用于矿井运输的控制协议①

李明华[1,2], 李鸿彬[1], 于 波[1], 贾军营[1]

[1](中国科学院 沈阳计算技术研究所, 辽宁 沈阳 110168)
[2](中国科学技术大学 计算机科学与技术学院, 安徽 合肥 230027)

摘 要: 在介绍矿用运输控制系统组成的基础上, 设计并实现了用于梭车控制台和手机终端之间的矿用运输控制协议. 阐述了用于矿井运输的控制协议的数据包格式, 通信方法, 实现过程. 最后给出实验结果, 实验结果表明矿用运输控制协议能较好的满足矿用运输控制需求.

关键词: 矿井运输; 控制协议; 梭车控制台; 手机终端; 打点; 急停

Control Protocol for Mine Transport

LI Ming-Hua[1,2], LI Hong-Bin[1], YU Bo[1], JIA Jun-Ying[1]

[1](Shenyang Institute of Computing Technology, Chinese Academy of Science, Shenyang 110168, China)
[2](University of Science and Technology of China, Computer Science and Technology, Hefei 230027, China)

Abstract: Based on the introduction of mining transport control system, mine transport control protocol used between shuttle car console and mobile terminal is designed and implemented. The article describes the mine transport control protocol's packet format and communication methods, and provides implementation process. At last, this paper presents test results. Test results indicate that mining transport control protocol can meet the requirements of mining transport control system well.

Key words: mine transport; control protocol; shuttle car console; mobile terminal; tap; emergency

运输系统是矿井生产的重要环节,为保证运输正常运行, 需要合理的管理和控制[1]. 随着网络通信技术和矿井运输设备的发展, 信息技术逐步渗透到矿井生产之中. 人们开始使用信息技术控制矿井中矿车的运动. 目前控制应用仍无标准应用层协议[2], 由于矿井运输系统的特殊性和复杂性, 现有的应用层控制协议很难满足矿井运输控制系统的要求. 根据矿井运输控制系统自身的特点, 本论文设计了用于矿井运输的打点、急停协议, 并对协议进行了实现. 结果表明, 本协议能较好的完成远程设备控制矿车的功能.

1 矿用运输控制系统概述

1.1 矿用运输控制系统的组成

矿用运输控制系统主要由梭车、梭车控制模块、梭车控制台、手机终端四部分组成.

梭车控制模块接收梭车控制台发送的控制命令, 实现对梭车的实际控制, 并向梭车控制台提供梭车的实时信息.

梭车控制台是系统中一个核心软件, 操作员通过控制台能实时看到梭车的运行情况. 控制台可以主动发送打点、急停等控制命令给梭车控制模块, 也能接收手机终端发送来的控制命令, 并以文字的形式显示发送控制命令的手机终端状态.

手机终端: 手机终端可以发送打点或急停控制命令到其余手机终端和梭车控制台, 并通过梭车控制台实现对矿车的控制.

1.2 矿用运输控制协议相关概念

矿用运输控制协议是手机终端和控制台之间的通信协议. 通信协议包括对报文格式, 通信方式等问题的统一规定, 需要在双方的通信中共同遵守[3]. 本协

① 收稿时间:2012-10-17;收到修改稿时间:2012-11-30

议将矿井运输控制协议分为打点协议和急停协议两种类型. 打点协议用于发送打点控制命令, 控制矿车正向运行、反向运行等. 急停协议用于发送急停控制命令, 实现特殊情况下紧急停车功能. 控制台和手机终端都设有打点按钮和急停按钮. 按下并抬起打点按钮产生一次打点, 多次打点构成一次打点过程. 急停按钮采用自锁型开关, 按压一次处于急停状态, 再按一次处于急停解除状态.

2 协议的设计

2.1 底层协议的选择

TCP[5]是基于连接的协议, 可靠性高, 但是数据传输速率慢. UDP 协议传输可靠性差, 但是 UDP 协议简单, 信道资源占用率低[4,6], 数据发送前不用建立连接, 开销和延迟小, 这一点对控制系统来说非常重要. 因此传输层协议选择 UDP 协议. RTP[7]具有可扩展性. 通常为一个具体的应用提供服务, RTP 只提供框架, 基于以上优点, 本协议根据矿用运输控制系统的具体要求在 RTP 协议基础上进行扩展.

2.2 协议数据包设计

打点数据包类型包括打点消息报文(TAP), 打点确认消息报文(TAP_ACK), 打点确认消息响应报文(TAP_ACK_Resp), 竞争错误报文(COMP_Err), 打点确认重发请求报文(TAP_ACK_Resend), 静音报文(TAP_Silence)6 种类型.

急停数据包类型包括急停报文(Emergency_Stop), 急停响应报文(Emergency_Stop_Resp)两种类型.

协议报文如下所示:

8字节账号				
8字节打点ID				
4字节序号				
1字节包类型	E	R	6位打点总数	2字节打点1持续时间
...				2字节打点（总数-1) 持续时间

图 1 打点报文, 打点确认报文, 静音报文

8字节账号	
8字节打点ID	
4字节序号	
1字节包类型	1字节保留位

图 2 打点确认响应报文, 打点确认请求报文, 竞争错误报文

8字节账号				
8字节急停ID				
4字节序号				
1字节包类型	E	R	6位打点总数	2字节打点1持续时间
...				2字节打点（总数-1) 持续时间

图 3 急停报文

8字节账号			
8字节急停ID			
4字节序号			
1字节包类型	E	7位保留位	2字节被控实时数据

图 4 急停响应报文

其中图中的 R 表示保留位, 默认置 0. 打点持续时间表示每个打点控制命令的持续时间, 以采样数为单位. 图 1 中的 E 表示一次打点是否结束, 0 表示打点结束, 1 表示打点没有结束. 图 3 中的 E 表示结束位, 默认置 0. 图 4 中的 E 表示急停控制命令是否已经执行, 1 表示已经执行, 0 表示没有执行.

