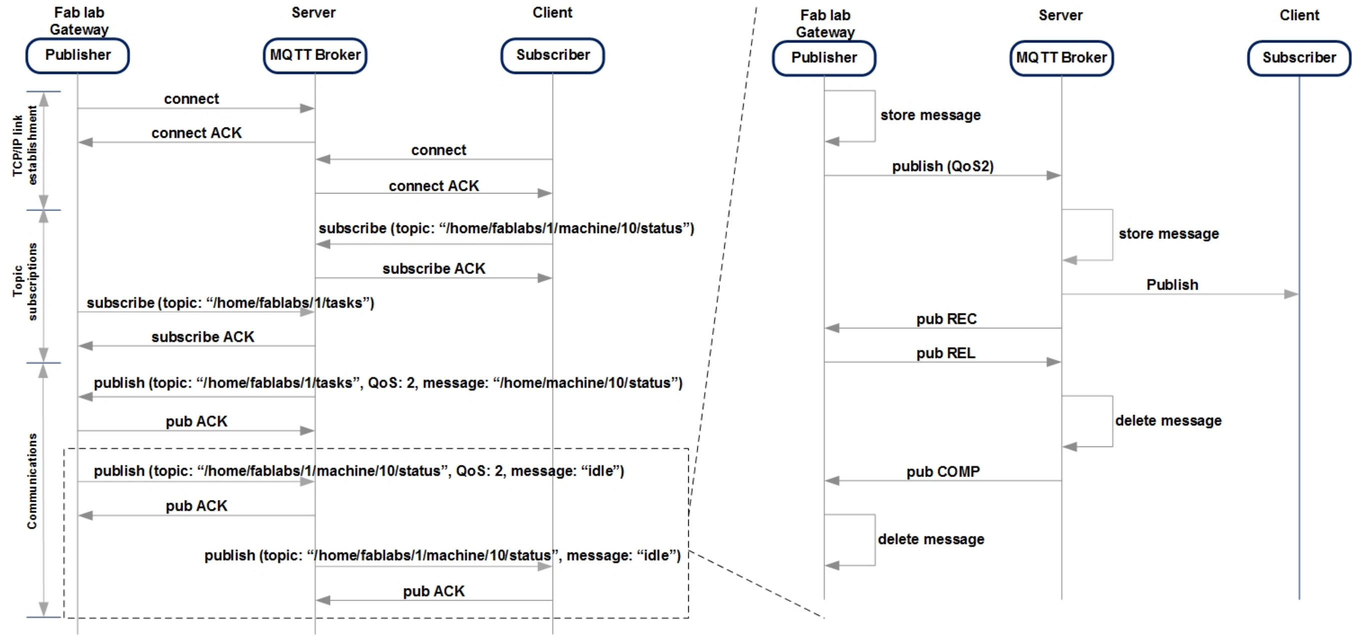

Fab lab Gateway
Publisher
Server
MQTT Broker
Client
Subscriber
TCP/IP link establishment
Topic subscriptions
Communications
connect
connect ACK
connect
connect ACK
subscribe (topic: "/home/fablabs/1/machine/10/status")
subscribe ACK
subscribe (topic: "/home/fablabs/1/tasks")
subscribe ACK
publish (topic: "/home/fablabs/1/tasks", QoS: 2, message: "/home/machine/10/status")
pub ACK
publish (topic: "/home/fablabs/1/machine/10/status", QoS: 2, message: "idle")
pub ACK
publish (topic: "/home/fablabs/1/machine/10/status", message: "idle")
pub ACK
store message
publish (QoS2)
store message
Publish
pub REC
pub REL
delete message
pub COMP
delete message

Fig. 5. Simplified sequence diagram and detailed message passing to support MQTT QoS 2 communications

Last but not least important, the dashboard also provides an interface to launch a local fabrication job on the selected machine. The interface allows uploading a design file on the selected machine and configure it via defining the material used, the material-dependent fabrication parameters, and by configuring the initial position of the machine head. A more detailed review of all the provided features and functionalities is available on the NEWTON Fab Lab page hosted on GitHub (https://gcornetta.github.io/gwWrapper/).

IV. FaaS Communication Protocol Stack

Fig. 5 depicts a simplified timing diagram that describes the communication between client applications and remote Fab Labs. It relies on a protocol stack that includes the MQTT protocol. The message exchange has four stages: link establishment, topic subscription, communication and disconnection (not illustrated for the sake of simplicity). Once the TCP/IP connections (*i.e.* between the client and the broker on one side, and between the Fab Lab Gateway and the broker on the other side) are established, both end nodes subscribe to topics they are interested in. For example, the client in Fig. 5 is interested in getting the status of machine number 10 of Fab Lab number 1 while the Fab Lab Gateway is interested in receiving commands from the broker regarding Fab Lab number 1. Once subscribed, end nodes start exchanging messages. The broker first sends a command to the Fab Lab Gateway asking for the status of machine number 10. The Gateway pings the machine and forwards the status information to the broker which in turn forwards it to the client.

To guarantee successful command delivery to end nodes, we make use of the highest QoS level supported by MQTT, namely QoS2. The QoS2-compliant messaging triggers on both the publisher's and broker's sides a number of operations, as illustrated in Fig. 5. More specifically, the broker acknowledges the receipt of a QoS2 Publish with a *Pub REC (receive)* message, stores a reference to the packet identifier and publishes the incoming message. When the sender receives the *Pub REC*, it discards the initial publish request and responds with *Pub REL (release)*. When the broker receives the *Pub REL*, it discards the stored reference and sends *Pub COMP (complete)*. Note that the level of the required QoS is specified in the message body exchanged by the communicating nodes.

