An SDN based approach for the ATLAS data acquisition network

Authors: Espen Blikra[(1)], Mikel Eukeni Pozo Astigarraga[(1)] on behalf of the ATLAS Collaboration
[(1)] CERN, Geneva, Switzerland

ATLAS is a high energy physics experiment at the Large Hadron Collider located at CERN. During the so called Long Shutdown 2 period, scheduled for 2019, ATLAS will undergo several modifications and upgrades in order to cope with higher luminosity levels. In particular, a new read-out chain will be built for the New Small Wheel muon detector[1] and Liquid Argon calorimeter[2]. Instead of subdetector specific electronic devices, the new system will make use of commodity servers and commercial network technologies. The new architecture will provide a common interface to both subsystems and it will be extensible to all subdetectors in the High-Luminosity LHC upgrade (2024)[3]. Thanks to the high speed network, interconnecting order of hundred 40 Gbps port, it will be possible now to provide connectivity to a well differentiated set of applications:

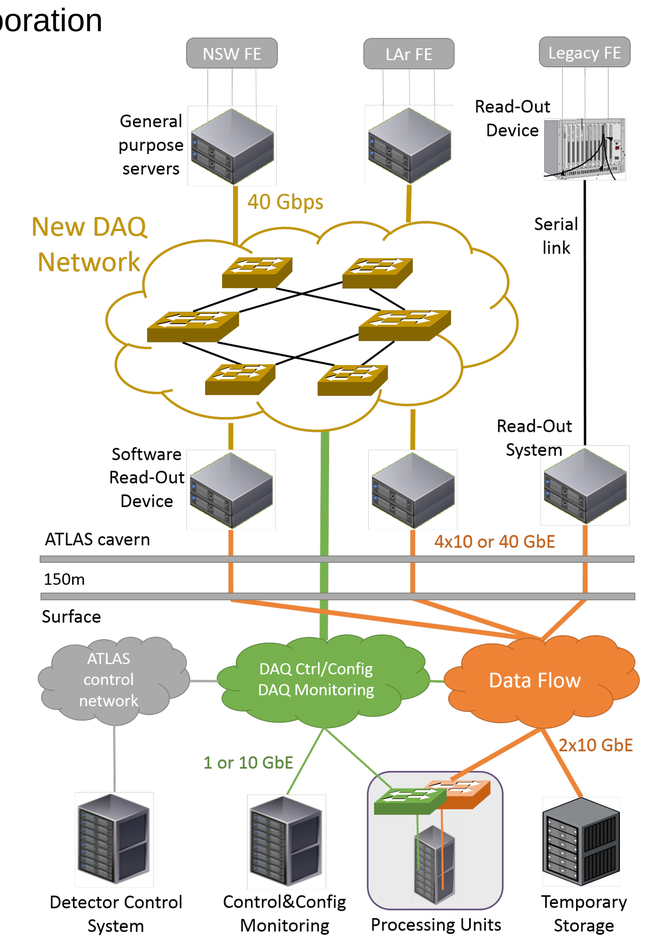
- **Detector Control System** and **Device management:** mission critical systems whose traffic cannot be disturbed by any other traffic sharing the underlying network. The bandwidth and latency requirements are not demanding and it is sufficient to ensure that packets will not be dropped in any circumstance.
- **DAQ Control and Configuration:** these applications provide control, configuration and monitoring of the processes running in the system. The traffic generated needs to be treated with high priority but consumes little bandwidth once the data taking process has started.
- **DataFlow applications:** The physics data read-out from the detectors has to be routed to the event filtering farm. This traffic consumes the majority of the bandwidth.
- **Monitoring** and **Calibration:** these applications are bandwidth hungry and their traffic should be treated with the lowest priority not to impact the physics data acquisition when running in parallel. A minimum guaranteed bandwidth should however be allocated.

