Journal of Software Engineering and Applications, 2016, 9, 425-438
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Software Defined Things in Manufacturing Networks

Arshdeep Bahga, Vijay K. Madisetti, Raj K. Madisetti, Andrew Dugenske

Georgia Institute of Technology, Atlanta, GA, USA
Email: arshdeepbahga@gmail.com, vkm@gatech.edu, raj.madisetti@gmail.com, dugenske@gatech.edu

How to cite this paper: Bahga, A., Madisetti, V.K., Madisetti, R.K. and Dugenske, A. (2016) Software Defined Things in Manufacturing Networks. *Journal of Software Engineering and Applications*, **9**, 425-438.
http://dx.doi.org/10.4236/jsea.2016.99028

Received: July 15, 2016
Accepted: September 3, 2016
Published: September 7, 2016

Copyright © 2016 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

IoT technologies are being rapidly adopted for manufacturing automation, remote machine diagnostics, prognostic health management of industrial machines and supply chain management. A recent on-demand model of manufacturing that is leveraging IoT technologies is called Cloud-Based Manufacturing. We propose a Software-Defined Industrial Internet of Things (SD-IIoT) platform for as a key enabler for cloud-manufacturing, allowing flexible integration of legacy shop floor equipment into the platform. SD-IIoT enables access to manufacturing resources and allows exchange of data between industrial machines and cloud-based manufacturing applications.

Keywords

Software-Defined Things, Cloud-Based Manufacturing, NETCONF, LoRa

1. Introduction

Internet of Things (IoT) comprises things that have unique identities and are connected to the Internet. The "Things" in IoT refers to IoT devices which have remote sensing and/or actuating capabilities. IoT devices can exchange data with other connected devices and applications (directly or indirectly), or collect data and process the data either locally or send the data to centralized servers or cloud-based application back-ends for processing [1]. IoT technologies are promising for industrial and manufacturing systems and the experts have forecast a trillion dollar impact on these sectors by IoT. IoT technologies are being adopted for manufacturing automation, remote machine diagnostics, prognostic health management of industrial machines and supply chain management.

A recent on-demand model of manufacturing that is leveraging IoT technologies is called Cloud-Based Manufacturing (CBM) [2]. CBM enables ubiquitous, convenient,

DOI: 10.4236/jsea.2016.99028 September 7, 2016

on-demand network access to a shared pool of configurable manufacturing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [3].

In this paper, we propose a Software-Defined Industrial Internet of Things (SD-IIoT) platform as a key-enabler for cloud-based manufacturing, especially towards integrating legacy shop floor equipment into the cloud environment. SD-IIoT platform enables access to manufacturing resources and allows exchange of data between industrial machines and cloud-based manufacturing applications. The contributions of this work are 1) a Software-Defined Industrial Internet of Things (SD-IIoT) platform is proposed; 2) a cloud-based big data analytics stack for IIoT is proposed; 3) an IIoT device management approach based on NETCONF and YANG is proposed; 4) an implementation case study based on the LoRa wireless communication technology is presented.

2. Related Work

CBM is a service-oriented manufacturing model in which service consumers are able to configure, select, and utilize configurable manufacturing resources. CBM leverages the four key cloud computing service models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Hardware-as-a-Service (HaaS), and Software-as-a-Service (SaaS) [4]. Software defined systems enable abstracting the control and management functionality from the underlying devices and providing these functionalities in a software layer. A software defined framework for Internet of Things (SDIoT) was proposed in [5]. The core of the SDIoT framework includes a control layer comprising an IoT controller, a software-defined network (SDN) controller, a software-defined storage (SDStore) controller and a software-defined security (SDSec) controller. In [6], a software-defined IoT architecture for decoupling smart urban sensing applications from underlying physical infrastructures is proposed. A software-defined Industrial Internet of Things architecture for managing physical devices and providing an interface for information exchange, is proposed in [7].

3. Software Defined Industrial Internet of Things (SD-IIoT) Platform

Figure 1 shows the architecture of the proposed SD-IIoT platform. The platform comprises a network of Software Defined Things (SDT) as IoT devices (end-nodes, routers and gateways) and a cloud-backend system. The IoT devices run an SD-IIoT controller which enables the attached industrial machines to communicate with the cloud and also allows cloud-based manufacturing applications to access the machines. The IoT devices in SD-IIoT offer a "plug and play" solution that allows machines to exchange data on their operations to the cloud and receive commands from cloud-based manufacturing applications. The SD-IIoT controller includes the following interfaces:

3.1. Northbound Interface

The northbound interface (cloud bridge) allows an IoT device (gateway) to publish the

