Linux Traffic Control源码简述

张智彬

1. 引言

Linux的Traffic Control（以下简称TC）是linux内核中内置的流量控制模块，通过该模块可以进行网络端口的收发流量控制。TC以树状结构管理多种流量控制规则，并提供了FIFO、TBF等调度方案。



图 1 TC命令实例

本文将着重分析TC的源码及其执行流程，而不涉及具体的细节。TC程序分为用户态和内核态两个部分，两个状态下的程序通过netlink进行数据交换。用户态代码主要包括三个部分：解析可选参数、创建netlink socket、解析TC命令并打包发送。执行完这三个主要步骤后，由内核执行TC命令。

1. 用户态代码

2.1 解析可选参数

可选参数控制了输入方式（命令行或bash文件）、换行符（回车或反斜杠）、上色、命名空间、是否显示执行细节等一系列设定。正如其名，一条TC命令可以不包含可选参数，在这种情况下按照默认设定执行命令。

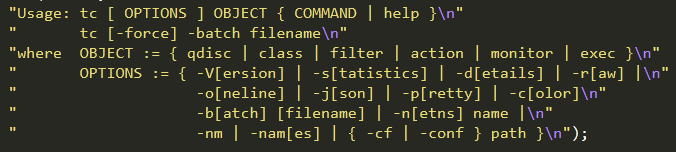


图 2 可选参数的输入帮助

这部分的代码逻辑较为简单清晰，全都位于tc.c中，采用了多个判断语句，根据字符串比较的结果修改对应的flag值，以备后续的代码进行判断。

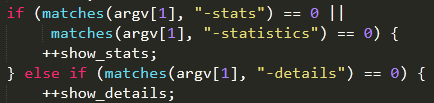


图 3 解析可选参数部分代码

2.2 创建Netlink Socket

在修改完可选参数的flag值后，用户态程序将创建一个netlink socket以供后续与内核的通信。在tc.c中，这一部分代码只是简单调用了rtnl\_open函数。

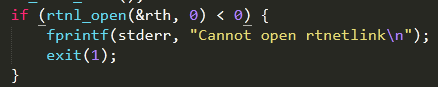


图 4 调用rtnl\_open函数

rtnl\_open函数的函数体在\iproute2\lib\libnetlink.c文件中，这个函数调用了同文件中的另一个函数rtnl\_open\_byproto，并添加了一个参数：NETLINK\_ROUTE，这个参数表示socket的协议是路由协议。rth指针指向的是一个struct rtnl\_handle类型的结构体，该结构体定义在\iproute2\lib\libnetlink.h中。rtnl\_open\_byproto函数的目的是将这个指针指向的结构体实例化。

