互联网体系结构复习

互联网回顾与基本结构

- 体系结构是什么？
  体系结构是一组抽象的原则，它指导系统在技术层面的设计。体系结构定义了如何将系统分解为不同部分，以及这些部分之间如何交互。
- 网络体系结构研究什么？
  网络体系结构研究这样的设计原则，它们指导了计算机之间通讯的协议和机制的技术设计。一个网络体系结构表示了在许多设计选项中的一套故意的选择，这种选择来自于对需求的理解。网络体系结构是网络的分模块设计。
- 互联网的体系结构？
  - 分层：
    Internet从上到下Application, Transport, Network, Link, Physical。
  - 端到端：
    所谓端到端原则，就是「(所寻求的)功能能被完整而正确地实现，只有在具有通讯系统的终端的应用的知识和帮助的时候」。
    其哲学是，如果应用可以完成它，就不要在更下层完成它。只有在
    (1) 被许多应用使用，且能提升它们的性能
    (2) 不伤害其他应用
    的时候，可以将功能实现在更下层，仅仅作为性能提升。

挑战和技术路线

- 分层 + 抽象
- 互联网设计的1 + 7个目标：
  - 连接已有的网络
  1. 可生存性
  2. 支持多种类的服务
  3. 能用于多样的物理网络
  4. 允许分布式管理
  5. 成本效益好(cost-effective)
  6. 允许主机很容易地附加到网络
  7. Resource accountability
- 面临的问题与路线之争
  - 问题：
    用户增多、规模扩大，产生新兴业务。出现问题：可扩展性、安全性、管控性、服务质量(QoS)保障、移动性、服务分发能力、绿色节能等。
  - 路线之争：
    + 革命型(clean-slate)
      新的网络体系结构，重新设计通信协议，根本上克服传统互联网体系结构的问题。

减缓实际部署的步伐。需要部署实验，但实验结果未必完全符合现实情况。

+ 演进型（evolutionary/incremental）
  现有网络体系结构下「补丁式」修补。新的组网技术（改造网络设备、网络拓扑）；升级已有网络通讯协议；现有网络中加入AI、大数据等技术手段。
  暂时解决问题，但是最终会变得复杂臃肿。

Clean-Slate

- 当前设计原则
  - 分层，解决目标0，3
  - 包交换（packet switching），解决目标5
  - 合作的网络组成的网络，解决目标1，4
  - 智慧的端系统，解决目标1，5
  - 端到端原则
- 什么是clean-slate，以及为什么这样？
  - Clean-slate就是从头开始设计一个系统，以提供更好的抽象和/或性能，同时基于新的核心原则，提供类似的功能。
  - 现在，人们已经不愿意或者无法在当前的体系结构上实验：
    + 旧的原则：当前的互联网设计原则内在地符合当前的互联网体系结构，难以挑战、难以改变。
    + 新的技术：新技术使新能力成为可能，质疑旧的设计原则。
- 设计目标和其所蕴涵
  如同前述。
- 控制平面 vs. 数据平面
  - 控制平面建立端到端连接的路径，内在地就是全局的，需要知道网络拓扑。分布式的routing algorithm在控制平面上，每个路由器存储转发信息，这就是「转发表」。
    挑战：需要在面临网络失败和拓扑改变的情况下计算大规模的路径。这在ISP需要自主权，不想透露其内部routing决策的情况下。
  - 数据平面切实地实现交换机逐个包的行为，内在是局部的。转发机制是数据平面的一部分。
    挑战：数据包来得非常快，必须很快（~10 ns）地处理转发操作。

软件定义网络（SDN）

- 为什么数据平面、控制平面分离？
  为了获得（对数据平面的）全局视野，实现逻辑上中心化的统一的控制平面。这使得
  - 网络更容易管理，避免错误路由配置，流量管理更灵活；
  - 集中化的「编程」更容易实现，集中计算routing table，然后分发；
  - 有利于控制平面的开放实现，non-proprietary。

