主要内容

- 软件定义网络 SDN（Software-Defined Networking）
  - SDN数据平面
  - SDN控制平面
  - SDN的应用
- 网络虚拟化
  - NFV的基本功能
  - 网络功能（Internet Middle box）（补充）
- 小作业2 讨论
- 大作业

SDN

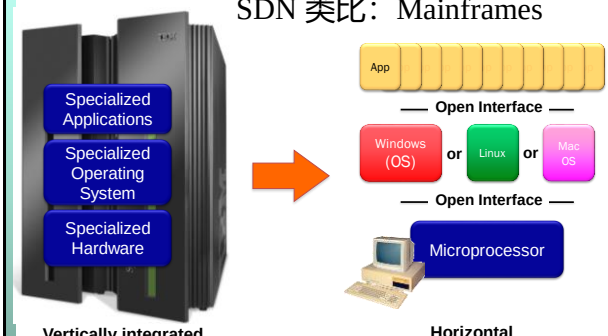
- 软件定义网络（Software-Defined Networking）
  - A network in which the control plane is physically separate from the forwarding plane, and a single control plane controls several forwarding devices.
- 两个阶段
  - In Phase 1, network operators took ownership of the control plane
  - In Phase 2, taking control of how packets are processed in the data plane —*正在进行中*

SDN的设计原则

- 数据平面和控制平面分离（Disaggregating the Control and Data Planes）
  - 数据平面：分组转发设备，如白盒交换机（white-box switches）
    - 定义转发抽象
  - 控制平面：软件实现各种功能
    - the control plane should be fully independent of the data plane and logically centralized

"Networks will [hopefully] be programmed by many, and operated by few."

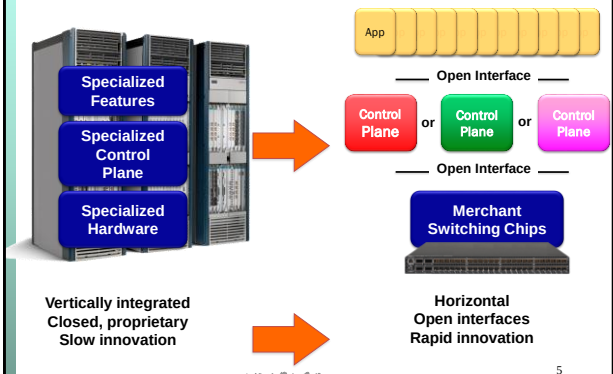
SDN 类比：Mainframes



Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

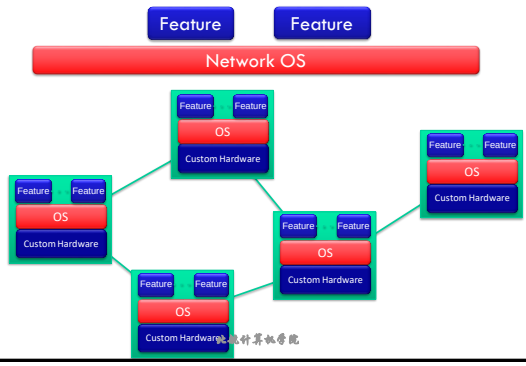
路由器 Routers/ 交换机Switches



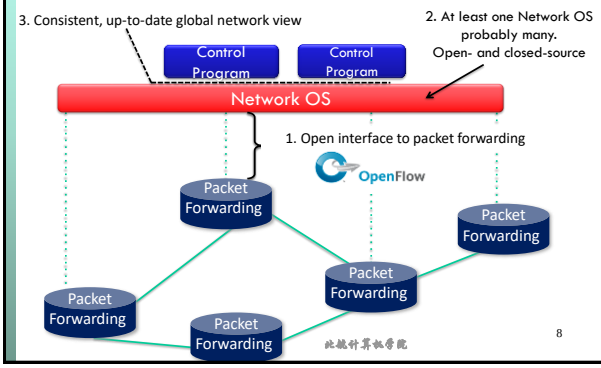
软件定义网络SDN的特点

- **集中控制**
  - 集中式控制平面
  - 获得网络资源全局信息，生成流（flow）转发规则
  - 全局资源配置和优化：流量工程，负载均衡等
- **开放接口**
  - 简单、快速的数据平面
  - 南向接口和 北向接口：应用和网络无缝集成
  - openflow：南向接口
- **网络虚拟化**
  - 屏蔽物理设备差异；逻辑网络与物理网络分离

网络的变化

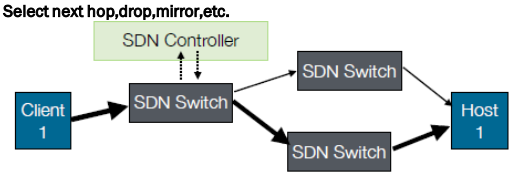


SDN的组件



SDN 工作流

- 数据平面 Data plane
  - 交换机switches（软/硬）：maintains a flow table
  - Flow = one point-to-point connection (Src/Dest IP and port)
  - Action = how switch should process the packet
- 控制平面 Control plane
  - populates flow table rules on switches
  - Can be based on business logic
  - Select next hop,drop,mirror,etc.



控制平面

- 网络操作系统Network OS
  - 分布式系统：distributed system that creates a consistent, up-to-date network view
  - Runs on servers (控制器controllers) in the network
  - 开源系统：NOX, ONIX, Floodlight, Trema, OpenDaylight, HyperFlow, Kandoo, Beehive, Beacon, Maestro, ... + more
- 控制程序Control Program
  - 网络应用：Control program operates on view of network（网络视图）
    - 输入: global network view (graph/database)
    - 输出: configuration of each network device
  - 应用本身可以不是分布式系统
    - Abstraction hides details of distributed state

数据平面

- 转发抽象 Forwarding Abstraction
  - 获取状态信息：Get state information from forwarding elements
  - 控制： Give control directives to forwarding elements
- 灵活性Flexible
  - 动作：Behavior specified by control plane
  - 原语：Built from basic set of forwarding primitives
- 最小化Minimal
  - Streamlined for speed and low-power
  - Control program not vendor-specific
- 协议
  - OpenFlow is an example of such an abstraction

SDN的相关研究

- Original Papers
  - 4D (2004/2005)
  - SANE/Ethane (2006/2007)
  - OpenFlow (2008)
  - Onix (2010)

explosion

- Open Source) Software
  - Open vSwitch (2009)
  - NOX/POX (2009/2011)
  - Beacon (2010)
  - Trema (2011)
  - Floodlight (2011/2012)
  - Ryu (2011/2012)
  - OpenDaylight (2013)
  - ONOS (2014)

控制器 Controllers

SDN的现状

- 开放网络基金会（ONF）支持OpenFlow协议，只更新到1.5.1版本（2015）
  - 主要应用场景：数据中心网络
- 控制器
  - 早期：六个SDN控制器NOX，POX，Floodlight，ONOS，OpenDaylight和Ryu
  - 目前：开源控制器OpenDaylight社区已经发布了11个版本，并且有50多个供应商支持社区。可帮助企业和服务提供商简化网络管理，包括物理网络和虚拟网络。
  - 版本庞杂；多种设备厂商的专用API

SDN的应用

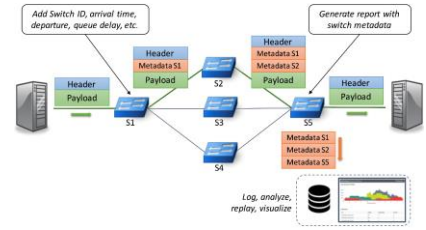
- 云服务提供商 cloud providers
  - Google, Facebook, and Microsoft，开源组件
- 大型网络运营商（large network operators）
  - AT&T, DT, NTT, and Comcast，接入网络
- 园区网（企业和大学）
  - 大学：supporting research and innovation
  - 企业：managed edge services offered by cloud providers
  - 产品：helping enterprises manage virtual networks.

应用：虚拟网络Virtual Networks

- 虚拟网络
  - Virtual Private Networks (VPNs)
  - Virtual Local Area Networks (VLANs)
- SDN context：使VLAN易于使用
  - set up, managed, and torn down programmatically
- 虚拟机环境下，虚拟网络管理
  - .VMWare's vSphere，NSX is the virtual network management subsystem of vSphere
  - OpenStack，a virtual network subsystem called Neutron
- 局限性
  - 配置，而非控制

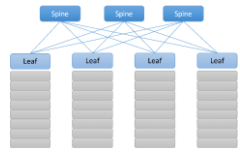
应用：网络遥测Network Telemetry

- In-Band Network Telemetry (INT)
  - 在处理分组时收集网络状态信息
- 指令编码
  - telemetry "instructions" are encoded into packet header fields



应用：交换网络Switching Fabrics

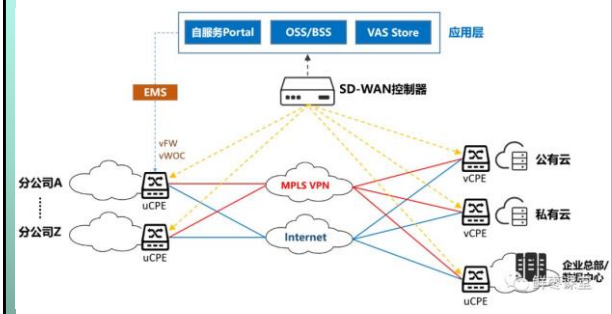
- 云数据中心cloud datacenters
  - a leaf-spine topology
  - 机架内： L2 forwarding (bridging) within a server-rack
  - 机架间：L3 forwarding (routing) across racks.
  - *Equal-Cost Multipath (ECMP)*



应用：广域网SD-WAN

- 云服务：数据中心之间的广域网链路，流量工程
  - 用于企业网络、数据中心、互联网应用及云服务
  - 降低了流量成本，提高带宽
  - 最优路径选择，实现负载均衡，保证服务质量
  - Google B4：*Traffic Engineering (TE)* control program
- Gartner Glossary
  - *SD-WAN provides dynamic, policy-based, application path selection across multiple WAN connections and supports service chaining for additional services such as WAN optimization and firewalls.*

