QoS-Aware Adaptive Resource Allocation Framework in the Integrated SDN Transport and IoT

Atefeh Meshinchi
Polytechnique Montréal
Montreal, Canada
email: atefeh.meshinchi@polymtl.ca

Ranwa Al Mallah
Ryerson University
Toronto, Canada
email: ranwa.almallah@ryerson.ca

Alejandro Quintero
Polytechnique Montréal
Montreal, Canada
email: alejandro.quintero@polymtl.ca

Abstract—**Internet of Things (IoT) is a key technological enabler to create smart environments and provide various benefits such as productivity improvement and business process cost minimization. For its successful deployment, various innovative applications are being developed in different IoT fields, each demanding various Quality of Service (QoS) requirements to achieve their desired objectives. In the context of a smart city, a huge number of IoT applications are being developed for emergency management operation (fire, flood, and earthquake) and city traffic congestion management. These applications require fast system reaction to get the valuable data and make appropriate decisions. Therefore, it is essential to design and develop a service model that ensures an appropriate level of QoS for such application. We propose a framework for QoS support in the IoT system considering the application specific QoS needs from the IoT networking subsystems such as the sensing and core transport networks. To achieve this, we take advantage of the Software-Defined Networking (SDN) technology integrated into the IoT system to provide a flexible and adaptive QoS-aware resource allocation scheme for IoT services. We put forward a programmable control layer to provide customized and generic support services for IoT applications.**

Keywords–Internet of Things; quality of service; software-defined Network; resource allocation.

I. INTRODUCTION

The Internet of Things (IoT) is the radical evolution of the current Internet into a network of interconnected objects. The IoT system is comprised of a large number of heterogeneous devices, and consequently diverse services and a multitude of applications. Each application requires a specific level of IoT system performance to operate appropriately and effectively. As a standard IoT architecture, multiple systems, such as devices, data, network and communication are involved in the fulfillment of the service [1].

Wireless Sensor Networks (WSNs), widely used in the IoT infrastructure, are comprised of hundreds of sensor nodes that provide the data for applications to be proceeded and transformed into useful outputs. Thus, the performance level of the services not only depends on the communication network (e.g., Internet), it also depends on the performance of the sensing network [2]. In comparison to Internet applications, which care about the performance of the data transmission services in terms of delay, loss rate, and bandwidth, Quality of Service (QoS) attributes, such as data accuracy and sampling rate are more valuable for IoT applications for service delivery [3].

Sensing nodes are generally low-end devices with strong constraints in energy, processing, storage, and transmission capabilities. Some sensors do not support the IP addressing scheme since implementation of the IP stack is resource-intensive and is too challenging in low-end devices. Therefore, the direct connection between applications and sensors cannot be established. An IoT gateway can act as a bridge between IP-based systems and non-IP sensors to facilitate data formatting and transmission. For cost efficiency, WSNs are generally designed as multi-service systems to be shared among multiple applications [4].

World-wide availability makes the Internet the best available option for inter-connectivity of IoT. However, the Internet enforces some challenges. In the legacy network, the control intelligence, which is implemented by various routing and management protocols, is embedded in every network element. Thus, vendor-dependent platforms and interfaces make the Internet evolution complex and slow. Also, the Internet provides best-effort services and is not capable to meet more specific requirements of applications in terms of service quality. In IoT, heterogeneous networks and devices create opportunities for a wide range of applications with very diverse QoS requirements, which may not be guaranteed by the best-effort Internet mechanisms [5]. On the other hand, Software Defined Networking (SDN) is a new paradigm for computer networking that appeared recently to break down the closeness of network systems in both control and data functions. SDN allows the network owners and administrators to programmatically initialize, control, and manage network behavior by decoupling the control plane from the data plane.

The Internet of Things is not a single technology like the Internet and it is compounded with many functions including sensing, processing, transmission, analysis, and deciding. So, many different technologies in terms of hardware, software, data, and communication are involved in the different layers of IoT architecture to implement the smart environments. Therefore, Quality of Service not only must be embedded in the production and design of all hardware and software components in IoT infrastructure, their integrations in the IoT system might raise needs for adaption and adjustment with the dynamic nature of IoT and application requirements. In other words, the overall IoT service quality and end-user satisfaction depend on various service providers like sensing service, network service, and cloud and it is carried out by all involved technologies and components of the IoT service.

Thus, the specific architecture would depend on the IoT application domains and the enabling technologies used in specific implementations.

Applications are fundamental to the IoT. They provide the presentation layer for the data captured from the billions of devices around the world. Unlike IT applications, IoT applications could be classified from different perspectives such as the type of information they manage, the type of recipient (person or system oriented), and their criticality. The IoT solution owner may define how the particular application must be treated from the QoS point of view depending on the business need. Besides, the growth of the innovative applications and their spontaneous deployment in the IoT system make the IoT Service Level Agreement (SLA) more dynamic and diverse. So, the application mission may vary over time regardless of the type of the traffic like data or voice which is the input for QoS classifiction in tradational IT environement. Apart from the flexibility and scalability of the IoT solutions, the simplicity of system control and management must be considered in the design of the solution so that IoT applications could be deployed easily and creatively.

The performance of an IoT application is impacted by the performance of the communication network and sensing data. Organizations need to design a flexible and scalable QoS framework to keep up with system growth, diverse application types and complexity of the system. To ensure that the system can provide the guaranteed service delivery, the QoS requirements of the application must be addressed at all involved subsystems and layers of the IoT architecture. In this paper, we aim to manage the IoT infrastructure resources to provide QoS support. To accomplish this goal, we integrate the SDN technology into IoT architecture to leverage SDN characteristics and features to design a QoS management framework, which could keep pace with innovative requirements of IoT application and support resource-dynamic environment. We propose a flexible and programmable control layer to provide customized and generic support services for the IoT applications.

The rest of the paper is organized as follows. In Section II, we provide a review of related studies that investigate different aspects of QoS in the IoT system. In Section III, we describe the proposed QoS support framework. Section IV details the framework workflow. Finally, in Section V, the primary conclusions and future work are outlined.

II. Literature review

Various QoS factors need to be taken into account when designing IoT services. This is due to the IoT architecture that depends on multiple subsystems. Approaches must consider QoS needs across the multiple subsystems, the diversity of the application domains, the volume and variety of the devices. Some research focus on enhancing the QoS control and management within IoT subsystems. For WSNs, a main component of the IoT infrastructure, studies investigate into the QoS-based design in the aspects of the routing protocols, clustering, and topology update. Solutions differ on the design methodology and details of the quality factors. Energy, bandwidth efficiency, storage, coverage optimization, and data accuracy enhancement are mostly targeted in the research.

