Synergy between SDN and NFV

Introduction

1. Users don't typically utilize only a single NF
    - This lecture discusses the technology used for multiple NFs to work in tandem
    - Provisioning multiple resources and multiple VNFs with cloud technologies and orchestrating their deployment using SDN addresses this issue

Outline

1. Network functions need appropriate management tools (control plane) for fulfilling realistic user scenarios
2. Basic components required for building a control plane for NFV
    - Dynamically allocating computational resources for hosting the NFs
        - Elastically scaling the instances of NFs to meet the SLAs for packet processing
    - Dynamically programming the network fabric for packet processing with the network functions

Limitations of Monolithic Software Middleboxes

1. Monolithic software middleboxes are not enough
    - NFV products and prototypes tend to be merely virtualized software implementations of products that were previously offered as dedicated hardware appliances
    - Simply virtualizing network middleboxes using NFV replaces monolithic hardware with monolithic software
        - Using previously discussed techniques like DPDK, VT-d, and SR-IOV
    - While this is a valuable first step - as it is expected to lower capital costs and deployment barriers - it fails to provide a coherent management solution for middleboxes
2. Multiple network functions need to be chained together
    - Administrator wants to route all HTTP traffic through:
        - Firewall -> IDS -> Proxy
    - All remaining traffic should be routed through:
        - Firewall -> IDS
    - A common way of achieving this chain is by manually configuring the routing policy
    - Manually configuring this routing policy is complicated
    - Upon node failure, routing needs to be manually updated
3. Dynamic scaling of network functions
    - Workload on each NF can change dramatically
    - Need additional instances to balance the load
        - Make use of elastic nature of virtualized infrastructure
        - Launch new instances on the fly based on demand
    - Stateful NFs require attention to affinity
        - Packets of a given flow should be processed by the same NF instance after scaling -> Flow affinity
        - Network function like Stateful Intrusion Detection System require both directions of traffic -> Session affinity
            * Both directions of load balancing should work in coordination
            * Makes the problem even more complicated
    - Network operators often purchase a customized load balancer for each type of middlebox
        - Requires manual configuration of load balancers when scaling happens
        - Adds another middlebox (load balancer) to the mix, increasing complexity
    - Each middlebox (e.g., Firewall) has its own vendor-specific scaling policy
        - When to trigger scale-in/out? -> Every vendor reinvents the wheel
        - How to make customized load balancer aware of scale-in/out events?
            * Done manually
    - Manual intervention makes scaling slow and leads to overload

    - Potential cause of failures

Need for an NF Control Plane

1. NF misconfiguration is a major problem
    - Examples of failures due to misconfiguration
        - Administrators need to train employees with new hardware in cases of hardware upgrades
        - Misconfiguring software after an upgrade
        - Misconfiguration can be due to incorrect IP addresses, incorrect routing/load-balancing configuration after scaling NF instances
    - All of these issues are faced by all middlebox vendors
        - Need a unified way of configuration and scaling middleboxes

	Misconfiguration	Overload	Physical/Electric
Firewalls	67.3%	16.3%	16.3%
Proxies	63.2%	15.7%	21.1%
IDS	54.5%	11.4%	34%

