THE GÉANT PROJECT COCO

Development of an SDN-based Virtual Private Network Service

Piotr Zuraniewski (TNO), Ronald van der Pol (SURFnet), Bart Gijsen (TNO), Marijke Kaat (SURFnet)

TNO innovation for life

COCO PROJECT OVERVIEW

- CoCo: Community Connection
- Goal: empower eScience community with easy VPN service
- Partners:
  - SURFnet: Dutch National Research and Education Network (NREN)
  - TNO: Netherlands Organization for Applied Scientific Research
- Sponsor: GÉANT – pan-European research and education network interconnecting Europe's NRENs
  - Sponsorhip via GN3plus Open Call programme
- Duration: Oct'13 – Mar'15

VPN IS NOT REALLY SOMETHIG NEW...

- Virtual Private Networks (VPNs) are around for ~20 years
- Number of technologies exist to assure private connectivity
  - MPLS, Q-in-Q, PBB,... + encryption
- It is enough to buy a specialized device (e.g. Cisco ASA)...
- ...and call network administrator to configure it

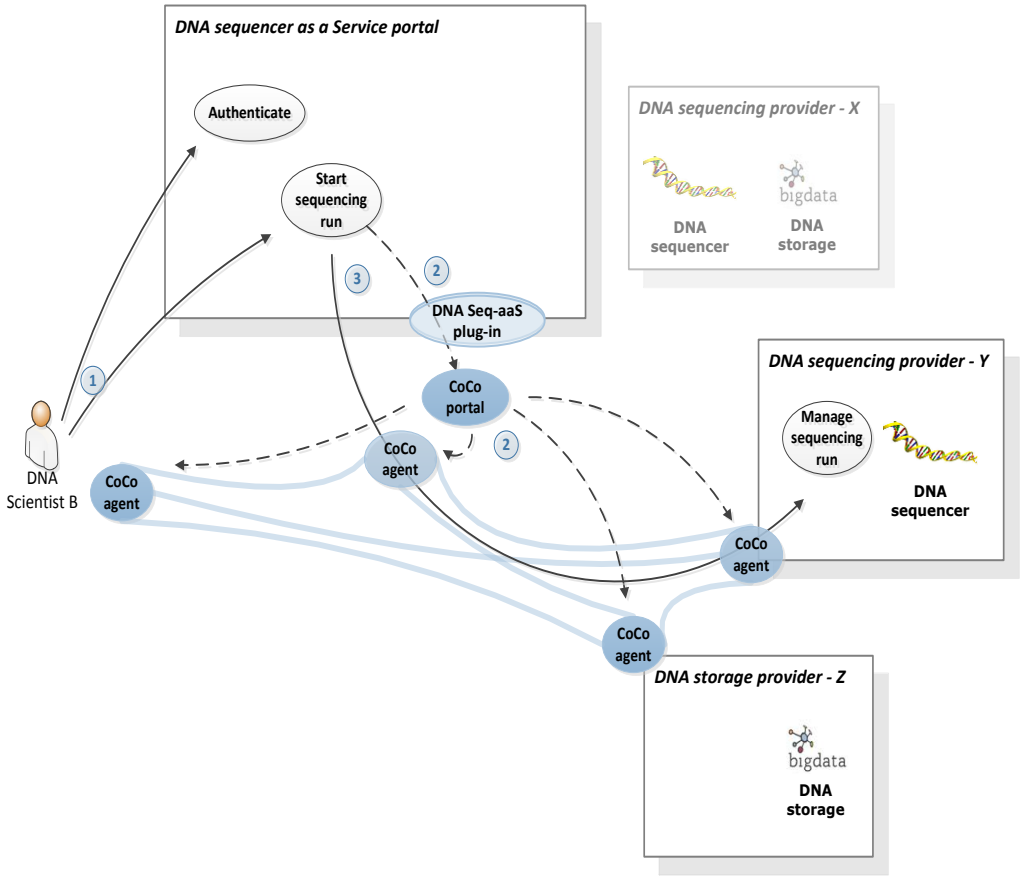
- When site needs to be added...
  - ...call network administrator to configure it
    - When site needs to be removed...
      - ...call network administrator to configure it
        - When end-point needs to be migrated...