SurgeTech Ex. 2005, Page 2 of 15

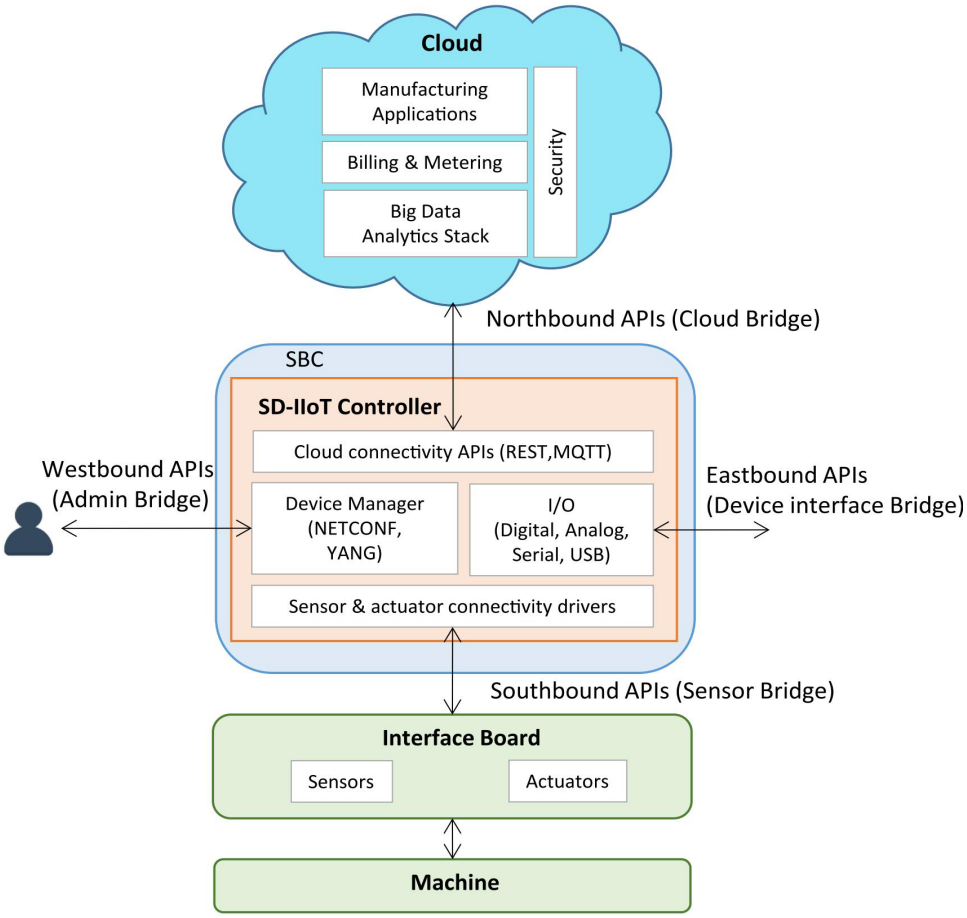

Figure 1. Proposed software defined industrial internet of things (SD-IIoT) platform.

data to the cloud. For communication with the cloud, protocols as MQTT and REST (over HTTPS) are used. The northbound APIs in the SD-IIoT controller include the cloud connectivity libraries (such as REST and MQTT libraries).

The cloud-backend system comprises a big data analytics stack, billing and metering services and manufacturing applications. The big data analytics stack allows analysis of data collected from the industrial machines. **Figure 2** shows the components of the proposed analytics stack. The components are described as follows:

3.1.1. Data Access Connectors

The Data Access Connectors includes tools and frameworks for collecting and ingesting data from various sources into the big data storage and analytics frameworks. The choice of the data connector is driven by the type of the data source. The types of connectors are described as follows:

- **Publish-Subscribe Messaging**: Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. We use publish-subscribe messaging frameworks such as Apache Kafka and Amazon Kinesis for the proposed analytics stack.

- **Messaging Queues**: Messaging queues are useful for push-pull messaging where the producers push data to the queues and the consumers pull the data from the queues. The producers and consumers do not need to be aware of each other. We use messaging queues such as RabbitMQ, ZeroMQ, RestMQ and Amazon SQS for the proposed stack.
- **Custom Connectors**: Custom connectors can be built based on the source of the data and the data collection requirements. These include REST and MQTT connectors.

3.1.2. Data Storage

The data storage block in the big data stack includes distributed filesystems and non-relational (NoSQL) databases, which store the data collected from the raw data sources using the data access connectors. We use the Hadoop Distributed File System (HDFS) for the proposed analytics stack. HDFS is a distributed file system that runs on large clusters and provides high-throughput access to data. With the data stored in HDFS, it can be analyzed with various big data analytics frameworks built on top of HDFS. For certain analytics applications, it is preferable to store data in a NoSQL database such as HBase. HBase is a scalable, non-relational, distributed, column-oriented database that provides structured data storage for large tables.

