# Hybrid SDN: Evaluation of the Impact of an Unreliable Control Channel

Mohammed Osman*, José Núñez-Martínez† and Josep Mangues-Bafalluy‡
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss, 7 - 08860 Castelldefels (Barcelona), Spain
Email: [*mohammed.osman, †jose.nunez, ‡josep.mangues]@cttc.cat

***Abstract*—Software defined networking (SDN) setups in general assume a perfect control channel between network devices and the centralized controller. Whilst this may be the case in certain controlled contexts such as data centers, when moving to wide area networks, this assumption does not necessarily hold true anymore. Therefore, there is the need to evaluate the impact of unreliable control plane behavior in SDN scenarios. This paper evaluates the impact of a lossy control channel on data plane throughput and latency. Additionally, it presents a hybrid SDN approach that dynamically switches between centralized and distributed operation at nodes, depending on the control channel packet loss ratio. Results show that throughput and latency at data plane can be maintained at acceptable level even in the presence of substantial control channel losses.**

## I. Introduction

Software Defined Networking (SDN) [1] separates the data and control planes of the network. It logically centralizes the control in an SDN controller which is in charge of telling each network device how to forward packets by installing the appropriate forwarding rules. One of the main advantages it brings is programmability, since there is a single entity (the logical controller) with which network management applications must interact to apply their policies. Through agreed upon APIs, the full potential of SDN can be exploited by network managers. This has been the case in controlled scenarios that do not pose challenging constraints to the deployment of such paradigm because of the availability of reliable and high capacity networks, for instance, in data centers. However, moving to wide area networks, the assumption of the availability of such reliable networks may not hold true anymore.

If SDN control is physically centralized, in the presence of unreliable conditions, three types of faults can be identified, namely i) data plane faults (the network element(s) or port(s) associated to the network element(s) fails), ii) control channel faults (the connection between the SDN controller and the data plane element(s) fails or it experiences losses), iii) SDN controller faults (the SDN controller fails). Therefore, a reliable control plane is key for the correct operation of SDN scenarios.

On a different front, hybrid SDN approaches have been introduced to enable the transition from centralized operation to legacy distributed operation. In this case, centralized and distributed operation coexist in the same network. There are various ways in which this can be done, depending on the scenario [2]. Ideally, the hybrid-SDN model should allow retaining the advantages of canonical-SDN (i.e., centralized logic and programmability) and legacy distributed networks (i.e., no single point of failure).

In this paper, we first characterize the impact of an unreliable control channel on data plane throughput, which substantially increases with the percentage of losses. After that, we exploit hybrid SDN to handle the unreliability of the control plane. This is done at the node level, and so, when impairments in the control plane are detected (e.g., packet losses), the node switches to distributed operation (i.e., it uses conventional distributed routing protocols). Results show that dynamically switching between modes of operation allows maintaining acceptable throughput level as well as latency at data plane even in the presence of highly unreliable control planes.

Previous work mainly focused on data plane reliability by designing efficient schemes for fast detection and recovery of the communication [3]–[5]. Much less effort has been devoted by the community to solve control plane unreliability (including controller failures and control channel losses). In order to overcome SDN controller failure, deployment of multiple SDN controllers [6] may be a solution (controller redundancy), but it may also lead to inconsistency issues. For instance, HyperFlow [7] proposes physically distributed controllers that are synchronized through a publish/subscribe system. Another option is to design schemes to handle control plane unreliability. For instance, ResilientFlow [8] restores the control channel through alternate paths (path redundancy) in the presence of channel failure. However, these research works focused on solving a single specific type of fault (SDN controller or control channel). They do not handle multiple faults at a time or degraded control channel performance (e.g., lossy control channel), which may be the norm when deploying multi-hop wireless networks to serve dense small cell deployments, for instance. To tackle SDN controller failures a solution may be by forming a hierarchy or cascade deployment model. In this context, since OpenFlow version 1.3 [6] supports multiple SDN controllers managing an equivalent set of forwarding nodes. But some inconsistent issues like *Event Ordering*, *Unreliable Event Delivery* and *Repletion of Commands* are still matters of concern. Ravana [9] ensures event ordering, correct event processing and execution of commands for exactly once during the SDN controller failure. In [10], integrated SDN principle has been proposed where

