This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

SDN-based Data Transfer Security for Internet of Things

Yanbing Liu, Yao Kuang, Yunpeng Xiao, and Guangxia Xu

***Abstract*—The exponential growth of devices connected to the network has resulted in the development of new Internet of Things (IoT) applications and on-line services, which may have diverse and dynamic requirements on received quality. Although, the emerging Software-Defined Networking (SDN) approach can be leveraged for the IoT environment, to dynamically achieve differentiated quality levels for different IoT tasks in very heterogeneous wireless networking scenarios, the open interfaces in SDN introduces new network attacks, which may make SDN-based IoT malfunctioned. The challenges lies in securely using SDN for IoT systems. To address this challenge, we design a SDN-based data transfer security model Middlebox-Guard (M-G). M-G aims at reducing network latency, and properly manage dataflow to ensure the network run safely. First, according to different security policies, middleboxes related to the defined secure policies, are placed at the most appropriate locations, using dataflow abstraction and a heuristic algorithm. Next, to avoid any middlebox becoming a hot-spot, an offline Integer Linear Program (ILP) pruning algorithm is proposed in M-G, to tackle switch volume constraints. In addition, an online Linear Program (LP) formulation is come up to handle load balance. Finally, secure mechanisms are proposed to handle different attacks. And network routing is solved flexibly, through dataflow management protocol, which are formulated via combining tunnels and tags. Experimental results demonstrate that this model can improve security performance and manage dataflow effectively in SDN-based IoT system.**

***Index Terms*—Internet of Things; Software-Defined Networking; Middlebox; Dataflow Management; Security.**

I. Introduction

THE continued evolution of new services and the growth of the information circulating the Internet, has led to the origin of ideas, concepts and paradigms such as the Internet of Things (IoT)[1]. However, traditional network infrastructure, which need high-level network policies and configuring protocols, are inefficient and have significant limitations to support the high level of scalability, high amount of traffic and mobility. Software-Defined Networking(SDN)[2] decouples the traditional closed network into data plane, control plane and application plane, which enables logically centralized control and management of the whole network. With this new design principle, the network could behave more flexibly and can easily adapt to the needs of different organizations. Besides, the centralized architecture allows important information to be collected from the network and in turn used to improve and adapt their policies dynamically. Thus, as shown in Figure 1, a programmable, flexible, flow-centric SDN-based IoT architecture is favorable. Although, open interfaces in SDN have simplified the design of secure applications in large and complex IoT, they are vulnerable to new network attacks [3]-[4], and this vulnerability inevitably reduces security in SDN-based IoT architecture. In IoT, the dataflow has to go through

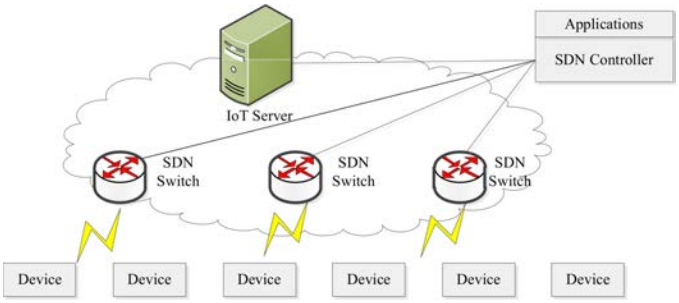

Fig. 1. SDN-based IoT architecture.

several processes before all required tasks are finished. Thus, proper handling of data flows in each device, is important for stable and secure network operation. Recent studies about the use of middleboxe and SDN[5]-[9] fall into three categories, including software realized middlebox, service chaining problems and integrating traditional middlebox into SDN networks.

The first challenge is the dynamic of dataflow in IoT.The number of users and dataflow volume in IoT vary over time. However, most existing dataflow control techniques assume a stable network. Therefore, such techniques can not actively consider network security. If a large number of data streams arrive simultaneously, the entire IoT network may become paralyzed. Thus, when developing dataflow control strategies, considering dataflow streams dynamics can improve the security and stability of the entire network.

The second challenge is the deployment of middleboxes in IoT.In traditional networks, the middlebox is deployed

Manuscript received XXXXXXXXXXXXX. This work has been supported by the National Science Foundation of China (Grant No.61772098& 61772099), Chongqing Innovative Team Fund for College Development Project (Grant No.KJTD201310), Chongqing Youth Innovative Talent Project (Grant No.cstc2013kjrc-qnrc40004), Science and Technology Research Program of the Chongqing Municipal Education Committee (No.KJ1500425), WenFeng and Foundation of CQUPT(No.WF201403) and Chongqing Graduate Research And Innovation Project(No.CYS14146).

Yanbing Liu is with Chongqing Engineering Laboratory of Internet and Information Security Chongqing University of Posts and Telecommunications, Chongqing, 400065 China (e-mail: liuyb@cqupt.edu.cn).

Yao Kuang is with Chongqing Engineering Laboratory of Internet and Information Security Chongqing University of Posts and Telecommunications, Chongqing, 400065 China (e-mail: 13618340827@163.com).

Yunpeng Xiao is with Chongqing Engineering Laboratory of Internet and Information Security Chongqing University of Posts and Telecommunications, Chongqing, 400065 China (e-mail: xiaoyp@cqupt.edu.cn).

Guangxia Xu is with the School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: xugx@cqupt.edu.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

in particular position. And recent studies have broken the restrictions, and enabled the middleboxes work collaboratively under the control of control plane. However, network latency is an important measure of overall IoT network performance. In addition, different security policies may have different demands on the order of middlebox sequences. So deploying the middleboxes at positions with the lowest network latency, becomes a new break point of middlebox deployment.

The last challenge is the management of dataflow streams in IoT.Due to the lack of a valid agreement to confirm the state of dataflow streams after going through middleboxes. The traditional L2/L3 dataflow forwarding rules may lead to inefficient use of available Ternary Content Addressable Memory (TCAM). When several middleboxes process one dataflow packet, the packet may be forwarded incorrectly. Therefore, to optimize or propose new rules in IoT to determine the status of data packets, and to develop correct forwarding decisions, become an important issue for dataflow management.

With the combination of dataflow monitoring mechanisms and dataflow rules,data transfer security will become more comprehensive and more considerate. In this paper, we focus on dataflow dynamics, middlebox deployment, and dataflow management, and propose a SDN-based data transfer security model in IoT based on middlebox, referred to as M-G. In this model, middlebox deployment location is selected first. Next, the dataflow in the entire network are controlled to ensure that the network can run normally. Finally, dataflow rules are determined in the way that related policies can be implemented effectively. Collaboration among traffic tunnels, the control plane, and the data plane is required to achieve the scalability and availability of M-G. Experiment results demonstrate that M-G can help manage dataflow in middleboxes properly, and improve the security and stability of the entire IoT network.

This paper makes the following contributions.

- A middlebox deployment position selection algorithm is proposed. The entire IoT network is viewed as an undirected symmetric graph with weighted edges. The greatest weight of middlebox sequence that respond to the defined secure policies are selected, and an effective deployment position is determined for this sequence.
- Two dataflow control algorithm is designed. Here, the NP-hard load balance problem is divided into two parts. First,the sub-sequences which do not exceed the switch volume constraints and satisfy the coverage requirements of logical chains are selected from the secure policy defined middlebox sequences. Next, to address the restriction on middlebox, the selected sub-sequences has been located to particular physical sequences where the allocation satisfy the restriction of middlebox.
- Secure mechanisms and a protocol for dataflow routing is presented. The secure mechanisms are proposed to defense spoofing and flooding attacks. The protocol is based on the combination of tags and tunnels. With the help of this protocol, the state of a packet in IoT is observed and then used, to determine the next hop of this packet.

This paper is organized as follows. The first section introduces background information. The second section includes the related works. The discussed problems are described in the third section. The proposed model is detailed in the fourth section. Experimental results are presented in Section V, and conclusions are given in Section VI.

II. Related work

As mentioned previously, the management of devices and network resources in IoT, is a big challenge, due to the different requests for different user-defined tasks. Recent studies into the use of middlebox in SDN-based IoT have focused on software-realized middleboxes, service chaining problems and integrating traditional middlebox into SDN network.