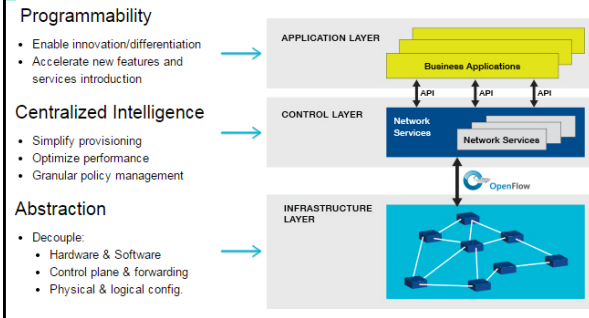
例子

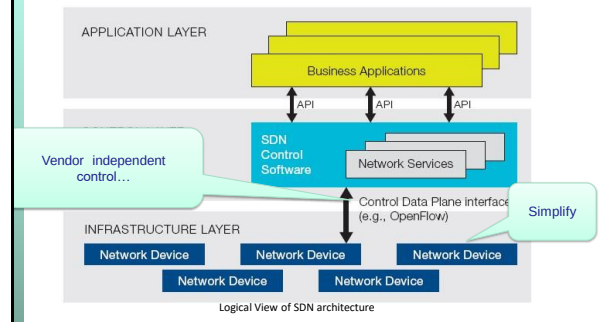


开放网络基金会 ONF

- 开放网络基金会（Open Networking Foundation，ONF）
  - 2011年由Deutsche Telekom, Facebook, Google, Microsoft, Verizon, 和 Yahoo!创立的非盈利组织，ONF已经拥有140多家会员
  - 2012年4月，ONF发布白皮书《Software-Defined Networking: The New Norm for Networks》
  - ONF推广开放SDN和OpenFlow技术及标准，加速开放SDN的部署和应用
  - 主流网络运营商
- ONF定义的SDN架构
  - 应用层：网络业务逻辑开发，资源编排
  - 控制器层：全局网络的管理
  - 基础设施层：各种网络设备，负责数据的转发

ONF SDN架构定义





Logical View of SDN architecture

ONF SDN中的接口

- 北向接口(NBI, Northbound Interface)
  - 控制器层与应用层的接口
  - 通过对北向接口的封装，应用层以软件编程的形式调用各种网络资源和把控整个网络的资源状态，并对资源进行统一调度。
  - 与具体应用需求相关，具有多样化的特征，很难统一
- 南向接口(SBI, Southbound Interface)
  - 控制器与基础设施之间的接口
  - OpenFlow协议：用于控制器和交换机之间的通信，控制器可以通过OpenFlow下发流表控制交换机，交换机也可以反馈信息给控制器，同时，OpenFlow也规定了交换机对报文的转发方式

ONOS

- 2015年6月，开放网络基金会ONF正式推出了一个开源SDN实现平台：Atrium
  - Atrium是分布式开源软件，集成了开源SDN组件
- 开放式网络操作系统(ONOS)： ONOS 1.5.0 (Falcon)
  - 可运行在支持OpenFlow的交换机或控制器上
- 面向服务提供商和企业骨干网
  - 目标：满足运营商提供敏捷和灵活的需求，摆脱设备供应商的限制
  - 降低网络的建设和维护成本

资源：

https://opennetworking.org/onos/

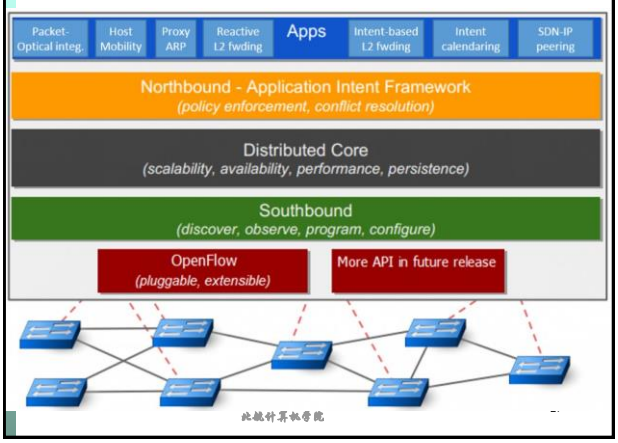
https://wiki.onosproject.org/

ONOS架构

- 分层结构
  - 应用层
  - 北向核心接口层
  - 分布式核心层
  - 南向核心接口层
  - 适配层
  - 设备层
- 其中南向核心接口层和适配层可以合起来称作南向抽象层，它是连接ONOS核心层与设备层的重要桥梁

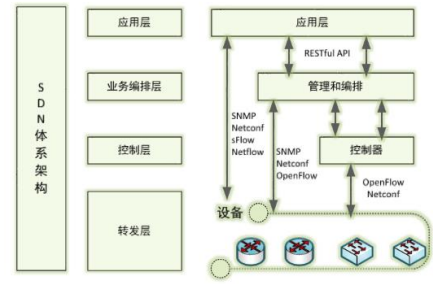
ONOS部署

- ONOS可以作为服务部署在集群和服务器上
  - 分布式核心平台：每个服务器上运行相同的ONOS软件，可以快速地进行故障切换
  - 可靠性极高的环境，将SDN控制器特征提升到运营商级别
- 南向抽象层
  - 由网络单元构成，它将每个网络单元表示为通用格式的对象
  - 通过这个抽象层，分布式核心平台可以维护网络单元的状态，而不需要知道底层设备的具体细节。
  - 南向接口确保了ONOS可以管控多个使用不同的协议的不同设备



南向接口的扩展

- 南向接口除了OpenFlow协议，也包含了NETCONF、SNMP等协议（支持已有设备）



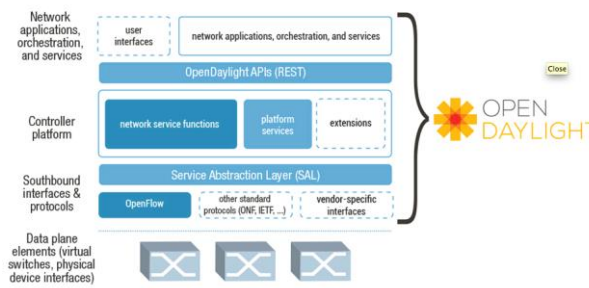
OpenDaylight

- 2013年4月8日，OpenDaylight开源项目推出
  - 参与者主要来自设备厂商，其中包括思科、Juniper等传统网络设备巨头，IBM、微软等传统IT软硬件设备巨头，还包括Arista、Big Switch等新兴网络设备厂商，以及VMware、红帽、思杰等新兴IT软件厂商
  - 与Linux基金会合作，其目标是成为SDN架构中的核心组件
  - 减少网络运营的复杂度，扩展其现有网络架构中硬件的生命期，支持SDN新业务和新能力的创新
- https://www.opendaylight.org/

OpenDaylight架构

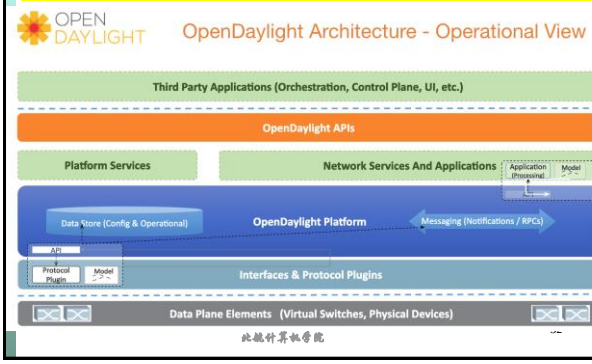
- 由应用服务层、控制平面层、南向接口层和数据平面层四层构成
  - 为应用(App)提供开放的北向API。支持OSGi 框架和双向的REST接口。
- 控制层提供基本网络服务和一些附加的网络服务
  - 这些附加服务都可以通过插件的形式安装加载
- 南向通过Plugin的方式来支持多种协议
  - 这些模块被动态挂载到服务抽象层(SAL)，SAL为上层提供服务，将来自上层的调用封装为适合底层网络设备的协议格式。
  - 可能暴露设备的细节给应用程序
- 由设备商主导的一个开源控制器，开放专用接口的方式保留传统设备

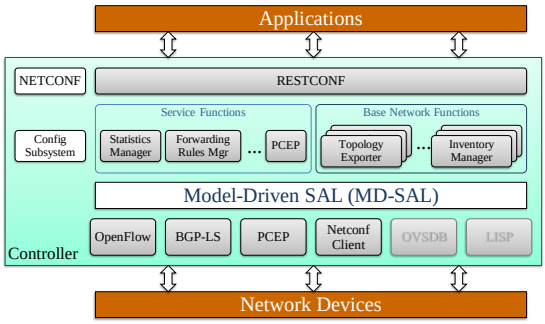
OpenDaylight



The OpenDaylight Sodium

the 11th release of the most secure, stable, scalable and performant (S3P) pervasive open source SDN controller platform.





SDN数据平面

OpenFlow协议

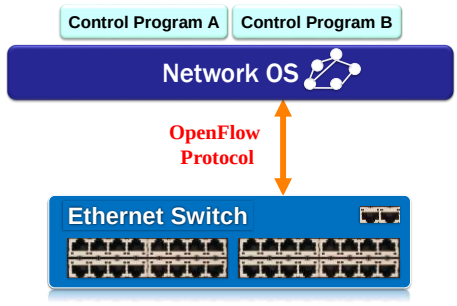
OpenFlow

- **基本概念**
  - **通信协议：**A communication protocol that gives access to the forwarding plane of the network switch or router
  - **基于流（flow）的控制**
- **特征**
  - **指令集合**
  - **控制平面和数据平面分离**
    - **数据平面 a Flow Table, and an action associated with each flow entry**
    - **控制平面 a controller which programs flow entry in the flow table**
- **组件**
  - **控制器：**OpenFlow controller
  - **交换机：**OpenFlow switch

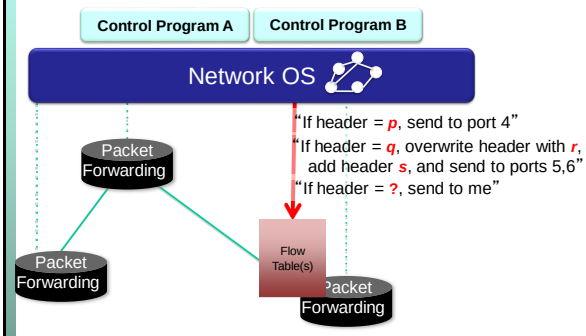
概念：SDN 与 OpenFlow

- SDN is a concept of the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.
- *OpenFlow* is communication interface between the control and data plane of an *SDN architecture*.
  - Allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual.
  - **协议：**Think of as a protocol used in switching devices and controllers interface.

