计算机网络

Computer Network

5

许可证软件

实验课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 炜

（国家示范性软件学院） 博士，副教授

School of Informatics Dr. Wei Huang

知识框架

- 项目信息
- 需求分析
- 搭建框架
- 版本迭代

项目信息

- 通过完成项目，掌握应用层传输的原理；了解传输过程中传输层协议选用、应用层协议设计和协议开发等概念。

需求分析

- 某软件A需要一个许可证程序来保证合法运行。用户提供支付费用，获得一个若干人的许可证。
  - 1、某组织管理员在购买许可证时，输入用户名、口令和许可证类型，许可证程序返回一个由10个数字组成的序列号。
  - 2、该组织的用户第一次使用软件A时，输入序列号。
  - 3、该组织用户运行软件时，向许可证服务器发送验证。
  - 4、许可证服务器查询得到该序列号的使用人数，如果未到达上限，则返回授权指令；否则，返回拒绝指令。
  - 5、软件A得到授权指令，允许用户使用软件。否则，提示用户稍后再试，退出程序。

需求分析

- 当软件A或非正常退出（崩溃被其它程序中止）时，许可证服务器应在扣除使用人数时剔除它。可以使软件A定期（如：30分钟）向服务器报告其状态，超过一定时间没有收到报告时，则认定崩溃。
- 当许可证服务器崩溃时，软件A应能在重新启动时恢复。许可证服务器重启后，如果新的软件A前来连接，服务器不可以因接受其连接而拒绝已认证用户的连接

搭建框架

- 第一步需要解决的问题
  - 程序 = 算法 + 数据结构
- 基本流程
  - 用户U运行被许可的程序P；
  - 程序P向服务器S请求许可；
  - 服务器检查当前运行程序P的用户数；
  - 如果上限未达到，S给予许可，程序P运行；
  - 如果达到上限，S拒绝许可，程序告诉用户U稍后再试。

搭建框架

- 数据
  - 许可证票据：整数
    - 进程号 + 票据编号 ，如：6589.3
  - 服务器与客户
    - 许可证服务器是服务器程序，程序P是客户端程序
  - 协议（数据格式：命令（四字母）+进程号）

	客户端	服务器
获取钥匙	HELO *mypid*	TICK *tickid* 或 FAIL no tickets
归还钥匙	GBYE *ticketid*	THNX *message*

搭建框架

- 数据
  - 通信系统
    - 在C/S之间传送的方式：UDP
  - 数据结构
    - 用一个表（二维数组）表示票据

ticket_id	process_id
1	1234
2	0
3	5678

  - SocketAPI

传输层协议选择

- **Internet传输数据的错觉**
  - 看上去是连续的、无缝的，实际上是分割的独立数据包
- **流模式：TCP协议**
  - 负责数据的分割、排序、重组等工作
  - 大文件、报文丢失影响大
- **数据报模式：UDP协议**
  - 用户自行处理
- **最终选择：UDP协议**

接收数据报文

- 接收端程序

```
int main(int ac, char *av[]) {
        ...
    /*  get a socket and assign it a port number */
        if( (sock = make_dgram_server_socket(port)) == -1 )
                oops("cannot make socket",2);
    /* receive messaages on that socket */
        saddrlen = sizeof(saddr);
        while( (msglen = recvfrom(sock,buf,BUFSIZ,0,
                 (struct sockaddr *) &saddr,&saddrlen))>0 ) {
                buf[msglen] = '\0';
                printf("dgrecv: got a message: %s\n", buf);
                say_who_called(&saddr);
        }
        return 0;
}
```

发送数据报文

- 发送端程序

```
int main(int ac, char *av[]) {
        ...
    /* get a datagram socket */
    if( (sock = make_dgram_client_socket()) == -1 )
            oops("cannot make socket",2);
    /* combine hostname and port number of destination into an address */
    if ( make_internet_address(av[1], atoi(av[2]), &saddr) == -1 )
            oops("make addr",4);
    /* send a string through the socket to that address */
    if ( sendto(sock, msg, strlen(msg), 0,
          (struct sockaddr *)&saddr,sizeof(saddr)) == -1)
        oops("sendto failed", 3);
    return 0;
}
```