Software-realized middleboxes are deployed as an application on top of the IoT architecture to achieve different user-defined tasks. A software architecture for the development of flexible and configurable security applications has been proposed [10]. Joseph et al. designed a Layer-2 solution in[11] to route traffic through middleboxes. Flowstream[12] envisions virtual middleboxes with an OpenFlow[13] frontend for routing. SoftFlow[14] deploys middlebox applications using OpenvSwitch[15]. However, these studies and methods do not consider the capacity constraints of SDN switches and middleboxes , which can influence network performance.

Studies that focus on service chaining problems in SDN-based IoT address limitations associated with middleboxes in traditional networks, i.e., middleboxes are placed at a fixed position, and making middleboxes works collaboratively under the control of control plane. Thus, IoT network vulnerabilities and latency are reduced. The authors of [16] proposed a method to select the most appropriate middlebox deployment positions, thus the performance of middlebox has been optimized. Middleboxes deployment problem in SDN was discussed in [17]-[20], where the related performance evaluation methods were discussed as well.The limitations, requirements and benefits of middlebox placement issues were described in[21]. A previously proposed method in [22] considered the chaining problem as a binary integer program problem, and solved it through a heuristic algorithm. The work more closely related to M-G is SIMPLE[23], which solves the placement problem via enumerating all possible positions. However, SIMPLE is time-consuming and inefficient.

Several studies have addressed integrating traditional middleboxes into SDN networks. The API interfaces have been extended to make the middleboxes connect with SDN controller [24]. Slick proposed a centralized controller in [25], which can install and move functions to middleboxes, through middlebox management methods in SDN. In [26] OpenFlow protocols were used to control route, and ensure network volume to pass particular points. The authors of [27] compared four middlebox placement algorithms. FlowTags[6] integrated middlebox into SDN with the fewest modifications and follows the original SDN binding policies. The solution in [28] constructs a SDN firewall using flow tables and firewall policies. Also, data transfer protocol among middleboxes and middlebox management techniques in entire network were discussed in [29]-[32].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

III. PROBLEM FORMALIZATION

A. Related Definitions

For convenience, the used symbols are shown in table1.

The proposed framework attempts to solve the following problems: 1) How to place middleboxes such as firewall intrusion detection systems (IDS) into SDN-based IoT system. 2) How to guarantee that the flow streams in middleboxes and switches do not exceed their capacity limits, in order to facilitate stable IoT network operation. Several definitions are given below as the basis of algorithm design and functional combination scheme.e basis of algorithm design and functional combination scheme.

Definition 1. G(N,A): The entire IoT network can be considered as an undirected graph G(N,A), where N is the set of switches and A is the set of links.

Definition 2. Processing policy: The secure policy is best expressed via a dataflow abstraction. The operator who operate the network annotate each policy (e.g. <External, Web>) with its ingress and egress locations and IP prefixes. For example, a external web traffic may be specified by a traffic filter such as:<src=internal prefix, dst=external prefixes,srcport=*,dstport=80,proto=TCP>. Here, $PoChain_c$ denotes the required middlebox policy chain for this class(e.g. Firewall-IDS)

Definition 3. Successor nodes SCS(i): for node i and j, if $a_{ij} \in A$, node j is the successor node of node i.

Definition 4. Dependency degree: two middleboxes are dependent if they appear consecutively in a policy chain, and the dependency degree is determined based on the amount of traffic with such middlebox chain requirement.

Definition 5. Given two different paths $p_u^{ri} \neq p_v^{ri}$ from an origin node r to a node i, p_u^{ri} FSD dominates p_v^{ri} if the inverse CDF of the link travel time distribution at λ confidence level satisfies $\phi^{-1}_{T_u^{ri}(yr)}(\lambda) < \phi^{-1}_{T_v^{ri}(yr)}(\lambda)$, $\forall \lambda \in (0,1)$.

Definition 6. Nondominate road DL; A path considered nondominated if it can not be FSD dominated by any other path.

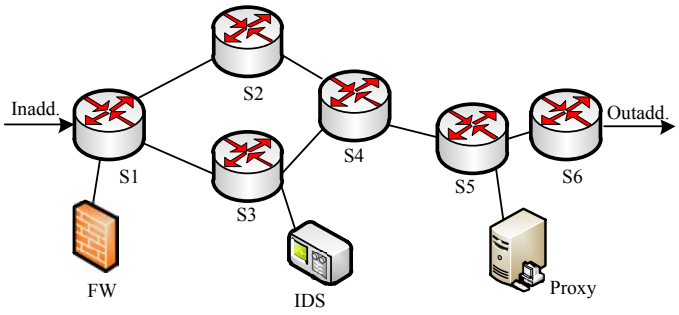

Fig. 2. Example of the related definitions.

To clearly explain above definitions, we use Figure 2 as an example to illustrate the meaning of those terms.

The middlebox chain (inadd.-firewall-IDS-Proxy-outadd.) is the PoChain the operator denoted for this topology. $PhSeq_c$ (S1,S2,S3,S4,S5,S6)is the positions that middleboxes may be placed. $PhSeq_{c,q}$ (S1,S3,S5) is the sub-sequence of $PhSeq_c$, that is chose to place middlebox. $I_{c,q}$ indicate whether a link

TABLE I
M-G FRAMEWORK SYMBOLS AND MEANINGS

Symbol	Meaning	Symbol	Meaning
$PoChain_c$	A secure policy class	$\phi(\cdot)$	Cumulative distribution function (CDF) of random variable X
$PhSeq_c$	The set of all physical sequences for $PoChain_c$	$\phi^{-1}(\cdot)$	Inverse CDF of random variable X
Δ	Length of discrete time interval	$E(\cdot)$	Mean of a random variable
$PhSeq_{c,q}$	One instance from $PhSeq_c$	$Route_{c,q}$	The switch-level route for $PhSeq_{c,q}$
M_j	A middlebox	$Rules_{k,c,q}$	The number of rules that required on switch S_k to route traffic through $Route_{c,q}$
S_k	A SDN switch	$I_{c,q}$	A binary indicator variales to denote whether a particular physical sequence has been chosen
$a_{i,j}$	A link connecting node i and node j	Cov	A target coverage level
$q_{u,v}$	The delay value in $a_{u,v}$	TCAM	The available space in SDN switch
$\oplus$	Path concatenation operator $p_u^{rw} = p_u^{ri} \oplus p^{iw}$ means that path p_u^{rw} goes through subpaths p_u^{ri} and p^{iw}.	$Ftprt_{c,j}$	The per-packet processing cost for a packet belonging to class c at the middlebox M_j
MaxMbO	The maximum occurrences across all middle boxes	$f_{c,q}$	The fraction of traffic for $PoChain_c$ that is assigned to each pruned $PhSeq_{c,q}$
V_c	The traffic volume in each middlebox	$MbUsed_j$	The number of chosen sequences in which a middlebox occurs
$ProcCap_j$	Per-class ftprt across all physical sequences	$Load_j$	The load on each middlebox
MMbLd	The maximum middlebox load across the network	λ	Confidence Level $\lambda \in (0,1)$
$T_u^{rs}(y_r)$	The travel time for a journey starting at departure time y_r and going through path p^{rs}		

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

has been chosen, in figure 2 $I_{1,2}$ is 0, and $I_{1,3}$ is 1. In $PhSeq_{c,q}$ (S1,S3,S5) the number of $MbUsed_j$ is 3, because there are three middleboxes in t.he chose sequence. And MaxMbO is the maximum number of all $MbUsed_j$ in the entire network, here it is 3, because there is only one sequence. $Frprt_{c,j}$ is the cost that a middlebox need to processing a packet. $ProcCap_j$ is the Ftprt across sequence S1-S3-S5.

B. Problem

Due to the reasons that network latency is important to measure network performance, and properly management of dataflow is the key to ensure network security. To reduce the network latency, the used middleboxes need to be placed at the locations, where the communication link is the shortest. To ensure the entire network can run stably and securely, the dataflow should be properly managed. The following problems will be investigated in this paper.

