Software Defined Networking and Centralized Controller State Distribution Reduction – Discussion of one Approach

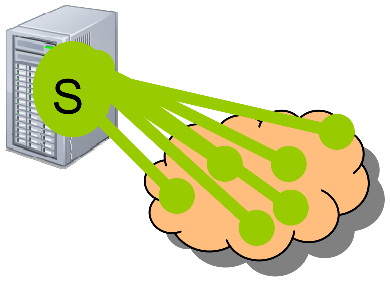
"Can 802.1 data plane changes improve SDN scale?"

Peter Ashwood-Smith

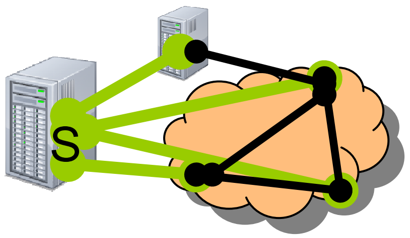
peter.ashwoodsmith@huawei.com

July 2012

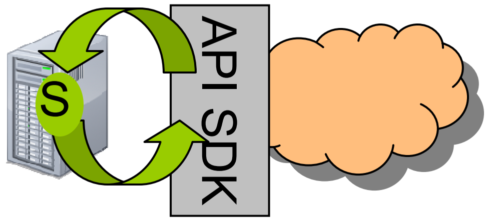
SDN in one slide – 5 flavors x many combinations...



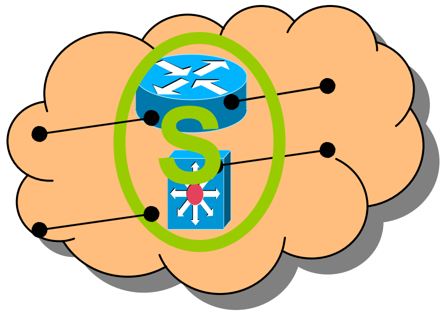
Software in server/network – controls every switch/router. Uses protocol like openflow. Two flavors, 1) ***re-active*** waits for packets before setting up flows. 2) ***pro-active***, pre-established routing. Network is physical. (Stanford, Google)



Software in server – 3) ***orchestrates*** attachment to network. Tunnel attachments connected to servers, routers, firewalls etc. Controls logical network. Physical network control unchanged. OpenStack etc. primarily DC environments for now.

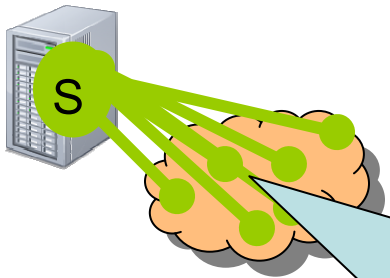


Software in server – part of 4) ***feedback*** loop to analyze, predict and act on network via API/SDK. Physical network control unchanged. Two major vendors describe this in their SDN solution.

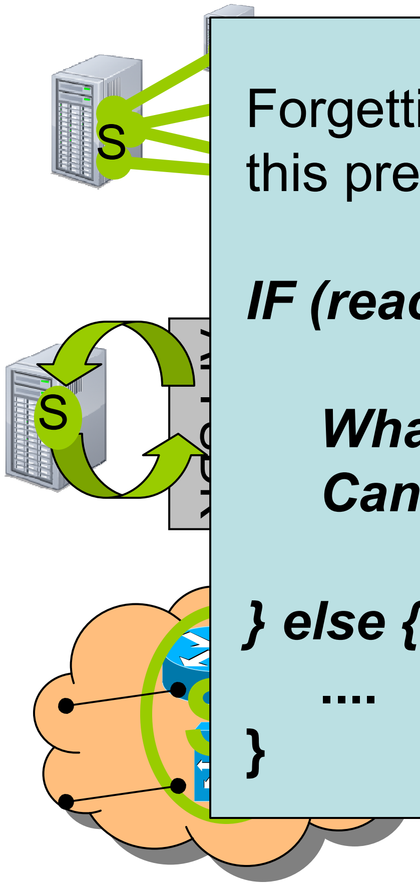


Software in server – 5) ***emulates*** complete router/switch or other network appliance. Slower than hardware but very flexible. Vswitch etc. Network can be logical or physical.

SDN in one slide – 5 flavors x many combinations...