辅助程序

- 生成服务器报文套接字

```
int make_dgram_server_socket(int portnum) {
    ...
    sock_id = socket(PF_INET, SOCK_DGRAM, 0);  /* get a socket */
    if (sock_id == -1) return -1;
    /** build address and bind it to socket **/
    strncpy(hostname, "0.0.0.0", HOSTLEN);
    make_internet_address(hostname, portnum, &saddr);

    if (bind(sock_id, (struct sockaddr*) & saddr,
        sizeof(saddr)) != 0)
        return -1;

    return sock_id;
}
```

辅助程序

- 生成服务器端报文IP地址

```
int make_internet_address(char* hostname, int port, struct sockaddr_in*
addrp)
/*
 * constructor for an Internet socket address, uses hostname and port
 *   (host,port) -> *addrp
 */
{
    struct hostent* hp;

    bzero((void*)addrp, sizeof(struct sockaddr_in));
    hp = gethostbyname(hostname);
    if (hp == NULL) return -1;
    bcopy((void*)hp->h_addr, (void*)&addrp->sin_addr, hp->h_length);
    addrp->sin_port = htons(port);
    addrp->sin_family = AF_INET;
    return 0;
}
```

初步编译和测试

- 编译和测试命令

```
$ cc dgrecv.c dgram.c -o dgrecv
$ ./dgrecv 4444&
[1] 19383
$ cc dgsend.c dgram.c -o dgsend
$ ./dgsend host2 4444 "test 123"
dgrecv: got a message: testing 123
    from: 127.0.0.1:1041
$ ps
PID      TTY      TIME         CMD
19383    pts/3    00:00:00     dgrecv
$
```

接收应答

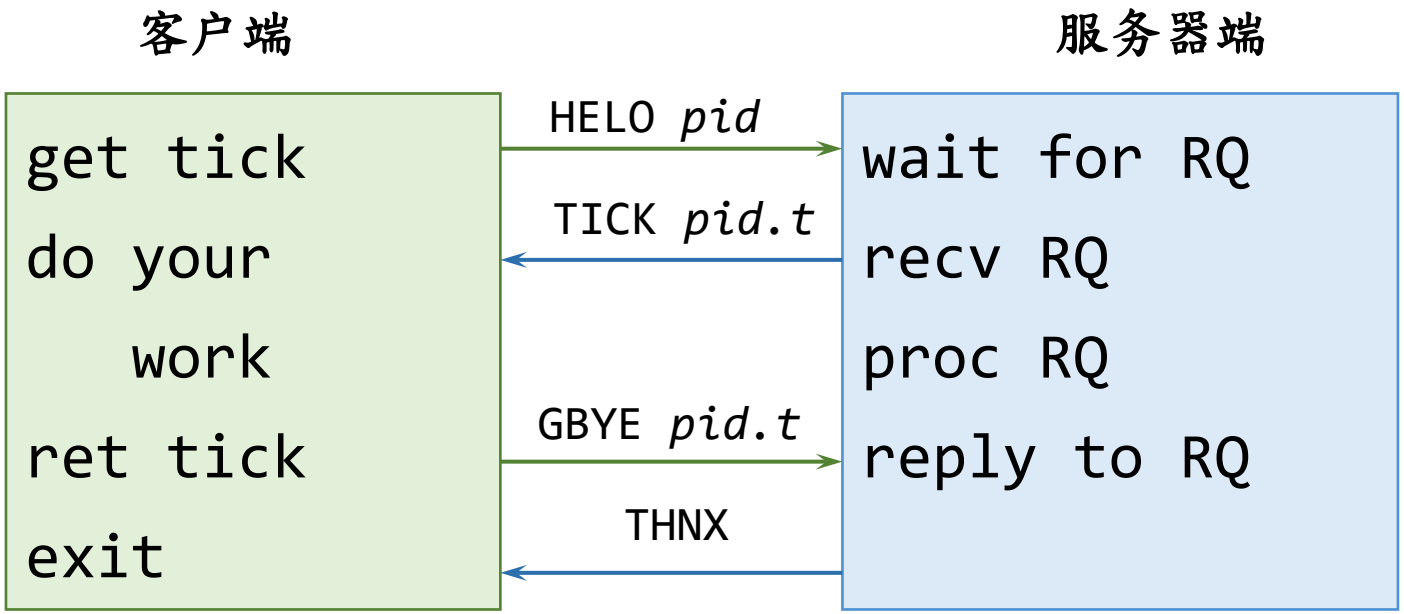
- 接收来自客户的消息，并发送应答

```
void reply_to_sender(int sock, char* msg, struct sockaddr_in*
addrp, socklen_t len)
{
    char reply[BUFSIZ + BUFSIZ];

    sprintf(reply, "Thanks for your %d char message\n",
strlen(msg));
    sendto(sock, reply, strlen(reply), 0,
        (struct sockaddr*) addrp, len);
}
```