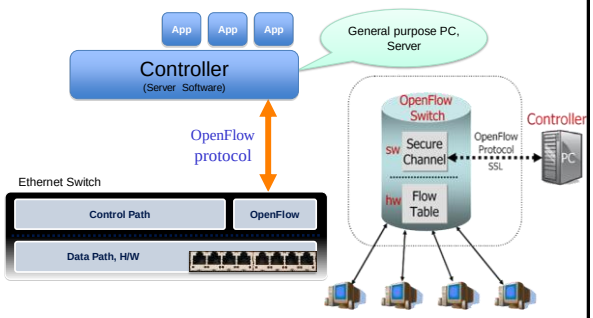
OpenFlow



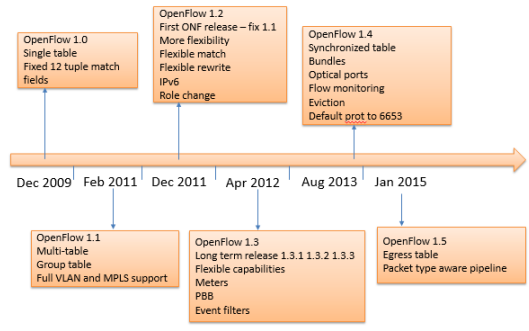
OpenFlow



OpenFlow 交换机和流表



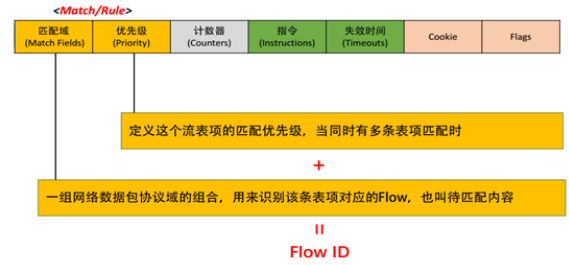
Openflow 版本演进



OpenFlow 流表

- 流 flow: 由各层协议头部域定义
- 通用转发 generalized forwarding: 分组处理规则
- 例：OpenFlow v1.3中流表项主要由7部分组成
  - 匹配域：用来识别该条表项对应的flow
  - 优先级：定义流表项的优先顺序
  - 计数器：用于保存与条目相关统计信息
  - 指令：匹配表项后需要对数据分组执行的动作
  - Timeouts、Cookie、Flags

OpenFlow 流表



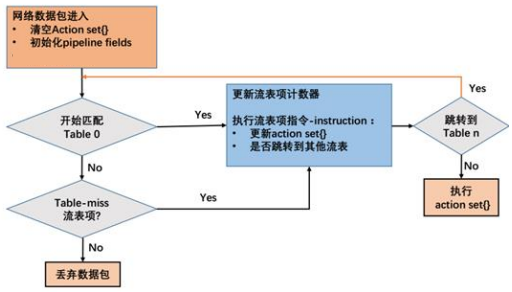
总结：Openflow交换机

- 一个Openflow交换机包括一个或者多个流表flow table和一个组表group table
- 流表中每个流条目包括三个部分
  - 匹配match—使用ingress port，packet header以及前一个flow table传递过来的metadata
  - 计数counter —对匹配成功的包进行计数
  - 操作instruction—修改action set或者流水线处理

例：OpenFlow v1.3 流表匹配流程

- 交换机解析进入设备的数据分组，然后从table 0开始匹配，按照优先级高低依次匹配该流表中的流表项
- 通常根据数据分组的类型，分组头的字段例如源MAC地址、目的MAC地址、源IP地址、目的IP地址等进行匹配。也可以通过数据分组的入端口或元数据信息来进行数据分组的匹配
- 如果匹配成功，则按照指令集里的动作更新动作集，或更新数据分组/匹配集字段，或更新元数据和计数器。
- 根据指令是否继续前往下一个流表
- 若数据分组匹配失败，如果存在无匹配流表项（table miss）就按照该表项执行指令。一般是将数据分组转发给控制器、丢弃或转发给其他流表。如果没有table miss表项则默认丢弃该数据分组

OpenFlow v1.3 流表匹配流程



OpenFlow 通信过程

- 建立安全通道
  - OpenFlow控制器是通过SSL/TLS（安全传输层协议）和OpenFlow交换机进行通信的
  - 控制器与交换机之间通过服务器证书和客户机证书进行认证。在一些OpenFlow版本中（1.1及以上），控制器和交换机之间的连接有时也会通过TCP明文来实现
- OpenFlow控制器启动后，对指定端口进行监听，默认的TCP端口为6633，后更改为6653。通过三次握手后，连接建立

OpenFlow 协议格式

- OpenFlow 消息结构
  - 版本号Version：OpenFlow 消息的版本号
  - 类型Type：指示消息类型，以及如何解析负载
  - 消息长度：指示消息开始和结束的位置
  - 标识Transaction ID (xid)：用来匹配请求和响应

Bit Offset	0 ~ 7	8 ~ 15	16 ~ 23	24 ~ 31
0 ~ 31	Version	Type	Message Length	
32 ~ 63	Transaction ID			
64 ~ ?	Payload			

OpenFlow 协议消息

- 三类消息
  - Controller-to-switch：控制器发出消息，用于管理和检查交换机状态
  - Asynchronous异步消息：交换机发出，用于将网络事件和交换机状态改变信息更新到控制器
  - Symmetric对称消息：交换机和控制器都可以发出

Controller-to-Switch

控制器发出消息

- ◆Features 用来获取交换机特性
- ◆Configuration 用来配置Openflow交换机
- ◆Modify-State 用来修改交换机状态(修改流表)
- ◆Read-Stats 用来读取交换机状态
- ◆Send-Packet 用来发送数据包
- ◆Barrier 阻塞消息

Asynchronous异步消息

交换机发出消息

- ◆Packet-in 用来告知控制器交换机接收到数据包
- ◆Flow-Removed 用来告知控制器交换机流表被删除
- ◆Port-Status 用来告知控制器交换机端口状态更新
- ◆Error 用来告知控制器交换机发生错误

Symmetric对称消息

对称消息，可以由控制器或交换机主动发起

- ◆Hello 用来建立Openflow连接
- ◆Echo 用来确认交换机与控制器之间的连接状态
- ◆Vendor 厂商自定义消息

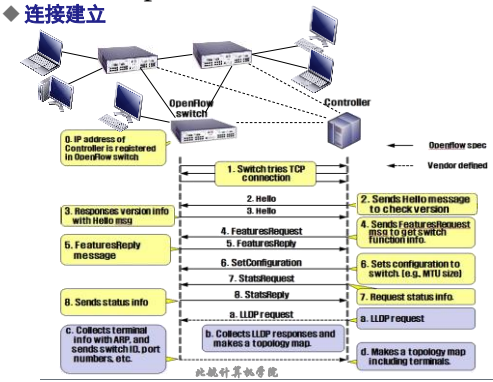
OpenFlow 协议消息

C: OpenFlow Controller AM: Asynchronous message CSM: Control/Switch Message
S: OpenFlow Switch SM: Symmetric Message

Category	Message	Type	Description
Meta Info. Configuration	Hello (SM)	C → S	following a TCP handshake, the controller sends its version number to the switch.
	Hello (SM)	S → C	the switch replies with its supported version number.
	Features Request (CSM)	C → S	the controller asks to see which ports are available.
	Set Config (CSM)	C → S	in this case, the controller asks the switch to send flow expirations.
	Features Reply (CSM)	S → C	the switch replies with a list of ports, port speeds, and supported tables and actions.
	Port Status	S → C	enables the switch to inform that controller of changes to port speeds or connectivity..
Flow Processing	Packet-In (AM)	S → C	a packet was received and it didn't match any entry in the switch's flow table, causing the packet to be sent to the controller.
	Packet-Out (CSM)	C → S	Instructs a switch to send a packet out to one or more switch ports.
	Flow-Mod (CSM)	C → S	instructs a switch to add a particular flow to its flow table.
	Flow-Expired (CSM)	S → C	a flow timed out after a period of inactivity.

OpenFlow 通信过程

◆ 连接建立



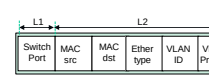
例： 流表项字段

◆ 流表Flow Table

Counters used when controller calculates paths

Flow entry	match field	counter	Action (Instruction)	priority	Timeout	cookie
1						
n		...				

Actions(Instructions)
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields
6. Etc.



- Match field= L1~L4 header information
  - OpenFlow 1.0 → 12 tuples
  - OpenFlow 1.1 → 15 tuples
  - OpenFlow 1.3 → 40 tuples (158 bytes)

Match fields of OpenFlow 1.1

例：流表项字段（续）

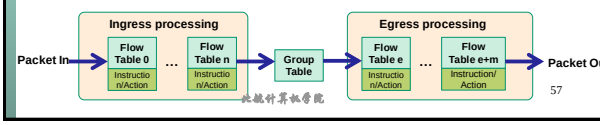
◆ Flow Table

Operation Mode	Switch Port	MAC src	MAC dst	Ether type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S_port	TCP D_port	Action	Counter
Switching	*	*	00:1f..	*	*	*	*	*	*	*	Port1	243
Flow Switching	Port3	00:20..	00:2f..	0800	vlan1	1.2.3.4	1.2.3.9	4	4666	80	Port7	123
Routing	*	*	*	*	*	*	1.2.3.4	*	*	*	Port6	452
VLAN Switching	*	*	00:3f..	*	vlan2	*	*	*	*	*	Port6 Port7 Port8	2341
Firewall	*	*	*	*	*	*	*	*	*	22	Drop	544
Default Route	*	*	*	*	*	*	*	*	*	*	Port1	1364

流水线 Pipelining

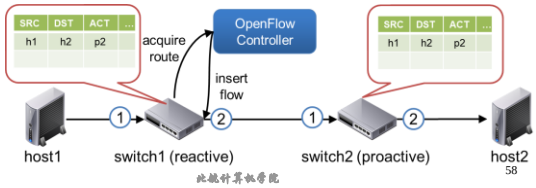
◆ Pipelining

- 流表顺序编号，从0开始
- 分组按流表顺序处理 (version 1.1)
  - If a flow entry is found, the instruction set included in that flow entry is executed
  - Instructions may explicitly direct the packet to another flow table ("goto-table")
  - Pipeline processing can only go forward and not backward
- 两阶段处理：Two stage pipeline processing (version 1.5)
  - 输入处理 Ingress processing
    - Mandatory, performed before egress processing, use the rules specified in ingress tables
  - 输出处理 Egress processing
    - Optional, performed in the context of output port, use the rules specified in egress tables
    - Egress table can be configured during feature request/reply phase
- Useful to manage complicated processing
  - E.g., table 1 for VLAN processing, table 2 for multicast group processing

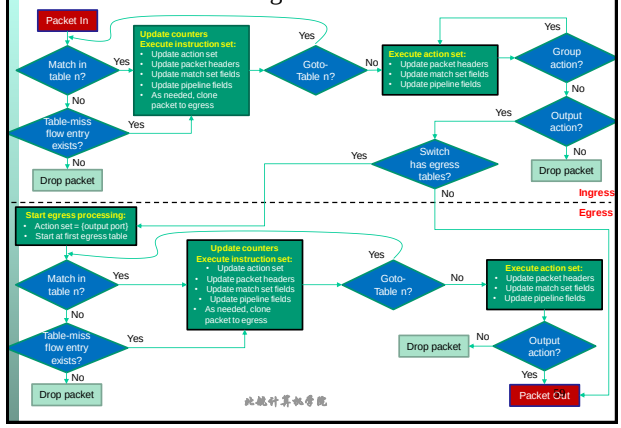


OpenFlow的分组转发

- 分组转发 Packet Forwarding
  - 反应式流插入 Reactive flow insertion
    - A non-matched packet reaches to OpenFlow switch, it is sent to the controller, based on the info in packet header, an appropriate flow will be inserted
    - Always need to query the path from controller during packet arrival → slow
    - Can reflect the current traffic status
  - 主动流插入 Proactive flow insertion
    - Flow can be inserted proactively by the controller to switches before packet arrives
    - No need to communicate during packet arrival → fast packet forwarding
    - Cannot reflect the current traffic status



Packet Processing Flowchart in OF Switch



OpenFlow的指令

- 指令 Instructions
  - 分组匹配流表项后，执行指令
  - 改变： action set and/or pipeline processing

Syntax	Description
Meter *meter_id*	Direct packet to the specified meter
Apply-Actions *actions*	Apply the specific actions immediately. Execute multiple actions of the same type.
Clear-Actions	Clear all the actions in the action set immediately
Write-Actions *actions*	Merge the specified actions into the current action set, if exists try to overwrite, otherwise try to add.
Goto-Table *next-table-id*	Indicate the next table in the processing pipeline. The table-id must be greater than the current table-id.

