An Application-aware QoS Routing Algorithm for SDN-based IoT Networking

Guo-Cin Deng
Institute of Computer Science and Engineering
National Chiao Tung University
Hsinchu, Taiwan 300
ericdeng.cs04g@nctu.edu.tw

Kuochen Wang
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan 300
kwang@cs.nctu.edu.tw

***Abstract*—With numerous emerging internet of things (IoT) devices, they generate big data. The big data transmitted to the cloud or fog will consume massive network bandwidth. This may result in the IoT network easily encountering network congestion. Moreover, there are IoT applications that need to transfer multimedia data with multiple quality of service (QoS) requirements. A state-of-the-art, MINA, intends to meet multiple QoS requirements of IoT applications; however, it is unable to guarantee QoS requirements of high-priority IoT applications and it is also unable to adapt to the current network status. To conquer the above problems, we propose an *application-aware QoS routing algorithm* (AQRA) for SDN-based IoT networking to guarantee multiple QoS requirements of high-priority IoT applications and to adapt to the current network status for better routing paths. Evaluation results have shown that, the AQRA has better fitness ratios of QoS requirements compared to MINA while multiple QoS requirements of high-priority IoT applications are guaranteed. The AQRA improves the average end-to-end flow performance by 10.75%, 11.88% and 10.82% compared to MINA in terms of delay, jitter and packet loss rate, respectively. The AQRA improves the standard deviation of end-to-end flow performance by 14.37%, 17.95% and 14.28% compared to MINA in terms of delay, jitter and packet loss rate, respectively. In addition, the runtime of the AQRA is 38.56% shorter than that of MINA.**

Keywords—application-aware, internet of things, quality of service, routing, software-defined networking.

I. INTRODUCTION

The huge volumes of various multimedia data produced from IoT devices will consume massive network bandwidth. Different IoT applications may run simultaneously and share the same networking resources. The traditional IoT network architecture is not scalable and efficient enough to handle a large amount of IoT data. This may result in the IoT network to easily encountering network congestion.

The software-defined network (SDN) has emerged as an efficient network management technology. Its control functionalities are decoupled from network devices and are centralized into a controller. The controller thus embraces a global network view and network programmability. IoT networking can benefit from the centralized control of SDN, such as dynamic flow control and flexible network resource management.

IoT applications need to transfer multimedia data with multiple quality of services (QoS) requirements, such as delay, jitter, packet loss rate and bandwidth. In addition, there are mission critical applications, such as gas monitors, smoke sensors, disaster sensors, etc., that need to transfer emergency messages in IoT networks. If these kinds of messages can not be delivered in time due to network congestion, it may cause serious consequences [1]. Therefore, it is necessary that we provide QoS guarantees to assure these messages delivered in time. The integration of SDN and IoT networking is helpful to achieve better QoS management of IoT applications.

This paper proposed an *application-aware QoS routing algorithm* (AQRA) for SDN-based IoT networking. The contributions of this paper are as follows. (1) The AQRA can differ mission critical applications from non-mission critical applications by classifying flows into different priorities dynamically according to 3GPP LTE QCI [1], which are set by IoT application providers. (2) The AQRA can calculate a set of routing paths according to multiple QoS requirements (i.e., delay, jitter and packet loss rate) of each IoT application using simulated annealing (SA). To adapt to the current network status, the proposed AQRA can adaptively adjust weights in the cost function of SA. The available bandwidth of each path is considered when deciding the final routing path in order to achieve load balancing among paths. (3) The AQRA monitors the QoS of each flow. The AQRA guarantees QoS requirements of high-priority IoT applications by controlling the admission of low-priority IoT applications. Nevertheless, the starvation avoidance of low-priority IoT applications is taken into account.

II. RELATED WORK

We introduce some representative related work on QoS improvements in IoT networking. The related work can be classified into two different types, architecture improvement and algorithm improvement.

A. Architecture improvement

There are existing researches presenting SDN-based network architecture for IoT networking. In [2], the authors proposed an IoT architecture that integrates SDN and Object Management Group's data distribution service (DDS) middleware that enables agile and flexible network orchestration to address mobility, scalability and QoS issues. In [3], the authors proposed a software defined based IoT framework to

forward, store, and secure the produced data from the IoT by integrating the software defined network, software defined storage, and software defined security. In [4], the authors proposed an edge computing for the IoT architecture with the SDN-based core network to overcome the scalability problem of the traditional IoT architecture. They also proposed a hierarchical fog computing architecture in each fog node to provide flexible IoT services. In [5], the authors proposed a software-defined fog network architecture for IoT that combines SDN and fog computing to address scalability and real-time data delivery issues of the traditional IoT architecture. However, the related work mentioned above only proposed architecture design principles without implementation and evaluation.