On the other hand, some studies focus on QoS support schemes and architecture and aim at measuring the QoS attributes necessary for QoS-aware service delivery. In [6], the authors model a sensor's quality of information including principles and policies to exchanging the quality-related metadata about the collected data. In [7], the authors consider data accuracy and latency to estimate the Quality of Information (QoI) from the end-user perspective. In [8], Irfan et al. perform an analytical model for application prioritization and scheduling in a finite-capacity queuing system considering application performance requirements. Authors in [9] introduce a broker-based QoS management framework across IoT architectural layer. Most of the approaches and frameworks proposed concentrate on one or multiple QoS factors within a particular IoT subsystem. The works seek optimized hardware designs, protocols and decision-making algorithms to monitor, design or manage particular QoS parameters in WSNs. The platforms have very high level design and do not consider the end to end QoS management within a closed IoT system.

In recent years, software-defined systems and the idea of having a programmable network gained more interest in the industry and research institutions [10]. SDN architecture is made up of three logical layers: the data, control and application layers. The communication between the forwarding devices in the data layer and the centralized controller in the control layer is done through newly designed interfaces known as southbound interfaces. OpenFlow is the first standard southbound interface. The control layer communicates with application layer through a Northbound interface. Northbound interface enable flexibility in the programming of business-specific needs and has a centralized control over the network elements. From this perspective, studies in the literature propose generic traffic engineering techniques or innovative QoS management applications in terms of resource reservation, routing, queuing, and policy enforcement [11].

Some studies apply SDN technology to cellular and wireless networks. Software Defined Wireless Network (SDWN) apppeared as a way to provide a unified control plane to manage dynamic and heterogeneous wireless technologies, such as WiFi, WiMAX, 3G, LTE, and 5G [12]. In [13], they focus on mobility management, leveraging SDN architecture and its application within heterogeneous wireless environments in data flow context. On the other hand, Software-Defined WSNs (SDWSNs) seek to apply the separation of the control plane and data plane to the WSN architecture and provide re-configurable sensors. However, heterogeneity, data-centric nature of WSNs, and lack of TCP/IP stack support make softwarization more challenging in WSNs [14]. The SDIoT framework model proposed in [15] offers a centralized control system over security, storage and infrastructure resources. Most proposals are conceptual and analytical models which target architecture and framework and they are not established so far since the design of the controller and the management application would be a very complex task considering scale and diversity of IoT device and application [16]. Thus, the concept of SD-IoT is in its infancy and standardization efforts in terms of framework, protocol, software-defined application and assessment tools are still underway.

As a more granular research in this domain, the authors in [17] proposed an intrusion detection solution within the SDN-based cloud IoT environment. They leverage a machine learning technique to analyse network flow statistics to detect anomalous actions. There have been various studies on applying SDN to 5G systems. As another use case, in industrial IoT solutions, the authors in [18] focus on failure occurrences and recovery management in smart grid networks by leveraging SDN controller and with the aid of real-time monitoring mechanisms to improve the system resiliency. They propose an SDN-based programmable platform to manage policies through a centralized controller within 5G Radio access elements by abstracting the heterogeneities in that layer. Unlike our work, most proposals are conceptual and analytical models which target architecture and framework and they are not established so far since the design of the controller and the management application would be a very complex task considering scale and diversity of IoT device and application. Some of the IoT enabling technologies like the Internet and the cellular access networks (e.g. 3G, LTE) have the evolved QoS functions within their closed systems. However, their integrations into IoT bring issues and limitations. With the realization of the 5G technology, the barriers of the access networks are rectified, since 5G provides less-delay and high-bandwidth access network for the IoT system. The Internet imposes some limitations in terms of the supported QoS mechanisms. Compared to web applications, IoT applications are of dynamic and data-centric nature and they would have diverse QoS needs. Therefore, current QoS differentiation mechanism could not meet the diverse and progressive QoS needs of the IoT applications.

In all the studies, SDN technology is being integrated into the closed environments such as WSNs and 5G networks to manage the resources based on their specific characteristic, and particular point of view like security and performance. In contrast, in our proposed framework, we aim to provide centralized and programmable middleware to manage resources within multi-layer IoT system in an end to end fashion. The framework provides a way to better-fit resources from multiple networkers like 5G, 4G, Internet, or/and WSNs serving particular application data by enabling end to end visibility and improving the resource management. We realized a lack of study in the field of performance management in which heterogeneous devices, network resources, and application needs could be controlled and managed in an easy and flexible way. In the next section, we propose a middleware-based and SDN-oriented QoS support framework for IoT application. The SDN technology is leveraged in the networking layer to provide a flexible and adaptive control on network resource allocation based on the application QoS requirements.

III. QoS Support Framework for IoT

The proposed framework, illustrated in Figure 1, consists of several databases and functional elements deployed in the different layers of the IoT architecture. We categorize the building blocks of proposed framework in infrastructure, control and application layers.

A. Application Layer

In the application layer, the framework is composed of two databases, which could be implemented in the cloud environment for scalability and high availability.

- *Global wireless sensing network Database*: This database includes the profiles of IoT WSNs, such as supported services, bandwidth, coverage information (location), and IP addresses of the IoT gateways used for intercommunication between WSNs and external IP network. These general information could be provided by the Sensing Network Provider (SNP) in the network deployment phase. Also, this database keeps the overall status of WSNs in terms of the energy residue level, sensor availability status, the quality level of the collected data (QoI), and the service cost at any given time. Dynamic characteristics of WSNs could be estimated over the time by the IoT-gateways based on the implemented algorithms. Figure 2 demonstrates a logical format of the WSN profiles and supporting services.
- *SLA-based application QoS Database*: In general, the performance aspects of the service in terms of QoS attributes expected by a customer is covered in the SLA. In the IoT system, SLA includes application QoS requirements from the communication and sensing networks. Either application owner or service provider can provide the QoS-relevant information based on the agreed SLA. Figure 3 demonstrates the logical format for the database.

B. Infrastructure Layer

The infrastructure layer includes the WSNs, IoT gateways, and the forwarding network. Forwarding elements are controlled by the network controller located in control layer and WSNs are controlled by the IoT-gateways. In the proposed framework, IoT-gateways provide QoS-aware resource allocation and task scheduling within the local WSNs, through implemented algorithms and functions illustrated in Figure 4. *Sensors Status Collector* collects the status of the sensors in terms of availability, energy residue, and the quality level of collected data and updates *Sensors Status Database*. *QoS-aware Sensing Resources Allocator* allocates sensors for any IoT application considering the QoS and energy status of sensors stored in *Sensors Status Database*. *Active Demand Database* keeps the currently active service demand and relevant configuration setup. When a new request is received, IoT-gateway verifies this database to verify whether this demand is replicated or any current active demand could satisfy this new request.

The overall network lifetime and the quality of information are calculated by *WSN Energy Residue Calculator* and *WSN Quality Level Calculator* respectively, and then *Global wireless sensing network Database* are updated to keep most up-to-date information regarding sensor status.