NOW VPNS REQUIRE MANUAL CONFIG – UNSUITABLE FOR END USERS

- Currently, VPN configuration and maintainance requires network engineering knowledge and hands-on experience
- This is not a desired situation for end users
  - eScience community members may be experts in e.g., physics or biology
  - ...but not so much in network engineering
- Workshop with eScience community in Q1 2015 to learn about specific needs
  - eScience community members want to share facilities like:
    - scientific instruments
    - data processing
    - storage
  - ...without need for assistance by network administrator
  - ...at the affordable cost

COCO USE CASE: DNA SEQUENCER AS A SERVICE

- Workshop conclusion: eScience community interested in CoCo-like service
- Refined use case: DNA sequencer as a Service (BigData!) with Wageningen University
- Key objective: increase accessibility of multiple sequencers at different locations
  - Better utilization needed for acceptable return of investments



COCO: USER INITIATED, EASY TO USE, ON-DEMAND CONNECTIVITY SERVICE

User controlled provisioning of VPNs on OpenFlow switches using OpenDaylight.

Provisioned VPNs:

- vpn1 update
- vpn2 update
- vpn3 update

Create new VPN

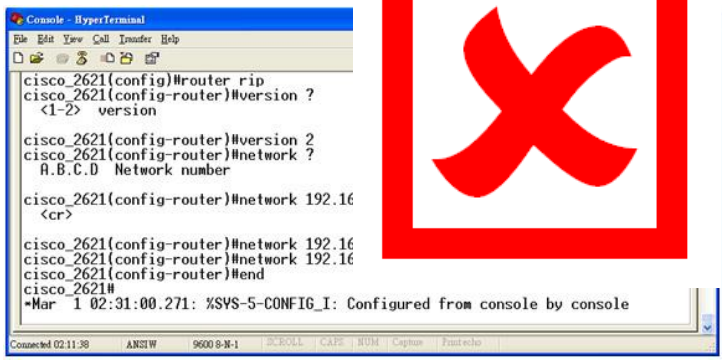
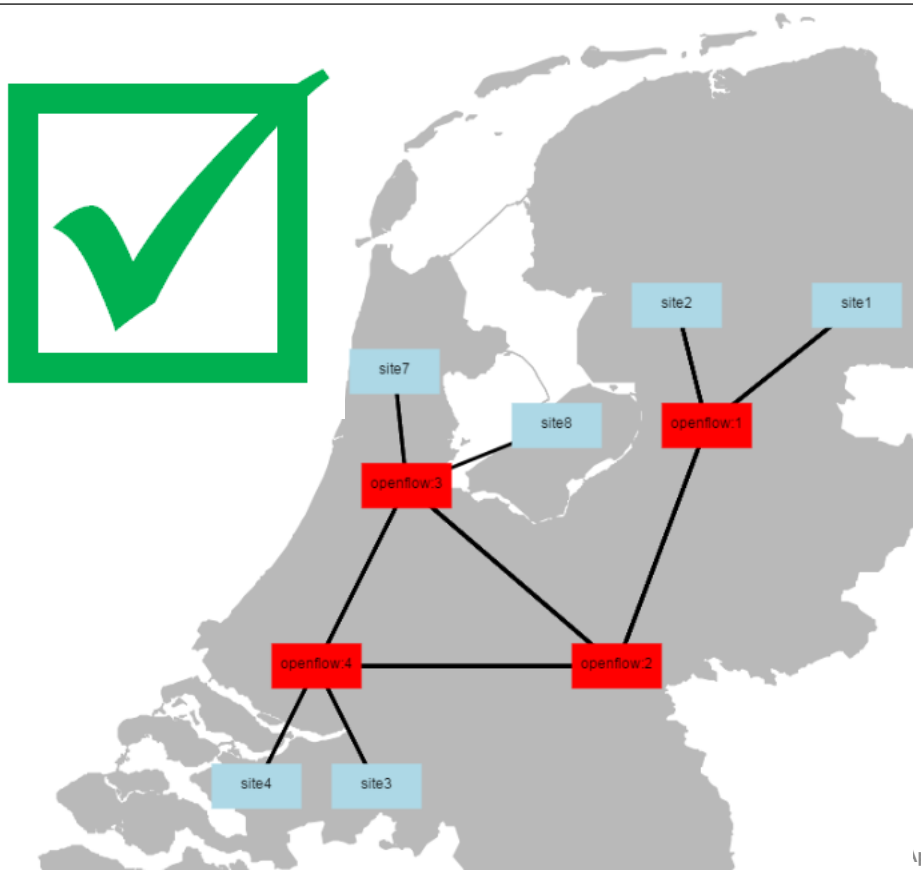
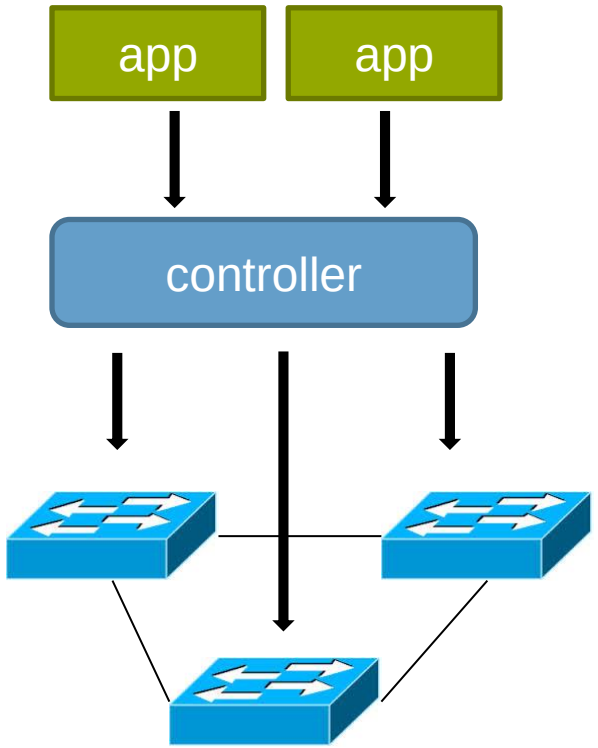