Software in server/network – controls every switch/router. Uses protocol like openflow. Two flavors, 1) ***re-active*** waits for packets before setting up flows. 2) ***pro-active***, pre-established routing. Network is physical. (Stanford, Google)



tc.

d.

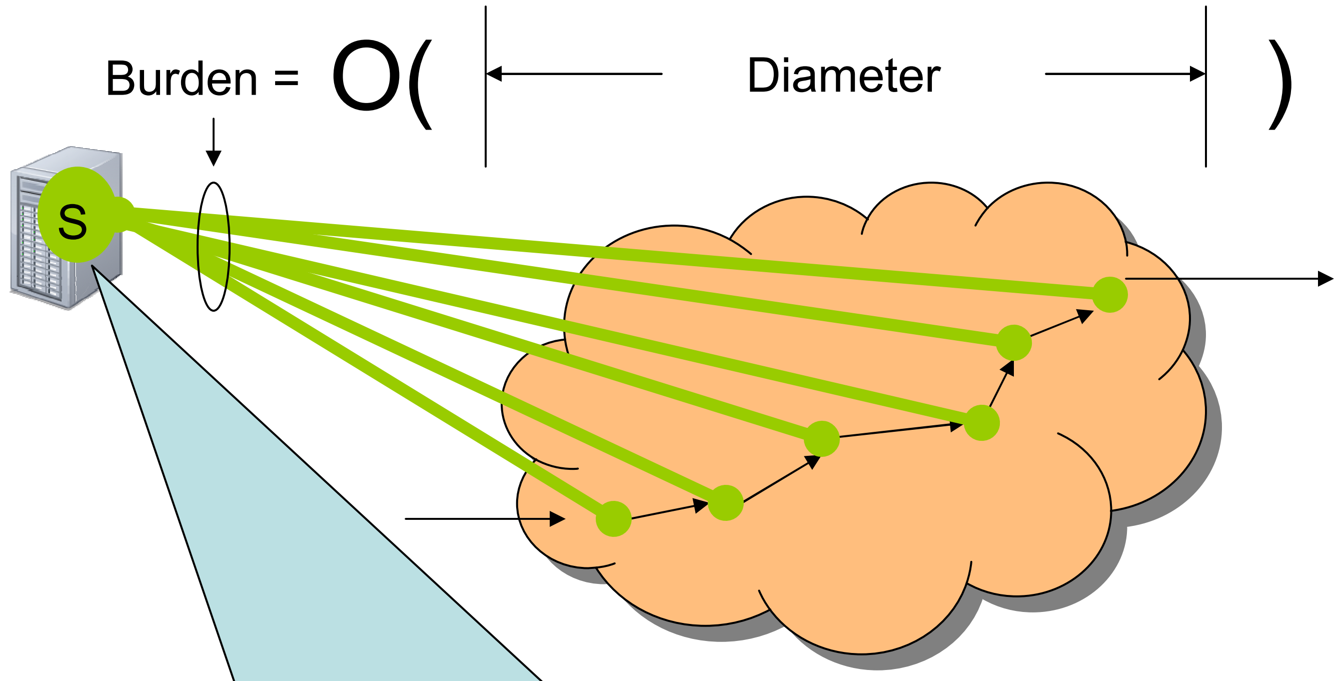
Forgetting all the other types of "SDN",
this presentation asks:

```
IF (reactive_pro-active_central_control == GOOD) {

    What can 802.1 do to data plane to make it even better?
    Can we improve its scalability?

} else {
    ....
}
```

