# Chapter 1

# Introduction

## 1.1 Traditional Networks

Traditional computer networks consist of various special purpose devices like routers, switches, firewalls, etc. Each device has a specific role in the network. It is the network operator's responsibility to configure these devices as per the defined policies to handle network events and traffic. The problem with traditional devices is that each device comes with a proprietary interface for control and configuration. Thus, devices from different vendors make the network operator's task more complicated and difficult. The network operator has to configure these devices carefully to ensure that the network policies are adequately implemented.

Traditional devices come as a black box with a coupled control plane and data plane. The control plane is the brain of the black box. All the decisions related to network management are taken/defined by the control plane. These decisions are applied in the data plane in the form of forwarding rules. To define the forwarding rules, some routing protocols require the network devices to share the topology information among themselves. The forwarding rules are simple match-action pairs. The header field of every incoming packet is checked against the match-field of forwarding rules until a match is found. Once a match is found, the corresponding action is performed. In traditional networks, the control plane is embedded in each device and devices are distributed over the network. This makes it difficult to have an updated global view of the network topology and take optimal network-wide decisions [5, 6, 7].

Network operator has to configure the layer-2 switches and layer-3 routers for routing. Network operator is allowed to use only certain protocols, for example, at layer-2, switches support STP (Spanning Tree Protocol), TRILL (Transparent Interconnection of Lots of Links) [8], etc. and at layer-3, routers support Open Shortest Path First (OSPF) [9] protocol, Routing Information Protocol (RIP) [10], Internet Control Message Protocol (ICMP) [11], etc. The network operator cannot implement a new routing protocol in the control plane of the existing devices. Thus, traditional networks impede innovation [12].

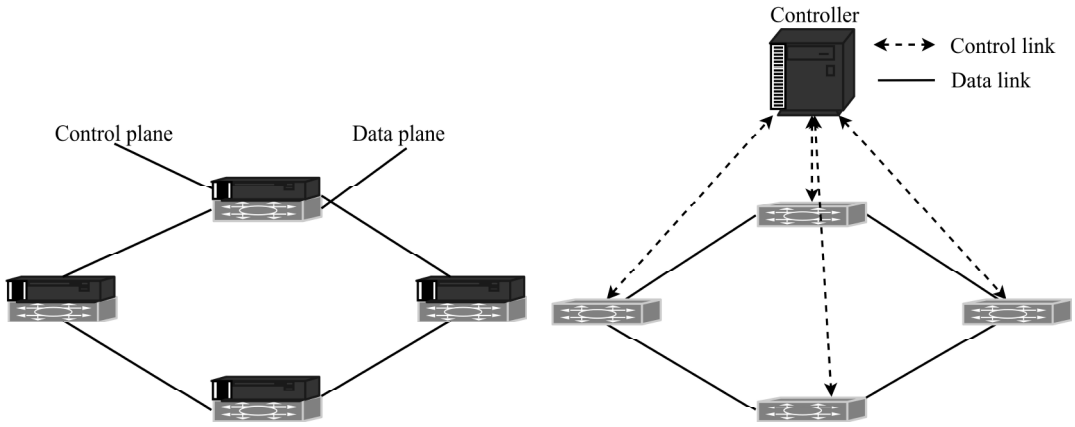
Another important aspect in networks is monitoring. Network monitoring is important because defining or changing network policies requires timely and consistent network statistics. Traditional tools such as tcpdump, JFlow [13], Netflow [14], and sFlow [15] use packet sampling and provide flow based measurements. These passive measurement tools are hardware based and need to be configured on individual devices in the network. Another problem with these passive tools is that they require access to the network devices, which can raise security concerns. Also, to collect the statistics, the network operator has to deal with heterogeneous devices and has to do per device configuration. Which can be a headache for the network operator.

## 1.2 Software Defined Networking

Software Defined Networking (SDN) is an emerging paradigm that provides unified control over the underlying network devices by decoupling the control and the data plane. In SDN, devices are programmable and are managed by a centralized controller. Open Networking Foundation (ONF) [16] defines SDN as follows,

*SDN is an emerging network architecture where network control is decoupled from forwarding and is directly programmable [17].*

Figure 1.1 shows the transition from traditional network to SDN. The control functionality is moved from the traditional devices, and logically centralized at the controller. In SDN, it is the controller's responsibility to manage the network by performing various functions such as, route network traffic, avoid congestion, handle network failures, network monitoring, load balancing, etc. The controller defines the network policies and communicates them to the switches. The switches are special purpose hardware devices




**(a)** Traditional Network **(b)** SDN Network

**Figure 1.1:** In traditional networks each switch has its own controller. Whereas, in SDN the controller is centralized and logically connected to each switch.

that implement the commands given by the controller in the form of forwarding rules. Each forwarding rule has two fields a match and corresponding action/s. These rules are stored in flow tables in the data plane. The action can be to forward the packet on an interface, drop the packet, modify the packet header field/s, or send the packet to the controller.