- How to find the most appropriate position for middlebox? Through dataflow abstraction, we annotate each policy (e.g. <External, Web>) with its ingress and egress locations and IP prefixes. And the most appropriate middlebox deployment location $PhSeq_{c,q}$ is selected by our heuristic algorithm.
- How to ensure that dataflow in the entire IoT network do not exceed switch and middlebox capacity, to avoid any single piece of equipment from becoming a hotspot? First, switch constraints is tackled. Here, MaxMbO, the maximum occurrences of middleboxes in a physic sequence $PhSeq_{c,q}$ is minimized to obtain the most appropriate route. Next, network loads is balanced. We run linear program formulation in the selected route to get $Load_j$, the loads on each middlebox, and the maximum loads on middlebox across the entire IoT network, and minimize it, to make the entire IoT network load balancing. As a result, the above problem can be solved, by obtaining the minimized MaxMbO and MMbLd values.
- How to ensure the state of each data packet, for correct policy implementation? Through data tracking, we use tunnels and tags to generate related flow rules. The tunnels indicate the tunnel to next switch, the tags includes the status of each dataflow in every switch. Thus, we can route the IoT network flexibly and effectively.

IV. FRAMEWORK

A. Framework Detail

To solve the above problems, the proposed model is based on network topologies, policy requirements, and related middleboxes. First, routes are selected based on the security policies, and effective middlebox deployment positions on each route are selected. Next, the amount of traffic on each line is limited, such that the traffic does not exceed switch and middleboxes capacities. Finally, SwitchTunnel is selected according to network size, for each physical sequence PhSeqc,q add ProcState to a packet header. The next step is determined according to the labels. The model framework is shown in Figure 3.

The dataflow streams of the entire IoT network are shown in Figure 4, In M-G middleboxes are connected to SDN switches. The controller route dataflow streams in SDN switches using OpenFlow(blue dotted lines in Figure 4). The most appropriate middlebox position is selected by the controller using a placement selection algorithm. We use an ILP pruning algorithm to control the volume of traffic in each route, and use an LP formulation to balance loads across middleboxes.

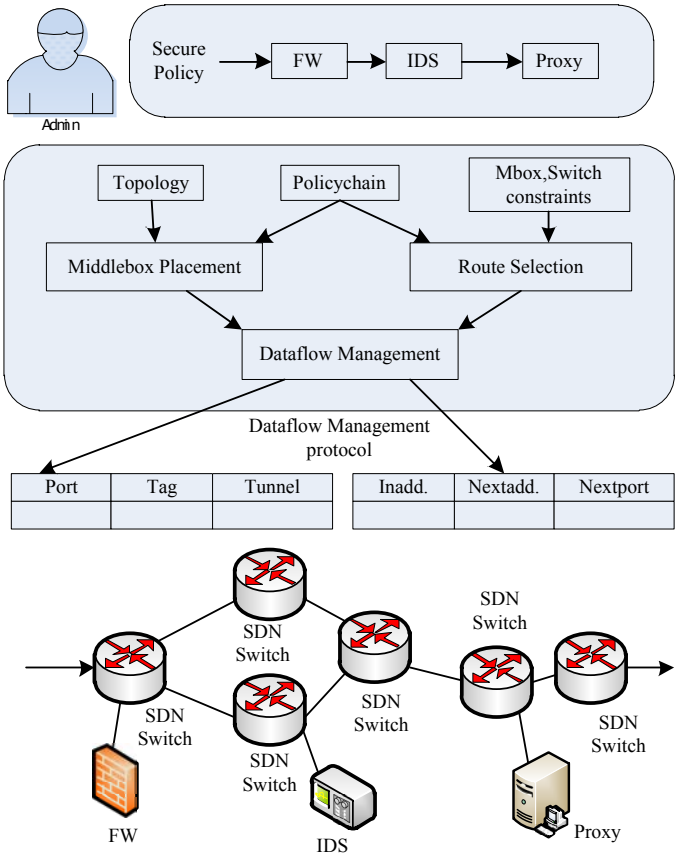

Fig. 5. Overview of M-G approach for using SDN to manage middlebox deployments and dataflow.

B. Algorithm

Figure 5 gives an overview of the M-G architecture , showing the inputs needed for various components, the interactions between them, and the interfaces to the data plane. Note that M-G only configured SDN-enabled switches, and did not extend middlebox to support new SDN-like capabilities. Each middlebox is directly connect to a SDN-enabled switch.

1) Middlebox Placement: We want to position middleboxes, where the IoT network delay is the shortest. To address this challenge, the entire IoT network is considered an undirected graph G(N,A).The IoT network delay at each link $a_{i,j}$ is $q_{u,v}$, $q_{u,v} = q_{v,u} = 1$. Different $PoChain_c$ values correspond to different middlebox sequences. All possible positions are expressed as $PhSeq_c = \|A\|$, and the final result finds $PhSeq_{c,q}$ to minimize the total network delay $\sum q_{u,v}$.

A heuristic algorithm is proposed to solve the middlebox placement problem in IoT. First, the middleboxes with the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

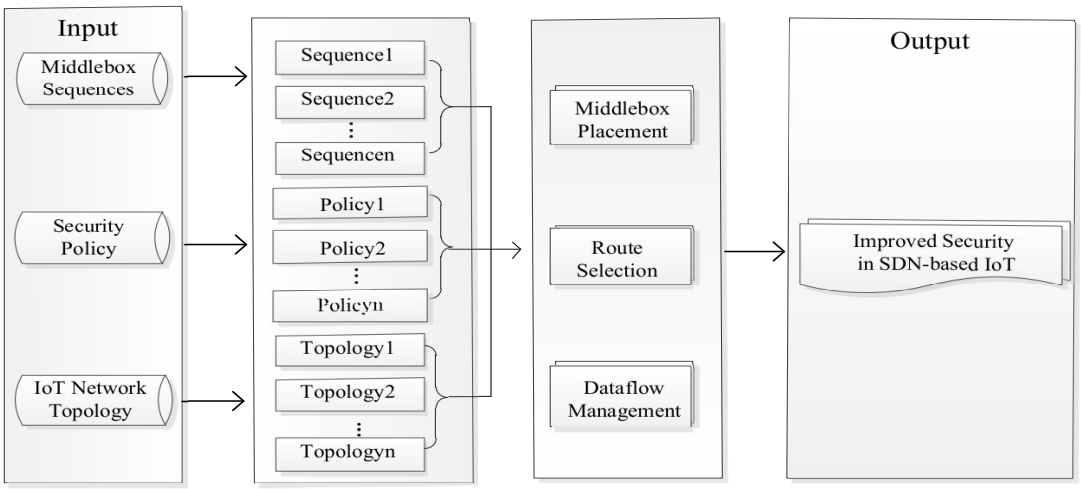

Fig. 3. Framework detail

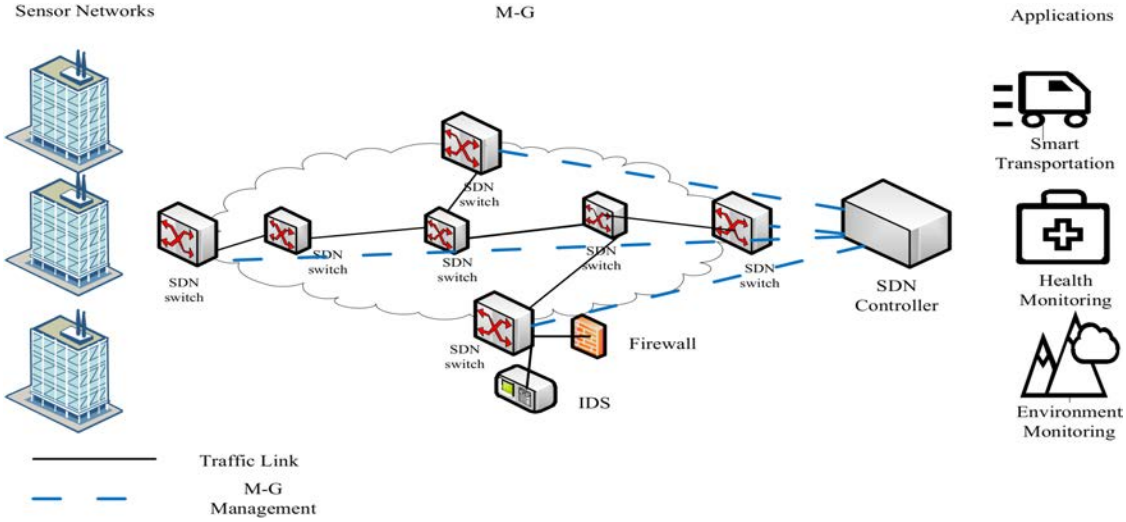

Fig. 4. Dataflow streams.

highest dependency degree are found using a greedy approach, and a middlebox dependency graph is formed between all middleboxes. Next, we find the most appropriate position for the chosen middlebox and remove it from the dependency graph.