Fig. 6 illustrates the two-level hierarchical architecture deployed to support inter- and intra-Fab Lab communications. Labels (1) to (9) identify the sequence of operations that take place from the moment a remote client issues a fabrication machine status request until receiving a reply from the system. The dual broker architecture decouples the remote client from the Fab Lab, adding an extra layer of security. Broker-to-broker message forwarding is managed by the MQTT bridge running on the cloud server.

Note that FaaS Fab Lab instantiation relies on a reliable communication stack (*e.g.* TCP, SSL), which is highly suitable for most fabrication services. This is because the remote fabrication time is anyway several order of magnitude longer than the time required for network communication. However, there exists some control services in industrial environments which require extremely fast reactions. In this case, alternative communication approaches should be considered. For instance, non-reliable transport layer protocols can be used for transmission. Another approach would be to replace SSL with payload encryption, as indicated in Section II.E.

V. FaaS Principle and Use Cases

FaaS is a new fabrication methodology which is inspired by the basic concepts found in the Industry 4.0 philosophy [21]. In Industry 4.0, machines are inter-connected, have the ability to co-operate and provide service interfaces to third

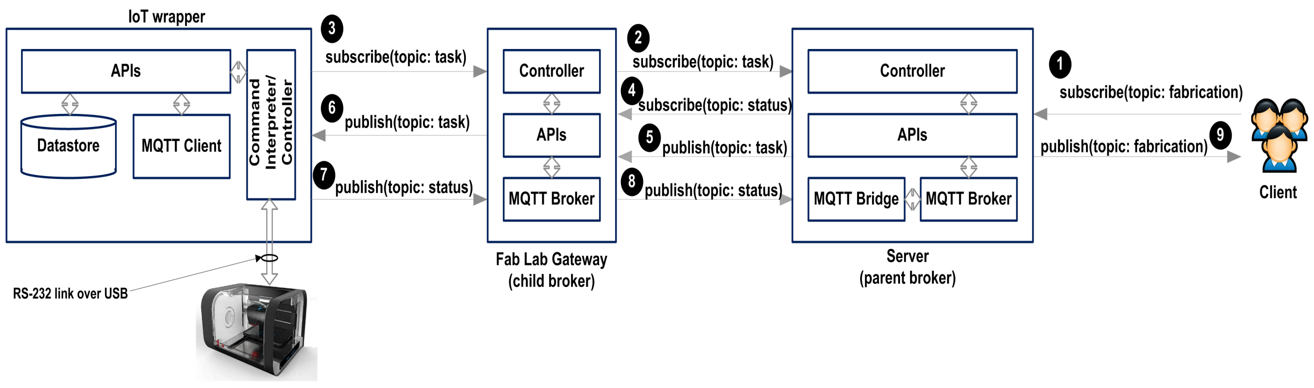
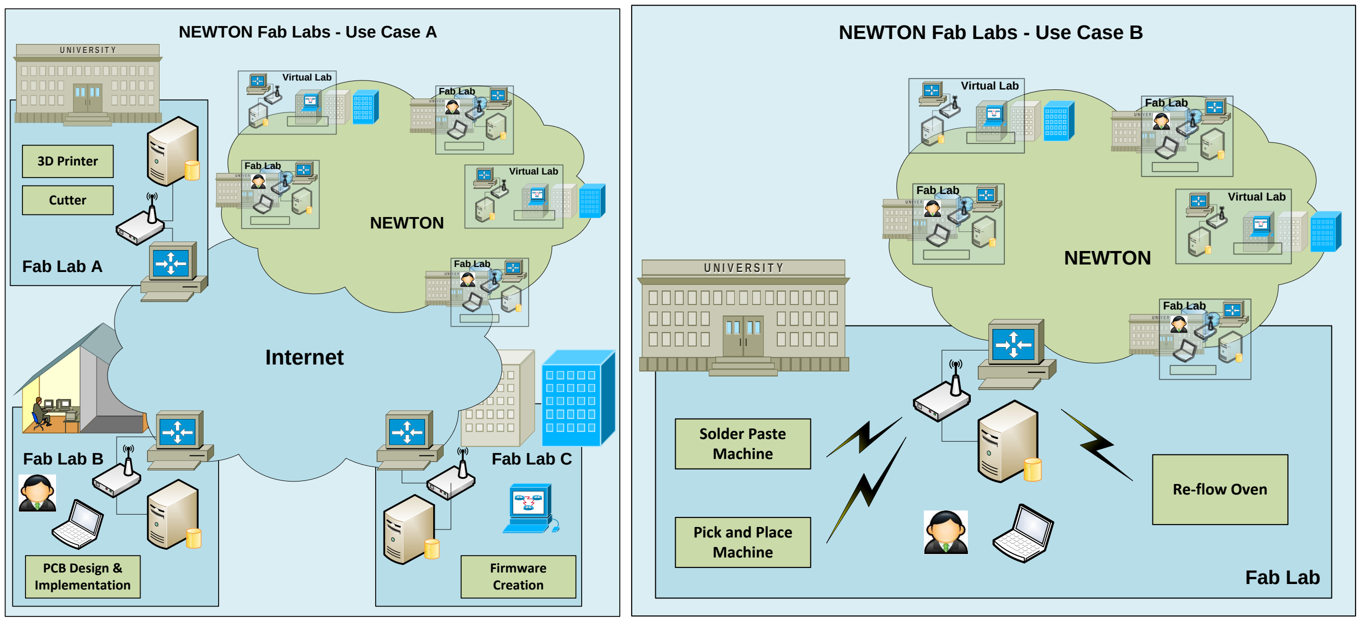


Fig. 6. Overview of the inter and intra-Fab Lab messaging flow



(a) (b)

Fig. 7. NEWTON FaaS Fab Lab use cases: (a) remote co-operation, and (b) assembly line sharing

parties. FaaS focuses on improving the fabrication efficiency and flexibility via integrating the fabrication process within an Internet of services framework. However, the main goal is improving the use of Fab Labs in an educational context. This is to facilitate machine and knowledge-sharing as well as reinforcing teacher-student cooperation.