B. *Algorithm improvement*

In MINA [6], the authors proposed an IoT SDN controller that monitors available resources and proposed a genetic algorithm (GA) based QoS-aware routing algorithm to meet multiple QoS requirements. However, this work has the following three problems: (1) it is unable to provide QoS guarantees for high-priority IoT applications. (2) Static weights used in its cost function cannot adapt to the current network status. (3) As more and more IoT devices accessing the network, load balancing is an issue that can be considered to reduce network delay; however, MINA did not consider it. In REAC (regressive admission control) [7], the authors proposed computing intelligence at the network edge for IoT networking to control traffic flows. With the REAC method, the edge router monitors the delay performance to admit flows to the network that guarantees good quality for high-priority flows. However, REAC only considered a single QoS requirement, delay. In addition, it did not consider the starvation problem of low-priority flows. In PFIM [8], the authors proposed a pre-emptive flow installation mechanism for IoT devices, which can learn the transmission intervals of periodic network flows and install the corresponding flow entry into an SDN switch before the arrival of packets. However, it also only considered a single QoS requirement, delay.

In summary, Fig. 1 is the classification of QoS improvement techniques in IoT networking. The QoS can be improved by using better algorithm or architecture. The proposed AQRA is classified into the algorithm improvement type. In addition, Table I shows the qualitative comparison of related work on QoS in IoT networking, including the proposed AQRA.

III. An Application-aware QoS Routing Algorithm

The proposed application-aware QoS routing algorithm (AQRA) for SDN-based IoT networking is detailed as follows. It consists of four parts: (1) SDN-based IoT network architecture, (2) traffic classification, (3) SA-based QoS routing, and (4) QoS-aware admission control.

A. *SDN-based IoT network architecture*

Fig 2 shows the SDN-based IoT network architecture. This architecture contains five layers: (1) *Application layer*: IoT applications or services belong to this layer. This layer connects to the network layer via Ethernet links. Each IoT application server can send messages to the SDN controller via northbound API (e.g., RESTful API). (2) *Network layer*: The network layer is the SDN-based core network, which consists of several SDN OpenFlow switches connected by Ethernet links. The SDN OpenFlow switches can communicate with the control layer by using the southbound API (e.g., OpenFlow protocol). This layer connects to the edge layer via Ethernet links. (3) *Edge layer*: This layer consists of several edge equipment (e.g., IoT gateways or access points). The edge equipment is OpenFlow-enabled so they can be controlled via the OpenFlow protocol by the SDN controller. This layer connects to the perception layer via wireless communication technologies, such as WiFi, ZigBee, etc. (4) *Perception layer*: IoT devices or sensor nodes belong to this layer. IoT devices forward/receive data to/from IoT applications by accessing the edge layer and the network layer. In addition, data forwarding is controlled by the control layer. (5) *Control layer*: The SDN controller is located in this layer. The SDN controller contains three modules: (5-1) *topology discovery module*: This module discovers all network elements in the data plane and builds network topology. (5-2) *network status monitoring module*: This module monitors and collects the network condition periodically. (5-3) *AQRA module*: The proposed AQRA is running as a module in the SDN controller. The AQRA can manage the behavior of elements in the network layer and the edge layer via southbound API, as well as receives the messages from the application layer via northbound API.

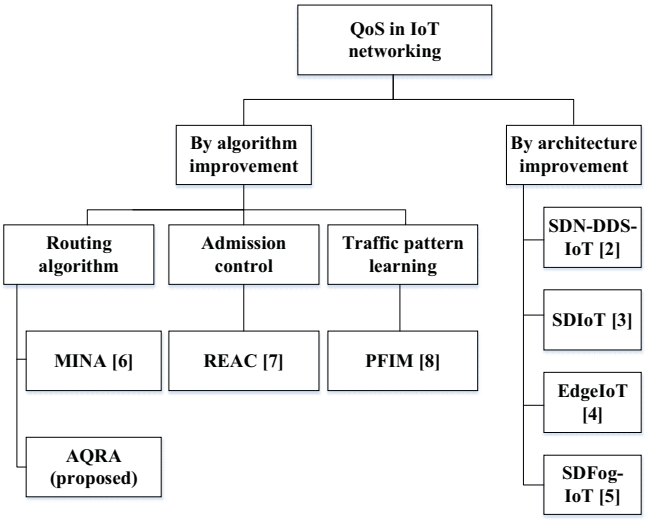

Fig. 1. Classification of QoS improvement techniques in IoT networking.