Let $r \in N$ and $s \in N$ represent the origin and destination nodes, respectively. p^{rs} is a path from the origin to the destination, and $T^{rs}(y_r)$ denotes the time required to traverse p^{rs} with departure time y_r. $T^{rs}(y_r)$ is expressed as follows.

$$T^{rs}(y_r) = \sum_{a_{ij} \in A} T_{ij}(Y_j)\delta_{ij}^{rs} \tag{1}$$

where Y_j is the arrival time at node i, and δ_{ij}^{rs} is the path-link incidence variable; $\delta_{ij}^{rs} = 1$ indicates that link a_{ij} is on path p^{rs}, if link a_{ij} is not on path p^{rs}, $\delta_{ij}^{rs} = 0$. Since network status differs over time, we model travel speed as a normal distribution. Let V_{ij}^{n} be the travel speed distribution of link a_{ij} during time interval (Δ_{n-1}, Δ_n). Suppose the dataflow enters link a_{ij} at time instance $y_i \in (\Delta_{n-1}, \Delta_n)$. Then, the travel distance at time instance $y_a \in (\Delta_{m-1}, \Delta_m)$ can be calculated as follows.

$$D^{y_i}(y_a) = V_{ij}^{n}(\Delta_n - y_i) + V_{ij}^{n+1}\Delta + ... + V_{ij}^{m-1}\Delta + V_{ij}^{m}(y_a - \Delta_{m-1}) \tag{2}$$

As travel speeds are random variables, so is travel distance D_{yi}^{ya} .

Let d_{ij} be the length of link a_{ij}. The probability that the dataflow has arrived at head node i before time instance y_a can be expressed as follows.

$$\lambda = Pr(D_{y_i}^{y_a} \geq d_{ij}) \tag{3}$$

Let $\phi(\cdot)$ and $\phi^{-1}(\cdot)$ be the CDF and inverse CDF of a random variable, respectively. The arrival probability λ can be expressed in terms of the CDF of the arrival time distribution as follows.

$$\lambda = \Phi_{Y_j(y_i)}(y_a) \tag{4}$$

where $Y_j(y_i)$ is the arrival time distribution of the dataflow arriving at head node i. Based on(4) , the inverse CDF of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

the arrival time distribution at the λ confidence level can be calculated as follows.

$$y_a = \Phi^{-1}_{Y_j(y_i)}(\lambda) \tag{5}$$

$\Phi^{-1}_{Y_j(y_i)}(\lambda)$ can be calculated by (2)-(5), thus, link travel time can be expressed as follows.

$$T_{ij}(y_i) = Y_j(y_i) - y_i \tag{6}$$

The shortest path can be calculated based on the link travel time. First a heuristic evaluation function is calculated to determine the level for path p_u^{ri} as follows.

$$F(p_u^{ri}) = E(T_u^{ri}(y_r)) + h(i) \tag{7}$$

$$\text{s.t } F(p_u^{rj}) = E(T_u^{rj}(y_r))+h(j) \geq F(p_u^{ri}) = E(T_u^{ri}(y_r))+h(i) \tag{8}$$

where h(i) is the mean path travel time from node i to destination s. Equation(8) indicates that $F(p_u^{ri})$ should increase monotonically with path extension.

Let $P^{ri} = \{p_1^{ri}, ..., p_n^{ri}\}$ be a set of nondominated paths from the origin r to node i. All nondominated paths are maintained in a priority queue that is ordered based on $F(p_u^{ri})$. In each iteration, the label p_u^{ri} with minimum $F(p_u^{ri})$ is selected. An acyclic temporary path $p_u^{ri} = p_u^{ri} \oplus a_{ij}$ is then constructed by extending the selected path p_u^{ri} to each successor node $\forall j \in SCS(i)$. The temporary path p_u^{ri} is inserted to p^{rj} if p_u^{ri} is a DL path to p^{rj} If p_u^{ri} dominate paths in p^{rj}. Then these dominated paths are removed. This iteration continues until the destination is reached or the queue becomes empty.

Algorithm 1 PLACEMENT SELECTION ALGORITHM

Input: original and destination nodes and departure time y_r;
Output: The least expected time path
Step1 Initialization
Create a path p_u^{rr} from original to itself
Set arrival time distribution $Y_r(y_r)$ = 0 for the path p_u^{rr}
Calculate $h(r)$ and $F(p_u^{rr})$
Set $P^{rr} = \{p_u^{rr}\}$ and $SE = \{p_u^{rr}\}$
Step2 Path selection
If $SE = \varphi$,stop; otherwise, continue
Select p_u^{ri} at the top of SE and remove p_u^{ri} from SE
If $i = s$, stop ;otherwise, continue
Step3 Path extension
For every successor node $j \in SCS(i)$
If $j \notin p_u^{ri}$,continue; otherwise, scan next successor node
Construct a temporary path $p_u^{rj} = p_u^{ri} \oplus a_{ij}$
Generate arrival time distribution $Y_j(y_r)$ for the temporary path p_u^{ri}
Calculate $h(j)$ and $F(p_u^{ri})$
If p_u^{rj} is a nondominated path, then insert it into P^{rj} and SE and remove all paths dominated by p_u^{rj} from P^{rj} and SE

The placement selection algorithm, first calculates $F(p_u^{rj})$ based on the time required to traverse each path. Then all paths ordered by $F(p_u^{rj})$ are put into a priority queue. The path with minimum $F(p_u^{rj})$ is selected, in each iteration, An acyclic temporary path p_u^{rj} is then constructed by extending the selected path to each successor node SCS(i). If p_u^{ri} is a nondominated path, it is inserted into p^{ri}. If p_u^{ri} FSD dominate any path in p_u^{rj} , the dominated paths are deleted from p^{rj}.The algorithm continues until the destination is reached or the queue becomes empty. The time complexity of this algorithm is $O(n * m)$.

2) Route Selection: After deploying middleboxes, two constraints exist. One is the cost to process dataflow streams in IoT, such as the mandatory use of the CPU, RAM and counter. The other is the volumn restrictions in SDN switches. To address these challenges, First, an offline Integer Linear Program(ILP) pruning algorithm is proposed to tackle the switch constraints. Then an online Linear Program formulation is proposed to deal with load balancing.

After the middlebox is deployed, we need to select the route which can minimize the network load. A binary indicator variable $I_{c,q}$ is used to indicate whether a particular sequence has been selected. To ensure that each logical chain has a related physical sequence, a target coverage level Cov is defined as follows.

$$\sum_q I_{c,q} \geq Cov \tag{9}$$

$$I_{c,q} \in \{0, 1\} \tag{10}$$

Then, the loads on each switch are determined to ensure that loads do not exceed switch constraints TCAM.

$$\sum_{S_k \in PhSeq_{c,q}} Rules_{k,c,q} \times I_{c,q} \leq TCAM_k \tag{11}$$

To ensure that no middlebox becomes a hot-spot, the number of middleboxes in the selected sequences is calculated using Linear Program pruning algorithm.

$$MbUsed_j = \sum_{M_j \in PhSeq_{c,q}} I_{c,q} \tag{12}$$

Then find the maximum occurrences.

$$MaxMbO = max(MbUsed_j) \tag{13}$$

Finally minimize the maximum number.

$$MaxMbO \geq MbUsed_j \tag{14}$$

Algorithm 2 ILP PATH SELECTION ALGORITHM

Input: $PoChain_c$
Output: Improved security network
Initialization:Cov, TCAM, $SI_{c,q}$, Srules, MbUsed, MaxMbO;
for each $PoChain_c$ **do**
all possible physical chains are $PhSeq_c$;
end for
for $PhSeq_{c,q} \in PhSeq_c$ **do**
if this $PhSeq_{c,q} ischoosed$ **then**
$I_{c,q} = 1$;
apply $Rules_{k,c,q}$;
else $\{I_{c,q} = 0\}$
$SI_{c,q} += I_{c,q}$;
end if
if $S_k \in PhSeqc, q$ **then**
$Srules += Rules_{k,c,q} * I_{c,q}$;
end if
if $M_j \in PhSeq_{c,q}$ **then**
$MbUsed += I_{c,q}$;
MaxMbO = max(MbUsed);
end if
end for
Obtain:
$SI_{c,q} \geq Cov$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

The ILP pruning algorithm first determines whether a particular sequence has been selected. Then the total number of selected sequences are calculated and let it larger than the coverage level Cov. Next the loads on each switch is calculated to ensure that the load does not exceed the constraints. Finally the total number of middleboxes in each selected sequence is obtained, and the maximum number is minimized. The time complexity of this algorithm is $O(n * m)$.