- 如何分离？
  南向接口是开放的API，比如OpenFlow。SDN数据平面通过这些API和数据平面的交换机交互。数据平面的交换机向控制平面上报其信息，控制平面整合全局的数据计算流标、下发表项。
- OpenFlow
  不足：
  - 控制功能和转发行为定义合并在一个标准中，必须修改整个规范才能支持新的数据路径协议和用例。
  - 增加一个匹配域，就要重新编写控制器和交换机两端的协议栈和交换机的数据包处理。
  - 未来ONF计划：核心标准功能和特定于应用程序、数据路径的元素分离。
- 流表的基本结构与实现
  流标项：<Rule, Action, Stat>
  - Rule匹配header中的任意多位。
  - Action是该项对应的动作，可以转发（到端口），丢弃，发送到控制器，有掩码覆写header，push或pop，按特定位速率转发。
  - Stat是包和字节计数器。
- P4是什么？
  数据平面的一种编程语言，支持对交换机等转发设备的数据处理流程进行软件编程定义。P4 + P4 runtime（一个南向协议）：灵活配置、协议独立、设备独立。P4被编译成特定设备的配置，加上P4Info传给交换机。P4Info还传给控制器，这样控制器和交换机之间通过P4 runtime协议交互。

网络软件化

- NFV（network function virtualization，网络功能虚拟化）是什么
  网络功能用软件实现，通过见效对硬件的依赖，提高网络灵活性。
  - 用「通用服务器+软件」替代专属硬件
  - 通过软件编程来设计、实现、部署、管理、维护网络设备和网络组件
  - 开放：毋须再使用昂贵且复杂的专属软硬件设备
- 为什么NFV
  NFV源自于运营商的需求：降低CAPEX（capital expenditure，资本支出），减少硬件成本投入；在通用平台上实现网络功能。
- NFV优势/缺点
  - 优势：
    + 减小设备投入成本（CAPEX）。利用IT通用设备的规模性，降低成本。
    + 提高设备功能上线的速度，避免运营商-设备商-标准组织的复杂流程。
    + 云化运营，多版本，多租户。单一平台服务多个用户。
    + 生态系统，开放。
    + 提供快速新的业务能力。灵活性：部署灵活，扩展灵活。
    + 个性化定制容易。

        + 减少运营成本：机房空间、人员、监控系统。
    - 缺点：
        + 系统集成能力的挑战。运营管理、业务部署都打破现有流程，需要运营商有强的集成能力。
        + 安全性的考虑，专有设备的安全性通常被认为会更高。
        + 关键性能上的挑战：目前产品仍然有不够成熟，特别是性能。
        + 技术栈的更新：传统的方案相关的运维人员，在面对NFV时，通常需要重新学习。
- NFV和SDN的关系
  NFV注重传统网络功能在通用硬件上的实现，虚拟化平台。SDN注重分离网络的控制和数据平面，实现集中控制。两者的技术独立又相互补充：
    - SDN提供的全局视图与集中控制能力，能够简化NFV对各类VNF实例的管理与操作。
    - SDN的集中控制能力能够有效解决NFV系统与传统运营系统之间的兼容性问题，促进二者的协作。
    - NFV为SDN提供NFVI平台，在上面可以部署各种SDN应用。以SDN控制器为例，将其虚拟化为一种VNF，从而支持灵活的部署和安装。
    - NFV使用标准商用硬件取代专用硬件，为SDN创造了可扩展和弹性的网络环境。

SDN控制平面的扩展性问题

- 纵向扩展（scale up）和横向扩展（scale out）
    - 纵向扩展：向系统中的节点加入更多资源，将服务器处理器容量升级成同一系列之更大者。向单一系统加入更多组件。
      性能受限；单台成本高、控制成本低；管理复杂性低、技术复杂性低；可用性低。
    - 横向扩展：向系统中加入更多节点，在集群中加入更多服务器。通过加入更多系统来扩展。
      理论性能潜力大；单台成本低、分布式控制成本高；管理复杂性高、技术复杂性高；可用性高。
- SDN扩展性问题的原因
    - 大量数据平面设备、大量流、大量应用产生的大量事件。控制平面的处理能力。
    - 控制平面和交换机之间的距离。控制流量的传输时间。
- 提升SDN控制平面扩展性的设计空间
    - 分布式：增加控制器的数目。负载均衡、降低控制流量的传输延迟、提高容错、动态扩展。分布式控制平面可分为：
        + 逻辑集中、物理分布。Onix、ONOS。
        + 完全分布式。
        + 分层控制平面：递归的网络控制，本域可解决则本域解决，否则上传到上一级。Kando、xBar。

