

组织：中国互动出版网 (<http://www.china-pub.com/>)

RFC 文档中文翻译计划 (<http://www.china-pub.com/compters/emook/aboutemook.htm>)

E-mail: ouyang@china-pub.com

译者：王安鹏 (anpengwang anpengwang@263.net)

译文发布时间：2001-5-23

版权：本中文翻译文档版权归中国互动出版网所有。可以用于非商业用途自由转载，但必须保留本文档的翻译及版权信息。

Network Working Group

J. Romkey

Request for Comments: 1055

June 1988

在串行线路上传输 IP 数据报的非标准协议

(RFC1055 A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP)

目录

简介	1
历史 (HISTORY)	1
可用性 (AVAILABILITY)	2
协议 (PROTOCOL)	2
缺陷 (DEFICIENCIES)	2
SLIP 驱动程序 (SLIP DRIVERS)	3

简介

TCP/IP 协议组运行在各种各样的网络媒介上：IEEE 802.3（以太网）和 802.5（令牌环）局域网（LAN）、X.25 线路、卫星链路以及串行线路。其中许多网络已经有 IP 分组的标准封装格式，但没有用于串行线路的标准。SLIP（串行线路 IP）目前已成为事实上的标准，广泛地用于在点对点串行连接上运行 TCP/IP。这并不是一个 Internet 标准，本备忘录的发布不受限制。

历史 (HISTORY)

SLIP 源于 80 年代初期的 3COM UNET TCP/IP 实现。SLIP 只是一个分组分帧协议，仅仅定义了一系列在串行线路上构造 IP 分组的字符。它没有提供地址、分组类型标识、错误检查/修正或者压缩机制。因为这个协议所作的工作这么少，通常很容易实现。

大约在 1984 年，Rick Adam 为 4.2Berkeley Unix 和 Sun Microsystem 工作站实现了 SLIP

并公之于众，并作为一种使用串行线路连接 TCP/IP 主机和路由器的简单可靠的方法很快流行起来。

SLIP 通常专门用于串行连接，有时候也用于拨号网络，使用的线路速率一般介于 1200bps 和 19.2Kbps 之间。SLIP 允许主机和路由器混合连接（主机-主机、主机-路由器、路由器-路由器都是 SLIP 网络通用的配置），因而非常有用。

可用性（AVAILABILITY）

SLIP 可用于大多数基于 Berkeley UNIX 的系统，并且被包括进了 Berkeley 的 4.3BSD 标准版。SLIP 可用于 Ultrix、Sun UNIX 和大多数派生自 Berkeley 的 UNIX 系统。一些终端集线器和 IBM PC 的实现也支持该协议。

Berkeley UNIX 的 SLIP 可以使用匿名 FTP 从 uunet.uu.net 上的 pub/sl.shar.Z 中获得。确保传输的是二进制文件，并使用 UNIX 解压程序打开它，然后把解开的文件作为 UNIX/bin/sh（如/bin/sh sl.shar）的 SHELL 命令使用

协议（PROTOCOL）

SLIP 定义了两个特殊字符：END 和 ESC。END 是八进制的 300（十进制 192），ESC 不同与 ASCII 的 ESCAPE 字符，是八进制的 333（十进制 219），本文中的 ESC 指的是 SLIP ESC 字符。发送分组时，SLIP 主机只是简单地发送分组数据。如果数据中有一个字节与 END 字符的编码相同，就连续传输两个字节 ESC 和八进制的 334（十进制 220）代替它。如果与 ESC 字符相同，就连续传输两个字节 ESC 和八进制的 335（十进制 221）代替它。分组的最后一个字节发出后，再传送一个 END 字符。

Phil Karn 建议稍微修改一下这个算法，分组的开始以及结束都使用 END 字符，这样可以刷掉线路噪声造成的不正确的字节。一般情况下接收方将只看到两个紧挨着的 END 字符并生成一个坏的 IP 分组。如果 SLIP 实现没有丢弃长度为 0 的 IP 分组，IP 实现就应该丢弃。如果存在线路噪声，接收到的由线路噪声造成的数据将被丢弃，而不会影响后续的分组。

因为没有“标准的”SLIP 规范，也就没有 SLIP 分组最大长度的实际定义。可能最好是接受 Berkeley UNIX SLIP 驱动程序使用的最大分组长度：1006 字节，其中包括 IP 头和传输协议头，但不含分帧字符。这样任何新的 SLIP 实现都应能够接收 1006 字节的数据报，在一个数据报内发送的字节数不应超过 1006。

缺陷（DEFICIENCIES）

有几种特性使许多用户希望 SLIP 提供而没有提供的。公平的讲，SLIP 只是一个很久以前设计的非常简单的协议，而在当时这些问题还并不真正重要。下面是对现有 SLIP 协议一般认识到的缺陷：