When a physical sequence is selected, the load balance problem must be addressed to protect the network.

First, we need to ensure that traffic on all logical chains is assigned to some physical sequence.

$$\sum_{PhSeq_{c,q} \in Pruned} f_{c,q} = 1 \tag{15}$$

$$f_{c,q} \in [0, 1] \tag{16}$$

Next, the load on each middlebox is modeled as follows.

$$Load_j = \frac{\sum_{\substack{M_j \in PhSeq_{c,q} \\ PhSeq_{c,q} \in Pruned}} f_{c,q} \times T_c \times Ftprt_{c,j}}{ProcCap_j} \tag{17}$$

Finally, a specific load-balancing objective is selected to minimize the maximum middlebox load across the network.

$$MMbLd \geq Load_j \tag{18}$$

Algorithm 3 LP LOAD BALANCE ALGORITHM

Input: $PoChain_c$
Output: load balanced network
for each $PoChain_c$ **do**
  if $PhSeq_{c,q}$ is selected **then**
    $0 \leq f_{c,q} \leq 1$;
  end if
  $Sf_{c,q} += f_{c,q}$;
  if $M_j \in PhSeqc, q$ **then**
    $Load_j = \frac{\sum_{M_j \in PhSeq_{c,q}} f_{c,q} \times T_c \times Ftprt_{c,j}}{\substack{PhSeq_{c,q} \in Pruned \\ ProcCap_j}}$;
    $MMbLd = max(Load_j)$
  end if
end for
Obtain:
  $Sf_{c,q} = 1$
  $MMbLd \geq Load_j$

The LP formulation algorithm first divides the whole network into several sequences based on $f_{c,q}$. , to ensure that traffic on all logical chains is assigned to some physical sequences. Next the load on each middlebox is modeled, and then a special load balance objective is selected to minimize the maximum middlebox load across the network. The time complexity of this algorithm is O(n).

V. SECURE MECHANISMS AND DATAFLOW MANAGEMENT PROTOCOL

The SDN controller set up flow rules based on switch ports provided switches and middleboxes. Thus, it is subject to spoofing attacks(MAC spoofing, IP spoofing, VLAN tag spoofing), since a dishonest switch can forge such information inside a packet. The controller is also subject to flooding attacks- an attacker may generate a large number of packets with arbitrary MAC and IP addresses, resulting in creating a large number of messages sent to the controller and a large number of flow tables inside a switch. Thus , it is possible to cause DoS or packet interception, as unknown packets are also flooded. For example, if the memory allocated for host profiles in a controller is full, an existing host profile(e.g., the oldest) will be overwritten, resulting in the flooding of a new packet destined to that host.

Table 2 summarize the secure mechanisms and the related attack forms. To defense these attacks, we propose a security mechanism. First, Allowing a switch to bind a port to one or several MAC addresses(MAC binding) to mitigate MAC spoofing attacks. Next, allowing limits on the number of MAC addresses to be associated with a switch port(MAC limiting) to mitigate MAC flooding attacks.

To mitigate VLAN Spoofing, an SDN switch can designate its ports as User-to-Network Interfaces(UNIs) and Network-to-Network Interfaces(NNIs), and remove VLAN tags in packets received from UNIs.

To mitigate IP spoofing, the controller plan IP address assignment and configure flow tables with IP prefixes to avoid learning IP based forwarding rules.

TABLE II
SECURE MECHANISMS AND RELATED ATTACKS

Mechanisms	Attacks
MAC binding	MAC spoofing attacks
MAC limiting	MAC flooding attacks
UNIs,NNIs ports	VLAN spoofing
Flow table with IP prefixes	IP spoofing

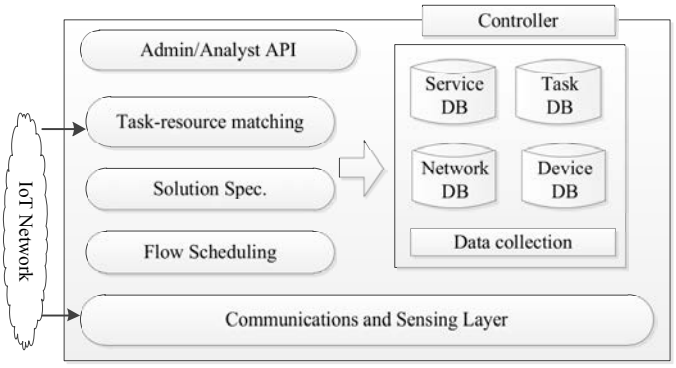

Fig. 6. IoT controller architecture.

The IoT controller architecture is shown in Figure 6, the data collection component collects network/device information from the IoT environment and stores it into databases. This information is then utilized by the layered components in the left side. The lower level Flow and Network layers decide which networks should be used for application flows and how application flows should be routed across the network. These decisions will be sent out to the corresponding devices via the communication and control layer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

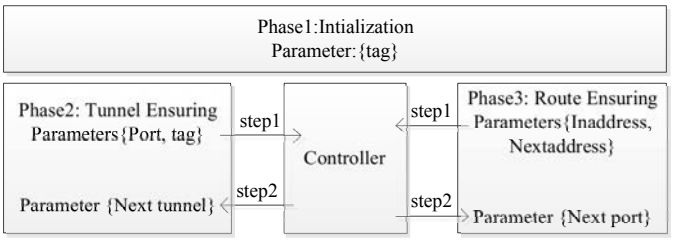

Fig. 7. Three stages of proposed protocol.

During the management of dataflows in IoT, a packet may go through the same switch many times, thus knowing the state of the packet is required to determine how to forward it.

A dataflow management protocol is proposed in IoT, to account for this challenge. The protocol involves three stages: initialization phase, tunnel ensuring phase and route ensuring phase. As shown in Figure 7, phase 2 and phase 3 are the execution stages, and including several steps. And the process details of the execution stages is shown in Figure 8.

Phase 1: Initialization. In this phase, the direction of a packet is determined by its ingress port. Then, a tag is given to the packet based on the security policy in M-G.

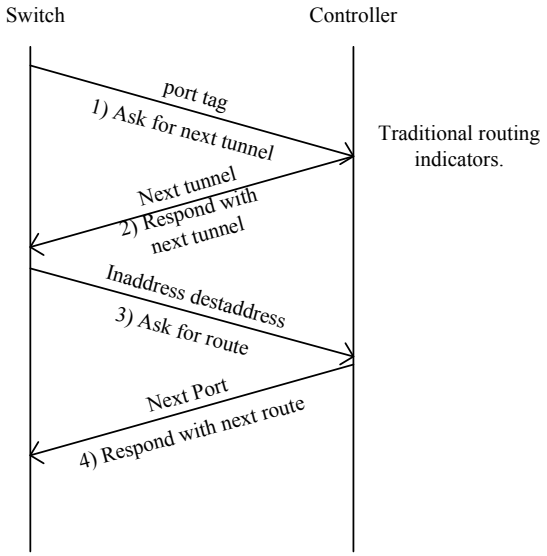

Fig. 8. The process of tunnel ensuring and route ensuring.

Phase2: Tunnel ensuring. The next tunnel of the packet is confirmed in this phase.

Step 1: The switch sends parameters including the in port of this packet and the tag to the controller.

Step 2: After receiving the parameters, the controller compute the traditional routing indicators based on port and tags, and return the next tunnel to the switch.

Phase3: Route ensuring. The switch get the route to the next tunnel in this phase.

Step 3: The switch sends its address and the address of the tunnel to the controller.

Step 4: After receiving the ingress address and destination address, the controller get the next hop based on route selection algorithm.

