# NetServ on OpenFlow 1.0

E. Maccherani[1], J.W. Lee[2], M. Femminella[1], G. Reali[1], H. Schulzrinne[2]

[1] DIEI – University of Perugia, Perugia, Italy {maccherani,femminella,reali}@diei.unipg.it

[2] Dept of Computer Science, Columbia University, New York, USA {jae,hgs}@cs.columbia.edu

Currently, the Linux kernel is used to implement the NetServ transport layer. Packet filters, used to intercept packets in the NetServ node, and rules, used to route them to the proper service container, are installed in the node forwarding plane by using the netfilter library through the iptables tool. This type of solution is optimal in a prototyping point of view, but could be a bottleneck in real deployment environments. In fact the packets data path is implemented in software and, not only for a packet elaboration, but also for a normal routing task, it could takes times; in terms of performances, it cannot compete with common hardware routers, making difficult to widely adopt NS in real networks. The need of having a fast data path is now solved through the integration of an OpenFlow (OF) enabled switch inside the NS architecture. The switch acts as an hardware forwarding plane for the NS node, providing wire speed for packets that only needs to be routed or partially modified.

An OF-enabled NS node is composed by an OF switch and one or more NS machines, also called Processing Units (PUs), directly connected to it through an Ethernet connection. All the network packets that comes in and out from a PU must go through the OF switch.

An OF controller (OFC), that is needed for the switch to operate properly, has been implemented as a normal NS service that runs inside the OSGi environment and can be deployed in any NS node. Our OFC handles all the normal routines that is requested by to the OF protocol v.1.0 and also implements both layer two switching and layer three routing capabilities. In addition, a whole software module is dedicated to the NS integration, enabling the correct packets flow in and out the several PUs. The OFC can be deployed not only inside a PU attached to the switch, but also in any NS node that is network reachable by the switch. Moreover a single controller can manage multiple switches, becoming a centralized forwarding intelligence. The OFC is deployed as an usual NS service through the NSIS signaling protocol. Inside the OFC SETUP message, we must include some information that are used by the controller to know which and how many PUs are attached to a certain OF switch. Our architecture also supports OFCs that are deployed as a normal process, without being inside a NS runtime environment, but losing all the dynamic capabilities that this type of setup enables.

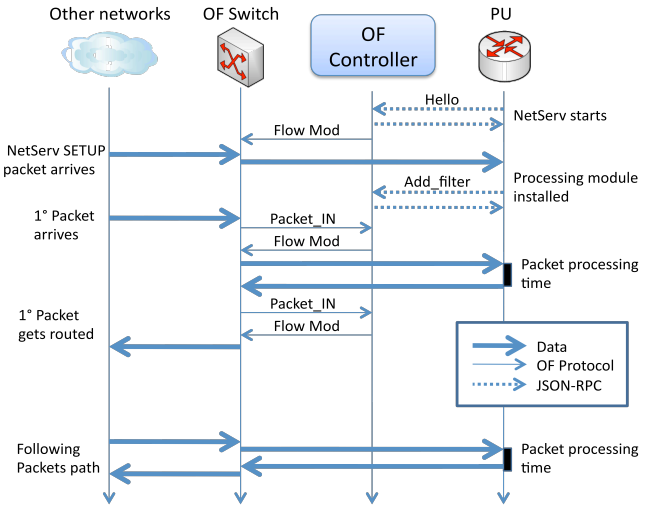



Figure 1 – Signaling Flow inside an OF-enabled NetServ node.

A PU attached to the OF switch is an ordinary NS node that acts exactly as expected in respect to the NSIS signaling and service handling. Some modifications to the NS controller software has been done so that it will be aware of being part of an OF-enabled NS node. A PU and the OFC can exchange information and state changes through the JSON-RPC protocol. This one is a very simple protocol that enables a remote procedure call encoded in the JSON format. When the PU is started, the NS controller sends a message to the OFC saying that it is alive. The OFC elects the first waking up PU of the node (in case a node has several PU attached) as master, and write a packet flow entry inside the switch's flow table to forward all NSIS signaling to the master PU.

Because of the flow match, if a NS SETUP signaling arrives to the master PU, the NS controller retrieves and installs the service. If the SETUP message contains packet filter rules, it also sets up netfilter tables. In addition, it sends a JSON-RPC method call to the OFC notifying it that a filtering rule exists for that switch, PU and service.