    - 卸载（offloading）：把部分控制功能卸载到本地数据平面，减轻控制平面负担。

可编程数据平面

- 数据平面的功能
  逐个包的行为，作用于包上的流算法，如转发、访问控制、包头域映射、流量监控（traffic monitoring）、缓冲和标记、整形（shaping）和调度（scheduling）、深度包检查。
- 数据平面面临的问题
    - 传统的网络数据平面功能深入地嵌入到设备软硬件中，设备生命周期内不可改变。
    - 现代网络部署在不同环境、支持多种用途、面临从尽力而为到性能保障的不同功能要求。
    - 矛盾：既要网络设备对特定任务优化，又要让它们是商用和通用的以降低工程成本。、
    - ⇒ 需要网络设备的可编程性，允许通过编程接口改变设备功能。
- 如何抽象数据平面功能
    - 多种数据平面设备（router、switch、firewall、NAT、etc）抽象为统一的match-action设备
        + match：匹配包头对应字段
        + action：丢包、修改、缓存、etc
    - OpenFlow
        + 数据平面提供match-action
        + list of <priority, predicate, action> tuples
        + 本质上是协议（OpenFlow协议中事先定义的协议字段）相关的，不能真正的软件定义。
    - PISA（protocol independent switch architecture）
        + parser，有限状态机，按预定义格式解包装报文头字段，支持自定义包头格式
        + pipeline，每级流水线包含多个可编程处理单元
        + deparser，将修改后的报文头字写回报文
- 可编程数据平面的优势
    - 给网络包处理带来完整的灵活性
    - 可以有用户定义的、面向控制平面可编程性和SDN的API
    - 对用户：允许网络设备设计者和用户实验新协议、设计独特应用，却不需要依赖专门的包处理ASIC供应商来实现。
    - 对商家：轻易用同一个包处理ASIC实现差异化产品，不需要为不同客户的使用场景设计不同设备。

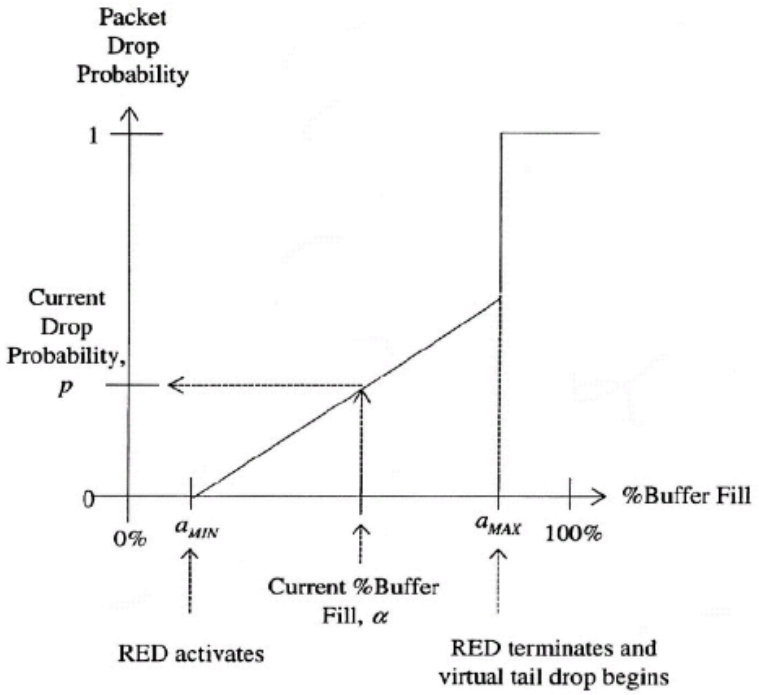
模块化设计

- 为什么需要模块化
  - 优势
    + 相对独立性：对模块单独设计、制造、调试
    + 互换性：模块接口标准化，容易模块间互换
    + 通用型：横系列、纵系列产品间模块通用
    + 消除依赖性：每个模块可以用最合适技术实现
    + 灵活性：模块内更新只要替换对应模块，不需全部更新
    + 易于实现和维护：庞大复杂系统分解为若干相对独立子系统，易于处理
    + 复用模块：更细粒度功能能被更多系统使用
    + 促进标准化：每一模块的功能有精确的说明
    + 鼓励创新：去中心化的决策，第三方可以在模块上创新
  - 缺点
    + 对广泛应用的接口缺乏灵活性
    + 协调各个模块的天然开销，修改接口需多方协作
- 模块化设计的步骤
  - 列举对想设计的理想的系统所有功能性和非功能性要求
  - 用这些要求导出设计含义（约束）
  - 当这些含义、约束被获得，搜索符合这些约束、能实现符合要求的系统的技术
  - 现实中：
    + 自顶向下：先识别主要任务，然后其中的子任务
    + 模块化：把系统分为更小的、能相互独立设计的子系统，它们之间通过接口传递信息来互操作
- 互联网的模块化
  - 分层：互补的关系垂直地堆叠的结构
  - 细腰的network（IP）层：尽力而为的包递送，每个包独立处理，允许丢失、延迟、损坏、乱序。不需负责、survivability好、可运行于几乎任何link层技术