Image from http://www.geego.com/



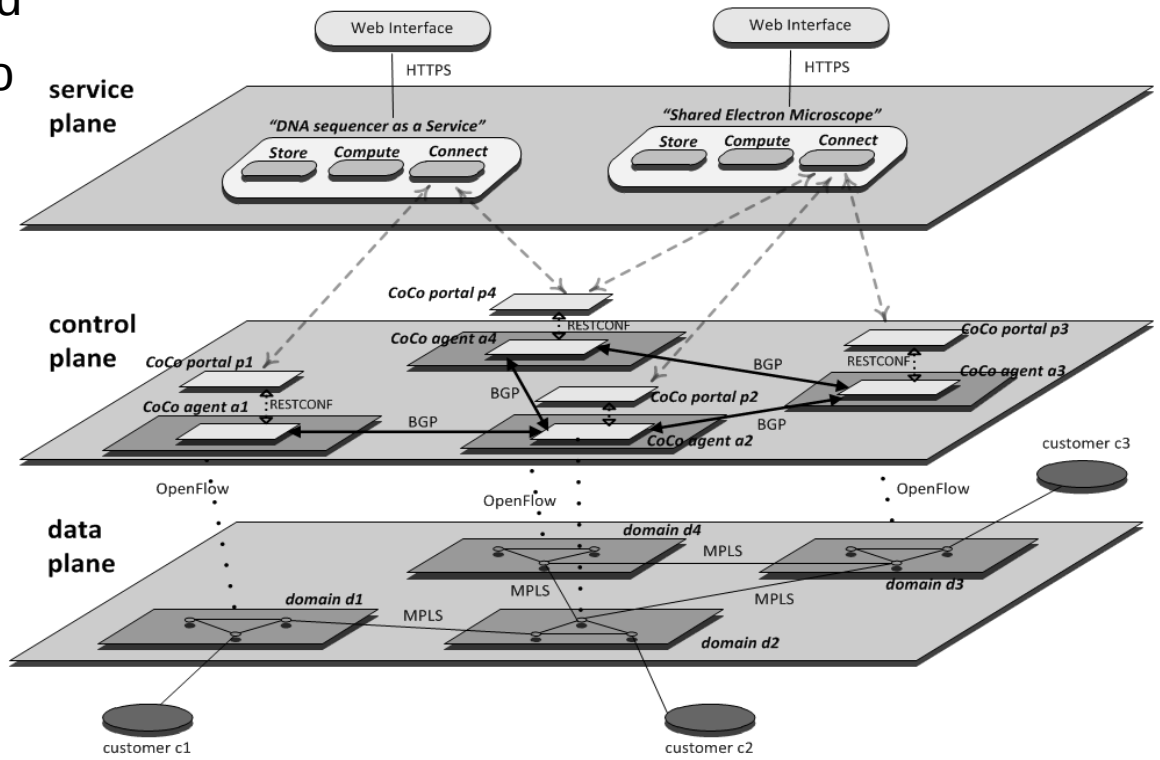
SDN AS FRAMEWORK TO REALIZE COCO

- Software Defined Networking framework allows for lots of freedom in creating and managing networks
- Application signals requirements to controller
  - „connectivity from site A to site B needed”
- Controller parses app requirements, translates it to form understandable by switches and sends appropriate commands
  - „Switch1: match in_port=1, action out_port=5”
- Switches install flows in their forwarding tables and move traffic
- No specialized hardware needed



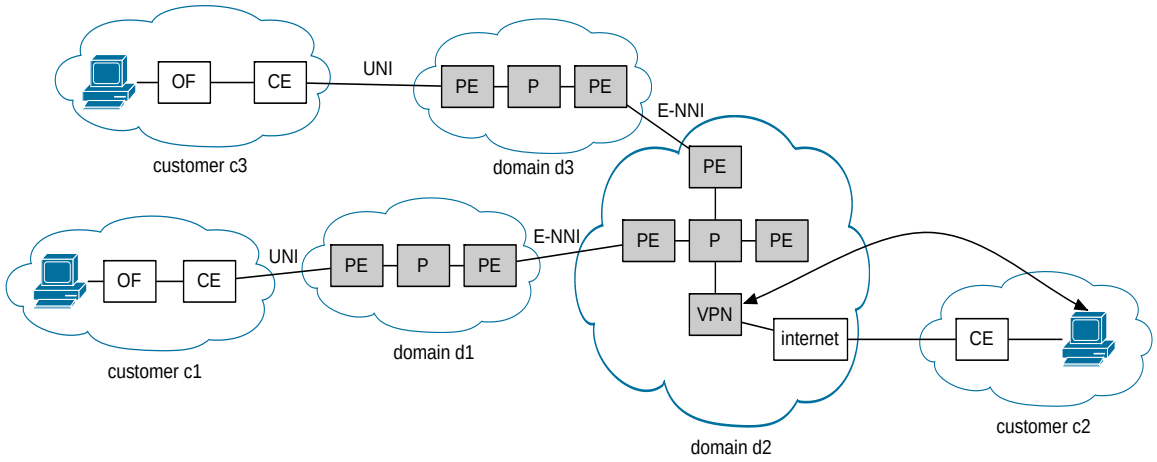
COCO LAYERED ARCHITECTURE – ULTIMATE GOAL

- Web portal as user fort-end
- REST API for web portal to controller communication (northbound interface)
- BGP for communication between controllers in different domains
- OpenFlow for controller to switches communication (southbound interface)



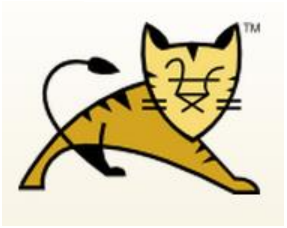
SOME ARCHITECTURE DETAILS: LAYER3 VPN, MPLS FORWARDING

- We have decided to make the following choices regarding architecture details
  - Layer3 (not Layer2) service
  - Double MPLS tagging:
    - External: aggregation and forwarding in network core
    - Internal: to differentiate between CoCo instances



COCO IS OPEN SOURCE AND BASED ON OPEN SOURCE (DE FACTO) STANDARDS

- When designing CoCo, we decided to use as much existing (or emerging) open source technology as possible
- Specifically, we have used
  - OpenDaylight controller (started with Hydrogen, now running Helium)
  - RESTconf and OpenFlow as north- and southbound interfaces
  - Tomcat and MySQL to host portal application and store information about CoCo instances
  - Eclipse J2EE with Maven plugin for portal software development
  - OpenStack to control Virtual Machines
  - Mininet for some test and prototyping (uses OpenVSwitch)
- Pica8 switches used in physical testbed



