Network Function Virtualization and Path Character

Ted Hardie (hardie@google.com)

Proponents of NFV commonly describe it as a general network architecture to allow virtualization of network element functions. The commonly cited functions are not, however, the routing and switching of the classical end-to-end Internet. The virtualized functions instead generally replace middleboxes: firewalls, load balancers, intrusion detection systems, and the like. In a network which has embraced this approach, the path for a particular flow through the network will be adjusted by an orchestration system or an SDN controller to pass through a set of nodes which provide virtualized functions believed to be appropriate for that flow. This paper reviews the impact of NFV on path character when uncoordinated with that goal and when it is driven by it.

There are several consequences of the NFV approach for path establishment and characterization in a network not focused on delivering specific path characteristics . The first consequence is that since the services chained together may vary from flow type to flow type, the pathway through the network may also vary, potentially substantially. If, for example, a service is provided by an external datacenter or provider, a significant number of route-miles may be added by passing the traffic through an overlay network to that provider. A corollary to this variability is that an end node wishing to assess path characteristics cannot be certain that the path test traffic takes will be the same path that application traffic will take. Given the ability to move virtual functions among container nodes, the end node cannot even be entirely certain that the path used for one session will be the same for subsequent sessions.

A second consequence is that the type of device providing network functions shifts to software systems on commodity servers (VMs, Docker containers, or similar). Because these servers are relatively inexpensive, they can scale out to meet demand. In a network not aiming to manage path characteristics, this scaling may have a variety of impacts on queue management within the path. While each queue within the virtualized function is independent, the hardware queues feeding the individual functions may not be. If the virtualization takes the form of a single function per hardware device, then active queue management of the virtualized function queue will stand in for management of the hardware device; if multiple functions are virtualized onto the same hardware, in contrast, then queue management will be either difficult or lacking. This issue arises even if there are multiple copies of the same function, or indeed whenever there is a lack of coordination across containers.

A third potential consequence of the NFV approach is that its very specialization may cause it to change the queue impact of its function. If a network function providing firewall services is completely separated from other routing and switching activity, will it recognize that high load is a signal of congestion and might be met by active queue management? That is, will NFVs drop packets and/or use ECN to manage load? Or will they see themselves as outside the queue discipline of the path, and scale out to increase the ability to absorb load from the flow?

While they are not required to use ECN or AQM, for a flow that is expecting that signal, its lack may be taken as a sign of a different path character than is actually the case.

These consequences strongly suggest that network function virtualization systems should be deployed with an understanding of their impact on active queue management and path characteristics. With that goal integrated into the network architecture, an NFV approach may provide significantly better performance than one in which particular functions are at fixed places in the network topology.

The first potential benefit would be in the topology itself. As noted above, network functions may be moved among containers by an orchestration system. If that movement is intended to improve specific path characteristics, it may result in substantially better performance. For example, WebRTC relies on tICE servers to provide access to STUN and TURN and for consent freshness. In current deployments, the ICE servers commonly sit in a fixed location outside the IPv4 NAT boundary for a specific administrative domain. When participating nodes are within the same administrative boundary, this causes at least the consent freshness traffic[1] to hairpin out of the boundary and back in. If the ICE servers were provided via an NFV, an orchestration system could move the function closer to the nodes participating when they were inside a shared NAT boundary.[2]

The second potential benefit is in queue management. In the best possible case, the orchestration system may scale a function out by provisioning new systems which are on distinct network paths from those already in use; this ensures that a local-fan out of one function does not result in congestion at the following function or node. A second option is to provision distinct NFV paths, so that the pathway through the virtual instances is never itself the source of congestion. These solutions may not always be possible when alternate paths are either unavailable or are equally loaded, but the implementation of ECN and/or specific drop regimes would provide an alternative which does not require new resources.

Before realizing these benefits, however, we must note an additional challenge. As more and more traffic is moved over encrypted channels, middlebox functions which rely on inspection of packet payload become less useful. To perform their functions they must either terminate the encryption state before inspection (commonly by acting as a back-to-back user agent) or rely on indirect markers which remain in the headers (e.g. port numbers). Neither is satisfactory.

One possible way forward would be to extend the virtualized network functions to the end system, so that they can be performed prior to encryption. In other words, the cooperating

---

[1] See draft-ietf-rtcweb-stun-consent-freshness for details on the mechanism; this is why even with direct IPv6 connectivity STUN is required.

[2] The mechanism for implementing this varies by the type of network; in an SDN network, this redirection could occur using OpenFlow match-actions; in a non-SDN network, an overlay network or anycast IP announcement might be required.

end systems would provide access to containers for the network functions and would chain them in a specified order. By having the end system act as a small scale network of NFVs in this way, it can apply the encryption in the final stage. The trust and update issues for such an approach are, of course, substantial, but it illustrates the difficulty of having both a desire to provide traffic which is confidential to network observers and available to virtualized middlebox functions.

In summary, network function virtualization takes the middle box issues of the current network and frees them from a significant current constraint: fixed topology. Removing that constrain without consideration to path character may result in a number of sub-optimal results, but doing so with that consideration offers some opportunities to improve on the current state.