网络服务质量

- 什么是QoS，主要参数指标有哪些
  - 定义：两个视角
    + 端用户：端用户感受到的他从网络提供者接收到的特定服务或应用的质量，比如音频、视频和数据
    + 网络：网络提供上述定义的端用户QoS的能力
  - metrics
    + 主观：MOS（mean option score）
      excellent，good，fair，poor，unsatisfactory
      $$MOS = \frac{N_E \times 5 + N_G \times 4 + N_F \times 3 + N_P \times 2 + N_u \times 1}{N}$$

        + 客观：丢失、延迟、抖动（jitter，数据包延迟变化）、吞吐量
- 延迟和jitter有何区别
    - 延迟（delay）是数据包从发出到接收经过的时间
    - 抖动（jitter）是数据包之间延迟的变化或不一致性
- 延迟主要来源
    - 编码和解码（codec）
        + 源头编码（source coding）
        + 包化（packetization）延迟
        + 信道编码（channel coding）
        + 抖动缓冲区（jitter buffer）延迟
    - 网络（network）
        + 包排队（packet queuing）延迟
        + 传播（propagation）延迟/光速延迟
    - 只有包排队延迟是不固定的：造成延迟变化，即抖动
- IP QoS的基本思路
    - 任务1：区分不同的流量或服务类型，区别对待不同类型的流量
        + 包标记（packet marking）：设置IP头中合适的域为特定的值，以区分IP数据包的类型
        + 包分类（packet classification）：将包按照一个分类规则分组。方法：
            ➔ 多域（multi-field，MF）：基于一个或多个头域分类，此外，其他像接口标识这样的参数也可用于分类
            ➔ 行为聚集（behavioral aggregate，BA）：只基于DiffServ Code Point（DSCP）值分类
    - 任务2：在网络中提供资源保障和服务区分，来区别对待不同种类的流量
        + 流量监控（traffic policing）：检查流入的流量是否遵守事先客户和IP网络服务提供者约定的速率。基于预定速率测量流量，并根据结果标记、重标记数据包。
        + AQM（active queue management）：RED、WRED和ECN，详见下一条（圆点）。
        + 包调度（packet scheduling）：调度队列中的包，使固定数量的输出端口带宽被「平等」且「最优」地分配于流量之间，详见下下条（圆点）。
        + 流量整形（packet shaping）：改变进来流量的速率，从而让出去的流量更平顺。如果进来的流量是爆发式的，它就要被缓存，从而输出流量没那么爆发性。
- ECN和RED如何工作
    - RED（random early discarding）：监测到拥塞开始的迹象，然后随机地从缓冲区丢弃数据包。Packet drop profile：



- Weighted RED：对不同「颜色」的数据包应用不同激进程度的packet drop profile。
- ECN（explicit congestion notification）：
  + 拥塞开始的迹象通过标记TCP和IP头中合适的域，被告知端系统。
  + ECT：ECN-capable transport
  + 源TCP应用设置ECT位为1以告知IP网络中的路由器「该包适用ECN」。
  + 当路由器预计到拥塞，它设置CE位和ECT位为1以告知端系统拥塞的发生。
  + 源和目标TCP主机用保留域的Echo flag和CWR flag，以及代码域SYN和ACK位来为了ECN「握手」。
  + 接收端TCP主机在TCP ACK包中设置echo位为1以告知源主机「网络中有拥塞」。

- 调度算法（FIFO，PQ，FQ）
  - FIFO（first-in first-out）：默认的排队机制，单个缓冲区，先到达者先被发出，完全平等，适合尽力而为网络。
    FIFO不区分流量的种类；TCP在拥塞期间会退避，因而更适合UDP。
  - PQ（priority queuing）：N条队列，优先级1到N。只有1到(j - 1)级队列都没有数据包时，j级队列才会被处理。
    Starvation。
  - FQ（fair queuing）：进入的包被分为N条队列，每条队列被分配1/N的输出端口带宽。Scheduler以round robin的方式访问这些队列（跳过空队列），每来到一个队列就从中传输一个包。
    不能按照不同输入种类的需求分配相应的带宽；实际带宽受单个数据包大小影响。