TABLE I. COMPARISON OF RELATED WORK ON QOS IN IOT NEETWORKING

Approach	Architecture	Algorithm	QoS metric
MINA [6]	SDN-based IoT	GA-based QoS routing with static weights	Delay, jitter or throughput
REAC [7]	IoT with edge computing	Admission control	Delay
PFIM [8]	SDN-based IoT	Traffic pattern learning	Delay
AQRA (proposed)	SDN-based IoT	Application-aware SA-based QoS routing with adaptive weights, load balancing, admission control and starvation avoidance	Delay, jitter, packet loss rate, and available bandwidth

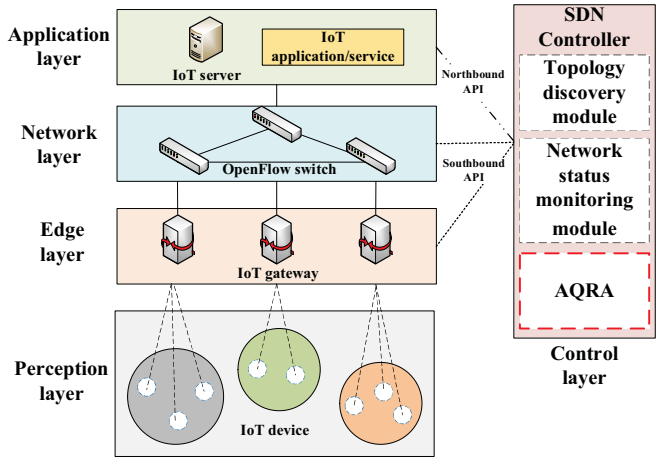

Fig 2. SDN-based IoT network architecture.

Fig. 3 shows the workflow of the AQRA. Each IoT application needs to send its *app_profile* to the AQRA. The *app_profile* contains the IP address of an IoT application server and the 3GPP LTE QoS Class Identifier (QCI) set by application providers. If an IoT device transmits data to an IoT application, the equipment in the edge layer will send a Packet_In message to the SDN controller when it receives the data flow. The AQRA will start to run after receiving the Packet_In message. There are three major steps in the AQRA: (1) traffic classification: the AQRA identifies and stores the *flow_info* of each flow according to the *app_profile*. The *flow_info* contains source/destination IP addresses, QoS requirements vector, priority, class, drop counter (*dc*) and drop probability (*dp*) of the flow. (2) SA-based QoS routing: the AQRA calculates a routing path for the flow and set up flow entries by sending Flow_Mod messages to the network layer and the edge layer. (3) QoS-aware admission control: the AQRA controls the admission of low-priority IoT applications by sending Flow_Mod messages to the edge layer.

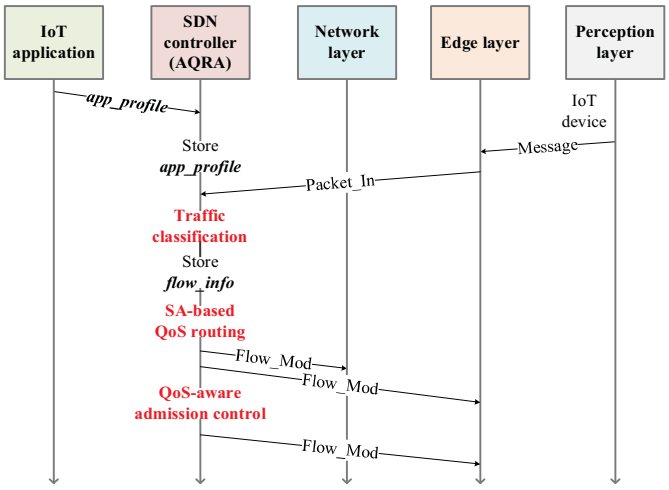

Fig. 3. The workflow of AQRA.