In order to illustrate the benefits of the FaaS approach, two use cases in the context of the NEWTON Fab Labs are discussed next. The first use case describes the use of FaaS within a remote educational context, where several groups of students along with their teachers collaborate on the same project. In the second use case, FaaS is used to share a small electronics assembly line for surface mounting devices between two labs.

A. *FaaS for Remote Cooperation*

Fig 7(a) depicts the first use case where three teams at three different locations will collaborate in the context of a common project. The three teams have remote access to NEWTON Fab Labs and should fabricate a robot which has the ability to climb stairs. The mechanical part of the robot has to be built using basic plate material which can be handled by a laser cutter and basic structures which can be implemented using a 3D printer. The electric driver part consists of small stepper motors while the electronic steering part is implemented using an Arduino platform and a motor-driver. The electronic parts and the motor driver are further integrated into a single Printed Circuit Board (PCB). To this end, team A will design the mechanical parts, team B will focus on the implementation of the printed circuit boards for electronics integration while team C will focus on the firmware creation.

The Fab Lab setup consists of a laser cutter machine, a 3D printer, a programmer for the Arduino boards and a PCB-milling machine. All the machines and the Arduino programmer are locally inter-connected with the local gateway and each machine registers its fabrication services and status-information at the cloud server. The status-information of each registered machine informs the user about the availability of the machine, the quality of service (QoS) which can be delivered and the required fabrication time. The QoS-indicator is a tuple reflecting the production quality and production reliability. Those two values are estimated by means of a Naive Bayesian Estimator which uses measurable system parameters like machine quality, production throughput, production latency, network reliability and sensor reliability.

During the design and prototyping phases, each team registers their design and sends it for fabrication. FaaS will distribute the fabrication tasks to the machines at the three different locations such that each team has similar fabrication outputs within a given time-frame (as if they were at the same premises). An important aspect to deal with is the potential differences in fabrication quality between the various Fab Labs, mainly due to differences in machines. Based on the received QoS value, the sender of the fabrication task and the receiver of the fabrication output will be able to predict the similarity between the original design and the locally fabricated copy.

B. NEWTON FaaS Fab Lab for Assembly Line Sharing

Fig 7(b) depicts the second use case. It demonstrates the capability of electronic assembly line virtualization within a single Fab Lab that does not avail from the inter-connected Fab Lab capabilities. Such an assembly-line consists typically of a Solder Paste Machine, a Pick and Place Machine and a Reflow Oven. In an industrial setup, the machines are physically interconnected by means of a conveyor, detectors and a control-system to automate the transfer of the printed circuit board from one machine to another. In a Fab Lab setup, we have experimented with off-the-shelf low-cost table-top machines to create a similar semi-automatic electronic-assembly line. Yet, these low-cost alternatives cannot be easily integrated as a full automatic assembly line. This implies that some manual interactions are still required. Consequently, these machines should be handled individually, implying that users are required to have a good understanding of the assembly line setup along with the subsequent fabrication steps.

With FaaS, it is possible to address this issue by representing the electronic assembly line as a single fabrication task. The server will redistribute the single assembly line task to a set of subsequent fabrication labs. This is done while taking into account the availability of the different machines as well as the availability of the Fab Lab technicians.

C. Multi Tenancy and Queuing Aspects

The multi-tenancy aspects of the system are taken care of by message queuing and task distribution mechanisms. Indeed, each machine has a separate (asynchronous) priority based task-queue and a generated user-wait list. Each user receives a dedicated time-slot in which the spawned tasks are executed. It is only during that time slot that the user also receives the possibility for remote-access to the machine. For the use-cases described, RabbitMQ has been used as message broker. By inducing specific rules, a dedicated priority mechanism has been implemented to ensure fairness without mitigating the safety aspect.

As such, each task is provided with the following meta-data: task-type indicator, user-type indicator and the machine-time budget. Based on these indicators, sub-priorities are calculated as follows:

1) User Priority (UP) is defined based on the type of users. There are three types of users:
   a) Fab Lab coordinator (UP=3)
   b) Fab Lab technician (UP=2)
   c) Fab Lab user (UP=1)
2) Task Type Priority (TTP) is given based on the type of the task. There are three types of tasks:
   a) Emergency (TTP=3)
   b) Cancellation (TTP=2)
   c) Normal Tasks (TTP=1)
3) Machine Time Priority (MTP) is calculated based on the daily-machine-time-budget. In fact, each user has a certain daily time budget to use the machine. The priority given for this indicator depends on the remaining time budget of the user minus the time for the new task. Note that this indicator can create a negative priority if the remaining budget is lower than the task-time. The MTP is calculated as follows:

$$MTP = \frac{(TotalUserBudget - TaskTime) * 3}{TotalUserBudget} \tag{1}$$

The total priority of each task is the sum of the three sub-priorities, with a maximum value of 9.

$$TotalPriority = UP + TTP + MTP \tag{2}$$

This method gives high priority to smaller tasks over longer ones. It also attributes high priority to users with high daily machine-time budget over users with low daily machine-time budget. This way, a reasonably fair task scheduling between the multiple users is embedded in the system architecture without compromising the safety aspect (*i.e.* due to the scheduling procedure). Note that tasks with equal priority are processed on a first come first serve basis.