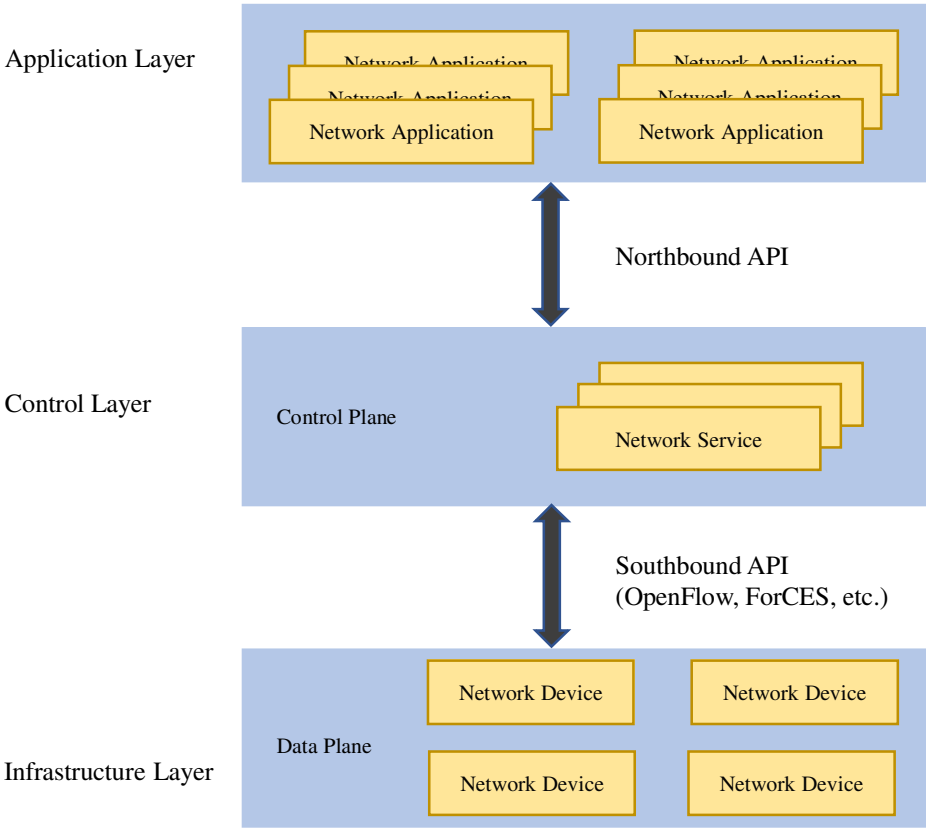



**Figure 1.2:** SDN Architecture

### 1.2.1 SDN Architecture

SDN architecture is shown in Figure 1.2. It comprises of three layers, Application layer, Control layer, and Infrastructure layer. The application layer allows the network operator to develop applications related to network security, network automation, network monitoring, etc. The infrastructure layer consists of multiple forwarding devices (i.e., SDN compatible switches). Switches in SDN have two major functions. One, they are responsible for forwarding the network traffic based on the rules defined by the controller. Two, they store the network status temporarily and send it to the controller. The control layer acts as a bridge between the infrastructure and the application layers. The communication between the control and the infrastructure layers is accomplished using the southbound Application Programming Interface (API). There are many options available for southbound API such as OpenFlow [18], NetConf [19], Forwarding and Control Element Separation (ForCES) [20] [21] [22] [23], Locator/ID Separation Protocol [24], etc. The most popular among these is OpenFlow [23, 5]. For the purpose of this thesis, we have used OpenFlow as southbound API. The communication between the application and the control layers takes place through northbound API.

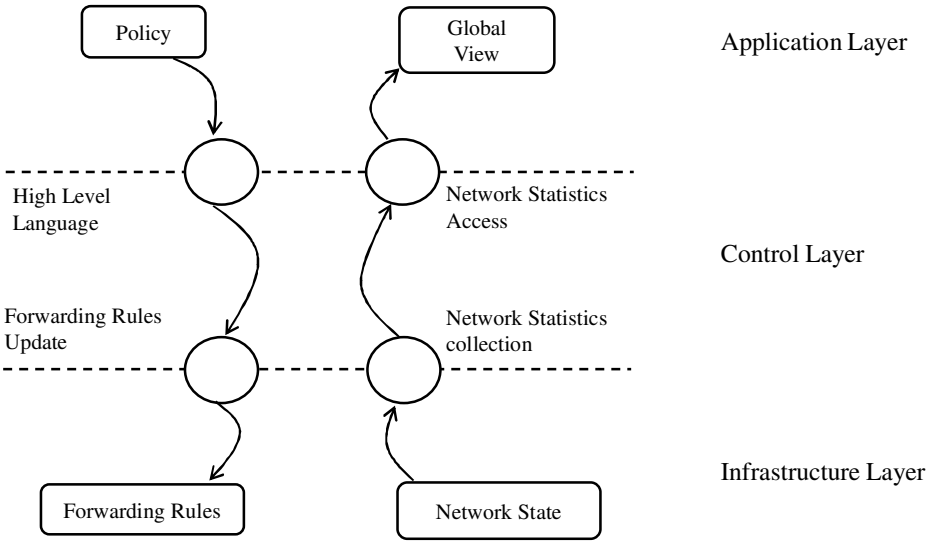



**Figure 1.3:** Logical Design of Controller

The control layer mainly deals with two functions. One is related to the translation of network policies, defined at the application layer, into packet forwarding rules for the infrastructure layer. Another one is related to collection of network statistics. It collects