许可证服务器版本 1.0

- 运行流程及其交互过程



许可证服务器版本 1.0

- 客户端和服务器端文件组成
  - 短文件：包含main函数
  - 长文件：包含票据管理函数

版本1：客户端

- 客户端版本1

```
/*****************************************************************
 * lclnt1.c
 * License server client version 1
 *  link with lclnt_funcs1.o dgram.o
 */
#include <stdio.h>
int main(int ac, char* av[]) {
    setup();
    if (get_ticket() != 0)
        exit(0);
    do_regular_work();
    release_ticket();
    shut_down();
}
```

版本1：客户端

- 客户端版本1

```
/*********************************************************************
 * do_regular_work  the main work of the application goes here
 */
do_regular_work() {
    printf("SuperSleep version 1.0 Running - Licensed Software\n");
    sleep(10);/* our patented sleep algorithm */
}
```

版本1：功能

- 获取票据

```
/*****************************************************************
 * get_ticket
 * get a ticket from the license server
 * Results: 0 for success, -1 for failure
 */
int get_ticket() {
    char* response;
    char buf[MSGLEN];

    if (have_ticket)/* don't be greedy */
        return(0);
    sprintf(buf, "HELO %d", pid);/* compose request*/
    if ((response = do_transaction(buf)) == NULL)
        return(-1);
```

版本1：功能

- 获取票据

```
    /* parse the response and see if we got a ticket.
     *   on success, the message is: TICK ticket-string
     *   on failure, the message is: FAIL failure-msg
     */
    if (strncmp(response, "TICK", 4) == 0) {
        strcpy(ticket_buf, response + 5);/* grab ticket-id */
        have_ticket = 1;/* set this flag  */
        narrate("got ticket", ticket_buf);
        return(0);
    }
    if (strncmp(response, "FAIL", 4) == 0)
        narrate("Could not get ticket", response);
    else
        narrate("Unknown message:", response);
    return(-1);
} /* get_ticket */
```

版本1：功能

- 处理事务

```
/*****************************************************************
 * do_transaction
 * Send a request to the server and get a response back
 * IN  msg_pmessage to send
 * Results: pointer to message string, or NULL for error
 * NOTE: pointer returned is to static storage
 * overwritten by each successive call.
 * note: for extra security, compare retaddr to serv_addr (why?)
 */
char* do_transaction(char* msg)
{
    static char buf[MSGLEN];
    struct sockaddr retaddr;
    socklen_t       addrlen;
    int ret;
```

版本1：功能

- 处理事务

```
    ret = sendto(sd, msg, strlen(msg), 0, &serv_addr, serv_alen);
    if (ret == -1) {
        syserr("sendto");
        return(NULL);
    }

    /* Get the response back */
    ret = recvfrom(sd, buf, MSGLEN, 0, &retaddr, &addrlen);
    if (ret == -1) {
        syserr("recvfrom");
        return(NULL);
    }

    /* Now return the message itself */
    return(buf);
} /* do_transaction */
```