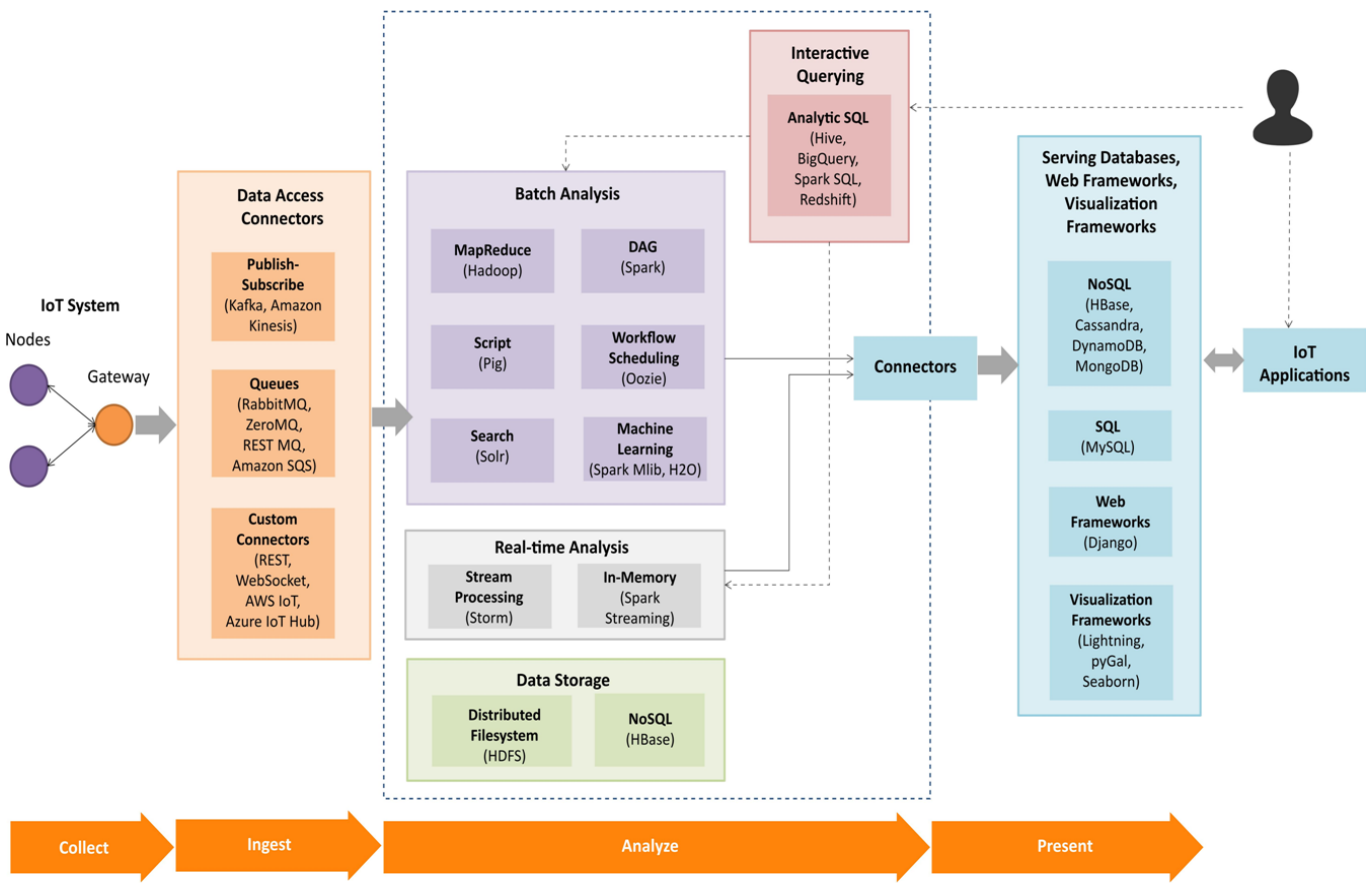

Figure 2. Proposed cloud-based big data analytics stack for IIoT.

3.1.3. Batch Analytics

The batch analytics block in the big data stack includes various frameworks which allow analysis of data in batches. These include the following:

- **Hadoop-MapReduce**: Hadoop is a framework for distributed batch processing of big data. The MapReduce programming model is used to develop batch analysis jobs which are executed in Hadoop clusters.
- **Pig**: Pig is a high-level data processing language which makes it easy for developers to write data analysis scripts which are translated into MapReduce programs by the Pig compiler.
- **Oozie**: Oozie is a workflow scheduler system that allows managing Hadoop jobs. With Oozie, you can create workflows which are a collection of actions (such as MapReduce jobs) arranged as Direct Acyclic Graphs (DAG).
- **Spark**: Apache Spark is an open source cluster computing framework for data analytics. Spark includes various high-level tools for data analysis such as Spark Streaming for streaming jobs, Spark SQL for analysis of structured data, MLlib machine learning library for Spark, and GraphX for graph processing.
- **Solr**: Apache Solr is a scalable and open-source framework for searching data.
- **Machine Learning**: Spark MLlib is the Spark's machine learning library which provides implementations of various machine learning algorithms. H_2O is an open source predictive analytics framework which provides implementations of various machine learning algorithms.

3.1.4. Real-Time Analytics

The real-time analytics block includes the Apache Storm and Spark Streaming frameworks. Apache Storm is a framework for distributed and fault-tolerant real-time computation. Storm can be used for real-time processing of streams of data. Storm can consume data from a variety of sources such as publish-subscribe messaging frameworks (such as Kafka or Kinesis), messaging queues (such as RabbitMQ or ZeroMQ) and other custom connectors. Spark Streaming is a component of Spark which allows analysis of streaming data such as sensor data, click stream data, web server logs, for instance. The streaming data is ingested and analyzed in micro-batches. Spark Streaming enables scalable, high throughput and fault-tolerant stream processing.

3.1.5. Interactive Querying

Interactive querying systems allow users to query data by writing statements in SQL-like languages. We use the following interactive querying frameworks for the proposed analytics stack:

- **Spark SQL**: Spark SQL is a component of Spark which enables interactive querying. Spark SQL is useful for querying structured and semi-structured data using SQL-like queries.
- **Hive**: Apache Hive is a data warehousing framework built on top of Hadoop. Hive provides an SQL-like query language called Hive Query Language, for querying data residing in HDFS.

SurgeTech Ex. 2005, Page 5 of 15

- **Amazon Redshift**: Amazon Redshift is a fast, massive-scale managed data warehouse service. Redshift specializes in handling queries on datasets of sizes up to a petabyte or more parallelizing the SQL queries across all resources in the Redshift cluster.
- **Google BigQuery**: Google BigQuery is a service for querying massive datasets. BigQuery allows querying datasets using SQL-like queries.

3.1.6. Serving Databases, Web & Visualization Frameworks

While the various analytics blocks process and analyze the data, the results are stored in serving databases for subsequent tasks of presentation and visualization. Web frameworks and serving databases can be used to build IoT applications. The applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

3.2. Southbound Interface

The southbound interface (sensor bridge) between the interface board and the SBC enables the SBC to capture sensor data from the interface board and also send control signals to the actuators. The Southbound APIs include sensor and actuator connectivity libraries which are installed on the SBC. Figure 3 shows a reference architecture of the interface board. The interface board provides easy and configurable connections to a variety of machines. The interface board makes use of digital, analog, serial and USB interfaces to capture data from a variety of sensors and systems. The interface board has a serial interface to the single-board computer (SBC). While modern industrial machines can directly communicate with the interface board (over digital, analog, serial or USB interfaces), many legacy machines make use of controllers that are impractical to access or digital communication is nonexistent. Therefore, the interface board makes use of sensors which are external to the legacy machines' control box.