VI. EXPERIMENT

A. Comparison between M-G and SIMPLE

M-G is most close to SIMPLE[23], both solves middlebox placement and route selection problems.The comparison between M-G and SIMPLE is shown in table III.

TABLE III
Comparison between M-G and SIMPLE

item	M-G	SIMPLE
placement selection	the shortest path	exhaustive search
security	secure mechanisms and protocols	none
latency	short	long
load	light	heavy

B. Experiment Configurations

In our experiments. POX was used as the controller, and OpenvSwitch was used as the SDN switch. The custom Click modules was used to act as middleboxes. Each switch was connected to a host with at most a middlebox. iPerf was run on the host to emulate different network matrices, and different port numbers/host addresses were used to distinguish traffic across chains. Each link has an emulated bandwidth of 100Mbps. The evaluated metrics are discussed below.

1) Placement:The aim of the placement selection algorithm is to find the deployment position for middlebox to reduce the network lentency. Here, we replayed the first week dataflow of open DARPA dataset to compare the latency in M-G and SIMPLE[23] to evaluate the effectiveness of the placement selection algorithm in M-G.
2) Volume constraints: The ILP and LP algorithms are proposed to tackle switch and middlebox volume constraints. After replayed the first week dataflow of open DARPA dataset, the load on each middlebox and the number of rules in each switch are used to measure the availability of M-G.
3) Security mechanisms: The aim of M-G is to improve network security. In case middlebox failure and traffic over load in some chains, M-G should rebalance the network in time, so that the whole IoT system could run stably and safely. The response time of M-G in those two scene are evaluated.
4) Protocol: The dataflow management protocol is used to efficiently manage dataflow. The time needed to deploy the protocol and the switch capacity after using the protocol are discussed, to evaluate the scalability of the protocol.

C. Availability

Effective middlebox placement in IoT is ensured using the placement selection algorithm. Then, Integer Linear Program, uses the pruning algorithm to select the appropriate route and Linear Program formulation is used to balance loads across middleboxes. To evaluate the availability of these algorithms, the following experiments were performed and the results are shown below.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

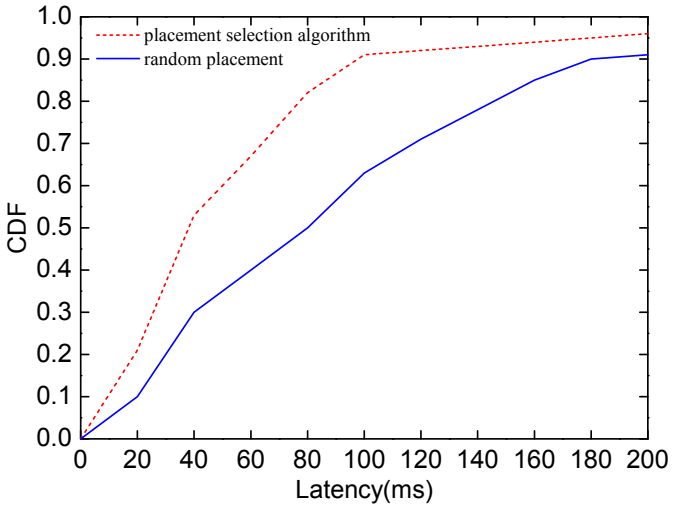

Fig. 9. Delay distribution of all flows.

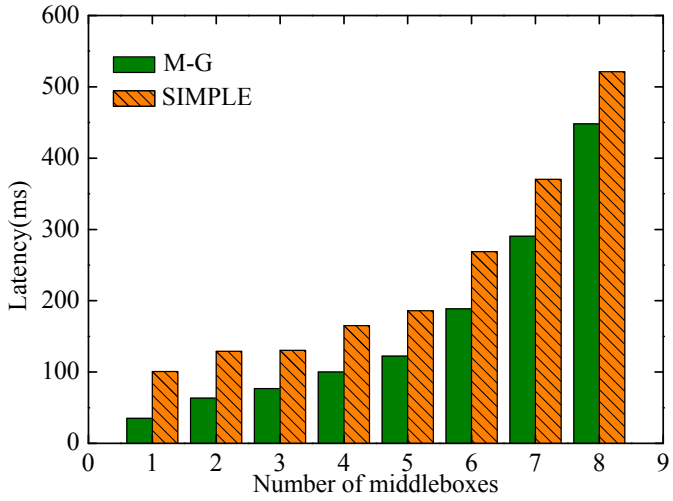

Fig. 10. Latency with different number of middlebox.

1) Availability of placement selection algorithm: The topology we used is Internet2. First, we allocate middlebox to each application, and compared the random placement and the placement selection algorithm. Figure 9 shows the delay distribution of all flows. For 50% of the flows, the proposed algorithm achieved 40ms latency, while random placement required 80ms. The number of middleboxes is varied in Figure 10, thus, it is easy to observe that the M-G can efficiently reduce the latency over time.

2) Availability of dataflow control algorithm: To evaluate the dataflow control algorithm, we compared the middlebox loads with and without the proposed M-G. The common one stands for that in each ingress-egress pair, the middleboxes closest to the ingress are selected. Here, the assumption is that there exist two types of middleboxes. i.e. Firewall and IDS, and that each switch is attached to one instance of a firewall and an IDS. As shown in Figure 11, after running M-G, the maximum loads are reduced by nearly 50%. Figure 12 shows the maximum middlebox load for different topologies. As can be seen, the performance gap is obvious. When running M-G, performance is improved by nearly four times.

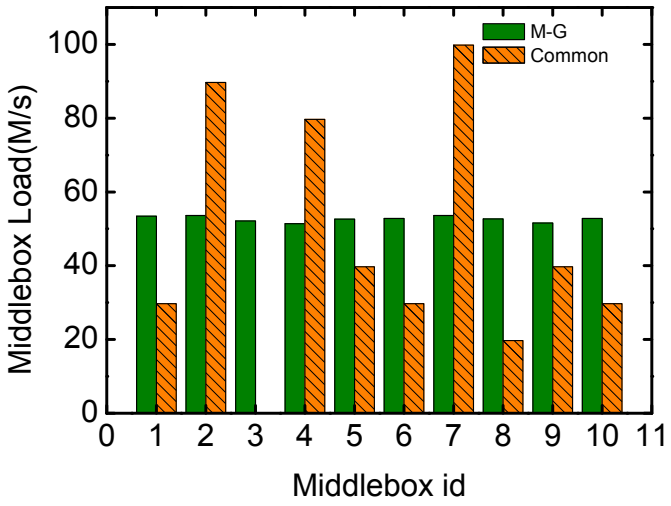

Fig. 11. Load on all middleboxes.

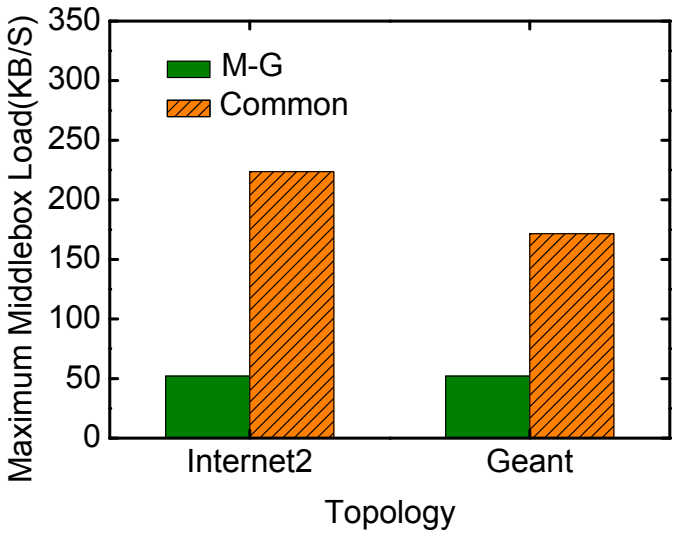

Fig. 12. Maximum middlebox load for different topologies.

3) Availability of Integer Linear Program pruning algorithm: Data from the first week of the open DARPA dataset was used to verify the availability of M-G under real-world settings. We replayed that dataflow and emulated the build and drop of rules in switches. If a rule was inactive in a time series it was removed. Here, n=1min and n=10min timeout thresholds were used and the different flow definitions are shown in Figure 13. As can be seen, increases to table size slowed down over time. Note that a greater timeout interval leads to fewer rules and faster convergence.