版本1：功能

- 释放票据

```
/*************************************************************
 * release_ticket
 * Give a ticket back to the server
 * Results: 0 for success, -1 for failure
 */
int release_ticket() {
    char buf[MSGLEN];
    char* response;

    if (!have_ticket)/* don't have a ticket*/
        return(0);/* nothing to release*/

    sprintf(buf, "GBYE %s", ticket_buf);/* compose message*/
    if ((response = do_transaction(buf)) == NULL)
        return(-1);
```

版本1：功能

- 释放票据

```
    /* examine response
     * success: THNX info-string
     * failure: FAIL error-string
     */
    if (strncmp(response, "THNX", 4) == 0) {
        narrate("released ticket OK", "");
        return 0;
    }
    if (strncmp(response, "FAIL", 4) == 0)
        narrate("release failed", response + 5);
    else
        narrate("Unknown message:", response);
    return(-1);
} /* release_ticket */
```

版本1：规则

- 规则
  - 产生短请求、发送消息给服务器、等待服务器应答
  - 检查应答、根据应答采取行动
- get_ticket
  - 通过发送命令HELO请求票据
  - 服务器发送TICK接受请求，FAIL拒绝请求
- release_ticket
  - 通过发送GBYE返回票据。如果票据合法，服务器返回THNX作为应答，否则，发送FAIL消息。

版本1：服务器端版本

- 服务器版本
  - 主函数是一个死循环
  - 接收客户请求
  - 处理请求
  - 发送应答

版本1：功能

- 处理请求
  - 请求由4个字符带一个参数构成。
  - 服务器先检查命令，调用对应的函数。
  - 即使命令不合法，服务器必须发送应答，否则会一直阻塞

```
/**********************************************************************
 * handle_request(request, clientaddr, addrlen)
 *   branch on code in request
 */
handle_request(char* req, struct sockaddr_in* client, socklen_t addlen)
{
    char* response;
    intret;
```

版本1：功能

- 处理请求

```
    /* act and compose a response */
    if (strncmp(req, "HELO", 4) == 0)
        response = do_hello(req);
    else if (strncmp(req, "GBYE", 4) == 0)
        response = do_goodbye(req);
    else
        response = "FAIL invalid request";

    /* send the response to the client */
    narrate("SAID:", response, client);
    ret = sendto(sd, response, strlen(response), 0,
        (struct sockaddr*) client, addlen);
    if (ret == -1)
        perror("SERVER sendto failed");
}
```

版本1：功能

- 处理HELO命令

```
/*****************************************************************
 * do_hello
 * Give out a ticket if any are available
 * IN  msg_pmessage received from client
 * Results: ptr to response
 *    NOTE: return is in static buffer overwritten by each call
 */
char* do_hello(char* msg_p)
{
    int x;
    static char replybuf[MSGLEN];

    if (num_tickets_out >= MAXUSERS)
        return("FAIL no tickets available");

    /* else find a free ticket and give it to client */
```

版本1：功能

- 处理HELO命令

```
    for (x = 0; x < MAXUSERS && ticket_array[x] != TICKET_AVAIL; x++)
        ;

    /* A sanity check - should never happen */
    if (x == MAXUSERS) {
        narrate("database corrupt", "", NULL);
        return("FAIL database corrupt");
    }

    /* Found a free ticket.  Record "name" of user (pid) in array.
     * generate ticket of form: pid.slot
     */
    ticket_array[x] = atoi(msg_p + 5); /* get pid in msg */
    sprintf(replybuf, "TICK %d.%d", ticket_array[x], x);
    num_tickets_out++;
    return(replybuf);
} /* do_hello */
```

版本1：功能

- 处理GBYE命令
  - 服务器将票据列表和签出列表中进行比较
  - 如果一致，服务器清除该项并致谢
  - 如果不一致，则一定是发生了错误（伪造票据问题）

```
/**********************************************************************
 * do_goodbye
 * Take back ticket client is returning
 * IN  msg_pmessage received from client
 * Results: ptr to response
 */
char* do_goodbye(char* msg_p)
{
    int pid, slot;/* components of ticket*/
```

