Abstract—Scalability of Software Defined Networking (SDN) approach is one of the key issues that many network operators are willing to address and understand. Indeed, the promised programmability and flexibility of a SDN network is paid with a non-negligible control traffic exchanged between the network nodes and the SDN controllers.

We consider a network controlled by a real OpenFlow-enabled controller, i.e. OpenDaylight. We evaluate analytically the number of OpenFlow messages for installing a new traffic flow, assuming the default reactive forwarding application available in OpenDaylight. We apply these results to the specific case of a large ISP network, comprising a backbone interconnecting many POPs. By evaluating exactly the amount of generated control traffic, we are able to assess the scalability of the reactive forwarding application in a practical relevant scenario for ISPs.

I. INTRODUCTION

Software Defined Networking (SDN) is a powerful network paradigm that permits to build highly adaptable and easily manageable networks. The main idea of SDN is to separate the control and data planes in the network: the control plane runs only in a centralized controller leaving the switching devices with only the functionality of forwarding packets. Thus, the controller has the complete view of the network and can easily implement any forwarding or dropping rule. Due to its increasing adoption, SDN is becoming the enabling technology for traffic engineering in operational networks.

In the last few years, many protocols and architectures have been proposed to support the SDN paradigm. The reader can refer to [1] for an updated and comprehensive view of the different approaches proposed and implemented so far. Due to its popularity, in the following we will consider specifically the OpenFlow (OF) protocol [2], used to install forwarding rules within the switches. Despite its relative simplicity, the OF protocol provides the basic capabilities to enable advanced reactive SDN policies.

Given its centralized nature, a natural question arises about the scalability of SDN approach. In particular, for large network operators, with wide geographical coverage as a national-level ISP, it is of paramount importance to understand if the approach is suitable to cope with the heterogeneity and complexity of their legacy networks. Note that an ISP scenario may appear very different from the benign “ecosystem” of a large datacenter, where the network design is simplified by the adoption of structured topologies and homogenous hardware devices. On the other hand, the long term evolution of ISP networks could lead to rethink the WAN as a kind of distributed datacenter.

Many solutions have been proposed to improve the scalability of native OF, mainly by distributing the control plane among different servers, while mimicking logical centralized decisions. As complementary approach, the software implementation of the controller can be optimized to maximize the performance. In addition to the above efforts to improve the scalability of the current solutions, we believe that a key stone in understanding the scalability of SDN in large ISP would be to develop comprehensive models to investigate the requirements in terms of bandwidth and delays for the control network to fully support the QoS requirements in the data plane. As a first step towards such models, in this work we evaluate in details the amount of control messages that are required to install each traffic flow. Thanks to this result, given a network topology and a set of injected flows, it is possible to deduce the corresponding bandwidth required in the control network.

In our paper we provide the following contributions:
1) we evaluate the exact number of control messages for each new flow, validated for the specific case of OF switches controlled by an open-source controller, namely OpenDaylight.
2) we apply the above result to compute the control traffic for reactive OF in a realistic topology of a nation-wide ISP, while varying the size of its POPs and the traffic matrix describing the users’ behavior.

The paper is organized as follows. Sec. II provides an overview of SDN and a description of the reactive forwarding policy considered in our model. Sec. III evaluates analytically the number of OF messages for flow setup for a generic network path. In Sec. IV we describe an ISP topology model and in Sec. V we apply the results from Sec. III to evaluate the bandwidth required for control traffic. Finally, in Sec. VI we draw our conclusions.

II. REACTIVE PACKET FORWARDING IN SDN

We consider a completely centralized implementation of a SDN network, in which one controller (running on a single physical server) is responsible for the control plane decision of all the switching nodes in the network. We can identify two opposite operational regimes, with very
different impact on the scalability and reactivity of the SDN control [3]:

1) **reactive regime**, in which the forwarding decision is taken flow-by-flow. Every time a new flow (i.e., a flow for which no forwarding rule is available) arrives at a switch, the switch interacts with the controller to install locally a forwarding rule to route all the packets of the flow. The main advantage of this scenario is that it enables the online implementation of flexible routing/dropping policies, without any preliminary knowledge of the traffic. Furthermore, flow entries are added only when needed, reducing the memory requirements in the switches. On the other side, for the initial packet of any flow an additional delay is experienced, and the load to the controller and in the control network is increased.

2) **proactive regime**, in which the controller pre-populates the flow entries on the network switches, thus avoiding the initial flow setup time. Note that this scheme can run off-line and requires a preliminary knowledge of the routing rules applicable in the network.