Central Controller Some Scale Limits

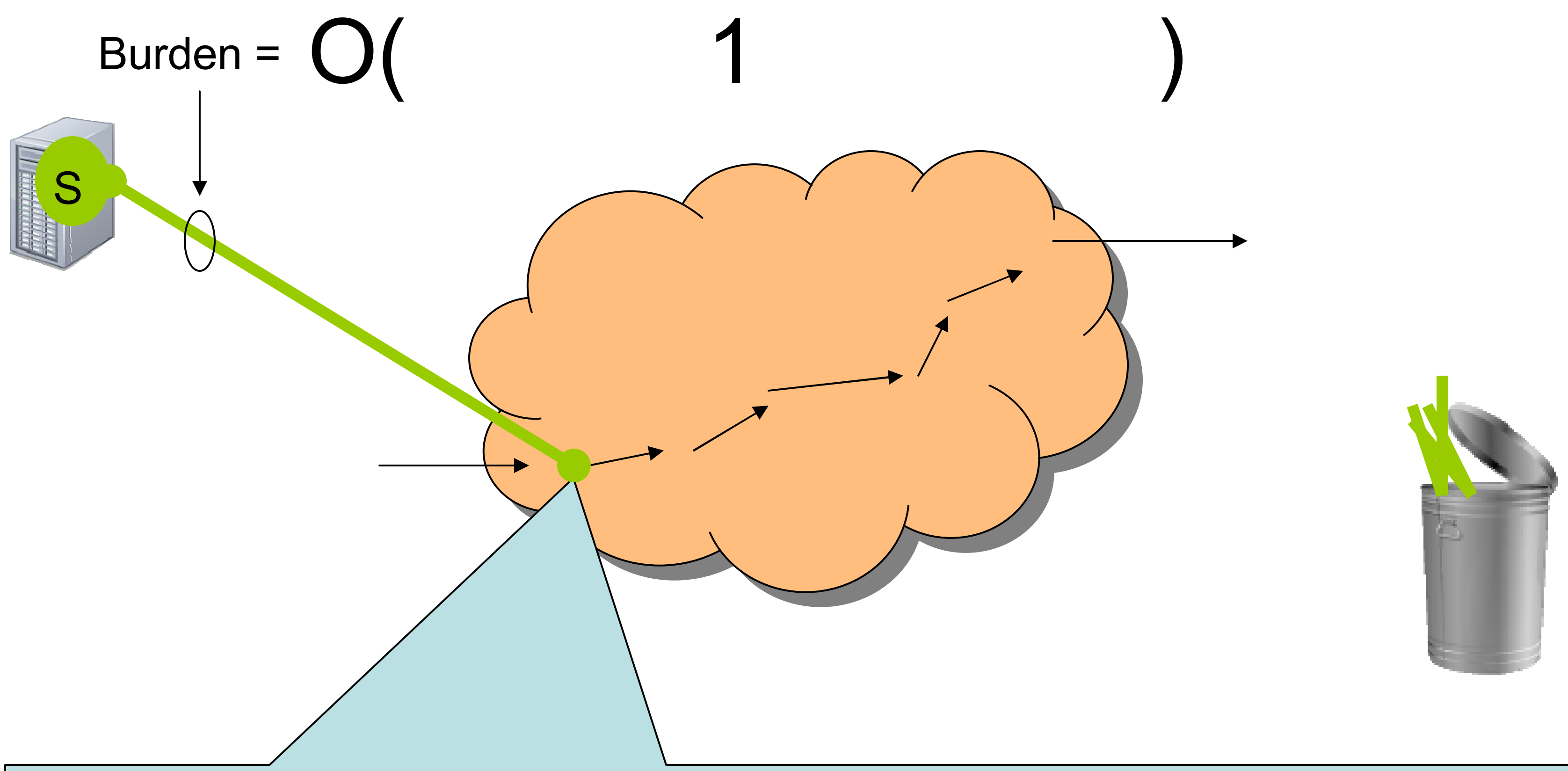


A controller has to send flow state to **every** device along the path.

Therefore as the diameter of the network grows the controller scale drops as its work/burden grows as function of **O(diameter)**

There are of course other problems but they are out of our control...

Tackle the distributed state problem by ...