VI. FaaS Real Life Deployment

FaaS real life deployment was performed as part of the EU-funded Horizon 2020 NEWTON project [22]. The NEWTON project has designed and deployed innovative technology-enhanced learning technologies, including support for virtual and Fab Labs [23], adaptive and personalised rich media delivery [24], gamification [25] and game-based learning [26]. The Fab Lab prototype has been deployed at CEU Madrid, Spain and tested remotely from both Ireland [27] and Spain [28]. This deployment has helped gain significant insights on several design and implementation aspects, including hardware design and interfacing, system monitoring and cloud deployment, data security as well as service deployment and orchestration.

In order to assess FaaS performance, a modern load testing framework called *Locust* was used to test the system infrastructure along with the various APIs. It allows to simulate users' behavior using Python scripts. Three scenarios were designed to stress all the Fab Lab APIs. They involve a variable number of concurrent users (50, 100 and 150) with

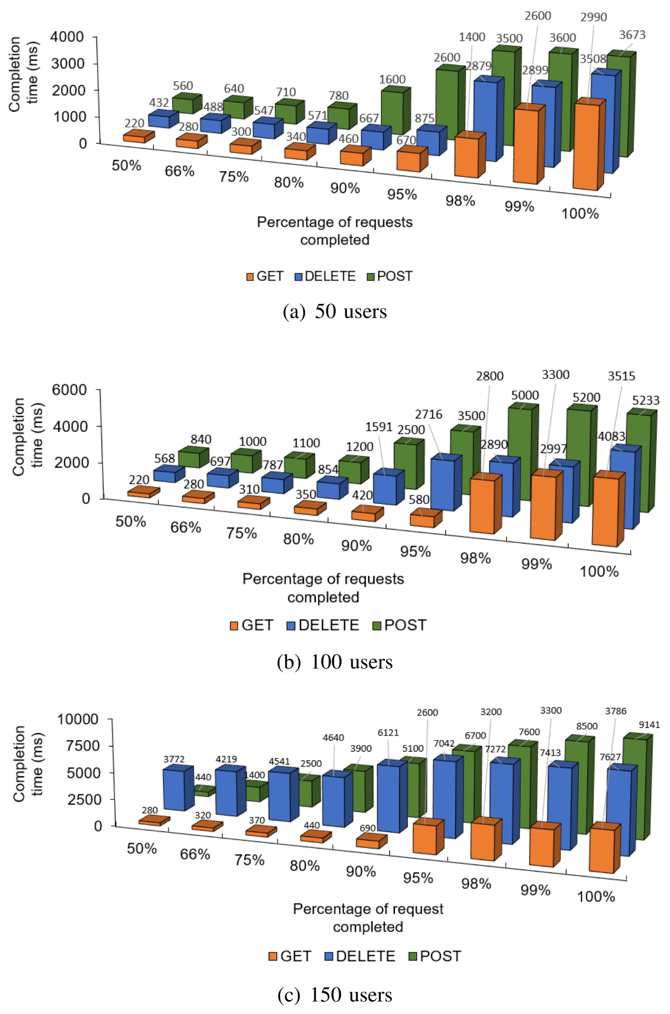


Fig. 8. Completion time with respect to the percentage of the various completed requests