Due to the much shorter temporal scales in the interaction between the controller and the network switches, the reactive regime poses the most critical challenges to the scalability of the network. Aware that real networks are expected to operate in an hybrid regimes among the two schemes, to trade flexibility with scalability, in our work we focus only in the scenario of reactive forwarding since it represents the worst-case scenario in terms of generated control traffic.

When considering specifically OF-based SDN networks running in reactive regime, the control traffic is generated by different OF messages that are devoted to (i) initial device configuration, (ii) flow setup, (iii) keep-alive schemes. The flow setup phase has the strongest impact on the control traffic because of its shorter time scale. In particular, when the first packet of a flow arrives at a switch and no forwarding rule is matched, the switch sends a “pkt_in” message to the controller with a copy of the packet (or just part of it) to ask for instructions. The controller processes the request and sends back to the switch a “pkt_out” message with the action (“drop” or “forward to a specific output port”) to handle the received packet. Usually, pkt_out is used for sending out ARP requests/replies and LLDP messages (used for network topology discovery). In addition to this OF message, the controller can send “flow_mod” messages to the switch to install a new forwarding rule. Thus, a subsequent packet of the same flow arriving at the switch will match the forwarding rule and will be directly sent to the destination port, without triggering any pkt_in message. To minimize the control traffic, the flow_mod packet is sent just after the first pkt_out sent by the controller.

A. Reactive Forwarding in OpenDaylight

OpenDaylight [4] (ODL) is an open source project supported by some of the leading industries to provide a “universal” SDN platform, to support SDN and Network Functions Virtualization (NFV) in a wide spectrum of networking scenarios. In February 2014 the first release of the platform, called “Hydrogen”, became available in three editions. We specifically consider the “base edition” providing a default layer-3 reactive forwarding application, denoted as “Simple Forwarding”.

We validate our findings with two complementary approaches. First, we run the SDN network emulator mininet [5], and test the application by recording through a network sniffer the complete sequence of OF messages exchanged for the flow setup in a simple linear topology connecting the switches, as shown in Fig. 1. Second, to generalize our findings for any topology, we also analyze in details the ODL source code.

The network topology, interconnecting all the switches, is known in advance to the ODL controller, thanks to the preliminary phase of topology discovery which occurs thanks to LLDP [6] protocol running among the switches. Note that the host position, defined as the switch port the host is connected to, is a-priori unknown, since the hosts do not run the LLDP protocol. Thus, the behavior of Simple Forwarding is driven by the fact that only the switch topology is known in advance, whereas the host position must be discovered in real time. In Sec. III we will discuss the detailed implementation of the forwarding process, exploiting ARP packets to discover the position of active hosts and installing forwarding rules in every switch to reach any known hosts, as better explained later.

III. Evaluation of SDN Control Traffic

We consider the flow setup process to establish a communication between two hosts by installing appropriate forwarding rule on OF switches. This process involves exchanging OF messages between the switches and the controller. In this context, a flow is defined at the IP layer and it is univocally identified by the pair: source IP and destination IP address. In the following, we evaluate the exact number $M$ of OF messages that are exchanged for each new flow setup, measured in terms of flow_mod messages (denoted as $m_{fm}$), pkt_in messages (denoted as $m_{pi}$) and pkt_out messages (denoted as $m_{po}$). We consider a generic bidirectional network topology interconnecting a set of $N$ switches. Within a switch, one port connected to
Another switch is denoted as “internal port”, whereas one port connected to an host (client and/or server) is denoted as “host port”. Note that an host port can be connected to many hosts (e.g., through non-OF switches/hubs). Let $H$ be the total number of host ports present in the network.

To describe the OF message exchange for Simple Forwarding Application, we distinguish among four possible cases, based on whether the positions of the source or destination hosts are known to the controller.

**Case “uu”: Both hosts are unknown**: At high level, we can summarize the whole message exchange by two facts: the hosts’ positions are discovered by the ARP requests sent by the source and destination hosts, and the forwarding rules to reach any host are installed in all the OF switches in the network. To describe the message exchange in detail, we refer to Fig. 2 as reference example and to the sequence of packets described in Table I.

Whenever a host starts a new IP flow towards another host, the first generated packet is an ARP request (ARP-REQ) which reaches the local switch (denoted as root node) at which the source host is connected (step 1 in Table I).

This ARP packet is sent to the controller through a pkt_in (step 1) and this allows ODL to learn the source host (H1) position. Now ODL computes the shortest path (using standard Dijkstra algorithm) from all the OF switches towards the root node and then installs a forwarding rule (in terms of “host specific rule”) in every switch in the network (step 2), through $N$ flow_mod messages. From now on, all the switches know how to reach the source host.