B. Traffic classification

In order to distinguish between high-priority applications and low-priority applications then providing QoS guarantee to high-priority applications, we propose a traffic classification mechanism to prioritize different flows and categorize different applications into different classes. Then we set source/destination IP addresses, QoS requirements vector, priority and class in the *flow_info* for each flow.

The QoS Class Identifier (QCI) [1] is a mechanism to ensure that traffic is allocated to appropriate QoS in the 3GPP Long Term Evolution (LTE). An IoT application provider can set the IP address of the IoT application server and the QCI value in the *app_profile* then sends them to the SDN controller. The AQRA obtains the QoS requirements vector and priority of each flow according to the QCI value set by the IoT application provider.

We categorize applications into three classes: high-priority class, medium-priority class and low-priority class, according to the resource type of QCI. High-priority class applications (priority = 1, resource type = Non-GBR) are those non real time mission critical services for delay sensitive data transfer. Medium-priority class applications (priority = 2 ~ 5, resource type = GBR) are those real time services with a stringent delay bound. Low-priority class applications (priority = 6 ~ 8, resource type = Non-GBR) are those non-mission critical services without a stringent delay bound. We provide QoS guarantee for applications belonging to the high-priority class. If all applications belonging to the high-priority class have been QoS guaranteed, then applications belonging to the medium-priority class are QoS guaranteed.

C. SA-based QoS routing

We define a QoS requirements vector $Q_i = < R_d, R_j, R_l >$, to represent delay, jitter and packet loss rate requirements of an application. The problem is to find path P for each flow, such that: $P_d \leq R_d$ and $P_j \leq R_j$ and $P_l \leq R_l$ where $< P_d, P_j, P_l >$ is a performance vector in which each element represents the delay, jitter and packet loss rate of a flow when using path P. This type of the problem is called the path selection problem. However, path selection with multiple constrains is an NP-complete problem [9]. Therefore, we use a heuristic algorithm to find an approximate optimal solution. The proposed AQRA is based on the simulated annealing (SA) algorithm, which is a well-known heuristic search algorithm to find an approximate global optimum without being stuck in local optimum based on a probability function. In contrast to the genetic algorithm (GA) used in MINA [6], the SA normally has faster runtime than the GA. Runtime of the GA increases exponentially with population size [10]. In order to meet the QoS requirements of time-sensitive IoT applications, we argue that the SA is more suitable to solve the path selection problem than the GA due to its shorter runtime.

The AQRA has following features: (1) Adaptive weights in the cost function is to adapt to the current network status to find better routing paths; (2) The search process of the SA is memory-less and therefore cannot avoid revisiting regions that are less likely to contain global minimum [11]. Therefore, we use a *recordPathList* to store history solutions and consider both a neighbor path and *recordPathList* when state transition occurs; (3) Load balancing among paths is achieved by selecting the path which has the maximum available bandwidth (*ABW*). Fig. 4 shows the flowchart of the SA-based QoS routing. We can divide the SA-based QoS routing into seven steps described as follows:

(1) Initialization: When the AQRA receives a Packet-In message, *flow_info* of the flow will be checked first to get the source/destination IP addresses and the QoS requirements vector. The AQRA computes an initial path *P* by using Dijkstra's algorithm, calculates the cost of initial path (C_p) using the cost function (equation (1)), set iteration count *t*, and initialize *recordPathList*.

(2) Calculate a neighbor path *N* by randomly selecting a node on the current path, and then construct a new shortest path to the destination.

(3) Select the lowest cost path *X* from neighbor path *N* and *recordPathList*.

(4) Accept the new path *X* according to the probability function (equation (4)). If the value of the probability function is larger than or equal to a random number whose value in the range [0, 1], path *X* will replace current path *P* and path *X* is appended to *recordPathList*.

(5) Step 2 to step 4 will iteratively run until iteration count *t* is reached.

(6) After the iteration is over, we select paths whose $C_p \leq 0$ from *recordPathList* to be candidate paths.

(7) Decide the final routing path: If the number of candidate paths is greater than one, we select a path *P* that has the maximum available bandwidth (ABW_P) from the candidate paths to be the final routing path. Otherwise, we select a path *P* that has the lowest C_p from *recordPathList* to be the final routing path.