Pre-scheduled Data Collection Setup includes application subscription information, predefined task and data collection setup in terms of QoS requirement, and this function is used when continuous data collection is needed by applications, so task setup is pre-scheduled in the gateway for efficiency and resource optimization purpose. The collected data could be stored either in the local or in the global storage.

C. Control Layer

The control layer consists of several functions implemented on top of the controller in the SDN network to handle QoS support routing management within the transport network. The building blocks and between-blocks relationships are illustrated in Figure 5.

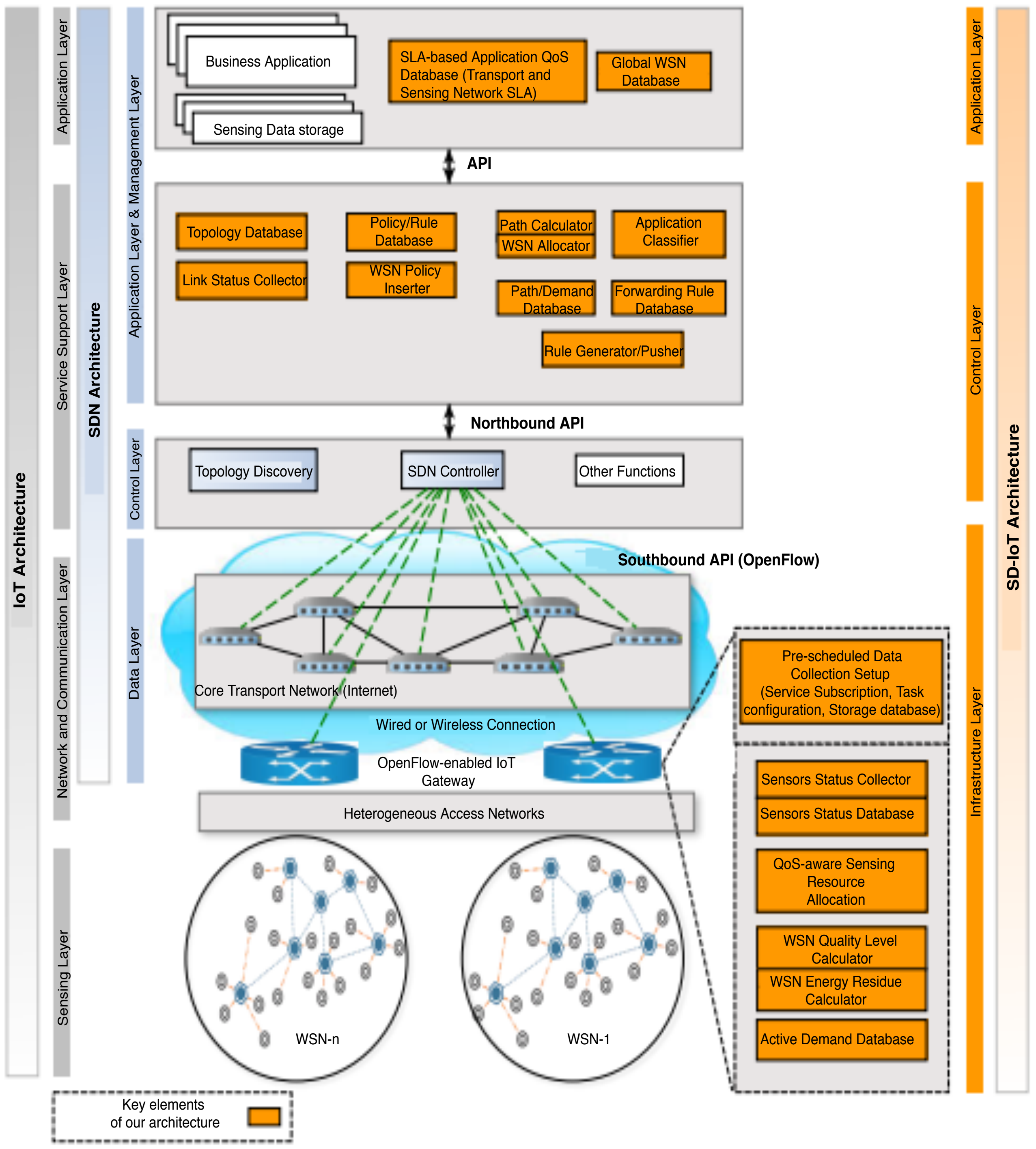

Figure 1. QoS framework for IoT through Transport SDN middleware.

According to the OpenFlow specification, the *Topology Discovery* function is implemented by default in the SDN-controller and enables the controller to discover the forwarding network topology. The controller discovers the network elements by exchanging HELLO messages and assessing their connection structures by the OpenFlow Discovery Protocol (OFDP) mechanism. The controller encapsulates an Link Layer Discovery Protocol (LLDP) packet as a packet-out message and sends it to the connected elements. The Network Element (NE) sends the received LLDP packet to all its neighbors which are connected directly to its active ports. A network element sends a received LLDP packet from another element to the controller as a Packet-in message since there is no matching entry in its Flow Table. The controller learns which network elements are connected directly to each other through received Packet-in messages and builds the global network physical

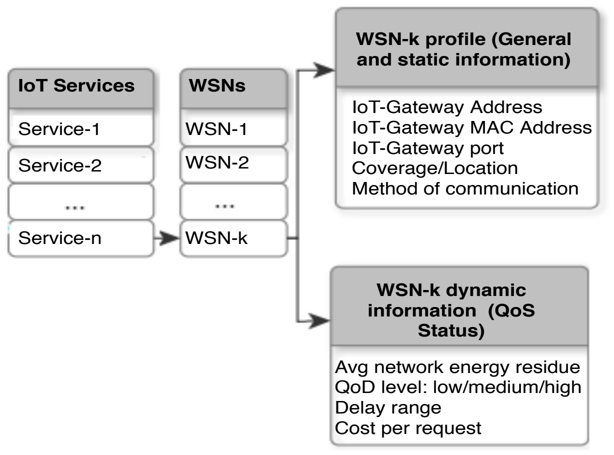

Figure 2. Logical format for Global sensing network Database.