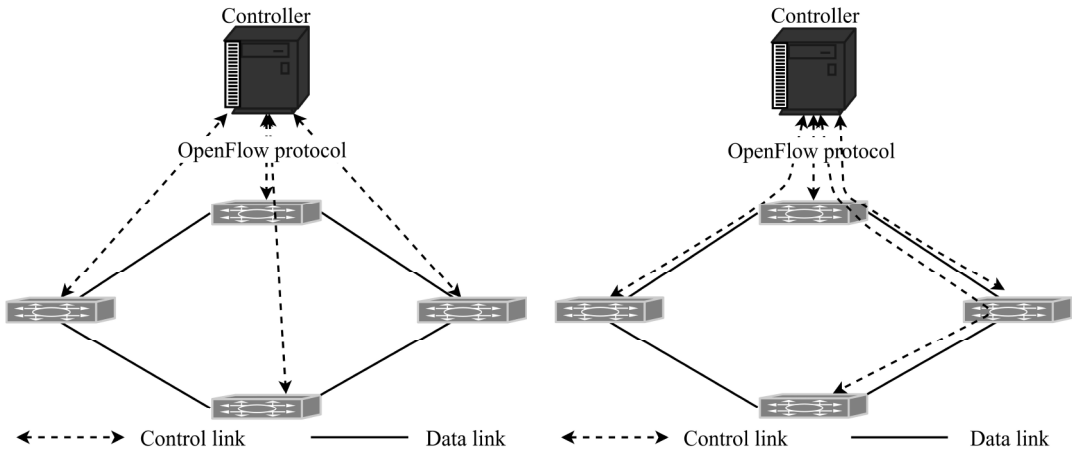
the network statistics, which can be used by the application layer for tuning network decisions. As shown in Figure 1.3, in the downward direction, the control layer takes the application requirements and translates them into forwarding rules for the infrastructure layer. In the upward direction, the control layer takes the network statistics and provides a network view to the application layer so that the application layer can define the network policies accordingly.

### 1.2.2 Controller Configuration

There are many options available for SDN controller such as, Ryu [25], Floodlight [26], POX [27], OpenDaylight [28], NOX [29], etc. A controller can be configured in two ways,

1. **Out-of-band Configuration:** In out-of-band configuration, the controller is connected to every switch through a dedicated link, as shown in Figure 1.4 (a). These links are called control links. Control links are used to send control messages between the controller and the switches.

2. **In-band Configuration:** In in-band configuration, the controller does not require dedicated links to every switch. Instead, it uses the data links (connecting two switches) to send control traffic, as shown in Figure1.4 (b).




**(a)** Out-of-band **(b)** In-band

**Figure 1.4:** Controller configuration

Both configurations have some pros and cons. Out-of-band is more secure compared to in-band, as it has a dedicated link to the controller [30]. But it requires additional

deployment cost and is difficult to manage. Whereas, in the case of in-band configuration, the control traffic is sent through the data links. Thus it has no extra deployment cost, but it can degrade the network's performance. In an in-band configuration, the link failure can affect both data plane and control plane communications. In the case of out-of-band configuration, if a dedicated control link fails, it disconnects the controller from a switch and affects the communication between them [31]. So, there exists a trade-off between in-band and out-of-band controller configurations.

## 1.3 Benefits of SDN

### 1.3.1 Easy to Configure

Prevalence of multiple devices in traditional networks poses a considerable challenge to the network operator. In traditional networks, each device has to be configured separately. In SDN, the unified control provides flexibility to the network administrator to programmatically configure all kinds of devices, for example, the controller can use southbound API for configuring each network device. This makes the network operator's job easier as she does not have to configure the network devices manually.

### 1.3.2 Global Topology View

When a switch joins an SDN network, it establishes a Transmission Control Protocol (TCP) connection with the external controller. When a connection is first established between the controller and the switch, they send OFPT_HELLO message to negotiate the OpenFlow version. Once the negotiation is done, a connection is established. Then the controller sends a FEATURE_REQUEST_MESSAGE to the switch as part of initial handshake, and the switch replies with a FEATURE_REPLY_MESSAGE. The FEATURE_REPLY_MESSAGE includes fields such as switch ID, number of active ports with their MAC addresses, number of tables supported, etc. [32] [33]. Link discovery is done by the controller using Link Layer Discovery Protocol (LLDP). An LLDP packet is encapsulated in a Packet-Out message and is sent to each active port of every switch in the network. The Packet-Out message installs a flow entry in the switch to forward the encapsulated LLDP packet through

the port mentioned in the Type Length Value (TLV) field of the LLDP packet [33]. When an adjacent switch receives the LLDP packet through a port different from the control port, it sends the LLDP packet to the controller as a Packet-In message. The Packet-In message contains information such as switch ID, port on which the switch received the LLDP packet, etc. [33]. Using this information, the controller is able to build the complete topology of the network. The controller maintains a database to store the network topology, which gets updated whenever a switch joins or leaves the network.