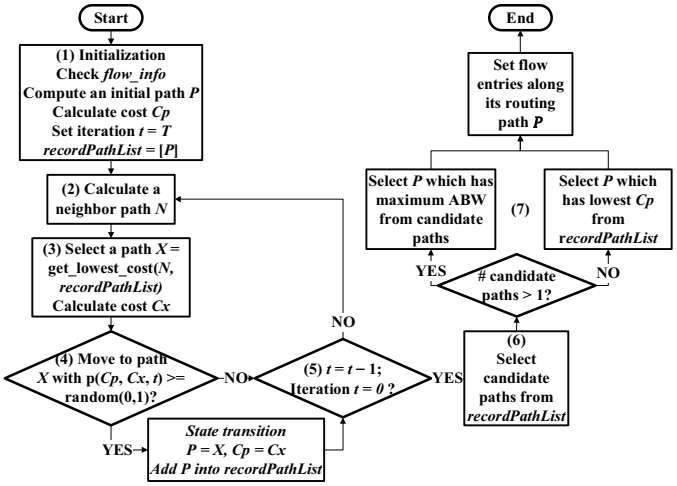

Fig. 4. Flowchart of the SA-based QoS routing.

In cost function (1), C_p is the cost of path *P*, the lower cost the better fit path. W_d, W_j and W_l are weights for each QoS requirement. Weights are adaptively adjusted by using equation (2). The weight will become larger if the corresponding QoS requirement could not meet in the past too many times. MR_d, MR_j and MR_l are miss rates for delay, jitter and loss rate requirements, as shown in equation (2). The miss rate MR_x is the rate of QoS requirement *x* that cannot be satisfied, as shown in equation (3), where *x* can be delay, jitter or packet loss rate.

$$C_p = W_d \frac{(P_d - R_d)}{R_d} + W_j \frac{(P_j - R_j)}{R_j} + W_l \frac{(P_l - R_l)}{R_l} \quad (1)$$

$$W_x = \frac{MR_x}{MR_d + M_j + MR_l} \quad (2)$$

$$MR_x = \frac{\#(flows\ cannot\ meet\ requirement\ x)}{\#(flows\ in\ histo\ )} \quad (3)$$

$$p(C_p, C_X, t) = \begin{cases} 1, & C_X < C_P \\ e^{\frac{-c|C_X - C_P|}{t}}, & C_X \geq C_P \end{cases} \quad (4)$$

Available bandwidth of a routing path *P* (ABW_P) [12] can be calculated using equation (5), where e_i is the i^{th} link in the routing path *P*, c_i is the capacity of e_i, b_i is the current bandwidth load on e_i, and a_i is the available bandwidth on e_i which can be calculated by equation (6). Therefore, the *ABW* of a routing path *P* is the minimum available bandwidth from links along with *P*.

$$ABW_P = \min_{e_i \in P} a_i \quad (5)$$

$$a_i = c_i - b_i \quad (6)$$

D. QoS-aware admission control

To guarantee QoS requirements of high-priority applications, we proposed a QoS-aware admission control mechanism. The QoS-aware admission control (AC) will be triggered when the QoS requirements of applications (flows) belonging to the high-priority class have not been satisfied. The basic idea of QoS-aware admission control is to drop low-priority or medium-priority flows that seize the network resource of high-priority flows until the QoS requirements can be guaranteed. However, dropping packets of low-priority flows too many times may result in the starvation problem, which means the low-priority flows have been denied service. To avoid the starvation problem, the AQRA records a drop counter (*dc*) and a drop probability (*dp*) for each flow in the *flow_info*. The *dc* is initialized to 0 by the AQRA when it receives the Packet_In message of the flow. The *dp* is calculated by using the following equation (7), where *dp* will decrease when *dc* increase.

$$dp(dc) = \begin{cases} 1, & dc = 0 \\ \frac{1}{dc}, & dc > 0 \end{cases} \quad (7)$$