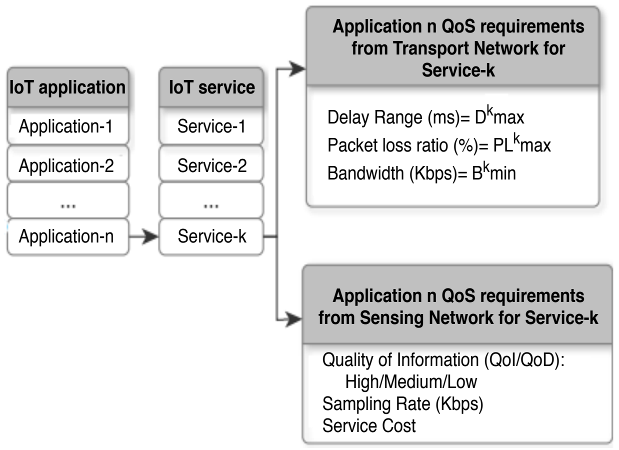
IoT application
Application-1
Application-2
...
Application-n
IoT service
Service-1
Service-2
...
Service-k
Application n QoS requirements from Transport Network for Service-k
Delay Range (ms)= D^k max
Packet loss ratio (%)= PL^k max
Bandwidth (Kbps)= B^k min
Application n QoS requirements from Sensing Network for Service-k
Quality of Information (QoI/QoD): High/Medium/Low
Sampling Rate (Kbps)
Service Cost

Figure 3. Logical format for IoT application SLA-based QoS Database.

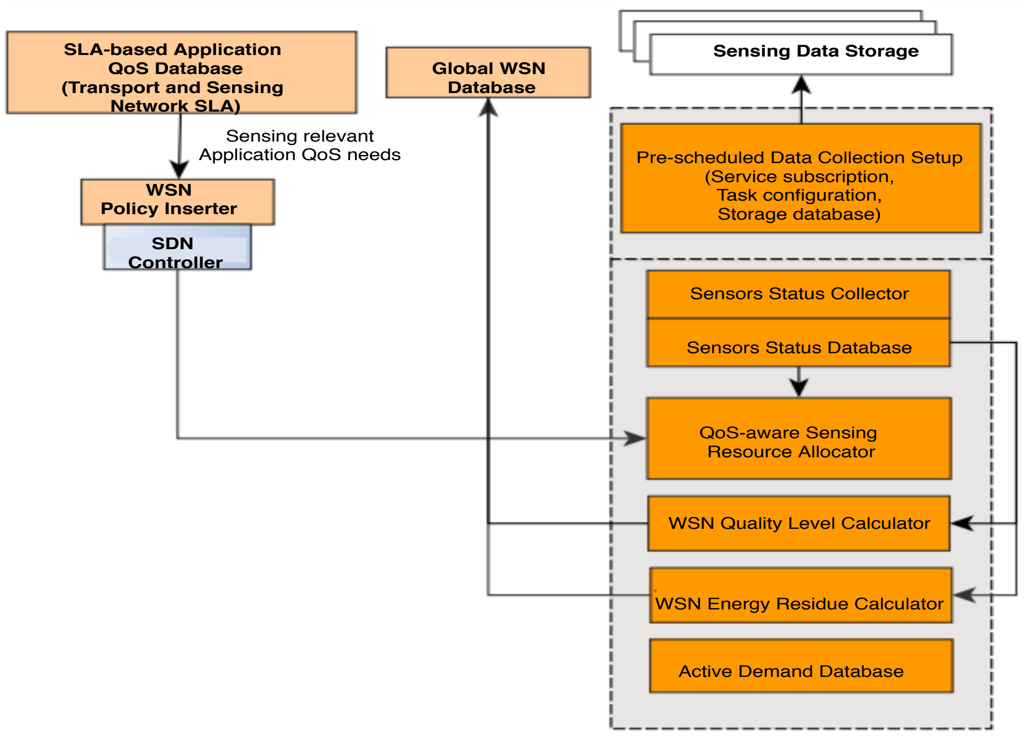
SLA-based Application QoS Database (Transport and Sensing Network SLA)
Global WSN Database
Sensing Data Storage
Sensing relevant Application QoS needs
WSN Policy Inserter
SDN Controller
Pre-scheduled Data Collection Setup (Service subscription, Task configuration, Storage database)
Sensors Status Collector
Sensors Status Database
QoS-aware Sensing Resource Allocator
WSN Quality Level Calculator
WSN Energy Residue Calculator
Active Demand Database

Figure 4. Infrastructure-layer components and their relationships within the framework.

topology [19].

Link Status Collector collects network performance parameters including the link packet loss ratio, bandwidth, and delay in real-time manner within the network topology discovered by *Topology Discovery* and stores the information in *Topology Database*.

As per the OpenFlow specification, several counters have been implemented in OpenFlow-enabled network elements to store packet processing records, such as flow, port, and queue [20]. They can be used to calculate the QoS parameters, such as link delay, link loss rate, and bandwidth. Flow-level counters provide information about a particular flow, e.g., a number of bytes forwarded, dropped, or erroneous and the duration of the flow to be delivered. Port-level counters provide more specific information about a particular port. Queue-level counters provide information about a particular queue attached to a particular output port.

In OpenFlow, for the traditional network, several tools have been developed to monitor and measure the network status, and they are classified in two big categories: passive or active monitoring [21]. The methods can be extended to the SDN and used in the implementation of the *Link Status Collector*. *Topology Database* is updated by both *Topology Discovery* and *Link Status Collector* periodically or in case of any changes in the network topology and link status, respectively.