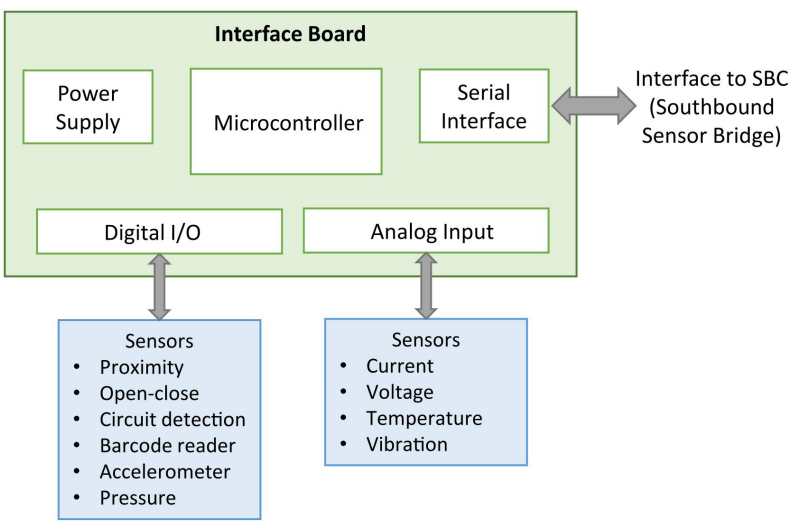

Figure 3. Reference design for interface board.

SurgeTech Ex. 2005, Page 6 of 15

3.3. Eastbound Interface

The eastbound interface (device bridge) enables connectivity to external systems over digital, analog, serial and USB connections. For wireless connectivity to other IoT devices various wireless communication protocols such as 802.15.4, ZigBee, LoRa or Bluetooth Low Energy (BLE), are used. The wireless communication modules for these protocols interface with the SBC over digital, analog, serial or USB interfaces.

3.4. Westbound Interface

The westbound interface (admin bridge) allows managing and configuring the IoT device using a router-like web interface and also viewing the device status and statistics. For device management, we use the Network Configuration Protocol (NETCONF). Network Configuration Protocol (NETCONF) is a session-based network management protocol. NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices [8]. **Figure 4** shows a generic IoT system management architecture based on NETCONF.

NETCONF works on SSH transport protocol and ensures reliable delivery of messages. For framing request and response messages NETCONF uses XML-encoded Remote Procedure Calls (RPCs). For retrieving and editing configuration data from net-

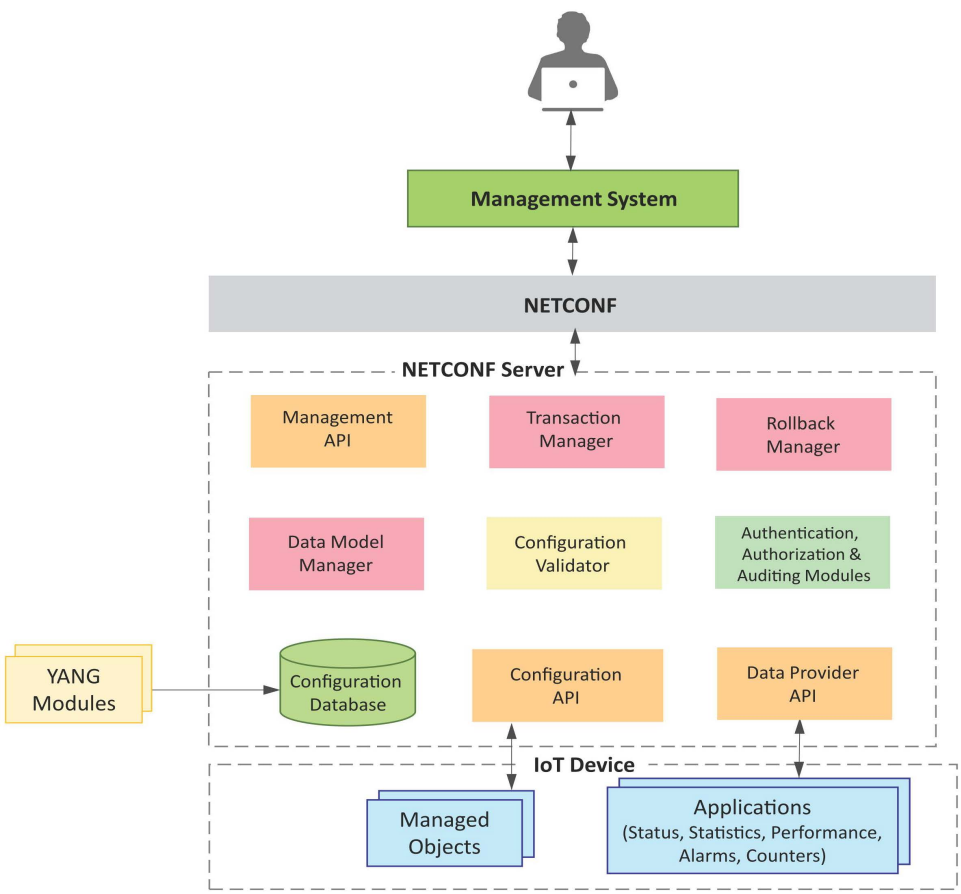

Figure 4. IoT device management with NETCONF-YANG.