The AQRA controls both existing flows and incoming flows at the same time. Fig. 5 shows the flowchart of the QoS-aware admission control for existing flows. We find a victim link (*vl*) which has the maximum link utilization (*lu*) in the routing path of the high-priority flows first. The link utilization is calculated using equation (8). It is decided by the port utilization (*pu*) of source and destination ports of the link. The port utilization is calculated using equation (9). Then we try to find a victim flow (*vf*) which belongs to the medium-priority class or low-priority class and has the lowest priority in the victim link. If there are multiple lowest priority flows, we choose the flow that has the highest drop probability (*dp*). If there are still multiple flows with the same priority and *dp*, we randomly select one flow from them to be the victim flow. We then drop the packets of the victim flow at the edge by sending Flow_Mod to the edge equipment and increasing the *dc* of the victim flow. The AQRA then checks whether the QoS requirements of the high-priority flows are satisfied or not after dropping the victim flow.

$$lu = max(port\ utilization_{src}, port\ utilization_{dst}) \quad (8)$$

$$pu = \frac{\text{current bitrate}}{\text{max bitrate}} \times 100\% \quad (9)$$

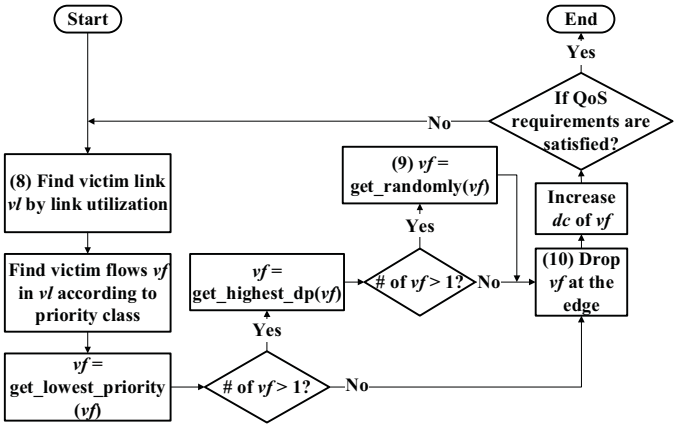

Fig. 5. Flowchart of the QoS-aware admission control for existing flows.

Fig. 6 shows the flowchart of the QoS-aware admission control for incoming flows. The AQRA calculates the routing path of an incoming flow and then checks whether the routing path of the incoming flow shares common links with high-priority flows. If yes, the AQRA checks the priority of the incoming flow. If it belongs to the high-priority class, it will still be admitted. If it belongs to the medium or low priority class, it will be dropped and *dc* is increased.

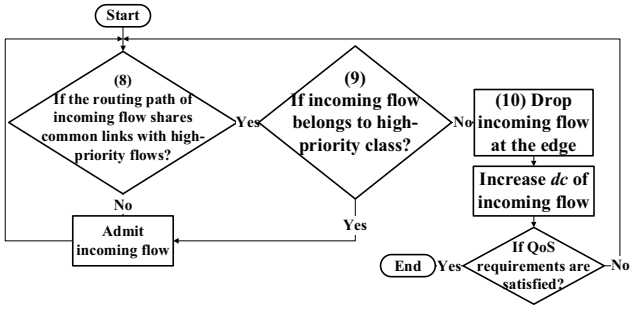

Fig. 6. Flowchart of the QoS-aware admission control for incoming flows.

IV. Evaluation

A. Simulation setup

Ryu [18] was used as the SDN controller and Mininet-WiFi [13] was used as the data plane emulator. We used the smart campus network topology [6] for evaluation. It includes three application servers (server1 provides mission critical delay-sensitive data transfer service and non-mission critical delay-insensitive data transfer service, server2 provides voice service and server3 provides video streaming service), three OpenFlow core switches, two OpenFlow edge switches, 15 OpenFlow-enabled access points and 45 end devices accessing the network via WiFi (IEEE 802.11n). We assigned each of the 45 end devices a service, randomly chosen from 10 mission critical delay-sensitive data transfer services, 11 non-mission critical delay-insensitive data transfer services, 15 voice services, and 9 video streaming services. Each service has its QoS requirements that are defined according to [1]. The required jitters for data transfer, voice and video streaming services are from [14], [15] and [16], respectively. We used the D-ITG (Distributed Internet Traffic Generator) [17] to generate test traffic and measure the performance of the network. The packet size of data transfer service is uniformly distributed in [100, 1000] bytes and inter-arrival time is uniformly distributed in [0.001, 0.1] seconds. Audio traffic and video streaming traffic are from real traffic traces [6]. The proposed AQRA is compared with MINA [6] in terms of fitness ratios of QoS requirements, average end-to-end flow performance, standard deviation of end-to-end flow performance and runtime. The fitness ratio is defined in equation (10) for metric x and equation (11) for three metrics all together, where x can be delay, jitter or packet loss rate.