Now the controller must discover the destination host position. Differently from what happens in a standard Ethernet switch, the ARP request is broadcasted only on host ports, present in all OF switches in the network, through $H$ specific pkt_out messages (step 3); thus, a switch connected only to other switches (without host ports) is not involved in this phase. The ARP reply (ARP-REP) sent by the destination host (step 4) allows the controller to learn about the destination’s position; now the controller computes the shortest paths from all the OF switches towards the root node of the destination host (H2) and install the corresponding forwarding rules in all $N$ OF switches (step 5). At this moment, all the OF switches (even if not involved in the shortest path between the two hosts) are equipped with the forwarding rules to reach both the source (H1) and the destination (H2) hosts. Finally, the ARP reply is sent back to H1 (step 6) who learns about the MAC of H2.

After updating its ARP table, the source host can send its IP packets to the destination host. Whenever an IP packet is received in a switch along its path, it is immediately forwarded towards the destination thanks to the already installed forwarding rules (step 7). When the first IP packet arrives at the destination, we highlight a peculiar protocol behavior. Indeed, during the past step 3, the controller has sent the ARP request from H1 to the root switch of H2 with its own MAC/IP addresses as source. Thus, differently from standard Ethernet network, the destination host is not able to learn the MAC address of the source host at the end of this step. This guarantees that the ARP reply generated by H2 will be sent to the controller, otherwise it would be sent directly to the source host (H1), thanks to the forwarding rules that have been previously installed to reach H1 (in step 2). In conclusion, to be able to send back an IP packet to H1, after step 7, the destination host sends an ARP request to learn the source host MAC address (step 8) which reaches the controller. The controller replies back directly to the destination host by sending an ARP reply (step 9); now H2 can update its own ARP table and, finally, transmit back the IP packet to the source host. From now on, all the IP packets in both directions will not generate other OF messages, if not when the forwarding rule expires.

In summary, the total number of OF messages exchanged

---

**Fig. 2. Example network with 3 hosts H1-H3 and 3 OF switches S1-S3.**

**Table I**

Sequence of packets exchanged during flow setup H1→H2 when both hosts are unknown, for the network of Fig. 2.

<table>
<thead>
<tr>
<th>Step</th>
<th>Communication</th>
<th>Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H1→S1</td>
<td>pkt_in (ARP-REQ:H2)</td>
</tr>
<tr>
<td>2</td>
<td>C→S1</td>
<td>flow_mod (how to reach H1)</td>
</tr>
<tr>
<td>3</td>
<td>C→S2</td>
<td>flow_mod (how to reach H1)</td>
</tr>
<tr>
<td>4</td>
<td>C→S3</td>
<td>flow_mod (how to reach H1)</td>
</tr>
<tr>
<td>5</td>
<td>S1→H1</td>
<td>ARP-REQ-H2</td>
</tr>
<tr>
<td>6</td>
<td>S2→H2</td>
<td>ARP-REQ-H2</td>
</tr>
<tr>
<td>7</td>
<td>S3→H3</td>
<td>ARP-REQ-H2</td>
</tr>
<tr>
<td>8</td>
<td>usual IP forwarding from H1 to H2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>usual IP forwarding from H2 to H1</td>
<td></td>
</tr>
</tbody>
</table>

1. We use as notation $uu$ to denote that both source and destination are unknown. Similarly we will later define $uk$, $ku$, $kk$, with $k$ standing for known.
is:

\[ M_{uu} = 1[m_{pi}] + N[m_{fm}] + H[m_{po}] + 1[m_{pi}] + N[m_{fm}] + 2[m_{po}] + 1[m_{pi}] = (H + 2)[m_{po}] + 3[m_{pi}] + 2N[m_{fm}] \]  

(1)

where we use as a notation \( X[m] \) to denote \( X \) messages of type \( m \).