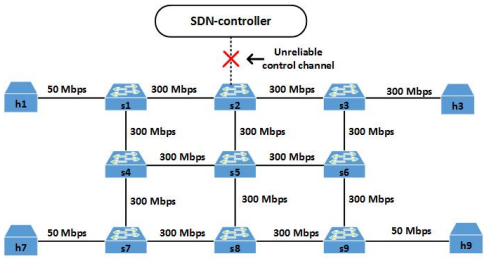



Fig. 1. Reference Network Diagram for both canonical-SDN and hybrid-SDN

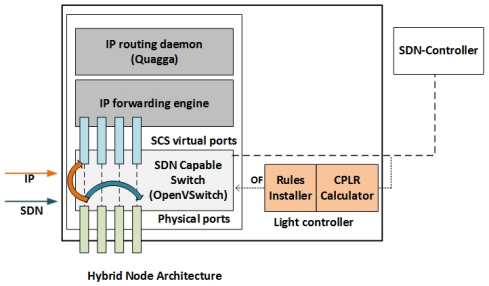



Fig. 2. Hybrid Node Architecture with integrated light controller

a module OLSR-to-OpenFlow (O2O) has been presented to configure control rules that are used to forward OpenFlow packets. After detection of the SDN controller failure by the O2O module, the O2O module dumps the OLSR routing tables into the forwarding nodes to recover SDN from the SDN controller failure. This approach needs translation of routing table into SDN rules.

Our scheme differs from state of the art solutions in the way control packet loss is handled. As control packet loss is of major concern, from the point of view of a network node, the failure of the SDN controller is equivalent to the failure of the control channel. Thus, we propose to take a look at control plane reliability from the point of view of the node experiencing the problem. We introduce logic at the node to detect the problems and make a decision on whether to operate in a centralized or distributed way. Therefore, our scheme provides a solution not only for control channel failure, but also for SDN controller failure. Evaluation confirms that this operation reduces the bad impacts of these kinds of control plane failures over data plane.

The rest of the sections of the paper are organized as follows. Section II illustrates our hybrid-SDN approach, including the hybrid node architecture with a light controller. The proposed approach is evaluated in section III, and finally, conclusions in the section IV.

## II. Hybrid Node Architecture with Light Distributed Controller

The architecture illustrated in Figure 2 aims to quickly adapt to changes in the control plane channel by combining both centralized and distributed control logic, thus, creating a hybrid control plane architecture. In this way, our architecture attempts to preserve the benefits of both worlds (i.e., centralized and distributed ones). Specifically, we propose to maintain a centralized control logic to preserve the benefits of canonical-SDN (i.e., simple network management, programmability) under reliable control plane conditions, whereas the distributed control plane is in charge of acting under unreliable conditions to quickly react to failures that avoid the inefficient use of a centralized control logic. Our hybrid control plane architecture proposes changes to the architecture of data plane nodes while preserving the architecture of centralized controllers with respect to canonical-SDN models.

The data plane node is divided amongst the data plane forwarding pipe and the control logic to decide the operation of the data plane forwarding pipe. In what follows we describe the architecture of the main components embedded in a data plane node. Last but not least, we describe the work-flow between the aforementioned building blocks and the centralized SDN controller.