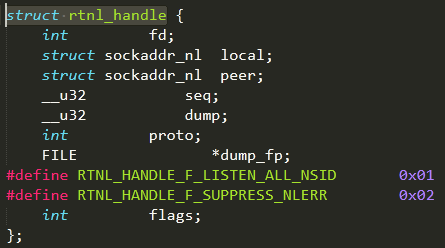


图 5 rtnl\_handle结构体

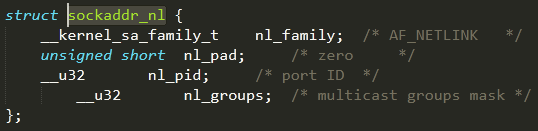


图 6 sockaddr\_nl结构体

结构体中，fd是文件描述符，指向新创建的socket的位置；protocol是协议名，这里使用的是NETLINK\_ROUTE；local的地址如图7所示设置。



图 7 设置local地址

2.3 解析TC命令并打包发送

这部分代码首先调用了tc.c文件中的do\_cmd函数，对命令的类型进行解析。

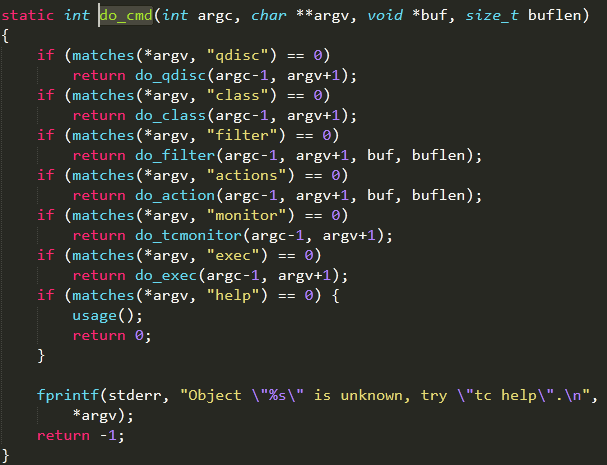


图 8 do\_cmd函数

根据不同的命令选择对应的函数进行调用，此处选用qdisc作为例子继续深入。do\_qdisc函数体位于tc\_qdisc.c中，根据下一个参数进行分类包装，调用tc\_qdisc\_modify函数进行下一步处理。此处的RTM\_NEWQDISC是命令类型，会被加入socket中，在之后内核部分的代码中根据它进行判断。

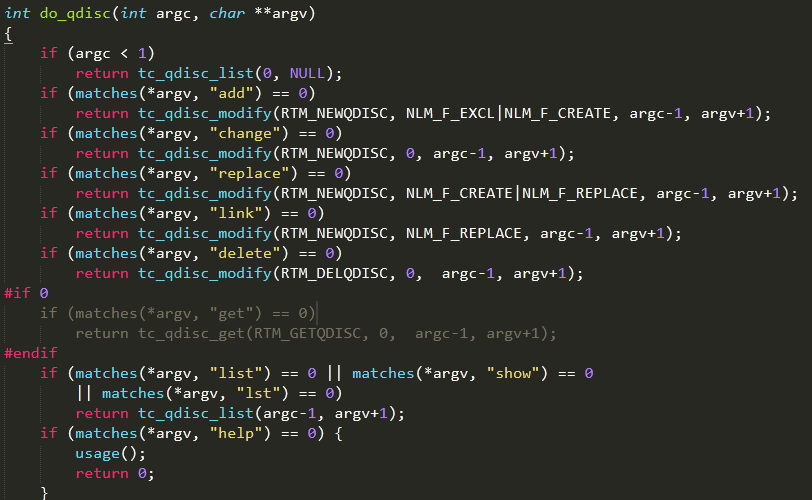


图 9 do\_qdisc函数

tc\_qdisc\_modify函数中会继续解析命令的后续参数，并填充到一个req结构体中，这个结构体中保存了需要通过socket进行传输的信息。当所有参数解析完毕后，这个函数会调用rtnl\_talk函数，在经过一系列的包装后，最终调用sendmsg函数，将socket发送给内核。

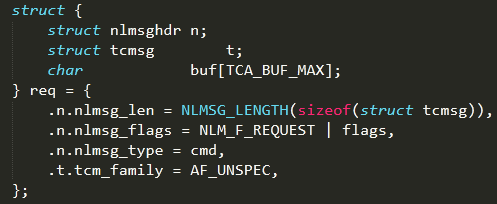


图 10 req结构体

1. 内核态代码

TC的内核态代码主要位于\net\sched中，包括了各种队列的具体实现、socket的接收和处理等代码。同上，仍以qdisc作为例子进行分析。

netlink在内核中通过rtnetlink\_rcv\_msg函数进行消息的接收，这个函数位于\net\core\rtnetlink.c中。它通过解析消息的family和type来确定链接到哪个函数来处理消息，如图11。

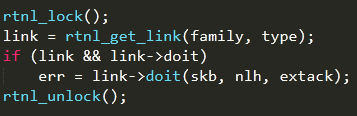


图 11 解析消息并调用对应处理函数

netlink的机制要求内核初始化时注册好所有处理函数，如下图所示。处理qdisc命令的函数位于\net\sched\sch\_api.c中。

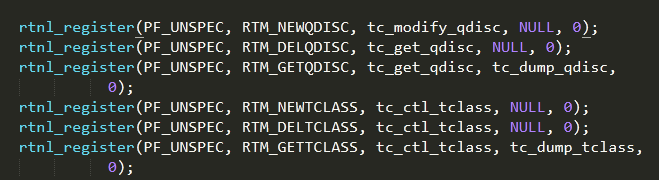


图 12 注册函数