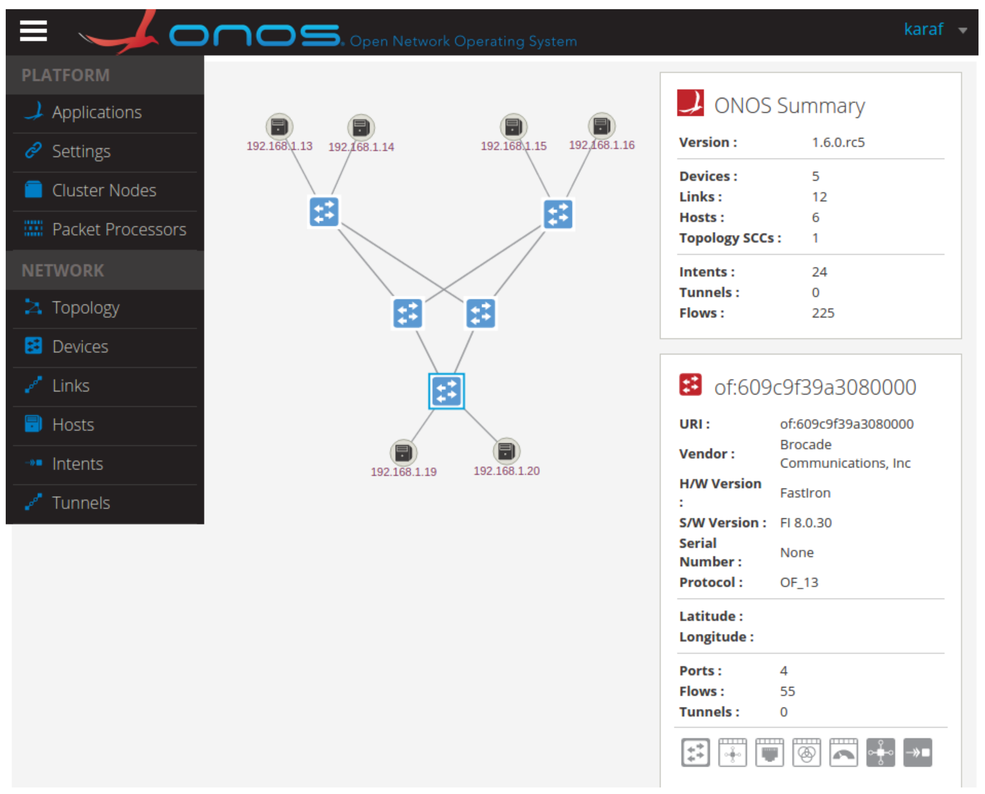
We can define therefore a minimum set of requirements for our network. First, in order to achieve the desired traffic differentiation, the network needs to mark the packets and apply different Quality of Service policies effectively. Second, some kind of network redundancy is needed to cope at least with the most frequent network failure scenarios in a transparent manner. Third, once the redundancy added, the network should make effective usage of all the available links providing a well-balanced link occupancy in normal working conditions.

To meet these requirements we have explored an SDN based solution and created a test model to study it. SDN stands for **Software Defined Networking** and it is an innovative approach to network management. The control plane of the network devices is separated from the data plane and moved to a centralized controller application providing a high level of flexibility. Instead of the classic network protocols used to build the network topology and the traffic forwarding rules, the SDN controller programmatically creates the rules and loads them into the network devices using the OpenFlow protocol[4].

We have performed an evaluation of the different open source controllers available and we have retained **ONOS**[5]. ONOS (Open Network Operation System) is an SDN platform providing a set of services and applications allowing a simple and robust deployment of a SDN network. Examples of the services provided by ONOS are the Topology Service, in charge of providing the shortest path between devices, and the Statistics Service which gathers traffic statistics from devices. Applications running in ONOS subscribe to services and can also receive notifications when network conditions change.

We have created our own ONOS application to deal with a leaf-spine network addressing the requirements of the new ATLAS DAQ network. Below we can find the snapshot of the ONOS web based GUI showing the unfolded topology of the test setup.





ONOS: flow rules and intents

The ONOS controller communicates with the switches using the OpenFlow protocol, where the forwarding rules are called **flow rules**. Flow rules consist primarily on three parts: a flow matching criteria, a packet modification instruction set and a forwarding rule. These flow rules installed in the network devices establish a communication path between hosts.

To make it easier to program the desired network behavior, ONOS provides a higher level abstraction called **intents**. Intents work by specifying the source and destination hosts and, similarly to the flow rules, a traffic selector criteria and the associated treatment. The intent framework will then find a path from the source to destination host and install the necessary flow rules on the intermediate devices. Should a network element fail, the intent service will try to find an alternative path and generate new flow rules.

The intent used in our solution is based on a so called multi-point to single-point intent, which establishes one way communication from all source hosts to a single destination host. Therefore, for bidirectional all-to-all communication, we need the same amount of intents as hosts in the network: O(N).

Problems with the Address Resolution Protocol (ARP)

A conventional Ethernet network uses either the destination IP or the destination MAC address to take a forwarding decision. An SDN network can take the same forwarding decision based on any criteria. We have chosen to do it based purely on IP addresses.