*1) Data Plane Forwarding Pipe:* We have adopted the Open Source Hybrid IP/SDN networking (OSHI) framework that has been designed in [11]–[13]. This framework allows nodes to concurrently run a distributed control plane and a centralized control plane. To attain this goal, the hybrid-node embeds an SDN Capable Switch (SCS) such as Open vSwitch, an IP-based forwarding engine (i.e., the one provided by the Linux kernel), and an IP routing daemon based on Quagga to calculate distributed routes (see Figure 2). The SCS is connected to the physical network via the physical interfaces while the IP forwarding engine is connected to the SCS via a set of internal virtual ports endowed in the SCS. The main flow table embedded in the SCS allows for distinguishing between regular IP packets that are needed to be processed by distributed control plane (i.e., Quagga routing daemon) and those needed to be processed by the SDN controller. VLAN (Virtual LAN) IDs have been used to distinguish between packets that need to be processed by a distributed control plane and packets that have to be processed by SDN controller. In particular, packets tagged with VLAN IDs are processed by SDN, while packets without a VLAN ID are processed by IP routing daemon.

*2) Light Controller:* We included a light controller in the data plane node that acts as a gateway to the SDN controller, in order to enable centralized or distributed control depending on the reliability of the control plane channel between data plane devices and the centralized SDN controller. The light controller is based on i) a monitoring framework that continuously infers the reliability of the control plane network by periodically monitoring the status of the control plane channel. In this paper, the metric is based on determining packet loss ratio of the control plane channel. The resulting metric referred to as Control Packet Loss Ratio (CPLR) that determines the status of the control plane channel between the data plane node and the SDN controller. The activation of the distributed control operation from the centralized control operation for a node is decided by the decision module installed in the data plane node. The decision module is based on the stats gathered

from centralized or the distributed control plane logic which is active in a given data plane node. The decision module referred to as CCRI (**C**PLR **C**alcultor and **R**ules **I**nstaller), periodically inspects CPLR of the control channel to characterize a lossy control channel and determine control channel failure (see Figure 2). Under high losses or failure, it triggers the actions to switch from centralized to distributed control plane operation.

*3) Interaction between the Data Plane Node Forwarding Pipe, the Light Controller, and the centralized SDN controller:* By default, packets are tagged with a VLAN ID to enable the use of a centralized control plane logic embedded in the SDN controller. In the following we summarize the centralized operation.

1) When a packet with VLAN ID reaches to a physical interface of the SCS, the SCS conducts a lookup in its flow tables to find a match for the current packet.
2) If a match is found in one of the flow tables, the packet is then forwarded according to the rule installed previously by the SDN controller into that flow table. Otherwise, the incoming packet is forwarded to the centralized SDN controller for appropriate handling according to the policies defined.
3) The centralized SDN controller installs the necessary OpenFlow rules to serve the current incoming packet. Thus, the forwarding data plane node simply follows the instructions set by the SDN controller to forward a packet.

During any impairment experienced by the control plane, either a failure in the SDN controller or the control plane channel failure due to the presence of a lossy control channel, CPLR measurements will infer an anomaly in the proper operation of the network. In what follows we describe the work-flow related to the distributed operation. The CCRI module calculates the CPLR of control channel periodically and once the CPLR exceeds a certain threshold level the network operation is switched from centralized to distributed mode by following these steps:

1) When the CCRI module of the light controller detects CPLR above the threshold level, it deletes the old rules from the SCS flow tables of data plane nodes that were installed with the intervention of the SDN controller.
2) The CCRI module of the light controller installs new rules in the SCS flow table of data plane nodes. One rule includes OpenFlow *POP_VLAN* action aiming to remove the VLAN tag of the incoming packets to the ingress port of a node and to forward the incoming packets to the IP forwarding engine.
3) Incoming packets arriving to the SCS ingress port are then forwarded to the IP forwarding engine in order to process the packets through the IP routing daemon. In the IP routing daemon, a distributed routing protocol, in our case, Open Shortest Path First (OSPF), routes packets to their intended destination.
4) On the other hand, the CCRI module of the light controller also updates the SCS flow table of data plane nodes by adding another rule that pushes the VLAN tag again to the packets using an OpenFlow *PUSH_VLAN* action, before the packets leave the SCS egress port and sends the packets towards the following hop towards the destination host. All these rules are deleted and installed without intervention of the SDN controller.