COCO PROTOTYPE DEVELOPED

- We developed single-domain CoCo prototype
  - Both physical (Pica8) and virtual (Mininet) test bed was used
- Validation tests performed: CoCo instance set-up with connectivity test takes less than 10s
- Code available on Github
  - https://github.com/rvdpdotorg/CoCo
- Inintial version of prototype demonstrated during SuperComputing'14 exhibition
- TERENA Networking Conference presentation in May'14
- INDIS workshop paper in Nov'14
- Journal paper submitted

CHALLENGES DUE TO RAPID TECHNOLOGY DEVELOPMENT

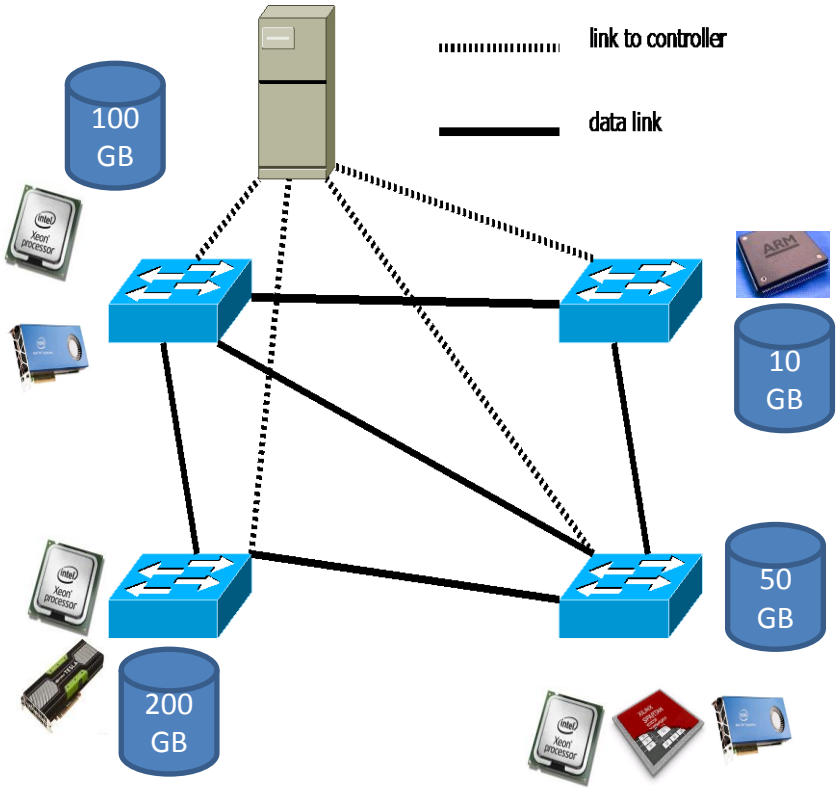
- ~~With SDN you can easily do whatever you want with your network~~
- In course of CoCo project, several challenges were brought to the surface
  - SDN related technologies evolve very rapidly
    - New, not necessarily backward-compatible software versions released frequently
    - Few standards are mature (n.b.: most standards are *de facto* standards)
    - Implementation quality of standards varies
    - Documentation frequently lagging behind development
    - Certain bug fixes/feature implementations still require vendor action

PLANS FOR 2015: MULTIDOMAIN, FEDERATED AUTHENTICATION

- We plan to continue work on CoCo, focusing in `15 on the following aspects:
  - CoCo instance to span across several domains
    - CoCo agent to exchange information about participating sites with its peers in other domains
    - Possible intergration with OpenDaylight VPNService project (Ericsson, SURFnet, Dell)
  - Extending CoCo portal with security
    - Authentication – preferably federated model, authorization
  - Adding more network administrator functionality and working on simulation environment