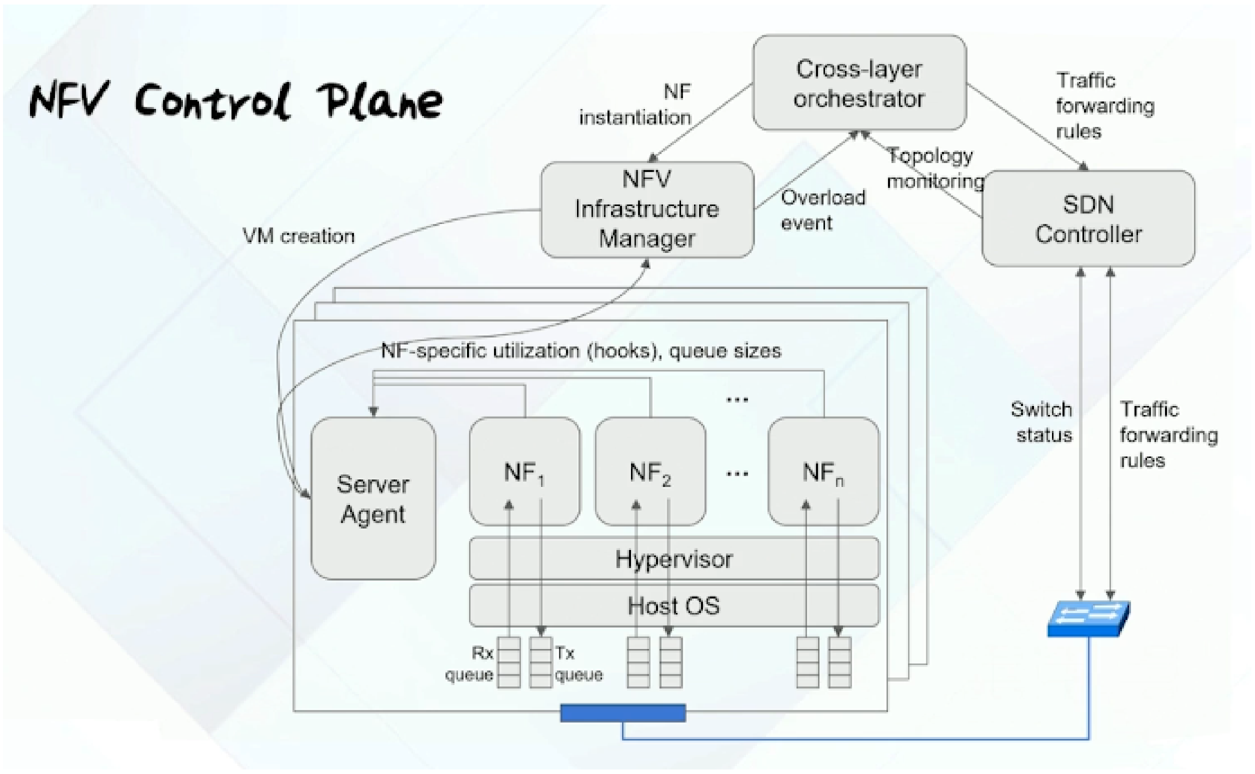
2. Need a control plane for middlebox management
    - Requirements of system admins:
        - Deploy chains of NF
        - Dynamically scale them based on workload
    - Infrastructure: Cluster of servers connected by switches
    - NFV infrastructure manager for deployment of NFs on server cluster
        - e.g., OpenStack
    - SDN to program switches for correct traffic forwarding to realize NF chain

Elements of NF Control Plane

1. OpenStack
    - Cluster of servers converted to a cloud-like IaaS service using the following services:
        - Nova (compute): Management (creation/deletion/migration) of virtual machines by communicating with the hypervisor on each host
        - Cinder (block storage): Persistent storage for VMs
            * Handles lifecycle of each block device - creation/attachment to VM/release
        - Neutron (network): IP address management, DNS, DHCP, load balancing
        - Clange (image): Manages VM images
    - Allows configuration of the cloud environment through API calls to the specific service
        - Create and scale up/down virtual machine instances
        - Exposes monitoring metrics of running instances
2. NFV + SDN Synergy
    - SDN allows programming switches to implement custom traffic forwarding paths
        - Helps implement NF chaining with NFs deployed on an arbitrary set of servers
        - Taking topology info as an input, SDN can setup forwarding rules for packets from upstream NF to downstream NF
    - Switches are programmed using south-bound API of SDN controllers
        - Switches need to be compliant with OpenFlow
        - SDN programs a switch by specifying which port a packet should be forwarded to based on its packet headers
        - Switch can also be programmed to modify packet headers, e.g., change destination MAC address or add VLAN tag