Case “ku”. Only the source host is known: The behavior is exactly equal to the previous case, but the \( N \) flow_mod messages to install the forwarding rules to reach the source host at each switch (step 2) are not needed. Thus, by revising (1) we obtain:

\[ M_{ku} = (H + 2)[m_{po}] + 3[m_{pi}] + N[m_{fm}] \]  

(2)

Case “uk”. Only the destination host is known: The behavior is similar to case \( uu \), but the \( H \) pkt_out messages to discover the destination hosts with ARP-REQ (step 3) and the pkt_in message with the corresponding ARP-REP (step 4) are not needed. Furthermore, all the \( N \) flow_mod messages to install the forwarding rules to reach the destination host (step 5) are useless. Thus, by revising (1):

\[ M_{uk} = 2[m_{po}] + 2[m_{pi}] + N[m_{fm}] \]  

(3)

Case “kk”. Both source and destination hosts are known: ARP requests are not broadcasted and no flow_mod messages are involved. Thus, the only messages we observe are the pkt_in and pkt_out messages exchanged to transfer the ARP requests and reply from/to the hosts to their attached switches. In summary,

\[ M_{kk} = 2[m_{pi}] + 2[m_{po}] \]  

(4)

From the above results, by combining (1)-(4), we can claim the following property:

**Property 1**: In Simple Forwarding Application, for each new flow, the number of OF messages scales at most linearly with the number of host ports and with the number of OF switches present in the network; i.e.,

\[ M = O(H) \text{ and } M = O(N) \]

Recalling that one host port can correspond to many hosts (clients and servers), Property 1 states that the total number of hosts does not directly affect the control traffic generated by each new flow. This fact guarantees some level of scalability when the number of hosts is very large.

Furthermore, note that the most of the control traffic is generated whenever the positions of the communication end-points (hosts) are unknown. After the controller has learnt the position of every host, the control traffic is negligible and aimed only at answering locally to the hosts ARP requests. Thus, Simple Forwarding Application can be considered a “reactive” application in terms of new IP hosts, and not directly at flow level. This fact too improves the scalability when the network becomes large.

### IV. A HOMOGENOUS MODEL FOR ISP NETWORK

We consider the scenario in Fig. 3 as depicting a general model of ISP, composed by a backbone network interconnecting \( P \) POPs. All the switching devices present in the POP are assumed to be OF switches managed by a single controller, running independently of the other POPs. Each POP is composed of \( B \) OF access routers. Each access router acts as a Broadband Remote Access Server (BRAS) and it is connected to \( D \) DSLAMs (Digital Subscriber Line Access Multiplexers) [7]; the corresponding \( D \) ports are configured as host ports. Each access router is responsible to provide the access to \( K \) users, distributed across the \( D \) DSLAMs.

As a preliminary scenario, we assume the same size for all the POPs, i.e., an homogenous scenario. The overall number of users \( U_{POP} \) in one POP is \( U_{POP} = KB \), whereas the total number of users supported by the ISP is \( U_{ISP} = PU_{POP} \). Thus,

\[ B = \frac{U_{ISP}}{PK} \]  

(5)

Note that this scenario can be easily extended to realistic non-homogenous cases.

Coming back to Fig. 3, two OF switches are responsible to provide load balancing and redundant paths between the access routers and two border routers connected to the backbone. The border routers are OF switches too and their ports towards the backbone network have been configured as host ports to keep the controller aware only of the internal POP topology and to route the data across the POP obliviously of the backbone routing policies. In total, the number of OF switches within the POP is:

\[ N = 4 + B \]  

(6)

According to the interconnection network shown in Fig. 3, the number of host ports defined so far is

\[ H = 2 + DB \]  

(7)
since each access router provides $D$ host ports towards the DSLAMs and each border router provides one host port to the backbone.

We only focus on the control traffic internal to the POP and due to flow setup by the users (i.e. the ISP customers). We assume that each user establishes $F$ distinct flows, during some observation window $T$. Since the concept of flow is defined at IP level, each flow is compatible with many sessions at higher levels (e.g. TCP, UDP or RTP).

We consider the following two opposite traffic scenarios:

1) **Local traffic**: All the users’ traffic is destined to a single internal server present to the POP, as shown in Fig. 4. This server is associated with $F$ multiple IP addresses, to support multiple network services. It could be a caching node supporting Content Distribution Network (CDN) or the ingress router to a datacenter. It is connected to the two intermediate switches through a host port, thus we need to increase by two the number of host ports:

$$H = 4 + DB$$

The total number of flows in the POP is $FU_{POP}$.

2) **External traffic**: Each user is connected to $F$ distinct servers, located externally to the POP, as shown in Fig. 5. As worst case, we assume that the sets of the external servers contacted by each user are disjoint. The total number of flows is $FU_{POP}$ as in the previous case. For a fair comparison, the internal server is still present, but no flow is directed to it. Note that the corresponding host ports are involved in the host discovery phase through ARP requests. The number of host ports is again given by (8).

**V. OPENFLOW CONTROL TRAFFIC IN EACH POP**

We evaluate the amount of OF messages exchanged in each POP due to the flow setup, based on the results (1)-(4). To obtain the corresponding bandwidth, we consider the packet sizes observed in the mininet emulator, and reported in Table II. Since all the sizes are similar, when evaluating the number $M$ of OF messages, to improve readability we will report the total number of messages independently of their types.

