

计算机网络

计算机网络基础

华中科技大学电信学院 2018



学习目标

- 了解计算机网络的定义、发展历史；
- 理解设计计算机网络的技术需求；
- 掌握电路交换与分组交换的概念与区别；
- 理解多路复用、统计多路复用的概念；
- 理解客户端服务器的通信模式；
- 掌握层次化计算机网络体系结构的原理，理解网络体系结构中的实体、协议、服务、接口等概念；
- 掌握层次化计算机网络体系结构的实现机制，理解报文的封装、层次间的多路复用与多路分解等概念；
- 掌握七层OSI/RM开放系统互联参考模型、各层功能；掌握TCP/IP的四层体系结构，各层功能，与OSI七层结构对比；提出五层的常用的体系结构
- 掌握主要的网络性能指标，包括带宽、吞吐量、时延、RTT、带宽时延积等，具备计算的能力；
- 理解基于套接字的网络编程的基本步骤；



提纲



- 背景
 - 核心问题: 构造一个网络
- 需求分析
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- 总结



什么是网络?

- 将两个或多个实体连接为在一起的系统
 - 例如: 交通运输网络, 电网, 邮政, 水务网络, 电话网



科技名词定义

中文名称: 网络

英文名称: network

定义1: 由具有无结构性质的节点与相互作用关系构成的体系。

应用学科: 地理学 (一级学科); 数量地理学 (二级学科)

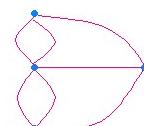
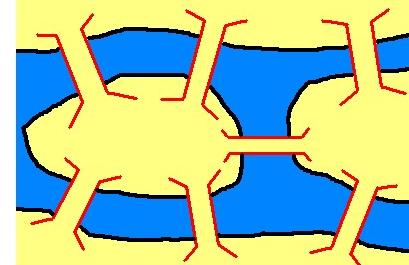
定义2: 作为一个整体看待的、由相互连接的电路元件所构成的集。可用支路和结点来表示。

应用学科: 电力 (一级学科); 通论 (二级学科)

定义3: 在物理上或/和逻辑上,按一定拓扑结构连接在一起的多个节点和链路的集合。

应用学科: 通信科技 (一级学科); 通信原理与基本技术 (二级学科)

哥尼斯堡七桥问题



以上内容由全国科学技术名词审定委员会审定公布

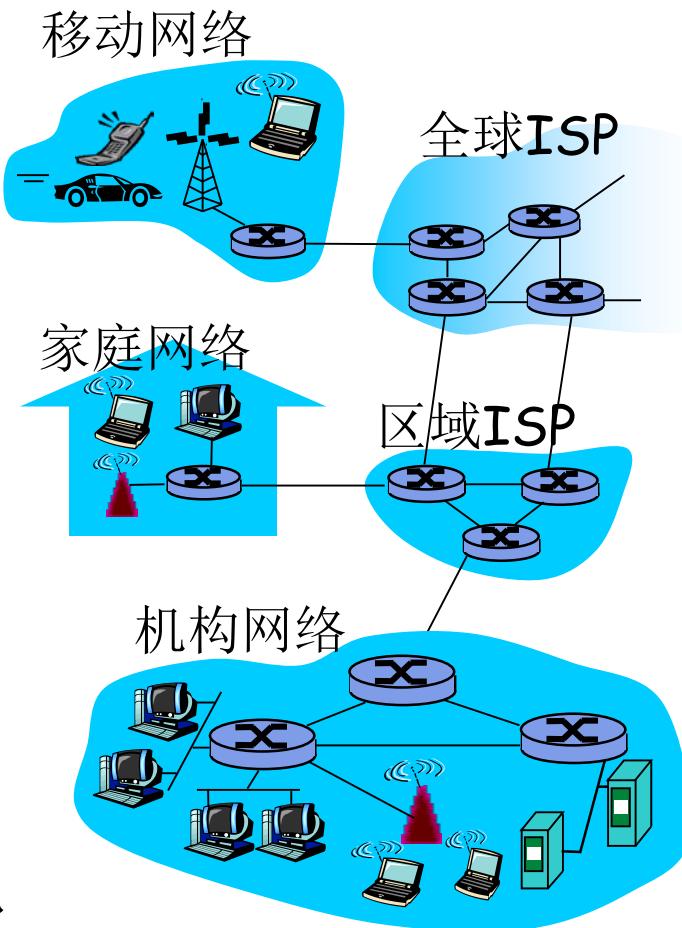
● 计算机网络

- 自主计算机的互联集合
- 互联网(Internet)是一个典型的计算机网络

什么是互联网：系统观点



- 数以百万计的接入设备：**主机=终端系统**
- 运行网路应用程序
- **通信链路**
 - 光纤、铜线、无线电波、卫星
- **传输**
 - $\text{传输速率} = \text{带宽}$
- **路由器：**转发数据包
(数据分组)





“有趣”的互联网应用



IP相框
<http://www.ceiva.com/>



基于网页的烤面包机 +
天气预报机



Tweet-a-watt:
监测能源使用



互联网冰箱



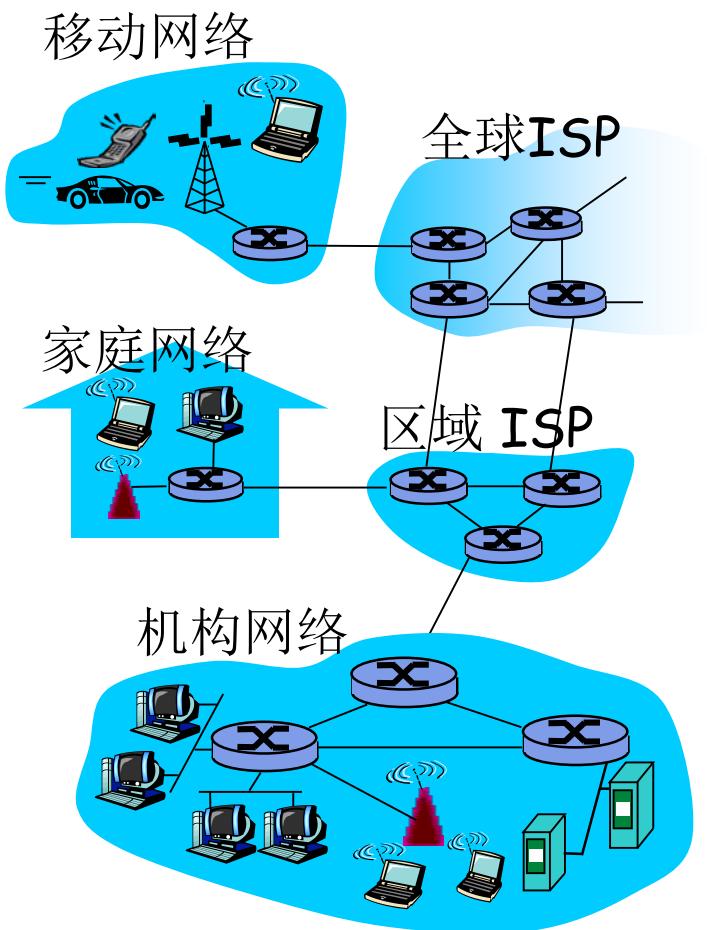
Slingbox: 远程观看
控制有线电视



互联网电话

什么是互联网：系统观点

- **协议**: 控制发送和接收信息
 - 例如: TCP, IP, HTTP, Skype, Ethernet
- **互联网**: “网络的网络”
 - 分层结构
 - 公共互联网和私有内联网
- **互联网标准**:
 - RFC: Request for comments
 - IETF: Internet Engineering Task Force





什么是协议？

口头协议：

- “现在是什么时间？”
- “我有一个问题”
 - 发出特定的信息
 - 当接收到信息或发生其它某些事件时采取相应的行动

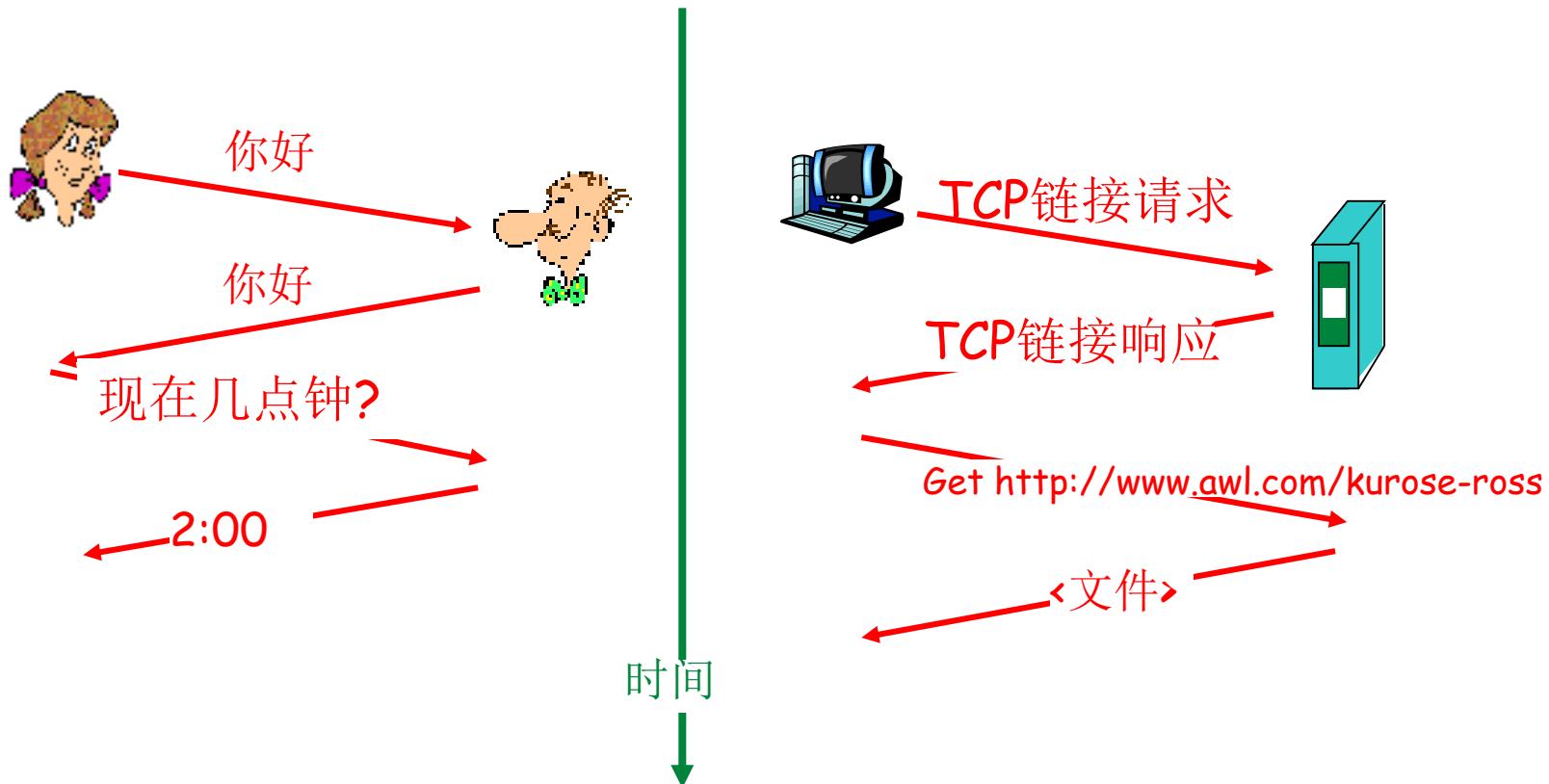
网络协议：

- 由机器执行，而不是人
- 所有的网络交互行为都由网络协议来控制

协议定义了格式，网络实体间发送和接收信息的时序及在信息传输和接收过程中需采取的行动

什么是协议?

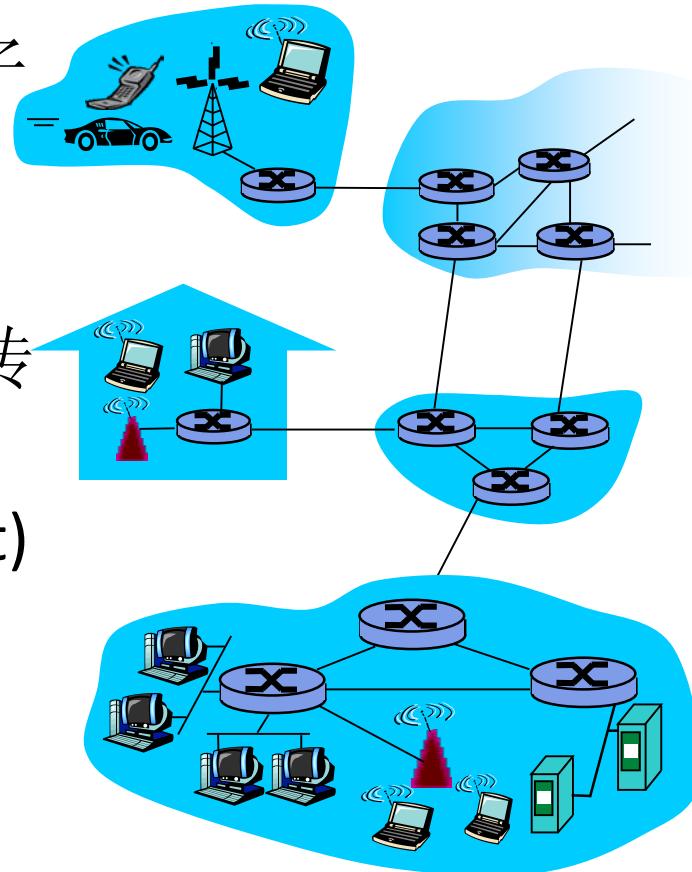
人类口头协议 vs. 计算机网络协议:



问题: 你能举出其他的口头协议吗?

什么是互联网：服务观点

- 通信基础设施支持分布式应用
 - Web, VoIP, email, 游戏, 电子商务, 文件共享
- 提供给应用程序的通信服务:
 - 从源到目的地的可靠数据传输
 - 尽力而为(不可靠Best effort)的数据传输

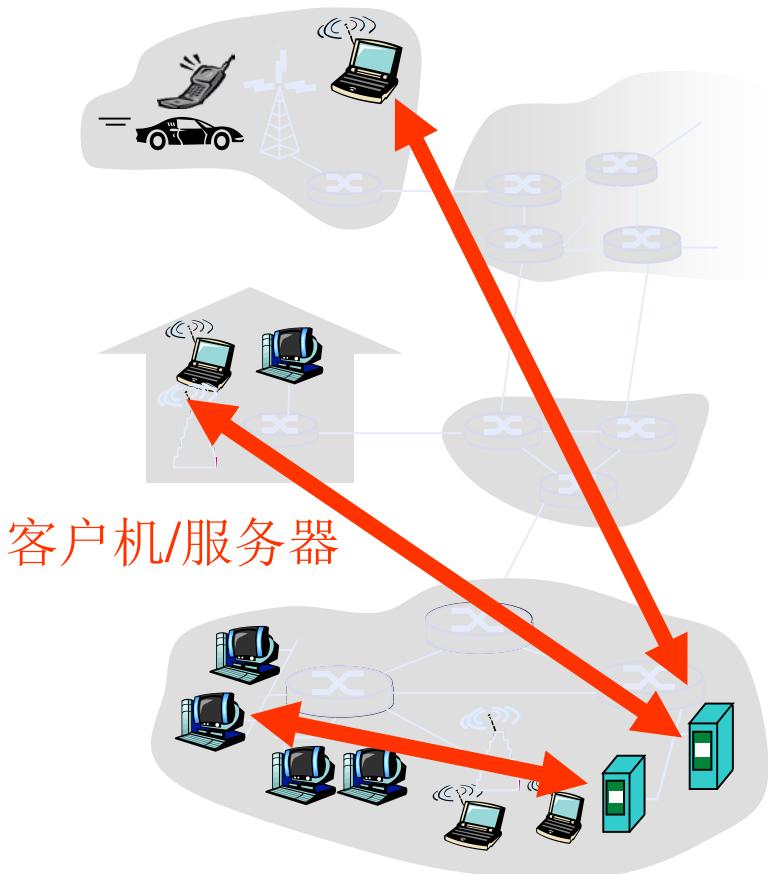




互联网的通信服务模式

- 客户机/服务器
- 对等服务架构(P2P)

客户机/服务器模式



服务器:

- 一直在线的主机
- 拥有固定的 IP 地址
- 服务器集群
 - 获得规模服务

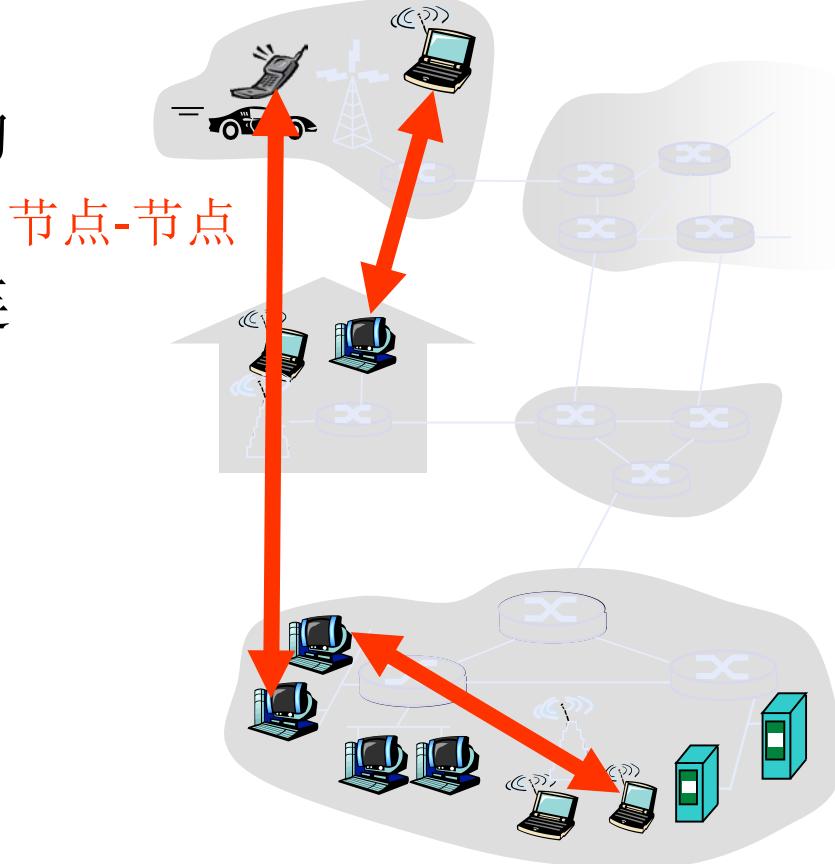
客户机:

- 与服务器通信
- 可能间歇性的连接主机
- 可能拥有动态的IP地址
- 客户机之间并不直接通信

P2P服务模式

- 没有一直在线的服务器
- 任意的端节点之间直接的通信
- 节点之间间歇性的互相连接，交换IP地址信息

网络规模的扩展性很强，但是很难管理





计算机网络 vs 其他通信系统



- 主要区别是什么?
 - 通用性
 - 对特定应用而言并非最优（打电话或传送电视信号），能够传送各种不同类型的数据，支持广泛的不断增长的应用
- 容易混淆的概念
 - 互联的计算机
 - 一台服务器 + 多台从属计算机
 - 一台大型计算机 + 远程终端
 - 分布式系统
 - 建立在网络基础之上的软件系统
 - 用户透明





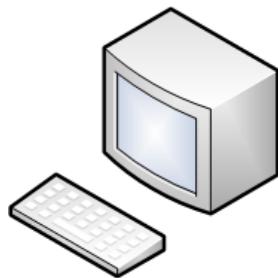
核心问题

如何构建一个网络

讨论

网络组件

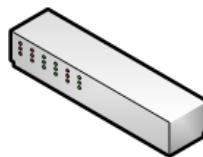
- 节点: 计算机及为互联的计算机提供数据中继的专用设备



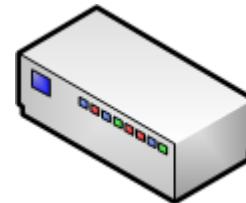
PC



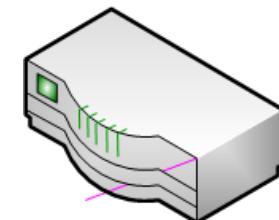
服务器



交换机

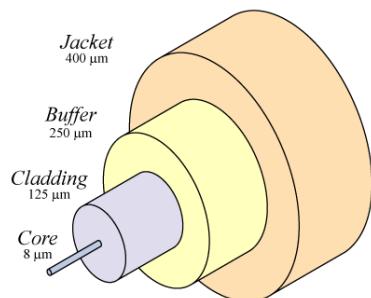


网桥

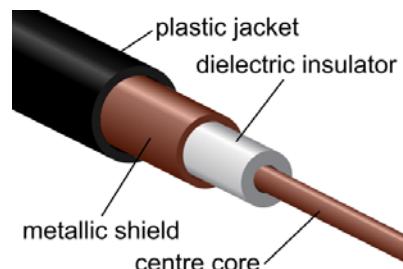


路由器

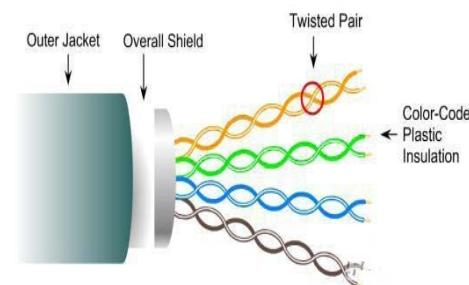
- 链路: 连接节点的传输媒质



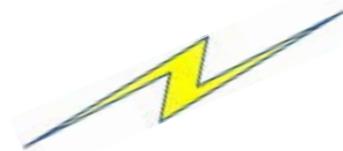
光纤



同轴电缆



双绞线



无线链路



网络设计

- 设计网络，就是将节点通过链路连接起来，使得节点之间可以可靠和有效率地交换信息
- 设计网络，需要考虑网络架构、协议、应用、接口、管理等多方面的问题
- 为了便于展开课程内容，在后续课程中采用“**系统方法**”逐步展开我们的讨论

系统方法(C-I-D-O)



1. 确定研究目标

- 通用的计算机网络系统

2. 进行需求分析

- 确定预期的网络特征、功能、性能等
- 了解目前可用技术实现手段

3. 开展系统设计

- 体系结构概要设计：组件、模块、接口.....
- 具体模块详细设计：协议、软件、硬件.....

4. 讨论系统实现

- 软件、硬件.....

5. 进行系统测试

- 评估系统的功能、性能，并进行优化



第1章的内容组织



问题：如何建造一个计算机网络？

系统方法的步骤	第1章的主要内容
设计目标	1.1 网络应用
需求分析	1.2 设计需求
系统的设计	1.3 网络体系结构
系统的使用	1.4 实现网络软件
系统的评估	1.5 网络性能



网络设计的驱动力 (一)

- Internet上的各种不同应用
 - 信息分发: WWW, BBS
 - 资源共享: 文件服务, 数据服务, BT下载
 - 个人通信: Email, 即时通信, 视频会议
 - 娱乐: 在线游戏, 视频点播
 - ...
- 不同的技术特征
 - 安全的文件传输: 丢包敏感
 - 流媒体: 时延敏感
 - 交互式多媒体应用: 双向
 - ...
- 总结
 - 应用的多样性增加了Internet设计的复杂性



网络设计的驱动力 (二)

- 谁负责开发网络?
 - 应用程序员、 网络设计者、 网络提供者
 - 企业、 政府、 个人
 - ...
- 在哪里构建网络?
 - 家里, 楼宇, 校园, 省市, 国家, 洲际, 全球
 - ...



概述

- 背景
 - 核心问题: 构造一个网络
- ● 设计需求
 - 可扩展的连通性
 - 高性价比的资源共享
 - 支持通用服务
 - 可管理性
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- 总结



设计需求

- • 可扩展的连通性
 - 提供多个计算机之间的连通性
- 高性价比的资源共享
 - 在资源约束条件下保证其通信的有效性
- 支持通用服务
 - 确定通用通信模式
 - 可靠性
- 可管理性

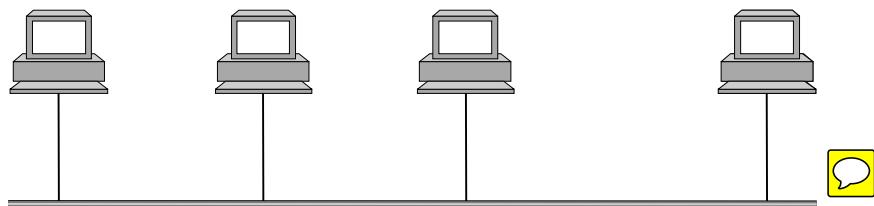


连通性



最简单的方式: 直连 (一条链路)

- 节点 (广义)
 - 主机: 互联的计算机
 - 节点 (狭义): 中间结点
- 链路
 - 点到点链路
 - 多点访问链路



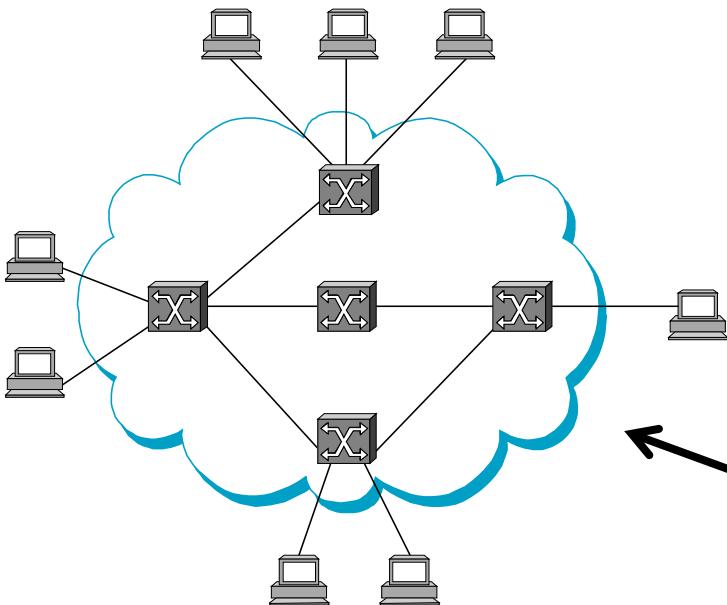


连通性



较复杂的方式: 间接连接 (更多的链路)

- 交换网络
 - 一种基础网络
 - 两种类型的交换网络 (不局限于计算机网络): 电路交换 vs 分组交换

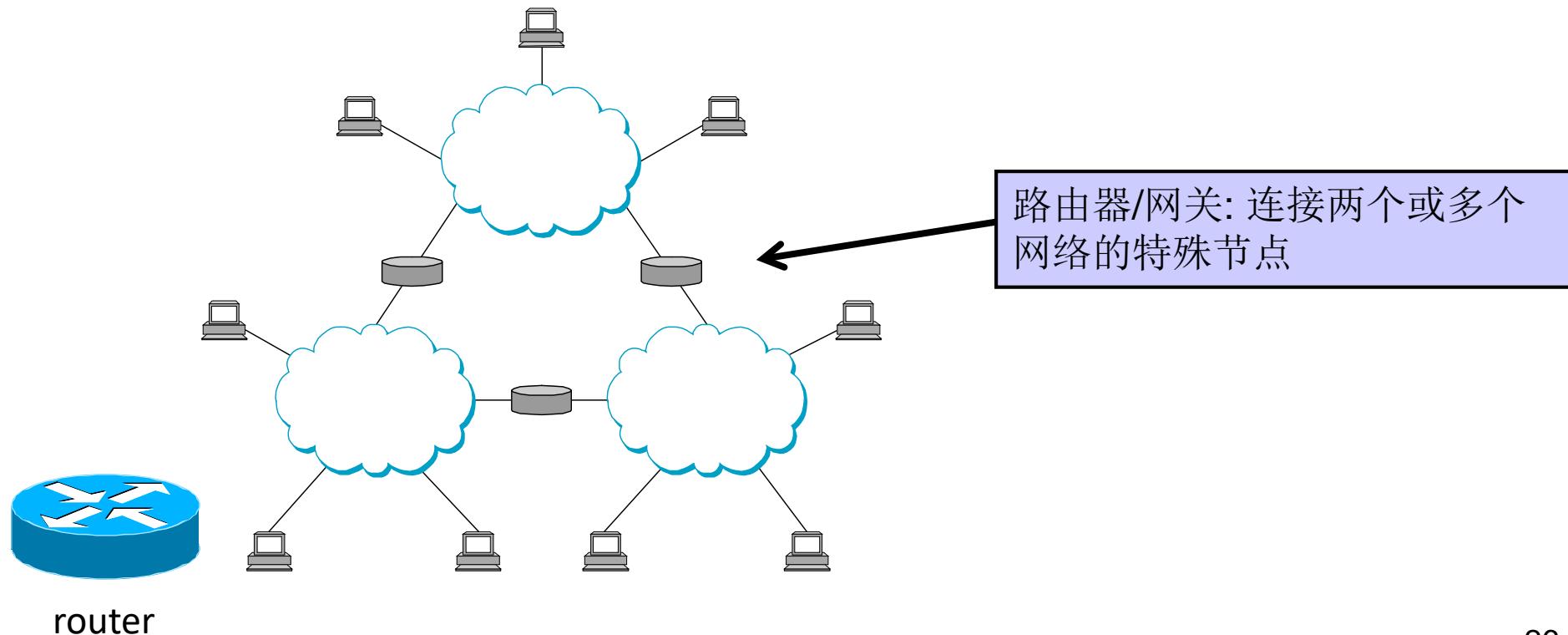


云可以用来表示任何类型的网络, 可以是一个最简单的点到点链路, 一个多路访问链路或者一个交换网络.

连通性

更为复杂的情况: 网络互联 (多个网络)

- 网络互联
 - 多个独立的网络 (云) 相互连接形成互联网





连通性

- 主机和主机的连通性
 - (1) 计算机之间的直接连接或间接连接
 - (2) 网络节点的识别 → 寻址
- 地址
 - 标识主机或节点的字节串
- 路由
 - 如何将报文转发至目的节点
- 基于目的地址的转发
 - 单播: 一对一
 - 多播: 一对多
 - 广播: 一对所有



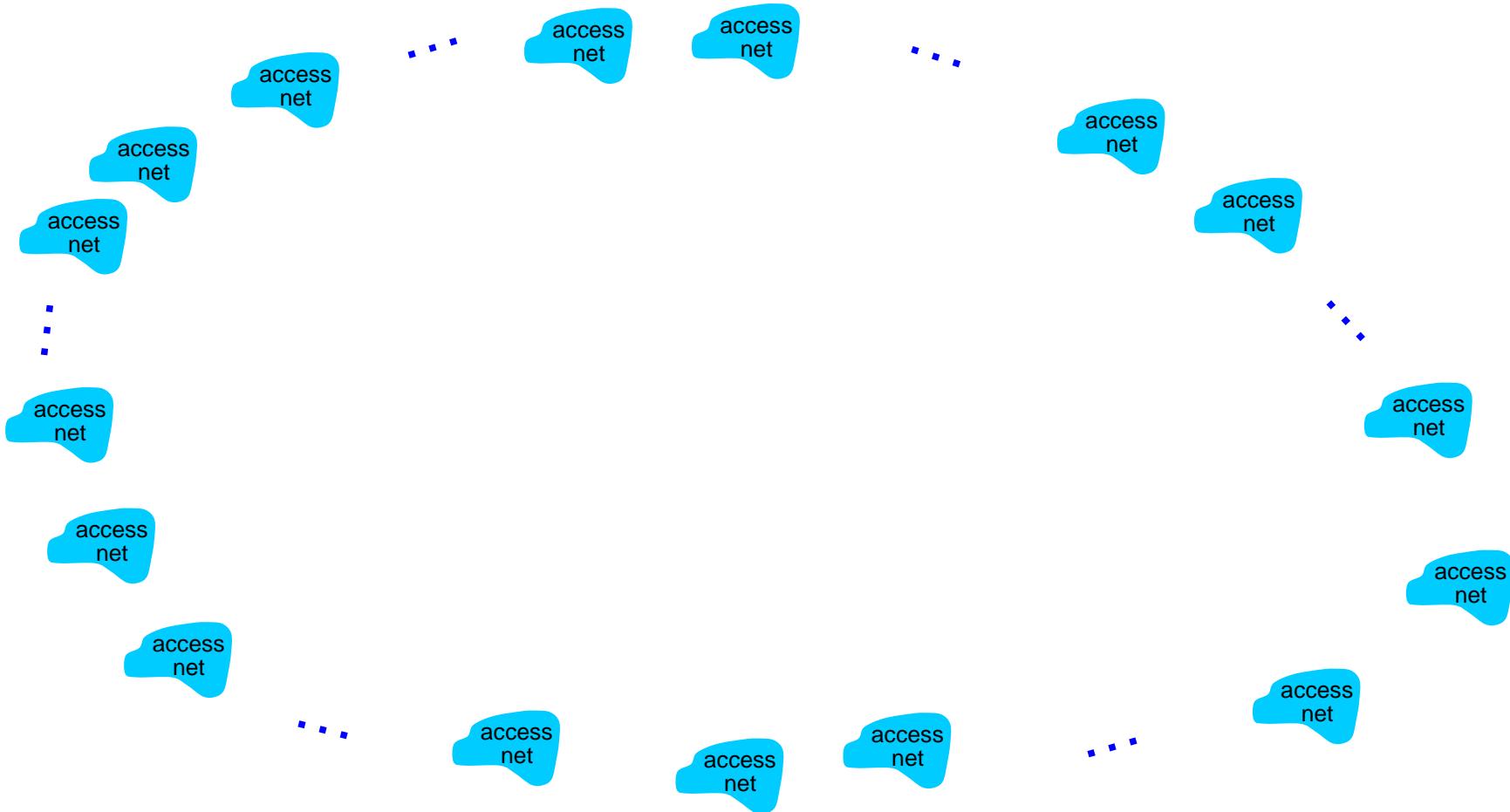
小结:连通性

- 以递归方式定义“网络”
 - 两个或多个节点通过一条物理链路链接
 - 或者 两个或多个网络通过一个节点互联
- 挑战
 - 如何为每一个节点分配一个地址?
 - 如何利用节点地址将报文发给正确的目的节点? 

可扩展的连通性



问题: 如何计算一个网络的效用?



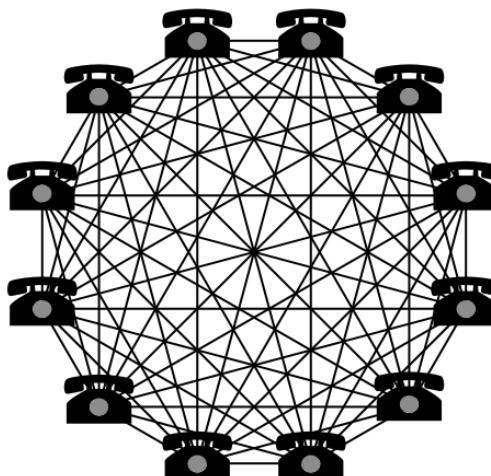
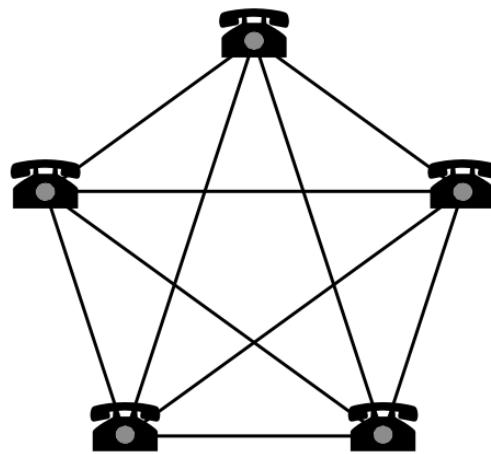
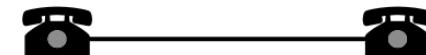


可扩展的连通性



问题: 如何计算一个网络的效用?