OpenFlow的动作

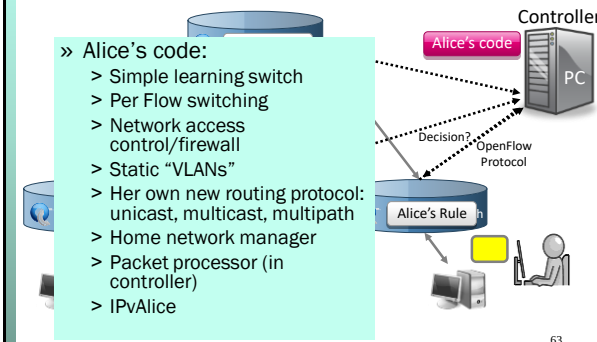
- 动作 Actions
  - 动作与每个分组关联
  - 当指令集合不包含 Goto-Table 指令时，停止pipeline 处理，并执行 actions

Syntax	Description
set	Apply all set-field actions to the packet
qos	Apply all QoS actions, such as set_queue to the packet
group	If a group action specified, apply the actions of the relevant group bucket(s) in the order specified by this list
output	If no group action is specified, forward the packet on the port specified by the output action
push_MPLS	Apply MPLS tag push action to the packet
push_VLAN	Apply VLAN tag push action to the packet
pop	Apply all tag pop actions to the packet

流表统计项： Basic Stats

Per Table	Per Flow	Per Port	Per Queue
Active Entries	Received Packets	Received Packets	Transmit Packets
Packet Lookups	Received Bytes	Transmitted Packets	Transmit Bytes
Packet Matches	Duration (Secs)	Received Bytes	Transmit overrun errors
	Duration (nanosecs)	Transmitted Bytes	
		Receive Drops	
		Transmit Drops	
		Receive Errors	
		Transmit Errors	
		Receive Frame Alignment Errors	
		Receive Overrun erorrs	
		Receive CRC Errors	
		Collisions	

例1：OpenFlow 应用

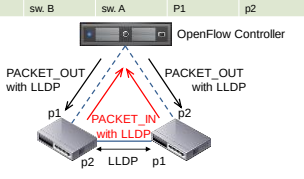


» Alice's code:
- > Simple learning switch
- > Per Flow switching
- > Network access control/firewall
- > Static "VLANs"
- > Her own new routing protocol: unicast, multicast, multipath
- > Home network manager
- > Packet processor (in controller)
- > IPvAlice

例2：OpenFlow的拓扑发现

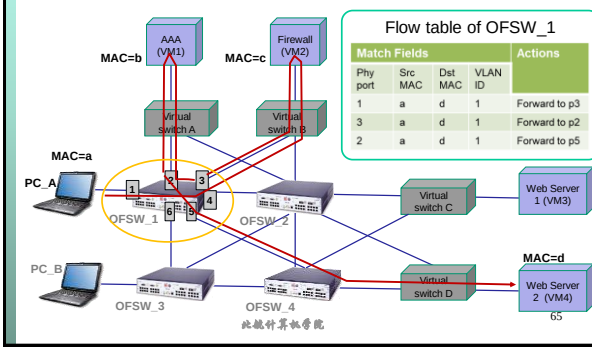
- ◆ 目标
  - ❖ To construct an entire network view
- ◆ 协议
  - ❖ Use the Link Layer Discovery Protocol (LLDP)

IDX	SRC	DST	SRC PORT	DST PORT
153	sw. A	sw. B	p2	p1
…	…	…	…	…
357	sw. B	sw. A	P1	p2



例3：路由控制

- ◆ Example of Routing Control (hop-by-hop routing)



Flow table of OFSW_1

Match Fields				Actions
Phy port	Src MAC	Dst MAC	VLAN ID	
1	a	d	1	Forward to p3
3	a	d	1	Forward to p2
2	a	d	1	Forward to p5

例4：测量网络时延

- 在SDN中测量交换机之间链路的时延
  - 控制器向交换机A下发一个Packet_out报文，指示交换机将其泛洪或者转发到某端口
    - 该报文应携带哪些必要信息？（如时间戳）
  - 交换机B收到了交换机A发送过来的数据包，若无法匹配对应流表项，从而packet_in到控制器。
  - 控制器接收到这个数据包之后，计算时间差T1：约等于数据包从控制器到交换机A + 交换机A到交换机B + 交换机B到控制器的时延。
  - 控制器向交换机B发送一个类似的报文。然后控制器从交换机A收到Packet_in报文，记录下时间差T2。
  - 控制器向交换机A和交换机B分别发送带有时间戳的Echo request(ping),测得控制器到交换机A,B的RTT分别为Ta，Tb。

例4：测量网络时延(续)

- 计算
  - 交换机A到交换机B的RTT ： T1+T2-Ta-Tb
  - 假设往返时间一样，则交换机A到交换机B的链路时延为（T1+T2-Ta-Tb）/2

探索：OpenFlow 应用

1. Dynamic access control
2. Seamless mobility/migration
3. Server load balancing
4. Network virtualization
5. Using multiple wireless access points
6. Energy-efficient networking
7. Adaptive traffic monitoring
8. Denial-of-Service attack detection

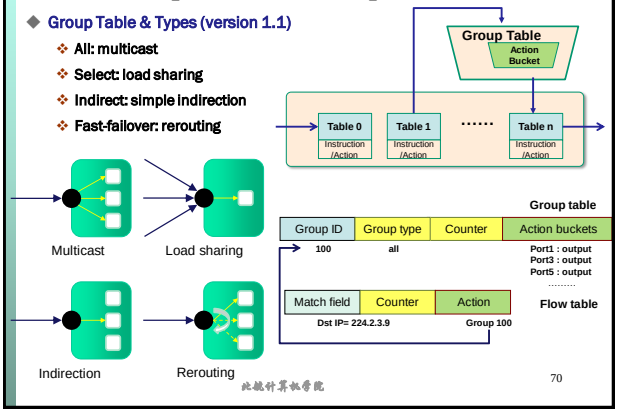
相关教程：
- **http://www.openflow.org/videos/**
- **http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial**
- **http://mininet.org/**

OpenFlow Group Table和Meter Table

应用场景（选）

OpenFlow Group Table

- Group Table & Types (version 1.1)
  - All: multicast
  - Select: load sharing
  - Indirect: simple indirection
  - Fast-failover: rerouting



OpenFlow Group Table

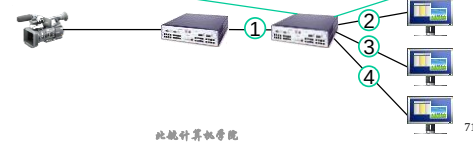
- Multicast
  - Type=all

Group Table

Group ID	Group Type	Counter	Action Buckets
100	All	999	Port2, Port3, Port4

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 6
Port 1	*	*	0800	*	224…	224…	4	4566	6633	Group 100



OpenFlow Group Table

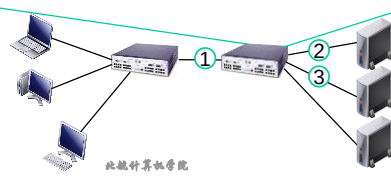
- Load Balancing
  - Type=select

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Select	999	Port2, Port3

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 1
Port 1	*	*	0800	*	1.2.3 …	*	4	*	80	Group 100



OpenFlow Group Table

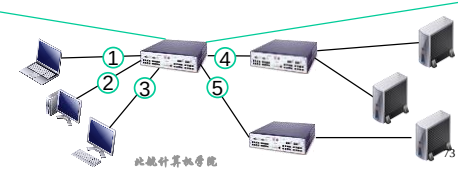
- Indirection
  - Type=indirect

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Indirect	777	Port 5

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	00:FF …	*	0800	*	1.2.2 …	11.1…	*	*	*	Group 100
*	00:FF...	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



OpenFlow Group Table

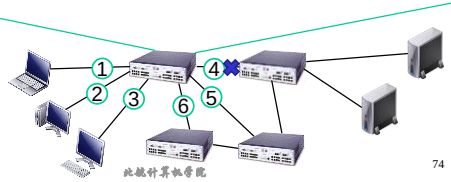
- Fast Failover
  - Type=fast-failover (ff)

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Fast-failover	777	Port4, Port5, Port6

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
Port 1	*	*	*	*	1.2.2	*	*	*	*	Port 7
Port 1	00:FF …	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



OpenFlow Meter Table

- Meter Table (ver 1.3)
  - Counts packet rate of a matched flow
  - QoS control → Rate-limit, DiffServ …

Meter Table

Meter ID	Band Type	Rate	Counter	Argument
100	Drop (remark DSCP)	1000 kbps	1000	xxx

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Inst. Meter	Action
Port 1	*	*	*	1.2.2	*	*	*	*	N/A	Port 7
Port 1	00:FF …	*	0800	1.2.3 …	11.1…	*	*	*	Meter 100	Port 2

SDN产品

SDN交换机

- 软件SDN交换机
  - 软件实现的SDN交换机通常与虚拟化Hypervisor整合，支持云计算场景中的多租户灵活组网等业务
- 硬件SDN设备
  - 支持基于硬件设备的组网，满足SDN网络与传统网络的混合组网需求。

第一代软件SDN交换机

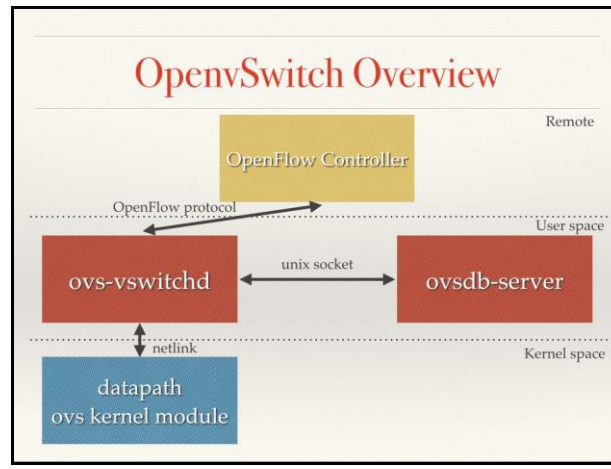
- OpenvSwitch（OVS）
  - Nicira Networks开发，遵循Apache 2.0开源代码版权协议(http://openvswitch.org)
  - 方便管理和配置虚拟网络
  - 可用于生产环境，支持跨物理服务器分布式管理、扩展编程、大规模网络自动化和标准化接口。
- Indigo
  - Big Switch开发，托管在Floodlight组织下的OpenFlow agent开源实现，（https://github.com/floodlight/indigo）
  - IO 复用与定时器管理框架, OpenFlow 连接管理, OpenFlow 状态管理
  - 配置模块: 提供了平台无关的配置接口。

开源软件： OpenVswitch

- 运行模式
  - Can Run as a stand alone hypervisor switch or as a distributed switch across multiple physical servers.
- 支持多种虚拟机平台
  - Default switch in XenServer 6.0, Xen Cloud Platform and supports Proxmox VE, VirtualBox, Xen KVM.
- 支持多种云计算平台
  - Integrated into many cloud management systems including OpenStack, openQRM, OpenNebula, and oVirt.
- 支持多种操作系统
  - Distributed with Ubuntu, Debian, Fedora Linux. Also FreeBSD.