SurgeTech Ex. 2005, Page 7 of 15

work devices, various operations are provided by NETCONF. Configuration and state data in NETCONF is encoded in XML. NETCONF uses a data modeling language called YANG [9] for defining the schema of the configuration and state data. YANG modules define the configuration data, state data and RPC calls. NETCONF provides a clear separation of the configuration and state data. The configuration data is stored on a NETCONF configuration datastore on the NETCONF server (which runs on the IoT device). The management application (which acts as a NETCONF client), establishes a NETCONF session with the server. NETCONF clients send requests to the server for editing the configuration data or retrieving the current configuration. A Management System is used to send NETCONF messages for configuring the IoT device. The management applications use a management API to start NETCONF sessions, read state data, read/write configuration data, and invoke RPCs. Transaction Manager is responsible for executing all the NETCONF transactions. The Rollback manager is used to rollback a current configuration to its original state. The Configuration validator ensures the validity of the resulting configurations after applying different transactions. For reading configuration data from the configuration datastore, a configuration API is used. The configuration API also allows writing operational data to the operational datastore. The Data Provider API is used for reporting device statistics and operational data [1].

The device management approach (based on NETCONF and YANG) used by the proposed SD-IIoT platform provides a predictable and easy to use management capability and the ability to automate system configuration. For Industrial IoT systems that consist of multiple devices, ensuring system-wide configuration can be critical for the correct functioning of the system. System-wide configuration (in which all devices are configured in a single atomic transaction) ensures that all the devices in an IIoT system run the same configuration. This "all or nothing" approach adopted in the SD-IIoT platform ensures that the system works as expected.

4. SD-IIoT Deployment Topologies

Figure 5 shows the deployment topologies for the SD-IIoT platform. The topologies are described as follows:

- **Point-to-Point Topology**: A point-to-point topology is the simplest topology that includes an end-node and a gateway. There is a direct wireless connection between the end-node and gateway. The range of the network in a point-to-point topology is limited to a single hop and depends on the wireless communication technology/protocol being used.
- **Star Topology**: In a star topology there is a single gateway node to which all the end-nodes are directly connected over a wireless link. Star topology provides a consistent and predictable performance as the data packets from an end-node travel one-hop to directly reach the gateway.
- **Mesh Topology**: A mesh topology consists of end-nodes, router nodes and a gateway. The end-nodes send data to the routers which route the data to the gateway.

SurgeTech Ex. 2005, Page 8 of 15

Data packets from an end-node travel multiple hops to reach the gateway node. Mesh topology is useful to extend the network range as the network range is not limited to a single hop as in the case of a star topology.

4.1. IoT Device Types

Within the proposed SD-IIoT platform, the following types of IoT devices are possible:

- **End Node**: The end-nodes interface directly with the machines. The end-nodes do not have Internet connectivity, therefore, to publish data to the cloud, the end-nodes send the data to a gateway node. For communication between the end-nodes

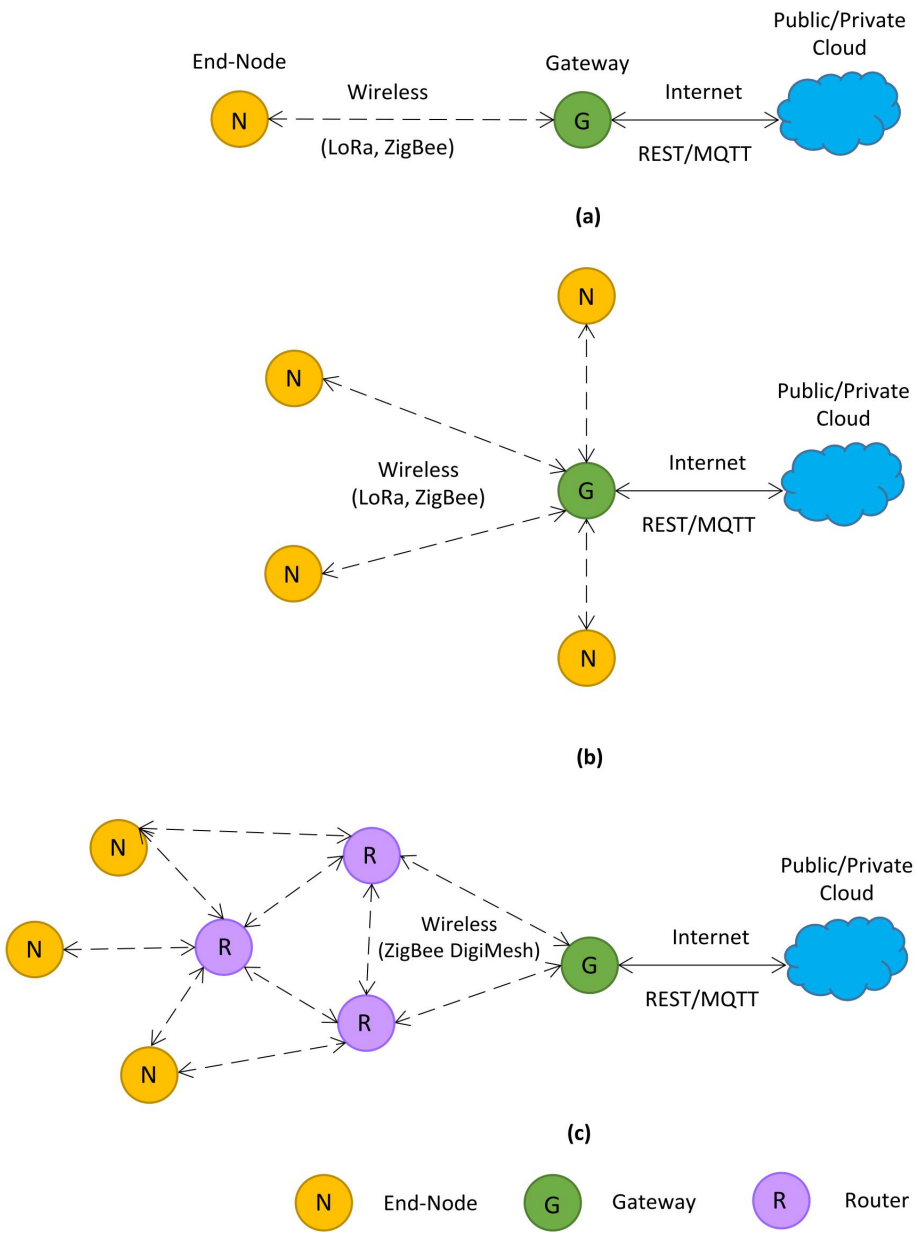

Figure 5. Deployment topologies for SD-IIoT: (a) point-to-point; (b) star topology; (c) mesh topology.