- WRR（weighted round robin）：双层RR，外层按权重分配带宽，内层FQ。实际带宽仍然受单个数据包大小影响。
- WFQ（weighted fair queuing）：队列之间按照权重分配带宽，每次发出计算出来「虚拟完成时间」最短的包。
  记t时间流i的第j个包的完成时间为$F_i(j,t)$，则：

  $$F_i(j,t) = \max\{F_i(j-1,t), R(t)\} + P_i(j,t)/w_i$$

  其中，
  + $P_i(j,t)$是第j个包的大小
  + w_i是流i的权重
  + $R(t)$是轮数：时间t时候，逐位RR所完成的服务轮数

- Token bucket原理
  - token以恒定速度r放入token bucket
  - token bucket的最大大小为depth d，满了后不能再放入token
  - 每个token允许发出数据包的1字节
  - 没有数据包发送时，token bucket关闭，不再漏出
  - 有数据包在缓冲区时，token被以C的速率取出，数据包相应地被发出
  - 没有token剩余时，数据包必须等待token被放入bucket，之后才能发出

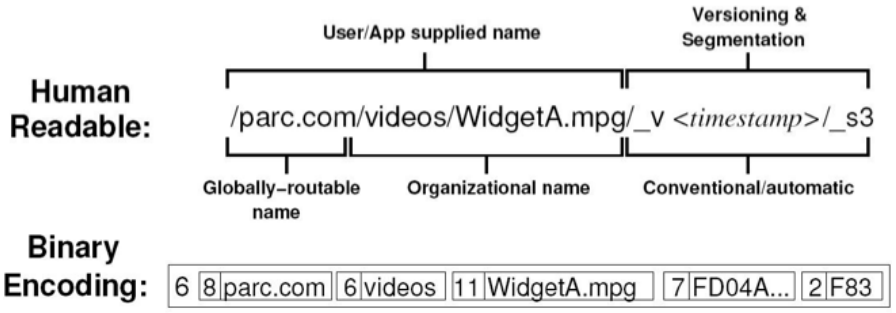
Middlebox，云原生

- 什么是middlebox
  除路由器/交换机意外的其他中间网络设备都叫middlebox，它们在源主机到目标主机的数据报路径上执行不同于普通、标准的IP路由器的功能。
- 为什么要middlebox
  - 原有地址空间不够用（NAT）
  - 完全用端到端实现，连接性能会比较差（Proxy, Cache）
  - 互联网体系结构设计之初并没有考虑安全问题（IPS,FW）
  - 传统架构不提供流之间的公平或者运营管理SLA（流量管理Packet classifiers, markers and schedulers）
  - 传统架构不提供流之间的公平或者运营管理SLA（流量管理Packet classifiers, markers and schedulers）
  - 协议转换 protocol converters (6to4/4to6)
  - etc
- middlebox和e2e的关系
  最初的e2e：底层对网络功能的实现仅作为性能提升。
- 为什么有人反对e2e
  - 可信的需求：原来假定端系统是合作的，现在会有恶意或误操作。
  - 新应用对性能的需求：IP的尽力而为没有QoS保证。
  - 运营商对服务的区分需求。
  - 第三方监管需要。
  - 用户对简单使用的需求。

- middlebox的实现方式
  - 硬件（physical NF，PNF）：专用硬件实现
  - 软件（virtual NF，VNF）：虚拟网络功能，标准商用硬件取代专用硬件
  - 云原生（cloud-native NF，CNF）：单体变为微服务