地址：

SLIP 连接的两台计算机都必须知道对方的 IP 地址才能传输。另外，在主机使用

SLIP 拨号连接一个路由器时，地址设置可能随时变化，路由器可能需要通知拨号主机 IP 地址的变更。SLIP 目前没有为主机提供通过 SLIP 连接交换地址信息的机制。

类型标识:

SLIP 没有类型字段。因此在一个 SLIP 连接上只能运行一个协议，即使在两台运行 TCP/IP 和 DECnet 的 DEC 计算机的配置中，如果使用 SLIP，也不可能让 TCP/IP 和 DECnet 同时使用一条连接两者的串行线路。因为 SLIP 是“串行线路 IP”，如果串行线路连接两台多协议计算机，这些计算机可以在这条线路上使用多个协议。

错误检测/修正:

嘈杂的电话线路可能破坏传输中的分组。因为线路速率可能很低（或许是 2400 波特），重新传输分组的代价很高。错误检测在 SLIP 层并非绝对需要，因为 IP 应用程序可以发现损坏的分组（IP 头部与 TCP 和 UDP 的校验和就可以满足），但是一些通用程序如 NFS 通常忽略校验和而依赖网络媒介检测损坏的分组。因为重新传输被线路噪声破坏的分组需要很长时间，如果自身能够提供某种简单的纠错机制就可以改善 SLIP 的效率。

压缩:

拨号线路非常慢（通常是 2400bps），分组压缩可以大幅提高分组的吞吐量。通常单纯的 TCP 连接分组流在 IP 和 TCP 头部有几个很少变动的字段，因而可以使用一种简单的压缩算法只发送头部变化的部分而不是整个头部。

围绕着 SLIP 后继者的设计与实现，几个不同的团体已经做了一些工作，可能会部分或者全部解决这些问题。

SLIP 驱动程序（SLIP DRIVERS）

下面的 C 语言函数发送并接收 SLIP 分组。它们依赖于 send_char() 和 recv_char()，这两个函数在串行线路上发送和接收单个字符。

```
/* SLIP 特殊字符编码
*/
#define END          0300    /* 分组结束标记 */
#define ESC          0333    /* 填充字节标记*/
#define ESC_END     0334    /* ESC ESC_END 表示数据字节 END */
#define ESC_ESC     0335    /* ESC ESC_ESC 表示数据字节 ESC */

/* SEND_PACKET: 发送长“len”的分组，起始位置为“p” */
void send_packet(p, len)
    char *p;
    int len; {

    /* 发送一个初始 END 字符，清除由于线路噪声可能堆积在接收方的任何数据
    */
    send_char(END);

    /* 为分组中的每个字符发送适当的字符序列
```

```
*/
while(len--) {
    switch(*p) {
        /* 如果与 END 字符相同，我们就发送
        * 两个特殊字符码避免接受方认为
        * 我们发出了 END 结束标记
        */
        case END:
            send_char(ESC);
            send_char(ESC_END);
            break;

        /* 如果与 ESC 字符编码相同，
        * 我们就发送两个特殊字符码
        * 避免接受方以为我们发送了 ESC
        */
        case ESC:
            send_char(ESC);
            send_char(ESC_ESC);
            break;

        /* 否则，我们就发送字符本身
        */
        default:
            send_char(*p);
    }

    p++;
}

/* 告诉接收方我们已经完成分组的发送
*/
send_char(END);
}

/* RECV_PACKET: 接收分组并放入地址为“p”的缓冲区，
* 如果收到的字节数大于 len，分组将被截断
* 返回保存在缓冲区的字节数
*/
int recv_packet(p, len)
    char *p;
    int len; {
    char c;
    int received = 0;
```

```
/* 使用循环读取字节直到接受完整个分组
 * 如果用完缓冲区就不再复制
 */
while(1) {
    /* 取一个字符进行处理
     */
    c = recv_char();

    /* 如果需要则处理填充字符
     */
    switch(c) {

        /* 如果是 END 字符就表示分组完成
         */
        case END:
            /* 一点小小的改进：如果分组没有数据则忽略掉。
             * 这意味着避免双 END 字符构成的空分组扰乱 IP，
             * 这种空分组用于检测线路噪声。
             */
            if(received)
                return received;
            else
                break;

        /* 如果收到 ESC 字符，则等待
         * 下一个字符来决定把什么字符存入分组
         */
        case ESC:
            c = recv_char();

            /* 如果“c”不是这两个字符中的一个，
             * 就违反了协议。最好的办法似乎是
             * 单独保留这个字符并填入分组
             */
            switch(c) {
                case ESC_END:
                    c = END;
                    break;
                case ESC_ESC:
                    c = ESC;
                    break;
            }
    }
}
```

```
/* 现在到了缺省处理情况，就让它保存字符
*/
default:
    if(received < len)
        p[received++] = c;
    }
}
```