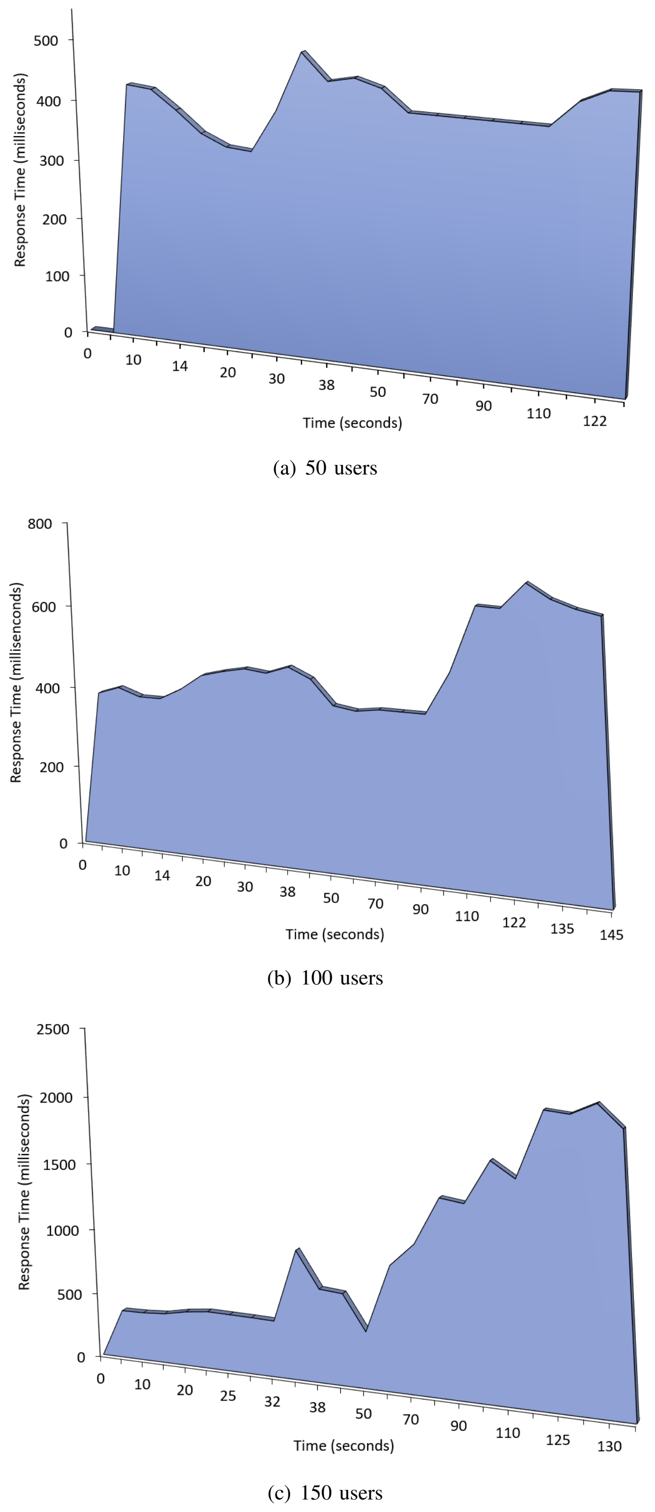
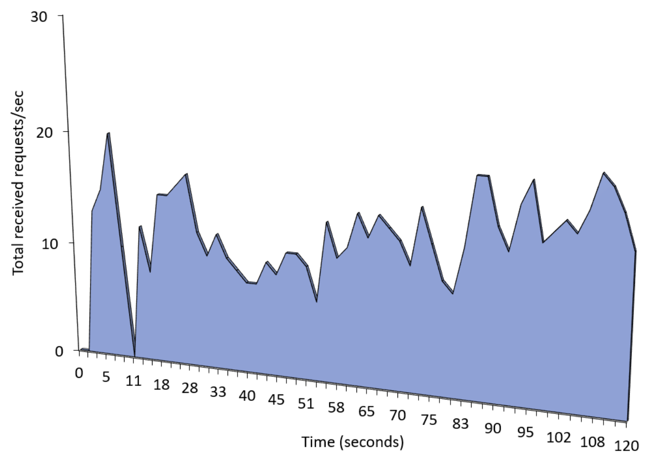


Fig. 9. Response time of the three scenarios in normal operating conditions

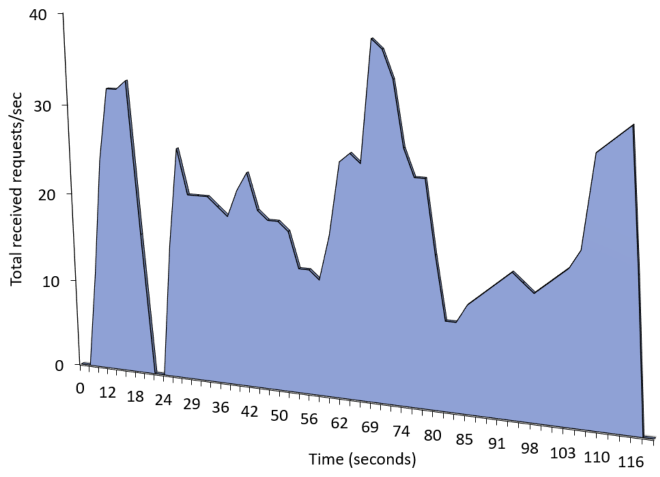
a fixed hatch rate of 5 users/sec. Each scenario is run for a duration of 2 minutes during which users perform multiple operations, including GET the available Fab Lab status, POST a job to the available Fab Lab, GET the status information of the submitted job, DELETE the submitted job, and GET the information of the jobs running in the available Fab Lab.

Fig. 8 depicts the percentage of requests completed in a given time interval for the three scenarios. We observe that the completion time increases with the increase in the number of concurrent users. For instance, 90% of the incoming requests are served on average in less than 910 ms in the 50-users scenario (Fig. 8(a)) and 3970 ms in the 150-user scenario (Fig. 8(c)). It is also observed that *POST* requests incur longer service time (*e.g.* 9141 ms in the case of 150 concurrent users). This is because POST requests are time-consuming as they involve different steps: uploading the fabrication job on the cloud hub; sending it to the Fab Lab Gateway that will deliver it to the target fabrication machine; and finally, updating the jobs queue in the fabrication machine. Still, these results are excellent considering that the Fab Lab infrastructure has been deployed on inexpensive Raspberry Pi III boards.

Fig. 9 illustrates the response time of the three scenarios in normal operating conditions as a function of time. We note that the response time increases with the increase in the number of concurrent users. For instance, the average response time, obtained by integrating the area under the curve, is 443 ms in the 50-users scenario (Fig. 9(a)) and reaches 1696 ms in the 150-users scenario (Fig. 9(c). The measured response time is the average of all the GET, POST and DELETE requests and is strongly biased by the POST and DELETE operation response times. Indeed, the delay of a POST request depends on the size