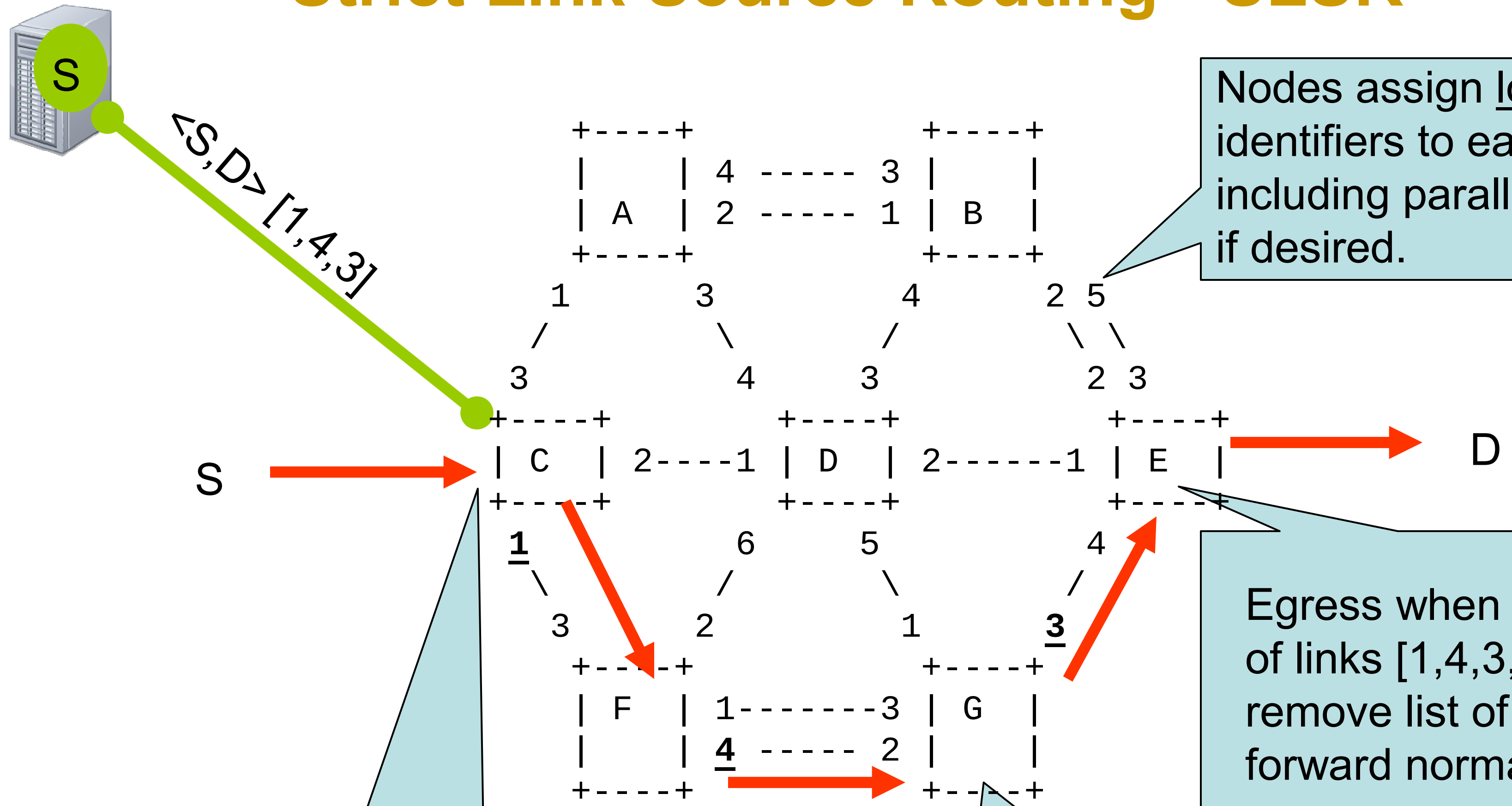


Install all state required to traverse network in only the ingress device.

Basically ingress switches attach a source route (nodes/links) to follow and tandem devices recognize and strictly follow it.