NF Control Plane Architecture

1. NFV control plane
   - NFV Infrastructure Manager handles scaling of functions
   - SDN controller manages traffic forwarding
   - Scaling and forwarding decisions are made by cross-layer orchestrator
   - Topology information is needed for deciding traffic forwarding rules
   - NFV Infrastructure Manager and SDN controller are the basic components needed to meet requirements of NFV control plane orchestration
     - These components are run in coordination by the cross-layer orchestrator



NFV Control Plane Architecture

2. Main concerns of cross-layer orchestrator
   - Virtualization platform tasks
     - How to place NFs across the compute cluster?
     - How to adapt placement decisions to changing workloads? (scaling)
   - Network programming tasks
     - How to setup complex forwarding between NFs?
     - How to ensure affinity constraints fo NFs?
       * All packets of a given flow must be processed by the same NF instance
       * Packets in both directions of a connection should be processed by the same NF instance
     - Ensure consistent routing for output packets of NFs that modify packet headers
       * For example: NAT changes IP header

Virtualization Platform Tasks

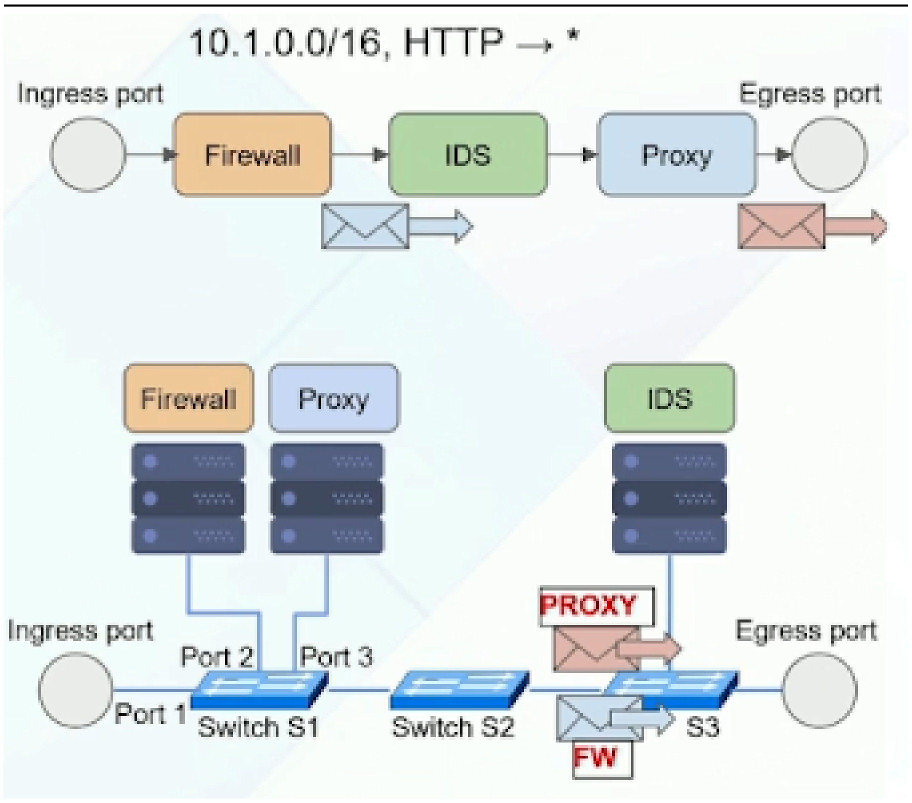
1. Placement of network functions
   - System administrator specifies an NF chain

    - Nodes represent NF or physical ports (network interfaces)
    - Edges represent data transfer between nodes. Each edge can be annotated with expected traffic rate
  - Right sizing
    - Use initial estimates of expected traffic rate to determine load on an NF and per-core capacity to determine number of instances of each NF
  - Optimizing the NF graph
    - Number of edges bewteen stages can be minimized by taking affinity constraints into account
    - For example: If both firewall and IDS need flow-affinity, we can simplify the graph
  - Placement of instances on compute cluster
    - Objective: Minimize inter-server traffic
      * Intra-server software forwarding is faster
      * Inter-server link bandwidth is a limited resource
    - Problem can be reduced to graph partitioning problem
      * NP hard