$$\text{Fitness ratio}_x = \frac{\#\text{ flows that meet requirement of } x}{\#\text{ flows}} \quad (10)$$

$$\text{Fitness ratio}_{all} = \frac{\#\text{ flows that meet all QoS requirements}}{\#\text{ flows}} \quad (11)$$

B. Simulation results and discussion

Fig. 7 shows the fitness ratios of QoS requirements. MINA only considers a single QoS requirement for a specific service (e.g., data transfer service requires a low packet loss rate, voice service requires low delay and video streaming service requires low jitter). Therefore, its fitness ratio of QoS requirement is higher than that of AQRA (without history). However, the fitness ratio of AQRA (with history) outperforms that of MINA since AQRA (with history) considers history solutions and the path with maximum available bandwidth. In addition, AQRA (with history & AC) can guarantee the QoS requirements of mission critical data transfer service via AC.

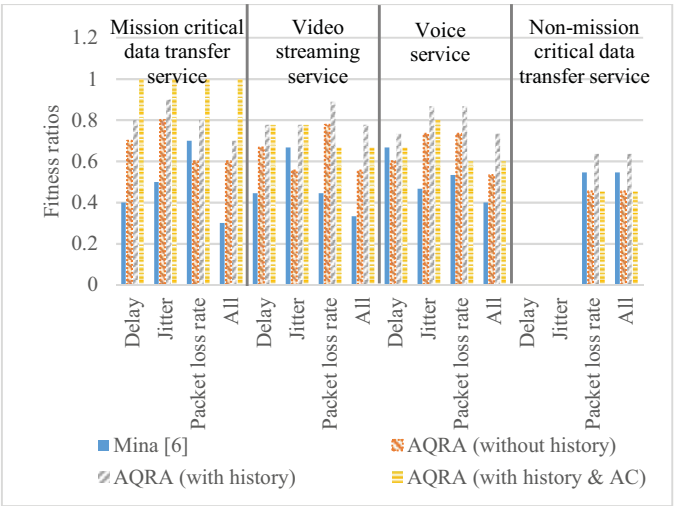

Fig. 7. Fitness ratios of QoS requirements.

Table II shows the average end-to-end flow performance of the four different algorithms. AQRA (with history) is 10.75%, 11.88% and 10.82% better than MINA in terms of delay, jitter and packet loss rate, respectively. This is because adaptive weights in the cost function and load balancing features enable AQRA to find better solutions. Table III shows the standard deviation of end-to-end flow performance. The standard deviation of AQRA (with history) is 14.37%, 17.95% and 14.28% better than MINA in terms of delay, jitter and packet loss rate, respectively. This means that the AQRA can allocate network resources in a more balancing manner. Compared to MINA, the runtimes of AQRA (without history), AQRA (with history) and AQRA (with history & AC) decreased by 55.38%, 46.29% and 38.56%, respectively, as shown in Fig. 8.

TABLE II. AVERAGE END-TO-END FLOW PERFORMANCE

Average end-to-end flow performance	Delay (ms)	Jitter (ms)	Packet loss rate (%)
MINA [6]	99.82	33.97	0.057
AQRA (without history)	92.28	31.25	0.053
AQRA (with history)	89.10	29.93	0.051
AQRA (with history & AC)	89.22	30.15	0.052

TABLE III. STANDARD DEVIATION OF END-TO-END FLOW PERFORMANCE

Standard deviation of end-to-end flow performance	Delay (ms)	Jitter (ms)	Packet loss rate (%)
MINA [6]	39.27	18.98	0.056
AQRA (without history)	38.83	18.75	0.054
AQRA (with history)	33.74	15.59	0.048
AQRA (with history & AC)	33.63	15.57	0.048

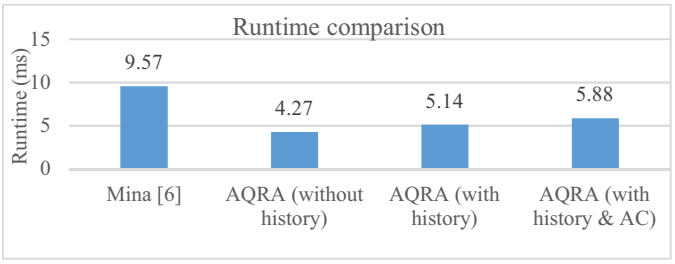

Fig. 8. Runtime comparison.