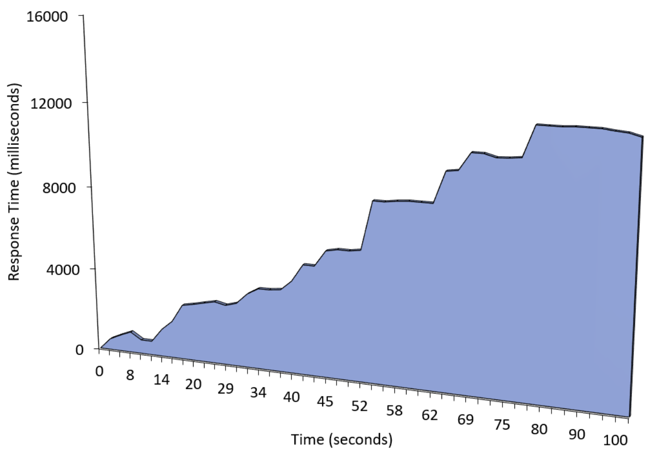


(a) 250 users

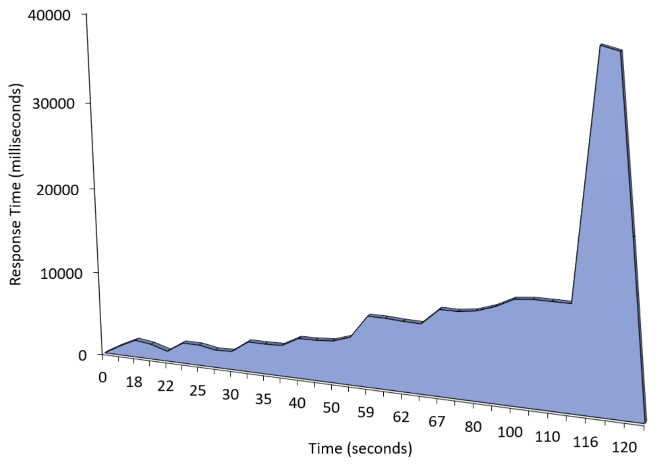


(b) 500 users

Fig. 10. Total requests per second of the two scenarios in peak load conditions



(a) 250 users



(b) 500 users

Fig. 11. Response time of the two scenarios in peak load conditions

of the image to be uploaded on the server and varies from 3693 ms (50-users scenario) to 9141 ms (150-users scenario) while the delay of a DELETE request fluctuates between 1119 ms (50-users scenario) and 4986 ms (150-users scenario).

We also examine the performance of the system infrastructure in peak load conditions. To this end, two scenarios have been used with different number of users (250 and 500). The maximum number of concurrent users is spawned in only 10 seconds and all the users are posting their jobs to the same machine. Again, each scenario is run for a duration of 2 minutes during which users perform multiple operations as in the previous scenarios.

Fig. 10 shows the total requests per second as a function of time for the two scenarios in peak load conditions. We observe that there is a peak of 20 requests/sec and 40 requests/sec for the 250-users and 500-users scenarios, respectively. The average number of requests per second is the integral of the curves depicted in Fig. 10(a) and Fig. 10(b) which equals to 11 and 13, respectively.

Fig. 11 depicts the response time of both scenarios as a function of time in peak load conditions. We observe that the response time increases with the increase in the number of concurrent users. For instance, the average response time is 3900 ms in the 250-users scenario (Fig. 11(a)) and reaches 11604 ms in the 500-users scenario (Fig. 11(b). Recall that the measured response time is the average time required by all requests (*i.e.* GET, POST and DELETE) and is strongly biased by the DELETE and POST response times. Indeed, the DELETE request carries out the following operations: removing the job from the machine's queue; removing the job reference from the machine database and signaling the success of the operation to the Fab Lab Gateway; removing the job reference from the Fab Lab Gateway and signaling the success of the operation to the cloud registry server; and finally, removing the job reference from the Registry server and notifying the end-user. The POST request, on the other hand, carries out the following operations: uploading the image on the cloud hub; sending the image to the Fab Lab Gateway, sending the image to the target fabrication machine; and finally, updating the jobs queue in the fabrication machine. Note that the message drop rate was 2% for the 250-users scenario and 9% for the 500-users scenario.

To further assess FaaS performance, the two use cases described in Section IV are considered. The small fabrication line for electronic assembly was used to discover the typical difficulties that can arise when a small series of prototypes, typically 100 units, are to be manufactured while the manufacturing of the stair climbing robot was employed to discover the typical requirements for remote collaboration and design sharing. The following figures of merit have been examined:

1) responsiveness of different message brokers;
2) QoS-oriented service composition;
3) multi-parameter machine resource scheduling;
4) state detection and anomaly detection.

Table I summarizes the main preliminary findings. We observe that the processing time increases linearly with the increase in the number of requests for services with and without QoS estimator. We also observe that in normal conditions, FaaS can correctly estimate the state of the machine as well as detect anomalies in 99% of the cases. Under high loads (*i.e.* an occupancy exceeding 70% of the available infrastructure), the rate of detecting anomalies remains at 99% while the rate of correctly estimating the state of a machine drops to 90%. Again, these results are outstanding considering the low-cost Raspberry Pi boards deployed.

VII. Conclusions

This paper introduces a novel concept: Fabrication as a Service (FaaS). FaaS enhances existing Fab Lab capabilities by providing the digital fabrication equipment with the possibility to communicate over the Internet in order to remotely control fabrication activities. Using this approach, the fabrication facilities are exposed to the Internet as software services, which may be consumed by third-party applications. The paper also describes FaaS deployment in the context of EU Horizon 2020 NEWTON Project. Testing results have demonstrated FaaS high potential in terms of performance and accuracy. We foresee that FaaS-based Fab Labs will have a huge impact not only on education, but also on industry, helping to develop new business models in which fab-less companies may schedule medium or large-scale fabrication batches hiring third-party remote fabrication services.