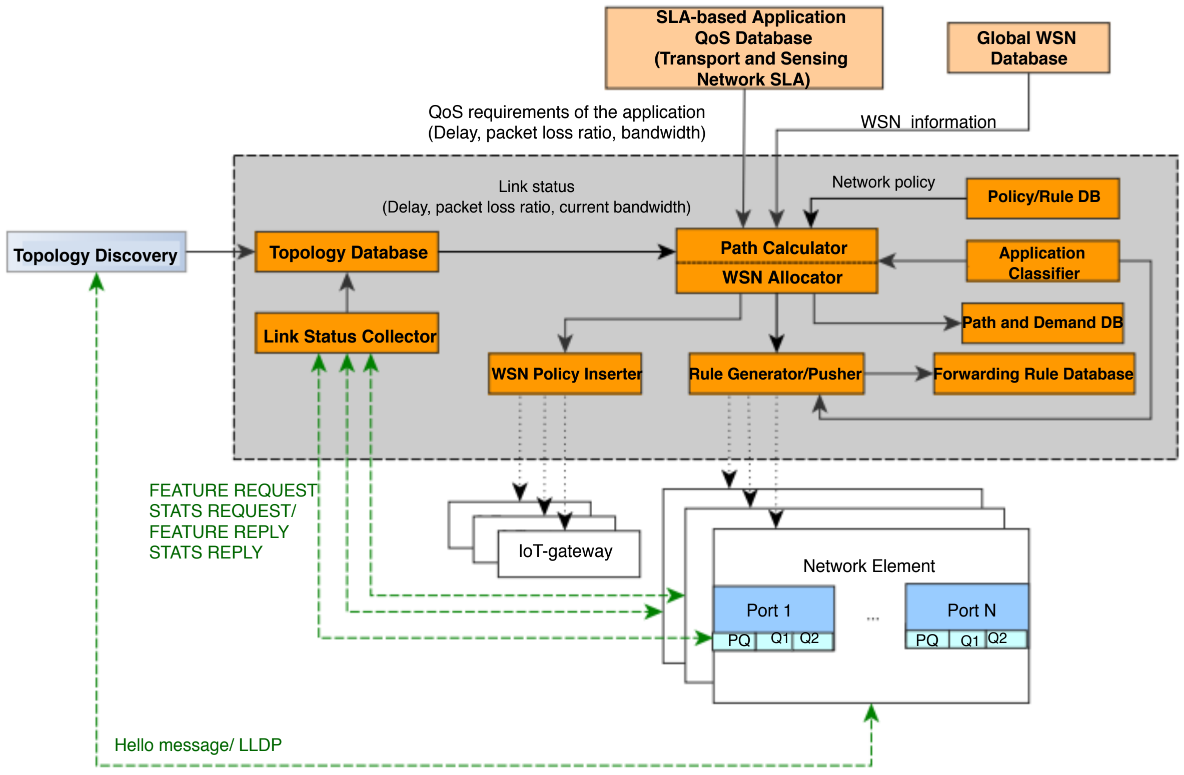

Figure 5. Control layer building blocks.

The messaging mechanisms in the OpenFlow protocol facilitates the communication between the *Link Status Collector* and the counters inside the network elements. FEATURE REQUEST/STATS REQUEST and FEATURE REPLY/STATS REPLY messages are used to request and return the value of counters respectively.

Policy/rule Database holds the network policy, such as bandwidth reservation, load balancing and admission control method to control the network resource allocation. The principles applied for network congestion or application preferences violation could also be kept in this database. The policy could be provided statically by network administrator or dynamically based on the network situations by the designed SDN application.

Application Classifier provides the application scheduling and prioritization within network elements. The default traffic scheduling algorithm implemented in the network elements is First-In First-Out where the traffic leaves the network element in the order in which they arrive. This method can not be suitable for an IoT environment since the scheduling algorithm does not take care of traffic QoS requirements and application class to prioritize them. Thus, mission-critical applications might experience non-acceptable delays or multimedia applications might not be served by proper capacity sharing. To provide a queuing mechanism as a fundamental scheduling technique in our framework, we classify IoT applications in three categories based on their sensitivity to delay and bandwidth: (1) control applications, also called mission-critical applications, such as city-traffic management and emergency management, which need very fast response with as less error as possible, (2) monitoring applications, such as intelligent security surveillance tasks, which are fed by cameras and need more throughput, and (3) analysis and inquiry applications, called general, such as the inquiry into the transported item state in the intelligent logistics, which are throughput and delay tolerant.

We propose three different queues in each port of the OpenFlow-enabled network elements: (1) Priority Queue (PQ) as the most prioritized queue includes mission-critical applications with intensive delay sensitivity. If the delay requirement of the application is less than a pre-defined threshold, it is marked as a high-prioritized demand and it is inserted in PQ of egress ports of network elements. (2) Q1 represents the data-centric application with bandwidth sensitivity, and less delay-sensitive compared to the pre-defined threshold. (3) Q2 contains applications with no strict QoS requirements, also called the best-effort application. Although queuing mechanisms guarantee the network bandwidth to the different applications, other techniques, such as *Complete Buffer Sharing* and *Preemptive Priority Scheduling* can also be implemented to minimize the network latency and increase efficiency of mission-critical applications.

Unlike IT applications which are categorized based on the type of traffic to which delay thresholds may be associated, IoT applications could be classified from different perspectives, such as the type of information they manage, the type of recipient (person or system oriented), and the criticality level of services they provide. Currently, IoT systems still suffer from the lack of standardized mechanisms to represent the diverse application requirements in order to be used as reference for our proposed model. Thus, such threshold do not exist. To setup the reasonable thresholds for QoS metrics, we plan to do experiments with multiple applications within an end to end deployed framework. It is only after an end to end implementation of the framework that simulations can enable us to gather data from production applications, and calculate the average value for the metric of delay threshold.

Moreover, in Complete Buffer Sharing scheme, the highest priority traffic push out the lower priority traffic. All packets in higher priority queue are served before a lower priority queue. In Preemptive Priority Scheduling, if a new process having a higher priority than the currently running process arrives, it gets selected immediately and the new process has not to wait until the currently running process finishes or yields. Having both mechanism in queue management would minimize the latency experienced by mission-critical applications served by the first queue in our proposed model.

A potential application of the three defined queues could be for instance in the context of a Smart City. A Smart City solution can leverage information and communication technologies to provide the critical infrastructure and services for city administration, transportation, education, health-care, public safety, and utilities. A City traffic and emergency management applications which need very fast response with as less error as possible, can be considered as delay-sensitive application served by PQ. Monitoring applications, such as intelligent security surveillance which are fed by cameras and need more throughput, can be classifed in Q1. Finally, analysis applications, such as the inquiry into the transported item state in the intelligent logistics, may be classifed as throughput and delay tolerant and may be pushed to Q2.

WSN Allocator determines the best-fit sensing network based on the current status of the WSNs (stored in *Global WSN Database* for the application request. *Path Calculator* queries the application QoS, policy and topology databases and calculate the SLA-respected routing path from the source of the demand to the IoT-gateway of the destination WSN for the application data-flow.

Rule Generator/Pusher translates the routing path made by *Path Calculator* into the actual configuration commands for each of the network elements. It generates the flow rules of the route information and configures the Flow Table of all elements along the paths. In OpenFlow channel, *FLOW MOD* message is used by the controller to add, delete, or modify the Flow Table entries in the network elements. *SET QUEUE/EN QUEUE* actions specify the particular queue in a particular port which the flow entry should be entered.

Path and Demand Database keeps the active application request and path associated to the request. When network topology or status is changed because of a fault or new design, *Path Calculator* verifies this database to determine whether any currently active path across the network is affected by this change. The new path is calculated for the impacted demands and re-installed in network elements. This database is updated based on the completion of the request and recalculation of the new path.

In OpenFlow, forwarding rules for all network-elements are kept in a data structure of the controller. In the proposed framework, *Forwarding Rule Database* keeps all active flow rules on the entire SDN network controlled by the controller. To have the optimized OpenFlow messaging, when *Rule Generator/Pusher* generates the rule based on the calculated path, it first verifies whether the associated flow rules exist in the Flow Table of the network element otherwise, the new flow entry is pushed in the network elements. To keep *Forwarding Rule Database* up-to-date, all the modification of the Flow Table entries must be reflected in this database. Therefore, the controller sends either an add or delete command, or receives the flow rule expiration notification from the network element, it modifies the database. To dynamically provide the sensing-related application QoS needs for the IoT-gateway, there could be two approaches. Whether IoT-gateway queries the SLA-based database to fetch the required information or *WSN policy pusher* enforce the required data in the OpenFlow-enabled IoT-gateway.