### 1.3.3 Minimization of Management Cost

In network management, network statistics play an important role in defining network policies, QoS assurance, load balancing, etc. Unlike traditional networks, SDN does not require specialized tools for statistics collection. In SDN, every switch maintains counters for every interface and flow entry in the switch. The controller can poll the switches by sending the statistics request messages, and the switches send back the corresponding statistics reply. The statistics can be polled at different granularity like per-flow statistics, aggregate flow statistics, per-port statistics, and queue statistics. Thus, the network operator gets rid of installing and configuring of monitoring tools.

### 1.3.4 Flexibility

In traditional networks, the switches and routers support a set of pre-defined protocols. Traditional devices do not provide the flexibility to change the behavior of existing protocols. For example, TCP/IP provides best-effort delivery, and it does not guarantee QoS. Security is not an inherent part of TCP/IP. To solve these issues, overlay solutions are provided. Also, the network operator cannot implement a new routing protocol or solution in the traditional devices [34, 5, 35, 36]. SDN provides the flexibility to change the policies and algorithms on the fly to adapt to the dynamic needs of the end-users. In SDN, the network operator can implement a new routing protocol at the centralized controller, and forwarding tables of all the switches get updated as per the new protocol.

## 1.4 Contribution of the Thesis

Though SDN provides many advantages over the traditional networks, but many challenges still exists in SDN [37]. This section lists various challenges identified in SDN, their state-of-the-art, and identified research gaps. We also provide a brief description of the solutions proposed in this thesis to handle each of the identified challenges. The identified challenges can be broadly classified into two categories,

1. Efficient statistics collection in SDN
2. Waypoint enforcement in Hybrid SDN

### 1.4.1 Efficient Statistics Collection

We have identified three problems related to efficient statistics collection, that are very crucial for network management,

- Globally consistent statistics collection.
- QoS parameters measurement.
- Optimal polling frequency.

#### 1.4.1.1 Globally Consistent Statistics Collection

In SDN, the controller manages the underlying network dynamically. The underlying network consists of switches that are distributed geographically. Each switch maintains statistics counters for each flow, port, and queue. These statistics can be polled by the SDN controller using statistics request messages, and switches send the corresponding statistics reply to the controller. The benefits of a centralized controller can be realised if the controller can take management decisions dynamically based on the current state of the network. Since the switches are distributed and have varying distances from the controller, it becomes very important to maintain consistency in network's statistics collection. There exists a solution, SpeedLight [38], that provides a mechanism to collect consistent statistics in P4 [39] based networks. Though SpeedLight does not always provide consistent statistics. It has to recollect the statistics in case it finds inconsistencies in the collected statistics. Thus, the solution is not efficient.

We propose two methods, OpenSnap and GlobeSnap, to collect consistent statistics in OpenFlow based SDN networks. OpenSnap takes Chandy-Lamport algorithm [40, 41] as a base to design a new algorithm for consistent statistics collection. It assumes that the switches are connected with First-In-First-Out (FIFO) channels, that is the packets are transmitted based on their order of arrival. It also assumes that the network does not contain any loop. In OpenSnap, the order of statistics collection is maintained using marker packets. The controller initiates the statistics collection by sending a marker packet to one of the switches. The switch sends the statistics to the controller and broadcasts the marker packet. Since the network does not contain any loop, the statistics collection algorithm terminates once the controller receives statistics from all the switches. For a network with Non-FIFO channels, it requires multiple runs to collect the statistics consistently. To improve the efficiency and robustness of OpenSnap, we propose GlobeSnap, a time-efficient and robust method to collect globally consistent statistics for OpenFlow networks. It provides consistent statistics in a single run even for a network with Non-FIFO channels. GlobeSnap does not require any additional marker packet for statistics collection. Instead, it uses the network traffic itself to ensure consistency.

#### 1.4.1.2 QoS Parameters Measurement

Over the years, there is a huge increase in multimedia applications such as VoIP (Voice over Internet Protocol), video conferencing, online gaming, etc. It is the responsibility of the network operator to meet the SLA (Service Level Agreement) and guarantee QoS (Quality of service) for these applications. Improving QoS in turn requires an effective and efficient measurement of QoS parameters (such as bandwidth, delay, jitter, and packet loss) and routing of traffic based on these parameters. In literature, there exist multiple works to estimate the QoS parameters. In this thesis, we focus on measuring link delay.

Delay measurements can be categorized into two types, active delay measurement and passive delay measurement.

1. Active delay measurement involves sending a time-stamped probe packet through the datapath. For example as shown in Figure 1.5, the controller sends a probe packet to switch $S_1$ to measure the link delay from switch $S_1$ to switch $S_2$. The