We assume a scenario corresponding to a nation-wide ISP providing the service to 16 million active users, for the setting of parameters of Table III. We investigate the control traffic as a function of the number of available POPs, among which the whole population of users has been divided.

**A. Local traffic scenario**

To maximize the control traffic, we consider the following sequence of events, assuming $F < U_{POP}$. First, $F$ users establish a flow each to distinct IP addresses of the server, each flow generating $MU_{uu}$ messages. Second, each of the remaining $U_{POP} - F$ users establishes one flow to one distinct (but already known) IP address of the server, thus generating $M_{uk}$ messages. Finally, each user establishes the remaining $F - 1$ flows to different servers, thus generating $M_{kk}$ messages. Recalling (1)-(4), combined with (6) and (8), we can evaluate the total number the messages as:

$$M = FM_{uu} + (U_{POP} - F)M_{uk} + U_{POP}(F - 1)M_{kk} = U_{POP}(AF + B + 4) + F(B(D + 1) + 9)$$

The bottom curve and the middle one in Fig. 6 show the number of messages according to (9), when varying the
POP size and the number of flows per user. The right y-axis reports the corresponding amount of bytes exchanged during the observation window $T$. The overall bandwidth needed by the control plane can be directly obtained by dividing this amount by $T$.

When $F$ is large, (9) can be approximated by $4U_{POP}F$ and thus the number of messages per flow tends to 4, as observed in the bottom curve. Instead, for $F = 1$, (9) is approximated by $U_{POP}(B + 8)$ and the numbers of messages per flow tends to $(B+8)$, which decreases as $P^{-1}$ due to (5), as shown in the middle curve. By comparing the two curves, it is clear larger values of $F$ permit to amortize the initial control messages to discover the endpoints’ IP addresses.

### B. External traffic scenario

The first flow established by each user generates $M_{uu}$ messages, whereas the following $F - 1$ generates $M_{ku}$ messages. Thus, by combining (1)-(2) with (6) and (8), we can evaluate the total number of messages:

$$M = U_{POP} M_{uu} + U_{POP}(F - 1) M_{ku} = U_{POP}((D + 2)B + 17 + (F - 1)(D + 1)B + 13))$$

(10)

The two top curves in Fig. 6 show (10) for two different values of $F$. They are coincident, since they can be both approximated as $M/(U_{POP}F) \approx DB$, which is independent of $F$. Note that this factor is due to the OF messages to advertise the endpoints of each flow to all the $DB$ host ports present in the access router of the POP. Finally, the curves decrease as $P^{-1}$ coherently with (5). In terms of required bandwidth, the amount of control traffic in large POPs (i.e. small $P$) may be relevant, due to the massive number of flows to manage within the POP.

### C. The role of the traffic matrix

The results reported in Fig. 6 refer to two opposite traffic matrices, in which we vary heavily the number of contacted servers, while keeping the same number of active flows (i.e., $F$ flows for very user). Comparing the two cases, the number of messages per flow in the case of external traffic is up to $33 \times$ larger than the case of internal traffic (i.e. for $F = 1024$). As a conclusion, we can claim:

**Property 2:** In Simple Forwarding Application, the amount of OpenFlow control traffic heavily depends on the traffic matrix between the communication end-points.

When designing the network for the control plane, this issue must be carefully considered to guarantee enough bandwidth for the control traffic processed by the SDN controller.

### VI. Conclusions

We focused our investigation on the amount of OpenFlow traffic due to the default reactive forwarding application (Simple Forwarding), available in OpenDaylight, which is a state-of-the-art SDN controller. Thanks to a detailed analysis of the interaction between the controller and a sample network implemented in Mininet, we were able to develop a numerical model evaluating the number of messages exchanged on the control plane due to the flow setup phase. As a corollary of our investigation, we were able to assess the asymptotic scalability of the considered approach when increasing the number of switches and hosts in the network. We also highlighted the crucial role of the number of ports configured as “host ports”.

To show the application of our model, we introduced a simplified but realistic model for a POP of a large ISP and evaluated the amount of control traffic when varying the POP size, given the same amount of overall customers in the ISP. We were able to observe the important role of the traffic matrix between the communication endpoints, suggesting possible design tradeoff between the POP size and the amount of control traffic that a controller must be able to process.

Our investigation provides a quantitative analysis on the non-negligible amount of control traffic in a SDN network running in reactive mode. Our results help to properly dimension the control plane network.

### References