Acknowledgments

The work described in this paper is part of the NEWTON project, which has been funded by the European Union under the Horizon 2020 Research and Innovation Programme with Grant Agreement no. 688503. Additionally, the support of Science Foundation Ireland (SFI) Research Centres Programme Grant Numbers 12/RC/2289 (Insight Centre for Data Analytics) and 16/SP/3804 (ENABLE) is gratefully acknowledged.

References

[1] N. Gershenfeld, "How to Make Almost Anything: The Digital Fabrication Revolution Essay," *Foreign Affairs*, vol. 91, pp. 43–57, 2012.

[2] P. Blikstein, *Digital Fabrication and 'Making' in Education: The Democratization of Invention In FabLabs*. Bielefeld: Transcript Publishers, 2013.

[3] R. Berry, G. Bull, C. Browning, C. Thomas, K. Starkweather, and J. Aylor, "Preliminary Considerations Regarding Use of Digital Fabrication to Incorporate Engineering Design Principles in Elementary Mathematics Education," *Contemporary Issues in Technology and Teacher Education*, vol. 10, no. 2, pp. 167–172, 2010.

[4] L. Johnson, A. S. Becker, V. Estrada, and A. Freeman, "NMC Horizon Report: 2015 K-12 Edition," The New Media Consortium, Austin, Texas, Tech. Rep., 2015.

[5] S. L. Chu, G. Angello, M. Saenz, and F. Quek, "Fun in Making: Understanding the Experience of Fun and Learning Through Curriculum-Based Making in the Elementary School Classroom," *Entertainment Computing*, vol. 18, pp. 31–40, 2017.

[6] S. L. Chu, R. Schlegel, F. Quek, A. Christy, and K. Chen, "'I Make, Therefore I Am': The Effects of Curriculum-Aligned Making on Children's Self-Identity," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI'17, 2017, pp. 109–120.

[7] K. Bang-Hee, J. Moon-Yong, and K. Jinsoo, "Development and Application of 3D-Hologram Maker Education Materials for High School Students in Korea," *Advanced Science Letters*, vol. 24, no. 3, pp. 2114 – 2117, 2018.

[8] J. R. Harron and J. E. Hughes, "Spacemakers: A Leadership Perspective on Curriculum and the Purpose of K–12 Educational Makerspaces," *Journal of Research on Technology in Education*, vol. 50, no. 3, pp. 253–270, 2018.

[9] P. Eversmann, "Digital Fabrication in Education - Strategies and Concepts for Large-Scale Projects," in *Proceedings of the 35th eCAADe Conference*, vol. 1, 2017, pp. 333–342.

[10] T. Martin, S. Brasiel, D. Graham, S. Smith, K. Gurko, and D. A. Fields, "FabLab Professional Development: Changes in Teacher and Student STEM Content Knowledge," in *Proceedings of FabLearn Conference*, 2014, pp. 1–4.

[11] L. F. Gul and L. Simisic, "Integration of Digital Fabrication in Architectural Curricula," in *Digital Fabrication in Education Conference, FabLearn Europe*, 2014, pp. 1–5.

[12] N. Padfield, M. Haldrup, and M. Hobye, "Empowering Academia Through Modern Fabrication Practices," in *Digital Fabrication in Education Conference, FabLearn Europe*, 2014, pp. 1–4.

[13] SCOPES Digital Fabrication - K-12 STEM Digital Fabrication. [Accessed 15-Feb-2018]. [Online]. Available: https://www.scopesdf.org/

[14] S. Tesconi and L. Arias, "Making as a Tool to Competence-based School Programming," in *FabLearn Europe Conference*, 2014, pp. 1–4.

[15] Fab Lab Limerick. A Space for Makers in Limerick City. [Accessed 15-Feb-2019]. [Online]. Available: http://fablab.saul.ie/

[16] WECREATE, Ireland's First Fab Lab in Cloughjordan Ecovillage. [Accessed 15-Feb-2018]. [Online]. Available: https://irishtechnews.ie/wecreate-building-irelands-first-fablab-in-cloughjodan-ecovillage/

[17] G. Cornetta, F. J. Mateos, A. Touhafi, and G.-M. Muntean, "Design, simulation and testing of a cloud platform for sharing digital fabrication resources for education," *J Cloud Computing*, vol. 8, no. 12, pp. 1–12, 2019.

[18] G. Cornetta, F. J. Mateos, A. Touhafi, and G.-M. Muntean, "Modelling and Simulation of a Cloud Platform for Sharing Distributed Digital Fabrication Resources," *Computers*, vol. 8, no. 2, pp. 1–47, 2019.

[19] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[20] V. Lampkin and *et al.*, *Building a Smarter Planet. Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM Redbooks, 2012.

[21] F. Shrouf, J. Ordieres, and G. Miragliotta, "Smart Factories in Industry 4.0: A Review of the Concept and of Energy Management Approached in Production Based on the Internet of Things paradigm," in *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, 2014, pp. 697–701.

[22] C. H. Muntean, D. Bogusevschi, and G.-M. Muntean, *Innovative Technology-based Solutions for Primary, Secondary and Tertiary STEM Education*. Paragon Publishing, September 2019.

[23] M. A. Togou, C. Lorenzo, G. Cornetta, and G.-M. Muntean, "NEWTON Fab Lab Initiative: A Small-Scale Pilot for STEM Education," in *Proceedings of EdMedia + Innovate Learning*, 2019, pp. 8–17.

[24] L. Zou, T. Bi, and G.-M. Muntean, "A DASH-based adaptive multiple sensorial content delivery solution for improved user quality of experience," *IEEE Access*, vol. 7, pp. 89 172–89 187, 2019.