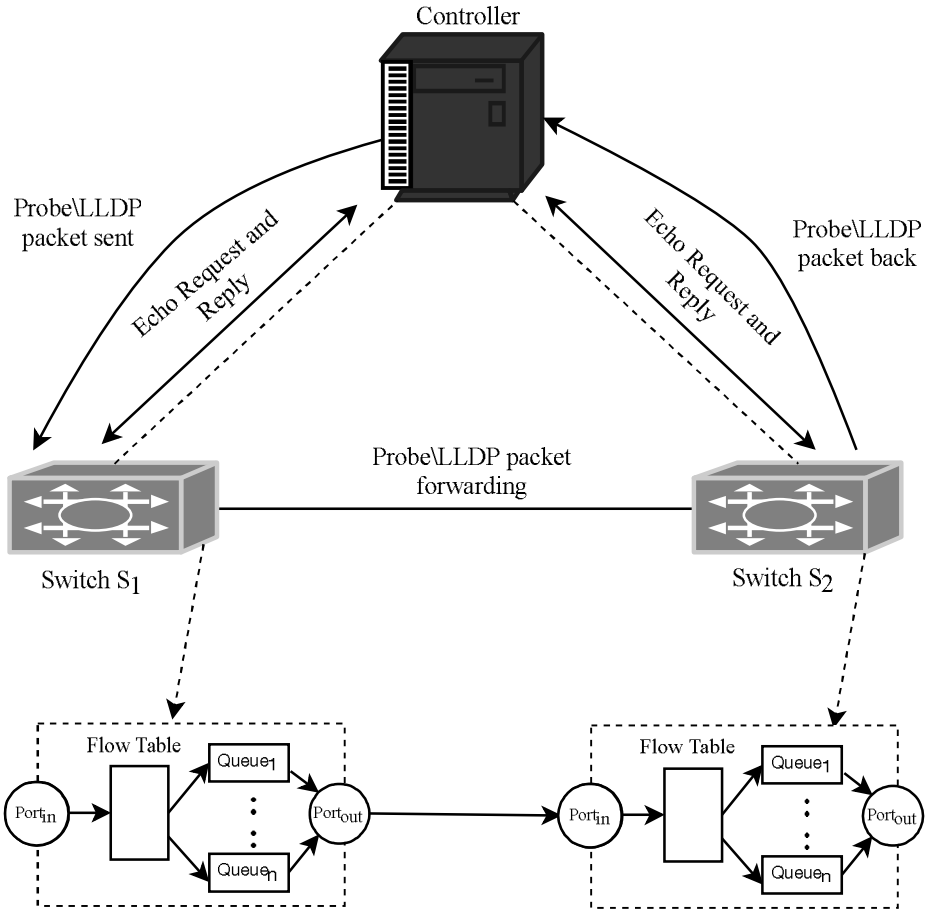



**Figure 1.5:** Mechanism to measure Link delay

controller guides switch $S_1$ to forward the probe packet to switch $S_2$. When the packet is received by switch $S_2$, it sends the probe packet to the controller as Packet-IN. The controller takes the difference between the arrival time of the probe packet from switch $S_2$ and the time when the controller sent the probe packet to switch $S_1$. This gives the total time taken by the probe packet through the link and back to the controller (let's say $T_{total}$). To measure the link delay, the delay from the controller to switch $S_1$ (let's say $T_{cntltoS_1}$) and switch $S_2$ (let's say $T_{cntltoS_2}$) is subtracted from $T_{total}$. Controller measure $T_{cntltoS_1}$ and $T_{cntltoS_2}$ using Echo monitoring. It sends Echo request messages to both the switches to measure their respective RTT (Round Trip Time) from the controller. To get $T_{cntltoS_1}$ and $T_{cntltoS_2}$, it takes half of the RTT of the corresponding links. The formula to calculate the link delay is,

$$Delay_{S_1toS_2} = T_{total} - (T_{cntltoS_1} + T_{cntltoS_2}) \tag{1.1}$$

Most of the existing methods given in [42, 43, 44, 45, 46, 47, 48, 1, 49, 50, 51, 52], use probe-packet based method (discussed above) to compute per-link and path delays.

2. Passive delay measurement methods measure delay by observing network traffic. They do not require additional probe packets to measure delay. Instead, the network statistics are polled by the SDN controller. To the best of our knowledge, not much work has been done on passive delay measurements in SDN. In [53], authors propose a solution to measure average queueing delay. But their solution works only for TCP flows.

   Active delay measurement methods are not efficient as they suffer from,

   - Data plane footprint, due to injection of probe packets into the data plane.
   - Monitoring overhead, due to the processing required at the controller to create probe packets and receive them.
   - Scalability issues, due to increase in the number of probe packets required with the increase in number of switches, links and queues at the egress path.

   Also, there is no passive delay measurement method that can provide a good approximation of link/path delay for real-world network traffic.

   Assuming that queueing delay is a significant contributor to variations in link delay. We propose an efficient passive delay estimation method, qMon, to monitor queueing delay in OpenFlow networks that leverages a queue statistics OpenFlow message. The controller polls queue statistics from Open vSwitches [54] at regular intervals. And then, using queueing theory, it estimates the average waiting time of packets. We further use the obtained queueing delay to measure the link/path latency. The proposed method solves the issues related to active delay measurement methods, stated earlier. It depends entirely on the queue statistics messages from the switches and, as a result, has zero data-plane footprints. qMon is scalable with respect to number of switches in the network, it requires only one queue statistics request message per switch to measure the queueing delay. Also, it is exempted from probe packet construction, thus monitoring and processing overheads are reduced.

#### 1.4.1.3 Optimal Polling Frequency

Statistics collection is important for network monitoring. There exist a few solutions to collect network statistics such as OpenTM [55], Planck [56], FlowSense [57], CeMon [2], MoMon [4], etc. These solutions can be grouped into two categories, push-bashed approaches such as Planck [56] and FlowSense [57] and pull-based approaches such as OpenTM [55], CeMon [2], and MoMon [4].