Figure 14 shows the time required to deploy rules in switch, which varied with different numbers of switches and middleboxes, because the controller deploys rules in switches sequentially.

D. Security

To evaluate the security of M-G, two situations were simulated using the Internet2 topology, i.e., middlebox failures

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

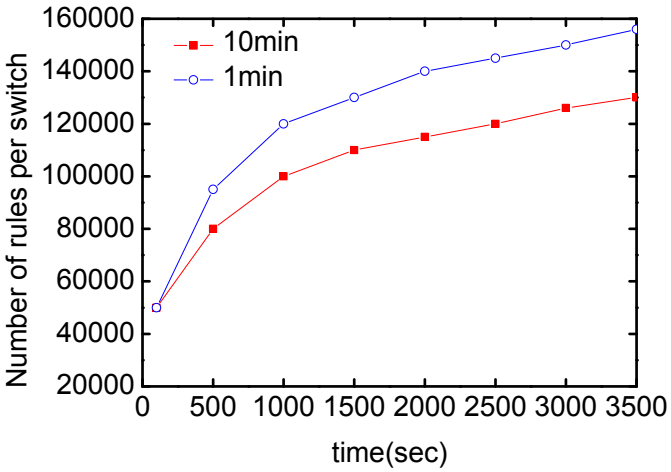

Fig. 13. Number of rules per switch with different timeout thresholds.

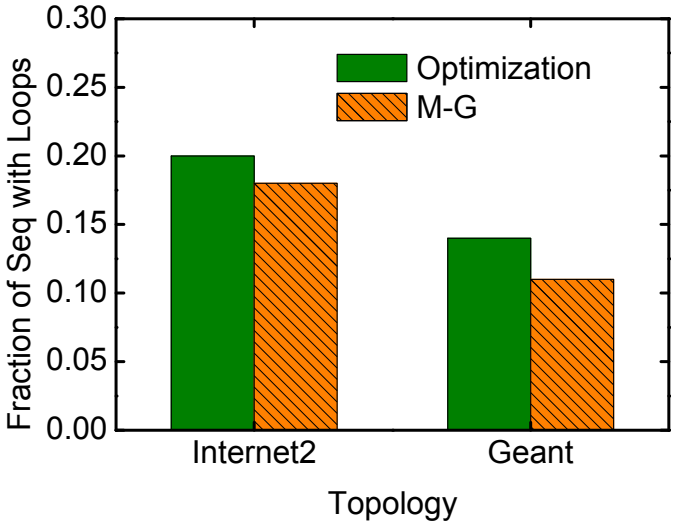

Fig. 16. Fraction of sequences with loops.

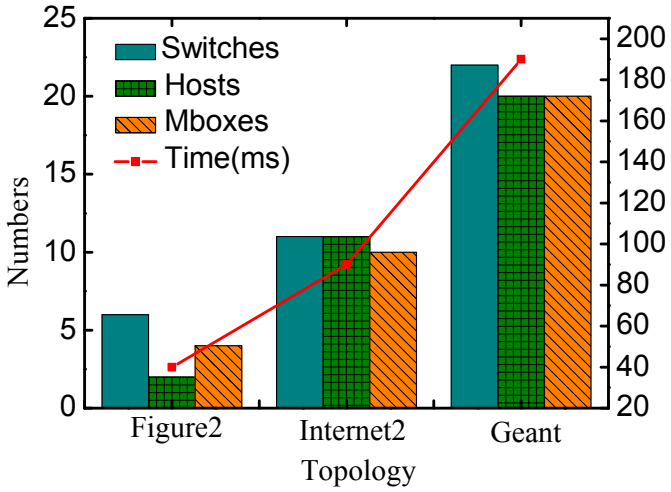

Fig. 14. Time required to deploy rules in switch.

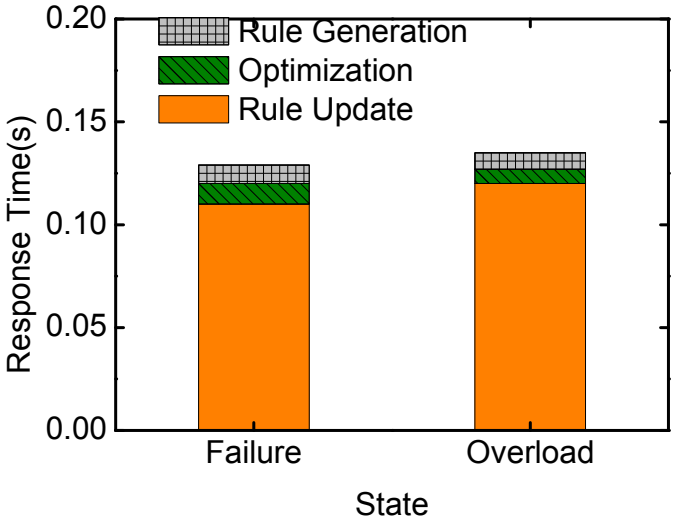

Fig. 15. Response time in case of middlebox failure and traffic overload.

and traffic overload on some chains. In both cases, the load must be rebalanced. Here, we focus on the time required to reconfigure the network. Figure 15 shows the response time to generate rules and update rules in both failure and overload two scenarios. As can be seen, the overall time is very low(<0.15s). This means that the network can react quickly. The optimization(without the ILP pruning algorithm) also proves the effectiveness of ILP algorithm.

In dataflow management protocol tunnels and tags were used to ensure the status of data packets. Figure16 shows the fraction of sequences selected by optimization(without the ILP pruning algorithm) and by M-G, which needs the ProcState tags. Although careful placement of middlebox can reduce the need for ProcState, it is necessary to use sequences with loops for correct policy traversal, under both failure and overload conditions.

E. Scalability

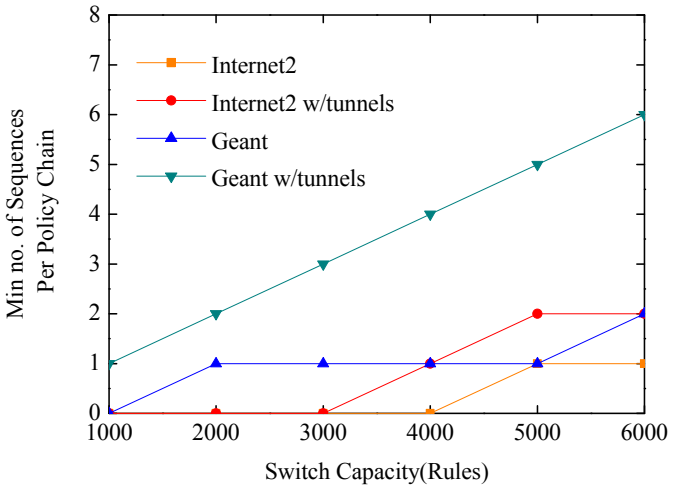

Fig. 17. Coverage with available switch capacity for selected topologies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2779180, IEEE Internet of Things Journal

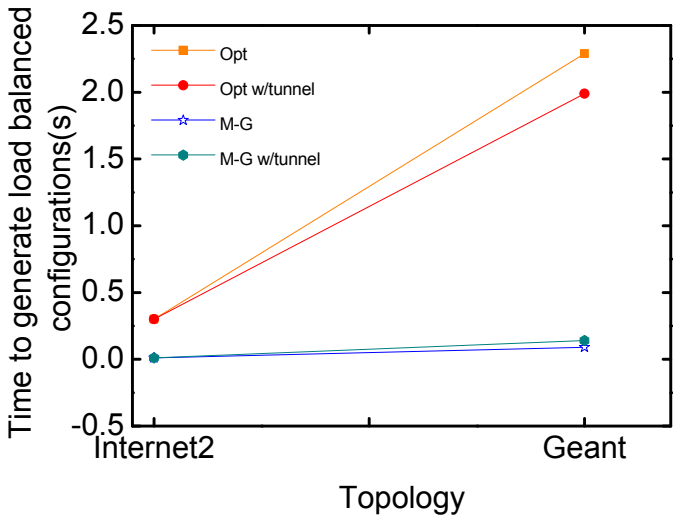

Fig. 18. Time required to deploy related dataflow streams management protocols.