VISION PAST 2015: BEYOND CONNECTIVITY-ONLY SERVICE

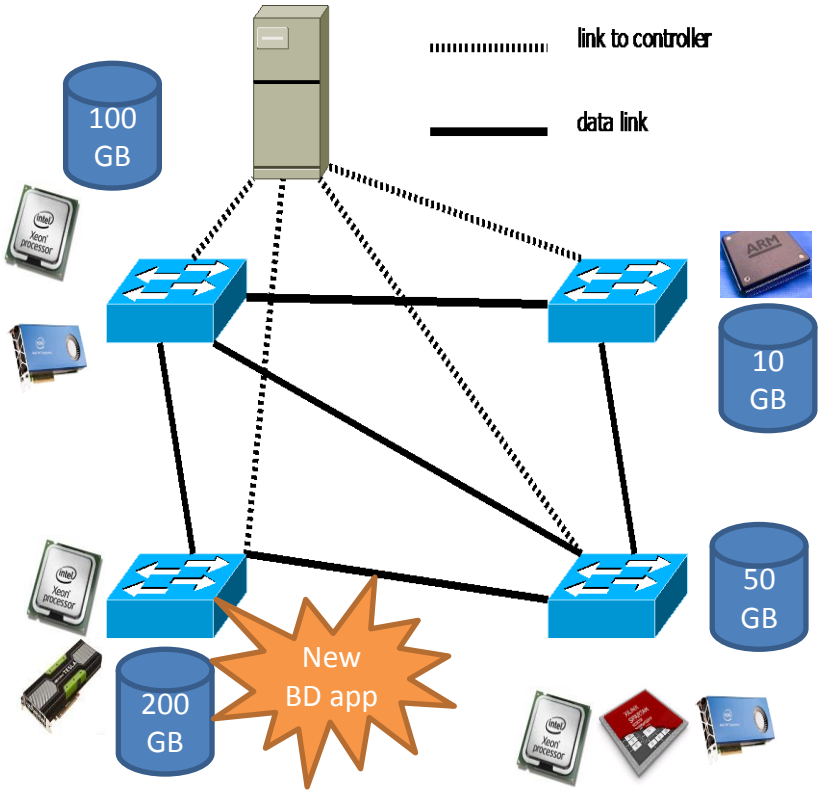
- System actively helping optimizing BigData analysis process
- Not only connectivity aspects considered but also storage and processing
  - Processing: possibility to use accelarators like Nvidia TESLA or Intel Xeon Phi
- System has large autonomy in deciding which resources are used and in what way



- Use case in „Sense Making of BigData" TNO programme

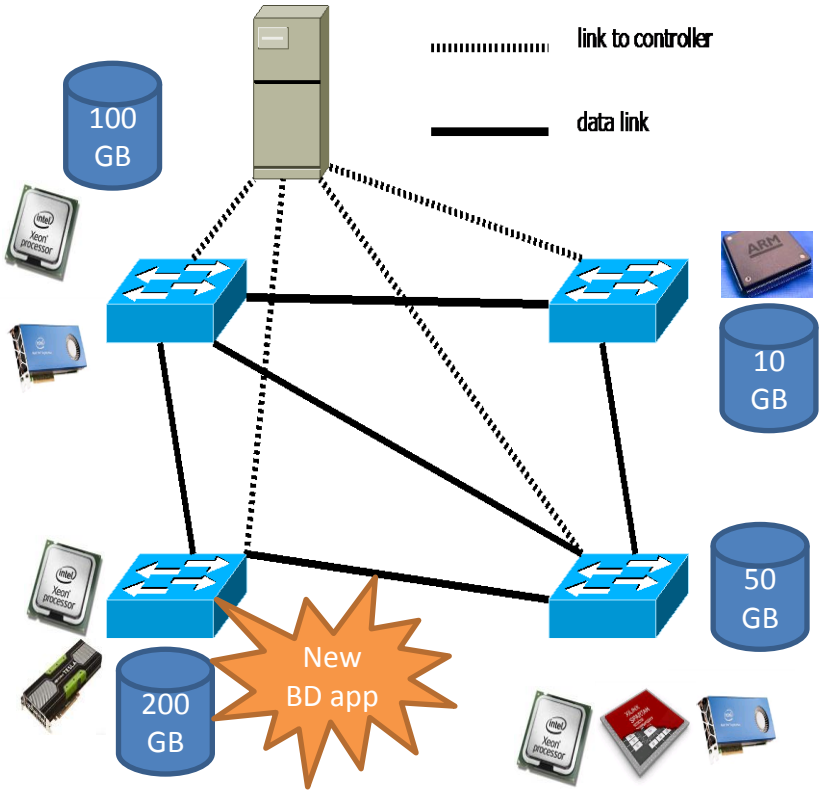
VISION PAST 2015: BEYOND CONNECTIVITY-ONLY SERVICE