- 梅特卡夫定律(Metcalfe's Law)
 - 网络价值将以二次方增长——与成员数量的平方成正比，而成本至多以线性增长。
 - 凭借经验的大致描述，而不是永恒的物理定律
- 局限性
 - 在有 n 个成员的通信网络中，每个成员可以与其他成员建立 $n-1$ 个关系。
 - 假设所有这些关系具有**同等价值**





设计需求

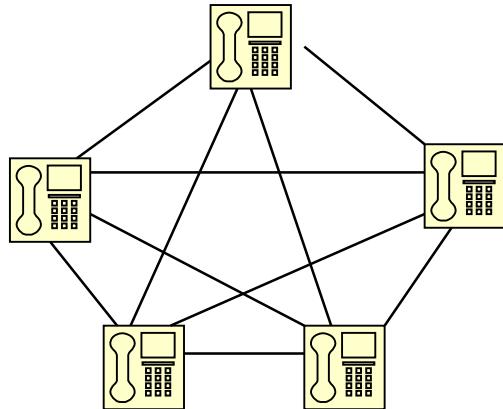
- 可扩展的连通性
 - 提供多个计算机之间的连通性
- → 高性价比的资源共享
 - 在资源约束条件下保证其通信的有效性
- 支持通用服务
 - 确定通用通信模式
 - 可靠性
- 可管理性



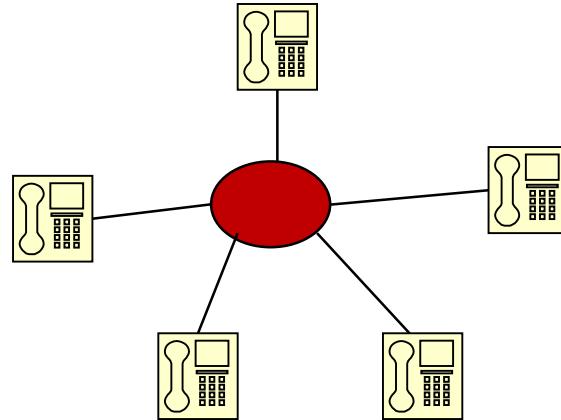
高性价比的资源共享

- 建立在连通性基础之上的需求
 - 所有的主机之间可以同时交换消息
 - 能够适应突发的计算机通信
 - 有效且公平的共享有限的网络资源
- 深入讨论交换网络，研究其资源共享策略
 - 交换网络
 - 多路复用技术

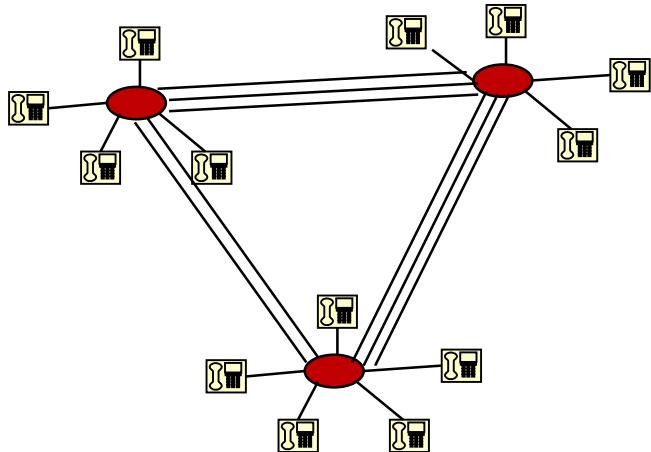
交换网络和复用技术



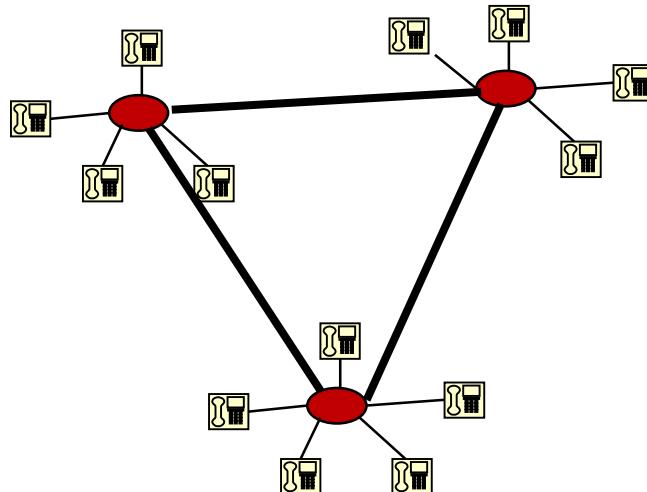
没有交换



星状交换



没有复用

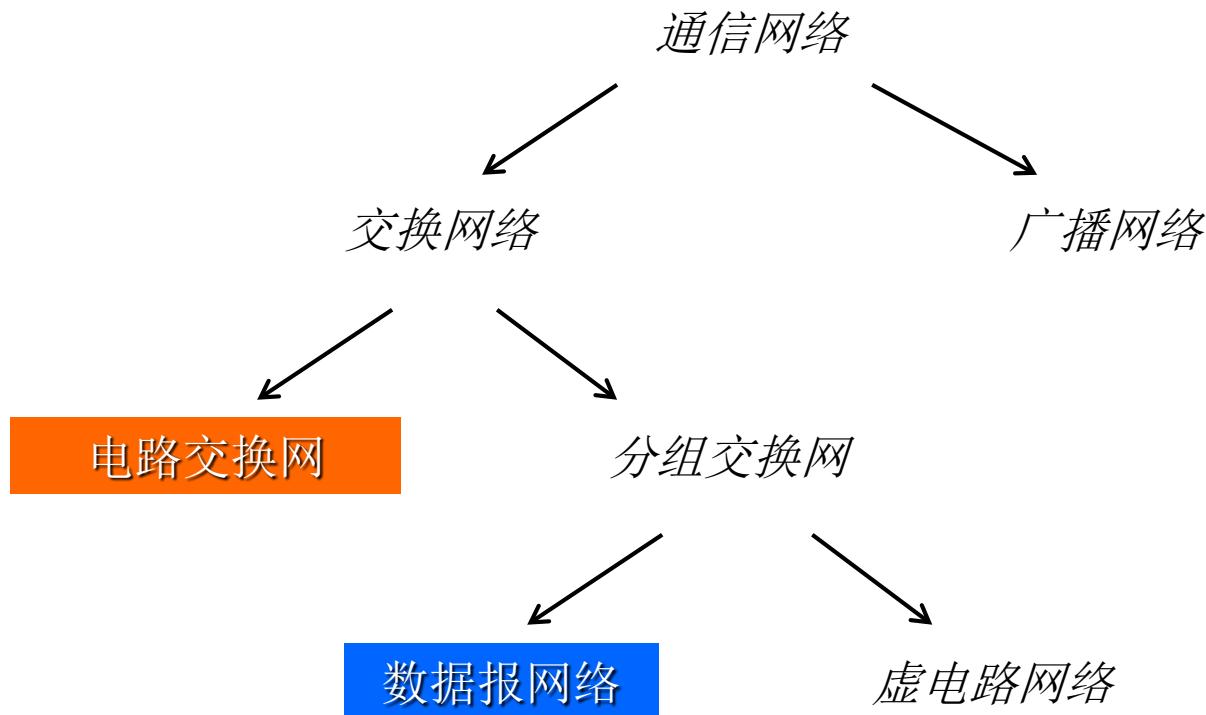


链路复用



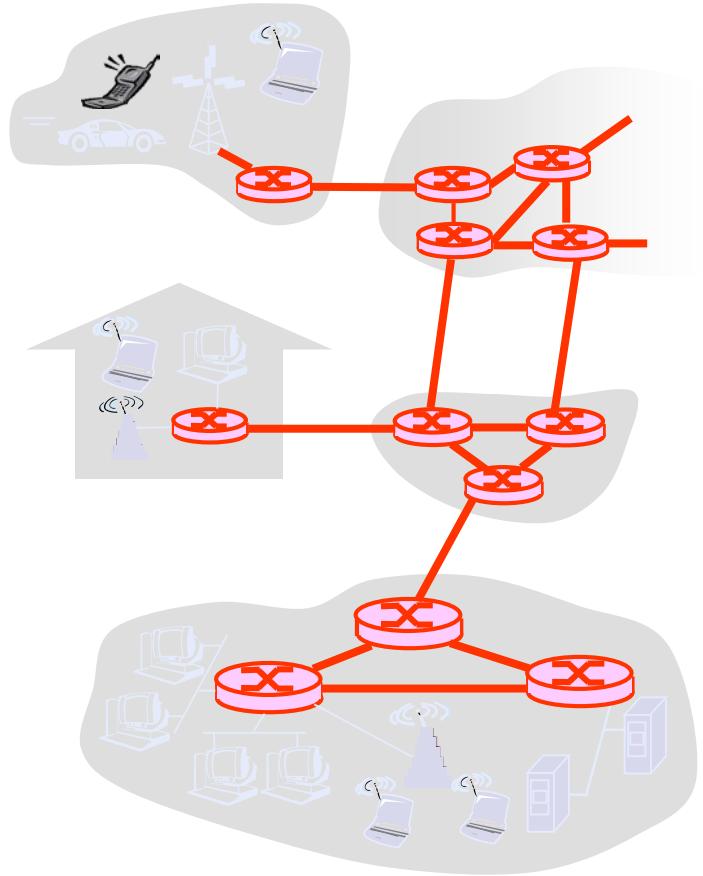
通信网络的分类

- 通信网可以根据节点交换信息的方式进行分类：



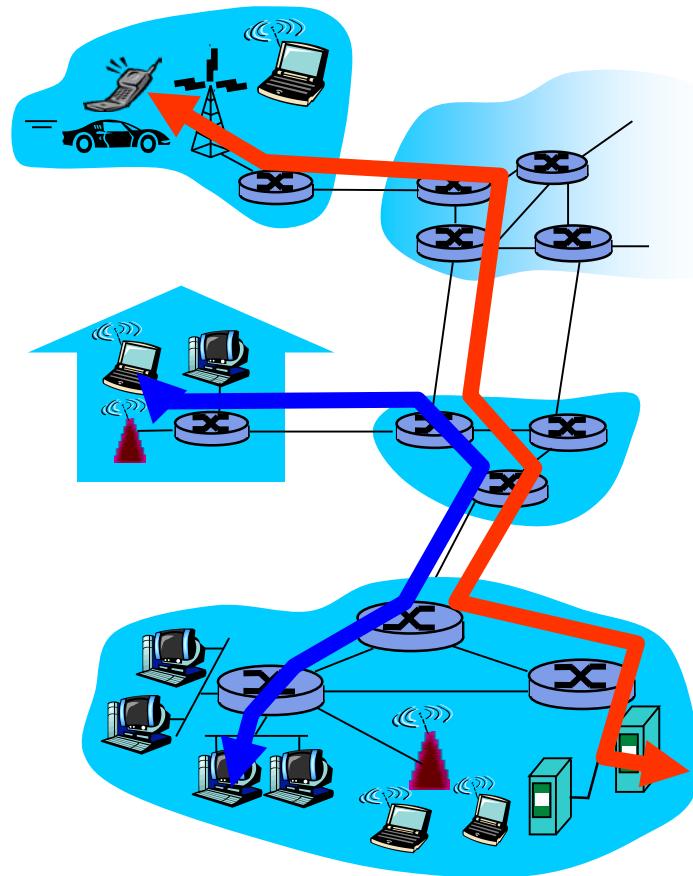
交换网络的技术实现

- 一个基本问题：数据是如何通过网络进行传输的？
 - 电路交换：每次通话需占用专用电路：电话网
 - 分组交换：数据通过离散的数据分组进行传输



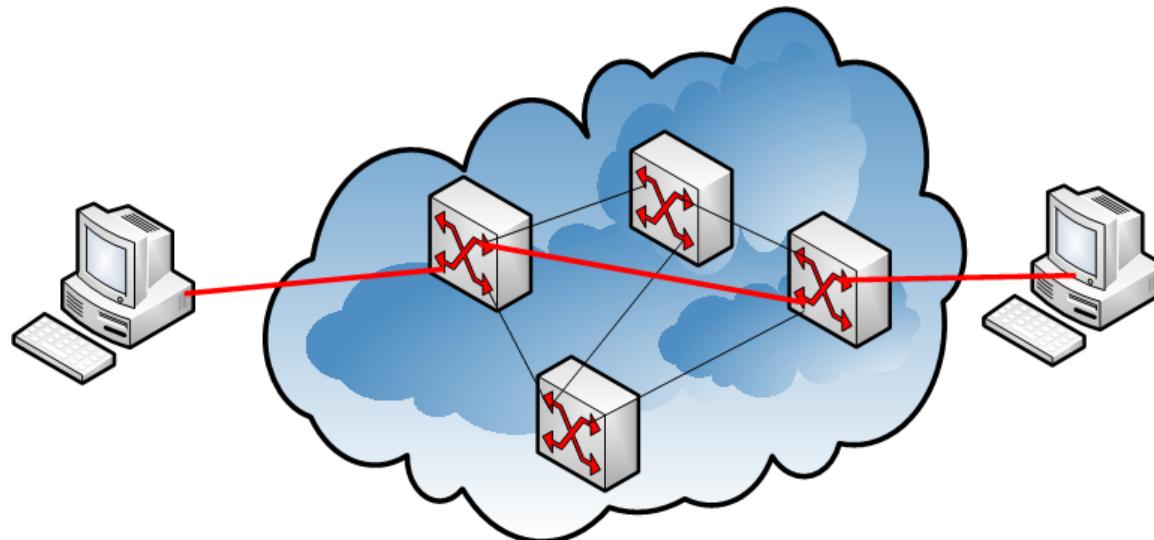
电路交换

- 通信终端为每次通话事先预留资源
 - 链路带宽，交换能力
 - 专用资源，不共享
 - 电路级别(服务质量有保障)的服务
 - 必须先建立呼叫连接

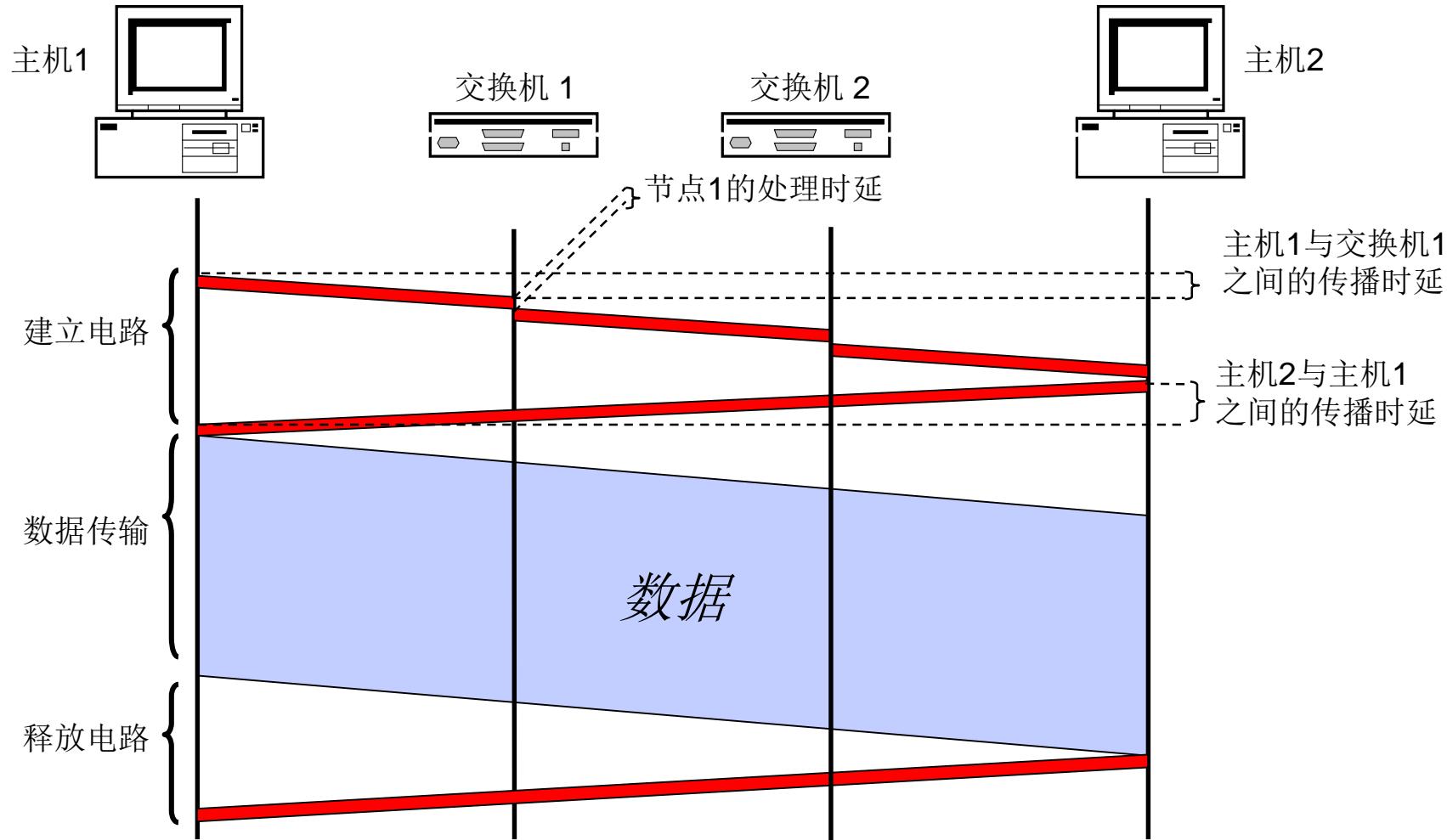


电路交换

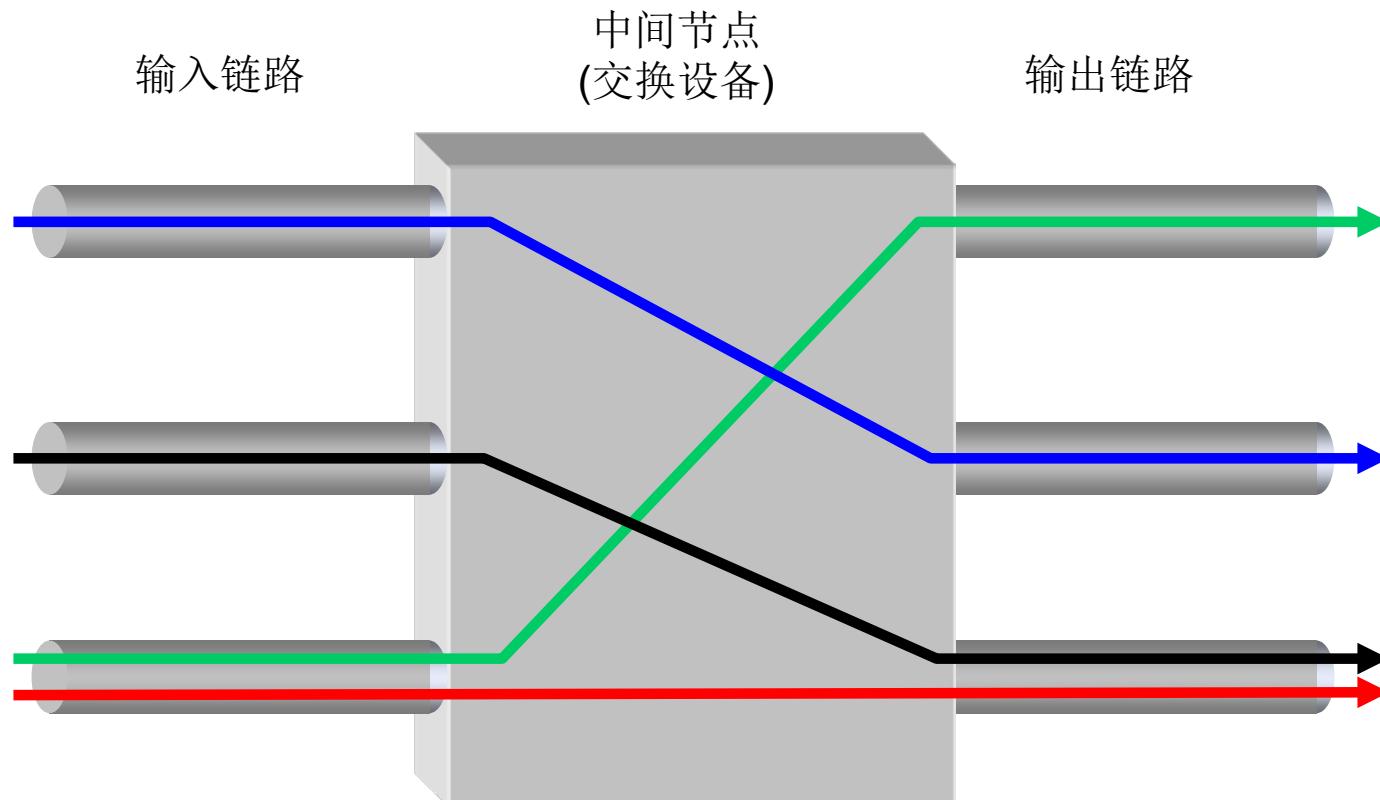
- 端到端的持续链接
 - 流经特定的通信路由
 - 数据报文中不需要目的地址
- 一旦通信结束, 连接自动断开并释放链路.



电路交换: 时序图



电路交换: 中间节点





电路交换

网络资源(例如带宽)被
分成“片”

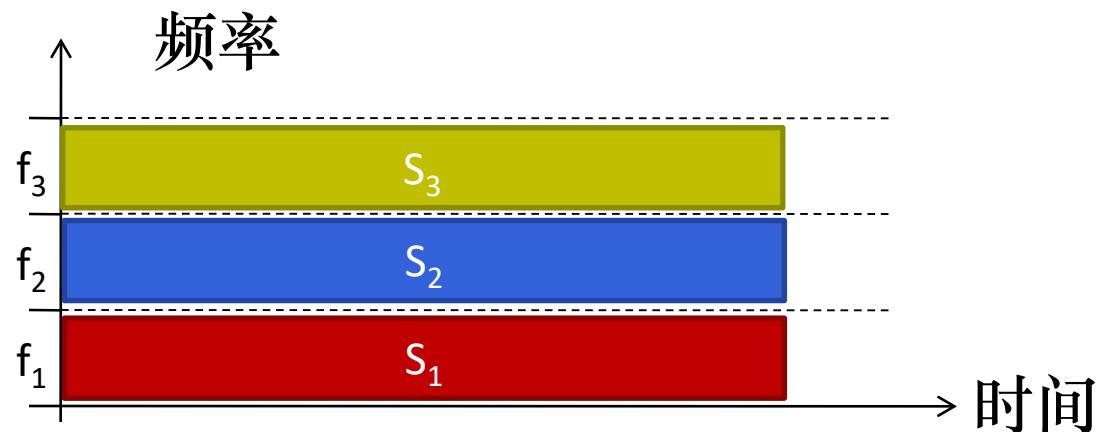
- 将资源“分片”分配给每个通话。
- 资源分片在没有被所属的通话使用时虽然是空闲的却不能共享。
- 将链路带宽“分片”：
 - 频分
 - 时分
 - 码分

电路交换：多路复用

- 时分复用



- 频分复用





案例分析

- 使用电路交换网络，将一个长为640,000 比特的文件从主机A传输到主机B需要多长时间？
 - 所有的链路带宽均为1.536 Mbps
 - 每条链路使用时分复用，每秒钟分为24片时隙
 - 建立端到端电路需要500 毫秒

我们来计算一下！

$$\begin{aligned}&\text{建立链路时间} + \text{文件传输时间} \\&= 500 \text{毫秒} + 640000 \text{比特} / (1.536 \text{Mbps} / 24)\end{aligned}$$



电路交换的优点

- 带宽保证(服务质量)
 - 具有可预测的比特率和时延
 - 适用于时延敏感性应用
- 通信可靠
 - 数据包丢失概率低
 - 数据包按序传送
- 路由机制简单
 - 基于时分或频分复用的数据转发
 - 不需要进行数据包头部的地址识别
- 数据包结构的额外开销较小
 - 不需要额外的协议(IP/TCP/UDP)首部



电路交换的缺点

- 浪费带宽
 - 突发流量将造成已建立连接的长时间空闲
- 受限连接
 - 当资源不足时无法建立新的连接
 - 无法向所有人提供“基本”服务
- 连接建立时延
 - 连接建立成功后方能进行通信
 - 即时较少的数据传送也无法避免数据连接建立时延
- 网络状态
 - 网络节点必须保存每一个已建立连接的状态信息
 - 对节点异常的容错性较差



分组交换

端到端的数据流被分成数据包

- 用户A, B的数据包共享网络资源
- 每个数据包获得全部网络带宽
- 资源在需要时才被占用

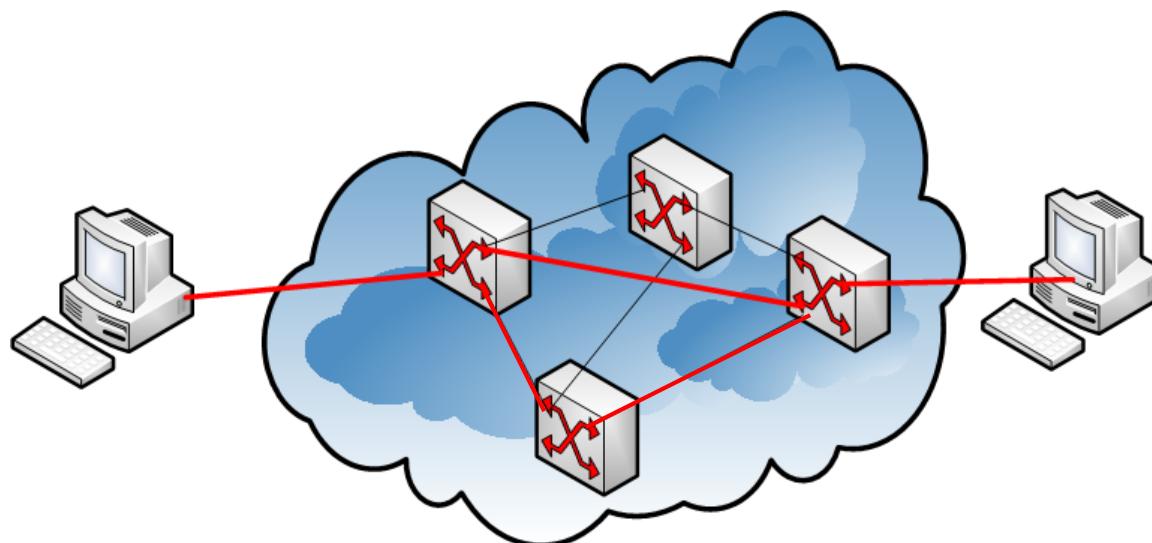


资源竞争：

- 总的资源需求可能超过可用资源
- 拥塞：数据包排队，等待可用链路
- 存储转发：数据包一次传输一跳的距离
 - 节点接收到完整的数据包才转发

分组交换网络

- 数据报文被划分为许多较小的包
 - 每个数据包均包含识别信息 (源地址/目的地址, 序列号等等)
- 数据包在网络中独立传送
 - 根据数据包中的目的节点地址进行转发
 - 某一数据流的包可能通过不同的路径传送, 节点会临时性地进行数据包存储



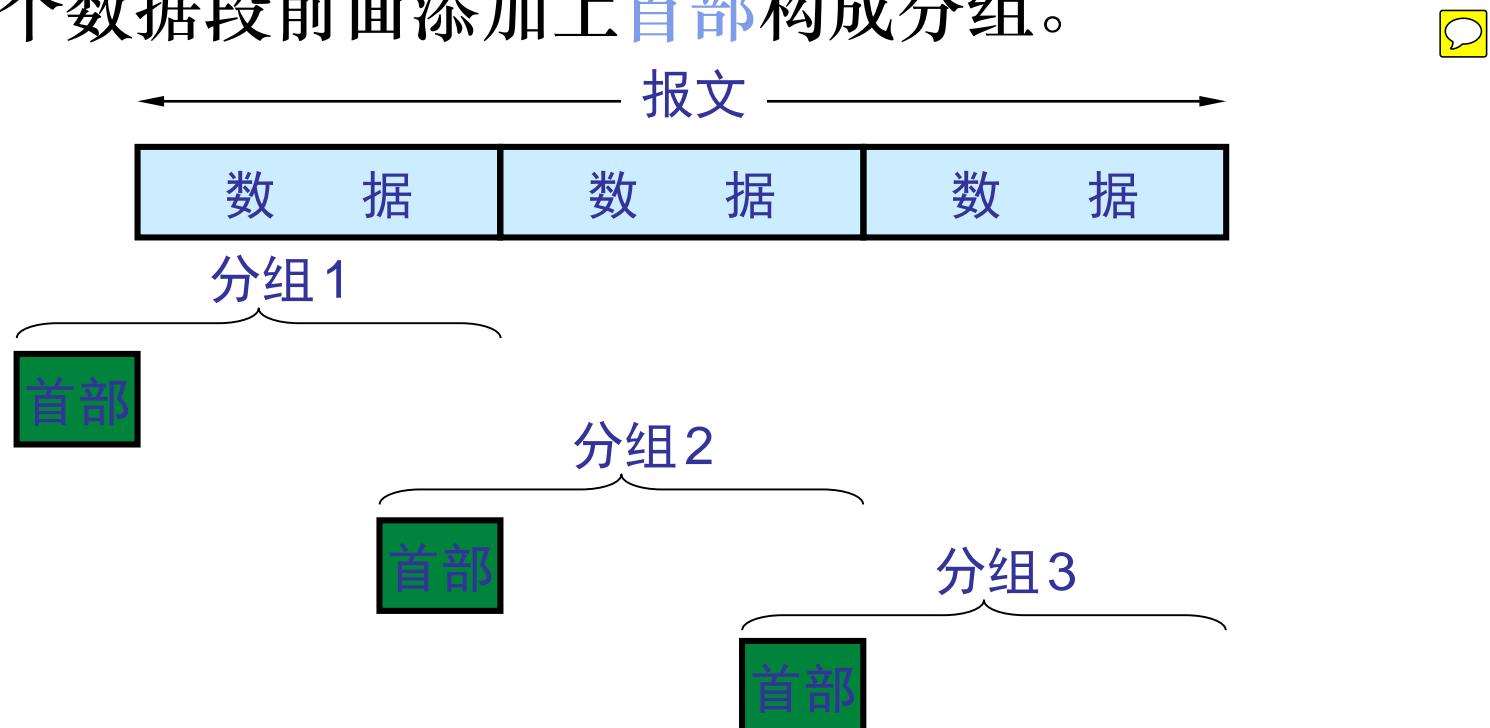


分组交换的原理（一）

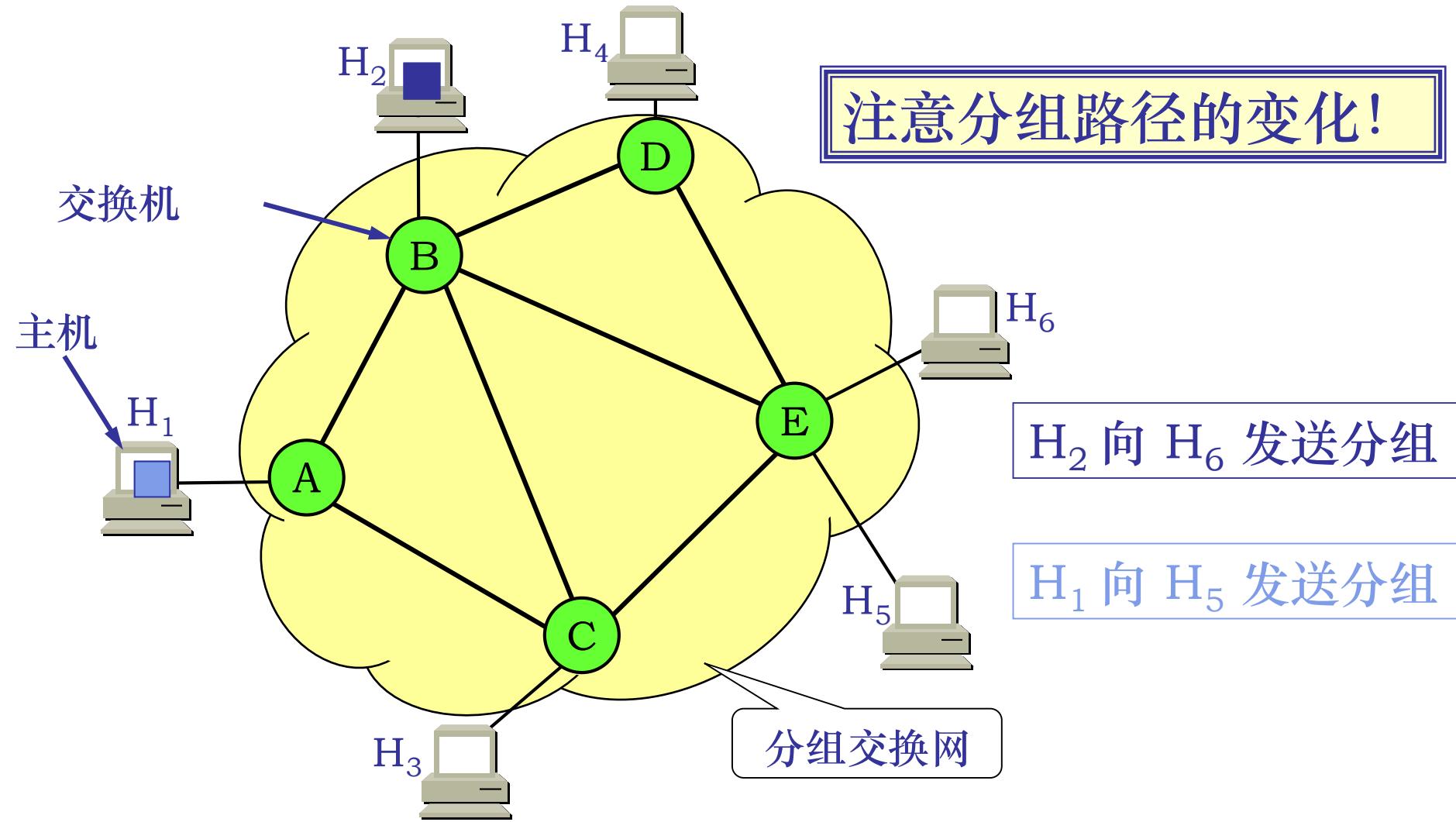
- 在发送端，先把较长的报文划分成较短的、固定长度的数据段。



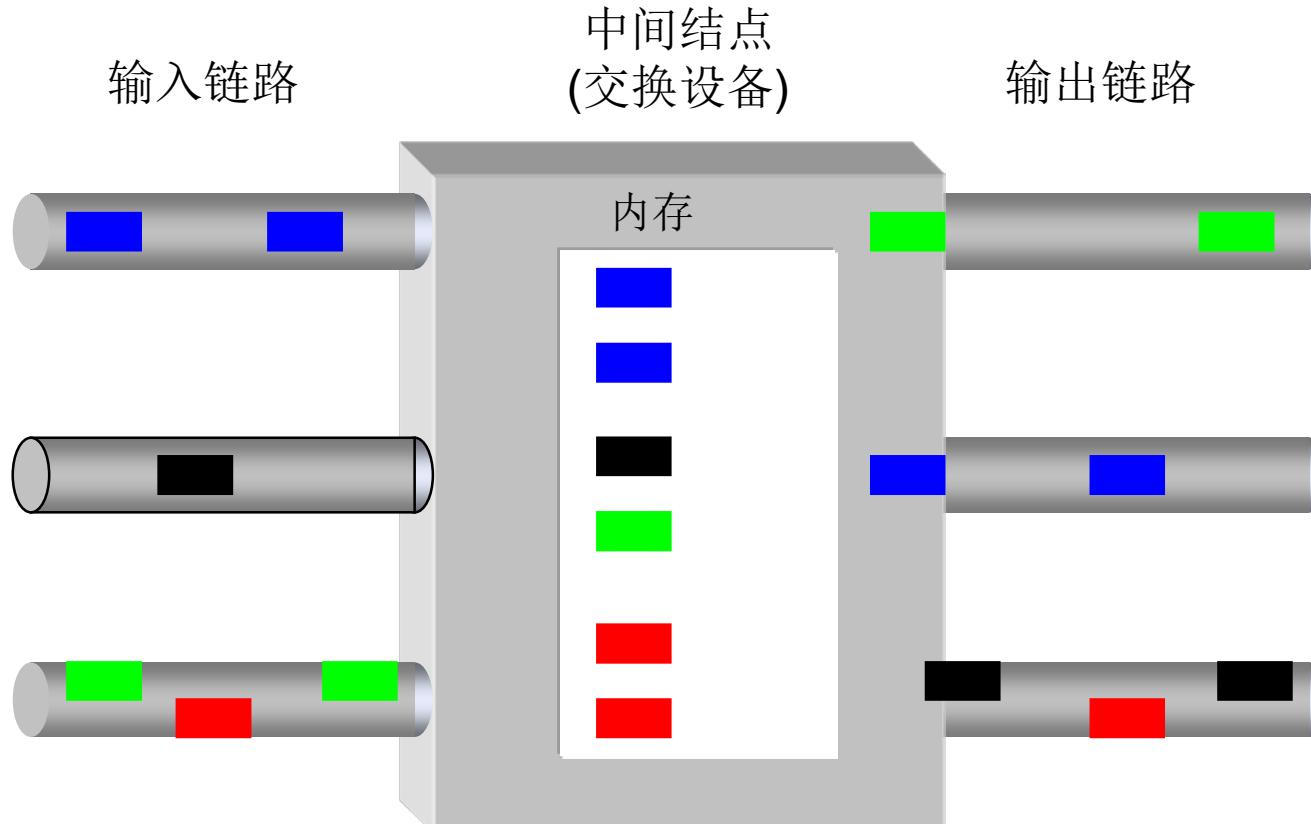
- 每一个数据段前面添加上首部构成分组。



分组交换的原理（二）



分组交换: 中间结点

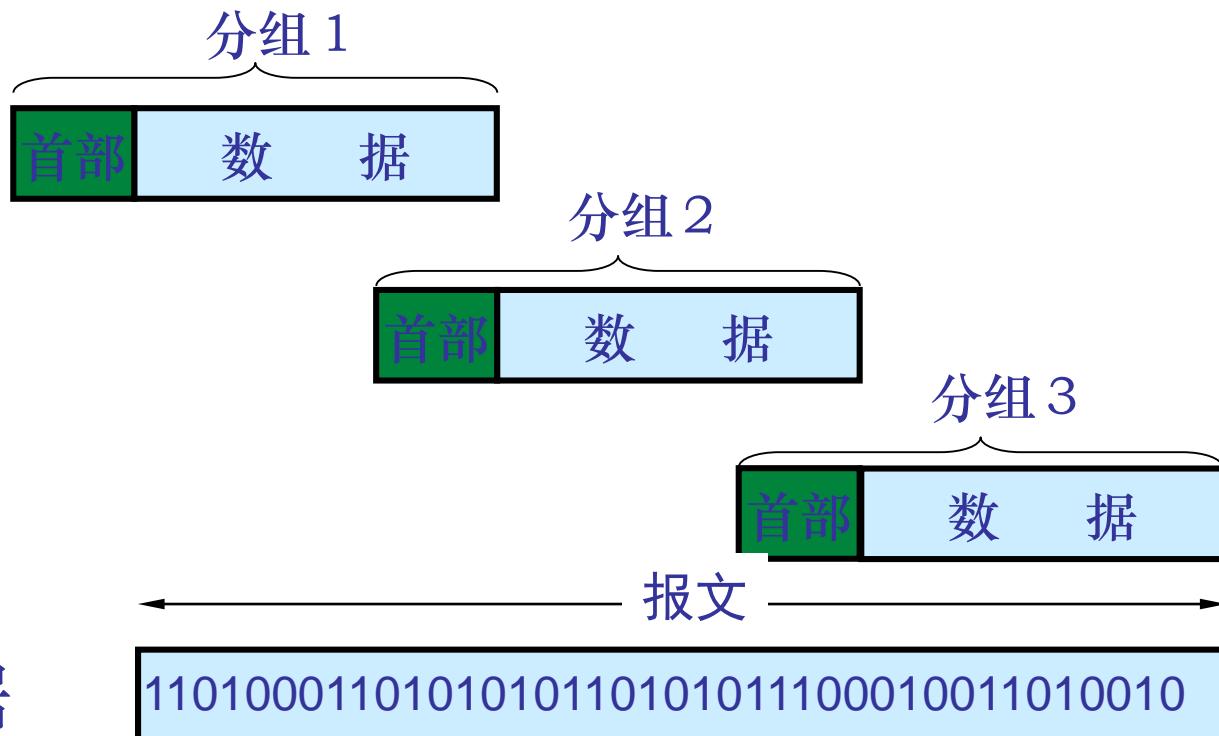


拥塞: 如果一个交换设备接收数据包的速率大于发送的速率, 则交换设备的存储空间将消耗殆尽, 部分数据包必须丢弃.



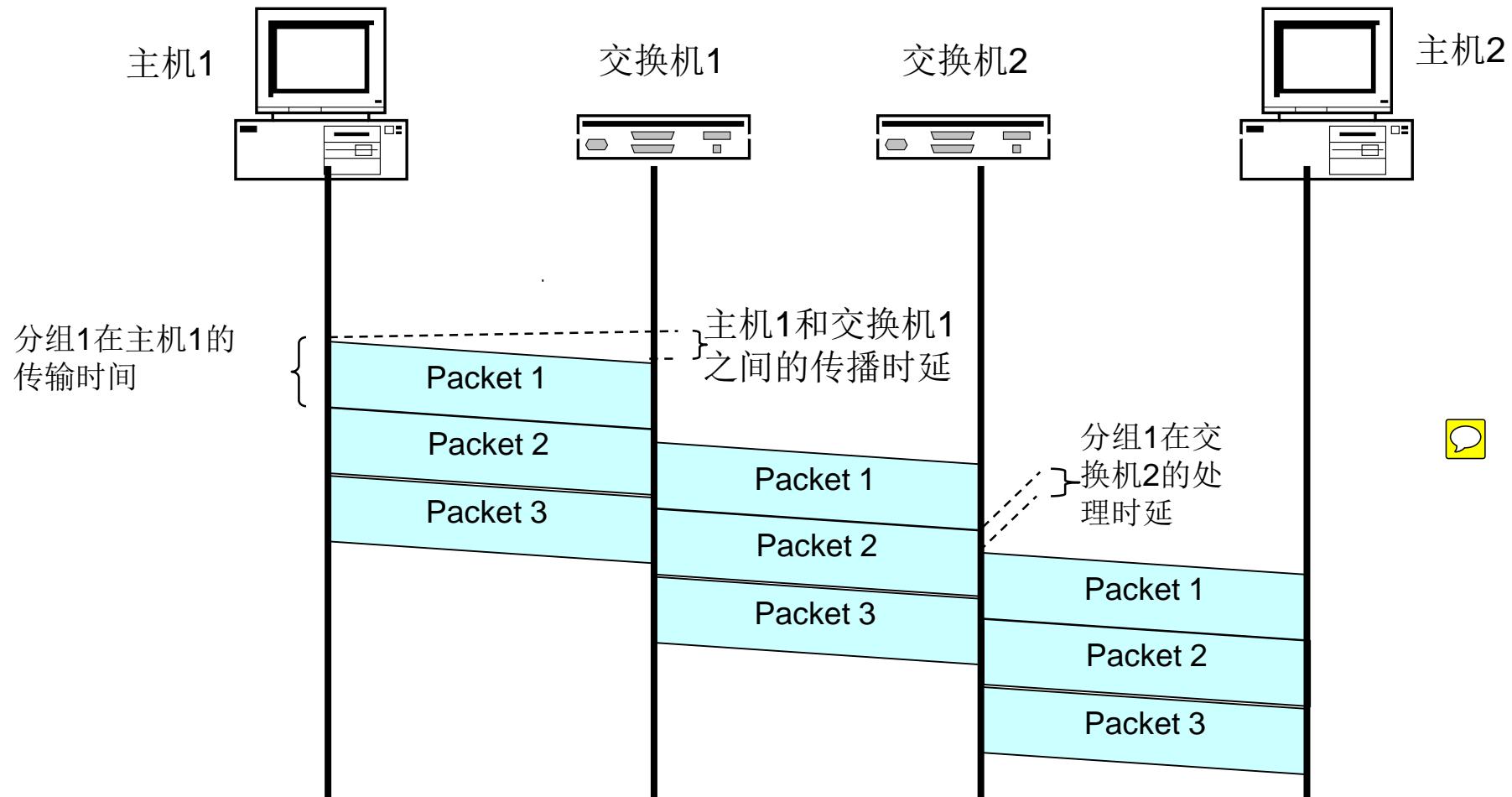
分组交换的原理（三）

- 接收端收到分组后剥去首部还原成报文。



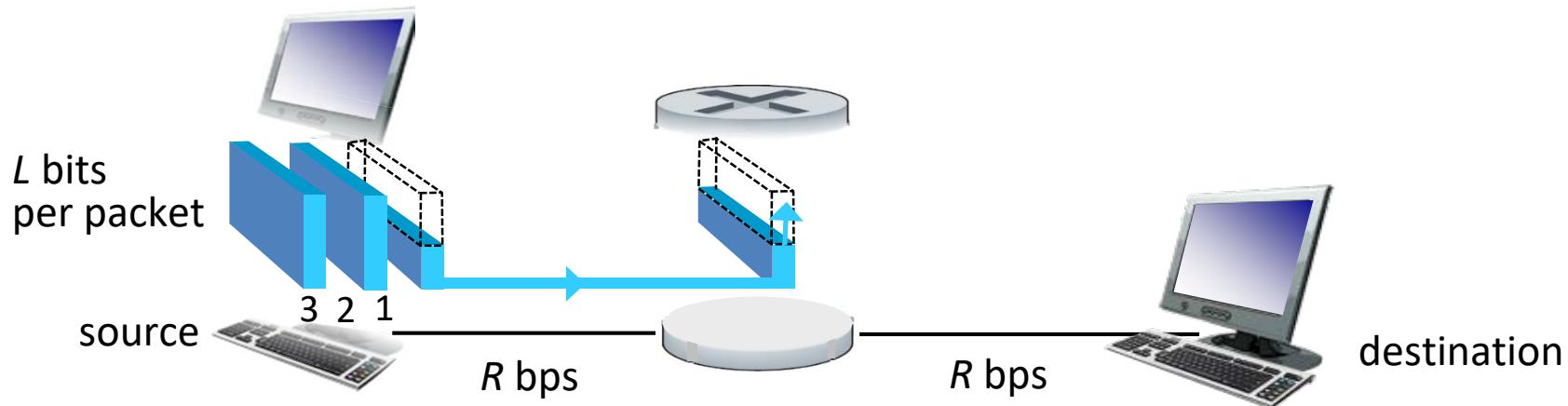
收到的数据

分组交换: 时序图



存储转发策略通过链路并行传输以高效利用链路带宽

分组交换：存储转发

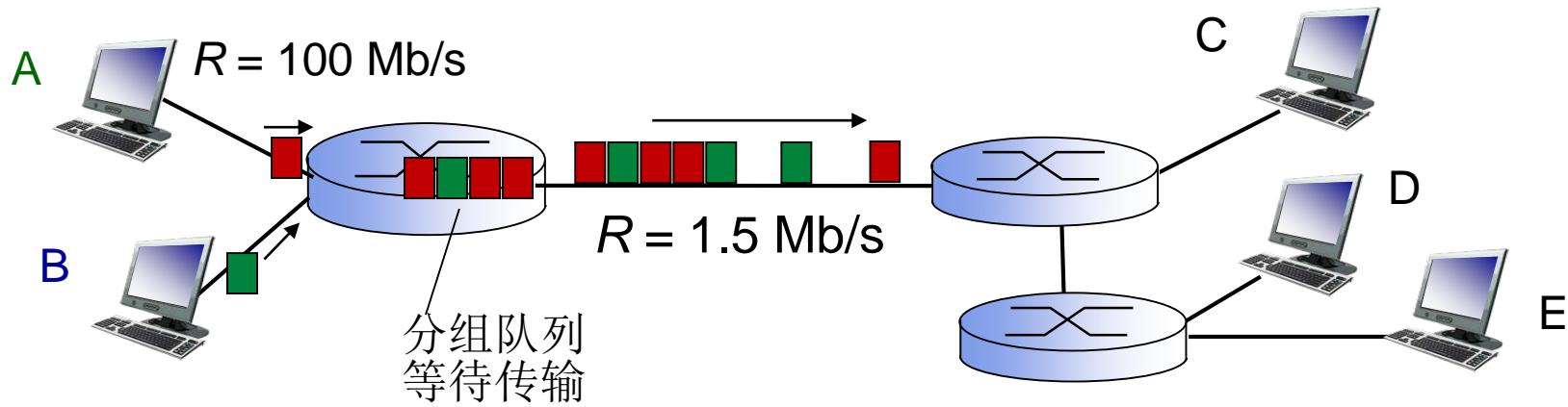


- 需 L/R 秒 将一个 L bits 的数据包传送到速率为 R bps 的链路
- **存储转发：** 整个数据包需到达路由器后才能被转发到下一条链路
- ❖ 端到端时延 = $2L/R$ (假设电波传输时延为 0)

一跳传输数值计算:

- $L = 7.5 \text{ Mbits}$
- $R = 1.5 \text{ Mbps}$
- 一跳传输时延 = 5 sec

分组交换: 排队时延, 丢包



排队和丢包:

- ❖ 当网络流量的到达速率持续大于链路的传输速率:
 - 分组排队, 等待链路的传输
 - 当路由器的内存(缓存)溢出, 出现丢包



分组交换的优势

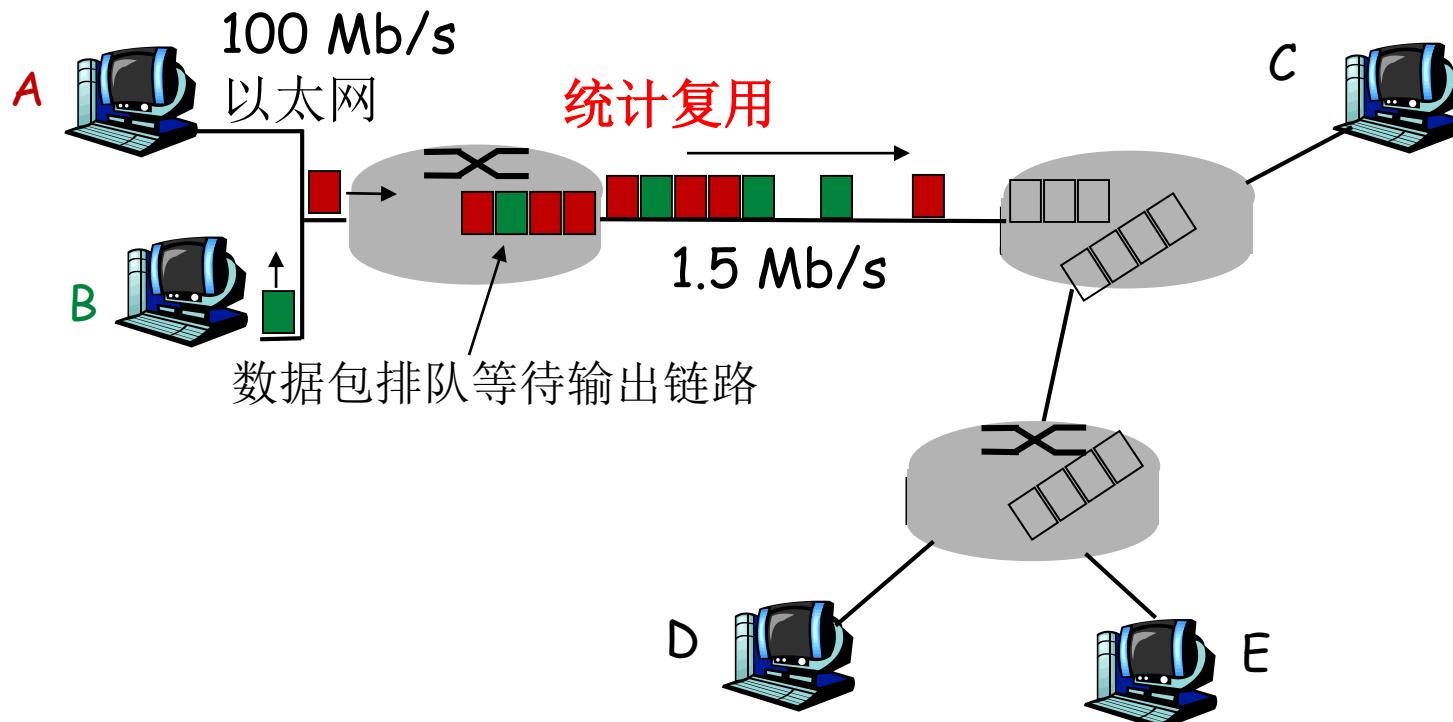
- 不存在带宽浪费 (并不一定完全准确)
 - 在链路空闲时不预留链路资源
- 多路复用技术
 - 统计复用
- 服务
 - 尽可能建立更多的连接
 - 不拒绝新用户服务请求
- 健壮性
 - 对网络错误及拥塞具有一定的自治愈性



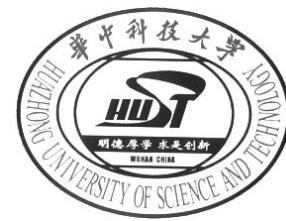
分组交换的缺点

- 无法保证带宽
 - 难以支持需要服务质量(QoS)的应用
- 每个分组的开销
 - 需要包含源地址和目的地址等信息的首部
- 复杂的端到端控制
 - 分组可能丢失、错误以及乱序到达
- 时延和拥塞
 - 无拥塞控制，可能导致不确定的时延甚至发生数据包丢弃

分组交换：统计复用



- **统计复用:** A和B的数据包顺序没有固定的模式，各自根据需要共享带宽。
- **时分复用:** 在TDM帧中每个主机被分配一个同样大小的时隙。



电路交换 vs 分组交换

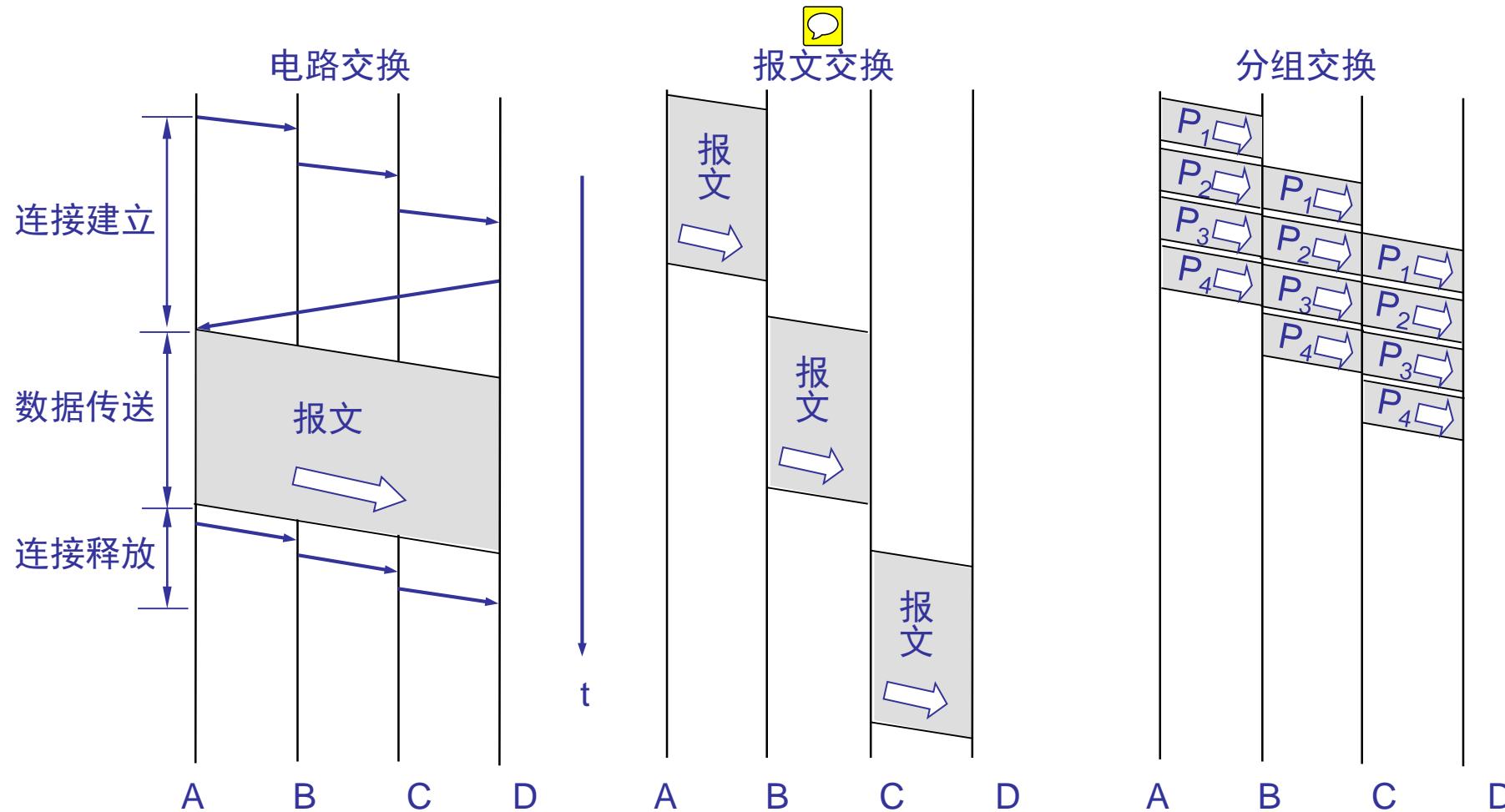
- 电路交换

- 数据传送之前需要在源节点和目的节点之间建立一条特定的电路
- 源节点通过该电路向目的节点发送比特流
- 案例：电话网

- 分组交换

- 分组：每一个数据流在给定时间内允许传送的数据块，每一分组独立转发
- 每一个节点接收到一个完整的分组后，将该分组存入临时内存，然后将分组转发至下一节点
- 案例：IP数据网

三种交换的比较

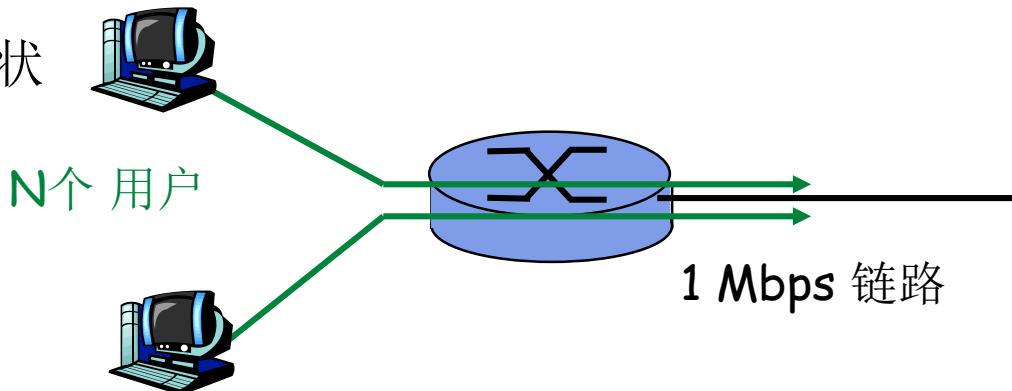


存储转发策略更加高效

分组交换 vs 电路交换

分组交换允许多个用户共享链路资源

- 1 Mb/s 链路
- 每个用户：
 - 活跃速率为100 kb/s
 - 有 10% 的时间处于活跃状态
- 电路交换：
 - 可容纳10 个用户
- 分组交换：
 - 可容纳35 个用户，同时有超过10 个活跃用户概率小于0.0004



问题: 0.0004是如何得到的?



高性价比的资源共享：小结

- 交换技术的选择：分组交换
 - 更适合计算机通信，传输突发流量的效率较高
- 复用技术的选择：统计型复用
 - 多个用户以分组为粒度共享网络资源（链路和节点）
 - 每一个交换设备可以为每一个分组调度使用与之相连的物理链路
- 挑战
 - 分组交换中间节点的存储资源耗尽时会导致网络的拥塞
 - 分组交换难以实现服务质量的保证

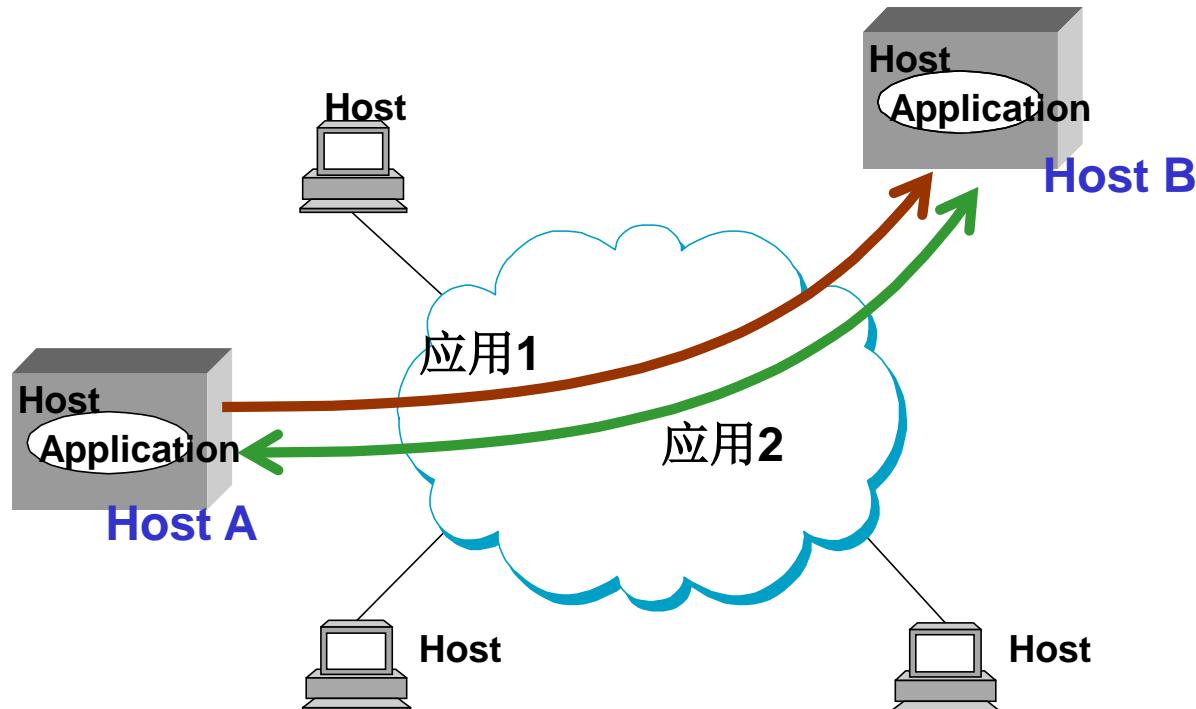


设计需求

- 可扩展的连通性
 - 提供多个计算机之间的连通性
 - 高性价比的资源共享
 - 在资源约束条件下保证其通信的效率
- ● 支持通用服务
 - 确定通用通信模式
 - 可靠性
- 可管理性

支持通用服务

- **目标1:** 网络支持各种不同的应用
- **动机1:** 简化目标, 对大多数的应用需求进行分类, 并提供相应的通用服务.



应用1: Host A send a text file in 1Mb size to Host B

应用2: Host A request streaming video from Host B



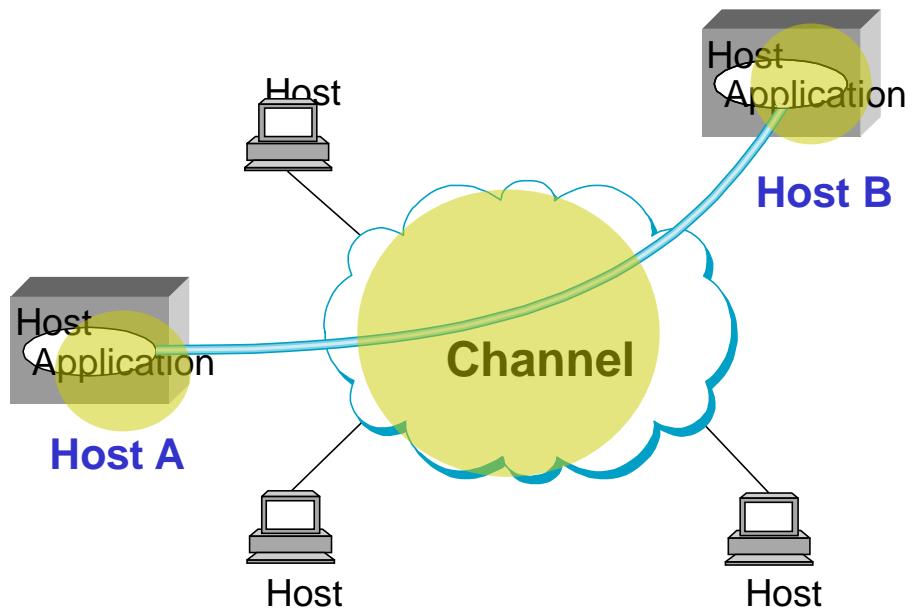
通用服务

- 什么是通用服务?
 - 一些可以被应用重复调用的构件
- 相同的通信模式
 - 相同的通信需求
 - 两种基本通信模式
 - **请求/响应:** 保证源端发送的每一个消息均可以被目的端所接收，且每一个分组仅传送一次.
 - **消息流:** 支持单向和双向传输,且支持不同的延迟特性, 需要保证传送的消息必须按序到达，支持多点播送.

通用服务

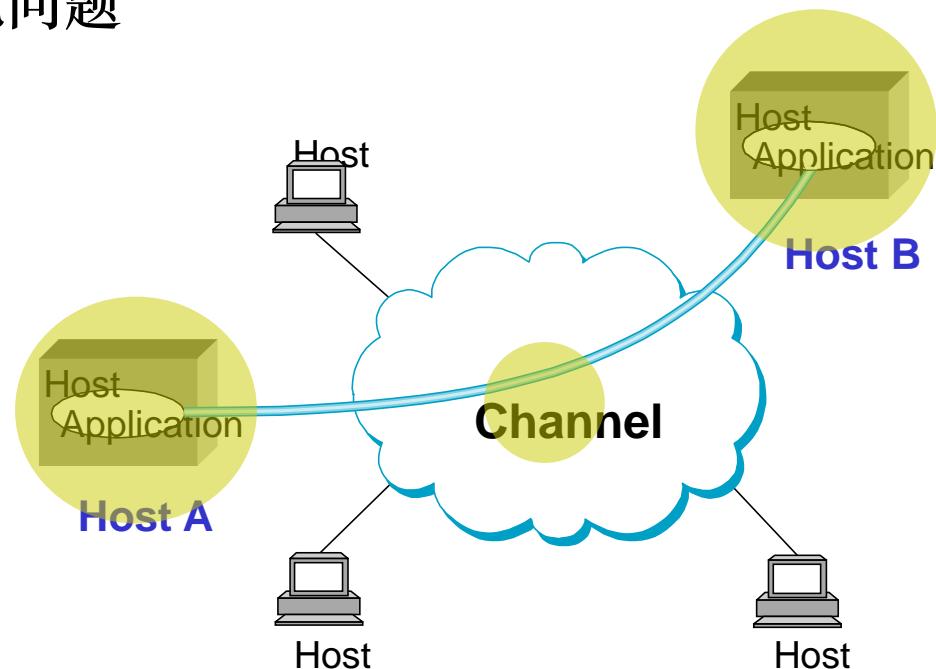


- **讨论:** 如何实现?
 - 主机与网络之间的功能分配问题



主机简单, 网络复杂

案例: 电话网络



主机复杂, 网络简单



支持通用服务

- **目标2:** 网络在现实网络条件下支持各种不同的应用
- **动机2:** 提供可靠的消息传送, 能够对故障进行分类处理

- 三类故障
 - 比特错误(比特级)
 - 外部电磁干扰等
 - 分组丢失(分组级)
 - 内存溢出, 或分组出现了不可纠正的比特错误
 - 如何区分分组是丢失还是延迟到达?
 - 链路故障/节点当机(链路/节点级)
 - 如何区分主机故障还是运行速度慢?
- 屏蔽部分故障
 - 使得网络对于应用程序而言具有更强的可靠性



小结: 支持通用服务

- 不同应用通信的需求可以映射为不同的通用服务模式
- 挑战
 - 设计通用模式以及功能的划分
 - 克服现实网络中的故障，提供各种不同级别的可靠性保证

互联网广泛应用“尽力服务”模型！

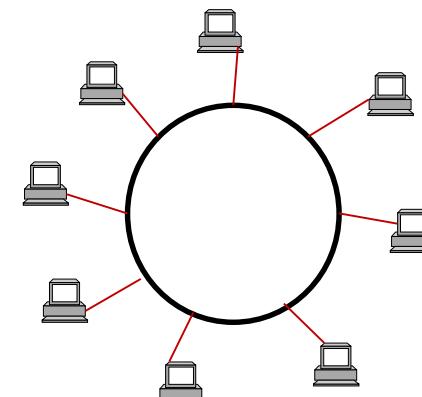
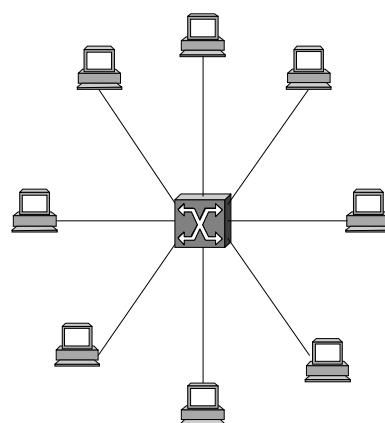


设计需求

- 可扩展的连通性
 - 提供多个计算机之间的连通性
 - 高性价比的资源共享
 - 在资源约束条件下保证其通信的效率
 - 支持通用服务
 - 确定通用通信模式
 - 可靠性
- ● 可管理性

可管理性

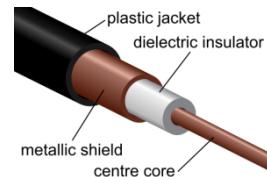
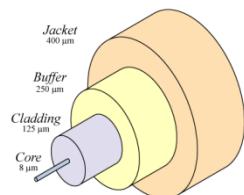
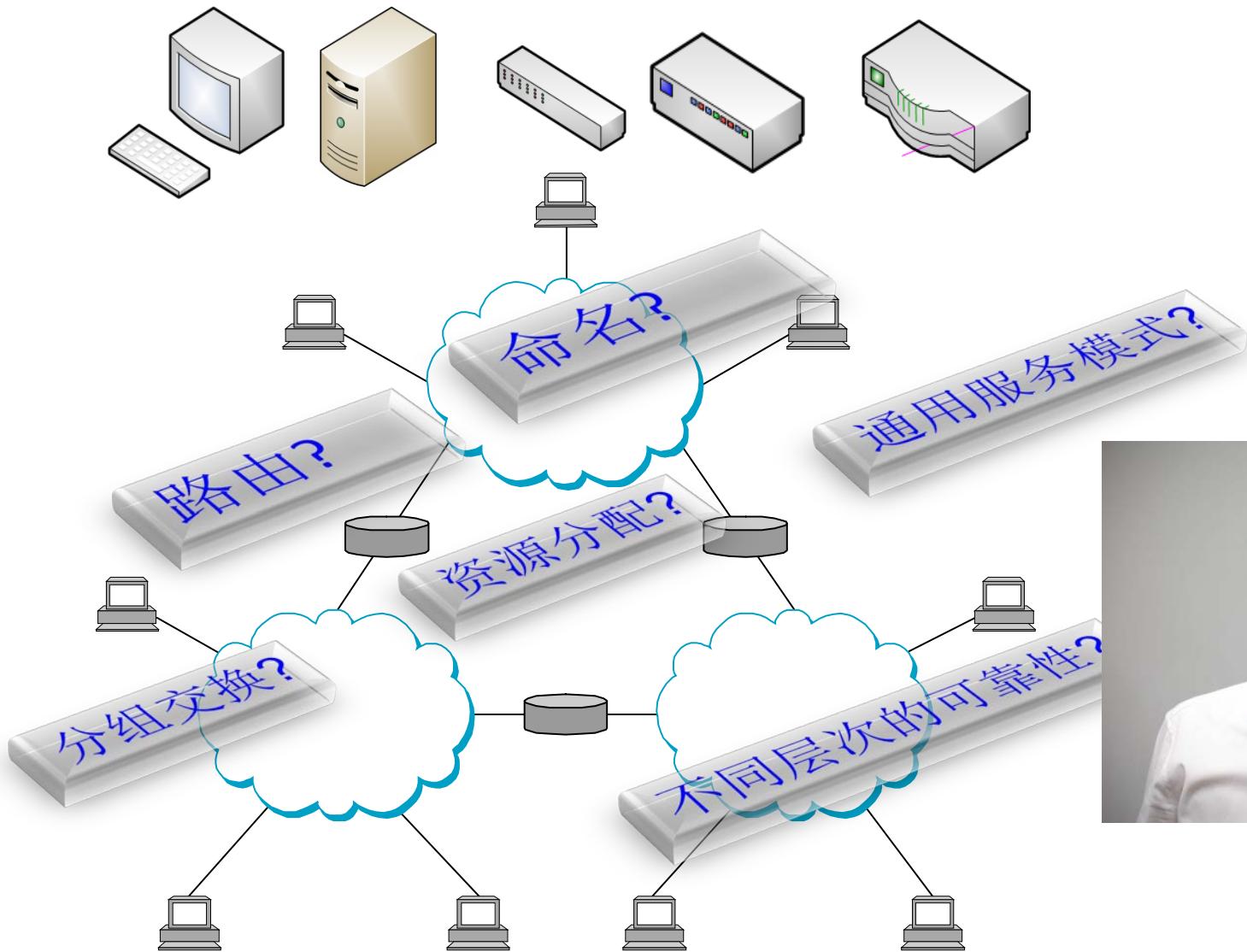
- 网络的可管理性
 - 如何应对规模的扩展?
 - 当需要扩大网络来承载更多流量时如何改变?
 - 当需要支持更多用户时如何配置或者改变?
 - 当出现问题或者性能下降时如何排除网络故障?
- 可管理性有时影响了技术竞争的结果
 - 星形拓扑的以太网 vs. 环形拓扑的令牌环网

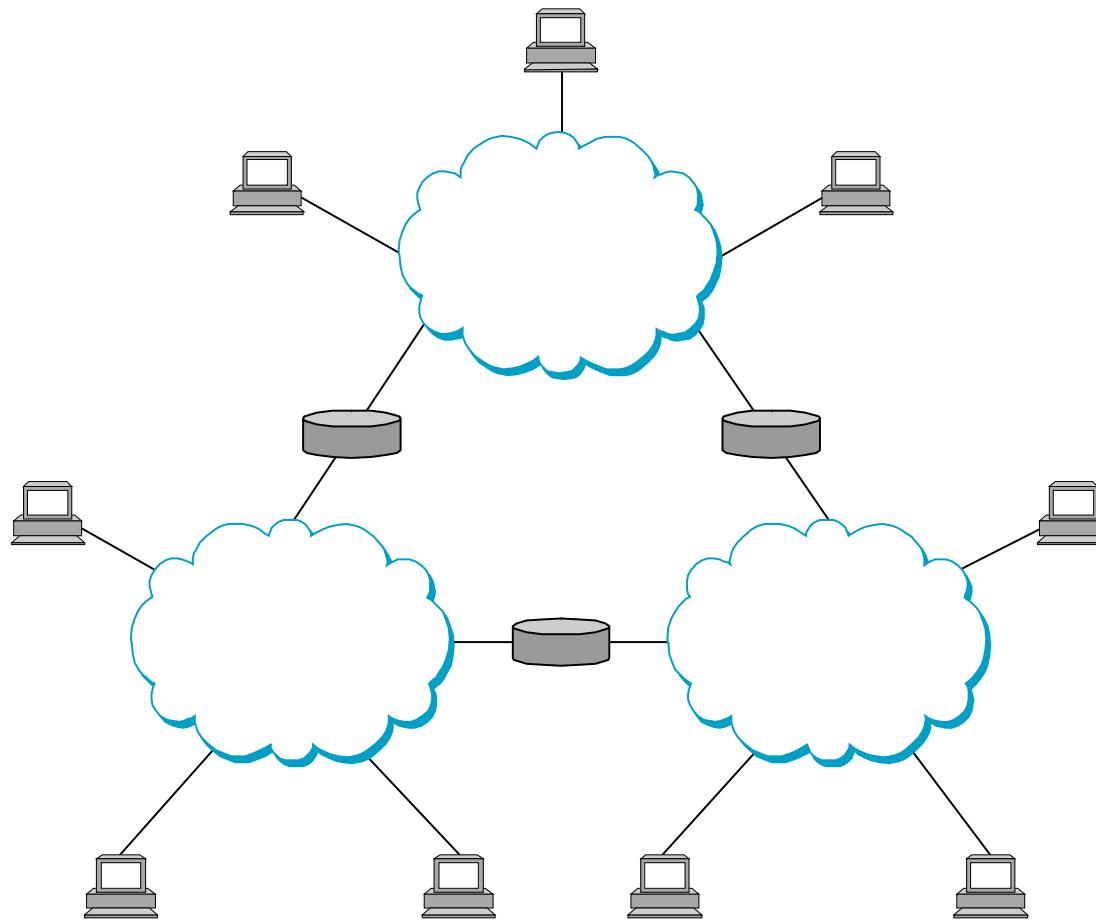




小结: 设计需求

- 从需求出发, 总结出网络设计的关键性问题
 - 连通性
 - 命名: 为每一个节点定义一个地址
 - 路由: 将消息沿正确的路径转发至目的节点
 - 高性价比的资源共享
 - 联网: 分组交换网
 - 资源分配: 保证流的公平性, 拥塞控制
 - 支持通用服务
 - 设计通用模式: 传输协议
 - 质量保证: 针对不同级别的故障提供可靠性保证
 - 可管理性





- 为了应对网络设计的复杂性，网络设计人员需要开发通用的设计蓝图
 - 即 网络体系架构 *network architecture*
 - 用来指导不同网络的设计和实现的细节



提纲

- 背景
 - 核心问题: 构造一个网络
- 设计需求
 - 可扩展的连通性
 - 高性价比的资源共享
 - 支持通用服务
 - 可管理性
- 网络架构
 - 分层与协议
 - 互联网体系结构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- 总结





网络体系结构

- 网络系统的本质
 - 异构性
 - 链路: 有线, 无线, 光
 - 交换: 电路, 数据报, 虚电路
 - 主机: PC, 服务器, ...
 - 不同的应用
 - FTP, Web, Email, media, ...
- 为了有助于处理网络的**复杂性**, 网络设计者制定了通用的蓝图
 - 通常称为 **网络体系结构**
 - 用以指导网络的设计与实现



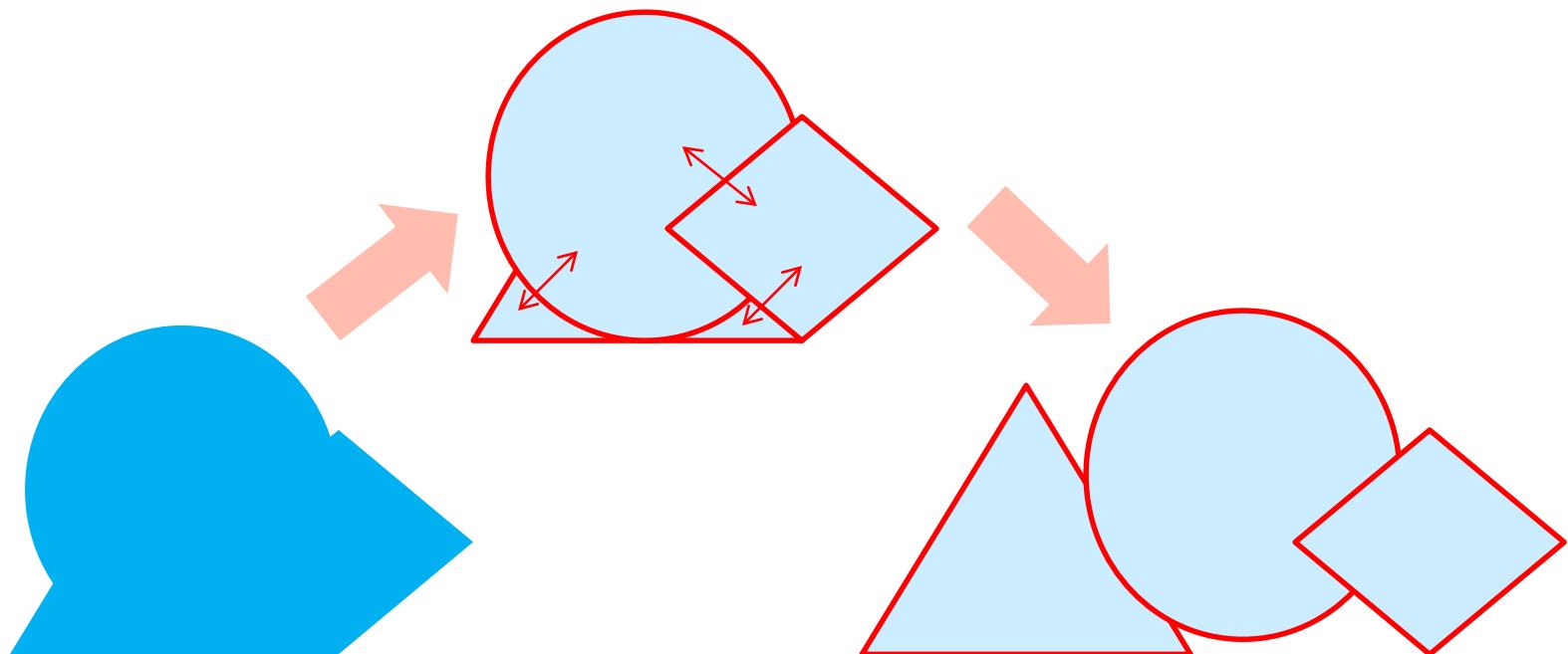
网络体系结构

- 体系结构(Architecture)不是实现(Implementation)
 - 体系结构是抽象的，而实现则是具体的
- 体系结构关注如何“组织”实现
- 体系结构是网络的模块化设计
 - 不同的模块如何组织?
 - 不同的模块之间如何交互?



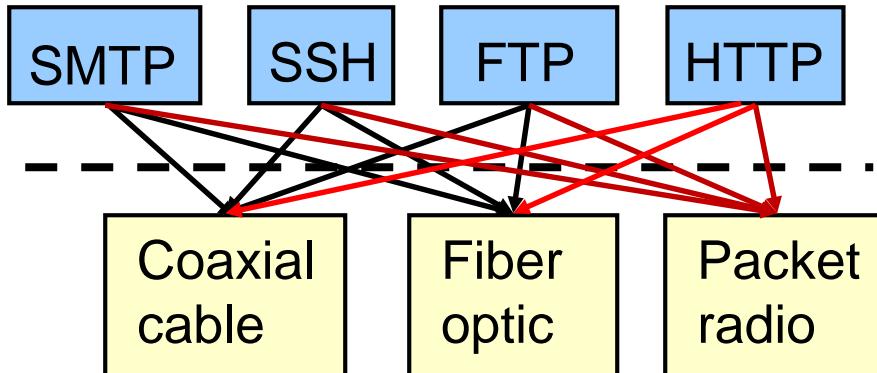
模块化设计方法

- 描述
 - 将整个系统划分为许多较小的更为简单的不同功能模块, 模块之间通过接口之间相互关联



为何需要分层?

应用程序

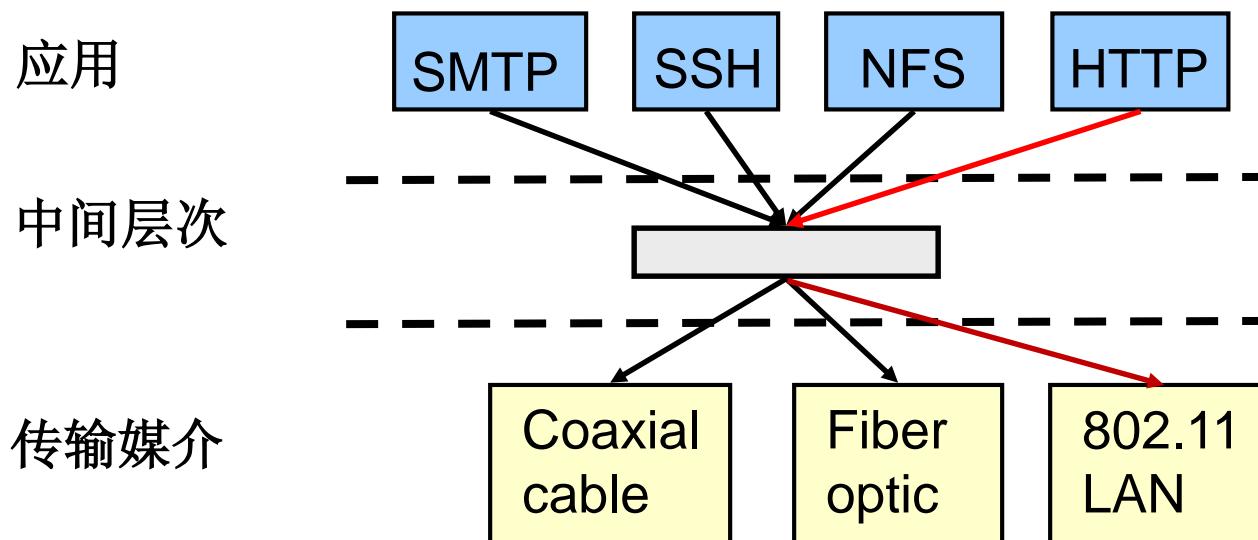


传输媒质

- 新的应用需要支持现有的所有传输媒质
 - 增加一个新的应用需要 $O(m)$ 次工作, m = 传输媒质的数量
- 新的传输媒质需要对现有的所有应用进行修改
 - 增加一种新的传输媒质需要 $O(a)$ 次工作, a = 应用的数量
- 整个系统需要 $O(ma)$ 次工作 → 工作繁杂, 工作量大

为何需要分层?

- 解决方法: 增加一个中间层提供各种不同网络技术的抽象
 - 增加应用或传输媒质需要 $O(1)$ 次工作
 - 增加中间层次是计算机科学领域的一种常用技术





分层的架构设计方法

- 分层(Layering)
 - 模块化的一种特殊形式
 - 将一个网络系统看作一系列实体的集合，一个实体提供的服务完全基于更底层实体所提供的服务
- 优点
 - 模块化、重用、功能抽象
- 缺点
 - 效率可能较低



分层结构中的一些基本概念

- 实体(Entity)
 - 定义: 构成网络系统各个层次的抽象对象
 - 任何可发送或接收信息的硬件或软件进程, 多数情况下指某个特定的模块
 - 向高层提供服务
 - 通过调用底层实体提供的服务与远程计算机的对等实体进行通信



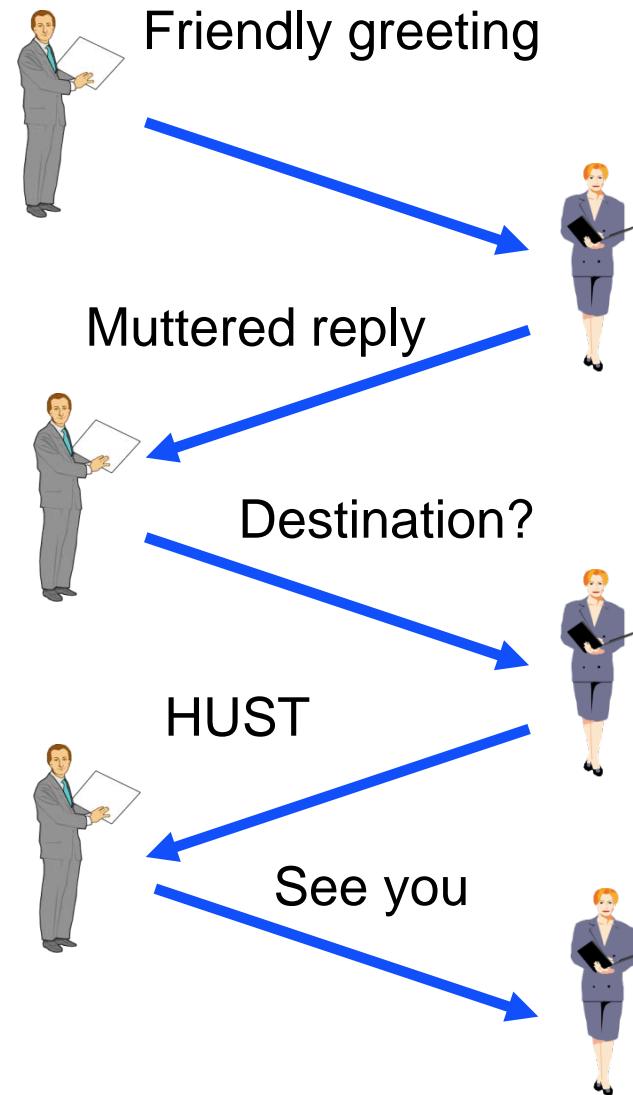
分层结构中的一些基本概念

● 协议(Protocol)

- 定义: 两个或多个网络**对等实体**之间通信所需遵从的特定规则
- 端到端通信: 协议的主要部分
- 向上层实体提供服务

● 协议三要素

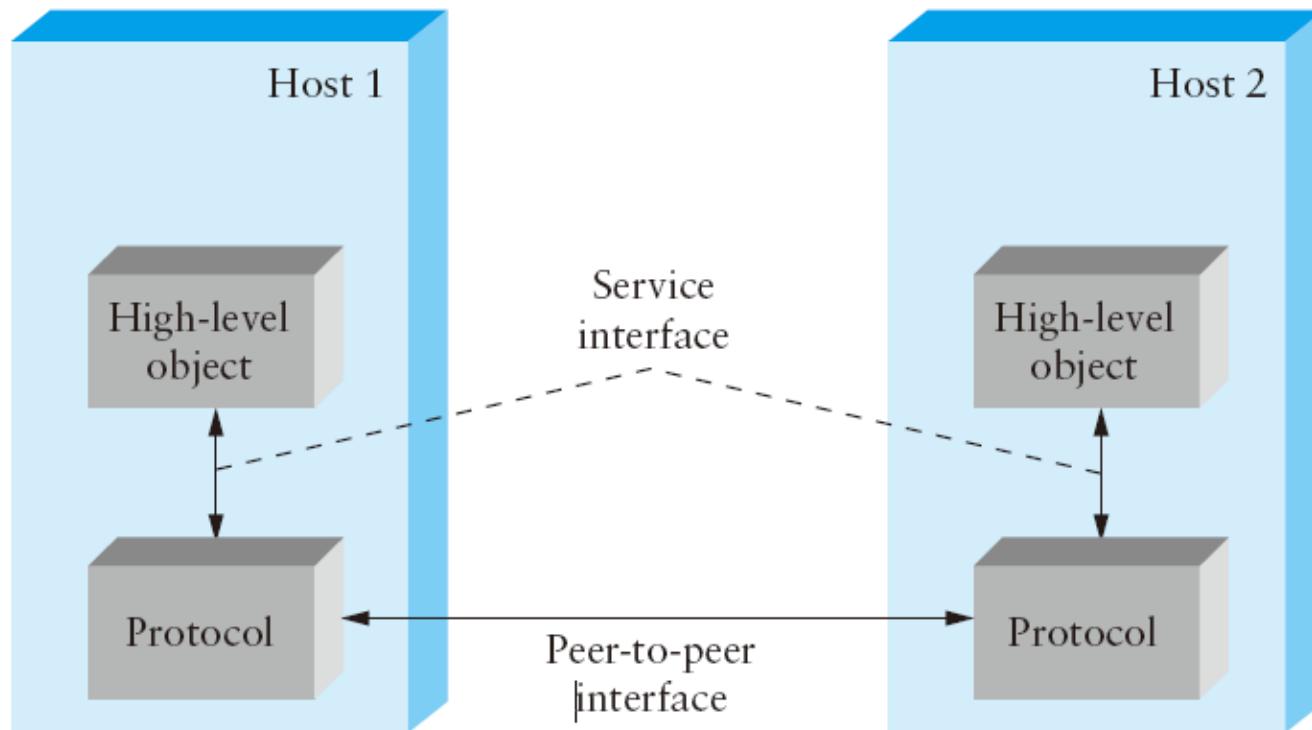
- **语法** 数据与控制信息的结构或格式。
- **语义** 需要发出何种控制信息, 完成何种动作以及做出何种响应。
- **同步** 事件实现顺序的详细说明。



分层结构中的一些基本概念

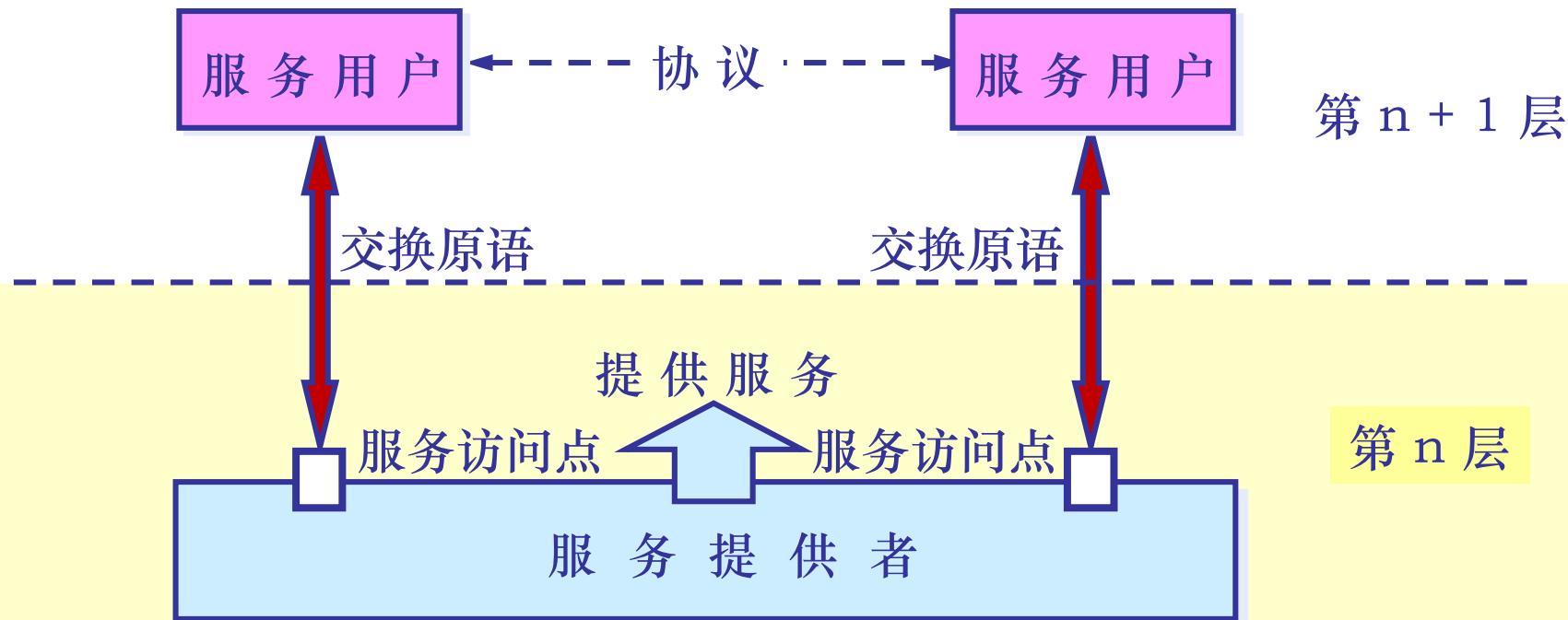
● 实体的接口(Interface)

- 垂直(向上): 服务接口
- 水平(对端): 与远程计算机对等实体的对等接口

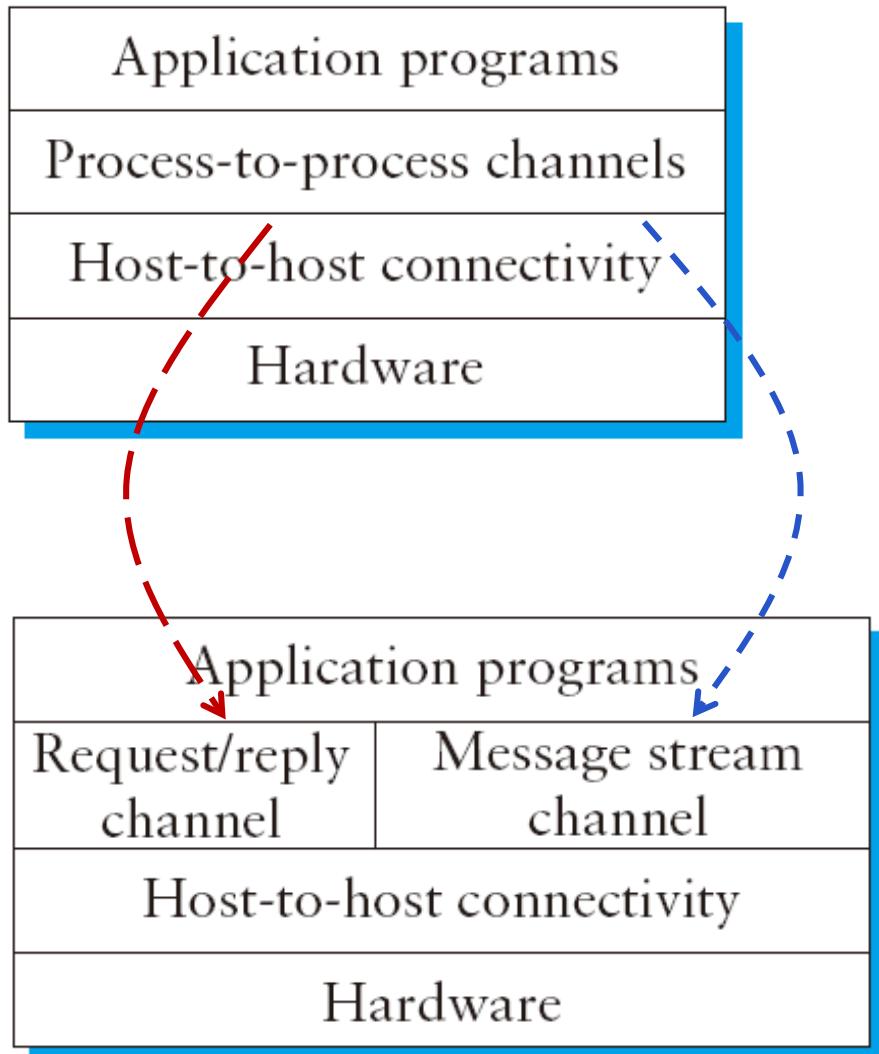


分层结构中的一些基本概念

- 实体、协议、服务



案例：一个分层的网络

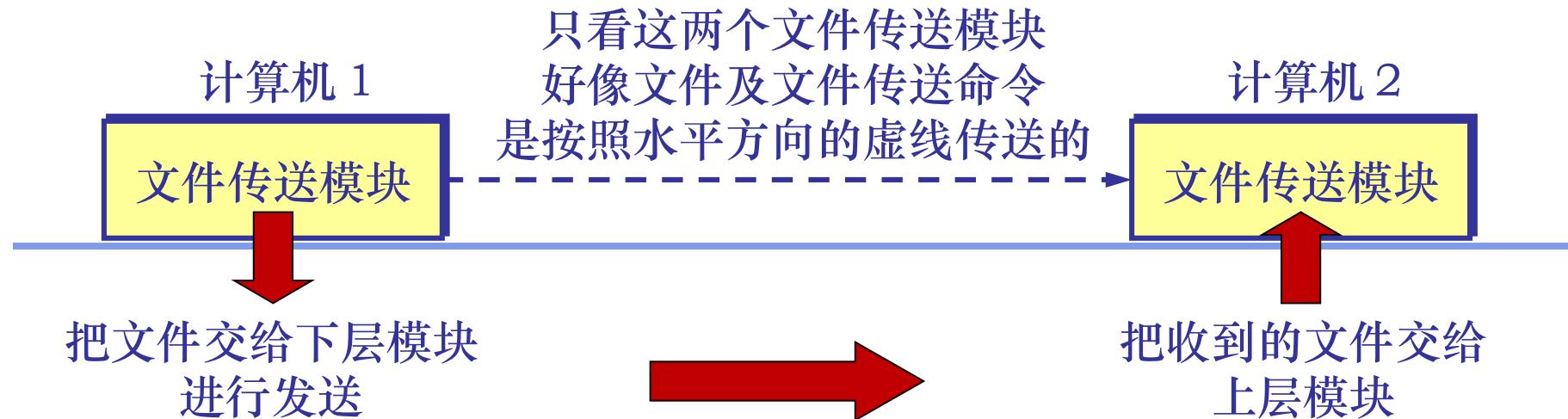


同一层次内可以具有不同的实现实体



案例：分层和协议

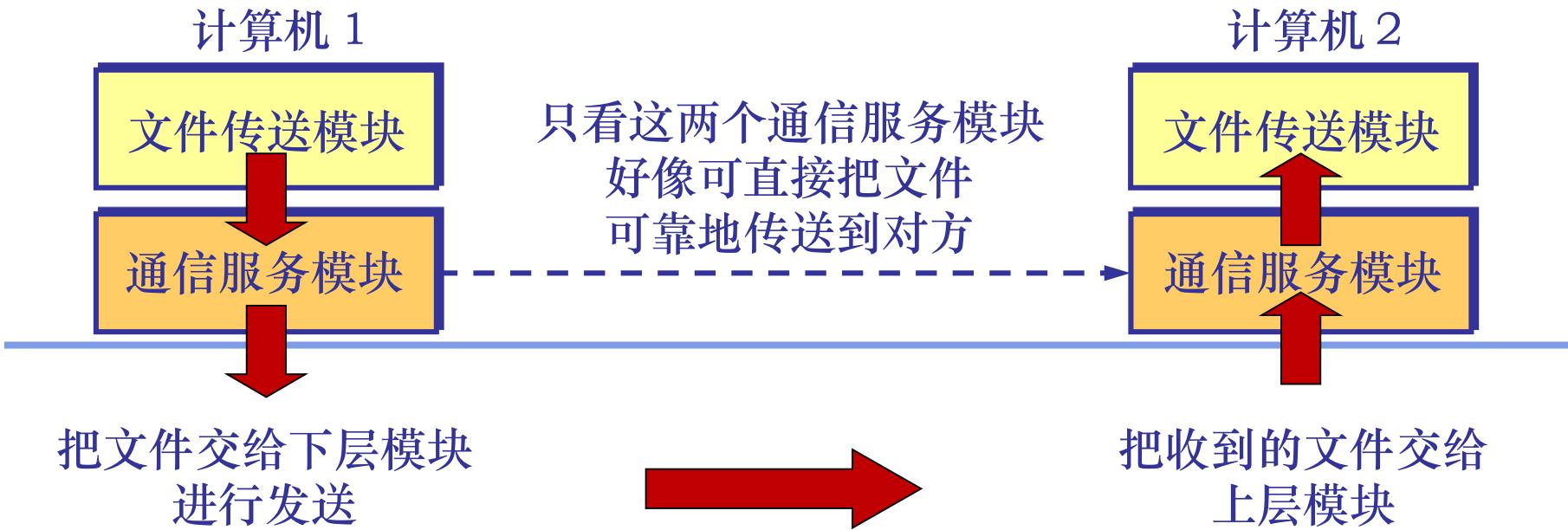
两个计算机交换文件





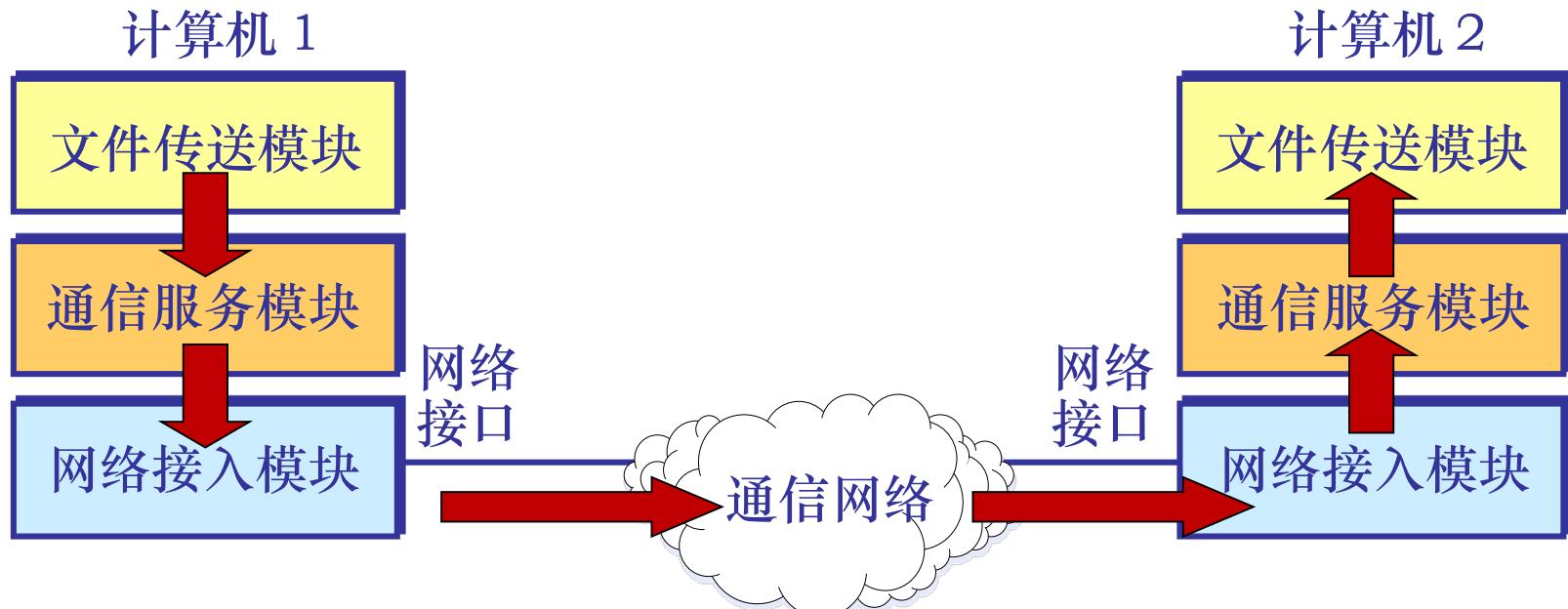
案例：分层和协议

再设计一个通信服务模块



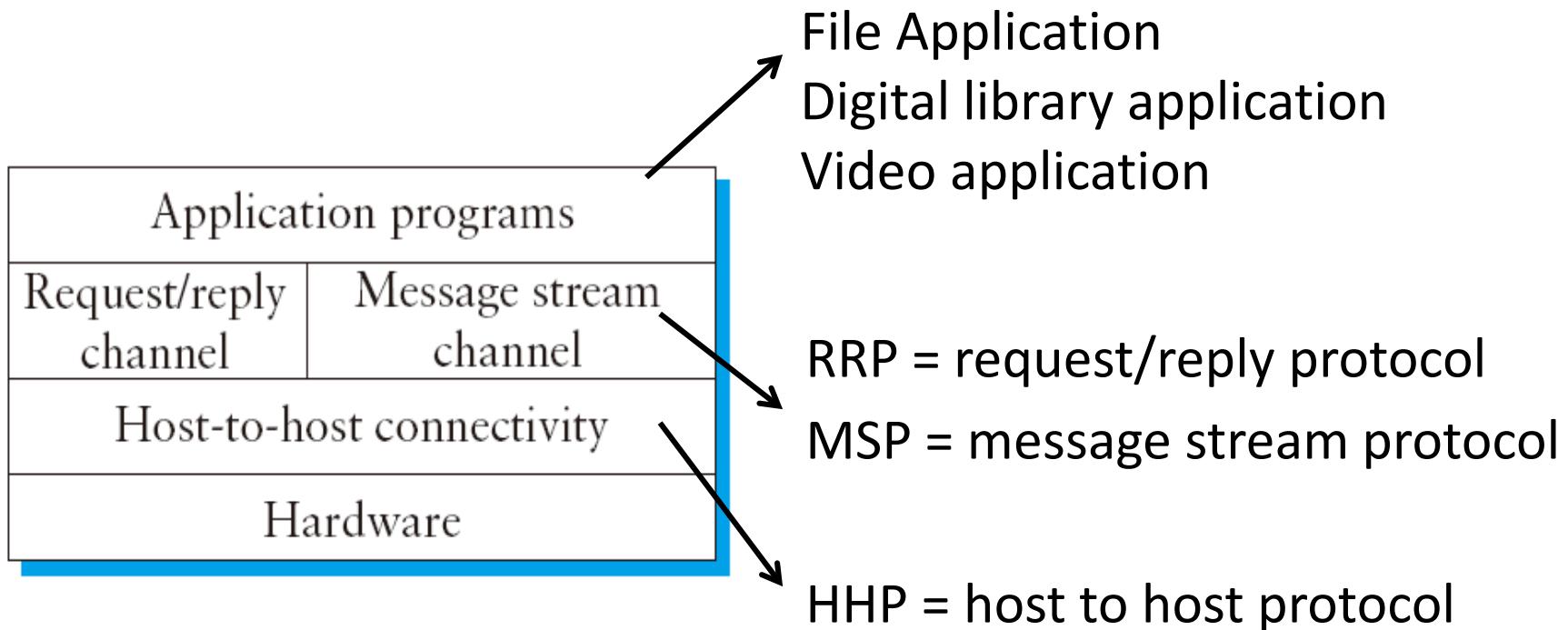
案例：分层和协议

再设计一个网络接入模块

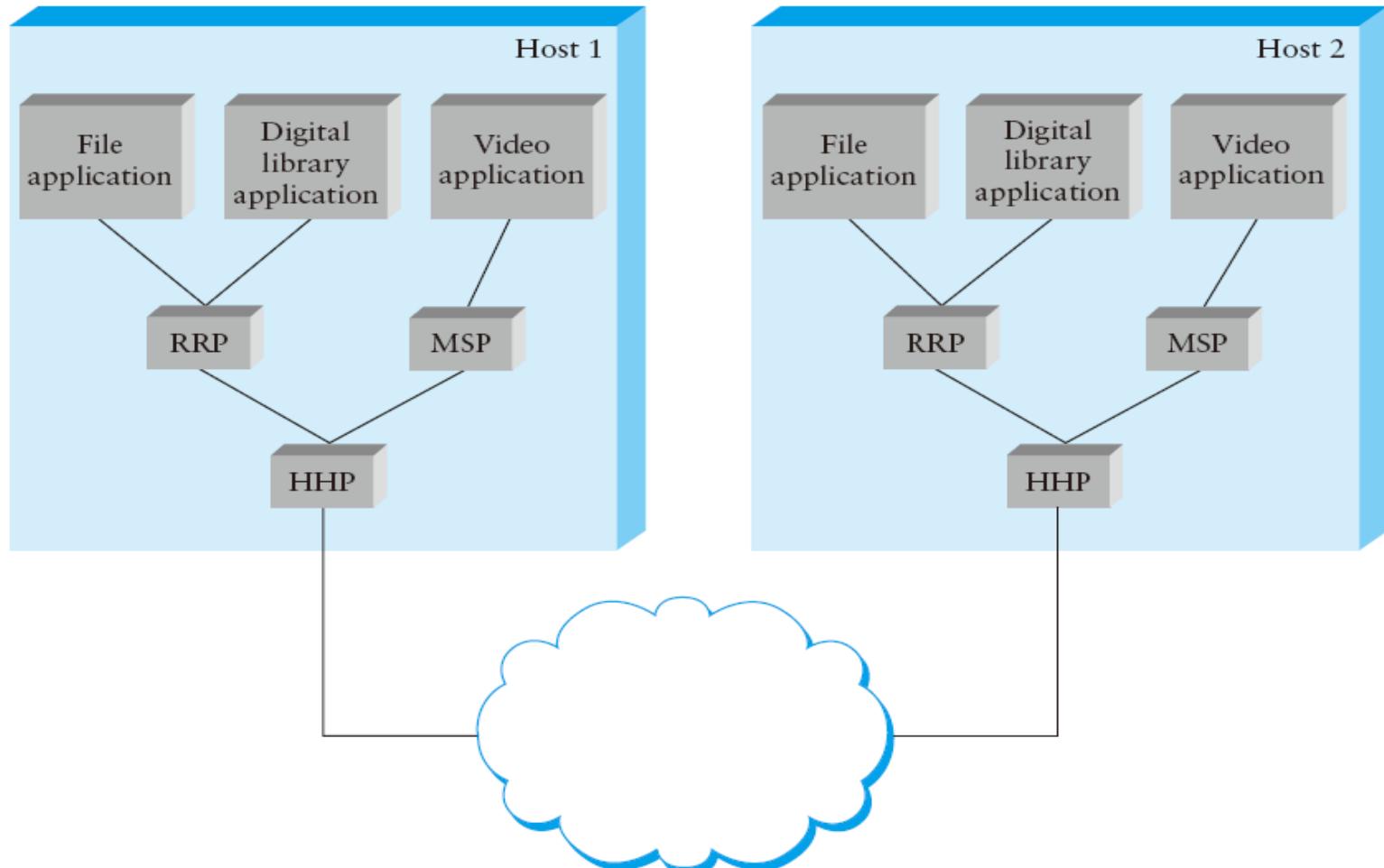


网络接入模块负责做与网络接口细节有关的工作
例如，规定传输的帧格式，帧的最大长度等。

案例：一个分层的网络



分层的协议图



- 问题1. 如何在不同层次之间共享信息？
问题2. 如何在同一层内共享信息？



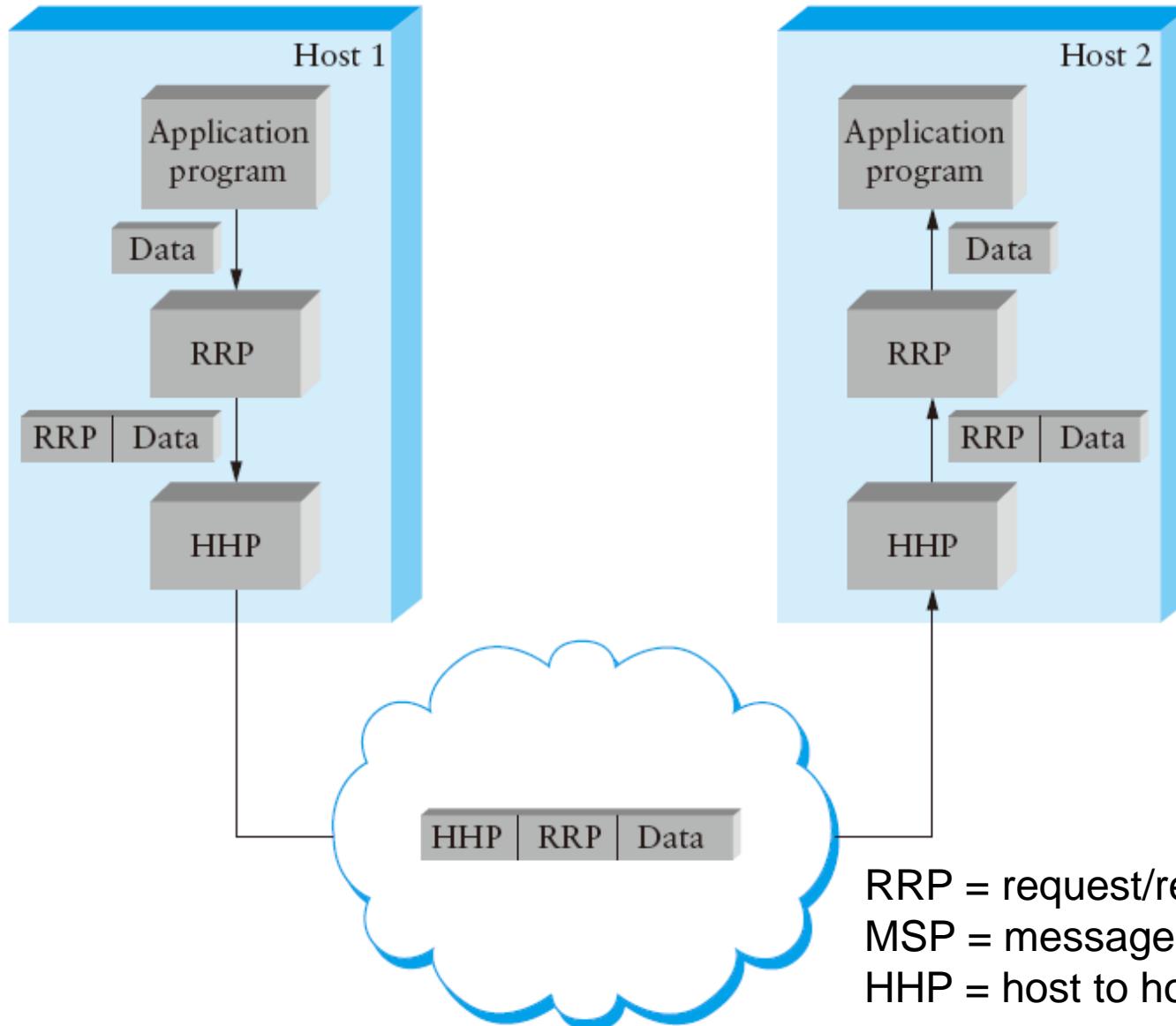
解决方案1. 封装(Encapsulation)

- 为什么要封装?
 - 通过封装, 协议实体可以在分组中携带信息通告对等实体如何处理收到的分组
- 示例



- 首部/尾部的加载及分离
- 透明传输
 - 某些特定应用需要进行分组的压缩/解压缩, 加密/解密

解决方案1. 封装

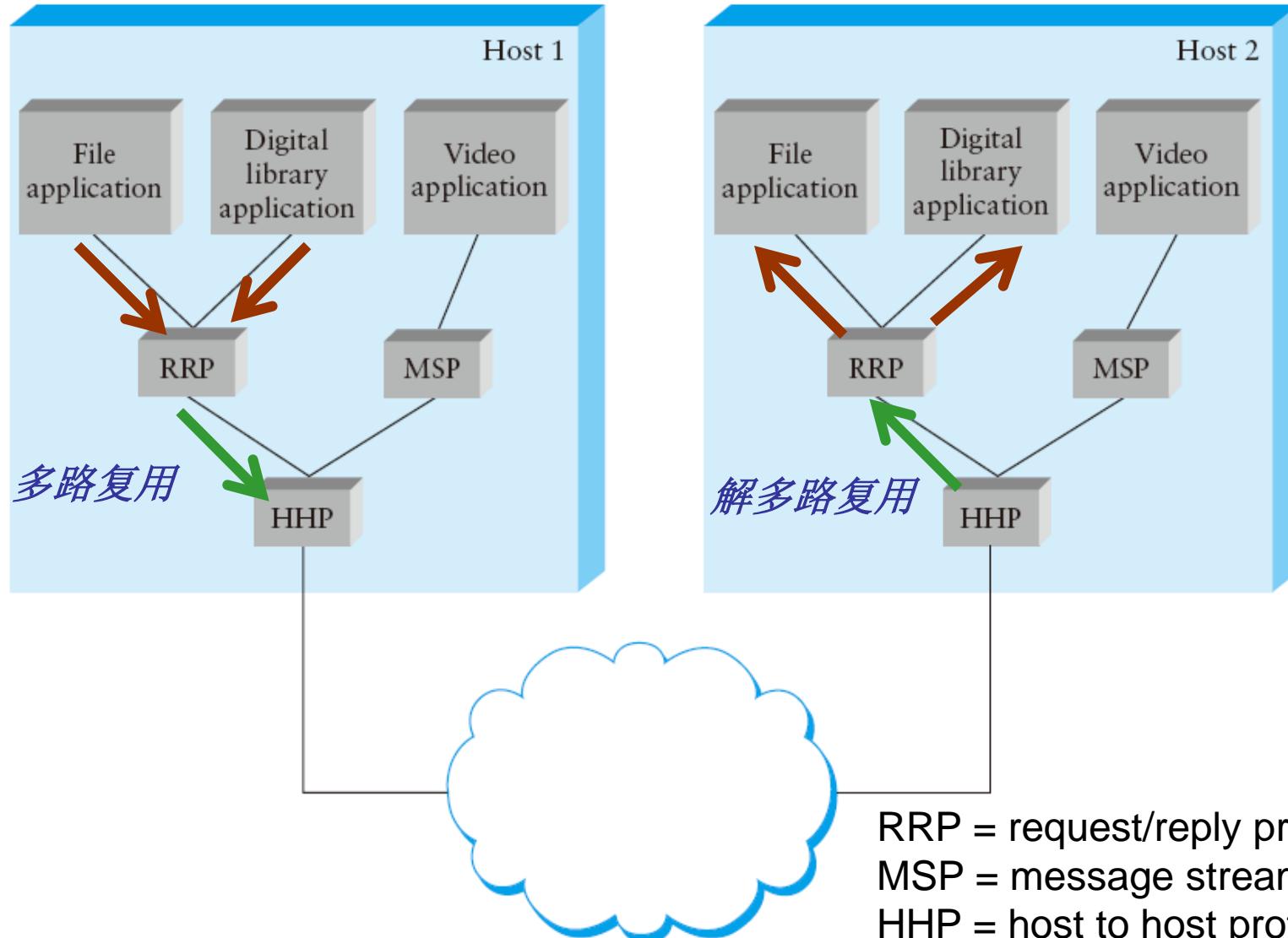




解决方案2. 复用与解复用

- 复用(Multiplexing)来自源节点不同高层实体的数据，将数据解复用(Demultiplexing)至目的节点对应的高层实体
- 需要能够区别数据所属的高层实体
 - 利用解多路复用密钥
- 各种不同类型的解多路复用密钥

解决方案2. 复用与解复用



复用与解复用

在接收端主机解复用:

把收到的报文传送到正确的**socket**

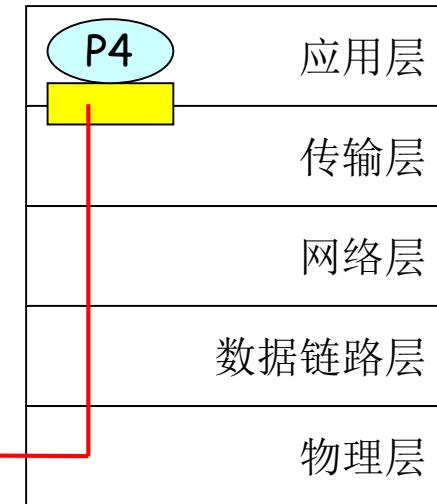
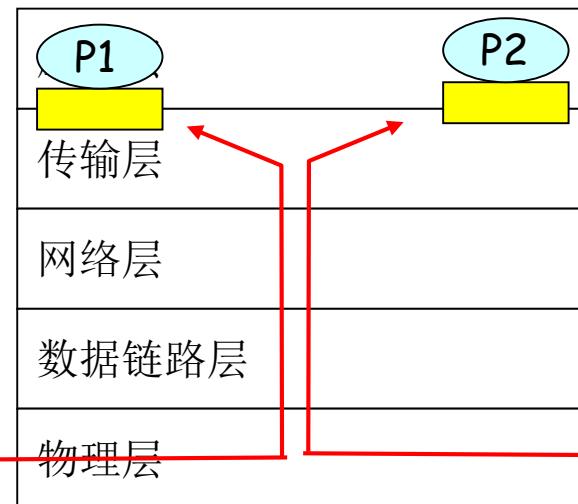
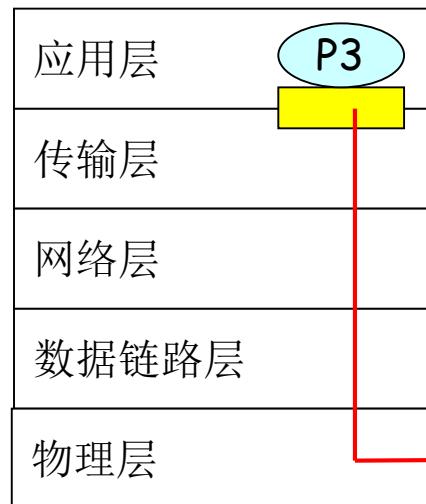
= socket



= 进程

在发送端主机采用复用:

从多个**socket**得到数据，并添加头部信息（后面的分用会用到）



主机 1

主机 2

主机 3



分层的设计方法：小结

- 分层的方法有助于厘清复杂系统各组件的交互
- 分层的实现：封装、复用与解复用
- 沿着分层的系统架构设计思路，出现ISO/OSI协议参考模型
 - ISO: International Standard Organization 国际标准化组织
 - OSI: Open Systems Interconnection 开放系统互联

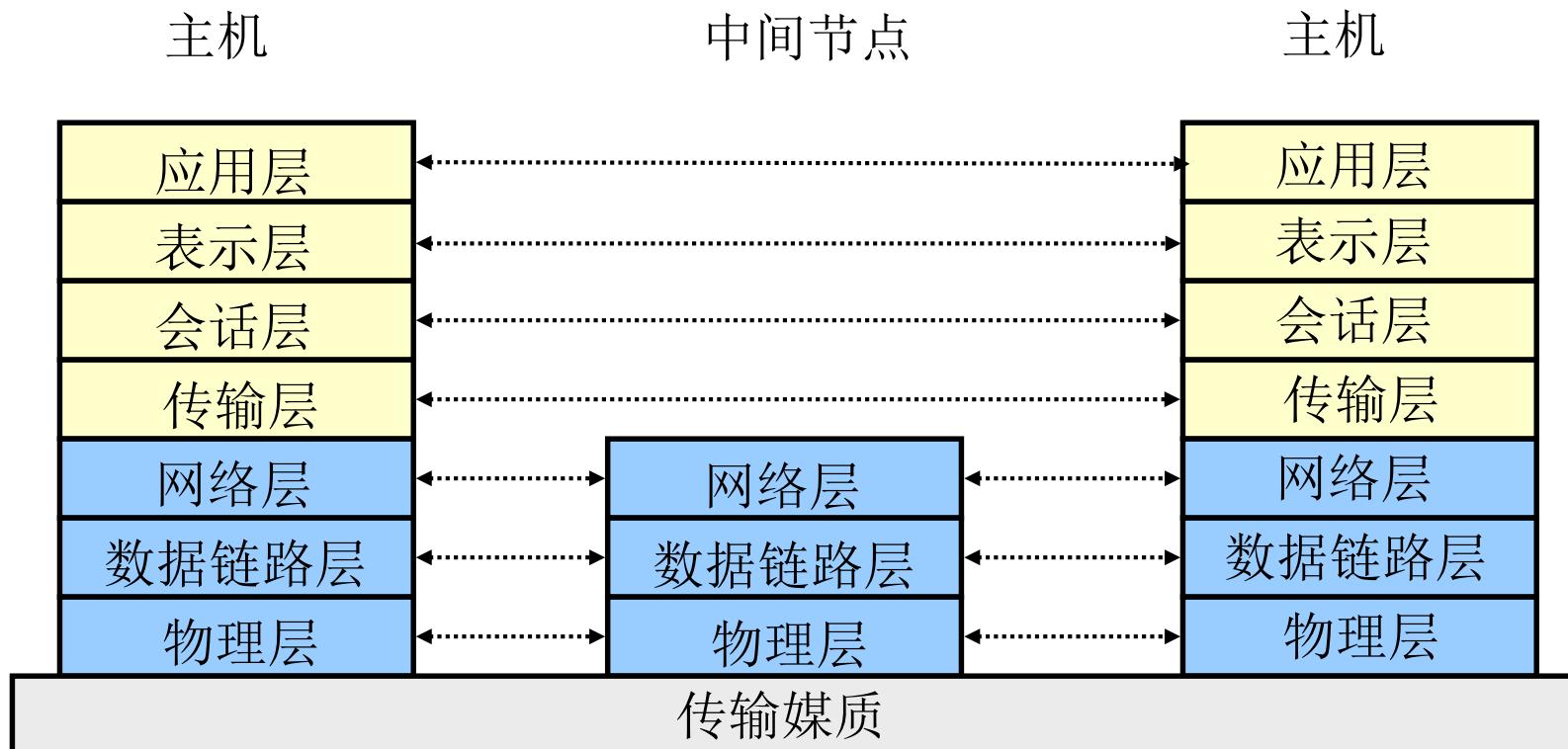


OSI 体系结构

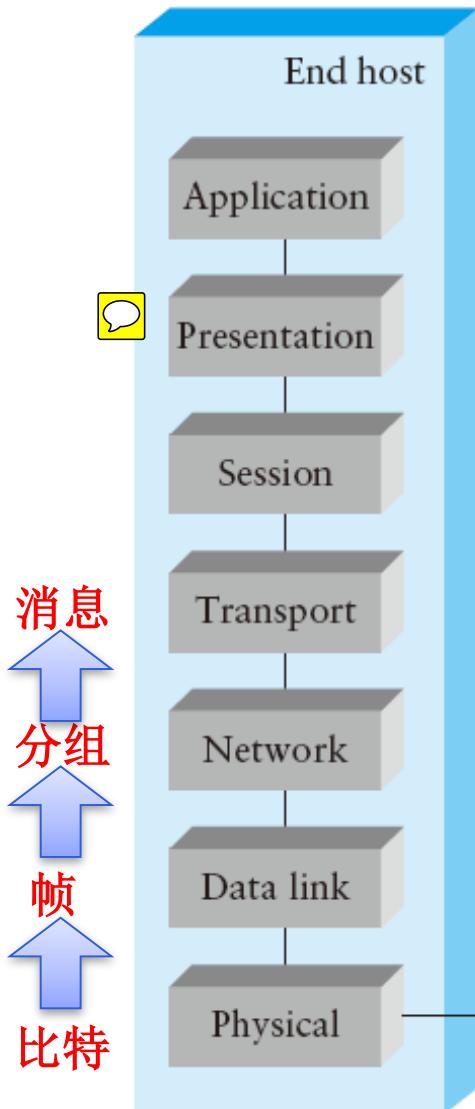
- OSI 参考模型
 - OSI – 开放式系统互联
 - 由 ISO (国际标准化组织) 制定
 - 最早开始于1978
 - 目标: 通用的开放式标准
 - 仅仅是参考模型
 - 示例: “X dot” 系列: X.25, X.400, X.500



OSI七层模型



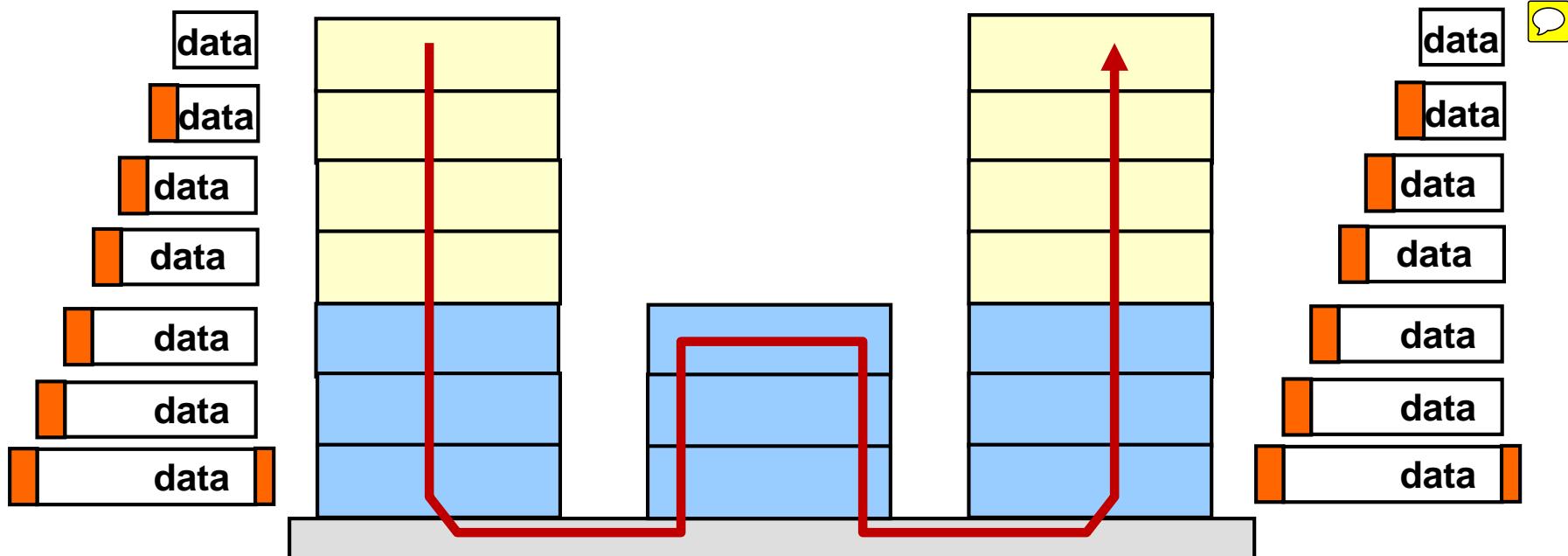
OSI七层模型各层功能定义



- › 应用层
 - 向终端用户提供各类应用服务, 例如: ftp, telnet
- › 表示层
 - 数据结构表示, 数据格式转换(加密、压缩)
- › 会话层
 - 提供会话管理, 接入控制, 数据传输同步等
- › 传输层
 - 实现终端进程之间的逻辑信道
- › 网络层
 - 处理分组交换网络中的路由选择
- › 数据链路层
 - 收集比特流组合成帧
- › 物理层 
 - 处理通信链路上的原始比特流传输

OSI七层模型

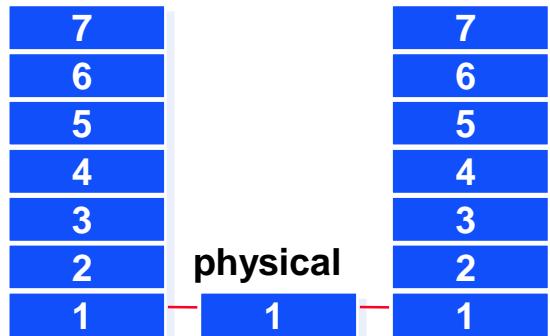
- 一个层只能与临近一层进行交互
 - 否则称为跨层(cross-layer)
- 每个层次可以在数据的前后追加本层附加的数据



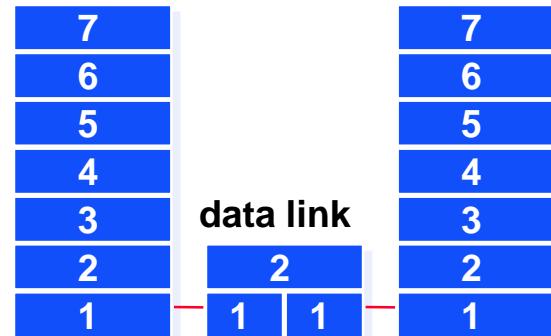
OSI模型的应用



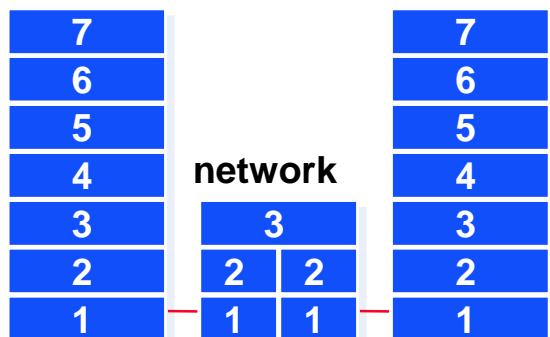
描述不同网络设备的功能



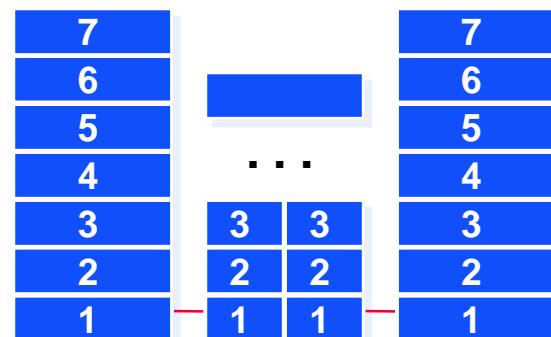
repeater



bridge
(e.g. 802 MAC)

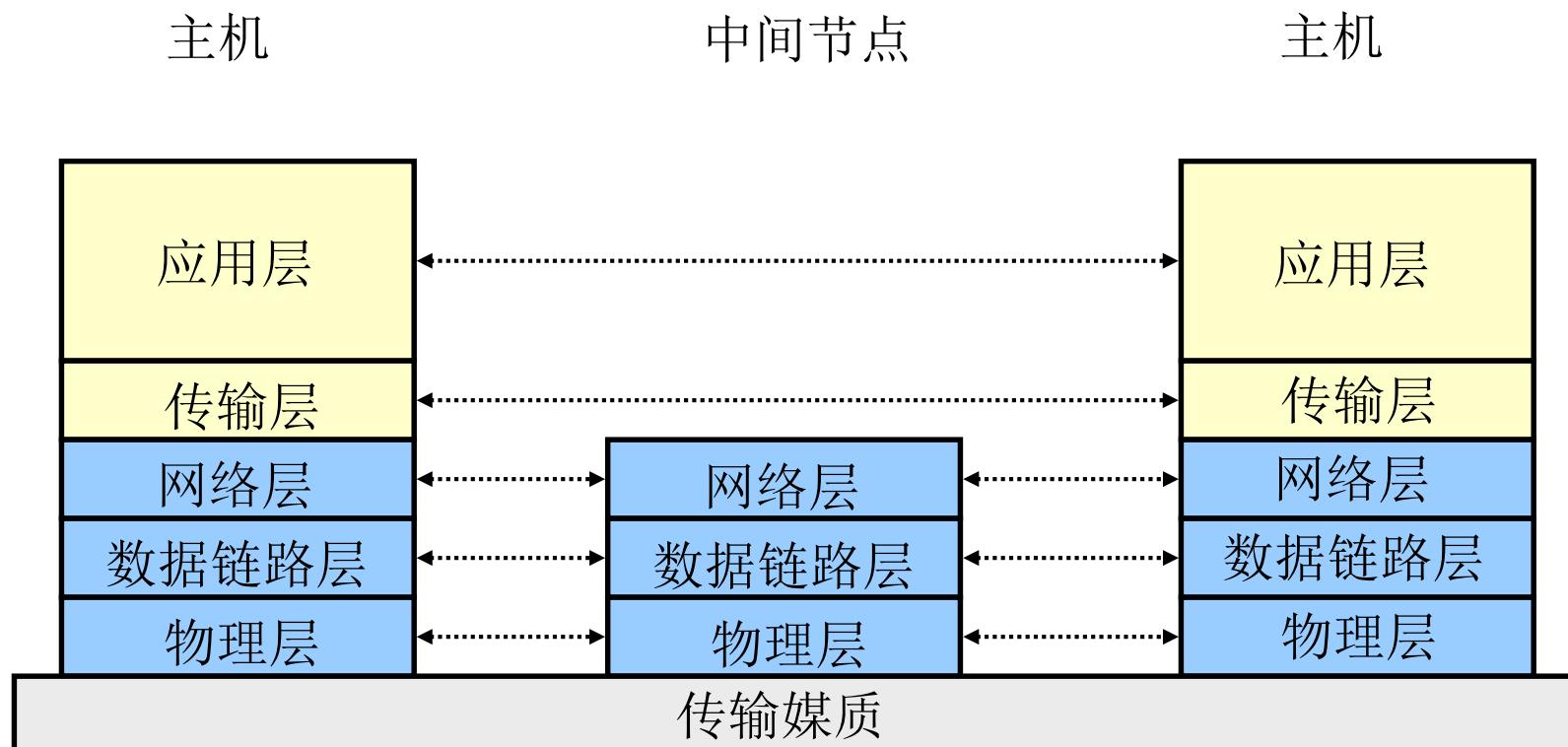


router



gateway

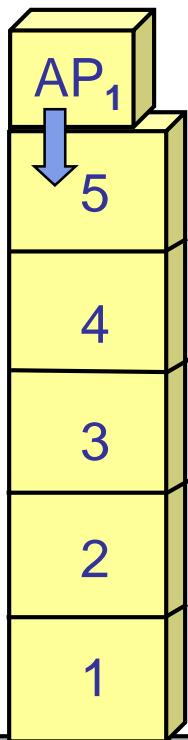
网络体系架构的五层模型



七层模型的上三层中的会话层和表示层在实际中应用较少，因此
网络研究和网络工程中常用上述的五层模型

案例：计算机 1 向计算机 2 发送数据

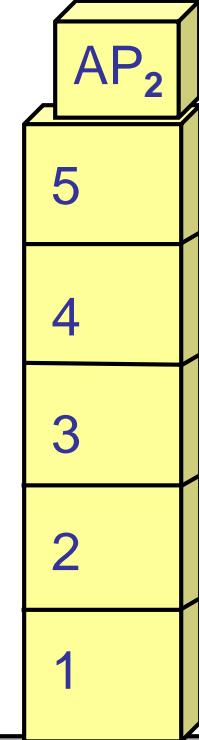
计算机 1



应用进程数据先传送到应用层

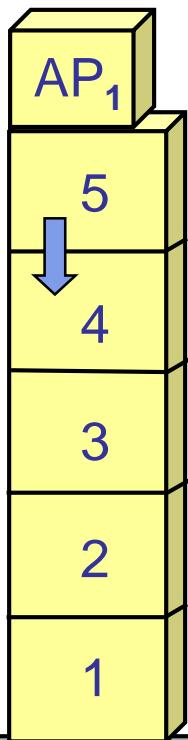
加上应用层首部，成为应用层 PDU

计算机 2



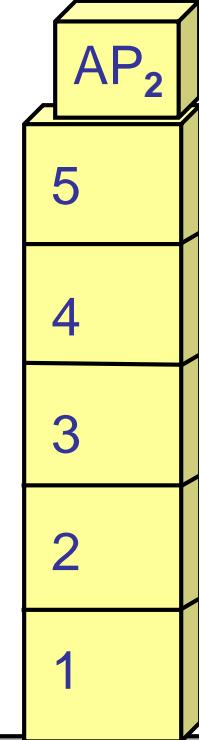
案例:计算机 1 向计算机 2 发送数据

计算机 1



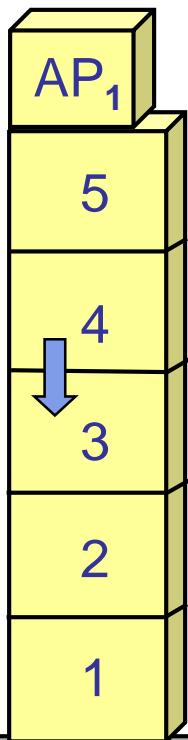
应用层 PDU 再传送到运输层
加上运输层首部，成为运输层报文

计算机 2



案例:计算机1 向计算机2 发送数据

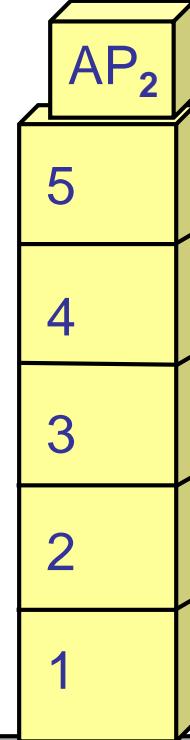
计算机1



运输层报文再传送到网络层

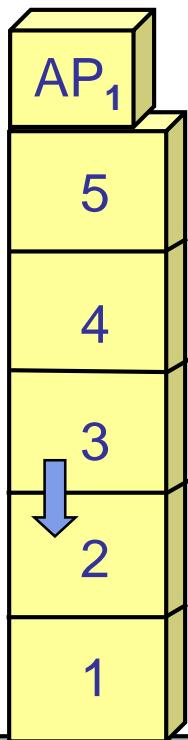
加上网络层首部，成为 IP 数据报（或分组）

计算机2

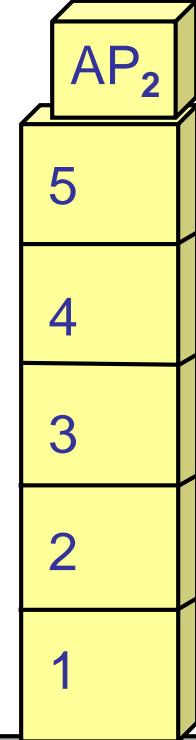


案例:计算机1 向计算机2 发送数据

计算机1



计算机2

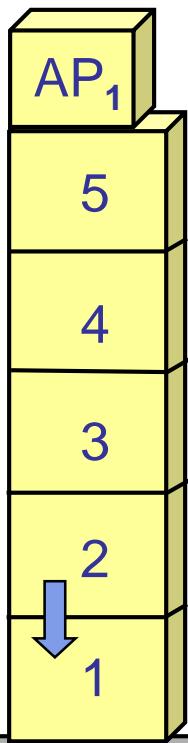


IP 数据报再传送到数据链路层

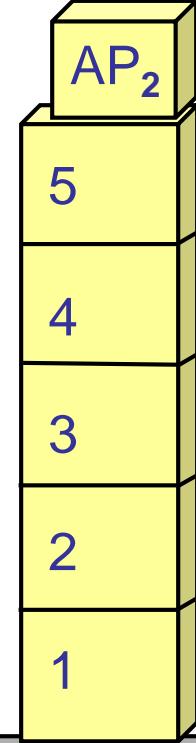
加上链路层首部和尾部，成为数据链路层帧

案例:计算机1 向计算机2 发送数据

计算机1



计算机2

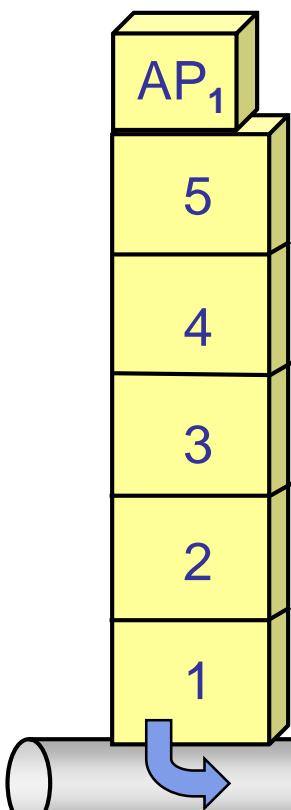


数据链路层帧再传送到物理层

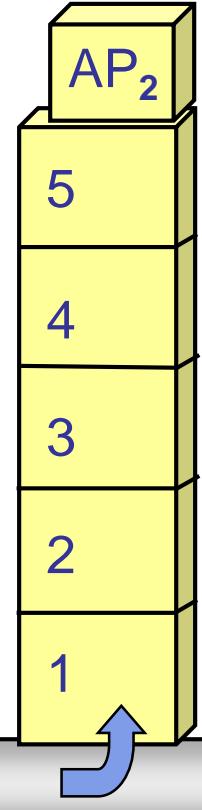
最下面的物理层把比特流传送到物理媒体

案例:计算机1 向计算机2 发送数据

计算机1



计算机2

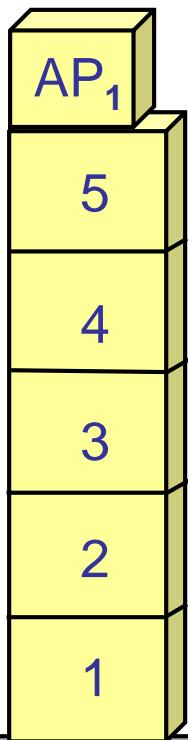


电信号（或光信号）在物理媒体中传播
从发送端物理层传送到接收端物理层

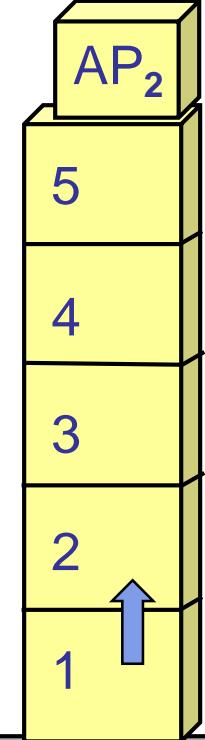
物理传输媒体

案例:计算机 1 向计算机 2 发送数据

计算机 1



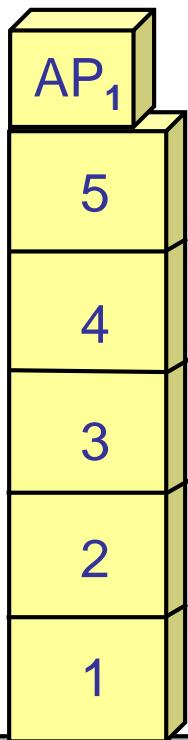
计算机 2



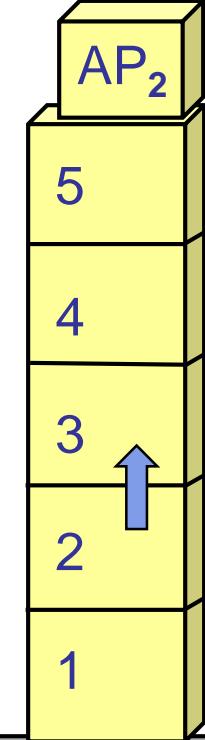
物理层接收到比特流，上交给数据链路层

案例:计算机1 向计算机2 发送数据

计算机1



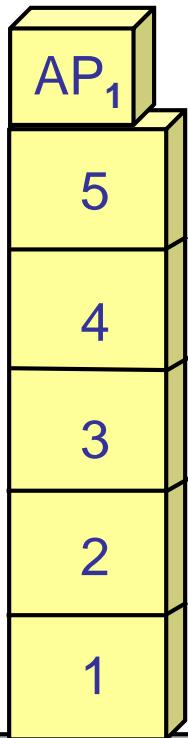
计算机2



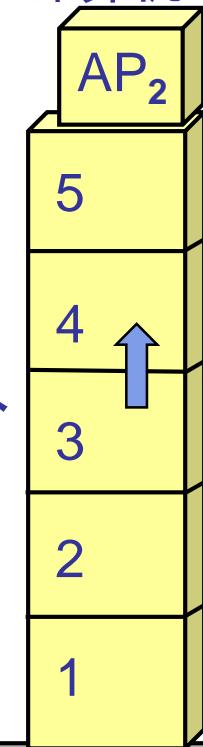
数据链路层剥去帧首部和帧尾部
取出数据部分，上交给网络层

案例:计算机1 向计算机2 发送数据

计算机1



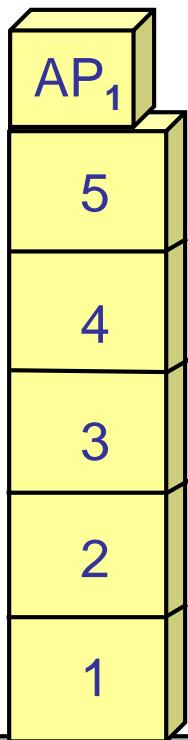
计算机2



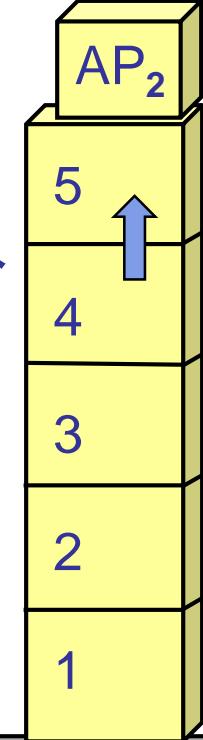
网络层剥去首部，取出数据部分
上交给运输层

案例:计算机1 向计算机2 发送数据

计算机1



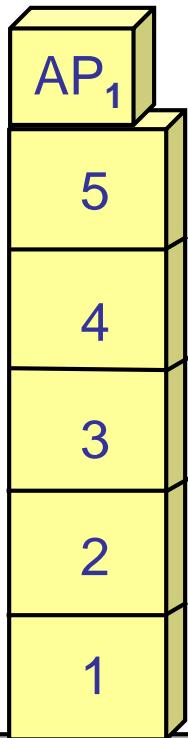
计算机2



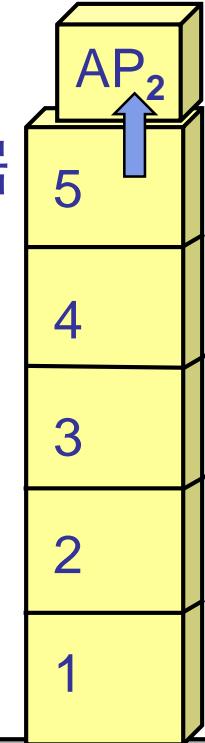
运输层剥去首部，取出数据部分
上交给应用层

案例:计算机1 向计算机2 发送数据

计算机1



计算机2

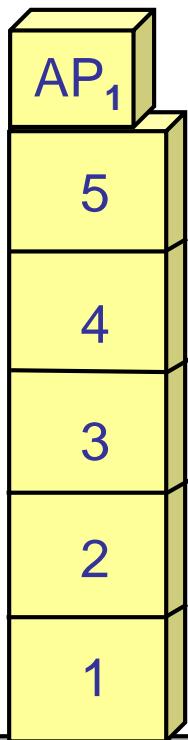


应用层剥去头部，取出应用程序数据
上交给应用进程



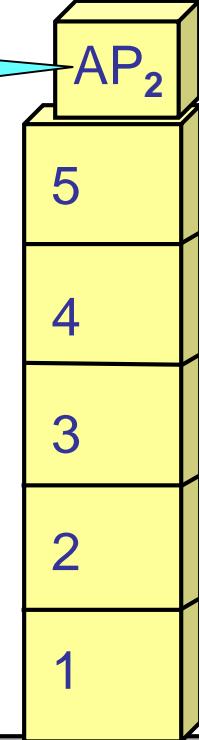
案例:计算机1 向计算机2 发送数据

计算机1



我收到了 AP_1 发来的
应用程序数据!

计算机2



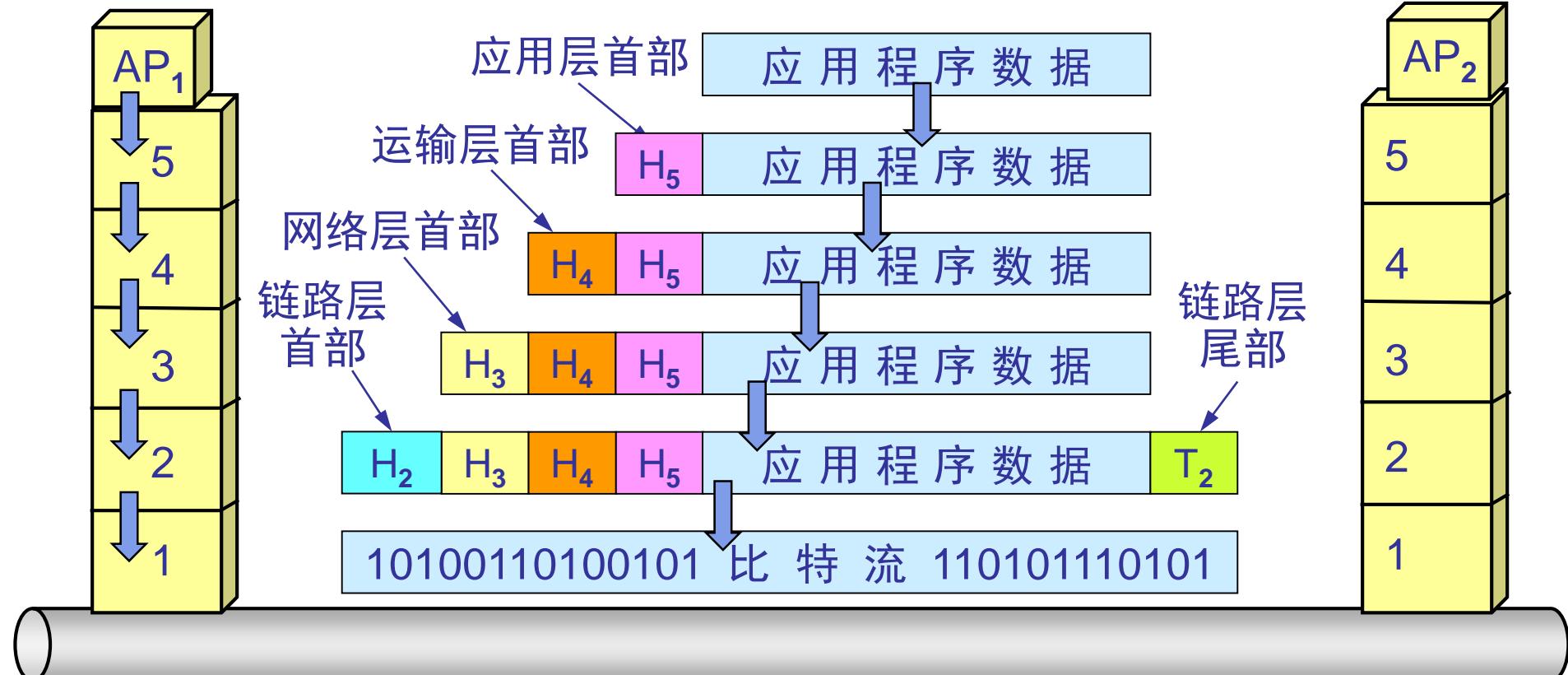
案例:计算机1 向计算机2 发送数据



计算机 1

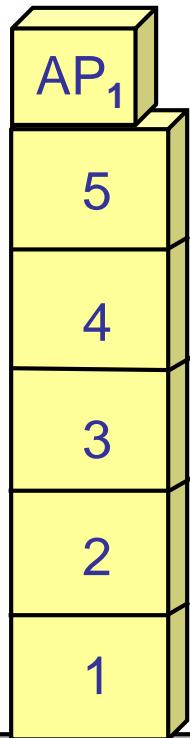
注意观察加入或剥去首部（尾部）的层次

计算机 2

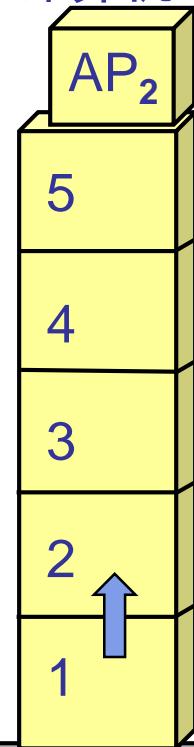


案例:计算机 1 向计算机 2 发送数据

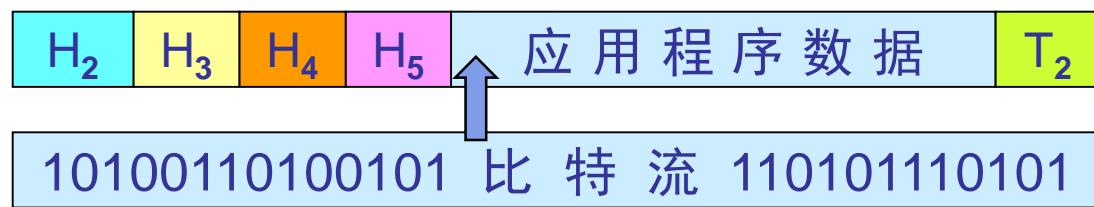
计算机 1



计算机 2

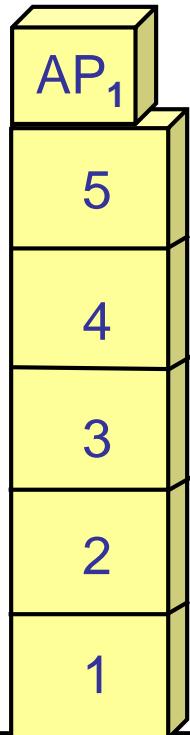


计算机 2 的物理层收到比特流后
交给数据链路层

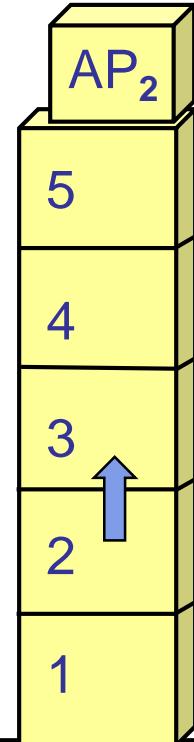


案例:计算机1 向计算机2 发送数据

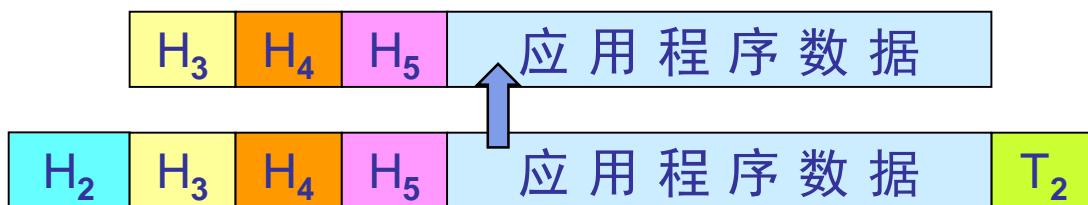
计算机1



计算机2

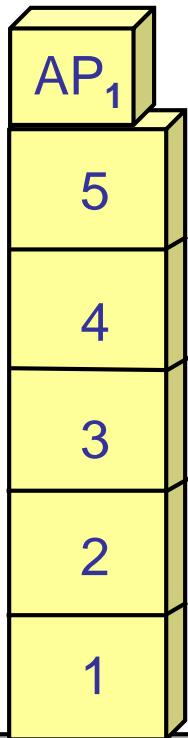


数据链路层剥去帧首部和帧尾部后
把帧的数据部分交给网络层

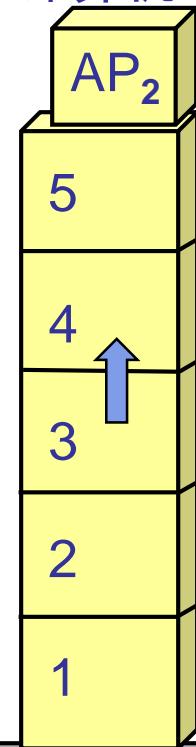


案例:计算机1 向计算机2 发送数据

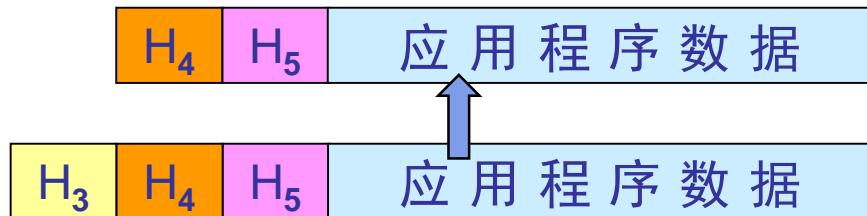
计算机1



计算机2

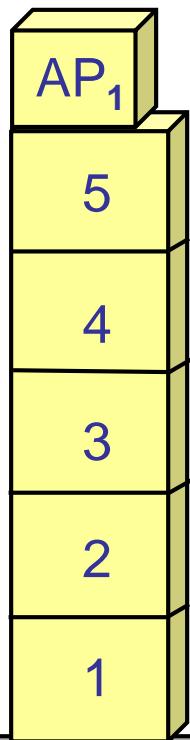


网络层剥去分组首部后
把分组的数据部分交给运输层

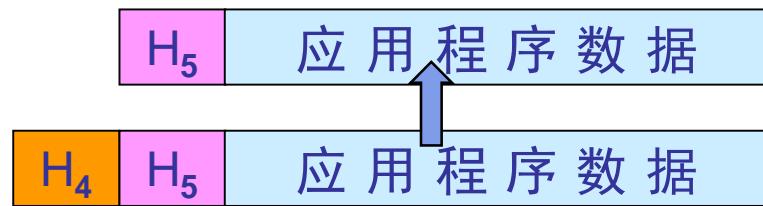


案例:计算机1 向计算机2 发送数据

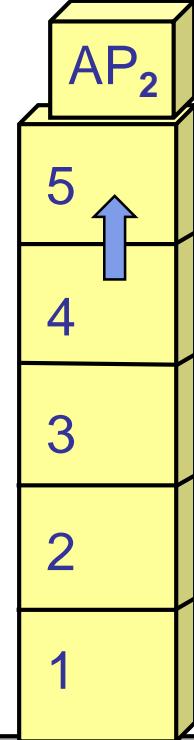
计算机1



运输层剥去报文头部后
把报文的数据部分交给应用层

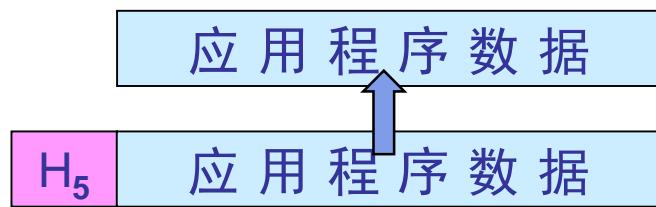
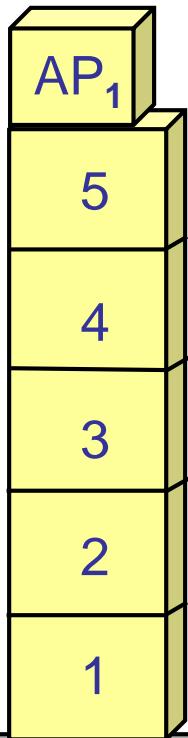


计算机2



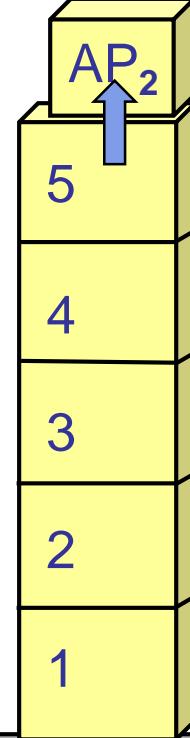
案例:计算机1 向计算机2 发送数据

计算机1



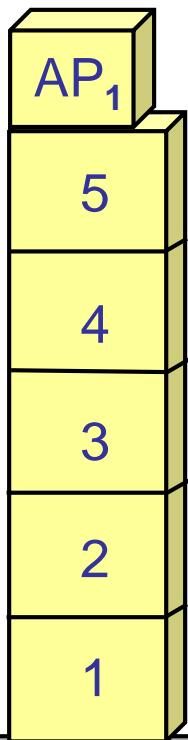
应用层剥去应用层 PDU 首部后
把应用程序数据交给应用进程

计算机2



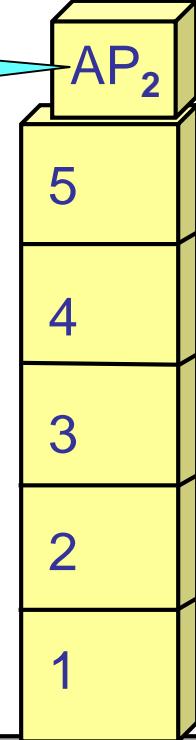
案例:计算机1 向计算机2 发送数据

计算机1



我收到了 AP₁ 发来的
应用程序数据!

计算机2





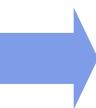
Demo

- Message Segmentation
- http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/message/messagesegmentation.html



提纲

- 背景
 - 核心问题: 构造一个网络
- 设计需求
 - 可扩展的连通性
 - 高性价比的资源共享
 - 支持通用服务
 - 可管理性
- 网络架构
 - 分层与协议
 - 互联网体系结构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- 总结



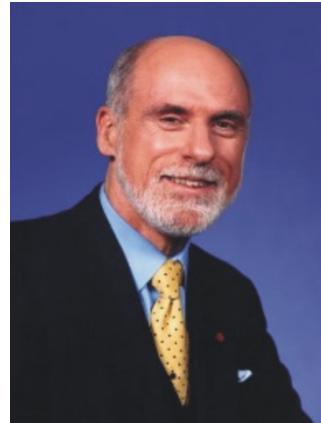


互联网体系结构



● Internet

- 全球范围, 通用的, 支持异构技术, 公众的, 支持各种不同应用的计算机网络
- 由 ARPANET 演变形成
- 由研究团体开发
- 标准 – IETF(因特网工程任务组)



Vinton Cerf Robert Kahn
TCP/IP 协议的发明者

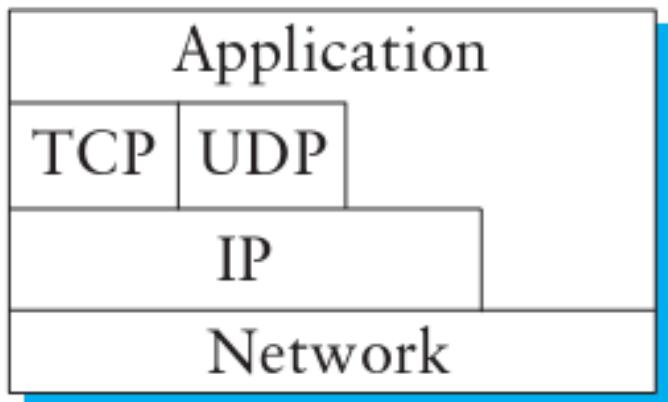


我们拒绝国王、总统和选举。我们信奉的是大体上的一致意见和可运行的代码。

D. D. Clark

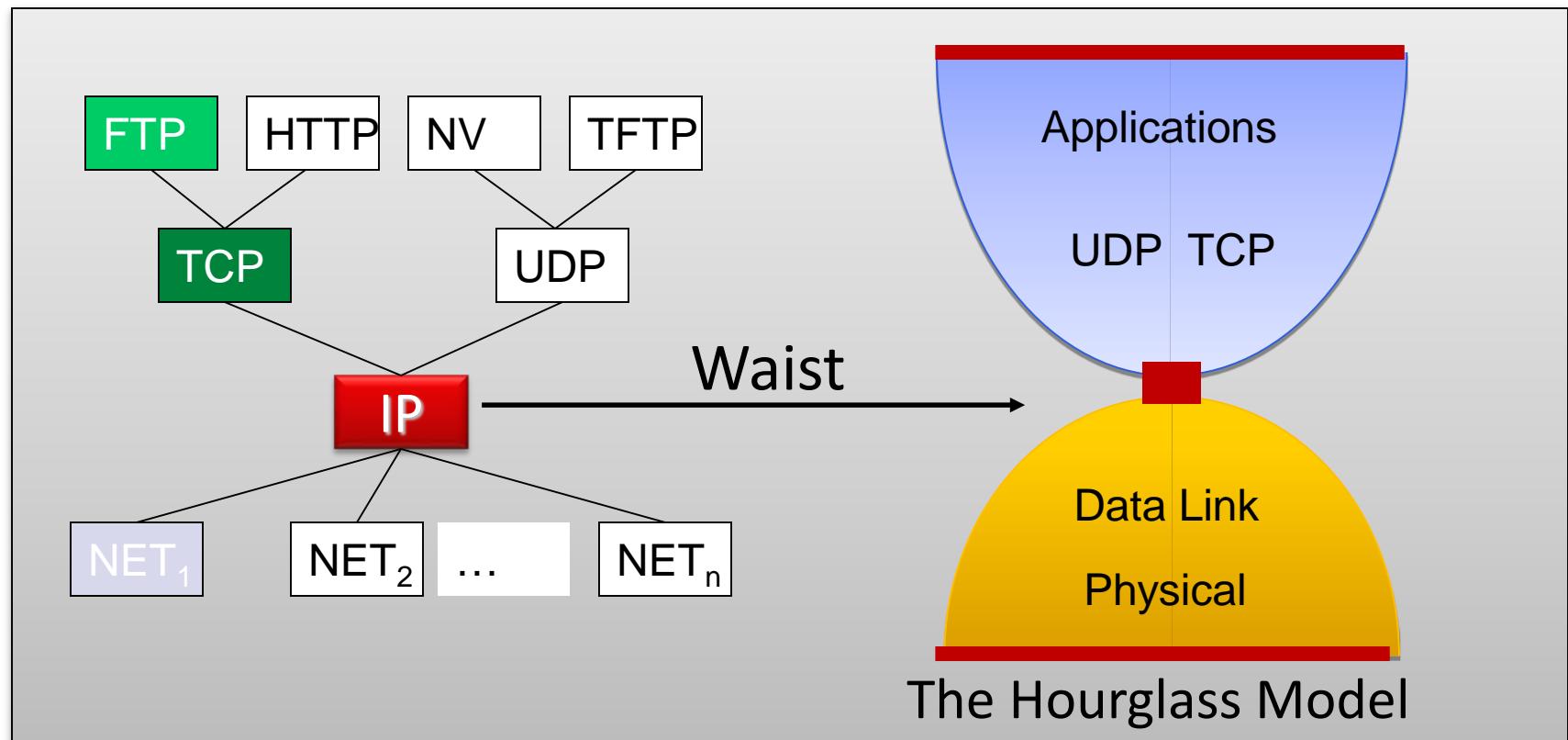


互联网体系结构



- 应用层
- 传输层
 - 进程与进程之间的消息传输, 两个主要的协议:
 - TCP (Transmission Control Protocol)
 - UDP (User Datagram Protocol)
- IP层
 - 主机到主机的数据分组传输
 - 唯一协议: IP
- 网络接入层
 - 没有指定该层的实际细节
 - 可以是任意一种底层网络

互联网协议栈

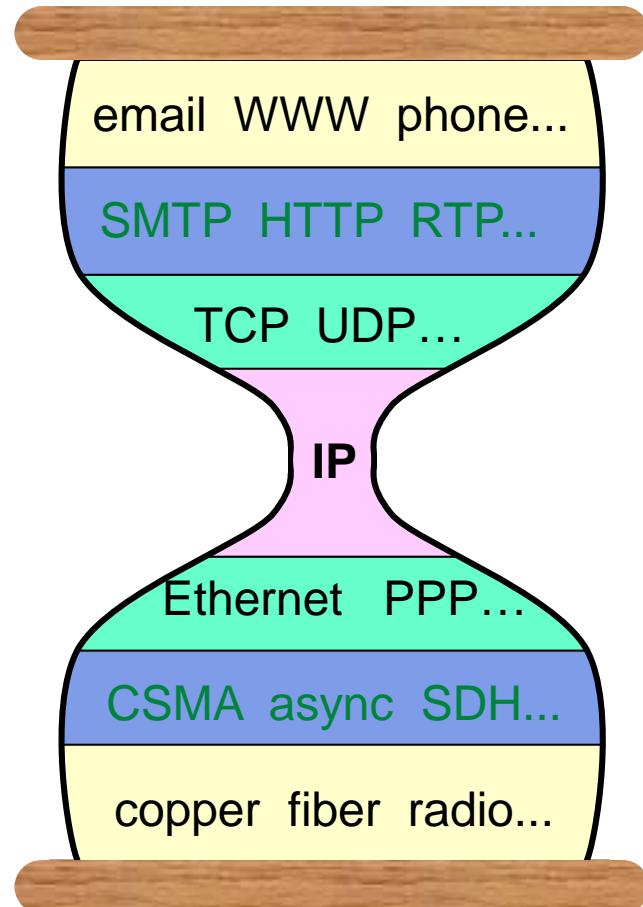


细腰的设计增强了IP协议的适应性



互联网体系结构的特点

- 沙漏形状: 顶部和底部宽, 中间窄
 - IP协议是焦点
- 未严格的划分层次
 - 应用可以跨过定义的运输层直接使用IP或一个底层网络
 - 不存在实际的参考模型
- 强调协议的实现
 - 依网络服务所需的各项功能为主线来刻画功能之间的关系

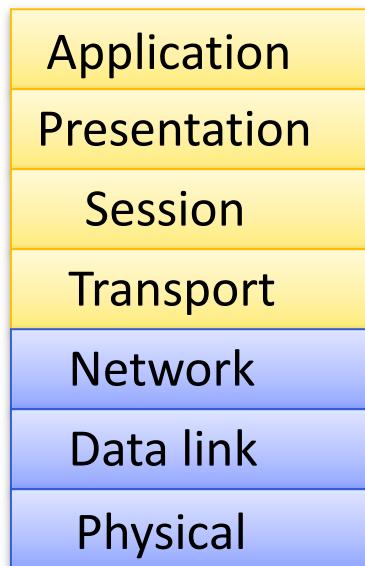




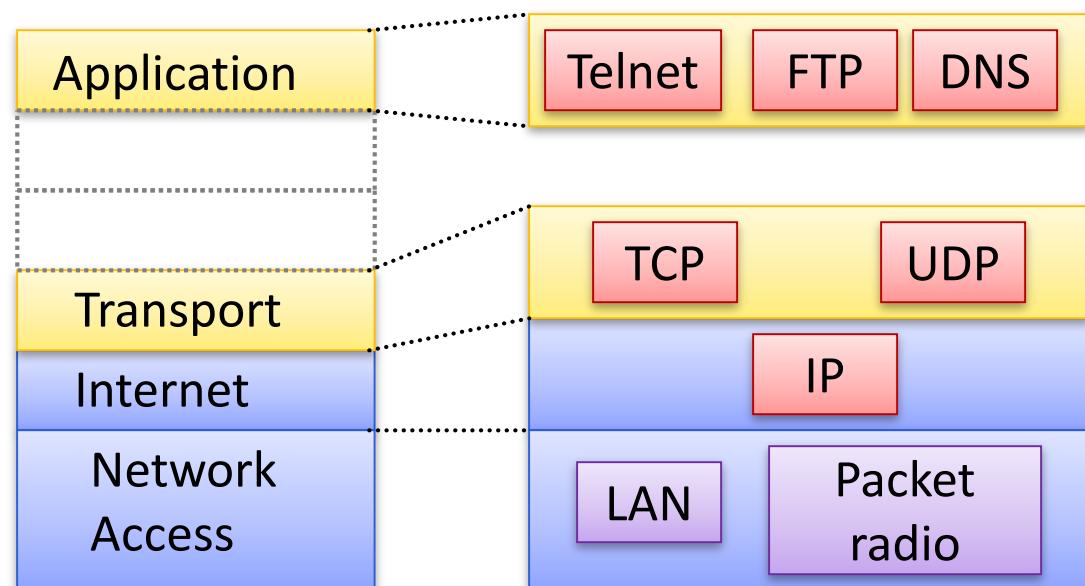
OSI模型 vs. 互联网体系结构



- ISO-OSI model
 - 网络互联的理论模型
- TCP/IP model
 - 事实的工业标准



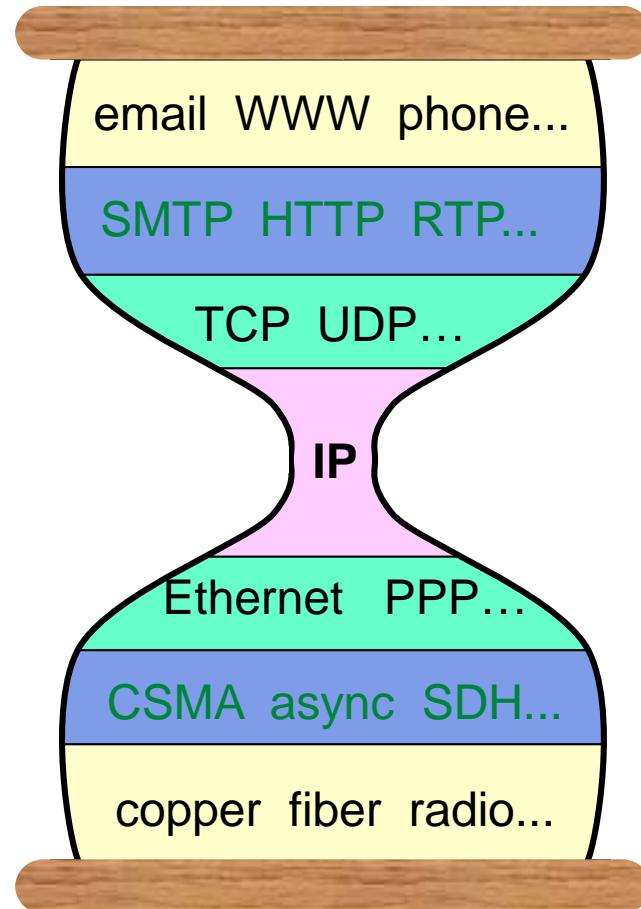
ISO OSI network architecture
(7-layer Reference model)



Internet Architecture
(4-layer TCP/IP protocol stacks)

比较: (1) 设计动机; (2)分层结构; (3)协议实现 (4) 概念定义

小结: 互联网设计原则(I)

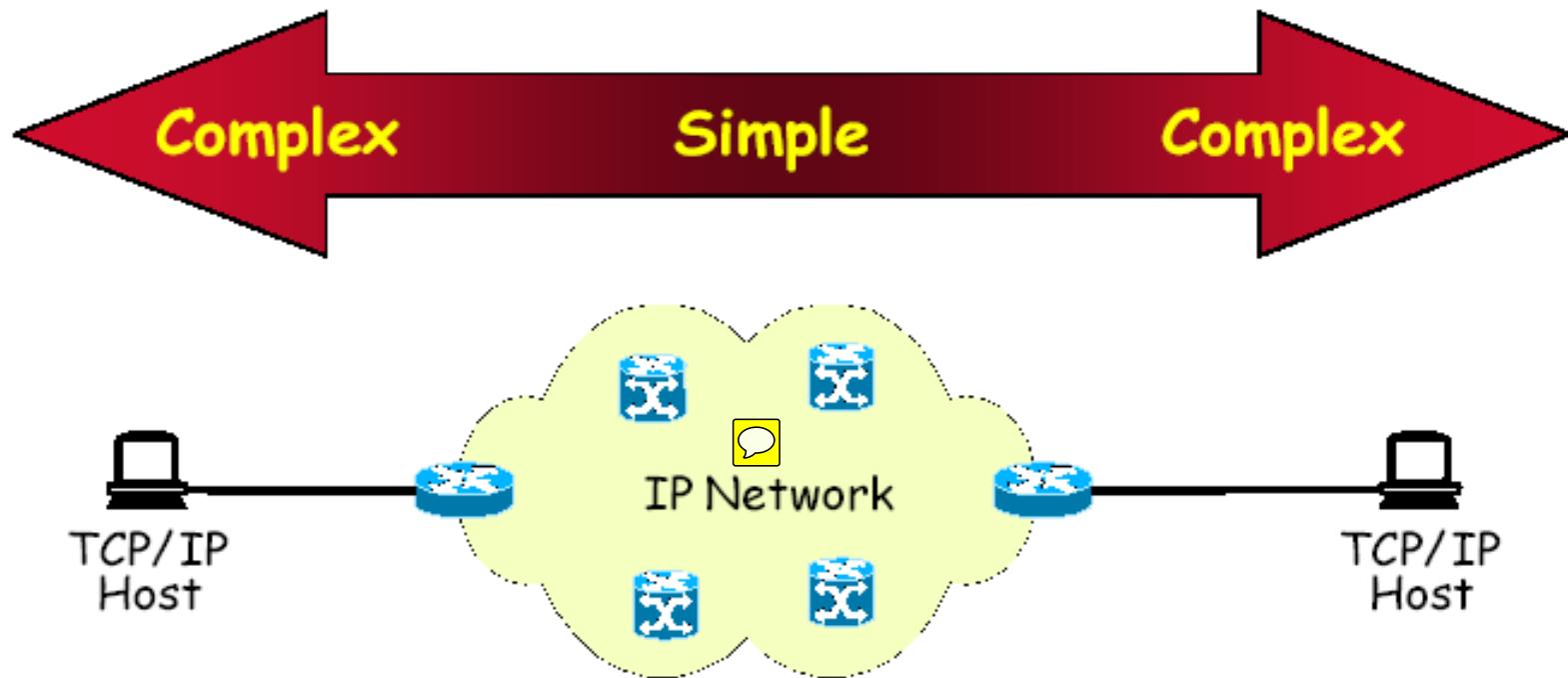


沙漏设计原则

沙漏的细腰代表最小的、经过精心挑选的通用功能集合，它允许高层应用和底层通信技术并存，共享各种功能，并快速发展。

沙漏模型是Internet能够快速适应用户新应用需求和网络技术更新的关键。

小结: 互联网设计原则 (II)



``Simple Core and Complex Host'' Philosophy



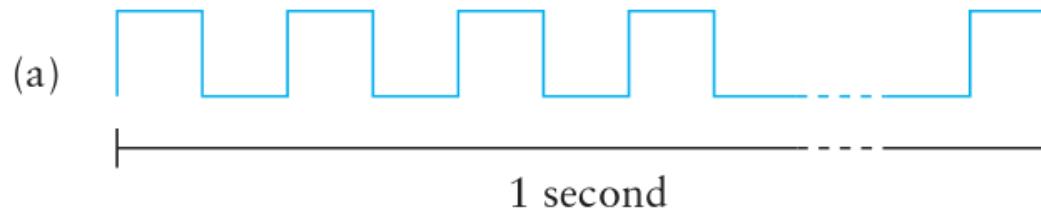
提纲

- 背景
 - 核心问题: 构造一个网络
- 需求分析
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- 总结

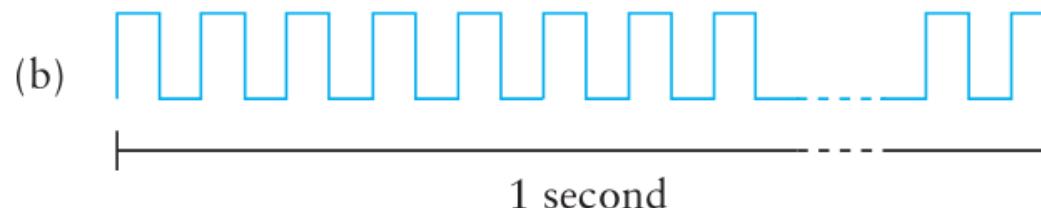


链路带宽

- 原始定义：信号的频带宽度，用 Hertz (Hz) 表示
- 网络定义：一段特定的时间内网络所能传送的比特数，用 bits per second (bps) 表示



(a) bits transmitted at 1 Mbps (each bit 1 μ s wide)

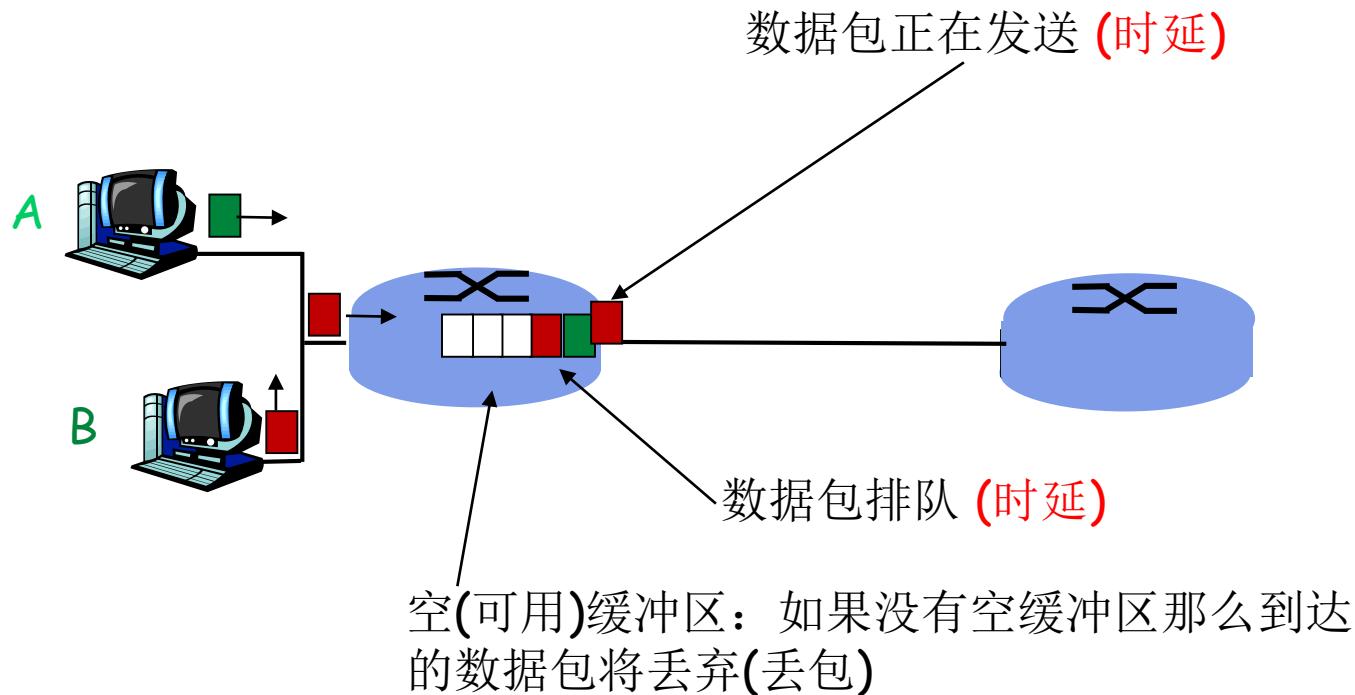


(b) bits transmitted at 2 Mbps (each bit 0.5 μ s wide)

丢包和时延是如何发生的?

数据包在路由器缓冲区中排队

- 数据包到达的比率超出了输出链路的容量
- 数据包排队, 等待处理



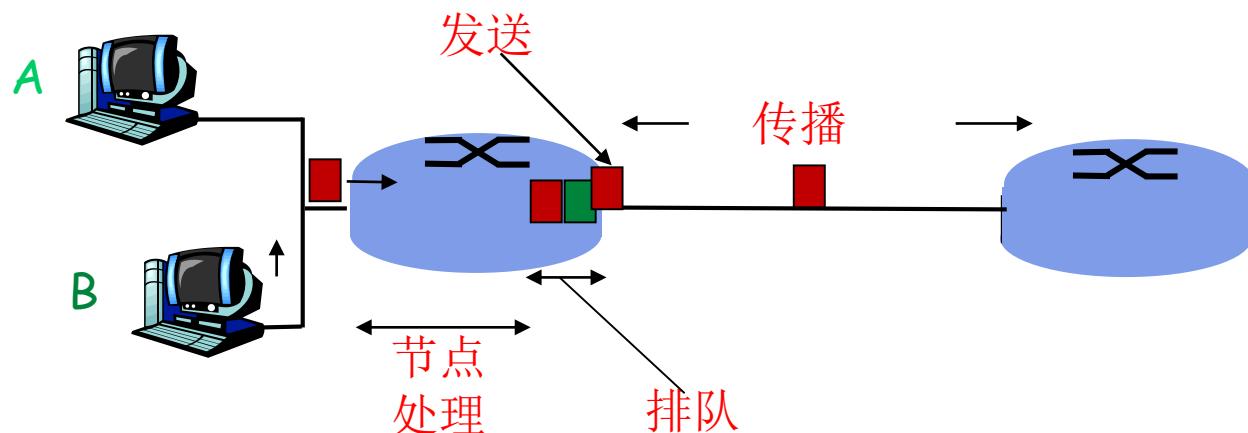


Demo

- Queuing and Loss
- http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/queuing/queuing.html

分组交换网中时延的四个来源

- 1. 节点处理时延
 - 检查位错误
 - 确定输出链路
- 2. 排队时延
 - 在输出链路等待数据发送的时间
 - 取决于路由器的拥塞状况



分组交换中的时延

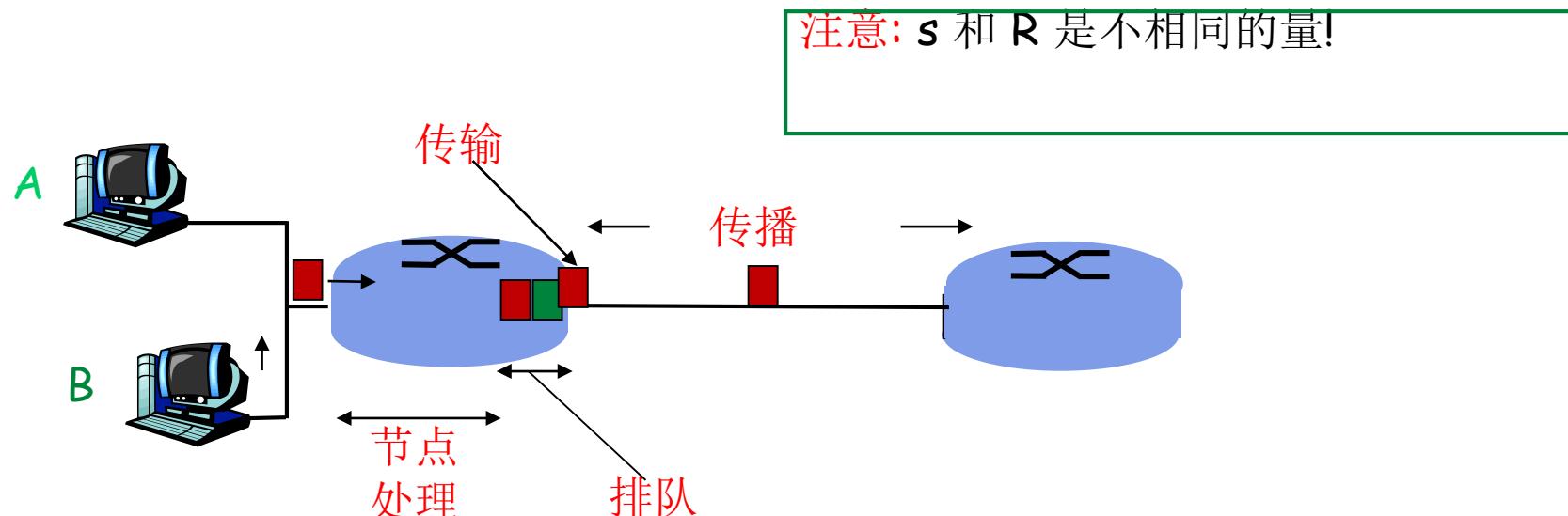


- 3. 传输时延:

- R =链路带宽 (bps)
- L =数据包长度 (bits)
- 传输所需时间 = L/R

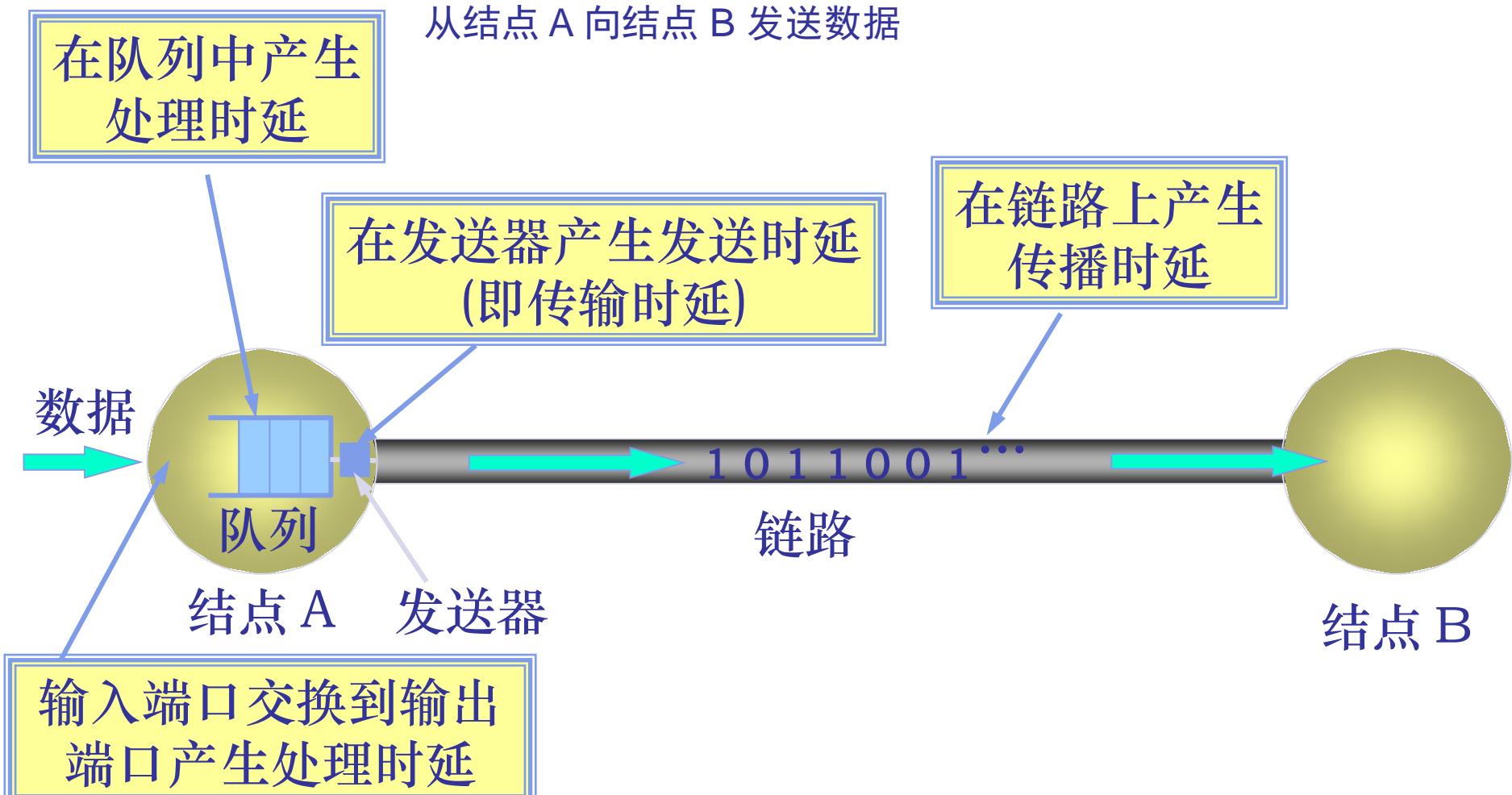
- 4. 传播时延:

- d = 物理链路长度
- s = 在介质中的传播速度 ($\sim 2 \times 10^8$ 米/秒)
- 传播时延 = d/s

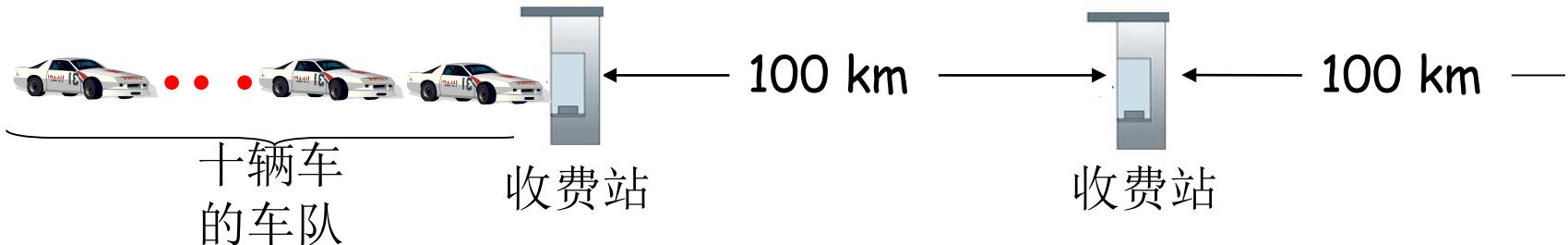


网络性能：端到端时延

- 定义：将报文从网络的一端传到另一端所需花费的时间



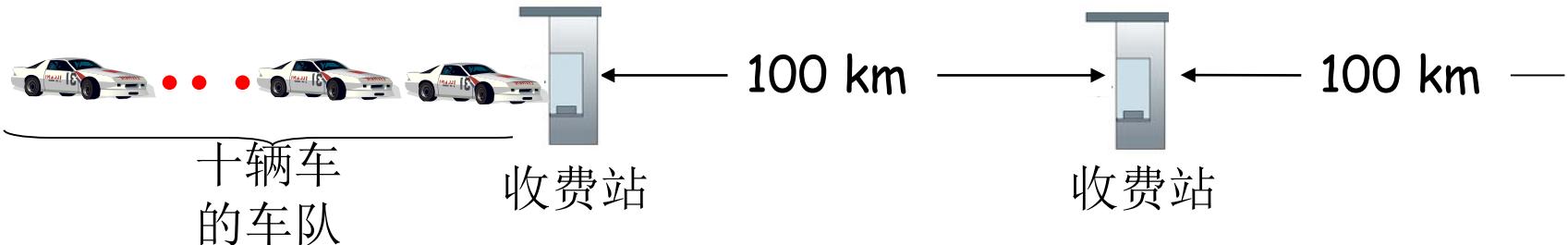
时延与带宽：车队比喻



- 车以100公里/小时的速度“传播”
- 收费站12秒服务一辆车(发送时延)
- 车~比特; 车队 ~ 数据包
- 问: 车队到达第二个收费站需要多久?

- 整个车队通过收费站所需时间= $12 * 10 = 120$ 秒
- 最后一辆车从第一个收费站“传播”到第二个收费站所需时间= $100\text{km}/(100\text{km/hr}) = 1 \text{ hr}$
- 答: 62 分钟

时延与带宽：车队比喻



- 车现在以1000公里/小时的速度“传播”
- 收费站服务一辆车现需1分钟
- 问：第一个收费站在服务完所有车之前是否有车达到第二个收费站？

- 是的可以！7分钟后，第一辆车到达第二个收费站而这时还有3辆车在第一个收费站
- 数据包中的第一个比特可以在第一个路由器发送完全部数据之前到达第二个路由器！



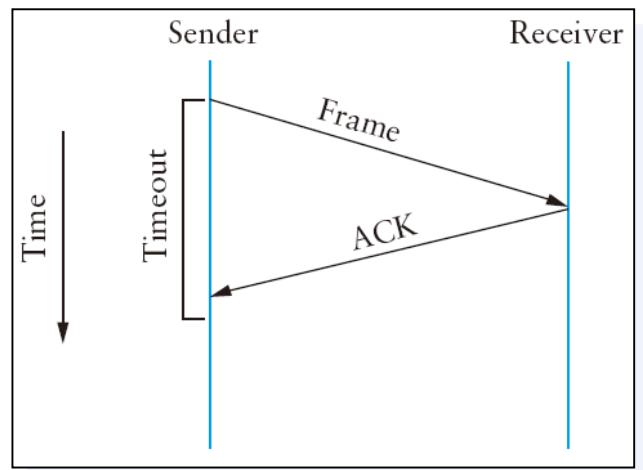
Demo

- Transmission versus Propagation Delay
- http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/transmission/delay.html

时延带宽积



- 等效于第一个比特到达接收方之间发送方可以发送的比特数
 - 网络中保持的比特数
- 发送方在接收到对方的确认前能够发完2倍的时延和带宽积的数据
 -  RTT \times 带宽



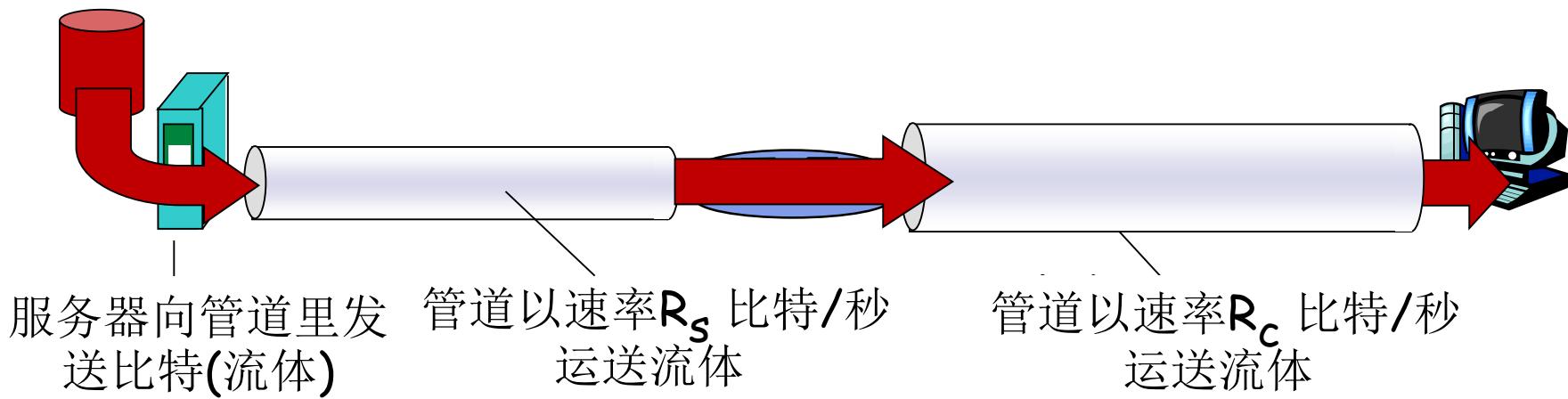
Link Type	Bandwidth (Typical)	Distance (Typical)	Round-trip Delay	Delay \times BW
Dial-up	56 Kbps	10 km	87 μ s	5 bits
Wireless LAN	54 Mbps	50 m	0.33 μ s	18 bits
Satellite	45 Mbps	35,000 km	230 ms	10 Mb
Cross-country fiber	10 Gbps	4,000 km	40 ms	400 Mb



网络性能: 吞吐量

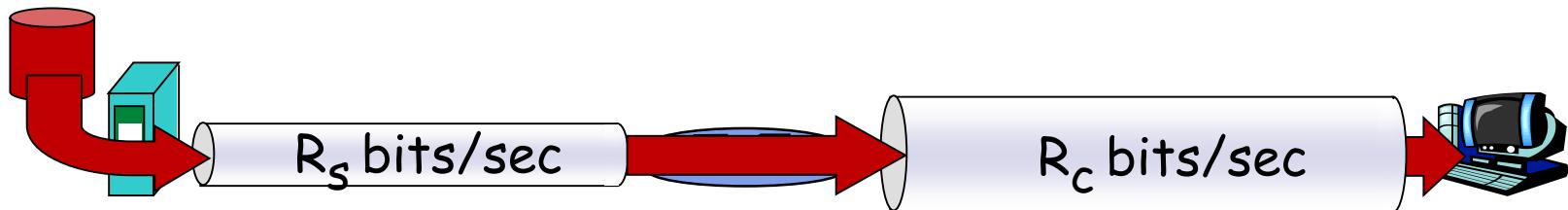


- **吞吐量:** 发送者与接收者之间比特交换的速度 (比特/单位时间)
 - **瞬时:** 在给定时间点的速率
 - **平均:** 在较长一段时间内的速率

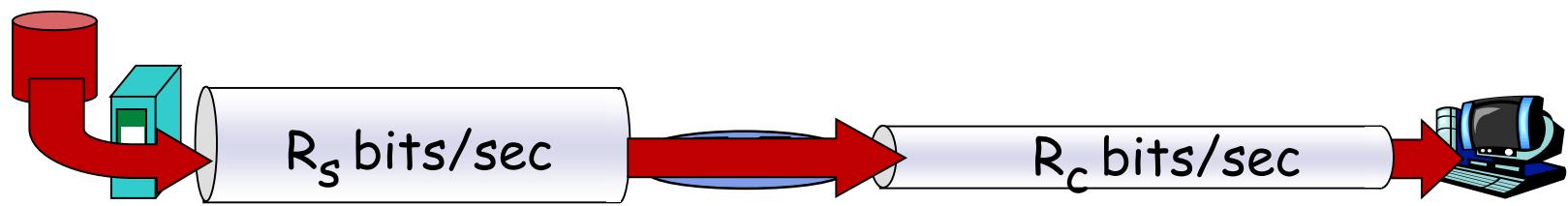


吞吐量(2)

- $R_s < R_c$ 平均端到端的吞吐量是多少?



- $R_s > R_c$ 平均端到端的吞吐量是多少?

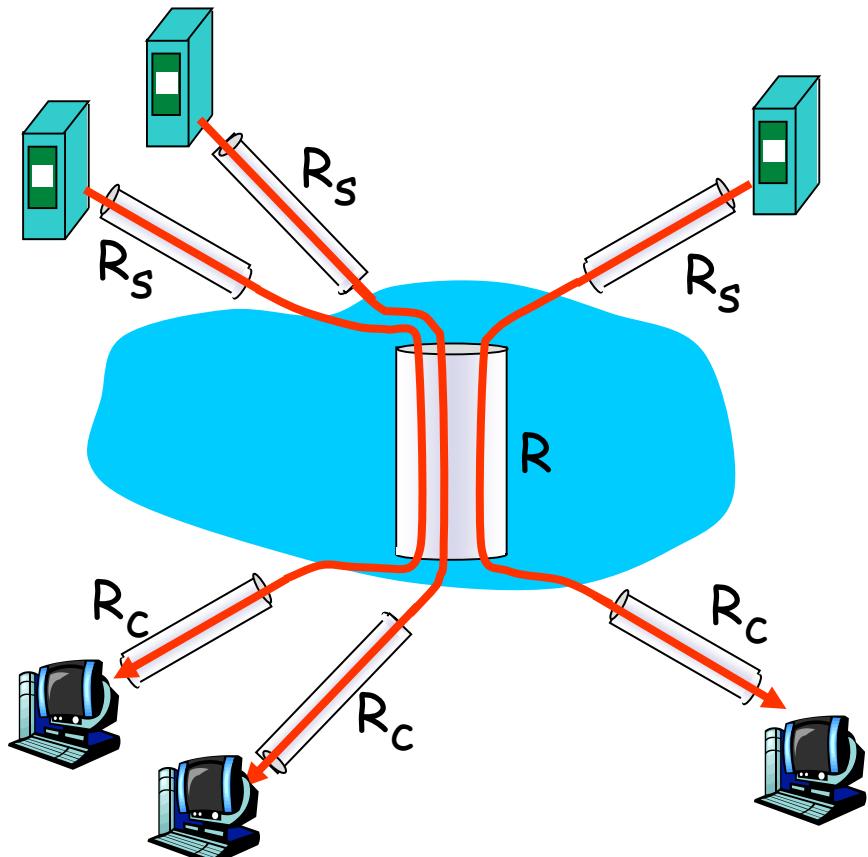


链路瓶颈

端到端的链路制约端到端的吞吐量

吞吐量: 网络场景

- 每个链接的端到端吞吐量:
最小值($R_c, R_s, R/10$)
- 实际上: R_c 或 R_s 往往是瓶颈



10 条链接(公平的) 共享骨干瓶颈链路
 R 比特/秒



应用性能需求

- 不同的应用带宽需求
 - 平均速率及突发流量
 - 示例: 压缩的视频流
- 速率变化带来的问题
 - 拥塞
 - 内存溢出导致的数据包丢弃
- 速率变化问题的解决方案
 - 用峰值速率和持续时间描述突发流量
 - 优化设计/合理分配内存容量

应用性能需求

- 时延抖动
 - 抖动: 时延的变化
 - 抖动主要产生于变化的队列时延



- 时延抖动的影响
 - 流媒体播放不平滑 (特别是多媒体应用)
- 广播应用的解决方案
 - 在接收方进行缓存
- 交互式流媒体应用的解决方案?



提纲

- 背景
 - 核心问题: 构造一个网络
- 需求分析
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- 总结





套接字编程

目标: 学习如何建立客户机/服务器应用程序，使用套接字进行通信

Socket API

- 1981年在BSD4.1 UNIX中提出
- 应用程序可以自由的使用该协议
- 客户机/服务器模型
- 通过socket API实现两种传输服务：
 - 不可靠数据报
 - 可靠的，面向字节流的数据报



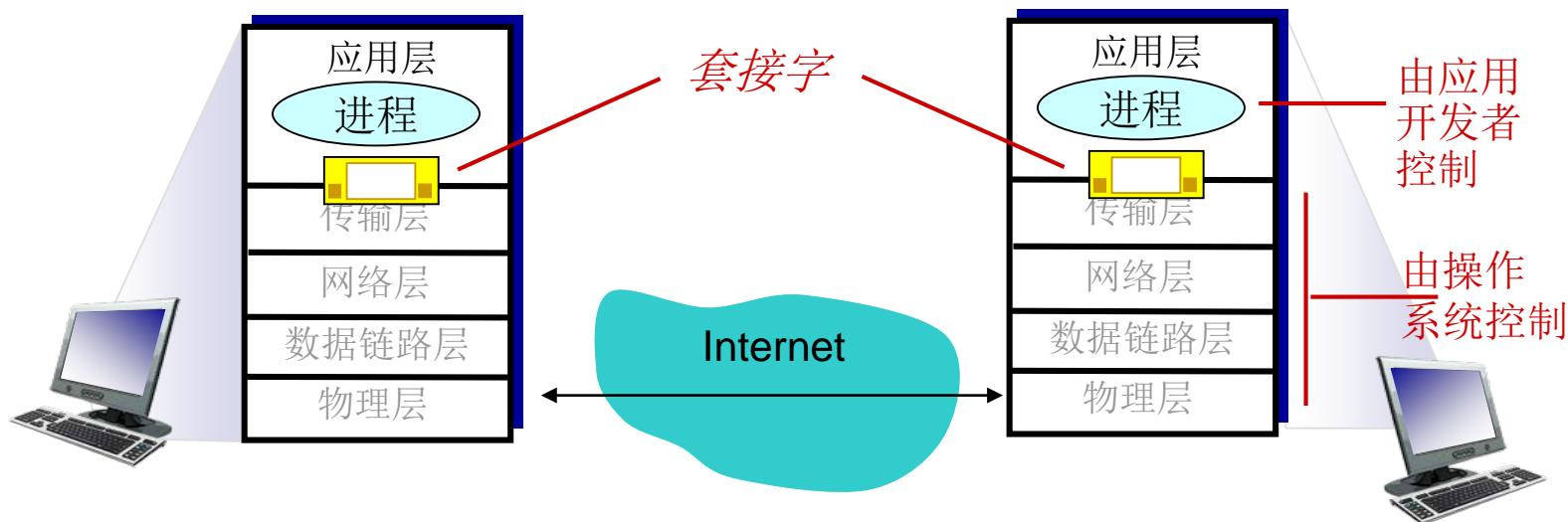
socket

一个本地的，
应用程序创建的，
操作系统控制的接口。
通过该接口
应用进程可以
从另一个应用进程
发送或者接收消息

套接字编程

目标: 学习如何建立客户机/服务器应用程序，使用套接字进行通信

套接字(socket): 应用进程之间的接口，实现端到端的传输协议





套接字编程

Socket API类别

- **PF_INET**: 用于互联网编程
- **PF_UNIX**: 用于Unix管道编程
- **PF_PACKET**: 直接访问网络接口 (*i.e.*, 绕过TCP / IP协议栈)

两种基本socket API分别支持两种传输服务:

- **SOCK_STREAM**: 可靠的, 字节流服务 (**TCP**)
- **SOCK_DGRAM**: 非可靠的数据报服务 (**UDP**)

应用案例:

1. 客户端从键盘读取一行字符（数据）并将该数据发送到服务器；
2. 服务器接收数据后并显示在屏幕上。



TCP套接字编程

服务器做好准备

- 服务器进程必须首先运行
- 服务器必须先创建socket等待处理客户端的连接请求

客户机与服务器的联系

- 创建客户端本地的TCP socket
- 指明服务器进程的IP地址和端口号
- 当客户端创建socket后：客户端的TCP socket与服务器TCP socket建立连接

- 服务器收到客户端连接请求后，**创建新的socket**用来与客户端通信
 - 允许服务器同时与多个客户端通信
 - 源端口号用来识别客户端

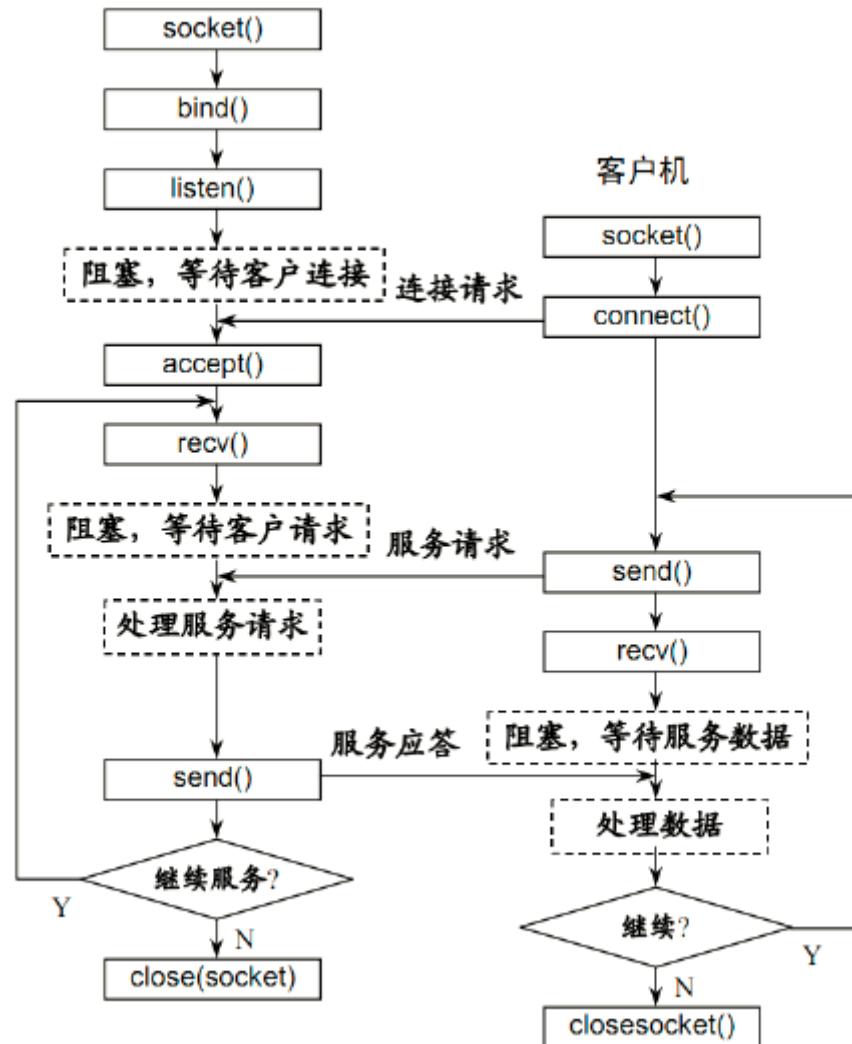
应用程序的观点

TCP协议提供了客户端和服务器之间可靠的，按次序的字节流传输

TCP套接字编程

客户机-服务器程序示例：

- 1) 客户机从标准输入中读取信息，通过socket向服务器发送；
- 2) 服务器从socket中读取消息；
- 3) 服务器将消息显示在屏幕上。





示例: C客户端 (TCP)

Socket编
程库文件

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 5432
#define MAX_LINE 256

int main(int argc, char * argv[])
{
    FILE *fp;
    struct hostent *hp;
    struct sockaddr_in sin;
    char *host;
    char buf[MAX_LINE];
    int s;
    int len;
    if (argc==2) {
        host = argv[1];
    }
    else {
        fprintf(stderr, "usage: simplex-talk host\n");
        exit(1);
    }
```

程序参数
处理



示例: C客户端 (TCP)

```
/* translate host name into peer's IP address */
hp = gethostbyname(host);
if (!hp) {
    fprintf(stderr, "simplex-talk: unknown host: %s\n", host);
    exit(1);
}

/* build address data structure */
bzero((char *)&sin, sizeof(sin));
sin.sin_family = AF_INET;
bcopy(hp->h_addr, (char *)&sin.sin_addr, hp->h_length);
sin.sin_port = htons(SERVER_PORT);

/* active open */
if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
    perror("simplex-talk: socket");
    exit(1);
}
if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
    perror("simplex-talk: connect");
    close(s);
    exit(1);
}

/* main loop: get and send lines of text */
while (fgets(buf, sizeof(buf), stdin)) {
    buf[MAX_LINE-1] = '\0';
    len = strlen(buf) + 1;
    send(s, buf, len, 0);
}
```

域名/IP转换

参数初始化

创建
客户端socket,

连接服务器

从屏幕读取
字符串



示例: C服务器(TCP)

Socket编
程库文件

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_PENDING 5
#define MAX_LINE 256

int main()
{
    struct sockaddr_in sin;
    char buf[MAX_LINE];
    int len;
    int s, new_s;
    /* build address data structure */
    bzero((char *)&sin, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = INADDR_ANY;
    sin.sin_port = htons(SERVER_PORT);

    /* setup passive open */
    if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk: socket");
        exit(1);
    }
```

在端口5432创建
欢迎 socket



示例: C服务器(TCP)

```
if ((bind(s, (struct sockaddr *)&sin, sizeof(sin))) < 0) {
    perror("simplex-talk: bind");
    exit(1);
}
listen(s, MAX_PENDING);
/* wait for connection, then receive and print text */
while(1) {
    if ((new_s = accept(s, (struct sockaddr *)&sin, &len)) < 0) {
        perror("simplex-talk: accept");
        exit(1);
    }
    while (len = recv(new_s, buf, sizeof(buf), 0))
        fputs(buf, stdout);
    close(new_s);
}
```

服务socket等待客户端连接

创建服务客户端的Socket，并读取信息

服务器将信息显示在屏幕上

while循环结束，等待另外的客户端连接

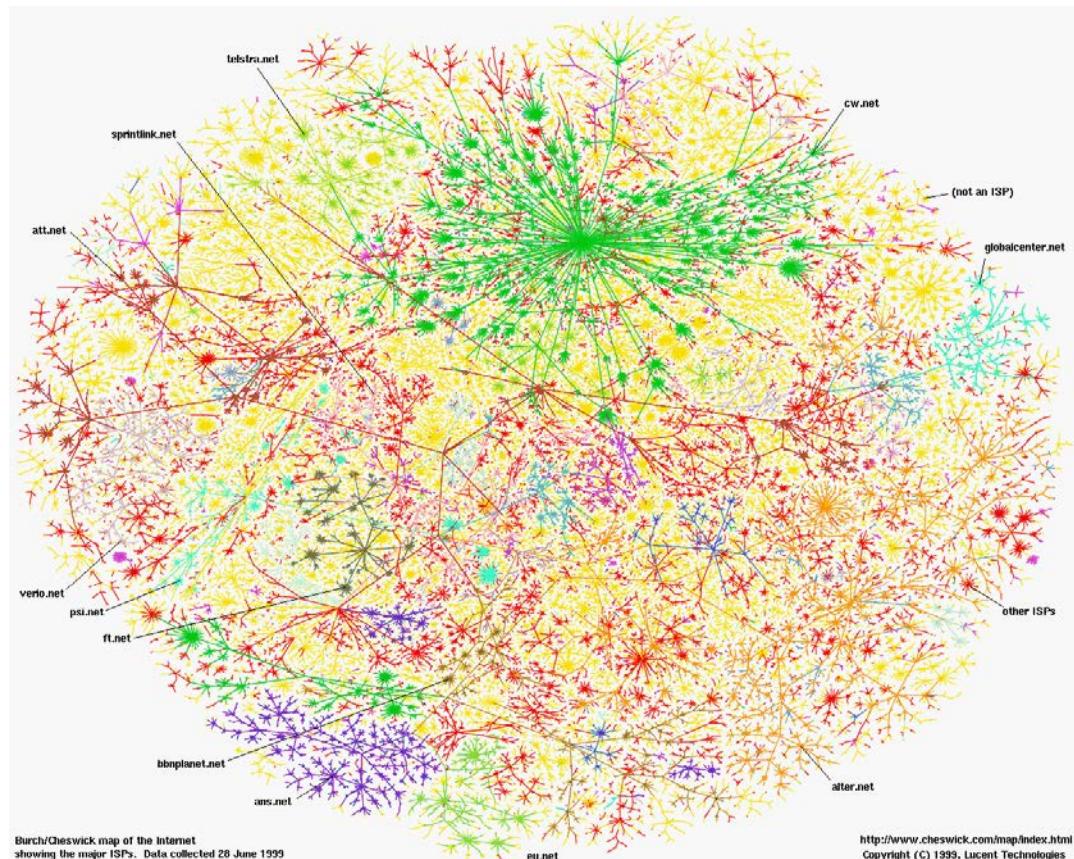


提纲

- 背景
 - 核心问题: 构造一个网络
- 需求分析
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- ● 互联网简史
- 总结

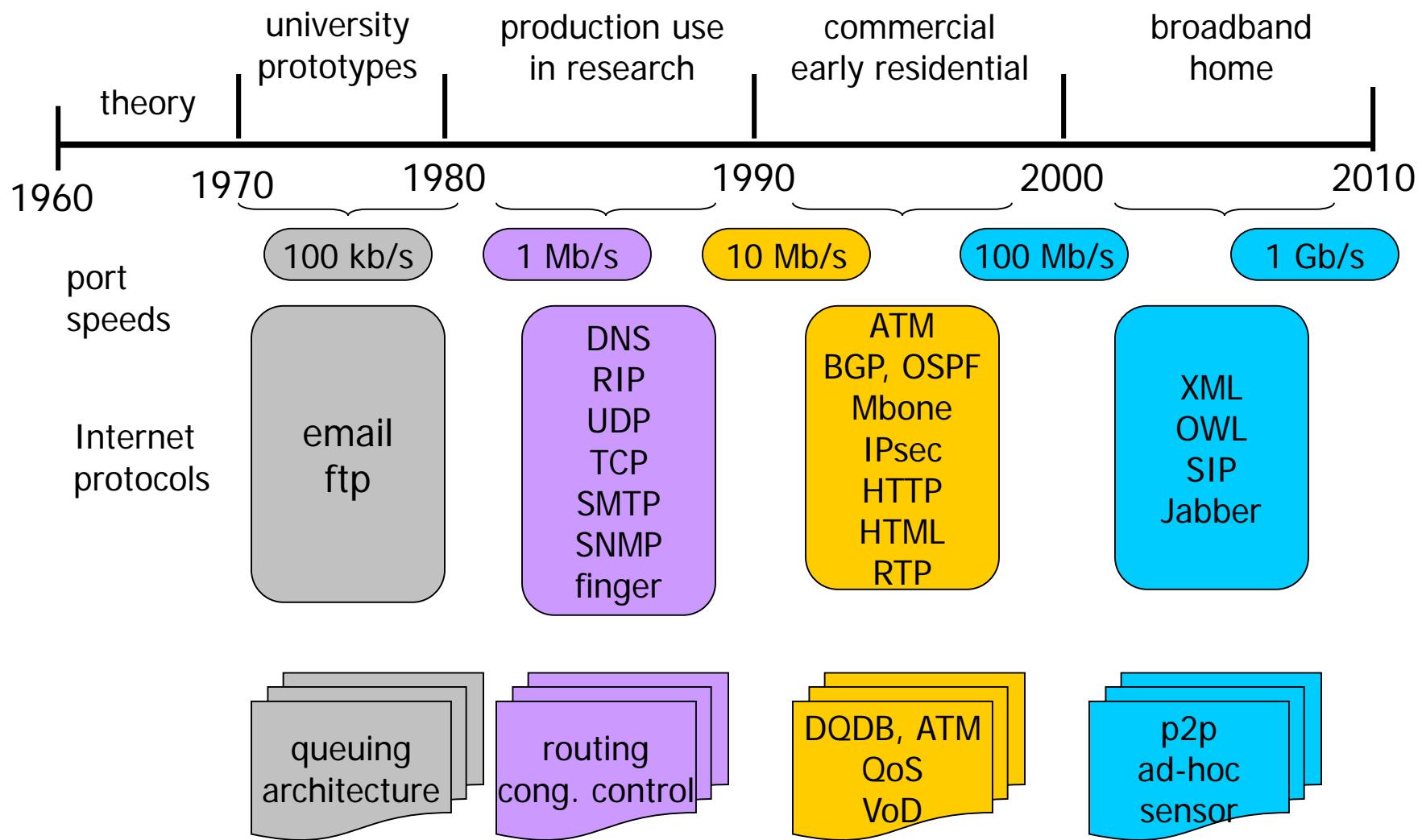


互联网的成长！





互联网时间线

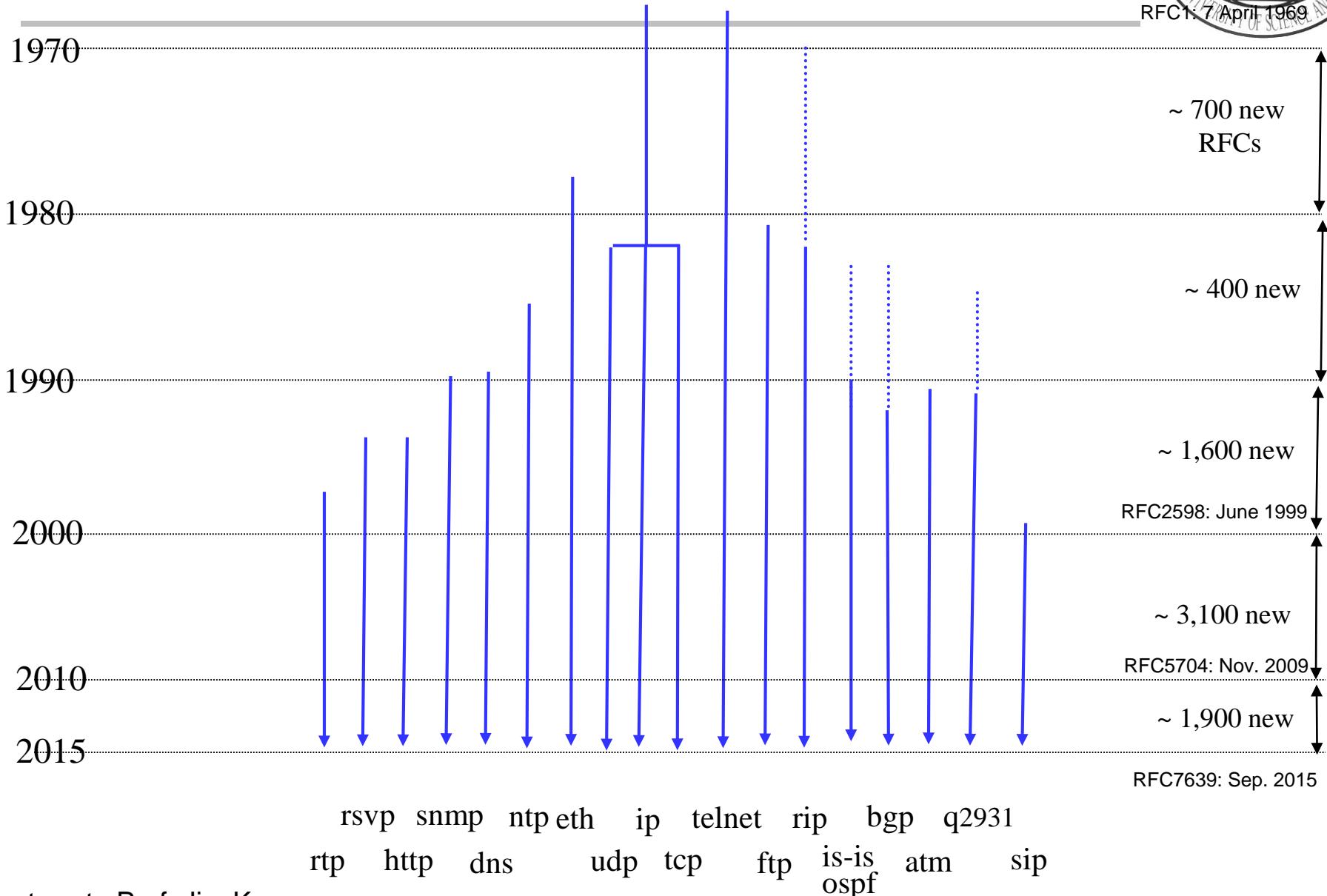




成熟的网络协议



RFC 1: 7 April 1969





互联网简史

1961-1972: 早期的分组交换

- 1961: Kleinrock - 利用排队论分析了分组交换的效率
- 1964: Baran - 在军用网中使用分组交换技术
- 1967: 提出ARPAnet网络构想
- 1969: 运行第一个ARPAnet分组交换网节点

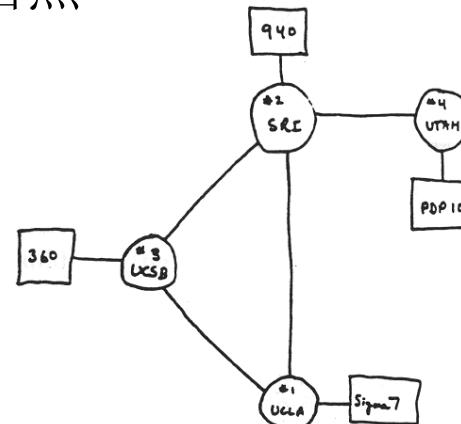


1961: Kleinrock - queueing theory shows effectiveness of packet-switching

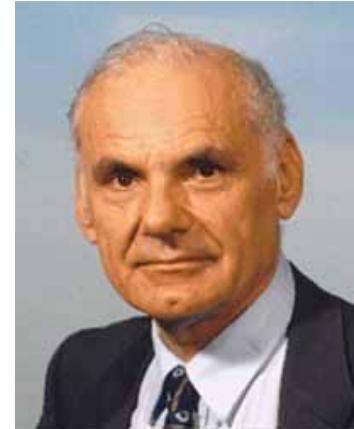


1964: Paul Baran - packet-switching in military nets

- 1972:
 - ARPAnet 公开示范
 - NCP (网络控制协议) 第一个端到端协议
 - 第一个 e-mail 程序
 - ARPAnet 拥有 15 个结点



THE ARPA NETWORK



1967: Lawrence Roberts: 宣布 ARPAnet 计划，资金来源于国防部的高级研究规划局





互联网简史

1972-1980: 互联网络, 新的和专用网

- 1970: 在夏威创建ALOHAnet 卫星网络
- 1974: Cerf 和 Kahn 提出互联网络体系架构
- 1976: 在Xerox PARC构建第一个以太网
- 70年到后期: 专用网络: DECnet, SNA, XNA
- 70年代后期: 交换固定长度的数据包 (ATM 先驱)
- 1979: ARPAnet拥有200个结点

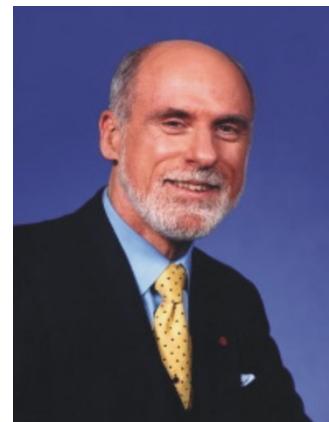
Cerf 和 Kahn 的互联网原则定义了如今的因特网架构:

- 简单、自治、网络互连无需自治区域内部的改变
- 尽最大努力交付服务
- 全球互联的路由器
- 分布式控制



1972: Robert Kahn -
ARPAnet public
demonstration

1973: Robert Kahn and
Vinton Cerf :
Transmission Control
Protocol (TCP)





互联网简史 (70s年代)

- Kahn明确提出了现行Internet的基本架构原则
- "open network architecture"
 - **简约, 自治**: 网络能够独立运行, 与其他网络互联时不需要调整;
 - **尽最大努力服务**: 互联网应当提供尽最大努力的端到端服务. 如果要求保证通信可靠性, 发送方有责任对丢失的报文进行重传;
 - **无状态路由设备**: 互联网中的路由设备不会为任何链接维护状态信息
 - **分散控制**: 互联网中无法实现中央控制.





互联网简史

1980-1990: 协议不断完善, 网络逐渐普及

- 1983: 部署 TCP/IP
- 1982: smtp
- 1983: DNS
- 1985: ftp
- 1988: TCP 拥塞控制
- 新的国家级网络: Csnet, BITnet, NSFnet, Minitel
- 100,000 个主机与互联网连接



V. Paxson: TCP

Van Paxson, Measurements and Analysis of End-to-End Internet Dynamics,
University of California, Berkeley, Ph.D. thesis, 1997



互联网简史

90年代至今: 网络商业化, Web应用及其它网络新应用层出不穷

- 90年代初: ARPAnet 退役
- 1991: NSF 限制NSFnet的商业应用, (NSFnet于1995年退役)
- 90年代初: Web应用
 - 超文本 [Bush 1945, Nelson 1960's]
 - HTML, HTTP: Berners-Lee
 - 1994: Mosaic, Netscape
 - 90年代后期: Web应用商业化

90年代后期:

- 新兴应用: 即时通讯, P2P 文件共享
- 网络安全凸显重要性
- 至少 5 千万主机, 1亿多用户
- 骨干链路速率达Gbps 级

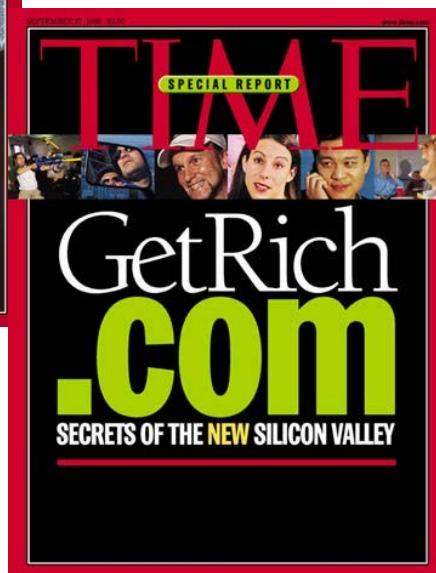
互联网简史 (90s 年代)



1995



1996



1999



2001



互联网简史

2000年至2005年：

- IP电话, IP视频
- P2P 应用: BitTorrent (文件共享), Skype (网络电话), PPLive (网络电视)
- 更多应用: YouTube, 网络游戏, 微博
- 无线, 移动应用



互联网简史

2005-现在

- ~7.5亿主机
 - 智能手机和平板
- 宽带接入的大规模部署
- 高速无线接入的普及
- 在线社交网络的蓬勃发展:
 - Facebook: 10亿用户
- 服务运营商 (Google, Microsoft)建设自己的私有网络
 - 绕过公共互联网，为用户提供“快速”网络接入访问其搜索、邮件等服务
- 电子商务, 大学、企业通过云平台提供服务 (eg, Amazon EC2)

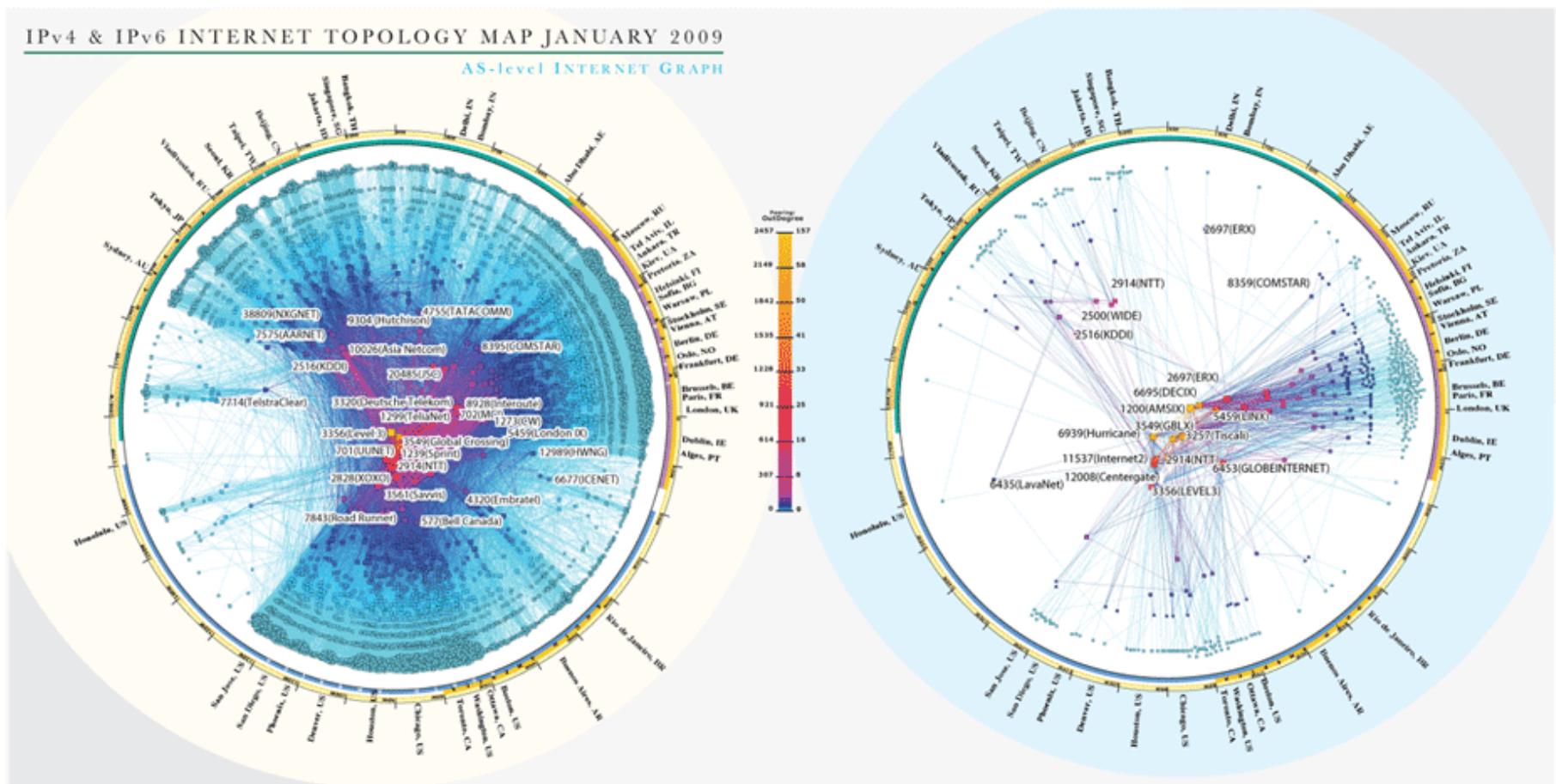


互联网拓扑



IPv4 & IPv6 INTERNET TOPOLOGY MAP JANUARY 2009

AS-level INTERNET GRAPH



ANALYSIS TEAM Bradley Huffaker, Asaf Levy
SOFTWARE DEVELOPMENT Young-Hyun, Matthew Laczkowicz
POSTER DESIGN Compuco

ARK HOSTS ARIN, APNIC, ARIN, RIPE, LACNIC, CAIDA

Cancer: CERNIC, CNRST, FORTH, PunkNet, HEAnet, Iowa State University, ARIN/RIPE2, National Research Council Canada - NRC Chile, Northeastern University, Purdue University, Southern Methodist University, TIRASNET, Universitat Leipzig, Universitat Politècnica de Catalunya, University of Cambridge, University of Hawaii, University of Luxembourg, University of Napoli, University of Oregon, University of Waterloo, University of Zurich, US Army Research Lab, Verizon

Number of IP address	Number of IP links	Number of ASes	Number of ASlinks
IPv4	4,853,991	5,682,419	17,29
IPv6	4,252	17,036	489

COOPERATIVE ASSOCIATION FOR INTERNET DATA ANALYSIS
San Diego Supercomputer Center, University of California San Diego
5000 La Jolla Drive, mc0505, La Jolla, CA 92093-0505, 858-534-5050, http://www.csda.org/
http://www.csda.org/research/topology/as_core_network/

Copyright (c) 2009 UC Regents
All rights reserved.

This visualization represents macroscopic snapshots of IPv4 and IPv6 Internet topology samples captured in January 2009. The plotting method illustrates both the extensive geographical scope as well as rich interconnectivity of nodes participating in the global Internet routing system.

For the IPv4 map, CAIDA collected data from 33 monitors located in 10 countries on 5 continents. Coordinated by our active measurement infrastructure, Archipelago (Ark), the monitors probed paths toward 7.4 million /24 networks that cover 95% of the routable prefixes seen in the RouteViews Border Gateway Protocol (BGP) Routing tables on 1 January 2009.

For the IPv6 map, CAIDA collected data from 6 Ark monitors located in 4 countries on 2 continents. This subset of monitors probed paths toward 1,491 prefixes which represent 88.6% of

globally routed IPv6 prefixes seen in RouteViews BGP tables on 1 January 2009.

We aggregate this IP-level data to construct IPv4 and IPv6 Internet connectivity graphs at the Autonomous System (AS) level. Each AS approximately corresponds to an Internet Service Provider (ISP). We map each observed IP address to the AS responsible for routing traffic to it, i.e., to the origin (end-of-path) AS for the IP prefix route. We then associate each IP address with its address in the BGP routing tables. For the IPv4 graph, we used the BGP IPv4 routing table provided by RouteViews. For the IPv6 graph, we used the IPv6 routing table collected by RIPE NCC.

The position of each AS node is plotted in polar coordinates (radius, angle), that are calculated as follows.

The outdegree of an AS node is the number of next-hop ASes that we observed accepting our probe traffic as it left this AS. The link color reflects outdegree value, from lowest (blue) to highest (yellow). Toward the center of the graph we have manually labeled some of the highest outdegree ASes with their associated ISPs.

To determine the longitude of an AS, we used the IPv4 BGP table from RouteViews to find a set of announced IPv4 prefixes for each AS. We subdivided each AS into smaller geographic regions and mapped its NetAcuity[®] to a single geographic location in January 2009. We then calculated the AS angle coordinate from the weighted average (by number of IP addresses in each mapped prefix) of the longitude coordinates of all such subdivided prefixes. NetAcuity currently only supports IPv4 mapping, so we use the IPv4-derived locations for ASes in both graphs.

Calculating AS coordinates as described above results in a large number of overlapping nodes (hundreds in the case of the IPv4 graph) which distort the graph's edge. To better visualize so many ASes at the edge, we refined our node placement algorithm to spread out overlapping nodes. This modification creates bulges in the outermost ring of the AS-cone corresponding to longitudes with substantial Internet infrastructure deployment, which also coincides with populous regions of the globe.

The IPv6 graph grew from 486 AS nodes in January 2008 to 515 nodes in January 2009. Over the same period we see an increase in the number of IPv6 ASes from 186 to almost 236. Whether these changes represent actual new AS allocations or result from modifications in our measurement methodology is not clear. Compared with the AS-core graph of January 2008, we observed a westward shift in the position of ISP TelstraClear due to its increased presence (per NetAcuity's mapping) in Australia.

$$\begin{aligned} \text{radius} = 1 - \log & \left(\frac{\text{outdegree}(\text{AS}) + 1}{\text{maximum.outdegree} + 1} \right) \\ \text{angle} = & \left(\frac{\text{longitude of the AS's BGP prefixes}}{\text{metacg}} \right) \end{aligned}$$



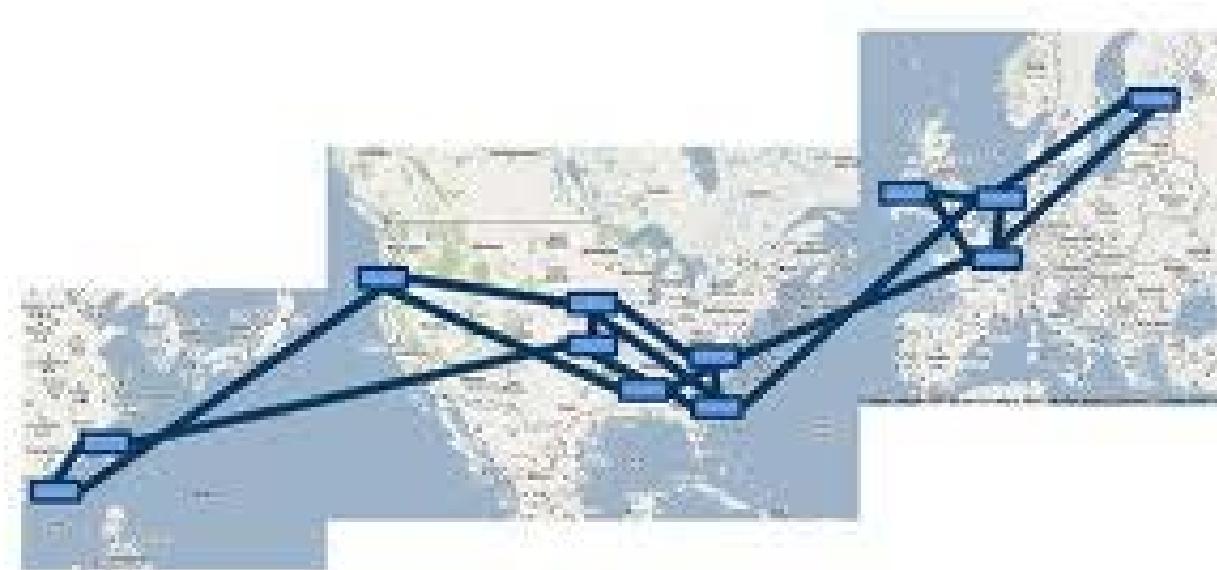
Google数据中心

- Estimated cost of data center: \$600M
- Spent \$2.4B in 2007 on new data centers
- Each data center: 50-100 megawatts of power





Google数据中心





中国互联网

- 一个朝阳行业
 - 起步于模仿，逐步独立创新
 - Yahoo(1994), Sina(1998)
 - Amazon(1995), JD(2004)
 - eBay(1995), Taobao(2003)
 - ICQ(1996), QICQ(1999), QQ(2000)
 - Google(1998), Baidu(2000)
 - Paypal(1998), Alipay(2003)
 - Facebook(2004), Renren(2009)
 - YouTube(2005), Tudou(2005), Youku(2006)
 - Groupon(2008), Meituan(2010)
 -



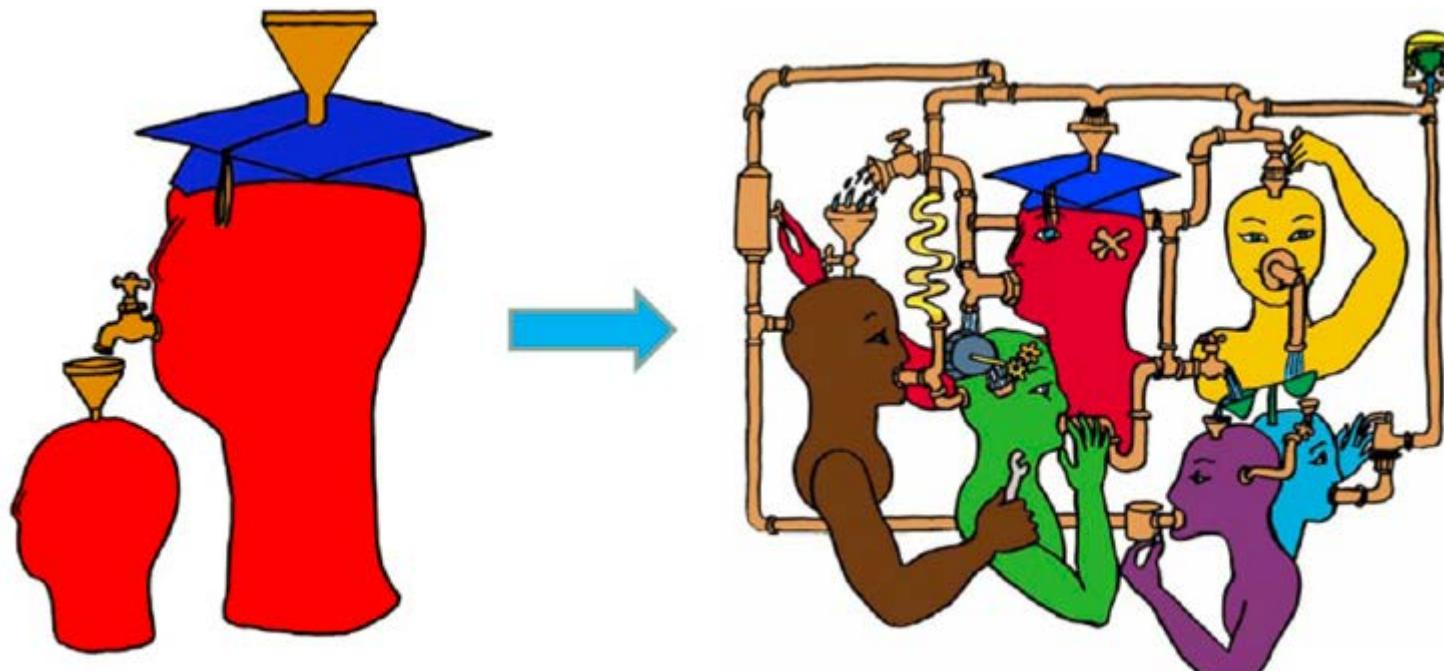
中国互联网

- 一个朝阳行业
 - 拥有全球最多的网民数量，6.5亿
 - 拥有全球领先的互联网公司，BAT (Baidu, Alibaba, Tencent) > 市值3000亿美元
 - 开始出现我国自主的互联网创新产品
 - 迅雷
 - 旺旺
 - 支付宝
 - 余额宝
 - 小米
 - 微信
 -

中国互联网



- China Great Firewall (GFW)
- “大中华局域网”
- 小讲师志愿者
 - 专题：GFW是什么？如何访问Google？如何翻墙？





提纲

- 背景
 - 核心问题: 构造一个网络
- 需求分析
- 网络架构
- 网络性能
- 网络编程
 - 基于套接字
- 互联网简史
- ● 总结



需求分析

我们从网络中期望得到什么？

- 需求 1: 可扩展的连通性
 - 由节点和链路组成的嵌套互联
- 需求 2: 高性价比的连通性
 - 通过统计多路复用方式共享硬件资源
 - 产生分组交换技术
- 需求 3: 提供进程到进程的通信服务
- 需求 4: 网络容易管理



系统设计

网络设计的蓝图是什么？

- 协议
 - 向高层实体提供通信服务
 - 定义与对等实体之间交换信息的格式与含义
- 分层体系结构
 - OSI体系结构
 - Internet体系结构

如何评价网络性能？

- 带宽、时延、时延带宽积、吞吐量



课程内容的组织

第 1 章: 概述

问题: 如何构造一个网络

第 2 章: 直连网络

问题: 物理上相连的主机

第 3 章: 网络互联

问题: 并非所有的网络都是直连的

第 4 章: 高级网络互联

问题: 存在多个网络

第 5 章: 端到端协议

问题: 进程间通信

第 6 章: 拥塞控制和资源分配

问题: 资源分配

第 7 章: 应用

问题: 应用需要自己的协议

第 8 章: 高级主题

- 无线和移动网络
- 多媒体网络
- 网络管理
- 网络安全

基本概念和课程概述



课程内容的组织

第 1 章: 概述

问题: 如何构造一个网络

第 2 章: 直连网络

问题: 物理上相连的主机

第 3 章: 网络互联

问题: 并非所有的网络都是直连的

第 4 章: 高级网络互联

问题: 存在多个网络

第 5 章: 端到端协议

问题: 进程间通信

第 6 章: 拥塞控制和资源分配

问题: 资源分配

第 7 章: 应用

问题: 应用需要自己的协议

第 8 章: 高级主题

- 无线和移动网络
- 多媒体网络
- 网络管理
- 网络安全

需求1的解决方案,
计算机连通导致的问题:
从简单到复杂



课程内容的组织

第 1 章: 概述

问题: 如何构造一个网络

第 2 章: 直连网络

问题: 物理上相连的主机

第 3 章: 网络互联

问题: 并非所有的网络都是直连的

第 4 章: 高级网络互联

问题: 存在多个网络

第 5 章: 端到端协议

问题: 进程间通信

第 6 章: 拥塞控制和资源分配

问题: 资源分配

第 7 章: 应用

问题: 应用需要自己的协议

第 8 章: 高级主题

- 无线和移动网络
- 多媒体网络
- 网络管理
- 网络安全

需求 2 的解决方案,
高性价比的分组交换网络,
路由选择及拥塞控制



课程内容的组织

第 1 章: 概述

问题: 如何构造一个网络

第 2 章: 直连网络

问题: 物理上相连的主机

第 3 章: 网络互联

问题: 并非所有的网络都是直连的

第 4 章: 高级网络互联

问题: 存在多个网络

第 5 章: 端到端协议

问题: 进程间通信

第 6 章: 拥塞控制和资源分配

问题: 资源分配

第 7 章: 应用

问题: 应用需要自己的协议

第 8 章: 高级主题

- 无线和移动网络
- 多媒体网络
- 网络管理
- 网络安全

需求 3 的解决方案,
向不同的应用提供通用服务



课程内容的组织

第 1 章: 概述

问题: 如何构造一个网络

第 2 章: 直连网络

问题: 物理上相连的主机

第 3 章: 网络互联

问题: 并非所有的网络都是直连的

第 4 章: 高级网络互联

问题: 存在多个网络

第 5 章: 端到端协议

问题: 进程间通信

第 6 章: 拥塞控制和资源分配

问题: 资源分配

第 7 章: 应用

问题: 应用需要自己的协议

第 8 章: 高级主题

- 无线和移动网络
- 多媒体网络
- 网络管理
- 网络安全

由 物理层和应用层带来的挑战,
网络使用的问题



挑战: 关于本课程的学习

- ❖ 计算机网络技术发展迅速
- ❖ 课程中只讨论小数据, 而网络工业界需要处理大数据
- ❖ 本课程并不涉及安全、可靠性等问题
- ❖ 大规模开放式在线课程
 - ❖ edX
 - ❖ Coursera (<http://www.coursera.com>)
 - ❖ Udacity
- ❖ 推荐慕课课程
 - ❖ Networks: Friends, Money, and Bytes (Prof. Mung Chiang at Princeton University)
<https://www.coursera.org/learn/friends-money-bytes/>
 - ❖ Software Defined Networking (Prof. Nick Feamster at Princeton University)
<https://www.coursera.org/learn/sdn>

谢谢！



黑晓军

华中科技大学
电子信息与通信学院

Email: heixj@hust.edu.cn

网址: <http://itec.hust.edu.cn/~heixj>



参考资料

- *Chapter 1 in L. L. Peterson and B. S. Davie, Computer Networking: A System Approach (5th edition), Morgan Kaufmann, 2012*
- *Chapter 1 in James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach (6th edition), Pearson Education Inc., 2012*
- 吴功宜, 计算机网络 (第3版), 清华大学出版社, 2011