[25] D. Zhao, A. E. Chis, G.-M. Muntean, and C. H. Muntean, "A large-scale pilot study on game-based learning and blended learning methodologies in undergraduate programming courses," in *Proc. of International Conf. on Education and New Learning Technologies (EDULEARN)*, 2018.

[26] C. H. Muntean, J. Andrews, and G.-M. Muntean, "Final frontier: An educational game on solar system concepts acquisition for primary schools," in *IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, July 2017, pp. 335–337.

[27] M. A. Togou, C. Lorenzo, E. Lorenzo, G. Cornetta, and G.-M. Muntean, "Raising Students' Interest in STEM Education Via Remote Digital Fabrication: An Irish Primary School Case Study," in *Proc. of International Conf. on Education and New Learning Technologies (EDULEARN)*, 2018, pp. 2835–2840.

TABLE I
SUMMARY OF SYSTEM PERFORMANCE

Evaluation technique	Scenario 1	Scenario 2
Brokers supporting MQTT	*Response time*: Apollo and RabbitMQ (32 ms) clearly outperform ActiveMQ (64 ms).	*Average total time to receive 10000 messages*: Apollo (380 ms), RabbitMQ (480 ms) and ActiveMQ (670 ms)
Service with and without QoS estimator	*Processing without QoS estimator*: Linear increase of processing time. 1) 1 second to process 100 requests 2) 12 seconds to process 1500 requests	*Processing with QoS estimator*: Linear increase of processing time. 1) 2 seconds to process 100 requests 2) 13 seconds to process 1500 requests
Non invasive machine status detection based on correlation techniques	*Fab Lab test based on current sensing with correlation-based state estimation.* Achieved correct state estimation: >99%.	*Crowded Fab Lab test based on current sensing with correlation-based state estimation.* Achieved correct state estimation: >90%.
Anomalies detection of sensor malfunctioning	*Fab Lab test based on active self-testing of sensor with frequency response evolution in time.* Achieved correct estimation: >99%.	*Crowded Fab Lab test based on active self-testing of sensor with frequency response evolution in time.* Achieved correct estimation: >99%.

[28] M. A. Togou, C. Lorenzo, M. Maddi, G. Cornetta, and G.-M. Muntean, "NEWTON Fab Lab Initiative: Attracting K-12 European Students to STEM Education Through Curriculum-Based Fab Labs," in *Proc. of International Conf. on Education and New Learning Technologies (EDULEARN)*, 2019, pp. 2901–2909.

Gianluca Cornetta received his Dr. Ing. degree from Politecnico di Torino (Italy) in 1995 and his Ph.D. degree from Universidad Politécnica de Cataluña (Barcelona, Spain) in 2001, both in electronic engineering. Presently, he is an Associate Professor with the Department of Information Engineering at Universidad San Pablo-CEU (Madrid, Spain). He is also a research fellow with the Department of Industrial Engineering (ETRO) of the Vrije Universiteit Brussel (Belgium) and an invited professor at Institute Supérieur d'Électronique de Paris (ISEP) where he teaches Radio-Frequency System design in the "Advances in Communication Environment" (ACE) Master. Prior to joining Universidad San Pablo-CEU he was a R&D Engineer at Infineon Technologies Gmbh developing embedded flash memories for the automotive and industrial markets and an ICT consultant at Tecsidel S.A. working in the field of real-time distributed embedded systems. He has published several papers in international journals and conferences, has authored four books and several book chapters, and has edited one book.

Abdellah Touhafi obtained the master in Industrial Engineering Sciences in 1992. In 1995 he finished his master in Electronic Engineering Sciences and in 2000 his PhD degree in Engineering Sciences from Vrije Universiteit Brussel (VUB). Currently he is professor at the Faculty of Engineering Sciences in VUB. He is a full member of the Industrial Engineering department, and associated member of the Electronics and Informatics department. His research interests include innovations in education, reconfigurable computing systems, embedded hardware design, sensor arrays, acoustics, environmental monitoring and embedded energy harvesting. He has cooperated in and coordinated several projects on national level and on European level. His research output includes 80+ co-authored articles, as well as 5 co-authored book chapters. He has been co-chair of the Mini-simposium ParaFPGA at the PARCO-conference in 2011,2013 and 2015 and is TPC member for several workshops and conferences. At VUB (INDI-department) he heads the RAPPTORLAB research group. Since december 2015 he is co-founder and the managing director of Lumency, a spin-off company specialised in smart city solutions.

Mohammed Amine Togou is a postdoctoral researcher with the Performance Engineering Laboratory at Dublin City University, Dublin, Ireland. He received a B.S. and M.S. degrees in computer science and computer networks from Al Akhawayn University in Ifrane, Morocco and a Ph.D. degree in computer science from the University of Montreal, Canada. His current research interests include connected vehicles, SDN-NFV, technology enhanced learning, IoT, smart cities and machine learning. Currently, he is working on the Horizon 2020 EU project NEWTON (http://newtonproject.eu).

Gabriel-Miro Muntean is an Associate Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and the Co-Director of the DCU Performance Engineering Laboratory. He has published over 350 papers in top-level international journals and conferences, authored four books and 18 book chapters, and edited six additional books. His research interests include quality, performance, and energy saving issues related to multimedia and multiple sensorial media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and chair and reviewer for important international journals, conferences, and funding agencies. Dr. Muntean is NEWTON EU Horizon 2020 project coordinator (http://newtonproject.eu) and senior member of IEEE and IEEE Broadcast Technology Society.