- Assume Software Defined Network connects several processing+storage sites (nodes or clusters)
- New BigData application (BDapp) to be served
- Application has constrains such as:
  - Data is distributed in certain way
  - Can use accelerators
- Question: how to optimally handle this request considering both BDapp and infrastructure constraints



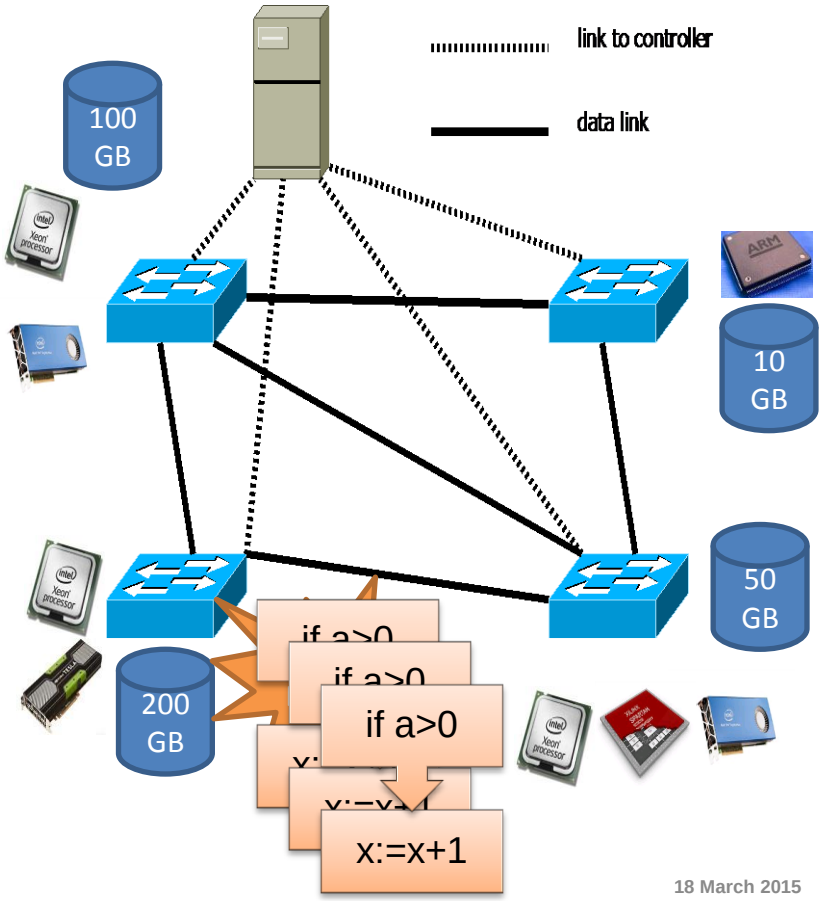
STRATEGY I: PULL DATA TO SINGLE PROCESSING NODE

- Based on overall network state and processing units state decide:
  - Pull data to single processing node;
  - Large transfers are expected, O(100GB)
  - SDN controller creates virtual network, reserves bandwidth etc.
  - Paths may not be shortest in terms of hops
- Strategy requires combining information from SDN controller and processing nodes



STRATEGY II: PUSH COMPUTATIONS ENGINES TO WHERE DATA IS

- Based on overall network state and processing units state decide:
  - Push computing engine to data location and fetch only results
  - Use Virtual Machines (VMs) images which hold analytics software
  - Maybe even use some lightweight containers like Docker images;
  - Small transfers expected; VMs: O(1GB), Docker: O(10MB)
- Again: combine information from SDN controller and processing nodes



SUMMARY

- SDN offers new possibilities for creating services which either do not yet exist or require significant effort to be brought up
- Such services may be initiated on-demand by applications/users
- Network admins will still have a job ;-)
  - More control offloaded to users or system itself requires (even more?) careful planning and implementation
- SDN + BigData = (happy) marriage
  - Intelligent network supports running BigData applications
- Control system running to large extend autonomously, making decisions dynamically, based on infrastructure state and applications requirements
  - E.g., switch on-flight between data-pull and computing-engine-push strategies