版本1：功能

- 处理GBYE命令

```
    /* The user's giving us back a ticket.  First we need to get
     * the ticket out of the message, which looks like:

     * GBYE pid.slot
     */
    if ((sscanf((msg_p + 5), "%d.%d", &pid, &slot) != 2) ||
        (ticket_array[slot] != pid)) {
        narrate("Bogus ticket", msg_p + 5, NULL);
        return("FAIL invalid ticket");
    }
    /* The ticket is valid.  Release it. */
    ticket_array[slot] = TICKET_AVAIL;
    num_tickets_out--;

    /* Return response */
    return("THNX See ya!");
} /* do_goodbye */
```

版本1：测试

- 测试版本1

```
$ cc lserv1.c lserv_funcs.c dgram.c -o lserv1
$ ./lserv1&
[1] 25738
$ cc lclnt1.c lclnt_funcs1.c dgram.c -o lclnt1
$ ./lclnt1 & ./lclnt1 & ./lclnt1 & ./lclnt1 &
    SERVER:GOT:HELO 25912 (10.200.75.200:1053)
    SERVER:SAID:TICK 25912 (10.200.75.2000:1053)
CLIENT[25912]:got ticket 25912.0
SuperSleep version 1.0 Running - Licensed Software
    SERVER:GOT:HELO 25913 (10.200.75.200:1055)
    SERVER:SAID:TICK 25913 (10.200.75.2000:1055)
CLIENT[25912]:got ticket 25913.1
```

版本1：测试

- 测试版本1

```
SuperSleep version 1.0 Running – Licensed Software
    SERVER:GOT:HELO 25914 (10.200.75.200:1059)
    SERVER:SAID:FAIL no tickets available (10.200.75.2000:1059)
CLIENT[25912]:got ticket 25913.1
    SERVER:GOT:GBYE 25915.2 (10.200.75.200:1055)
    SERVER:SAID:THNX See ya! (10.200.75.2000:1055)
CLIENT[25915]:released ticket OK
```

处理现实的问题

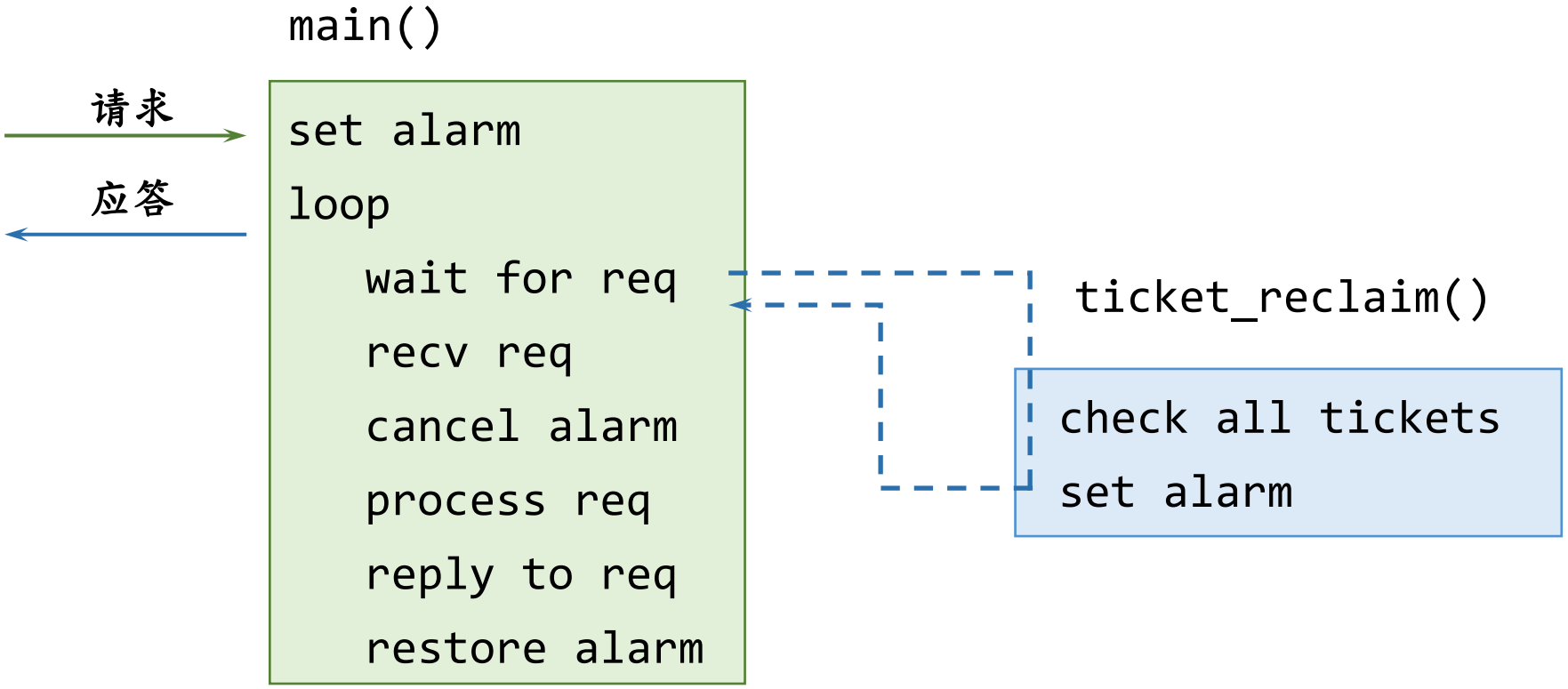
- 许可证服务器能很好地工作
  - 前提是所有进程是正常工作的。
- 软件应该解决问题
  - 如果SuperSleep程序被另外一个用户杀死，该如何？
  - 程序发送段存取错误而被内核杀死，该如何？
  - 占用的票据该如何处理？
  - 如果服务器崩溃了怎么办？
  - 在服务器重启后又将发生什么呢？