## III. Performance Evaluation

As data plane, we used Mininet [14] version 2.2.1 and Open vSwitch 2.3.0 supporting openFlow 1.3 to model a *N*N* grid topology. As SDN controller, we considered Ryu [15] to set forwarding policies into data plane nodes. As traffic flow generators, we used *iPerf* [16] to inject TCP traffic flows.

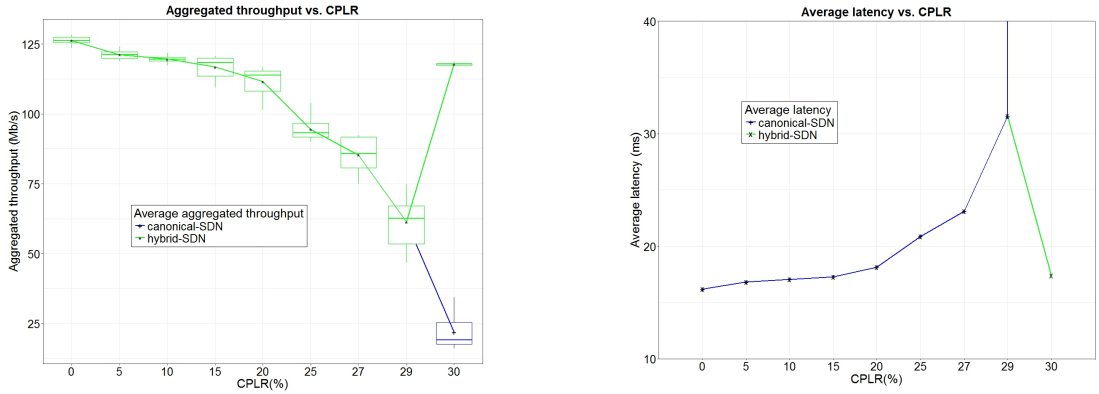
### A. Canonical-SDN Operation

First, we carried out an experiment in order to observe the impact of control plane failure over data plane throughput in canonical-SDN operation. Total duration of each experiment was 120 second and each experiment was repeated 10 times. We set up a canonical-SDN with 9-nodes grid mesh network, where nodes lack distributed control plane logic. As depicted by Figure 1, four hosts were connected to the four corner nodes of the topology. Three TCP flows have been generated from three hosts (*h1*, *h7*, *h9*) which were destined to upper right corner host (*h3*).

When a new packet comes to a switch, the switch looks up in its flow tables for a flow entry match for the packet. If no match is found in the flow tables, the switch sends the packet as a *PacketIn* message to the SDN controller using the *flow-miss* entry rule. Then, the SDN controller calculates the shortest path for the packet and, using a *PacketOut* message, the SDN controller instructs the switch to install flow rules into the flow table of the switch. The switch keeps the rules for infinite period of time unless the switch gets further instruction from the SDN controller to modify or delete the rules. With subsequent flow of packets, the switch simply forwards the packet according to the rules installed previously.

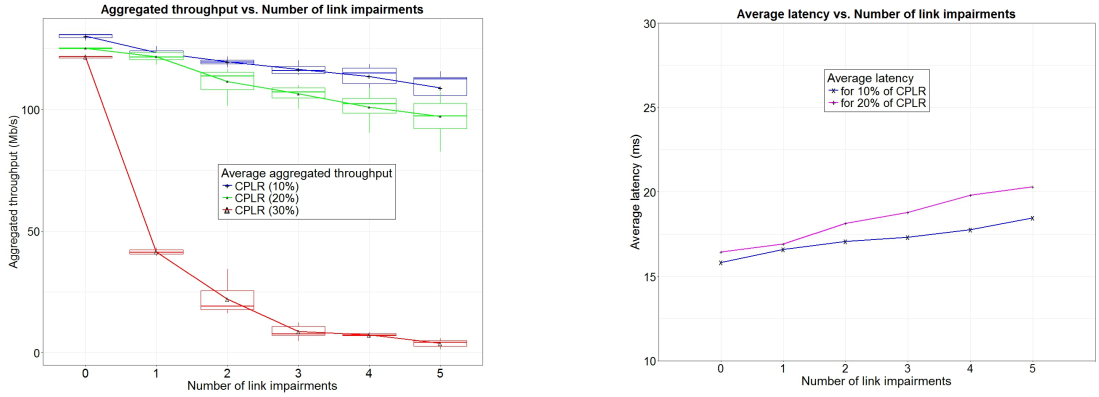
As soon as any link impairment happens on the data plane, the affected switches instruct the SDN controller immediately about the impairment by sending *PortStatus* messages. In our case, when the SDN controller receives *PortStatus* messages, the SDN controller instructs, by means of *FlowMod* messages, the affected nodes as well as all the nodes in the affected path to delete all the rules from their flow tables except the *flow-miss* entry rule. After instructing to delete the old rules, the SDN controller again calculates a new shortest path and instructs all the switches in the new path to install new rules in their flow table. The switches then forward packets according to the new rules.