SurgeTech Ex. 2005, Page 9 of 15

and the gateway various wireless communication technologies such as LoRa, 802.15.4, ZigBee or Bluetooth Low Energy (BLE), are used. The end nodes implement the Southbound, Eastbound and Westbound interfaces.

- **Router**: The router nodes route the data sent by the end devices to the gateway. The router nodes implement the Eastbound and Westbound interfaces.
- **Gateway**: The gateway receives data from the end-nodes and sends it to a cloud backend over the Internet. The gateway implements the Northbound, Eastbound and Westbound interfaces.
- **Full Node**: A full node directly interfaces with the machines and publishes the data directly to the cloud. The full nodes implements the Northbound, Southbound, Eastbound and Westbound interfaces.

4.2. Forward & Reverse Path Communication

4.2.1. Forward Path

The forward path communication from the end-nodes to the gateway and finally to the cloud provides the ability to gather sensor data from the machines, machine maintenance and performance monitoring data. Each message transmitted by an end-node is acknowledged by the gateway. If an end-node doesn't receive an acknowledgement within a timeout period, the message is re-transmitted. The timeout period and number of retransmissions can be configured in the end-node.

4.2.2. Reverse Path

The reverse path communication from the cloud to the gateway and finally to the end-nodes provides the ability to update the "things" or the devices in the SD-IIoT platform over-the-air (OTA) and also send control commands to the machines. OTA updates allow adding new functionality or software updates for performance improvement. Whenever a new update is available the administrator pushes the new SD-IIoT controller image from the cloud to the gateway. The gateway advertises the image meta-data to all the end-nodes. Upon receiving the meta-data of the new image, the end-nodes compare the same with the meta-data of the currently installed image. The end-nodes then send an acknowledgement message to the gateway with a decision whether to trigger the update process or ignore the update. When an update trigger message is received, the gateway splits the new image into small fixed size chunks (typically chunk sizes vary from 5 kB to 50 kB) and transmits the chunks. The end-nodes acknowledge the receipt of each chunk and if the transmission of any chunk fails the gateway re-transmits the same. Upon receiving all the chunks, the end-nodes re-assemble the chunks to extract the new controller image. After successfully extracting the controller image, each end-node notifies the gateway about the image receipt. Upon receiving the image receipt messages from all the nodes in the network, the gateway broadcasts an activate message to all the end-nodes. Upon receiving the activate message, the end-nodes apply the new image. This activation process ensures that the updates are either applied to all the nodes or to none. The SD-IIoT platform also provides the flexibility to define "groups" of nodes that can be configured, controlled or

setup together.

5. Implementation Case Study

In this section we present an implementation case study of the proposed SD-IIoT platform based on the Beaglebone Black SBC, Arduino UNO microcontroller board and the LoRa wireless communication modules.

Industrial and manufacturing systems use various sensing technologies to obtain data on the performance and operations of machines. Collection and analysis of data can be challenging due to complex data communication systems and proprietary networks. One of the major challenges faced by Industrial Internet of Things (IIoT) systems is to enable communications over long ranges using very low power levels. Existing solutions which are based on WiFi or ZigBee communication protocols have limited range and high power requirements. We present a LoRa-based wireless communication system for the proposed SD-IIoT platform that allows collection of data from machines deployed in indoor environments. The data collected is published to cloud-based backend systems for further processing and analysis.

LoRa is optimized for long range communication with low-power consumption and supports large networks with millions of devices and low data rates (ranging from few bytes to few kilobytes per second) [10]. LoRa uses unlicensed frequencies that are available worldwide, for example, 868 MHz for Europe, 915 MHz for North America and 433 MHz band for Asia. These frequency bands are lower than the popular 2.4 GHz band and have much less interference. The use of lower frequencies enables much better coverage and great penetration in built environments. LoRa has ten predefined modes, including the largest distance mode (Mode-1), the fastest mode (Mode-10), and eight other intermediate modes. Data rates range from 0.3 to 38.4 kbps based on the LoRa mode. Typical ranges for LoRa communication are upto 13 miles (line of sight) and 1.2 miles (non line of sight, built environments). The LoRa physical layer uses a spread spectrum modulation technique. LoRa uses adaptive power levels which depend on the data rate needed and the link conditions.

Figure 6 shows the components of the IoT end-node and IoT gateway used for the implementation case study. For the end-node and gateway we used the BeagleBone Black SBC, Arduino UNO microcontroller board and LoRa SX1272 communication shield. The SBC captures data from the connected sensors and sends it to the microcontroller board over serial port. The microcontroller board receives data from the SBC and sends it to the gateway over the LoRa frequencies. The gateway receives data from the end-nodes and sends it to a Cloud backend over the Internet.