2. Dynamic scaling of NF instances
   - Hypervisor maintains a Server Agent to monitor system metrics
     - Queue sizes
     - Provide hooks to NFs to report instantaneous load-related metrics
       * e.g., number of active flows (load balancer), cache hits/misses (proxy)
   - Uses metrics to determine overload of a particular network function
   - Cross-layer orchestrator uses NFV Infrastructure manager to instantiate new instance of NF (i.e., "scale-out")
   - SDN controller is used to redirect traffic to the new NF instance
     - While maintaining flow/connection affinity
     - Old flows are still sent to the old instance
     - Only new flows are sent to the new instance
   - A similar approach can be used to "scale-in" NF instances

Network Programming Tasks

1. How to perform unambiguous forwarding?
   - How to identify whether a packet has traversed only firewall, or both firewall and IDS?
   - This determines the destination of that packet
2. Identify stage of NF chain that a packet belongs to
   - Use TCP/IP headers to identfiy the NF chain that the packet is part of
   - To determine which segment of the chain that this packet is part of
     - Use topology information: e.g., packet arriving on S1 on port 2 has traversed firewall and destined to IDS
     - Tag packets with flow state information: e.g., on packets exiting firewall and those exiting proxy enter S3 through port 1
       * Can embed chain segment information using dest MAC addr, VLAN tags, MPLS tags, or unused fields in IP header
       * Tag each packet of a flow on very first switch it encounters (S1)
3. Forward packets to the correct downstream instance
   - Multiple instances of same network function exist for same chain
   - Use a high-level policy to determine which NF instance should process this flow/connection (based on affinity)
     - Example 1: Round-robin (uniformly distribute flows/connections between NF instances)
     - Example 2: Weight round-robin (distribute flows/connections in order to equalize resource utilization)