To observe the impact of the unreliable control plane over data plane throughput and latency, in a controlled way we generate losses of control messages that are exchanged between the switches and the SDN controller. We used *netem* [17] to add impairments to the control channel (the link between the SDN controller and the network node). We also broke




(a) (b)

Fig. 3. Comparison between aggregated throughput and average latency behavior of canonical-SDN and hybrid-SDN




(a) (b)

Fig. 4. Aggregated throughput and average latency behavior against multiple link impairments for certain CPLR in canonical-SDN

data plane links randomly by maintaining a sequence of link down/up at some periods of our emulation time.

Figure 3a and 3b depict the aggregated throughput and average latency over different Control Packet Loss Ratios (CPLRs) respectively. Results reveal that the increment of CPLR causes degradation of the data plane throughput and substantial growth of latency in canonical-SDN. In the evaluated scenario, during the presence of link impairment in the data plane, the affected nodes send *PortStatus* messages to the SDN controller to let the SDN controller know about link failure. In this case, due to the faulty control link between the SDN controller and the switches, the control messages (i.e., *PortStatus*, *PacketIn*, *PacketOut*, and *FlowMod*) get dropped and can not be exchanged on time. Moreover, *LLDP (Link Layer Discovery Protocol)* messages that are used by the SDN controller for discovering network topology, also get dropped. As the SDN controller and a switch use a single path TCP connection to maintain communication between them, dropped packets are retransmitted again. Though dropped control packets are retransmitted again, loss of the control packet affects the restoration time of a new path at data plane due to untimely delivery of the control packets between the SDN controller and the forwarding nodes. Therefore, with increasing CPLR, the SDN controller requires more and more time to establish a new alternate path at data plane because of the control packet loss, which, in turn, affects the aggregated throughput and latency at data plane substantially.

Since the focus of this paper is on quantifying the advantage of hybrid operation, we did some preliminary evaluations to understand what were the CPLRs that allowed the control channel to operate (even with difficulties) and those for which it was impossible in most repetitions. In particular, we observed that value to be 30%. In this case, there is an immense degradation of aggregated throughput and huge increment of latency at the data plane for 30% of CPLR, as the SDN controller and the forwarding nodes can not maintain proper communication between them because of control packet loss. Thus, the threshold for switching from centralized to distributed mode of operation was set to this value.

Figure 4a depicts the aggregated throughput and Figure 4b illustrates average latency at data plane in case of canonical-SDN while the network was experiencing an increasing number of data plane link impairments for a certain CPLR. We

observe that aggregated throughput at data plane substantially degrades and on the other hand latency increases with increasing number of broken links at data plane. This was evaluated for various values of CPLR to assess its dependency on varying degrees of data plane link impairments. As CPLR increases, so does restoration time due to control packet loss and consequent retransmission of control packets between the network nodes and the centralized SDN controller. During the CPLR of 30%, degradation of the aggregated throughput at data plane is enormous and latency is substantially higher due to substantial loss of control packets, which causes several retransmissions that affects proper communication between the SDN controller and the forwarding nodes. As latency is very high during CPLR of 30%, it is not represented in the figure.