Next, we focused on the scalability and optimality of M-G. The available TCAM size in switches and the number of policy chains per class in the Internet2 and Geant topologies are compared. For brevity, we only show results obtained when the length of each policy was three. Figure 17 shows that the coverage for each logical chain increases, after using switch tunnels. A coverage of 0 means no feasible solution was found. Note that we could only find feasible solutions when switch tunnels were used.

Figure 18 shows the time required to deploy related dataflow streams management protocols in M-G and optimization (without ILP pruning algorithm) using switch tunnels. By using tunnels and tags the required time was obviously decreased and the efficiency of M-G was improved.

VII. CONCLUSION

Middlebox placement, the middlebox and flow table capacity constraints of SDN switches are key challenges when combining middlebox and SDN together. The goal of this study was to use middlebox in SDN-based IoT to manage dataflow, and improve the stability and security of the network. To this end, M-G, a SDN-based data transfer security model in IoT based on middleboxes was proposed. M-G attempts to improve the availability of SDN-based IoT secure applications and actively respond to network threats. We first addressed middlebox placements. Appropriate positions are selected using a placement selection algorithm, which reduces network latency. Next, we considered network balance. Here, an ILP pruning algorithm and an LP formulation are deployed to balance load across middleboxes and switches. Finally, we investigated dataflow management. The secure mechanisms can defend several attacks. By using the proposed dataflow management protocol, the status of a packet is observed and the route is properly determined. Our experimental results demonstrate that the proposed M-G model and corresponding protocols manage dataflow in middleboxes effectively, and can improve the overall IoT network security and stability.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Elsevier Computer Networks, vol. 54, no. 15, pp. 2787−2805, Oct. 2010.
[2] N. McKeown, "Software-defined networking," IEEE INFOCOM keynote talk, vol. 17, no. 2, pp. 30−32, Apr, 2009.
[3] M. C. Dacier, H. Konig, R. Cwalinski, F. Kargl, and S. Dietrich, "Security Challenges and Opportunities of Software-Defined Networking," IEEE Security Privacy, vol. 15, no. 2, pp. 96−100, Mar, 2017.
[4] S Scott-Hayward, G. O'Callaghan, and S. Sezer. "Sdn Security: A Survey," IEEE SDN for Future Networks and Services.
[5] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. "Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags," 11th USENIX Symposium on Networked Systems Design and Implement.
[6] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN Be Your Eyes: Secure Forensics in Data Center Networks," The Workshop on Security of Emerging Network Technologies.
[7] L. Guo, J. Pang, and A. Walid, "Dynamic Service Function Chaining in SDN-enabled networks with middleboxes," 2016 IEEE 24th International Conference on Network Protocols.
[8] S. G. Kulkarni et al., "Name enhanced SDN framework for service function chaining of elastic Network functions," 2016 IEEE Conference on Computer Communications Workshops.
[9] G. Xiong, P. Sun, Y. Hu, J. Lan, and K. Li, "An Optimized Deployment Mechanism for Virtual Middleboxes in NFV- and SDN-Enabling Network," KSII Transactions on Internet& Information Systems, vol. 10, Aug, 2016.
[10] M. A. Jamshed, D. Kim, Y. Moon, D. Han, and K. Park, "A Case for a Stateful Middlebox Networking Stack," ACM SIGCOMM Computer Communication Review.
[11] D. A. Joseph, A. Tavakoli, and I. Stoica, "A Policy-aware Switching Layer for Data Centers," ACM SIGCOMM Computer Communication Review.
[12] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow Processing and the Rise of Commodity Network Hardware," ACM SIGCOMM Computer Communication Review, vol. 39, no. 2, pp. 20−26, Mar,2009.6.
[13] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69−74, Mar,2008.
[14] E. J. Jackson et al., "SoftFlow: A Middlebox Architecture for Open vSwitch.," 2016 USENIX Annual Technical Conference.
[15] Open vSwitch−An Open Virtual Switch. (2015,Sep) [Online]. Available:http://www.openvswitch.org.
[16] Y. Zhang et al., "StEERING: A software-defined networking for inline service chaining," 2013 21st IEEE International Conference on Network Protocols.
[17] Passive Performance Measurement for Inline Service Chaining, by Y. Zhang and J. Halpern,(2016,May,26). Patent US20160149788 A1.
[18] S. G. Kulkarni et al., "Name enhanced SDN framework for service function chaining of elastic Network functions," 2016 IEEE Conference on Computer Communications Workshops.
[19] A. A. Mohammed, M. Gharbaoui, B. Martini, F. Paganelli, and P. Castoldi, "SDN controller for network-aware adaptive orchestration in dynamic service chaining," 2016 IEEE NetSoft Conference and Workshops.
[20] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," IEEE Communication Magazine, vol. 55, no. 2, pp. 216−223, Feb, 2017.
[21] W. John et al., "Research Directions in Network Service Chaining," 2013 IEEE SDN for Future Networks and Services.
[22] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network Function Placement for NFV Chaining in Packet/Optical Datacenters," Journal of Light Technology, vol. 33, no. 8, pp. 1565−1570, Apr, 2015.
[23] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 27−38,Aug, 2013.
[24] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward Software-defined Middlebox Networking," Proceedings of the 11th ACM Workshop on Hot Topics in Networks.
[25] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A Slick Control Plane for Network Middleboxes," Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking.

[26] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," 2012 20th IEEE International Conference on Network Protocols.
[27] M. Allamanis, S. Scellato, and C. Mascolo, "Evolution of a Location-based Online Social Network: Analysis and Models," Proceedings of the 2012 Internet Measurement Conference.
[28] Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building Robust Firewalls for Software-defined Networks," Proceedings of the Third Workshop on Hot Topics in Software Defined Networking.
[29] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is It Still Possible to Extend TCP?," Proc 2011 ACM SIGCOMM Conference on Internet Measurement Conference.
[30] R. Craven, R. Beverly, and M. Allman, "A Middlebox-cooperative TCP for a Non End-to-end Internet," ACM SIGCOMM Computer Communication Rev, vol. 44, no. 4, pp. 151−162, Aug,2014.
[31] K. R. Khan, Z. Ahmed, S. Ahmed, A. Syed, and S. A. Khayam, "Rapid and Scalable Isp Service Delivery Through a Programmable Middlebox," ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 31−37, Jul,2014.
[32] F. Qian, V. Gopalakrishnan, E. Halepovic, S. Sen, and O. Spatscheck,"TM3:flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking[C]". Proceedings of the 11th ACM Conference on Emerging Network Experiments and Technologies.

Yanbing Liu received the M.S. degree in computer science and technology from Beijing University of Posts and Telecommunications, China, in 2001 and the Ph.D. degree from University of Electronic Science and Technology, China, in 2007. He is currently a Professor and a Ph.D. supervisor of the Chongqing University of Posts and Telecommunications. He was a recipient of the National Science and Technology Award and several Chongqing Science and Technology Awards. He is currently an Executive Director of the Chongqing Youth Federation of Science and Technology. He has authored over 60 refereed papers. His research interests include information security and management, security in cloud computing, and Internet of things.

Yao Kuang received the B.S. degree in information security from Chongqing University of Posts and Telecommunications, China, in 2015. She is currently working toward the M.S. degree in computer science and technology with Chongqing University of Posts and Telecommunications, China. Her research interests include information security and management, software-defined networking, cloud computing, and Internet of things.

Yunpeng Xiao is an Assistant Professor of Chongqing University of Posts and Telecommunications, China. He received his Ph.D. degree and master degree in Computer Science and Engineering from Beijing University of Posts and Telecommunications, China, and his B.Sc. degree in Information System from Chongqing University of Posts and Telecommunications, China. His current research interests are in the areas of big data, mobile Internet and information security.

Guangxia Xu received her PhD and M.S. Degrees in computer science from the Chongqing University, China. She is currently a professor of Chongqing University of Posts and Telecommunications. Her research interests include Information Security and Network Management, Big Data Analytics for Network Security, and Internet of Things(IoT) Data Streams Analytics. She is currently the research vice director at Network and Information Security Engineering Center in Chongqing, China. She is also a committee member as Fault Tolerant Computing of China Computer Federation, and a vice chairman of the Information Security Association in Chongqing, China.