Figure 7 shows the results of the outdoor and indoor tests with the SD-IIoT devices (end-nodes and gateway). We measured the SNR (Signal to Noise Ratio), RSSI (Received Signal Strength Indicator) of the channel and RSSI of the received packets at the gateway. In the outdoor test we placed the end-nodes at different locations in an urban built-up environment, so that the communication between the nodes and the gateway is non-line-of-sight. For each point the number of buildings through which the signal had

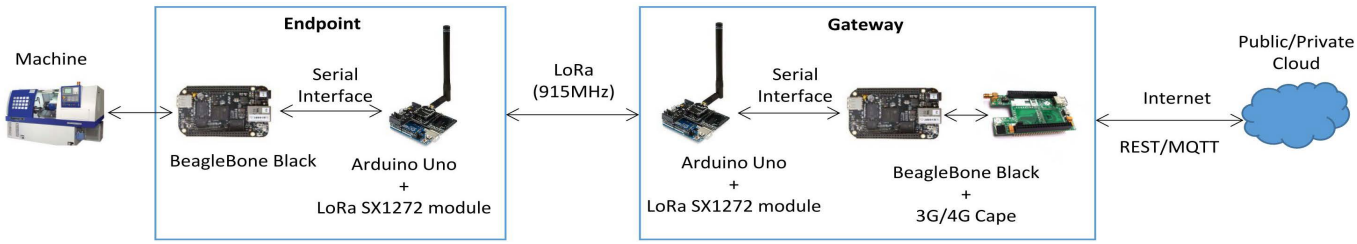

Figure 6. A realization of the proposed SD-IIoT platform based on Beaglebone Black, Arduino UNO and LoRa SX1272 communication modules.

Point	Range (m)	Through number of buildings	Mean SNR (dB)	Mean Packet RSSI (dBm)	Mean Channel RSSI (dBm)	Margin (dB)
1	546	6	-11	-128	-115	6
2	943	4	-1	-118	-116	16
3	1460	4	-3	-120	-115	14
4	2100	3	6	-101	-115	33

(a)

Floor	Mean SNR (dB)	Mean Packet RSSI (dBm)	Mean Channel RSSI (dBm)	Margin (dB)
1	6.5	-37	-115	97
2	6.3	-49	-113	85
3	6.2	-82	-112	52
4	5	-104	-116	30
5	3.3	-106	-113	28
6	2.6	-108	-114	26
7	1.3	-109	-113	25
8	-0.7	-115	-112	19

(b)

Figure 7. Test results for the SD-IIoT devices: (a) outdoor tests; (b) indoor tests.

to pass is shown in the table in Figure 7(a). For the indoor tests we placed the gateway on the first floor of an eight storied building and the end nodes on each of the eight floors. The SNR provides information on the link quality. LoRa communication modules can typically demodulate received signals with SNR values as low as −20 dB [11]. The channel RSSI provides information on the noise level of the channel by reporting the signal level detected (even in the absence of any signal being transmitted). The packet RSSI provides information on the received signal strength of the last received packet. If this value is greater than the sensitivity the packet is successfully detected. For these tests we used the LoRa Mode-1 with bandwidth of 125 kHz, coding rate of 4/5, spreading factor of 12, sensitivity value of −134 dB and a transmit power level of 7 dBm. The outdoor and indoor tests demonstrate that the SD-IIoT end-nodes and gateway are able to communicate over long ranges using low power levels.

Figure 8 shows the screenshot of the SD-IIoT controller dashboard that allows configuring the Northbound, Southbound, Eastbound and Westbound interfaces. Figure 9 shows a prototype of the SD-IIoT node developed for the implementation case study.

6. Conclusion & Future Work

We presented a Software Defined Industrial Internet of Things (SD-IIoT) platform as a key-enabler for cloud manufacturing allowing rapid integration of existing legacy shop floor equipment into a flexible framework. The SD-IIoT platform makes use of diffe-

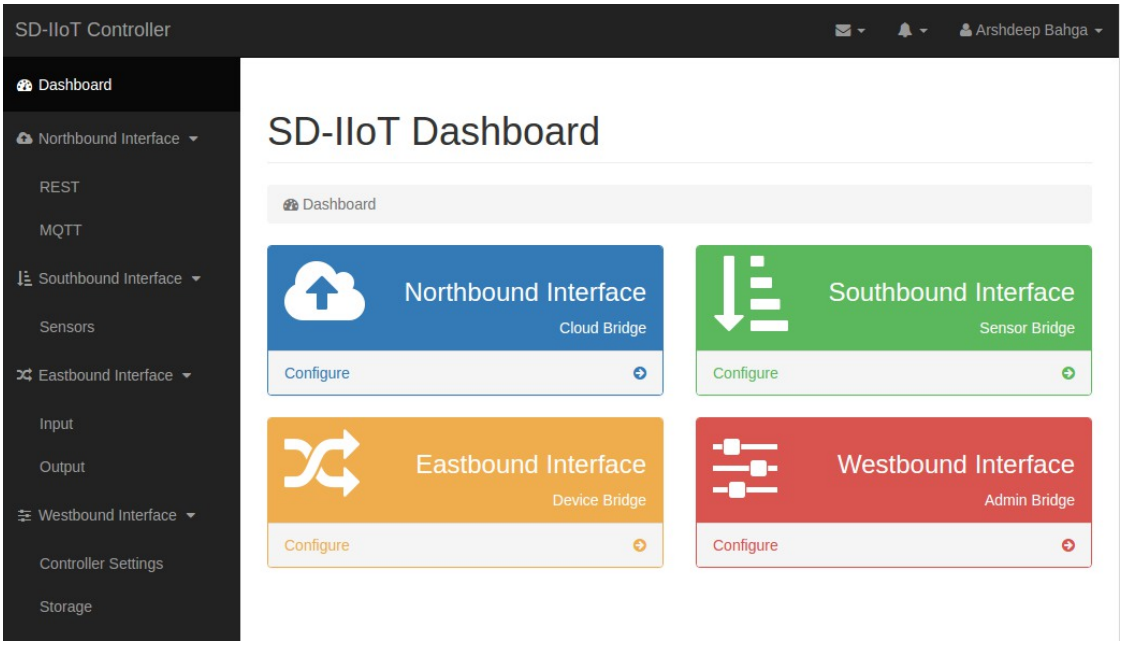

Figure 8. Screenshot of SD-IIoT controller dashboard.