内容中心的网络

- 互联网演进的两种路线
  - 改良式：可演化、打补丁、增量部署；无法根本解决问题、破坏沙漏，复杂可扩展性问题。
  - 革命式：重新设计、持续创新、全新体系；脱离实际，部署代价高、仿真模型简单，难以反映现实状态。
  - IPv6升级的困难：基础设施升级难度大
    + 渐进部署
    + 纯IPv6的网络部署
- ICN（content/information-centric network）/NDN（named data network）的思想
  - 主要设计思路
    + 用户不再关心由哪一台主机提供服务，而是关心如何更快、更准确和更高效地获取数据。
    + 舍弃基于IP（位置）的通信方式，采用基于内容本身的通信方式，根据内容本身命名和检索网络中的数据。
    + 在转发路由机制中通过对内容数据的名字匹配检索获取信息，从而建立一个分布式网络。
  - 路由的原理
    + 节点存储模块：
      ➔ 内容存储（Content Store, CS）：是否已经有缓存了
      ➔ 待定 Interest 表（Pending Interest Table, PIT）：是否已经有其他人请求过了，那就不用再重复发请求了，只需记录一下，在包回来时给所有请求都分一份就行
      ➔ 转发信息库（Forwarding Information Base, FIB）：查询哪条路径能满足请求，把兴趣包转发下去
    + 当一个Interest包到达，路由器根据Interest中的Content Name，首先查找内容缓存，如果缓存中有被请求的内容，则响应该请求，并丢弃该Interest包
    + 如果内容缓存中没有被 请求的内容，则查找 PIT，如果 PIT 中有该 Content Name条目，则在该Content Name条目中增加face，并丢弃该Interest包
    + 如果PIT中没有该Content Name条目，则查找FIB，如果在FIB中找到，则按照查找到的所有face口转发Interest，并在PIT中记录。如果FIB中也没有该Content Name条目，则丢弃该Interest包
    + 命名：



网络算法设计

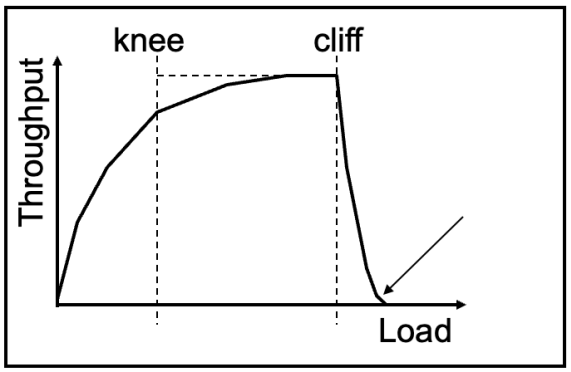
1. 避免频繁事件的明显浪费
2. 避免频繁事件的明显浪费
    a. 预先计算
    b. 延迟计算
    c. 分摊开销
3. 放宽系统要求
    a. Trade Certainty for Time（牺牲确定性换时间）
    b. Trade Accuracy for Time（牺牲精度换时间）
    c. Shift Computation in Space（在空间中移动计算）
4. 利用系统组件
    a. 利用局部性
    b. 用空间换速度
    c. 利用硬件特性

    仅对确认为是系统瓶颈的组件运用该原则
5. 增加硬件
    a. 使用交织内存和流水线，提高查表速度
    b. 使用宽字内存，提高访存效率
    c. 结合DRAM和SRAM，兼顾大内存和访问速度
6. 用高效的定制例程代替低效的通用例程
7. 避免不必要的通用性
8. 不要受参考实现的束缚
9. Pass Hints in Module Interfaces（在模块接口中传递线索）
10. Pass Hints in Protocol Headers（在协议头中传递线索）
11. 优化预期情形
12. 增加或利用状态
13. 优化自由度
14. 针对有限规模问题使用特殊技术
15. 使用算法技术构造高效的数据结构

拥塞控制

- 拥塞的原因与现象

    - 原因：对资源需求超过了容量。
    - 现象：从用户的视角，注意到的服务质量因网络负载的增加而下降。
- 拥塞控制（control）和拥塞避免（avoidance）



    - 拥塞控制：保持在cliff之左
    - 拥塞避免：保持在knee之左
- 如何检测拥塞
    - closed-loop：有反馈的系统
        + 发送者基于接收者的反馈调整其速率
        + 显式信号：向发送者发一个包（会有控制流量congestion collapse的问题）；设置包头中的一个位（可能被自私的接收者颠覆）
        + 隐式反馈：接收者端采取的测量，能用于推断网络中发生了什么
        + 延迟和丢包可以表明拥塞，但是它们有很多原因，需要对网络的隐式假设，例如「延迟增加总是意味着排队增长」意味着假定了「一系列包总说走过同一路径」。丢包也可能是因为无线网络。
    - open-loop：无反馈的系统
      需要对网络的先验知识，如瓶颈带宽。
- 拥塞控制机制的设计选项
    - 没有从网络中的显式反馈
    - 端系统从观察到的丢包和延迟推断拥塞
    - TCP采取方法应对
    - 网络可以帮助，路由器可以用单个位告知拥塞，或者显式告知发送者应该发送的速率
    - 基于窗口的操作：
        + 如果感知到拥塞，缩小窗口
        + 否则增大窗口