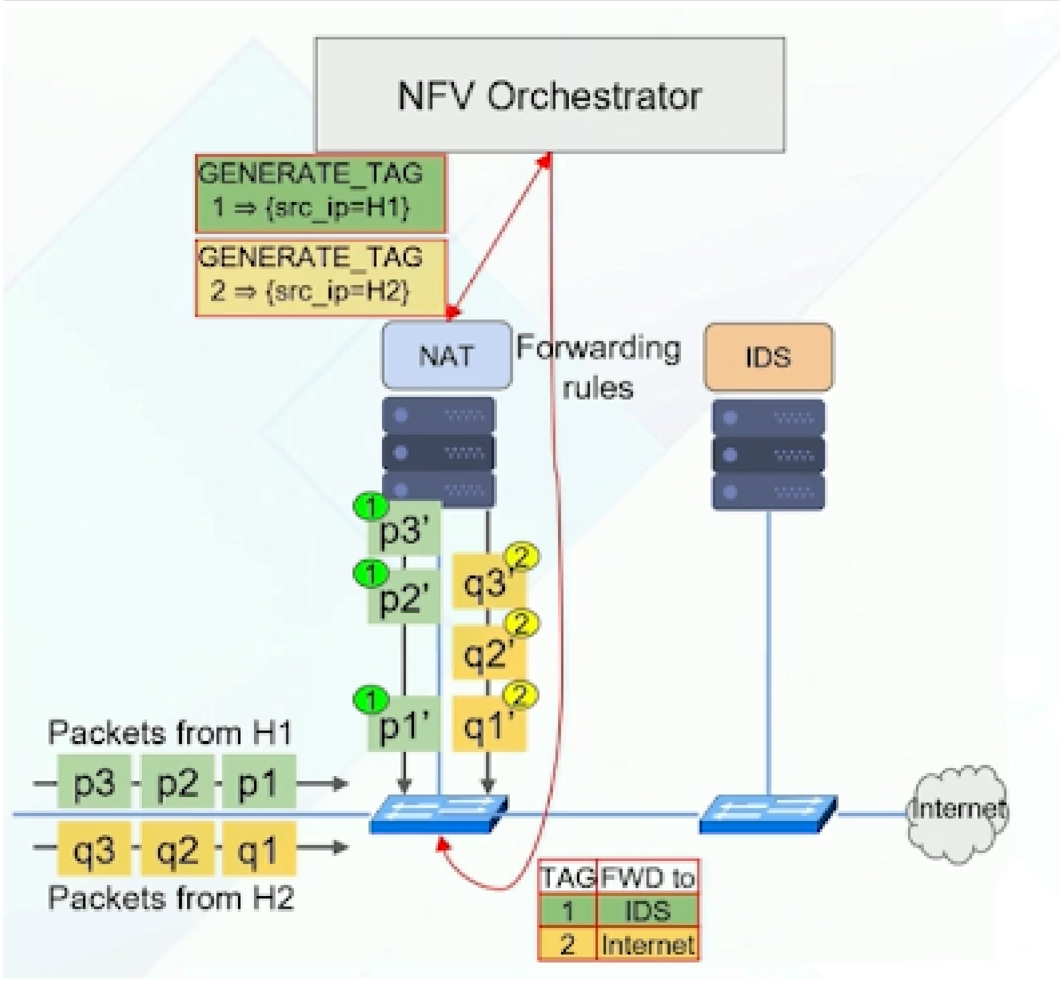
NFV Chaining

4. Satisfying affinity requirements of NFs
   - Flow-level affinity
     - Requirement: Direct all packets of a particular flow to a particular NF instance
     - Solution: Use OpenFlow packet header matching to identify flow and forward packets
   - Connection-level affinity
     - Requirement: Direct all packets of traffic in both directions to particular NF instance
     - Solution: Maintain flow to NF instance mapping in orchestrator. For the backward direction flow, the mapping should select the same instance that was used for forward direction flow

NFVs that Modify Packets

1. Handling NFs that modify packet headers and payload
   - Many NFs actively modify traffic headers and contents
     - For example, NATs rewrite the IP addresses of individual packets to map internal and public IPs. Other NFs such as WAN optimizers may spawn new connections and tunnel traffic over persistent connections
2. Possible solution: Instrument NFs to expose header/payload transmformations
   - Ideally, we would like fine-grained visibility into the processing logic and internal state of each NF to account for such transformations
   - Long-term solution is to have NFs expose such transformation information to cross-layer orchestration (e.g., mapping between internal and external IP addresses for NAT)
   - Limitation: Vast array of middleboxes and middlebox vendor + proprietary nature of middlebox functionality -> achieving standardized APIs and requiring vendors to expose internal states does not appear to be a viable near-term solution
3. Types of packet modification by middleboxes
   - No packet modification