In order to minimize the impact on the hosts we have kept them unaware of the forwarding technique used. Hence they will try to use the ARP protocol to resolve the MAC address of the next IP hop as usual. The network devices will forward all ARP requests to the controller who will, in turn, reply always with a dummy destination MAC address. The subsequent traffic will be then forwarded from the source to the destination. The problem arises when the destination host drops the incoming packets not containing an allowed destination MAC address.

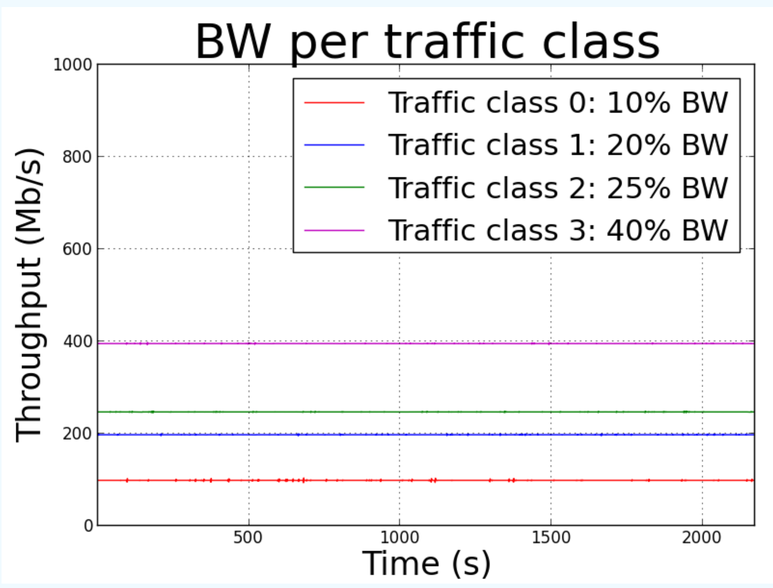
To ensure correct MAC on all packets, the intents created also include a packet modification rule to change the destination MAC in the packets. This is possible because the controller is provided with the host interface configuration information.

Quality of Service (QoS)

SDN provides big flexibility on the criteria to differentiate and mark the flows. The switch forwarding engine can then honor the QoS policy assigned to each type of traffic. Additionally it is possible to change the QoS policies dynamically if needed (e.g. maximum available bandwidth to monitoring traffic can be increased in case the system needs to be debugged).

In our model we have chosen to differentiate the traffic based on the TCP port used by the applications and mark the packets as soon as they enter the network by setting the PCP field on the VLAN header to a given value.

As shown in the graph, the build-in scheduler of the switches assigns guaranteed percentages of the link capacity to each application traffic. The assigned bandwidth is in this case: Traffic class 0=40%, traffic class 1=25%, traffic class 2=20%, and traffic class 3=10%, which closely resembles the data gathered in the graph as the test is measured on a 1 Gbps link.

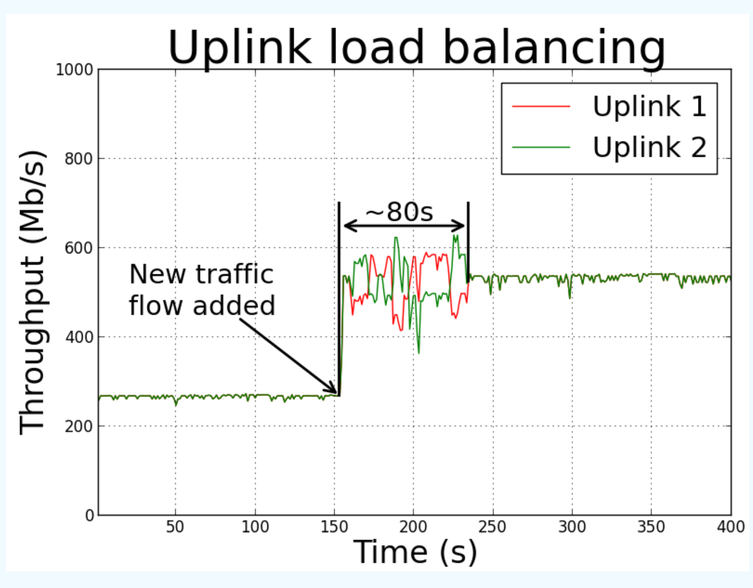


Load balancing

To achieve a well-balanced uplink occupancy and effective use of the available links, our application implements an automatic load balancing solution. The forwarding paths are decided by intents. Since we need the intents to make path choices based on traffic statistics, we created a new intent type which chooses the least loaded path.