OpenVswitch 组件

- ovsdb-sever: OVS的数据库服务器
  - 用来存储虚拟交换机的配置信息。它与manager和ovs-vswitchd交换信息使用了OVSDB(JSON-RPC)的方式。
- ovs-vswitchd: OVS的核心部件
  - 它和上层controller通信遵从openflow协议，它与ovsdb-server通信使用OVSDB协议，它和内核模块通过netlink通信
  - 它支持多个独立的datapath（网桥），它通过更改flow table实现了绑定、VLAN等功能。
- ovs kernel module: OVS的内核模块
  - 处理包交换和隧道，缓存flow，如果在内核的缓存中找到转发规则则转发，否则发向用户空间去处理。



硬件SDN设备

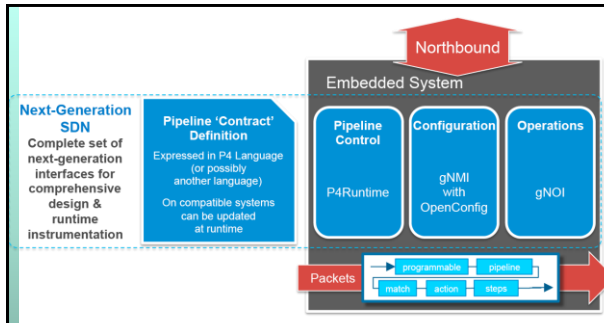
- 绝大多数硬件交换芯片是为二层和三层交换设计的，而OpenFlow的转发模型却是wildcard match + action，导致在很长的一段时间以内人们只能把OpenFlow消息转化为TCAM表项
  - 流表中各个表项的长度及其中包含的匹配域可自定义，而非固定的格式，不再适合采用预先定义好的硬件电路进行流表的实现。
  - 采用TCAM(Ternary Content Addressable Memory，三态内容寻址存储器)技术完成相关流表信息的存储和查询。
  - 容量限制：支持10^3数量级的TCAM表项

瓶颈问题

- 方案1
  - 重新设计硬件交换芯片，让硬件交换芯片支持多级流表，每一级流表都支持match + action
- 方案2
  - 利用现有硬件交换芯片优化OpenFlow协议中的match和action，尽量把match+action在二层和三层的流表中实现。
  - TCAM流表处理复杂ACL

ONF：下一代SDN架构Stratum

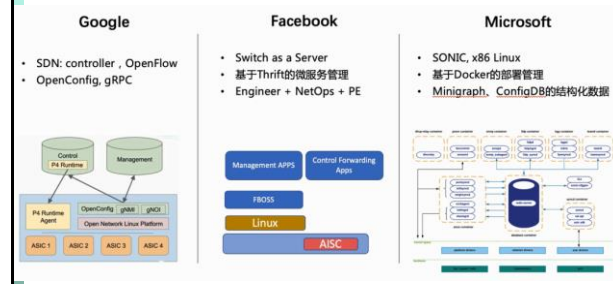
- 开放网络基金会（ONF）：2019年开源Stratum项目
  - 面向运营商的下一代SDN架构，Google贡献首个版本的代码
  - Stratum是一种面向SDND 独立于芯片的交换机操作系统
  - 可以在各种交换机硅片和各种白盒交换机平台上运行
  - 采用Apache 2.0开源许可证
- 数据平面可编程
- 白盒供应商、芯片制造商和运营商参与
  - 英特尔/Barefoot，NTT Communications、中国联通、博通、Edgecore Networks、Delta Networks、英业达、Stordis和PLVision
- 云服务提供商：Google Tencent腾讯



https://opennetworking.org/stratum/

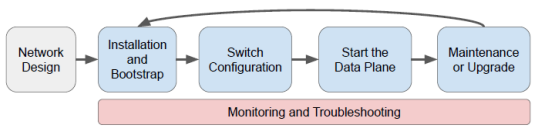
白盒交换机

- Google通过自研交换机构建了SDN网络，用controller来集中管控。在开源上有大家熟悉的OpenConfig、gRPC等。
- Facebook把自研交换机设计成了类似于服务器，交换机和服务器采用同一套运维管控体系，全面采用了微服务化的运维方式。
- 微软采用了SONiC，是标准的Linux，基于Docker的部署管理，采用了Minigraph、ConfigDB的结构化数据来进行管控运维。



白盒交换机的生命期

- Design
- Installation & Bootstrap
- Switch Configuration
- Start the Data Plane
- Monitoring & Telemetry
- Reboot
- Upgrade



SDN研究挑战

- Heterogeneous Switches
- Controller Delay and Overhead
- Testing and Debugging
- Programming Abstractions
- Distributed Controller
- Security…

网络功能虚拟化NFV

From https://opennetworking.org

第一代SDN的问题

- 可编程网络接口不一致
  - OpenFlow pipeline：不同厂商实现方法不同
  - 配置和管理模型不同
  - 不同协议实现需要控制平面支持
- 实际：控制平面独立于特定硬件
  - 不同控制平面有不同的抽象：new abstractions are either least common denominator (e.g. SAI) or underspecified (e.g. FlowObjectives)
  - 与设备绑定：Other control planes have exploited specific APIs are essentially "locked in" to specific vendors

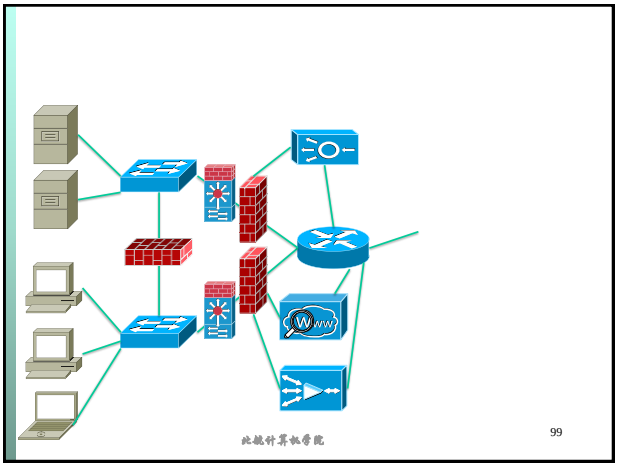
网络功能虚拟化NFV

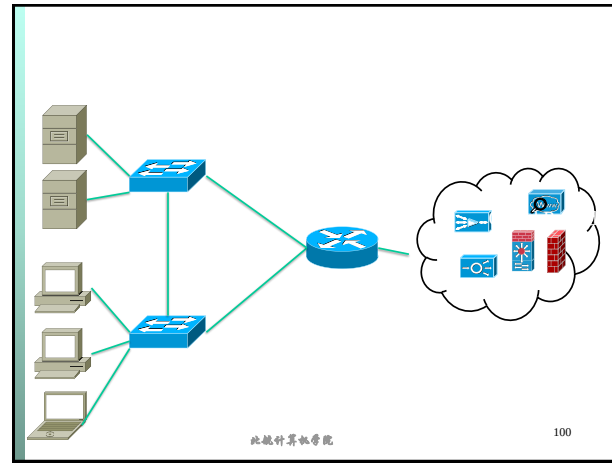
- Network Functions Virtualization（NFV）
  - 由欧洲电信标准化协会（ETSI）组织于2012年10月提出
  - 通过IT虚拟化技术，利用标准化的通用IT设备来实现各种网络设备功能。
- 目标
  - 实现硬件资源与软件功能的解耦，通过标准的x86服务器、存储和交换设备，来取代通信网中私有专用的网元。

Network Function Virtualization: turn these middleboxes into software-based virtualized entities.

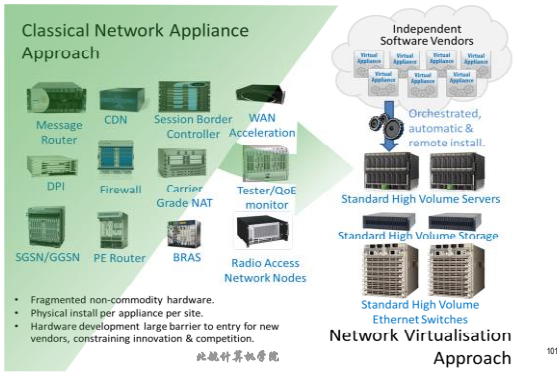
NFV标准化进程

- 2012年10月，由AT&T、BT、Deutsche Telekom、Orange、Telefonica等7家运营商在欧洲电信标准化协会（ETSI）发起成立了NFV行业规范工作组（NFV ISG）
- 2013年10月，ISG发布了第2版NFV白皮书，有25家运营商加入NFV阵营
- 2014年11月，ISG发布了第3版NFV白皮书
  - 4个文档，包括NFV架构、用户案例、虚拟化要求以及名词术语
- 2014世界移动通信大会（MWC）上OpenNFV宣布成立
  - 建立一套面向运营商的网络功能虚拟化解决方案，提供更加快速、简单、低价的服务。
- 中国
  - 通信标准化协会（CCSA）主导SDN/NFV标准化工作
  - 2017年，AT&T OpenECOMP项目与中国移动OPEN-O项目合并后形成的ONAP





Network Functions Virtualisation



虚拟化网络功能VNF

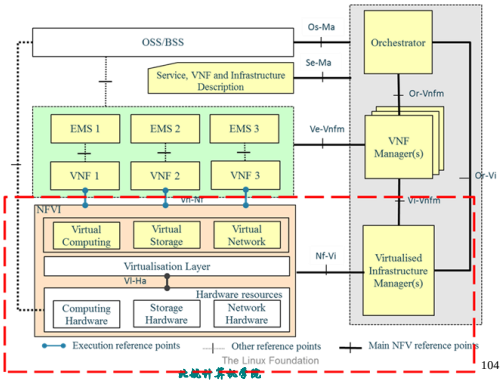
- **网络功能（Network Function, NF）**
  - 传统网络基础设施中的功能模块，它具有固定的内部功能以及良好定义的外部接口。例如，家庭网络中的网关、Firewall，IDS、IPS以及用于提升企业网络性能的代理（Proxy）、缓存（Cache）和WAN Optimization等。

http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf 2014,12

- **虚拟化网络功能（Virtual Network Function，VNF）**

  Software implementation of a network function capable of running over NFV infrastructure
  - 能部署在虚拟资源上的各类软件NF
  - 可由相互独立的软件开发商根据NFV标准进行开发

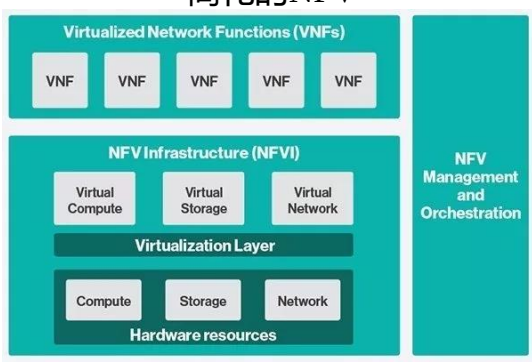
ESTI NFV基础设施结构



ESTI NFV基础设施结构

- **硬件资源**
  - 主要包括由计算硬件设备组成的计算资源，存储设备构成的存储资源以及由节点和连接链路组成的网络资源。
  - 硬件资源通过虚拟化层（例如，VM，虚拟机管理器等）向VNF提供计算处理能力，存储能力以及网络连接性。
- **虚拟化层**
  - 主要负责抽象硬件资源，并将VNF和底层硬件资源解耦。
  - VNF通过使用经过虚拟化层抽象和逻辑切分的物理资源，可以运行在逻辑独立的物理硬件资源之上。
- **虚拟化资源**
  - 是对计算资源、网络资源和存储资源的抽象。
  - 与硬件资源相对应，包括虚拟化计算资源，虚拟化存储资源和虚拟化网络资源。

简化的NFV



NFV 的挑战

- **云计算环境实现NFV**
  - Leverage and adapt cloud technologies to implement NFV
- **灵活配置**
  - using general purpose infrastructure to perform customized tasks.
- **自动管理**
  - Realize the function, but not the reduced management. Manually intensive management
- **可扩展性：Rapid growth of IP end points**
- **移动性：Network end point mobility**
- **弹性部署**
  - Elasticity: VNFs are created, adjusted, and destroyed.
- **支持多租户：隔离Multi-tenancy**

NFV使网络更复杂

- 网络状态可视化
  - 传统网络的状态监控：网络性能监测(NPM)，流量镜像(SPAN)等功能
  - 每个VNF都负责服务链中的某个专用功能，若不同VNF之间的流量无法监测，难以定位和隔离问题
  - 将NFV流量回传到物理网络或代理会导致网络流量增加，浪费网络带宽并增加延迟。
- 安全问题

SDN+NFV

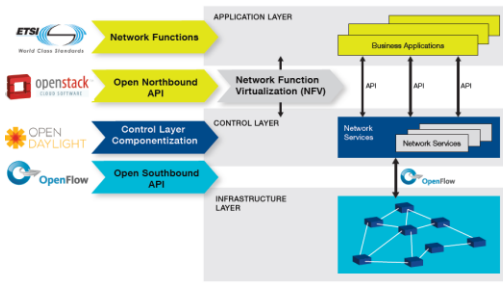
Network Function Virtualization (NFV) v1

- 迁移：Migrate specialized networking hardware (e.g. firewall, load balancer) to commodity servers
- 虚拟化：Virtualized network functions (VNFs) are packaged and distributed as VMs or containers, which are easier to deploy
- 存在问题
  - CPU不一定适合所有网络功能：延迟和抖动；包处理开销和能耗等
  - NFV数据平面拓扑结构低效
    - Additional switching hops required to implement sequences of VNFs (service chains), especially when placement algorithms are not optimize

NFV vs. SDN

- 虚拟化需求
  - 在通用硬件平台上，通过虚拟化实现软件定义的网络服务
  - 按需部署网络服务
- 互补
  - SDN 支持编排orchestration, 路由routing
  - NFV 作为支持SDN运行 的“子层“substrate”

SDN and NFV map



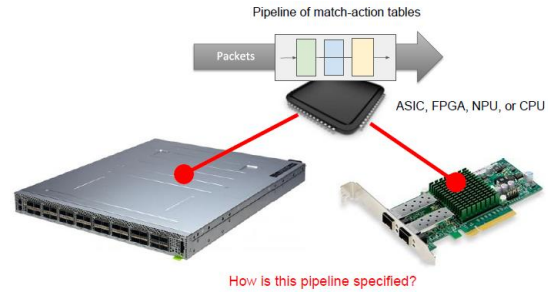
存在问题

- SDN (fabric) and NFV (overlay) are managed separately存在的问题
  - 增加运维成本
  - 跨栈优化困难
  - 缺乏可视化故障定位和端到端优化
  - 资源池分离

下一代SDN

- ONF提出下一代SDN
- 新技术开发
  - Hardware: Programmable ASICs, FPGAs, Smart NICs
  - Software: P4
- 云化部署和管理
  - Zero touch operations
  - Containerization
- 开源

分组转发流水线



P4

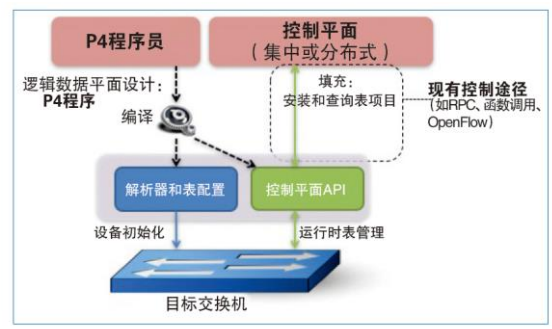
- 2014年，Barefoot与英特尔、谷歌、微软，以及斯坦福大学和普林斯顿大学联合发表了一篇名为《P4：Programming Protocol-Independent Packet Processors》的论文。
- P4是一种对网络数据包处理的领域编程语言（Domain Specific Language）
  - 目标无关性：P4语言不受制于具体设备，所有可编程芯片都可以使用P4编程。
  - 协议无关性：P4可以表达任何转发行为。
- 斯坦福大学教授Nick McKeown
  - 发起成立了开放网络基金会（ONF），以及负责制定P4标准的P4.org
  - 在Barefoot公司担任联合创始人和首席科学家

P4（续）

- **p4.org社区包括网络、云系统和学术机构领域的各种公司**
- **针对网络交换机的数据平面编程。**
- **2019年4月9日，开放网络基金会（ONF）宣布已完成与P4.org的合并，并将主持所有P4活动和工作组主要功能**
- **工业界**
  - **研制了一系列高性能的可编程硬件，其中主要包含Barefoot Tofino，Cavium Xpliant等**
  - **Barefoot Networks公司：开发基于P4的网络芯片Tofino和软件开发套件Capilano，最高可以达到6.5Tbit/s 的线速数据分组转发速率，性能远超传统交换机。**

P4的抽象模型

- **V1Model由5个模块组成**
  - **Parser: 解析器， 解析并且提取数据包头的各个字段。**
  - **Ingress： Ingress处理，定义Ingress流水线。**
  - **TM： Traffic manager，用于流量控制（一些队列相关的metadata在此更新）。**
  - **Egress： 定义Egress流水线。**
  - **Deparser：用于重组数据包，因为数据包在处理过程中经历了分解和处理。所以最后转发的时候需要进行重组**



PISA（Protocol Independent Switch Architecture）

- ◆PISA：通用的、协议无关的、高速、可编程的交换机芯片架构
  - ❖主要数据通路是由大量“匹配-动作”单元以流水线的方式组合而成
  - ❖由通用的逻辑单元和流水线组成，因此与具体协议无关，并且可以通过编程实现各种标准或自定义的网络包处理规则，而无需进行架构修改

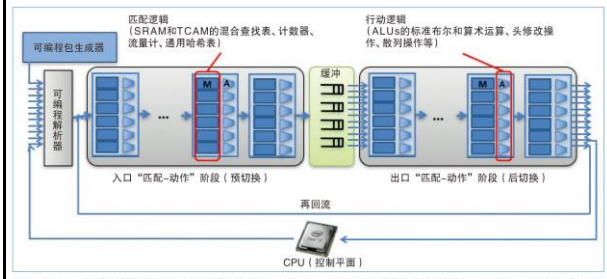


图2 协议无关的交换机架构。到达PISA交换机的数据包先由可编程的解析器解析，再通过入口侧一系列的“匹配-动作”阶段，然后经由队列系统交换，由出口“匹配-动作”阶段再次处理，最后重新组装发送到输出端口

Internet上的Middlebox

（补充）

什么是Middlebox?

- ◆Middlebox的定义
  - ❖Also called a “network appliance” or a “network function.”
  - ❖也称为：中间盒，中间设备，网络功能

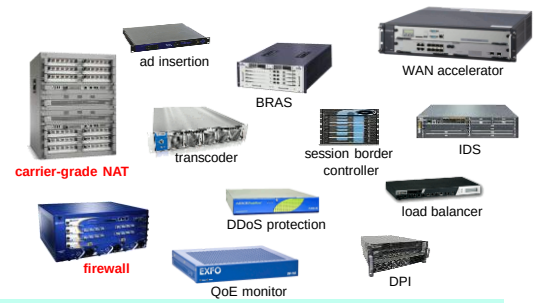
“A middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host.”

— B. Carpenter. RFC 3234. Middleboxes: Taxonomy and Issues.

什么是Middlebox？ (续)

- Middlebox的功能
  - 主要考虑：security，performance
  - 位于端到端的通信主机之间
  - 对端到端状况不可知
- 安全功能
  - 防火墙，入侵检测系统IDS，入侵防御系统IPS，VPN等
- 性能优化
  - Proxy/Cache， WAN Optimizers，Protocol Accelerators
- 其他功能
  - NAT, Accounting，protocol converters（6to4/4to6）

A Middlebox World：专用硬件设备



Middleboxes: hardware-based network appliances. Now a fundamental part of Today's operational networks.

针对Middleboxes的两种观点

- 反方
  - Violation of layering
  - Cause confusion in reasoning about the network
  - Responsible for many subtle bugs
- 正方：实际应用的需求
  - Solving real and pressing problems
  - Needs that are not likely to go away

NAT

Network Address Translation

NAT的历史

- IP 地址空间耗尽
  - Clear in early 90s that 2^{32} addresses not enough
  - Work began on a successor to IPv4
- 现状：
  - Share addresses among numerous devices
  - ... without requiring changes to existing hosts
- 缓解问题的权宜之计
  - Intended as a short-term remedy
  - Now, NAT are very widely deployed
- RFC1631及相关协议：将RFC1918定义的私有IP地址映射到公有IP地址
- NAT: RFC3022

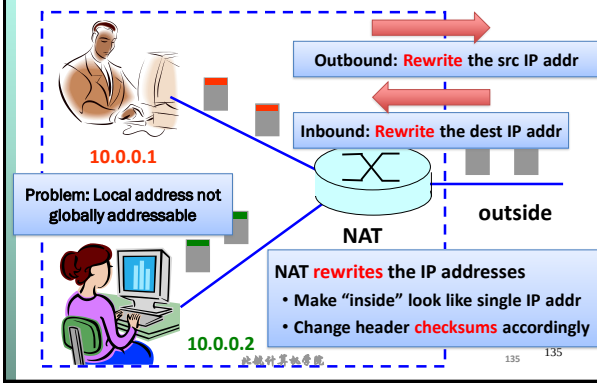
私有IP地址

- RFC1918规定的保留地址
  - 10.0.0.0～10.255.255.255/8
  - 172.16.0.0～172.31.255.255/12
  - 192.168.0.0～192.168.255.255/16

改变寻址模型

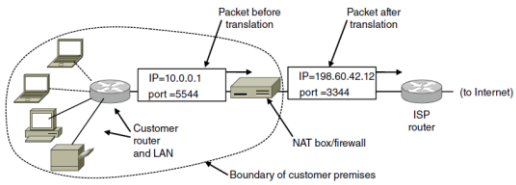
- 原始IP模型：端到端
  - Every host has a unique IP address
- 意义
  - Any host can find any other host
  - Any host can communicate with any other host
  - Any host can act as a server
    - Just need to know host ID and port number （主机+端口号）
- 没有认证机制和安全保障
  - Packet traffic observable by routers and by LAN-connected hosts
  - Possible to forge packets
    - Use invalid source address

NAT (Network Address Translation)



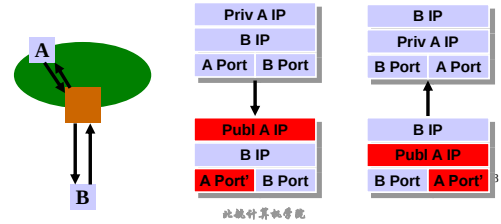
NAT 的基本方法

- **基本方法：端口复用型**
  - NAT (Network Address Translation)将一个（或多个）外部IP地址映射到多个内部IP地址
    - 使用 TCP/UDP端口（port）建立连接
    - 违反分层原则（家庭网络中常用的方法）



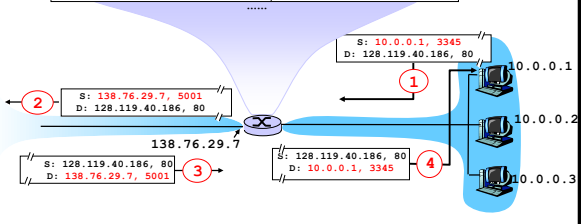
NAT 的地址映射

- 映射方法：NAT maps (private source IP, source port) onto (public source IP, unique source port)
  - reverse mapping on the way back
  - destination host does not know that this process is happening
- 实现：NAT functionality fits well with firewalls



例子

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345



支持NAT的防火墙的特点

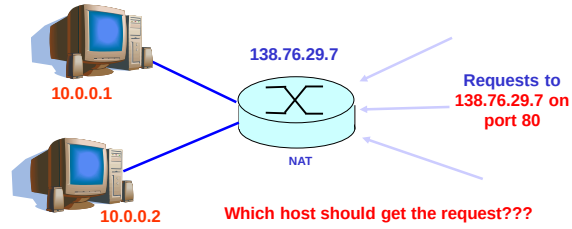
- **优点**
  - 对外部网络隐藏IP地址
    - Easy to change ISP: only NAT box needs to have IP address
    - Fewer registered IP addresses required
  - 防止远程攻击
    - Does not expose internal structure to outside world
    - Can control what packets come in and out of system
    - Can reliably determine whether packet from inside or outside
- **缺点**
  - 违反“端-端”原则
    - Contrary to the "open addressing" scheme envisioned for IP addressing
  - 难以支持 peer-to-peer应用

NAT的实现

- 家庭网络
  - Integrates router, DHCP server, NAT, etc.
  - Use single IP address from the service provider
  - ... and have a bunch of hosts hiding behind it
- 园区网：Campus or corporate network
  - NAT at the connection to the Internet
  - Share a collection of public IP addresses
  - Avoid complexity of renumbering end hosts and local routers when changing service providers

在 NATs后面运行的服务器

- Port #s（端口号） are meant to identify *sockets*
  - Yet, NAT uses them to identify *end hosts*
  - Makes it hard to run a server behind a NAT

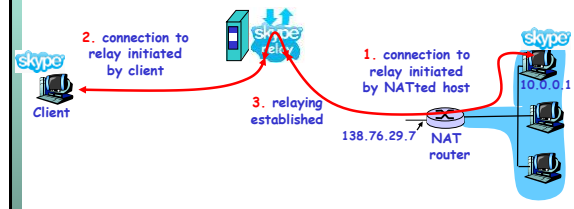


在 NATs后面运行的服务器

- 显式配置 NAT
  - E.g., internal service at <dst 138.76.29.7, dst-port 80>
  - ... mapped to <dst 10.0.0.1, dst-port 80>
- P2P 应用的挑战
  - Especially if *both* peers are behind NAT boxes
- 解决方法？
  - Existing work-arounds (e.g., in Skype)
  - Ongoing work on "NAT traversal" techniques

穿透 NAT （Traversal） ？

- 中继 relaying (used in Skype)
  - NATed server establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



NAT端口映射方式

- **NAT按照端口映射方式分类，可以分为：**
  - 全锥形Full Cone NAT
  - 限制锥形Restricted Cone NAT
  - 端口限制锥形Restricted Port Cone NAT
  - 对称型Symmetric NAT
- **限制的严格程度和对局域网内主机的保护由松到紧依次为：**
  - Full Cone＞Restricted Cone＞Restricted Port Cone＞Symmetric NAT

RFC3489的相关定义

- 全锥形Full Cone NAT
  - 当内网主机第一次向外发送UDP数据包时，NAT会为之分配一个固定的公网{IP:端口}
  - NAT维护一个映射表：内网主机的内网{IP:端口}与公网{IP:端口}是一一对应关系
  - 一旦这个映射关系建立起来（内部主机向某一外部主机发送一次数据即可），任何外部主机就可以直接向NAT内的这台主机发起UDP通信了，此时NAT透明化了
- 最容易联通的NAT

RFC3489的相关定义-续

- 限制锥形Restricted Cone NAT
  - NAT维护了一个内网{IP:端口}到公网{IP:端口}的映射
  - 还维护了一个{外部主机IP, 公网{IP:端口}}到内网{IP:端口}的映射。
  - 因此，要想外部主机能够主动向该内部主机发起通信，必须先由该内部主机向这个外部发起一次通信。

RFC3489的相关定义-续

- 端口限制锥形Port Restricted Cone NAT
  - NAT维护了一个从内网{IP:端口}到公网{IP:端口}的映射，
  - 还维护了一个从{外部主机{IP:端口}, 公网{IP:端口}}到内网{IP:端口}的映射。

RFC3489的相关定义-续

- 对称型 Symmetric NAT
  - 当内网主机创建一个UDP socket并通过它第一次向外部主机1发送UDP数据包时，NAT为其分配一个公网{IP1:端口1}，
  - 当内网主机通过这个socket向外部主机2发送UDP数据包时，NAT为其分配一个公网{IP2:端口2}
  - 公网{IP1:端口1}和公网{IP2:端口2}不会完全相同
    - 或者IP不同，或者端口不同
  - 外部主机只能在接收到内网主机发来的数据时，才能向内网主机回送数据

NAT穿透技术

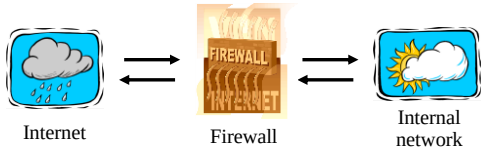
- UPnP协议实现穿透
  - 动态创建端口映射规则，前提则需要连接客户端和NAT设备本身支持UPNP协议
- ALGs应用层网关（Application Layer Gateways）
  - ALG识别了相应报文之后对负载信息进行解析，然后进行地址转换，重新计算校验和。
    - ALG可以处理的协议：DNS，FTP，H323，SIP，HTTP，ILS，MSN/QQ，NBT，RTSP，PPTP，TFTP、GRE等
  - 目前linux防火墙可以支持常见的大部分协议的ALG。
- STUN技术（Simple Traversal of User Datagram Protocol Through Network Address Translators ）
  - 借助UDP协议进行UDP打洞，相应RFC可以参考RFC 3489和RFC 5389，其只能在非Symmetrict NAT情况下成功穿透。
- TURN（Traversal Using Relays around NAT:Relay Extensions to Session Traversal Utilities for NAT），即使用中继穿透NAT:STUN的扩展，参考RFC5766

防火墙

防火墙 Firewalls

- 安全问题
  - Lots of vulnerabilities on hosts in network
  - Users don't keep systems up to date
    - 补丁**Lots of patches**
    - 零日漏洞 **Zero-day exploits**
- 解决方法
  - Limit access to the network
  - Put firewalls across the perimeter of the network

防火墙 Firewalls



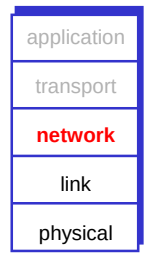
- 网络的访问控制机制
- 防火墙确定过滤或转发出入内网的报文

有关术语

- 防火墙类型
  - 包过滤防火墙Packet filter
    - 工作在网络层
  - 基于状态检测的包过滤防火墙 Stateful packet filter
    - 工作在传输层
  - 应用层代理 Application Level proxy
    - 应用层
  - 链路级网关 Circuit Level Gateway
    - 工作在传输层

包过滤防火墙

- 工作在网络层
- 过滤条件
  - 源IP地址 Source IP address
  - 目的IP地址 Destination IP address
  - 源端口 Source Port
  - 目的端口 Destination Port
  - 标志位 Flag bits (SYN, ACK, etc.)
  - 流出Egress/进入 ingress



包过滤防火墙

- 配置访问控制表
  - Configured via Access Control Lists (ACLs)

Action	Source IP	Dest IP	Source Port	Dest Port	Protocol	Flag Bits
Allow	Inside	Outside	Any	80	HTTP	Any
Allow	Outside	Inside	80	> 1023	HTTP	ACK
Deny	All	All	All	All	All	All

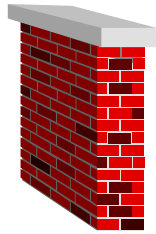
- Q: 作用？
- A: 只允许内网访问web服务器，限定来自web响应的进入包，其他流量被禁止

包过滤规则

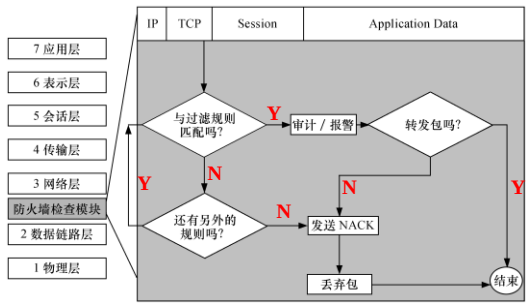
- 根据流经该设备的数据包地址信息决定是否允许该数据包通过
- 不检查数据区，只检查地址区
- 判断依据有(只考虑IP包)
  - 数据包协议类型TCP、 UDP、 ICMP 、IGMP等
  - 源、目的IP地址
  - 源、目的端口FTP、 HTTP 、DNS等
  - IP选项源路由、记录路由等
  - TCP选项SYN 、ACK、 FIN 、RST等
  - 其它协议选项ICMP、 ECHO、 ICMP、 ECHO 、REPLY等
  - 数据包流向in或out
  - 数据包流经网络接口eth0 eth1

安全缺省策略

- 两种基本策略，或缺省策略
  - 没有被拒绝(deny)的流量都可以通过：黑名单
    - 管理员必须针对每一种新出现的攻击，制定新的规则
  - 没有被允许(allow)的流量都要拒绝：白名单
    - 比较保守
    - 根据需要，逐渐开放

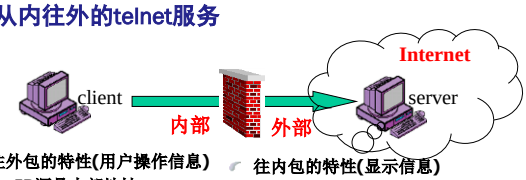


包过滤防火墙模型



例1：包过滤防火墙的设置(1)

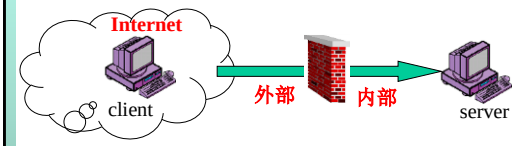
- 从内往外的telnet服务



- 往外包的特性(用户操作信息)
  - IP源是内部地址
  - 目标地址为server
  - TCP协议，目标端口23
  - 源端口>1023
  - 连接的第一个包ACK=0，其他包ACK=1
- 往内包的特性(显示信息)
  - IP源是server
  - 目标地址为内部地址
  - TCP协议，源端口23
  - 目标端口>1023
  - 所有往内的包都是ACK=1

例1：包过滤防火墙的设置(2)

◆从外往内的telnet服务



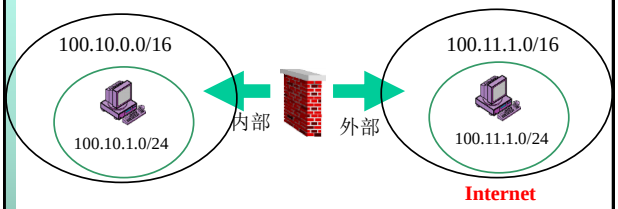
- 往内包的特性(用户操作信息)
  - IP源是外部地址
  - 目标地址为本地server
  - TCP协议，目标端口23
  - 源端口>1023
  - 连接的第一个包ACK=0，其他包ACK=1
- 往外包的特性(显示信息)
  - IP源是本地server
  - 目标地址为外部地址
  - TCP协议，源端口23
  - 目标端口>1023
  - 所有往内的包都是ACK=1

例1：针对telnet服务的防火墙规则

服务方向	包方向	源地址	目标地址	包类型	源端口	目标端口	ACK
内向外	外	内部	外部	TCP	>1023	23	*
内向外	内	外部	内部	TCP	23	>1023	1
外向内	外	外部	内部	TCP	>1023	23	*
外向内	内	内部	外部	TCP	23	>1023	1

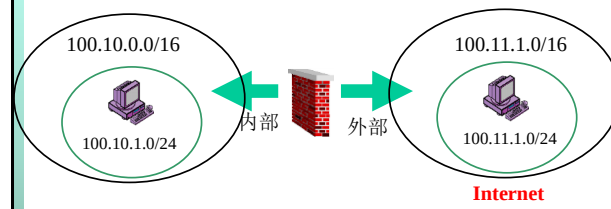
*: 第一个ACK=0, 其他=1

例2：包过滤防火墙的设置



例：设网络100.10.0.0/16不愿意其它Internet主机访问其站点；但它的一个子网100.10.1.0/24和Internet上一个大学的实验室100.11.1.0/24有合作项目，因此允许该大学的实验室访问该子网，而不允许该大学的其他网段访问。

例2：包过滤防火墙的设置



	director	type	Src	port	dest	port	action
1	in	*	100.11.1.0/24	*	100.10.1.0/24	*	allow
2	out	*	100.10.1.0/24	*	100.11.1.0/24	*	allow
3	both	*	*	*	*	*	deny

包过滤防火墙

- 在网络层上进行监测
  - 并没有考虑连接状态信息
- 通常在路由器上实现
  - 实际上是一种网络的访问控制机制
- 优点：
  - 实现简单
  - 对用户透明
  - 速度快，效率高

168

包过滤防火墙的缺点

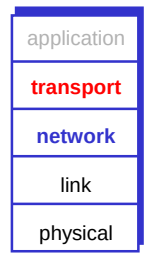
- 容易遭受IP地址欺骗
- 提供较低水平的安全性
- 缺少状态感知
  - 无法看见TCP连接
  - 无法分析应用数据（深度包检测DPI）
- 创建规则比较困难

169

基于状态检测的包过滤防火墙

- Stateful Packet Filter
  - 增加状态 state
- 工作在传输层
- *可以记忆TCP连接，标志位等*
- 甚至可以记忆UDP包（如DNS请求）

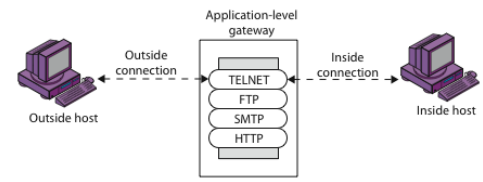
“动态检测防火墙”



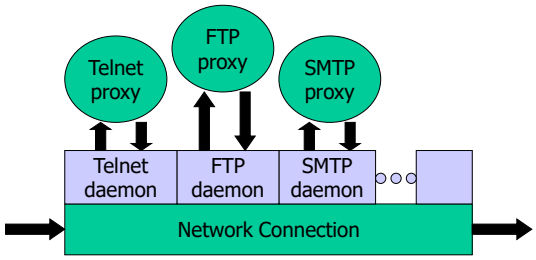
170

应用层代理

- Application Level Gateway (or Proxy)
  - 针对特定应用层协议：如http, ftp, smtp 等



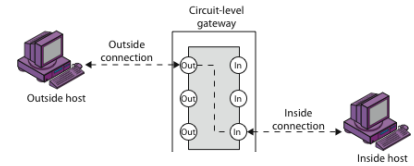
应用层代理体系结构



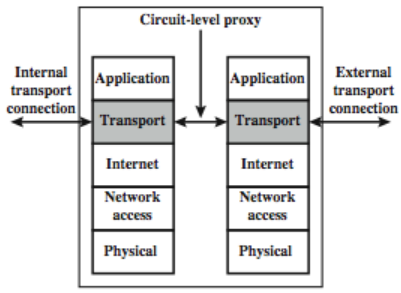
172

链路级网关

- 链路级网关 Circuit Level Gateway
  - 中继（relays）TCP连接，限制允许建立的连接
  - 一旦连接建立，不再检查中继流量的内容
  - 通常用于可信任的内部用户向外发起连接

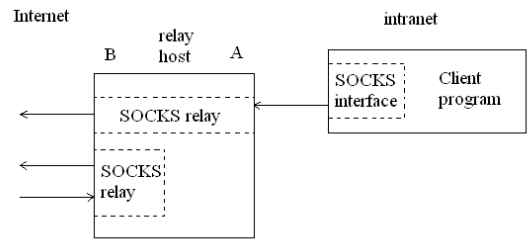


链路级网关：传输层



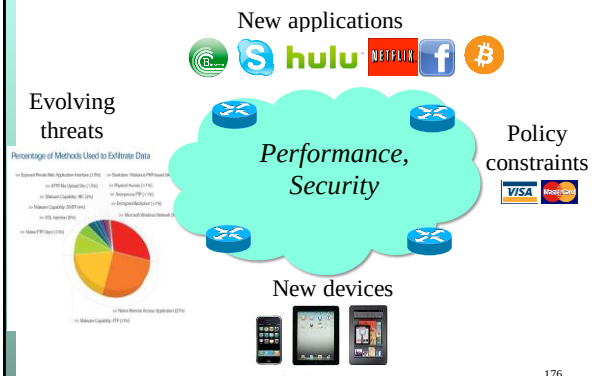
174

例：SOCKS connection



175

网络演化的需求：性能与安全



传统路由器和Middlebox比较

传统网络转发设备：
- 交换机
- 路由器

挑战：
- 安全和隐私
- 差异化服务
- 智能化

网络转发与中间设备特点对比

	转发设备	中间设备
处理任务	送达	安全、统计、优化
逻辑对象	网包	网流
物理对象	L2～L4 包头	L3～L7 包头＋载荷
决策基础	拓扑	资源、策略
管控方式	局域自治	广域集中
产品形态	集线器 交换机 路由器	中间设备 （防火墙、IDS/IPS 防病毒/防垃圾）

传统路由器和Middlebox比较（续）

- Middlebox
  - 有状态stateful
    - remember fine-grained data that is updated as frequently as every packet or every connection.
  - 执行复杂操作
  - 用户：企业用户，运营商ISP

存在问题

- 应用现状
  - 拥有超过10万台主机的大型企业网络中平均部署1946台不同功能的网络功能设备以及2850台三层路由器
  - 拥有不超过1000台主机的小型网络平均部署10台网络功能设备，7台三层路由器
  - 网络功能设备在企业网中的平均数量达到三层路由设备的70%。
- 存在问题：难以管理
  - 部署、管理、更新网络功能设备开销大
  - 管理困难：不同厂商产品，策略配置复杂
  - 网络功能设备失效率高
  - 难以扩展

完成小作业（3）

◆ **专题3 “数据中心网络”**

1.任意选择1篇论文进行阅读

2.每人独立完成论文评论（paper review），评论内容要求：

- **作者主要观点和要解决的问题**
- **研究方法评论（关键技术，优点和局限性）**
- **论文的主要贡献**
- **其他**
- **注意：不是翻译，篇幅不限**

3.作业提交（两个文档）

- **.docx文件**
- **.pptx文件（约 10 页左右，请勿超过15页，课堂讨论用）**