V. CONCLUSION

In this paper, we have presented an *application-aware QoS routing algorithm* (AQRA). The AQRA find multiple best fit paths that meet QoS requirements based on the SA with adaptive weights in the cost function to adapt to the current network status for better routing paths. It then selects the final routing path that has the maximum available bandwidth while achieving load balancing. The AQRA classifies flows into different priorities according to 3GPP LTE QCI and it guarantees multiple QoS requirements of high-priority IoT applications by controlling the admission of low-priority IoT applications with starvation avoidance. Evaluation results have shown that, the AQRA (with history & AC) has better fitness ratios of QoS requirements compared to MINA, and multiple QoS requirements of mission critical applications can be guaranteed. The AQRA (with history) improves the average end-to-end flow performance by 10.75%, 11.88% and 10.82% compared to MINA, in terms of delay, jitter and packet loss rate, respectively. The AQRA (with history) improves the standard deviation of end-to-end flow performance by 14.37%, 17.95% and 14.28% compared to MINA, in terms of delay, jitter and packet loss rate, respectively. In addition, the runtime of the AQRA (with history & AC) is 38.56% shorter than that of MINA.

ACKNOWLEDGMENT

The support by the Ministry of Science and Technology under Grants MOST 105-2221-E-009-091-MY3 and MOST 106-2622-8-009-017 is gratefully acknowledged.

REFERENCES

[1] H.-C. Jang, C.-W. Huang and F.-K. Yeh, "Design a bandwidth allocation framework for SDN based smart home," in *Proc. of IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1-6, Oct. 2016.

[2] A. Hakiri, P. Berthou, A. Gokhale and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 48-54, Sep. 2015.

[3] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk and A. Rindos, "SDIoT: A software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453-461, Jun. 2015.

[4] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22-29, Dec. 2016.

[5] S. Tomovic, K. Yoshigoe, I. Maljevic and I. Radusinovic, "Software-defined fog network architecture for IoT," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181-196, Oct. 2016.

[6] Z. Qin, G. Denker, C. Giannelli, P. Bellavista and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Proc. of IEEE Network Operations and Management Symposium (NOMS)*, Krakow, pp. 1-9, May 2014.

[7] M. Jutila, "An adaptive edge router enabling internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1061-1069, Dec. 2016.

[8] P. Bull, R. Austin and M. Sharma, "Pre-emptive flow installation for internet of things devices within software defined networks," in *Proc. of 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 124-130, Aug. 2015.

[9] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sep. 1996.

[10] Adewole A.P., Otubamowo K., and Egunjobi T.O., "A comparative study of simulated annealing and genetic algorithm for solving the travelling salesman problem," *International Journal of Applied Information Systems*, vol. 4, no. 4, pp. 6–12, Oct. 2012.

[11] S. Sun, F. Zhuge, J. Rosenberg, R. M. Steiner, G. D. Rubin, and S. Napel, "Learning-enhanced simulated annealing: method, evaluation, and application to lung nodule registration," *Applied Intelligence*, vol. 28, no. 1, pp. 83–99, Jan. 2007.

[12] P. C. A. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár, "Challenges and solution for measuring available bandwidth in software defined networks," *Computer Communications*, vol. 99, pp. 48–61, Feb. 2017.

[13] R. R. Fontes et al. "Mininet-WiFi: Emulating software-defined wireless networks." in *Proc. of IEEE 11th International Conference on Network and Service Management (CNSM)*, Nov. 2015.

[14] M. Marchese, *QoS over heterogeneous networks*, New York: Wiley, Apr. 2007.

[15] M. Karam and F. Tobagi, "Analysis of delay and delay jitter of voice traffic in the Internet," *Computer Networks*, vol. 40, no. 6, pp. 711-726, Dec. 2002.

[16] Video over IP network performance analysis, *White Paper*, *Certus Digital, Inc.,* 2017.

[17] A. Botta, A. Dainotti and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, iss. 15, pp 3531-3547, Oct. 2012.

[18] Ryu OpenFlow controller [online], Available: *https://osrg.github.io/ryu*.