过去考试问题

一、单选题

1. 下面关于分层体系结构，说法错谈的是
A. 分层是一种模块化的设计。
B. 严格分愿可能会带来性能损失。
C. 某一功能可能在不同厚重复实现。

D. 相邻层间的依赖性导致层的普换和升级困难。
Answer D.

2. 下面关于ISO分层模型和互联网分层模型，说法正确的是
A. ISO 分层模型早于互联网出现。
B. ISO 的传输层和互联网的传输层完成功能一样。
C. ISO 模型从概念上定义了不同层的服务，接口与协议。
D. ISO 分层模型是国际标准组织刺定的，因此会收代互联网分层顿型。
Answer C.

3. 下面哪种对互联网的改变是「演进型」路织?
A. IPv6
B. NDN
C. XIA
D. SDN
Answer A.

4. 下面哪点不是最初互联网体系结构设计的目标
A. 互联已有的网络
B. 可生存性
C. 分布式管理
D. 安全
Answer D.

5. 对OpenFlow协议描述不正确的是?
A. 流表项可以抽象为<Match,Action>原语
B. OpenFlow 协议主要目的是用于OpenFlow规则的下发
C. OpenFlow 协议的匹配域增加需要修改控制器和交换机才能支持。
D. OpenFlow 流表位于数摄平面
Answer B.

6. 互联网IP 层设计为 Best-Effort（尽力而为 ）的主要原因是
A. 应用层对数据包丢失/延迟/乱序到达的需求
B. 灵活性与兼容性
C. 减少了调度算法的开销
D. 公平分配资源
Answer B.

7. 互联网为什么引入middlebox，描述正确的是
I. 解决真实且急迫的安全问题
II. 端到端原则扩展性差
III. 提升网络性能
IV. 第三方监管的需要
A. I、II

B. I、III
C. I、III、IV
D. I、II、III、IV
Answer C.

8. middiebox与路由器机比，通常不具备下面哪个特点？
A. 通常可以处理包头和负荷。
B. 需要维护数据包状态
C. 决策机制依赖网络拓扑。
D. 支持对数据包进行复杂的操作
Answer C.

9。下面哪个点一般不认为是NFV的优势
A. 开放生态系统，允许更多创新
B. 按需的能力扩展
C. 用户个性化定制功能更容易
D. 数据包处理速度更快
Answer D.

10. 端到端模型（End-to-End Argument）有多种不同的解读，下面哪几种解读是合理的
I. 通信系统的一些功能需要在端系统才能正确地实现。
II. 如果功能能在应用层实现，尽量不要放到底层。
III. 不能维护网络中间状态。
IV. 除非在底层实现一个功能能带来众多收益，且不影响其他应用，否则不要实现在底层。
A. I、II、IV
B. I、II
C. II、III、IV
D. I、II、III、IV
Answer A.

11. 下面哪一点不是RED算法的特点？
A. 能提升网络稳定性。
B. 避免全局同步现象。
C. 无需维持每个流的状态信息
D. 支持面向连接和无连接的传输协议
Answer D.

12. 下面哪个点一般不认为是可编程数据平面的特点？
A. 允许自定义数据包处理行为
B. 用户可以快速开发网络功能。
C. 实现网络的完全可视化，可追溯每一个数据包的转发过程
D. 无需控制平面与数据平面之间的协议。
Answer D.

13. 转发和路由的区别，描述不恰当的是
A. 转发是数据包的发送行为
B. 路由是数报包的路径选择
C. 在SDN网络中，路由和转发没有明显区别
D. 路由通常涉及复杂的算法，转发的算法相对简单
Answer C.

二、简答题

1. 流量控制（flow control）与拥塞控制（congestion control）有什么区别?

流量控制用来保证发送端不会以高于接收者能承受的速率传输数据。一般涉及到接收者向发送者发送反馈。
拥塞控制确保发送者不会以超过网络承受能力发送数据。
因此，发送者可能是因为流量控制的原因，也可能是因为网络承受能力（拥塞控制）原因改变发送速率。

2. 对于现有互联网体系结构的研究，大致包括革命型和演进型两种路线，请简要分析两种路线的特点和典型思路?