The load balancing is achieved by continuously tracking the link load differences on the uplinks of the leaf switches and correcting them. When an imbalance is discovered, one of the intents using the heaviest loaded link will be triggered to find a new path. If the imbalance persists, more intents will be triggered until the load is balanced. The intent with the heaviest use of the link will not be chosen to avoid only moving the problem.

In the image we observe the uplink load balancing when a new flow is added to the system. The system oscillates until it finds the right working point.

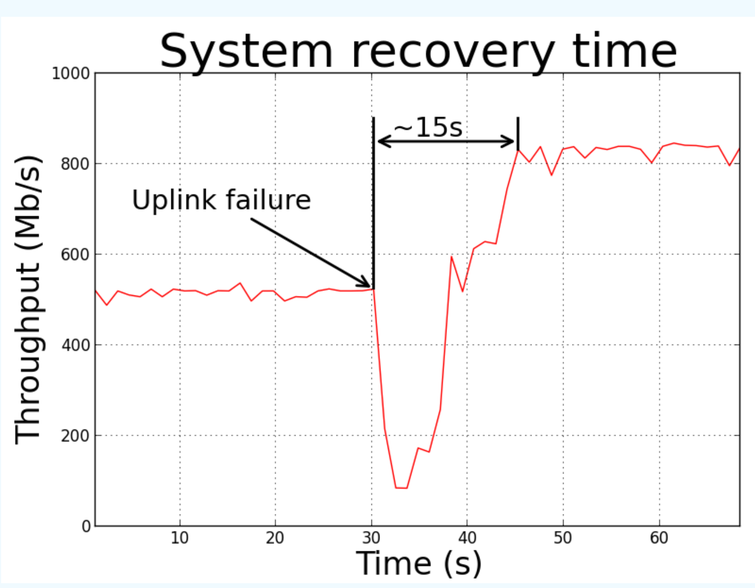


Automatic failover

The fault tolerance in this solution is provided by the intent framework of ONOS. The intent framework continuously listens for failing network elements, and tries to resolve them as soon as they are detected. Typical failures handled by the intent framework are links and devices failing.

The amount of time the system takes to recover will depend on the type of error and the amount of intents which have to find new paths. Given a link error all of the intents seem to be recompiled, even if the link error should not affect them. This results in downtime for the traffic not using the link, and longer downtime for the traffic using the link because all intents and flow rules have to be reinstalled.

In the graph we observe the time the system needs to recover from an uplink failure. The plot represents the throughput on the link that absorbs the traffic from the faulty uplink.



Conclusions and future work

In our experience SDN is a promising technology providing high flexibility on the network control plane. It allows to create new network functionality and topologies With the ONOS controller we were able to deploy our SDN application meeting some of the ATLAS data acquisition network requirements. However we encountered some missing or underperforming implementation which degraded the solution, particularly with the intent framework.

The current solution raises scalability issues since the amount of flows installed for each intent is exponential. Currently for each leaf switch the number of flows will be number of ports times number of hosts in the topology. In future versions we plan to change this to one flow rule for each host in the topology for each switch. With better usage and modification of intents, we should be able to improve recovery and balancing time.

We would like to expand the QoS solution by using Metering. Meters can be used to rate limit the bandwidth used by chosen flows. e.g. we can use it to prevent instantaneous saturation of the network due to large spikes in the monitoring traffic.

OpenFlow 1.0 only uses one table for all flow rules, while OpenFlow 1.3. allows us to install flow rules in multiple tables. We intend to use this to separate the QoS rules from the forwarding rules. The packets will then first be tagged in the QoS table, which sends the packet to the forwarding table where the forwarding decision is taken. This reduces the amount of intents and flow rules needed, providing improved scalability and reduced complexity.

OpenFlow hybrid mode is a feature where a SDN and conventional network can be combined. We want to explore this feature's ability to help with gradual deployment of the solution and provide basic forwarding in case of connection failure to the controllers.

References

[1] Konstantinos Ntekas - The ATLAS New Small Wheel Upgrade Project - PoS TIPP2014 (2014) 331

[2] Hong Ma - Upgraded Trigger Readout Electronics for the ATLAS LAr Calorimeters for Future LHC Running – 2015 J. Phys.: Conf. Ser. 587 012019

[3] FELIX: a High-Throughput Network Approach for Interfacing to Front End Electronics for ATLAS Upgrades- J. Phys.: Conf. Ser. 664 (2015) 082050

[4] https://www.opennetworking.org/sdn-resources/openflow/ (August 2016)

[5] http://onosproject.org/ (August 2016)

CERN ATLAS ONOS Open Network Operating System