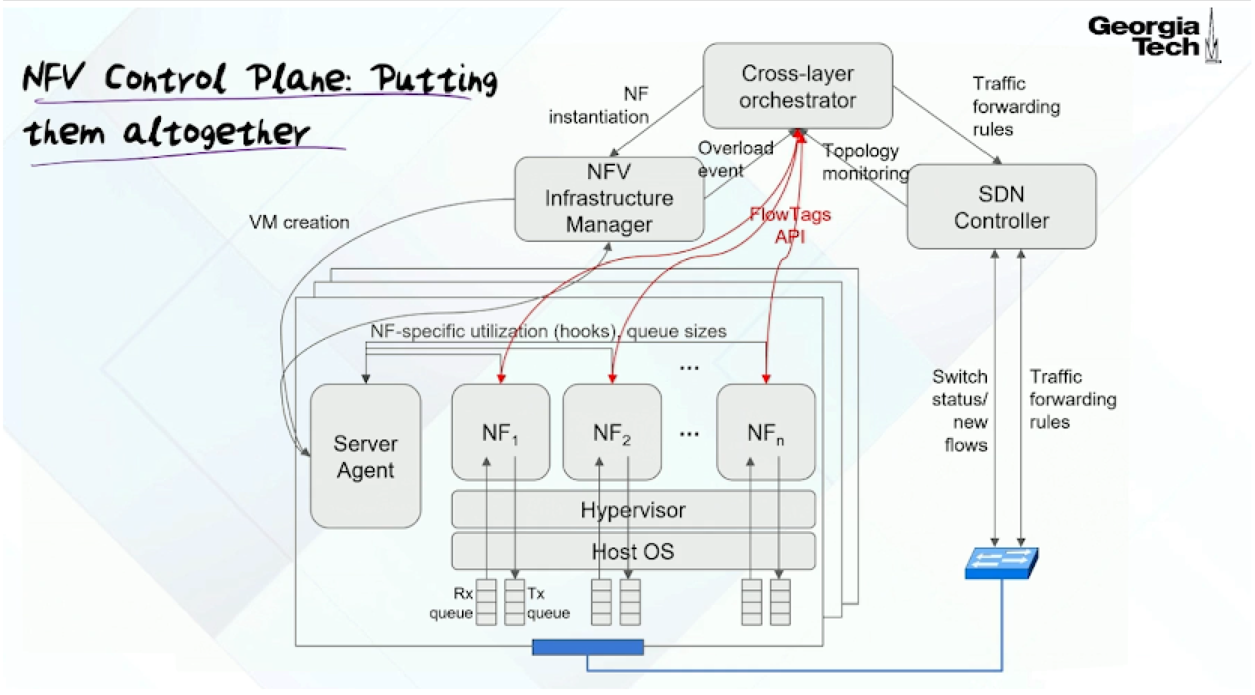
    - Firewall, IDS, IPS
    - TCP/IP headers match exactly
  - Header modification
    - NAT, load balancer
    - Packet payload matches exactly
    - Packet transformation occurs at the timescale of each flow
  - Payload modification
    - WAN optimization, HTTP proxy (change HTTP header)
    - High correlation in packet payload
    - Packet transformation occurs at the timescale of each session
  - Complex modifications
    - Encrpyted VPN, compression
4. Flow correlation
  - Packet collection at SDN controller
    - For each new flow coming into the middlebox, collect first P packets
    - Collect first P packets for a time window W for all flows going out of the middlebox
  - Calculate payload similarity
    - For each (ingress, egress) pair of flows compute a similarity score
    - Use Rabin fingerprint to hash chunks of packet stream to detect overlapping content
  - Identify most similar flow
    - Identify egress flow with highest similarity score with the ingress flow
  - No modification to middleboxes required, but flow correlation is not 100% accurate
    - Does not support complex packet modifications
5. FlowTags
  - Allow middleboxes to "tag" packets based on the internal content
    - Middleboxes generate tags for each flow based on the processing content
    - Use the SDN controller to keep mapping from tag to original IP header (in figure below, only src ip is assumed to be part of header)
    - Perform tag-based forwarding using OpenFlow



NFV FlowTags

Putting it All Together

1. NFV control plane: end-to-end system providing scalable NF chains with two components
   - Virtual Infrastructure Manager: e.g., OpenStack
     - Virtualizes a cluster of servers
     - Provides horizontal (scale-in/out) elasticity)
   - SDN-controlled network
     - Allows programming network datapath for complex routing policies
   - Combine functionalities using cross-layer orchestrator
     - Efficient placement of NF instances
     - Trigger scaling in/out of instances based on observed workload
     - Setup complex NF chains
     - Ensure flow and connection affinity
     - Handle NFs that modify packets (header and payload)



NFV Controller

Conclusion

1. Design an NFV control plane that allows chaining multiple network functions together and scaling network function instances in and out using cloud technologies and SDN