### B. Hybrid-SDN Operation

In the previous section, the operational degradation of a canonical SDN network due to the unreliability of the control channel was characterized. In this section, we quantify the gains that a hybrid SDN operation offers. In this direction, we repeat the same set of experiments with hybrid-SDN operation under control plane and data plane impairments. We emulate the same network topology that is used for the canonical-SDN scenario. But the difference is that the forwarding nodes are hybrid nodes, which combine centralized and distributed network operations and can take autonomous switching decisions between both modes. Initially, the network is operated in centralized mode. We configure the light controller (see Figure 2) to monitor switch-to-controller communication and to measure its CPLR in every 5 sec. Then, we start increasing the CPLR and when control channel losses reach 30%, the CCRI module of the light controller performs network switching from centralized to distributed operation. It can be observed from Figure 3 that during the control plane impairments, canonical-SDN fails to maintain proper network operation, as explained in the previous section, whereas hybrid-SDN is able to obtain aggregated throughput and latency values equivalent to those obtained for the lowest CPLR values in the canonical SDN case.

## IV. Conclusions

Due to centralized control over the networks, any impairment on the control plane is hazardous for proper operation of the canonical-SDN (i.e., centralized controller) because communication between the SDN controller and a forwarding node is hampered during the control plane impairments. Hybrid-SDN not only leverages the advantages of canonical-SDN, but also mitigates the drawbacks of canonical-SDN in terms of reliability. Hybrid-SDN indeed keeps the network operational irrespective of the level of control plane impairments by switching network control from centralized to distributed. In fact, our hybrid-SDN approach switches network control logic when CPLR of the control channel reaches an unacceptable level that limits proper communication between the SDN controller and the forwarding nodes. Additionally, it allows obtaining aggregated throughout and latency values equivalent to those obtained by canonical-SDN under low CPLR. We believe that our CPLR based network control logic has a high potential in (more unreliable) wireless transport deployments (e.g., multi-hop networks in dense small cell deployments). Evaluation of this scenario will be our next step.

## Acknowledgment

This work was partially funded by the grant TEC2014-60491-R (5GNORM) and the 5G-Crosshaul project (H2020-671598).

## References

[1] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, 2012.

[2] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014.

[3] H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster, "Coronet: Fault tolerance for software defined networks," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*. IEEE, 2012, pp. 1–2.

[4] M. Kuźniar, P. Perešíni, N. Vasić, M. Canini, and D. Kostić, "Automatic failure recovery for software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 159–160.

[5] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in openflow networks," in *Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the*. IEEE, 2011, pp. 164–171.

[6] *OpenFlow v1.3*. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[7] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[8] T. Watanabe, T. Omizo, T. Akiyama, and K. Iida, "Resilientflow: Deployments of distributed control channel maintenance modules to recover sdn from unexpected failures," in *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*. IEEE, 2015, pp. 211–218.

[9] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller fault-tolerance in software-defined networking," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 4.

[10] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmsdn)." in *WiMob*, 2013, pp. 89–95.

[11] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, and E. Salvadori, "Oshi-open source hybrid ip/sdn networking (and its emulation on mininet and on distributed sdn testbeds)," in *2014 Third European Workshop on Software Defined Networks*. IEEE, 2014, pp. 13–18.

[12] S. Salsano, P. L. Ventre, F. Lombardo, G. Siracusano, M. Gerola, E. Salvadori, M. Santuari, M. Campanella, and L. Prete, "Hybrid ip/sdn networking: open implementation and experiment management tools," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 138–153, 2016.

[13] *OSHI home page*. [Online]. Available: http://netgroup.uniroma2.it/twiki/bin/view/Oshi

[14] *Mininet*. [Online]. Available: http://mininet.org/

[15] *RYU*. [Online]. Available: http://osrg.github.io/ryu/

[16] *Traffic generator, iPerf*. [Online]. Available: https://iperf.fr/

[17] *netem*. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem