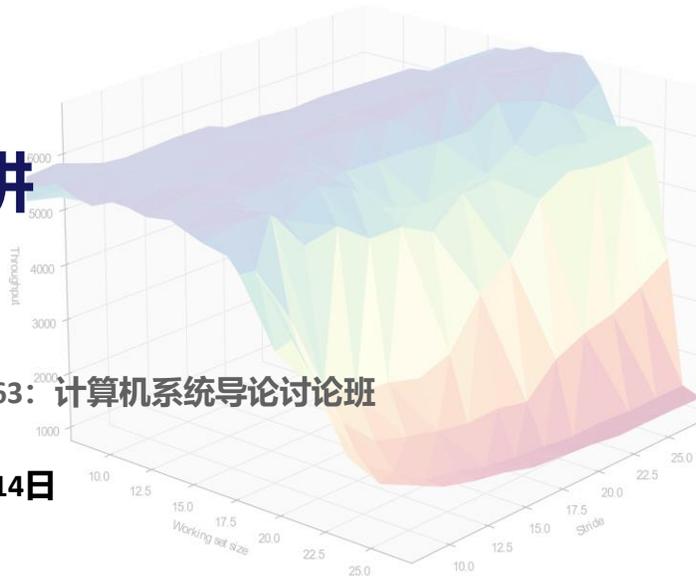


第十讲

PKU 04832363: 计算机系统导论讨论班
王畅
2021年12月14日



Activity 1

轮流回答问题

书本内容梳理

■ 客户端—服务器编程模型（请求—响应）

- 客户端和服务器都是进程，例如以套接字的形式进行读写

■ 网络的层次结构

- 实现上的层次：物理层（电缆）、链路层（以太网、交换机）、网络层（路由器、IP协议）、传输层（TCP、UDP）、应用层（HTTP、SMTP），参看图11-8
- 物理上的层次：局域网（由集线器、交换机构成，HUB不加分辨地转发；网桥）、互联网（一般概念，小写internet，路由器）
- 协议：不同型号设备之间交换数据的机制，包括命名机制（IP地址）和传送机制（发包）。知道几个（有的应用层，有的传输层）：IP、HTTP、POP3、SMTP、FTP、DNS、TCP、UDP
 - 基于连接vs不基于连接
- 图11-7的传送过程，记忆！

交换机一般会比集线器更快，因为不需要广播，而且通常不会共享带宽。

轮流回答问题

■ 下面有关计算机网络概念的叙述中，正确的是

- A. 大写字母的Internet用来描述互联网的一般概念，而小写字母的internet用来描述一种具体的实现，也就是全球IP互联网。
- B. 在一个基于集线器的以太网中，如果往一台主机发送一段数据帧，那么其他主机无法看到这个帧。
- C. IP协议提供基本的命名方法和递送机制，因此我们能够借助IP协议，从一台主机往另一台主机发送包，即使两台主机不在同一个LAN内。
- D. 当一段数据通过路由器，从LAN1被发送到LAN2时，附加的互连网络包头和局域网帧头始终保持不变。

4

C。A反了，Internet是指全球IP互联网，是一个具体的实现。B，集线器是广播的，大家都能看到。D，会发生反复的打包和拆包过程，因此帧头会有变化。

书本内容梳理之IP地址

- **属于网络层**
 - 准确地说，IP地址是用于标识主机的接口，并非主机
- **和传输层合在一起有一族TCP/IP协议栈**
 - 单纯的IP或者UDP是不可靠的传输，TCP则是可靠的（实践上一一定可靠，但不是理论上的可靠）
- **IPv4地址**
 - 32位无符号整数，点分十进制转换
 - IP地址永远是大端法！
 - IP地址转换的相关API使用
 - 特殊IP地址：127.0.0.1、255.255.255.255等

5

IP地址是network adapter，网络适配器的东西，所以一个主机可以有多个IP地址。特别地，路由器一般都有多个IP地址。

255.255.255.255是局域网中用来广播的IP地址。

虽然IP地址struct中只有一个成员，但这是因为要求**必须**有这个成员，而各系统的实现可能有更多字段。

UDP和TCP是传输层的两种重要协议。TCP是有连接的（可以理解就是有listen、accept的过程），且在实践上保证一定可靠（理论上不可能可靠，相关理论分析可以参见拜占庭将军问题）。UDP是无连接的（可以理解就是发送一次），不保证可靠，不管对方是否收到。像FTP、HTTP、SMTP等需要有文件传输的应用层协议一般基于TCP，视频传输、音频传输、DNS等稍有错误影响不大的会用UDP。

书本内容梳理之域名及其解析

- 人类可读可记的地址
- 一个命名的层次结构
 - 根 (/)
 - 第一级域名, 如com、gov、edu
 - 第二级域名, mit.edu中的mit
 - 第三级域名...
- 由DNS服务器维护域名和IP地址的关系
 - nslookup的用法
 - (在大陆) nslookup www.google.com 会发生什么?

6

DNS是基于UDP的应用层协议。由全球DNS服务器的网络维护, DNS服务器也是分布式的层次结构, 即本地服务器会缓存一些DNS数据。

轮流回答问题

- 使用浏览器打开网页www.pku.edu.cn的过程中，下列网络协议中，可能会被用到的网络协议有___个。
 - ① DNS ② TCP ③ IP ④ HTTP

- 假设有一个HTTPS（基于HTTP的一种安全的应用层协议）客户端程序想要通过一个URL连接一个电子商务网络服务器获取一个文件，并且这个服务器URL的IP地址是已知的，以下哪种协议是一定不需要的？
 - A. HTTP
 - B. TCP
 - C. DNS
 - D. SSL/TLS

7

题1：4个。DNS用来解析域名到IP地址，进一步IP也会被使用；TCP用来可靠地传输文件；HTTP是基于TCP的。

题2：C，IP地址是已知的，所以可以直接访问。SSL/TLS是应用层中的一种安全传输的协议。

轮流回答问题

- 关于IP, 以下说法正确的是:
 - A. IP协议提供了可信赖的主机与主机之间的数据包传输能力。
 - B. IP地址的长度固定为32位。
 - C. 一个域名可以映射到多个IP, 但一个IP不能被多个域名映射。
 - D. 一个域名可以不映射到任何IP。

8

D. IP是网络层, 其传输是best effort, 不保证正确。IPv6协议的地址长度为128位, B错误。C则是明显的, IP地址和域名之间属于“关系”但不保证是“映射”, 更不保证是单射或者满射。

轮流回答问题

■ 判断以下说法的正确性：

- A. HTTP协议规定服务器端使用80端口提供服务。
- B. 使用TCP来实现数据传输一定是可靠的。
- C. Internet上的两台的主机要通信必须先建立端到端连接。
- D. 在Linux中只能通过Socket接口进行网络编程。
- E. HUB会把它任意端口上接收到的帧只转发到它的目的地去。
- F. 当在不同的LAN中的主机A和主机B通信的过程中，他们的数据包中的LAN frame header不会变化。
- G. 162.105.0.0是一个B类地址。
- H. 同一台主机每次进入相同的网络，通过动态地址分配的到的IP地址总是相同的。
- I. TCP是一种可靠的无连接协议。
- J. UDP是一种不可靠的无连接协议。
- K. Web浏览器与web服务器通信采用的协议是HTML。
- L. 数字数据只能通过数字信号传输。

9

错误，一般用80，也可以用其他的，比如8888、8080

正确，实际上保证可靠

错误，有一些协议是无连接的，比如UDP

错误，可以通过更加底层的适配器等接口进行操作

正确，属于HUB重要特性之一

错误，会发生反复的打包和拆包过程，因此帧头会有变化

正确，此题超纲

错误，会通过DHCP协议获得地址，有可能不同

错误，TCP有连接

正确，UDP不可靠，无连接

错误，HTML是内容的编码方式

错误，也可以用模拟信号传输，此题超纲

轮流回答问题

- 如果两个局域网高层分别采用TCP/IP协议和SPX/IPX协议，那么可以选择的互连设备应是
 - A. 网桥
 - B. 集线器
 - C. 路由器
 - D. 交换机

10

C, 因为协议不同, 所以需要路由器来完成工作。另一方面, 只有路由器属于网络层, 能够handle IP协议。

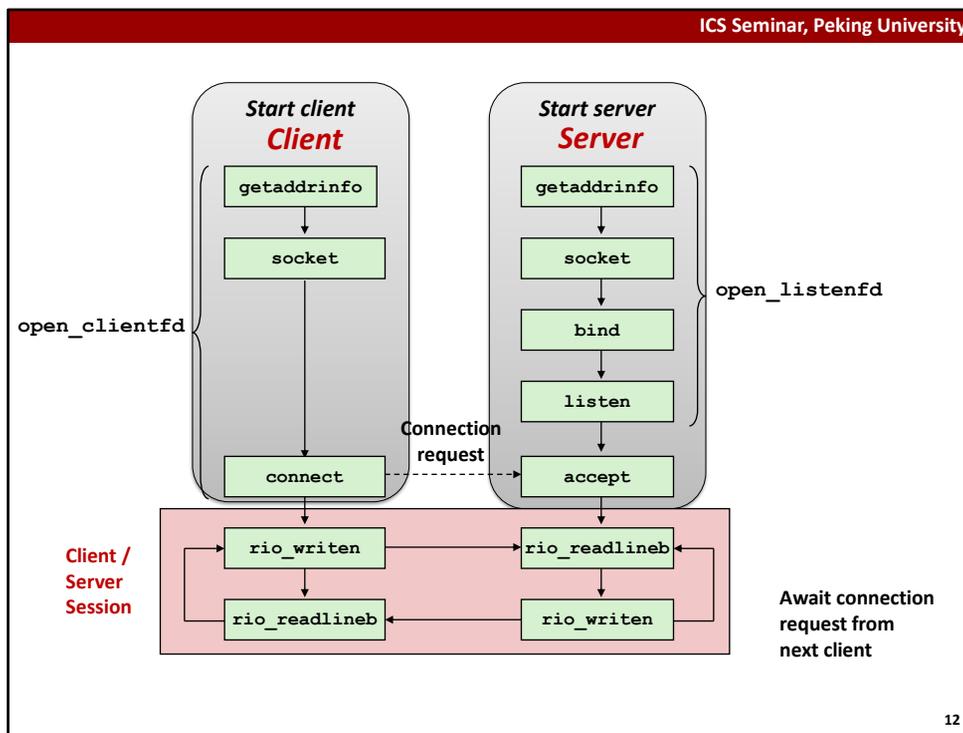
书本内容梳理之套接字

- 套接字就是有相应描述符的打开文件
 - 网络应用=(进程, 端口号, 协议类型)
- 知名端口: 22、23、25、80、443
- 各种API的使用
 - 仔细阅读书P657~664的内容, 记住细节!
 - 知道创建一个网络服务器和客户端的基本流程, 会填空!

11

关于端口的说明: 1024以下是系统保留的, 一般没有管理员权限不能监听。严格地说, 一个应用是一个3-tuple, 协议类型不同有可能可以在同一个端口内运行 (ICS中不考虑); 而一个连接则是一个5-tuple (双方需要使用相同的协议进行交互, ICS认为4-tuple)。

知名端口。22: ssh; 23: telnet; 25: SMTP; 80: HTTP; 443: HTTPS。



此图要背下来，还要明确顺序以及各过程调用API的参数。

轮流回答问题

- 下面有关套接字接口的叙述中，错误的是
 - A. 套接字接口常常被用来创建网络应用。
 - B. Windows 10系统没有实现套接字接口。
 - C. `getaddrinfo()`和`getnameinfo()`可以被用于编写独立于特定版本的IP协议的程序。
 - D. `socket()`函数返回的描述符，可以使用标准Unix I/O函数进行读写。

13

B, 系统一般都实现了套接字接口。其他都是正确的。注意D不够准确, 其返回的描述符是半打开的, 并不是能马上读写。

轮流回答问题

- **Suppose an HTTP server is running on a host at port 80. Can we also run another application that uses UDP at port 80? (1pt) Why? (4pts)**
 - A. Yes, because the port number is used by TCP and UDP to de-multiplex data, there is no conflict between the TCP and UDP applications using the same port number.
 - B. Yes, as long as the UDP application also uses HTTP so de-multiplexing is not needed.
 - C. No, since HTTP uses UDP, two applications running at port 80 are not allowed.
 - D. No, because the port number is used to de-multiplex data, there will be conflict if two applications use the same port number.

14

The answer is A. We must note that inside an IP datagram, there's a field called upper-layer protocol. As long as the upper-layer protocol is different, the datagram will be handed over to different protocols to be further processed. The port number is only used by a particular transport-layer protocol to de-multiplex data, and there's no conflict in the question here because there are two different (protocol, IP, port number) tuples.

书本内容梳理之HTTP和Proxy

■ Web内容

- MIME类型, 例如text/html
- 静态和动态内容

■ HTTP协议

- 服务器一般运行在80端口 (但不是必须)
- 请求头、请求负载、响应头、响应负载
- 状态码 (200、404、502、403、301)
- 动态内容用POST来请求
- telnet www.bbs.pku.edu.cn

15

了解HTTP协议的详细内容 (看书) 。

注: 现今bbs不能用23端口telnet访问了, 可以ssh访问 (你的用户名@bbs.pku.edu.cn) 。

telnet

```
telnet sina.com.cn 80
Trying 123.126.45.205...
Connected to sina.com.cn.
Escape character is '^]'.
GET
HTTP/1.1 400 Bad Request
Server: nginx
Date: Tue, 14 Dec 2021 05:35:20 GMT
Content-Type: text/html
Content-Length: 150
Connection: close
X-Via-CDN: f=edge,s=cnc.beixian.union.197.nb.sinaedge.com,c=111.205.230.100;

<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx</center>
</body>
</html>
Connection closed by foreign host.
4s 13:35:30
```

16

HTTP请求的正确格式参看书本（上面是不对的，所以返回状态400）

-

Activity 2

问题求解

查看内存映射

```

> sudo cat /proc/423/maps
55d5caf1000-55d5caf4000 r--p 00000000 08:10 48857          /usr/bin/ping
55d5caf4000-55d5caf6000 r-xp 00003000 08:10 48857          /usr/bin/ping
55d5caf6000-55d5cad02000 r--p 0000d000 08:10 48857          /usr/bin/ping
55d5cad02000-55d5cad03000 r--p 00010000 08:10 48857          /usr/bin/ping
55d5cad03000-55d5cad04000 rw-p 00011000 08:10 48857          /usr/bin/ping
55d5cad04000-55d5cad27000 rw-p 00000000 00:00 0
55d5caf1000-55d5cb002000 rw-p 00000000 00:00 0
7f856b205000-7f856b207000 r--p 00000000 08:10 22848          /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
7f856b207000-7f856b209000 r-xp 00002000 08:10 22848          /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
7f856b209000-7f856b20c000 r--p 00006000 08:10 22848          /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
7f856b20c000-7f856b20d000 r--p 00006000 08:10 22848          /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
7f856b20d000-7f856b20e000 rw-p 00007000 08:10 22848          /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
7f856b20e000-7f856b211000 r--p 00000000 08:10 22862          /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f856b211000-7f856b218000 r-xp 00003000 08:10 22862          /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f856b218000-7f856b21a000 r--p 0000a000 08:10 22862          /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f856b21a000-7f856b21b000 r--p 0000b000 08:10 22862          /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f856b21b000-7f856b21c000 rw-p 0000c000 08:10 22862          /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f856b21c000-7f856b222000 rw-p 00000000 00:00 0
7f856b22e000-7f856b260000 r--p 00000000 08:10 23782          /usr/lib/locale/C.UTF-8/LC_CTYPE
7f856b260000-7f856b261000 r--p 00000000 08:10 23788          /usr/lib/locale/C.UTF-8/LC_NUMERIC
7f856b261000-7f856b262000 r--p 00000000 08:10 23791          /usr/lib/locale/C.UTF-8/LC_TIME
7f856b262000-7f856b3d5000 r--p 00000000 08:10 23781          /usr/lib/locale/C.UTF-8/LC_COLLATE
7f856b3d5000-7f856b3d6000 r--p 00000000 08:10 23786          /usr/lib/locale/C.UTF-8/LC_MONETARY
7f856b3d6000-7f856b3d7000 r--p 00000000 08:10 23785          /usr/lib/locale/C.UTF-8/LC_MESSAGES/SYS_LC_MESSAGES
7f856b3d7000-7f856b6bd000 r--p 00000000 08:10 2303          /usr/lib/locale/locale-archive
7f856b6bd000-7f856b6bf000 rw-p 00000000 00:00 0
7f856b6bf000-7f856b6c3000 r--p 00000000 08:10 35235          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f856b6c3000-7f856b6d6000 r-xp 00004000 08:10 35235          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f856b6d6000-7f856b6e0000 r--p 00017000 08:10 35235          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f856b6e0000-7f856b6e1000 r--p 00020000 08:10 35235          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
7f856b6e1000-7f856b6e2000 rw-p 00021000 08:10 35235          /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0

```

18

第一列是虚拟地址，第二列是权限（`rwX`，是否可读可写可执行；`p/s` 映射是私有的还是共享的）。

考点S: ECF、缓存、文件和虚存的综合考题

- 熟练掌握地址翻译的整个过程 (结合cache、TLB, 不要忘记任何一个环节)
 - 主要考怎么用页表, 但不能忘记其他内容
- 内存映射和共享对象的基本机制
 - 进程地址空间独立, fork出来的进程, 虚拟内存映射到的位置暂时是同一个位置, 直到触发COW机制
- 用户态, 只能访问虚拟地址
- Section 16 2、5
- Section 17 4、8、11、13、16、20

19

这里需要澄清一下, 我们在课本上学的地址翻译过程是抽象的, 和具体的系统还是有一些差别, 因此在做题的过程中, 虽然接触了很多具体的东西, 但不要混淆了。

一般而言, 系统都希望每级每个页表恰好占一页, 这样便于设计页表中物理地址的条目。以IA-32的页表 (VPN两级各10位, VPO 12位) 为例, 其31~12位是20位物理页号, 剩下有一些位表示读写权限和dirty位 (dirty位很重要, 写发生后马上要修改!)。系统会先根据VPN在TLB中搜索 (注意只有最后一级页表条目会在TLB中!), 如果有就直接根据其内容拼出物理地址。否则, 第一次系统先用 $CR3 + VPN1 * 4$ 得到一级页表条目, 然后从一级页表条目中查出二级页表的物理页号, 再从用物理页号 + $VPN2 * 4$ 得到二级页表条目, 再从二级页表条目中查出真正的物理页号, 后者和VPO拼在一起, 得到最终的物理地址。

这个过程在x86不同的系统中会有不同, 但是大同小异。

any
questions?

Thanks & 感谢观看