处理客户端的崩溃

- 如果客户端崩溃了，客户将不会归还票据
- 定期检查票据数组，确认其中的每个进程是否活着
  - 如果服务器已经不存在，可以将该进程从数组中去除
  - 释放占用其中的票据
  - 检查得越来越频繁，数组的精确性越高

处理客户端的崩溃

- 收回丢失的票据：调度
  - 使用alarm和signal技术来周期地调用一个函数。



处理客户端的崩溃

- 收回丢失的票据：调度
  - 使用alarm和signal技术来周期地调用一个函数。
  - 同时处理两件事情的程序时，要考虑函数间冲突
    - 处理请求的同时，被信号调用触发回收丢失票据的函数
    - 两个函数共享变量或数据结构吗？是的！
    - 发放票据：修改签出列表；释放票据：也需要修改签出列表
    - 处理请求的时候要关闭alarm
- 收回丢失的票据：编程
  - 服务器希望回收不存在进程的票据，如何判断进程活着？
  - 用popen运行ps，然后从输出中找PID

处理客户端的崩溃

- 收回丢失的票据：编程

```
/************************************************************************
 * ticket_reclaim
 * go through all tickets and reclaim ones belonging to dead processes
 * Results: none
 */
void ticket_reclaim() {
    for (i = 0; i < MAXUSERS; i++) {
        if ((ticket_array[i] != TICKET_AVAIL) &&
            (kill(ticket_array[i], 0) == -1) && (errno == ESRCH)) {
            /* Process is gone - free up slot */
            sprintf(tick, "%d.%d", ticket_array[i], i);
            narrate("freeing", tick, NULL);
            ticket_array[i] = TICKET_AVAIL;
            num_tickets_out--;
        }
    }
    alarm(RECLAIM_INTERVAL);/* reset alarm clock */
}
```

处理客户端的崩溃

- 主函数加调度收回票据函数，正常操作中关闭alarm

```
int main(int ac, char* av[])
{
    struct sockaddr client_addr;
    socklen_t addrlen;
    char      buf[MSGLEN];
    int  ret, sock;
    void  ticket_reclaim();/* version 2 addition */
    unsigned  time_left;

    sock = setup();
    signal(SIGALRM, ticket_reclaim); /* run ticket reclaimer */
    alarm(RECLAIM_INTERVAL); /* after this delay     */

    while (1) {
        addrlen = sizeof(client_addr);
        ret = recvfrom(sock, buf, MSGLEN, 0,
            (struct sockaddr*) & client_addr, &addrlen);
```