In particular **Strict Link Source Routing – SLSR** is a candidate

Strict Link Source Routing - SLSR



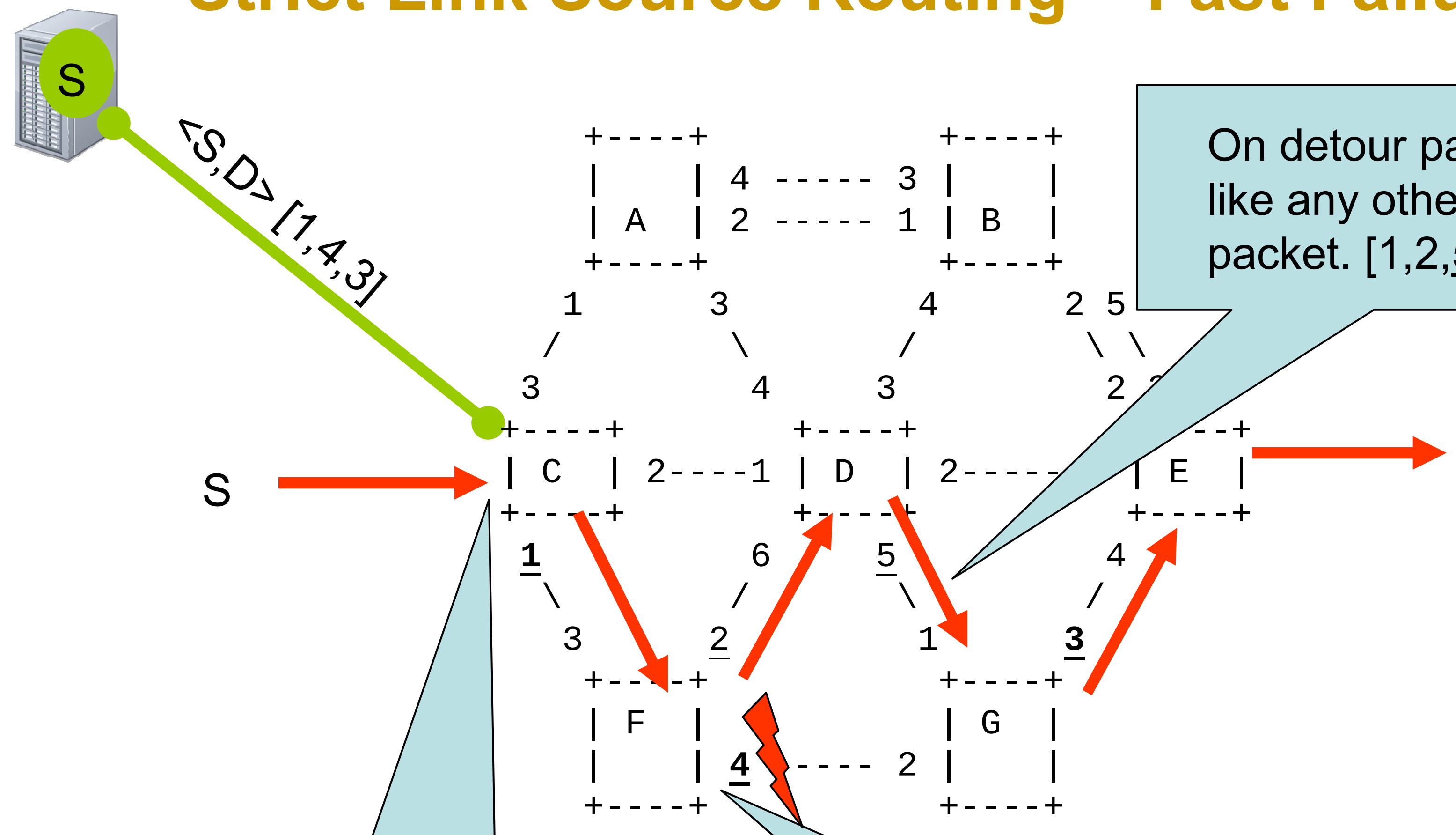
1 Nodes assign local identifiers to each link including parallel links if desired.

2 Ingress node, configured by controller installs <S,D> flow entry and header with list of links to follow [1,4,3]

3 Tandem nodes have no state related to path. Just take list [1,4,3], extract current next link 3 and forward to that link.

4 Egress when at end of links [1,4,3,_] remove list of links forward normally.

Strict Link Source Routing – Fast Failure



3 On detour packet looks like any other SLSR packet. [1,2,5,3]

2 Ingress node, configured by controller installs <S,D> flow entry and pushes list of links to follow [1,4,3]

1 Nodes assign detours as just list of links to get around a failed link..eg if 4 fails replace with [2,5]

SLSR

- **No per path tandem state**, controller scales greatly improved.
- **Local redirection** under failure similar to MPLS FRR possible (node/link detours).
- Forwarding to **link level** granularity.
- WYSIWYG routing.. Great for OAM.
- Very **fast** tandem routing, reduces high speed memory requirements (few hundred values v.s. 10’s thousands).
- Interesting options for multicast with ‘fragments’ of multicast tree encoded in header.....
- Theoretically possible to use MPLS or IPv4/6 but IP Source Routing deprecated and MPLS labels far too big and limited stack size limits diameter.

Theoretically this is a very simple and efficient idea. Why is source routing so frowned upon? Is it appropriate to resurrect it in the SDN context? Are people here interested in working on it? If so where?