In push-based approaches, the controller does not request the switches for statistics instead, the switches send the statistics to the controller. OpenFlow based switches do not have the capability to decide when to send the statistics to the controller. The researchers have provided workarounds for this, for example, FlowSense [57], and Planck [56]. FlowSense [57], sends the statistics to the controller in FlowRemoved message when a flow expires. This does not provide real-time statistics as there will be a sparse number of FlowRemoved messages in case of a large number of elephant flows. Planck [56] uses port mirroring to monitor the network traffic. Mirroring of network's traffic makes this solution less secure.

In Pull-based approaches, the controller collects the statistics by sending the statistics request messages to the underlying switches. The controller can request the network statistics at different granularity by polling the underlying switches. The polling rate can be constant or can be adjusted dynamically by the controller. Polling at a higher rate provides better accuracy but also increases the polling overhead. Whereas polling at a lower rate reduces the overhead but might decrease the accuracy. For a dynamic polling rate, it is vital to determine an optimal polling rate so that the controller can maintain the accuracy of the collected statistics with least polling overhead. Existing pull-based solutions, CeMon [2], and MoMon [4], consider the rate of change in measured statistics as a metric to update the polling frequency. They increase the polling frequency if the change in measured statistics is more than a threshold. In case the measured statistics changes linearly, they might increase the polling frequency. Increasing the polling frequency for the above case does not add much to the accuracy but increases the polling overhead. If the change in measured statistics is linear, the change in rate of change is zero. Thus, polling at a lower/same rate would also provide the same accuracy. Taking this as a

motivation, we propose a new *pull-based* method that adjusts the polling rate depending on the change in the curvature of the polled data. The proposed solution provides the best trade-off between the accuracy and polling overhead compared to the state-of-the-art methods.

### 1.4.2 Waypoint Enforcement in Hybrid SDN

Over the past few years, SDN has evolved to give a new direction to computer networks. It has been widely adopted by companies such as Google to maintain their backbone network infrastructure [58], NTT to provide automated cloud-gateway [59], Microsoft to achieve high utilization [60]. It offers many advantages over traditional networks such as flexibility, fine-grained control on flow scheduling, global view of the network, etc. These benefits and adoption of the SDN network by the enterprise give the motivation for SDN deployment.

The transition from traditional networks to SDN cannot happen overnight. There exist a few challenges in this transition. First, it requires a considerable amount of investment to buy SDN hardware. In addition to this, when it comes to architectural updates, it requires hiring experienced SDN programmers to design, debug, update, and operate the SDN network [61, 62, 63]. Second, even after complete deployment, it requires some time to build a production-level SDN controller. Third, SDN implementation is not stable yet. Different researchers propose different design schemes that have led to lack of standardization in SDN. Therefore, a direct and sudden transition from traditional networks to pure SDN seems unlikely.

A hybrid deployment of SDN can be one of the plausible intermediate paths, primarily because it provides an environment where both legacy and SDN nodes can work together. Thus, an incremental deployment strategy to deploy SDN would be the right choice. Further, Hybrid SDN can provide benefits of both the traditional networks and the SDN paradigm.

*Hybrid SDN network is defined as a network which consists of both traditional and SDN devices [64], as shown in Figure 1.6.*

In literature there exist many solutions for incremental deployment of SDN like [65, 66,

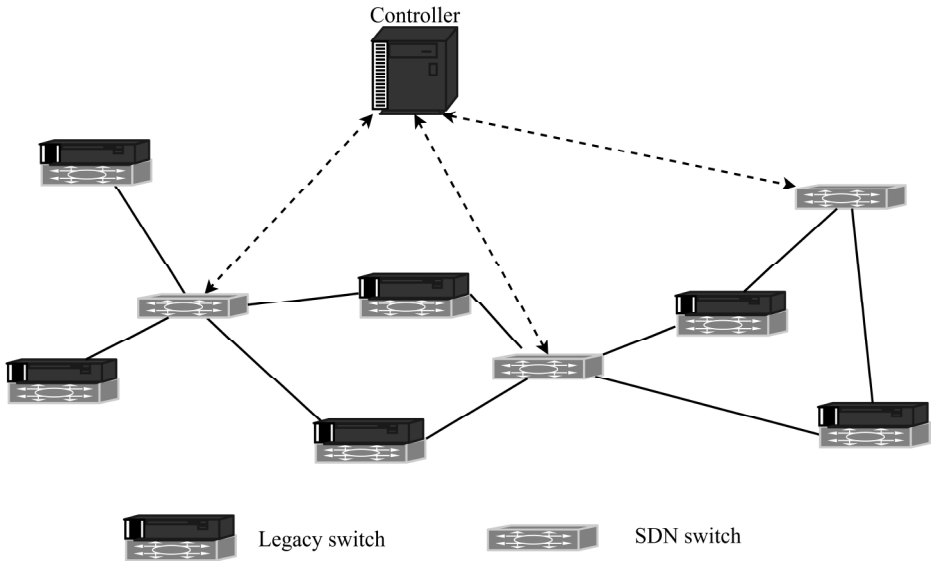



**Figure 1.6:** Hybrid SDN network, traditional and SDN switches co-exists

67, 68, 69, 70, 71]. We discuss all these works in detail in Chapter 6. It has been shown that incremental deployment of SDN provides better network management, traffic engineering [72, 73, 72, 74]. Deployment of SDN switches in the network alone cannot help much until the network traffic pass through an SDN switch. To leverage the power of SDN, traffic has to be diverted through an SDN switch. This is defined as waypoint enforcement in the literature [68]. A varying degree of waypoint enforcement can be achieved using methods proposed in [66, 67, 68]. Once a packet reaches an SDN switch, it is subjected to the security policies installed by the network controller in the SDN switches. This control over packets is desirable because it helps to leverage the power of SDN. All the existing methods to achieve waypoint enforcement, provide only partial waypoint enforcement.

In this thesis, we propose a novel framework to bring the entire network's traffic under the SDN controller by redirecting the traffic such that every packet goes through at-least one SDN switch. Our method achieves full waypoint enforcement even with a single SDN switch. We utilize unused IP addresses as virtual IP addresses to achieve full waypoint enforcement. These virtual IP addresses are accessible only via SDN switch. The idea is, when a source wants to communicate with a destination in the network, the source is provided with the virtual IP address of the destination. And the network is configured such that any packet with a virtual IP address is forwarded towards an SDN switch.

## 1.5 Other Challenges in SDN

This section lists a few more challenges in SDN other than the one discussed in the previous section.

1. *Scalability:* SDN architecture pushed all the control functionalities to a centralized controller. It provides flexibility to develop new control applications and makes policy enforcement easier [75]. As the network size increases, the load on the controller also increases and it can become a bottleneck. As a result, researchers developed many solutions at the control plane and data plane to improve SDN scalability.

   Researchers have proposed many high performance controllers like NOX-MT [76], Maestro [77], Beacon [78], Kandoo [79], Maple [80]. A performance evaluation of these controllers is given in [78]. These controllers mainly use well-known techniques from high performance computing, computer architecture, and networking such as pipelining, buffering, and parallelism to improve control plane performance. To improve the scalability, it is suggested to distribute the logically centralized controller [23, 81]. Kandoo [79], Onix [82], HyperFlow [83] have used this approach to scale the control plane. Distribution of controllers also has issues like inconsistencies in decision making, controller placement problem, and load distribution among the controller. In [84], the authors propose a solution for controller placement problem by considering the latency between the controller and switches as metrics. In [85], the authors propose ElastiCon, an architecture that dynamically increases or decreases the controller pool based on the traffic conditions. For better utilization of controller resources, some switches' control is dynamically migrated from one controller to another. In [70], two separate controllers are used in a network, edge controller, and fabric controller. The Edge controller controls the ingress and egress switches, whereas the fabric controller controls the fabric switches.

   DevoFlow [86] and [87] reduce the control plane overhead by delegating a few works to data plane. In DevoFlow [86], forwarding decisions for short flow are taken locally by the switches. If a switch identifies an elephant flows, it is forwarded to the controller, and the controller provides a least congested path for the flow. In [87], the counters are removed from ASIC and shifted to a general purpose CPU.

Thus improve programmability. To reduces the controller overhead, DIFANE [88] proactively pushes the flow entries to the switches.

2. *Northbound API:* The communication between the controller and switches is well researched and is defined in protocols [23]. But the communication between the application layer and controller is not standardized yet. The controllers such as Opendaylight [28], FloodLight [26] Onix [82], and Nox [29] have developed their own Northbound APIs [12, 89, 90]. It is not easy to propose a single northbound API because each application has different requirements [12]. To address this issue the discussions about standardization of northbound API have already begun [91, 92, 89, 93, 94, 95, 96]. Several proposals have been initiated like Nettle [97], Frenetic [98], NetCore [99], Pyretic [100], Procera [101], NetKAT [102], etc.

3. *Security:* In any network, security is a top concern because compromising a network component could be very damaging. An adversary may steal sensitive information or even bring the whole network down. Cyber-attacks have become the top concerns around the globe [12, 103, 104, 105, 106, 107]. Security is not an inherent part of SDN architecture. Thus, it is vulnerable to attacks. A list of common threats is given in [108], which also includes threats in SDN networks. OpenFlow based SDN networks are subjected to various dependability and security issues like tampering [109], spoofing [109], repudiation [109], denial of service [109, 110, 111].

4. *Resilience:* In an SDN network, it is the responsibility of the controller to ensure full-time service even in case of a failure. OpenFlow 1.2 provides the provision of master-slave controller configuration to ensure resilience in case of a controller failure [75]. [112] lists various challenges in designing a resilient network. Many works have been proposed in the literature to achieve resilience [75]. The proposed works mainly use network overlaying [113, 114, 115], replication [116], multi-path networking [117, 118, 119], and multihoming [120] to ensure resilience.

## 1.6 Thesis Organization

The thesis is divided into two parts, part I is dedicated to pure SDN, and part II is dedicated to Hybrid SDN. A pure SDN network consists of only SDN compatible switches and SDN controller/s. In part I we have four chapters.

- The first two chapters are dedicated to consistent statistics collection in OpenFlow based SDN networks. In Chapter 2 on page 21, we propose OpenSnap, an algorithm to collect consistent statistics in OpenFlow based network where switches are connected through a FIFO link/channel. In Chapter 3 on page 41, we present, GlobeSnap, a time-efficient, robust, and synchronous method to collect globally consistent statistics for OpenFlow based networks. GlobeSnap collects consistent statistics for all flows in a single round and is, therefore, time-efficient. Moreover, GlobeSnap is robust since it resumes the statistics collection process from where it left in case of an interruption. GlobeSnap also provides a near-synchronous snapshot of statistics of the switches traversed by a given flow. We also propose a mechanism to persistently store switch states in OpenFlow based networks using registers, multiple flow tables, and multiple pipelines. We find that GlobeSnap outperforms the state-of-the-art approaches in consistency evaluation. We also compare all these solutions in terms of the percentage of consistency achieved. We define the percentage of consistency as the number of rounds providing consistent statistics out of the total number of rounds of statistics collection. OpenSnap [3] with Non-FIFO channels provides least consistency whereas, simple polling, CeMon [2], and OpenNetMon [1] provides 59.89%, 52.25%, and 43.19% consistent statistics respectively. Both OpenSnap with FIFO channels [3] and GlobeSnap provides 100% consistent statistics. Further, we present two use-cases that are sensitive to inconsistent flow statistics, that is, computing packet loss and identifying bottleneck links.

- In Chapter 4 on page 77, we propose an efficient passive delay estimation method, qMon, to measure link delay in SDN networks. Existing approaches to dynamically obtain delay measurements in SDNs, are based on calculating the transit time of a specially constructed control packet (probe packet) that travels through the data links. These approaches are not efficient as the probe packet injected into the data

plane incurs considerable overhead. Additionally, a separate probe packet is required to measure the delay of each queue if more than one queue is present on the egress port of a switch. Thus, these approaches are not scalable. The proposed method, qMon, leverages the OpenFlow protocol to obtain queue statistics from OpenFlow switches at regular intervals, which are further employed to estimate the mean queueing delay for each interval. Thus, the proposed approach differs from the existing active probing based approaches as no packet is injected into the data plane to measure delay. The results show that for poisson traffic and for bursty traffic with large ON intervals, the Round Trip Time (RTT) values estimated using qMon and ping utility demonstrate high correlation when the measured RTT value is considered as time-series data.

- In Chapter 5 on page 105, we propose a new method for dynamic polling, which provides the best trade-off between the accuracy and polling overhead. We use curvature-based sampling as the basis for our method and compute the change in rate of change of the polled statistics to decide the polling frequency. We use real traffic traces for the experiments. The experimental results demonstrate that the proposed method provides better accuracy compared to MoMon [4] and CeMon [2]. The proposed method achieves 23% accuracy over MoMon [4] and has roughly three to four times better accuracy than CeMon [2].

In part II, we have two chapters. The details of each chapter are given below,

- In Chapter 7 on page 151, we propose a method to achieve 100% waypoint enforcement in Hybrid SDN networks. The proposed framework leverages unused IP addresses as virtual IP addresses to divert the network traffic towards the SDN switches. In Chapter 6 on page 120, we provide a detailed survey on Hybrid SDN papers with different dimensions like co-existence of both the paradigm at the control plane only and co-existence of both the paradigm at control plane as well as on data plane. We also present the pros and cons of each.

The overall contribution and organization of this thesis is summarized in Figure 1.7.

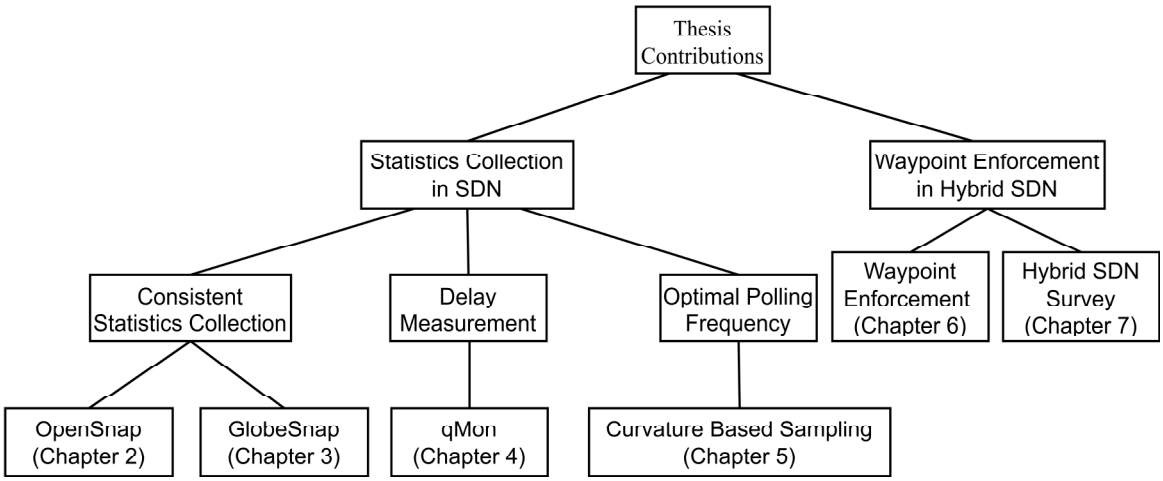



**Figure 1.7:** Thesis contribution and summarization