When ordinary network packets arrives to the OF switch, it behaves as usual, matching an entry in its flow table, or sending the packet to the OFC if no match was found. In this case, the OFC checks if the packet could match one of the NS service filters and, if there's no match, the packet is routed to destination as usual. If the OFC founds a match, it writes a packet flow entry inside the switch's flow table in order to forward all this packet flow to the correct PU. This "flow mod command" contains a match for every header field supported

by the OpenFlow Protocol, and two different actions. The first one the modification of the destination Ethernet MAC address; this is because it must exactly match with the MAC address of the PU's MAC NIC, otherwise the Linux Kernel will drop the packet. The second action is to output this packets flow to the switch port that is connected to the correct PU (the one that runs the requested service). Following packets of the same flow don't need to go through the OFC, because a matching rule already exists inside the switch's flow table, so they are forwarded to the PU at wire speed. Packets arrives to the PU and, as in a normal NS node, they are delivered by netfilter to the correct processing service, and then, after the elaboration, they come out modified to the same network interface (connected to the OF switch). The switch asks again to the OFC how to handle this new packets flow (new because the input NIC of the switch is different from the previous one, or maybe because the service has changed some data inside the packet header). The OFC knows that every packet that arrives to a NIC that connects a PU must not pass through the service matching rules check (because it has already been processed), but must go straight to the routing/switching module.

In comparison to the default NS architecture, the OF integration allows to separate the data path to the control path. NetServ, besides all its advanced features, is essentially a programmable router, so it has to accomplish all the routing stuffs as fast as possible, in order to be deployed in real environments. This path separation leaves the whole programmability and flexibility of a software component into the upper control layer, where new network algorithms and services can be dynamically deployed and tested. The forwarding layer is instead an hardware data path that can ensure fast wire speed performances, as good as other hardware routers. Obviously wire speed performances can be reached when packets doesn't need to pass through a PU, but only routed to the correct destination, otherwise is clear that it is necessary to take in account also the elaboration time of the packet that in this case is software based (we can also design an architecture where instead some modules are hardware processing modules, implemented by NetFPGA cards.

Having an OF-enabled data path can also help to remove some processing power resources constrain. As we have seen before, we can attach several PU to a single OF switch. In addition, the OFC listens for a wide range of JSON-RPC calls so that a remote service can control all OF switch features. In order to increase the packet processing speed, we can install the same elaboration service in multiple PUs attached to the same switch, and split the desired packet flow through all the PU, so that we can have packet processing in parallel. This also helps to reduce overload situations in which the bottleneck is represented by the processing power of a single PU. An external controller or service, for example always deployed inside a NS node, can utilize these remote calls to access all OF switch counters that are maintained per-flow, per-port and per-queue (i.e. received/transmitted packets and bytes, packets matches, packets drops). Monitoring the state of the node, a management agent or the NAME itself, can discover that a specific threshold is going to be reached and decides that the elaboration must be splitted across several PUs. He sends appropriate signaling messages to these PUs, that will install the service and then it informs the OF switch that a certain type of packet flow must be equally splitted to NICs where the PUs are attached. Now, to our best knowledge, it is not possible to perform this type of action inside an OF switch, but it will when the OF specifications v.1.2 will be ready. It will contain an extension of the usual flow mod command that can match every bit inside the packet header, so we can utilize the IP ID field of the IP header, comparing it to a bitmask, to separate the packet flow. Our implementation splits the flow utilizing another technique; we send the whole packet flow to every PU that must process it, without splitting it. The packet separation is done inside the linux kernel, taking advantage of the netfilter u32 module that can extend a filtering rule matching a certain bit's pattern of a packet.

The integration of the OF switch inside the NS architecture enables advanced features and capabilities in an autonomic management point of view. External services, or the NAME itself, can exploit the JSON-RPC calls to the OFCs (and hence to OF-enabled NS nodes) in order to totally control the data path layer. Gathering several information about the state of the nodes or packet statistics is useful in order to have an accurate monitoring situation of the network. The data path maintains counters divided by table, flow, port and queue, and they include received/transmitted packets and bytes, flow duration, packet matches and drops, frame/overrun/CRC errors, collisions. Using this information, NAME can easily detect fault situations and alarms, and plan requested actions to restore a stable network state.

In the matter of an active network management, i.e. to avoid bottleneck or to achieve a certain network policy, the OF data path can be utilized to directly mangle packet flows, such as dynamically redirect or output it to certain ports, split/join/drop flows, manipulate VLANs tags and priority, change several fields of layer two/three/four headers, create queue linked to an output port in order to provide Quality-of-Service support. All this capabilities are performed in hardware, so they run at wire speed. In addition, as we have mentioned before, NAME can choose to instantiate additional PUs attached to an OF data path to elaborate packets of the same flow, to prevent an overload situation, or to assure a specific output throughput. Additional PUs can also be used to install multiple services onto the same node, for example when the memory or the elaboration limit of a single PU as been reached.

The OFC sends periodically Link Layer Discovery Protocol (LLDP) messages over the network, in order to advertise network devices about the identity and capabilities of the switches that he handles. It also listens for incoming LLDP frames, so that it can reconstruct the network topology (at least inside the local area network where the OF switches are located). NAME can asks this information to OFCs and acquire the knowledge about local network topologies. This will help it to better know the actual context and then define needed actions and strategies to pursue active policies.

According to the system needs, such as the fulfillment of some policies, the NAME can choose to replicate or move to any NetServ node the OFC itself (if it is running inside a NetServ container).