T3: SDN scalability and implementation challenges

Summary

<table>
<thead>
<tr>
<th>Aspects</th>
<th>Support points</th>
<th>Criticizing points</th>
</tr>
</thead>
</table>
| Controller design        | ● Group management of switches and hierarchical controller architectures can be used to ensure large-scale control over thousands of switches (if this may be needed and the controller CPU cannot cope, which is also doubtful).  
                            ● FPGAs are a valuable alternative to the CPU/ASIC dilemma. They can also offer an improvement in the # flow rules supported by a switch.                                                        | ● An evaluation on the controller hardware abilities to process flows should be carefully evaluated to establish bound on performance and supported # flows.  
                            ● Careful evaluation of all aspects from current networks that a controller should take over is needed.  
                            ● Flexibility of the NBI to the applications may bring in practical implementation challenges for the controller.                                    |
| Controller comm. delay   | The flow initiation delay and the controller-switch comm. delay in general can be fine tuned by deciding on the granularity of decision splitting between the control and data plane.                                      | The communication delay between controller and switches depend on the physical placement of the controller and could come at high penalty. Similar the speed of reaction to failures also depends on the controller placement. |
| Distributed controller   | ● Consistency can be inspired by approaches successfully applied in distributed systems.  
                            ● Peering between controllers handling their own OpenFlow domain can support distributed deployments. Each domain can be seen as a virtual single unit from outside.                    | Introduces consistency problems. Failure recovery from a controller failing needs moving state and re-informing switches about the new link. Mechanisms are needed for that.                    |
| Statistics collection    | *Detailed statistics collection can enable better informed decisions at the controller that match better the requirements of applications.*                                                                   | Stats collection from switches can heavily load the controller-switch link and should be addressed.                                                                                                               |
| Proactive rules          | ● Decreases the load on the controller-switch link, computational effort and flow initiation delay.  
                            ● Higher # flow rules at lower match cost can be handled with chained flow tables realized in RAM, avoiding the less efficient handling of TCAM.                               | ● Decreases the flexibility to handle individual traffic.  
                            ● Moving more responsibility to switches can solve some scalability issues but I will raises questions on what to be realized in CPU and what in ASIC. The tradeoffs between high performance (ASIC) and flexibility in behaviour change (CPU) should be carefully leveraged. |
| installation/Rule        |                                                                                                              |                                                                                                                                                                                                             |
| aggregation              |                                                                                                              |                                                                                                                                                                                                             |
Security & privacy

| Single controller is a single point of failure. Securing the controller is not trivial and it can be hijacked. Switches are sensitive to flow table exhaustion attacks. Moreover, there are no mechanisms to protect the network from misbehaving switch. Multi-tenant isolation between users/applications should be ensured in the flow rule creation. Securing the controller-switch comm. network is another challenge for authorization, privacy and security. |

Critique

Pappas, Christos

1. The "solutions" proposed as SDN scalability remedies actually just introduce tradeoffs that harm other useful properties of SDN. For instance, proactive installation of rules reduces the number of requests to the controller, which in turn reduce the processing load on the controller and the flow initiation overhead. This constrains flexibility and the more dynamic decisions that reactivity offers. Furthermore, aggregation of rules is proposed to reduce the flow initiation overhead and to reduce the flow table sizes. Consequently, such an approach prevents more fine-grained control, again limiting flexibility. In addition, a distributed controller is proposed to balance the computational load, but this introduces more consistency problems. Traditional networks are the extreme case where each switching node has a control-plane element and this distributed environment causes consistency issues. The other extreme case of one controller limits the consistency problems, but introduces scalability problems. Distributing the controller over fewer nodes trades consistency properties for scalability. When scalability solutions are proposed, these tradeoffs must be explicitly stated. Also, the issue of storing and managing all the statistics gathered from the forwarding devices is not touched; controller scalability has not only to do with the application processing demands.

2. The paper "On Scalability of Software-Defined Networking" makes many claims that are either unjustified or have logical fallacies. Hence, it is unclear why scalability problems are not a concern. For the flow initiation overhead: "Assuming controllers are placed in close proximity of switches, the controller-switch communication delay is negligible". Controller placement is a challenging research problem and this assumption neglects it. Regarding resiliency to failures and specifically when network partitioning occurs and there are multiple controllers which have to be discovered: "Therefore, given a scalable discovery mechanism, controller failures do not pose a challenge to SDN scalability". This sentence assumes a scalable mechanism to conclude that there are no scalability problems; this is not very convincing. Also, "In traditional networks, link failure notifications are flooded across the network, whereas with SDN, this information is sent directly to a controller. Therefore, the information propagation delay in SDN is no worse than in traditional networks." This depends on the network topology and the location of the failure, as well as the placement of the controller in the SDN network. The claim is too vague and not convincing.

3. Security is an easy attacking point against SDN, since many security concerns are raised and have not been addressed effectively (they are poorly described in the paper). Having a single controller means a single point of failure. Attacking the controller can cause a DoS or controller subversion means that an adversary can even take full control over the network. Having multiple controllers can eliminate the single point of failure, but it is unclear what the subversion of one controller means, how to detect it, and how to deal with it. Another concern is application security,
since a controller can server multiple applications. This problem seems more important in multi-tenant environments; for example, privacy concerns. Another problem is data plane security: First, the increasing state that flow tables introduce can cause state exhaustion attacks (e.g., flood a switch with random flows) that make the switch practically unusable for legitimate flows. Second, the trust model for the data plane is also unclear. How are the control plane decisions enforced in the data plane? For example, if a malicious switch does not forward traffic according to the control plane, how is this detected and how mitigated?

Shinde, Pravin

*Paper: On Scalability of Software-Defined Networking*

**Criticism**
- Talks about controller scalability issues in terms of flow initiation overhead and number of flows handled, but it's not clear how much time can controller spend in making decision about each flow for a dataCenter like setup, and how many flows are expected per second.
- The paper stresses that SDN gets benefits by having powerful CPU to setup new flows, or for adapt to a link failure. But capacity of a typical controller CPU is not mentioned.
- By reading this paper, it is not clear if we loose all benefits of SDN if we put a powerful CPU inside traditional switch.

**Arguments**
- Compares it with scalability of hardware based controller and exposes the issues with it due to limited capacity of controller CPU and limited communication bandwidth available between controller CPU and the ASIC.
- The paper describes in details all the steps involved in flow setup process and converging on the link failure.
- The paper compares SDN and traditional hardware switch based approach process for setting up flows and recovery from failures by observing the steps involved and guessing theoretical upper bounds on them to see how worse they will be.
- The paper also discusses how the environment like dataCenter and service provider network differs with number of switches and the latencies between them, and how one needs different SDN deployments for these different setups.
- The paper discusses a dimension of scalability from the perspective of manageability and complexity in extending the network, which is typically not discussed.

*Paper: Are we ready for SDN? Implementation challenges for Software-Defined Networks*

**Criticism**
- The paper seems to be written from commercial point of view, and is focusing more on showing importance of hybrid systems for SDN based of NPUs and ASICs.
- The paper also focuses on security concerns and discusses issues like DoS attacks by overflowing the buffers within switches by sending large number of new flows. Even though few of these concerns are valid, I think SDN is still very new and experimental to worry too much about these security issues.
- The paper discusses an hybrid approach in protocols as well to simplify the deployment of SDN in existing network with existing infra-structure, and how it can be deployed partially. I feel that this approach is again commercially motivated and will dampen the benefits of SDN due to the limited flexibility and usefulness of hybrid deployment.

Xinyuan, Yu

1. failure recovery
In the "on scalability of software-defined networking", the opinion has been raised that the failure recovery process is no worse than in traditional networks.

However, the failure recovery in SDN has its own difficulties. A slave controller needs to synchronize with master controller. Part of network may be left brainless. And the failed switch may affect the switch-controller communication. I think it is more difficult for SDN recovering from failure. As a result, scalability is more difficult in SDN than traditional network.

2. hybrid infrastructure
In the paper of “Are We Ready for SDN”, it is mentioned that a hybrid SDN infrastructure is required to achieve.

However, a hybrid infrastructure may bring more problems. For example, some routing decisions between MPLS and SDN may conflict, and circles may form. I think whether traditional, SDN- enabled, and hybrid network nodes can operate in harmony is still a question.

3. different architecture
The API between control plane and data plane should be able to handle the needs of different architecture. This makes it difficult for SDN to implement into the existing network.

Miladinovic, Djordje

1. In the paper "On Scalability of Software-Defined Networking" it is claimed that "SDN, by itself, is neither likely to eliminate the control plane design complexity or make it more or less scalable.” I feel that this attitude is a bit pretentious and should be questioned. A relatively small number of controllers is carrying the burden of a whole traditional network and at the same time they have to be synchronized among themselves and control the data plane, which could be more complex.

2. Regarding scalability, it is important to pay attention how to offload the processing from a controller in charge of a certain subnetwork (a controller can be a part of a network of physically distributed controllers) to switches. There are methods like proactive rule placement, and short-lived flows but is that really enough on a huge scale?

3. To enable deploying of SDN technology on a large scale, there are many security issues to be resolved. DoS attacks, as well as unauthorized access attempts seem like serious threats.

Schmid, Stefan

1. SDN tries to be as flexible as possible. The API provided by a SDN controller and exposed to an application should reflect this flexibility and therefore might introduce a complex abstraction layer. The network operator depends now on a programmer or software developer to build applications, but the built application will then again limit how the operator can modify the network behavior. Since the SDN controller interface might not be simple enough for a network operator, he will depend on software developers and the software they provide, which makes the work flow more complicated and more expensive.

2. The forwarding hardware in SDNs, besides from the packet switching, now also has to be able to run the communication protocol and install rules for packet flows. To increase packet switching performance and increase throughput, ASICs can be used. These ASIC designs are getting more complex and complicated to build since they now have to provide more functionality. SDN protocols are extended and changed rapidly. This means, that also
the forwarding hardware needs to be designed in a more general way. Most of the time, general purpose CPUs are used which makes the platform more flexible but provides less performance and energy efficiency. This introduces a trade-off between flexibility and performance. Until a SDN protocol is "finished" people will build their system on top of flexible platforms and might never see the full potential of their system implemented in ASIC.

3. SDN uses a separate and isolated network between controllers and the forwarding hardware. This also means that an additional network needs to be managed and maintained. The network has to be protected and if different organizations want to work together a mechanism to regulate access control needs to be introduced. The network also needs to be fast enough to handle all controller requests and to update the network state between controllers. For a big network, the control network (for distributed control) might also get significantly complex. What hardware will be used for those networks?

Defense

Chothia, Zaheer

1. "logically centralized; physically distributed"
Fundamentally there is no reason the decision logic or state queries/updates should present a bottleneck. One could claim (as the first paper does): the controller can be scaled out just as any distributed system, for which there are established techniques to achieve coordination, consensus and consistency. The second paper references work which establishes this is feasible and demonstrate benefits. Even multi-site should not pose severe obstacles: the DB community have developed scalable geo-replicated storage systems (e.g. Spanner/ Mesa: several hundred queries/sec, >1M updates/sec, latency in ~10 ms range).

2. Performance
#flows handled by a switch
New OpenFlow spec allows a chain of multiple flow tables + not restricted to TCAM. "Modern switches and routers support forwarding information bases (FIB) in range of 64K to 512K entries"

#switches managed by a controller
One concern voiced is performance of packet processing in software; not a major issue => Intel DPDK with an 8 core Xeon can handle 255 Mpps (packets/sec) at 40 Gbps line rate.

Alternate designs are possible, e.g. (i) hierarchical "controller of controllers" or (ii) multiple autonomous systems + peering protocol (cf. IP/MPLS). Aggregation of flows switches is also a possibility but decreases granularity of control (although 'these concerns are not unique to SDN' and also apply to traditional networks).
REF: https://www.sdxcentral.com/articles/contributed/openflow-sdn/2013/06/

- As a counterpoint recall this presentation from the SWITCH workshop (Dec 2014). Current SDN switches lack atomicity and are slow as molasses for updates: ~40 rules/sec and several seconds to apply. This is on the critical path for flow setup and can be a dominating factor! (I suspect this will be resolved in the coming years.)
REF: http://wan.poly.edu/pam2015/papers/73.pdf

3. Other dimensions: performance is just one axis; others 'profoundly affect how a system can accommodate growth' => manageability, extensibility, evolution, testing/verification. SDN has reinvigorated interest in these topics (e.g. ndb: network debugger, Coq formal verification).
Also consider fault resilience: link failure in a traditional network requires flooding notifications and re-converging. With SDN this loss event is sent directly to the controller so the 'information propagation delay is no worse than before'.

4. (The observation on security is a scary prospect: SDN controllers will be hijacked; they're a treasure trove and a tremendous platform to launch DDoS attacks and wreak havoc!)

Lee, Tae Ho

1. Scalability is an important problem of SDN that needs to be thoroughly investigated if SDN were to be realised in large-scale enterprise and data-centre networks. It is important to have such papers that emphasise this fact and try to summarize the ongoing research efforts in this direction.

2. I find the following statement very interesting: “To preserve scalability one should design control applications with the weakest possible consistency level.” I wish the paper would provide more detailed examples by discussing the consistency level that the cited papers have proposed as well as their advantages and limitations due to their design choice. Even if the cited papers do not mention these aspects explicitly, the authors’ interpretation and analysis would be useful.

3. It is interesting to think that SDN provides another design space that is not possible in traditional network (i.e. more flexibility)—it allows to balance the granularity of the control by adjusting set-up latency time.

4. In mathematical optimisation problems, often times a dual problem is derived from the primal problems to solve the optimisation problems. (Also, the solutions would discuss any differences between the solutions to the primal and dual problems) Perhaps the paper could try to formulate the scalability arguments in the following manner: 1. define the objective of the problem including the requirements from the network. 2. Then it maybe helpful to show how the traditional and SDN approaches aim to solve the problem and explain any trade-off between the two approaches. Such approach may provide more structured method to compare the scalability problems.

5. At the current state of the paper, the scalability argument by comparing SDN to traditional network seems unconvincing and meaningless (e.g why and how does it help to know that scalability of SDN is no worse than the traditional network?)

van Gelder, Jasper

The main idea of SDN is clear: Decouple the control plane from the data plane so both components can evolve independently, which should open the market for innovation and thus improve the capability of our networks.

1. SDN offers great opportunities to improve the resilience of networks by implementing distributed controllers such as is done by Onix. In this way control requests can be spread over multiple controllers and it becomes much easier to extend the network. Although it is not perfect (and very hard to build distributed networks), it shows one of the advantages of decoupling the 2 planes.

2. I agree with paper [1] that there are a lot of security issues that should be tackled. I don't know enough about the current security models but when you decouple the data plane from the control plane you directly have extra attack vectors that you have to protect (controller<>application, controller<>controller, controller <> dataplane).
3. One other thing that could definitely help is the use of FPGAs it perfectly fits in the nature of a software defined networks and might be able to (partially) solve some of the scalability/latency problems that are mentioned. It atleast should offer some kind of performance gain over CPUs and provide more flexability then ASICs.

Birkner, Ruediger

1. Controller Scalability - In the initial SDN proposals, we always have one controller with a global network view. However for networks with many nodes and hosts, one single controller might not be up to the task. Nevertheless, several ideas try to address this problem and improve the scalability of an SDN setup:
   a. It is possible to deploy several controllers that all only control a subpart of the network. ONIX is a try at a distributed controller.
   b. To take some burden from the controller, it is possible to categorize flows according to their longevity and size. The controller will only handle those flows that are worth handling.
   c. By employing high performance servers and exploiting the multi-core systems through parallelism, the performance of controllers can be increased.

2. Node Scalability - Another bottleneck are the switches in the network. TCAMs that are used for the flow rules, are expensive and cannot store too many flow rules. Research has shown that FPGAs and their block RAM can be used to store many flow rules and still process the packets at line rate.

3. Failure Resiliency - Due to the fact that the controller or controllers have a network wide view, they are able to react to failures a lot quicker and reroute packets.