演进型主张在现有互联网网络体系架构下进行「补丁式」的修补，即基于现有的TCP/IP协议栈，
(1) 提出新的组网技术，改变现有网络形态，包括对网络设备或拓扑进行改造
(2) 对已有的网络通信协议进行升级
(3) 在现有网络中应用如人工智能、区块链、大数据等新技术手段。
从而暂时解决现网中日益暴露的各种问题，使得现有网络架构仍然能够在一定程度上适应新的发展需求。

革命型认为任何技术体系都有它的生命周期，经过若干年就可能发生较大的革新，IP网络体系结构也不例外。
由于IP网络体系结构在设计 的时候并没有充分考虑到当前网络应用的复杂性，因此「革命型」技术路线主张采取「Clean Slate」（从头再来）的策略，即在不受现有互联网约束的基础上探讨新的网络体系结构，重新设计网络通信协议，并将其定义为未来网络体系架构。
目标是为了从根本上克服传统互联网体系结构在可扩展性、安全性、可管可控与QoS保障、移动性、服务分发以及绿色节能等方面的问题，更好地适应未来发展需要，实现网络的可持续发展。

3. 控制平面和数据平面分别完成什么功能？并各举两个例子。

数据平面，负责将数据包从一个接口转发到另一个接口的所有功能和过程。例如。转发Traffic monitoring、Buffering and marking、Shaping and scheduling、Deep packet inspection。

控制平面，决定使用什么策略来处理致据包所有功能和过程。例如，计算转发表。绘制网络拓扑结构、计算安全策略等。

4. 影响SDN控制平面扩展性的根源是什么？SDN实现控制平面扩展性的思路有哪些?
主要原因为集中的控制平面设计，需要维护全网状态、流处理开销等。

SDN 控制平面扩展性的思路:
(1) 提开单个控制器的性能
(2) 控制功能下放到数据平合
(3) 控制器的功能分布到多个节点

5. 互联网设计之初的设计目标包括哪些

Fundamental goal: multiplexed utilization of existing interconnected networks.
Second level goals:
1. Internet communication must continue despite loss of networks or gateways (Survivability).
2. The Internet must support multiple types of communications service.
…

6. 互联网中数据包延迟的原因一值有哪些？哪种原因的延迟被TCP认为是拥塞？

处理延迟，传播延迟，排队延退
TCP 认为缓冲区排队的延退是拥塞。

7. 简述WFQ解决了FQ的什么问题。以及WFQ基本工作原理。

WFQ解决了FQ中包大小对带宽分配的影响问题。
WFQ根据数据流的特性（如五元组信息、IP优先级或DSCP值）进行动态分类。并为每个数据流分配一个虚拟时钟和相应的权重。当数据包到达时，WFQ 会根据华所届数据流的权取将其放入机应的队列中。在每个时钟周期，WFQ会选择虚拟时钟值量小的数据滚，即最「饥饿」的数据流。并发送其队列中的一个数据包。

三、分析题

1. 实现IP QoS主要方法有哪些，可编程交换机的出现，对IP Qos 机制设计有何质的机遇？试用一个具体例子描述如何利用可编程交换机实现某种QoS机制。

(1) 区分不同的流:标记/分类。
(2) 给不同流不同的资源保障，Policing、AQM、Scheduling、Shaping

可编程交换机可以支持标记，也可以支持资源保证。可以用具体例子说明。

2. 模块化设计的一般流程是什么？另外请以NDN为例子，说明不同模块完成什么功能. 模块接口是什么？除了NDN，能否列出2种采用模块化设计的工程系统例子。

流程：1. list all the functional and non-functional requirements of the ideal system that one wishes to design, 2. use these requirements to derive the design implications. (constraints) 3. 0nce such design implications or constraints are derived, the next step is the search for technology that satisfies these constraints and allows one to implement a system with desired requirements. carrying out the above steps requires a lot of system knowledge..

或者回答: top-dom 也可以。divide the overall system into smaller sub-system that can be designed independently of each other such that these gub-system caninter-operate by exchanging appropriate information at the interface to provide the functionality of the overall system.

NDN的模块：NDN中有两类角色，分别为消费者（Consmer）和生产者（Producer）

有两类包，分别为Interest包和Data包，内容（Content）均由名字（Name）作为标识。

工程例子，数字通信系统、计算机体系结构、互联网均可。