IV. Framework workflow

We describe the workflow diagram within the proposed framework when an IoT service request is initiated by an application. There are three data delivery models in an IoT system: (1) Query-driven, where data is generated on demand, (2) Event-driven, where data is generated in response to an event, and (3) Continuous or time-based in which real-time data is generated continuously or periodically.

In query-driven model, pull approach is used for data collection so that all sensors are kept silent until a request arrives from the associated application. In our scheme, the service request is received by the QoS management module on top of the SDN controller. First, *Path Calculator/WSN Allocator* verifies the application subscription for the requested service through querying *SLA-based Application QoS Database*. If the application subscription is approved, energy-efficient WSN is nominated to serve the request based on the WSN status information in the *Global WSN Database*. Next step is the calculation of the optimized routing path across the core transport network between the application and IoT-gateway of the determined WSN. To calculate the path, the designed QoS support routing function takes into account the source and the destination of the data, the QoS requirements of the application, and performance status of network links.

Moreover, application QoS requirements from the WSN are deployed within the programmable IoT-gateway by the remote controller, so IoT-gateway could assign the sensor and schedule the task by applying the received QoS information in the resource allocation and routing algorithms. Figure 6 and Figure 7 summarize the flow of steps when the data collection request is initiated by a query-driven application in the proposed framework.

In the event-driven data model, the sensors are programmed to report the data only when an event of interest occurs. The data collection approach used for the event-driven application is called the push approach in which sensors pro-actively collect data, either data is stored in the pre-defined storage or data is sent to the application. Therefore, data flows from the sensing layer towards the application layer. When IoT-gateway receives the collected data, it sends the data transfer request to the controller. Again, the QoS management module on top of the controller verifies the requested QoS needs and calculates the routing path across the core transport network respecting its preferences and the network status.

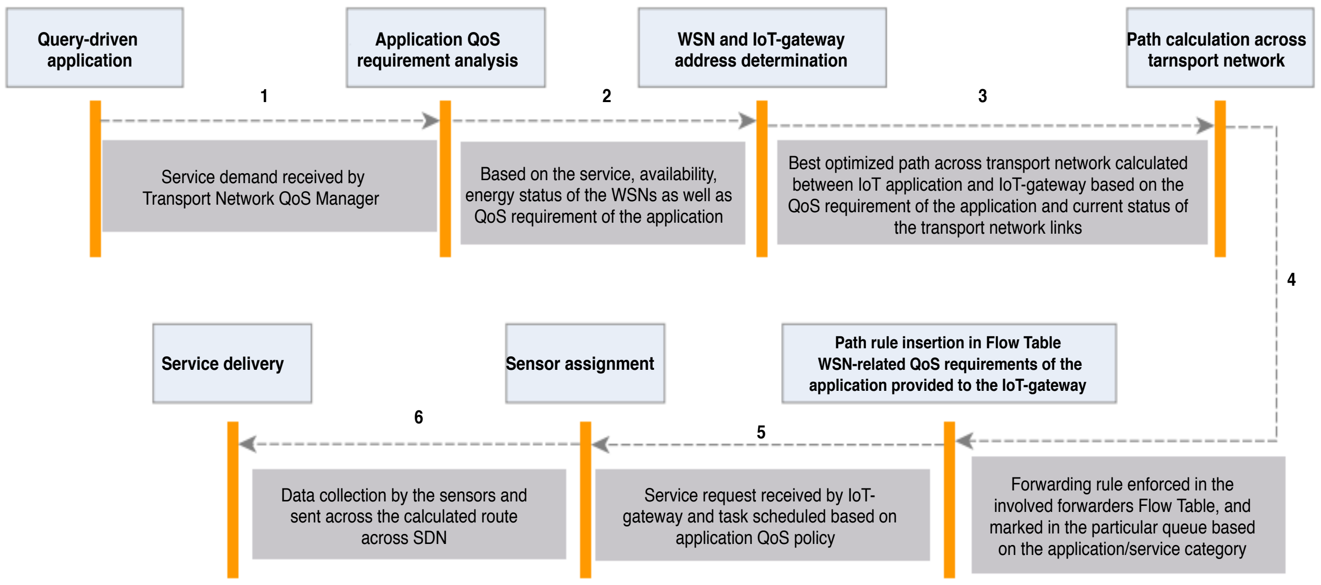
Query-driven application
Application QoS requirement analysis
WSN and IoT-gateway address determination
Path calculation across tarnsport network
1
2
3
4
Service demand received by Transport Network QoS Manager
Based on the service, availability, energy status of the WSNs as well as QoS requirement of the application
Best optimized path across transport network calculated between IoT application and IoT-gateway based on the QoS requirement of the application and current status of the transport network links
Service delivery
Sensor assignment
Path rule insertion in Flow Table WSN-related QoS requirements of the application provided to the IoT-gateway
6
5
Data collection by the sensors and sent across the calculated route across SDN
Service request received by IoT-gateway and task scheduled based on application QoS policy
Forwarding rule enforced in the involved forwarders Flow Table, and marked in the particular queue based on the application/service category

Figure 6. Sequence diagram when the proposed QoS support framework receives the service request from a query-driven IoT application.

Push approach could be more useful when multiple applications subscribe to the same data. To have a more efficient data collection process, *Prescheduled Data Collection Setup* provides the capability to publish data of subscribed applications so that applications could query the database and fetch the required data. In the continuous-based data model, same steps described for event-driven applications are followed to transfer the data from the IoT-gateway to the appropriate application server or database.

V. Performance evaluation

We present the performance evaluation in terms of architectural, network and application perspective. We expand on the application of the proposed framework in the sections below by describing its characteristics and advantages from three perspectives.

A. *Architectural perspective*

The proposed framework has been mapped into both multi-layer IoT architecture and SDN architecture. The SDN control layer/management layer provides a software layer between device and network infrastructures and applications. It works as the middleware and it enables the implementation of unified support services for the IoT system. In our design, the QoS module implemented over the controller provides the service support for QoS management in IoT framework: the end-to-end QoS routing across the SDN network, and QoS-aware sensing network allocation and sensor assignment. QoS needs of IoT application aims to be respected in each relevant layer, either core transport or sensing layer. Moreover, the resources and the routing path are allocated dynamically per-demand, depending on the specific service requirements and network resources status. Consequently, this design is adaptive to any changes in network and application QoS needs. The SDN northbound interfaces enable the enforcement of the quantified SLA-related QoS attributes directly in the resource allocation functions, which in the closed network is not possible due to the lack of such standard interfaces and programmable capabilities.

Since multiple programs could be implemented in the SDN controller, we are able to develop multiple algorithms and apply them in different conditions. We could use the best-effort routing path algorithm for the application class which has no strict QoS needs. For the QoS-based applications, a newly designed routing algorithm which considers the application constraint is applied. Since the framework uses the up-to-date information about the network status and application needs, it could provide the routing path dynamically over time.