处理客户端的崩溃

- 主函数加调度收回票据函数，正常操作中关闭alarm

```
        if (ret != -1) {
            buf[ret] = '\0';
            narrate("GOT:", buf, &client_addr);
            time_left = alarm(0);
            handle_request(buf, &client_addr, addrlen);
            alarm(time_left);
        }
        else if (errno != EINTR)
            perror("recvfrom");
    }
}
```

处理客户端的崩溃

- 现在可以周期性地检查票据了
- 只在列表满了有客户请求被拒绝的时候检查更好吗？

处理服务器崩溃

- 服务器崩溃通常有两个严重后果
  - 签出列表丢失，失去进程持有票据的记录
  - 新客户无法运行，因为分发程序已经不存在
- 重启服务使得新的客户可以运行，但会有两个新问题
  - 重启后票据数组是空的，服务器含有未被取走的票据列表
  - 如果重启可以给其它客户发送许可，重启就制造了票据
  - 旧的服务器票据客户归还票据时，将被认为是伪造的

处理服务器崩溃

- 票据验证
  - 客户周期性向服务器发送票据副本
  - 服务器检查签出列表
    - 如果为空，则认为是先前实例赋予的，加入列表中
    - 客户提供票据来验证，签出列表重新被填入
  - 重建签出列表解决了表丢失问题，但导致其他问题
    - 新客户在表重建好之前请求票据，服务器可能分发一个已给予其他客户的票据给该客户
    - 持有旧的票据的客户在验证时，服务器会拒绝它。
  - 服务器拒绝表中没有的票据，要求客户申请新的，更好？

处理服务器崩溃

- 协议中增加验证
  - 新的事务
    - **CLIENT:** `VALD tickid`
    - **SERVER:** `GOOD` **or** `FAIL invalid ticket`
  - 客户端增加验证
    - 10秒的睡眠时间分割成两个5秒，期间进行验证

处理服务器崩溃

- 服务器增加票据验证

```
/*************************************************************************
 * do_validate
 * Validate client's ticket
 * IN  msg_pmessage received from client
 * Results: ptr to response
 */
static char* do_validate(char* msg) {
    int pid, slot;          /* components of ticket */

    /* msg looks like VAD pid.slot - parse it and validate */
    if (sscanf(msg + 5, "%d.%d", &pid, &slot) == 2 && ticket_array[slot] ==
pid)
        return("GOOD Valid ticket");
    /* bad ticket */
    narrate("Bogus ticket", msg + 5, NULL);
    return("FAIL invalid ticket");
}
```

测试版本2

- 测试包含中止客户和服务器，以及重启服务器
- 为便于测试，客户睡眠时间为两个15分钟
- 服务器每5秒尝试回收票据

分布式许可证服务器

- 当它们运行在不同机器上，会有问题吗？
- 问题1：重复进程ID
  - 扩展票据表项，增加标识主机
- 问题2：回收票据
  - 服务器通过调用kill命令向客户回收票据
  - kill发送信号0给持有票据的进程
  - 服务器知道客户运行的主机
  - 为什么不在每台机器上运行1个服务器的实例？
  - 本地服务器解决了向主机发送信号的问题，又带来新问题

分布式许可证服务器

- 问题3：主机崩溃
  - 如果其中1台机器停止运行会发生什么？
  - 主服务器如果还在运行，如何收回票据？
  - 客户端如何验证票据？
  - 如果主服务器停止运行，谁来分发票据？
  - 方法1：客户端服务器和中央服务器通信
  - 方法2：每个客户都和中央服务器通信
  - 方法3：客户服务器和客户服务器通信

计算机网络

Computer Network

5

谢谢观看

实验课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 炜

（国家示范性软件学院） 博士，副教授

School of Informatics Dr. Wei Huang