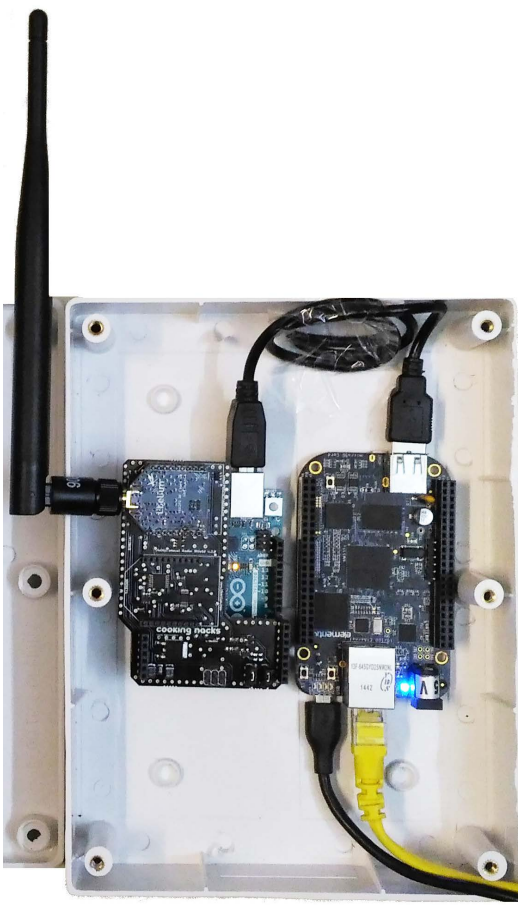

Figure 9. SD-IIoT node prototype based on BeagleBone Black SBC, Arduino UNO microcontroller board and LoRa SX1272 communication shield.

rent types of IoT devices (end-nodes, routers, gateways and full nodes). Each IoT device runs an SD-IIoT controller image which implements Northbound, Southbound, Eastbound or Westbound interfaces depending on the type of device. A full node implements all interfaces. A cloud-based big data analytics stack for IIoT was proposed that allows batch, real-time or interactive analysis of IIoT data. A device management approach based on NETCONF and YANG was proposed. We described an implementation case study of the proposed platform based on Beaglebone Black single-board computer, Arduino Uno microcontroller and LoRa communication modules. Future work will focus on integration of the SD-IIoT platform with real industrial machines and comparison of different wireless communication technologies for forward and reverse path communication.

References

[1] Bahga, A. and Madisetti, V. (2014) Internet of Things: A Hands-On Approach. VPT/CreateSpace Inc., Atlanta.

[2] Wu, D., Rosen, D.W., Wang, L. and Schaefer, D. (2015) Cloud-Based Design and Manufacturing: A New Paradigm in Digital Manufacturing and Design Innovation Computer-Aided Design. *Computer-Aided Design*, **59**, 1-14.

[3] Xu, X. (2012) From Cloud Computing to Cloud Manufacturing. *Robotics and Computer-Integrated Manufacturing*, **28**, 75-86. http://dx.doi.org/10.1016/j.rcim.2011.07.002

[4] Wu, D., Thames, J.L., Rosen, D.W. and Schaefer, D. (2013) Enhancing the Product Realization Process with Cloud-Based Design and Manufacturing Systems. *Journal of Computing and Information Science in Engineering*, **13**, 1-14.

[5] Jararweh, Y., Al-Ayyoub, M., Darabseh, A., Benkhelifa, E., Vouk, M. and Rindos, A. (2015) SDIoT: A Software Defined Based Internet of Things Framework. *Journal of Ambient Intelligence and Humanized Computing*, **6**, 453-461. http://dx.doi.org/10.1007/s12652-015-0290-y

[6] Liu, J., Li, Y., Chen, M., Dong, W. and Jin, D. (2015) Software-Defined Internet of Things for Smart Urban Sensing. *IEEE Communications Magazine*, **53**, 55-63.

[7] Wan, J., Tang, S., Shu, Z., Li, D., Wang, S., Imran, M. and Vasilakos, A. (2016) Software-Defined Industrial Internet of Things in the Context of Industry 4.0. *IEEE Sensors Journal*, **PP**, 1. http://dx.doi.org/10.1109/JSEN.2016.2565621

[8] NETCONF Configuration Protocol. http://tools.ietf.org/html/rfc4741

[9] YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF). http://tools.ietf.org/html/rfc6020

[10] LoRa Technology. LoRa Alliance. https://www.lora-alliance.org/What-Is-LoRa/Technology

[11] LoRa Radio Shield for Arduino. Libelium. https://www.cooking-hacks.com/lora-radio-shield-for-arduino-900-mhz

SurgeTech Ex. 2005, Page 14 of 15

Scientific Research Publishing

Submit or recommend next manuscript to SCIRP and we will provide best service for you:

Accepting pre-submission inquiries through Email, Facebook, LinkedIn, Twitter, etc.
A wide selection of journals (inclusive of 9 subjects, more than 200 journals)
Providing 24-hour high-quality service
User-friendly online submission system
Fair and swift peer-review system
Efficient typesetting and proofreading procedure
Display of the result of downloads and visits, as well as the number of cited articles
Maximum dissemination of your research work

Submit your manuscript at: http://papersubmission.scirp.org/