Compared with the traditional QoS approaches such as IntServ and DiffServ, SDN-based core transport network resolves the limitation of the traffic differentiation and application classification according to their particular needs. As IoT system grows, it might bring the new class of applications with different QoS needs. Our architecture is flexible and fast-adapted based on the business needs. Applications can be classified to be treated differently regardless of the type of the traffic they process.

Based on the network resource status information and application request history captured by centralized SDN controller, it would be easier to learn the traffic pattern and predict the future traffic trend to extend the network based on business forecasts and avoid congestion. In fact, the authors in [22] proposed to leverage techniques based on artifical intelligence to resolve the resource management problem such as the traffic load prediction-based channel allocation problem to avoid traffic congestion. They propose a traffic load prediction-based adaptive channel assignment algorithm that aims to integrate SDN into IoT framework to improve the system control and management. The algorithm offers a centralized control system over security, storage and infrastructure resources. As a new algorithm, their work could be integrated into our proposed framework.

Implementing IoT data preprocessing and aggregation pro-

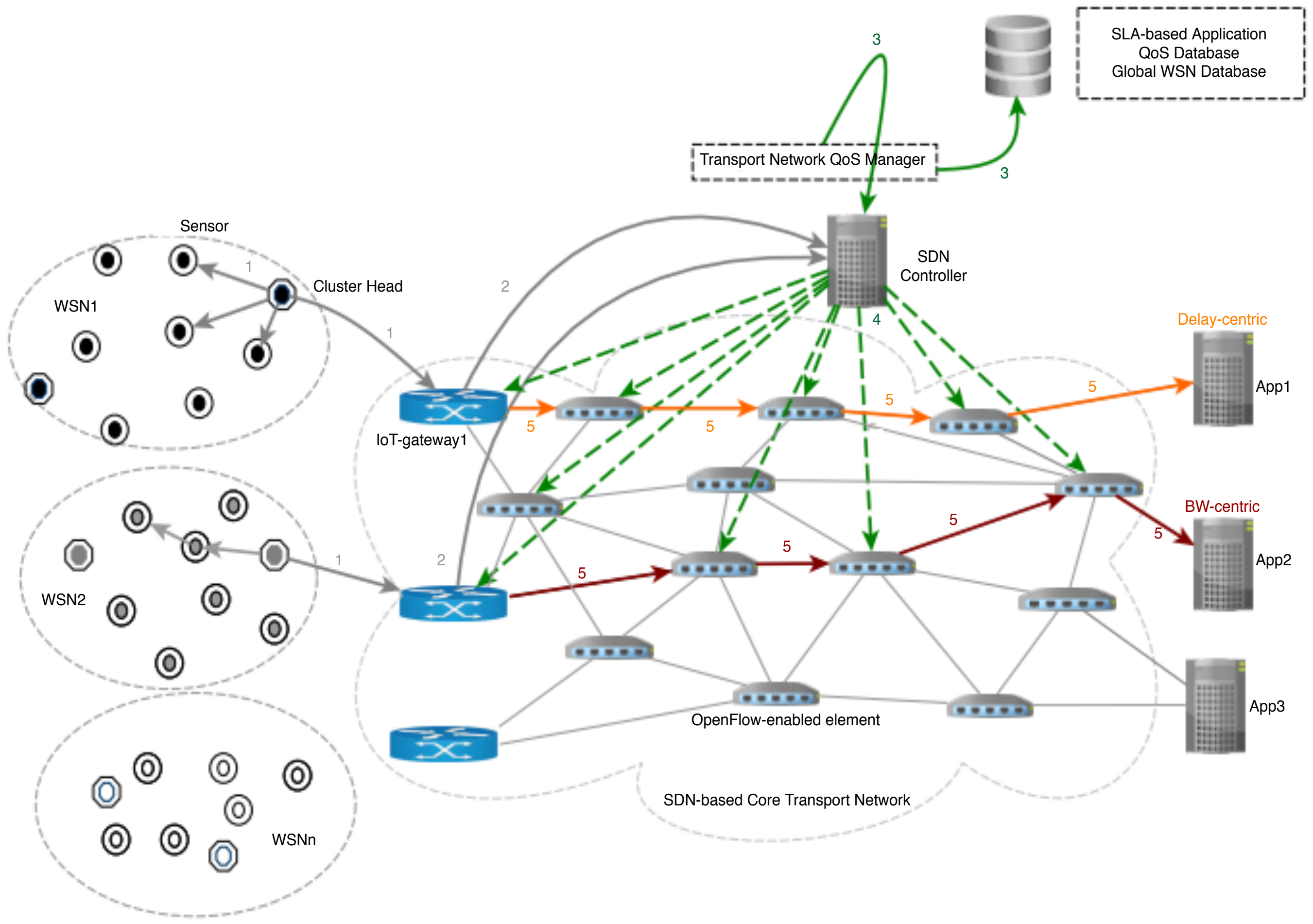

Figure 7. Mapping the steps into the framework components.

cess at the edge of the network within the local IoT-gateway not only enhances the transmission resource optimization and SDN network throughput, it speeds up the task scheduling and data acquisition for the user. The OpenFlow-enabled IoT-gateway also enables the sensing network programmability. The centralized algorithms and mechanisms in the IoT-gateway could be reprogrammed based on business and application needs, or when new optimized methods and solution are invented to manage the sensor resources in terms of clustering, routing, and task scheduling.

In [23], the authors proposed an SDN-capable IoT-GW that could facilitate the deployment of our proposed end to end QoS framework. We could leverage from the techniques of their study. With aid of specialized intelligent routing techniques developed in IoT gateways, wired and wireless devices could be managed in flexible and efficient way, although WSNs with low-end devices could be characterised and managed with a data-driven approach through centralized IoT-GW.

B. Network perspective

Network resource management in the globally distributed network such as transport network and the Internet is extremely complicated. One of the main benefits of the SDN-based core transport network is that it simplifies the network operation and management, compared with the traditional IP-based network which is defined in isolation and each vendor-dependent protocol only addresses a specific problem. We could leverage from SDN and its capabilities to develop customized network control and management services for core transport network. The controller as a centralized brain of SDN provides the single global map of the network and it abstracts the core transport network topology from the application layer. It enables the intelligent and agile decisions making regarding flow direction, control, and speedy network reconciliation when a link fails. The SDN controller can run multiple algorithms simultaneously in the field of network operation and maintenance. Therefore, network developer could deploy a wide range of the customized network application in terms of security, monitoring, performance management and fault-diagnosis. Also, scalability can be improved by centralizing the controller, where there is more global and less detailed view of the network elements.

Our framework dynamically collects the transport network topology and links status information. It increases the awareness of the network resource status at any given time. The design routing path excludes the higher utilized and congested links from the logical network topology when computing the route across SDN. It aims to make the link utilization balanced and it prevents the congestion probabilities in the transport network. It increases the transport network availability and accordingly, the IoT service availability. Suppose that the IoT sensing resources are available but the transport network is congested, it leads to failing the data transfer.

Furthermore, the centralized QoS management function does not deal with low-level configuration of data plane network elements. All the information such as SLA and network policy are specified in an abstracted level. Thus, reconfiguration of low-level settings in the network elements is not needed. All lead to saving a lot of resources, workforces and time.

C. Application perspective

IoT is placing new demands on network infrastructure due to the diverse application domain. With our proposed architecture, IoT applications can customize their own QoS requirements in terms of data acquisition and transmission. The designed path computation and QoS management functions compute a path with respect to the application QoS constraints which increases the user satisfaction, as well as, enabling of innovative application deployment.

Furthermore, using the queue policy to classify different applications based on their QoS needs and the operation criticality guarantees the quality of service for high priority demands when multiple demands fight for the available shared resources. Also, the collection of the current state of the network elements and being notified in case of changes or failure, these approaches help the SDN controller to be aware of the current network status and the occurred events such as link up/down or the node join/leave.

Network-state awareness and its involvement in the design of the routing computation algorithm decrease the possibility of link congestion and increase the network availability and throughput. Also, status-aware QoS-support resource allocation algorithm in sensing network fits the task and sensor data into application characteristic and needs. Additionally, the softwarization and centralization of the network services make the system to be in convergence with the changes. All of these approaches have an impact on the application satisfaction index (such as Quality of Experiences).

VI. Conclusion

The large number of IoT devices enable a wide variety of services in many different application domains, such as smart city, smart transport, smart home, and smart health. Each service and application depending on their goals and criticality may expect various QoS requirements from the IoT system in terms of data acquisition, transmission, and processing. There must be QoS approaches at every layer of the IoT architecture to ensure an acceptable level of performance especially for safety applications. We proposed a flexible and programmable control layer to provide customized and generic support services for the IoT applications. It is achieved by the integration of the core transport SDN into IoT architecture. We designed a status-aware and QoS-aware resource allocation framework across the IoT networking infrastructure. This framework has not been implemented in an end to end fashion due to the many underlying systems involved. Simulations must be planned in multiple phases. As a future work, we plan on simulating this framework in phases to account for the different technologies involved in the different layers of the IoT architecture. In our plan, the first step, QoS management in SDN-based network/Internet will be simulated considering the application QoS DBs as input. Afterwards, QoS management will be formularized for wireless network and broadband network (LTE). Finally, the end to end QoS calculation will be designed taking into account the underlying systems in the simulated environment. In another direction, we intend to model a status-aware and SLA-aware routing mechanism across the core transport software-defined network. Using different cost metrics for different application types, the model will find the less-delay and less-loss rate paths not only for delay-centric applications but also for bandwidth-centric applications.

References

[1] T. Kurakova, “Overview of the internet of things,” Proceedings of the Internet of things and its enablers (INTHITEN), 2013, pp. 82–94.

[2] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “A comparative study of lpwan technologies for large-scale iot deployment,” ICT express, vol. 5, no. 1, 2019, pp. 1–7.

[3] D. Chen and P. K. Varshney, “Qos support in wireless sensor networks: A survey.” in International conference on wireless networks, vol. 233, 2004, pp. 1–7.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” Future generation computer systems, vol. 29, no. 7, 2013, pp. 1645–1660.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” IEEE Internet of Things journal, vol. 1, no. 1, 2014, pp. 22–32.

[6] C. Bisdikian et al., “Building principles for a quality of information specification for sensor information,” in 2009 12th International Conference on Information Fusion. IEEE, 2009, pp. 1370–1377.

[7] C. Bisdikian, L. M. Kaplan, and M. B. Srivastava, “On the quality and value of information in sensor networks,” ACM Transactions on Sensor Networks (TOSN), vol. 9, no. 4, 2013, p. 48.

[8] I. Awan, M. Younas, and W. Naveed, “Modelling qos in iot applications,” in 2014 17th International Conference on Network-Based Information Systems. IEEE, 2014, pp. 99–105.

[9] R. Duan, X. Chen, and T. Xing, “A qos architecture for iot,” in 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing. IEEE, 2011, pp. 717–720.

[10] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, “Software defined wireless sensor networks application opportunities for efficient network management: A survey,” Computers & Electrical Engineering, vol. 66, 2018, pp. 274–287.

[11] A. Mirchev, “Survey of concepts for qos improvements via sdn,” Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), vol. 33, no. 1, 2015, pp. 31–39.

[12] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, “Software defined wireless networks (sdwn): Unbridling sdns,” in European workshop on software defined networking, 2012, pp. 1–6.

[13] H. Yang and Y. Kim, “Sdn-based distributed mobility management,” in 2016 International Conference on Information Networking (ICOIN). IEEE, 2016, pp. 337–342.

[14] T. Miyazaki et al., “A software defined wireless sensor network,” in 2014 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2014, pp. 847–852.

[15] Y. Jararweh et al., “Sdiot: a software defined based internet of things framework,” Journal of Ambient Intelligence and Humanized Computing, vol. 6, no. 4, 2015, pp. 453–461.

[16] S. K. Tayyaba, M. A. Shah, O. A. Khan, and A. W. Ahmed, “Software defined network (sdn) based internet of things (iot): A road ahead,” in Proceedings of the International Conference on Future Networks and Distributed Systems. ACM, 2017, p. 15.

[17] T. G. Nguyen et al., “Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks,” IEEE access, vol. 7, 2019, pp. 107 678–107 694.

[18] S. Al-Rubaye, E. Kadhum, Q. Ni, and A. Anpalagan, “Industrial internet of things driven by sdn platform for smart grid resiliency,” IEEE Internet of Things Journal, vol. 6, no. 1, 2017, pp. 267–277.

[19] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, “Efficient topology discovery in software defined networks,” in 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS). IEEE, 2014, pp. 1–8.

[20] O. S. Specification, “Open networking foundation,” Version ONF TS-015, vol. 1, no. 3, 2013, pp. 1–164.

[21] V. Mohan, Y. J. Reddy, and K. Kalpana, “Active and passive network measurements: a survey,” International Journal of Computer Science and Information Technologies, vol. 2, no. 4, 2011, pp. 1372–1385.

[22] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, “An intelligent traffic load prediction-based adaptive channel assignment algorithm in sdn-iot: A deep learning approach,” IEEE Internet of Things Journal, vol. 5, no. 6, 2018, pp. 5141–5154.

[23] M. Ojo, D. Adami, and S. Giordano, “A sdn-iot architecture with nfv implementation,” in 2016 IEEE Globecom Workshops (GC Wkshps). IEEE, 2016, pp. 1–6.