2.3 协议通信

2.3.1 正常情况下打点控制命令交互

控制台和手机终端的打点交互过程正常情况下分为发送打点控制命令、打点确认、打点确认响应三个阶段. 如图 5 所示, 具体数据交互过程如下:

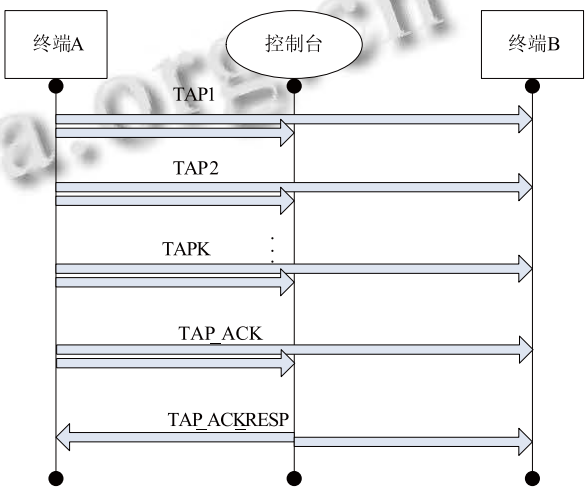

图 5 正常情况下打点数据包交互

① 终端 A 向控制台和终端 B 发送打点消息, 不同打点消息个数代表不同的命令.

② 控制台和终端 B 接收到终端 A 发送的打点消息不用进行回应.

③ 终端 A 发送打点控制命令后超过 1000ms，如果没有发送下一个打点控制命令，则发送打点确认报文并启动超时机制等待控制台的打点确认响应.

④ 终端 B 接收到打点确认报文不用进行回应，控制台收到打点确认报文，向所有终端发送打点确认响应报文.

⑤ 终端 A 发送打点确认报文后 2000ms，打点控制命令交互结束.

2.3.2 存在竞争错误时打点控制命令交互

控制台与手机终端的打点交互过程在出现竞争情况下，如下图 6 所示，包括以下几个步骤:

① 终端 A 向终端 B 和控制台发送打点消息，不同打点消息个数代表不同的命令.

② 终端 B 还没有收到终端 A 发送的打点消息，此时终端 B 也发送打点消息.

③ 终端 A 收到终端 B 发送的打点消息，知道发生了冲突，向所有终端发送竞争错误消息. 发送竞争错误 2000 毫秒后打点过程结束.

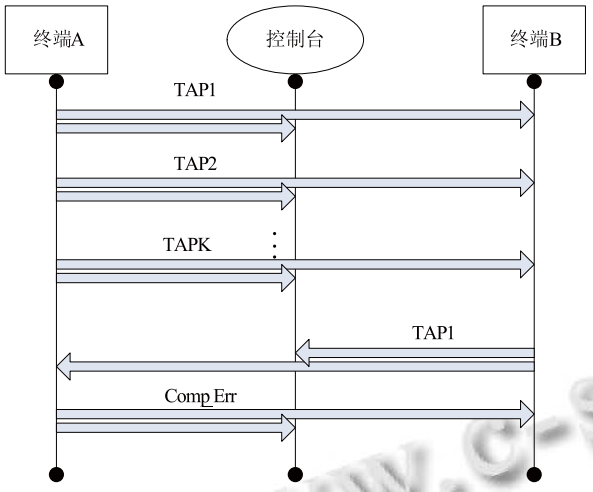

图 6　竞争情况下打点数据包交互

2.3.3 在出现丢包时打点控制命令交互

控制台和手机终端在丢包情况下打点交互过程如下图 7 所示，具体步骤如下:

① 终端 A 向终端 B 和控制台发送打点消息，不同打点数目代表不同的命令.

② 终端 B 收到第 K-1 个打点消息后，1400ms 内没有收到打点消息也没有收到打点确认消息. 终端 B 向终端 A 每隔 300ms 发送打点重发请求消息，直到收到确认，或者是请求超过 5 次.

③ 终端 A 收到终端 B 的打点重发请求消息，向终端 B 重新发送第 K 次打点消息.

④ 终端 A 打出最后一点后，超过 1000ms 没有后续点发出，则向接收端发送打点确认消息，并启动超时机制.

⑤ 终端 B 收到打点消息后，超过 1400ms 没有收打点消息，也没有收到打点确认消息，则向终端 A 每隔 300ms 发送打点确认重发请求消息，直到收到确认，或者是请求超过 5 次.

⑥ 终端 A 收到终端 B 的打点确认重发请求报文，向终端 B 重新发送打点确认报文.

⑦ 控制台收到打点确认报文后，向所有的终端发送打点确认响应报文.

⑧ 终端 A 发送打点确认报文后超过 500ms 没有收到打点确认响应报文，则每隔 300ms 重新发送打点确认报文，直到收到打点确认响应报文，或重发打点确认报文超过 3 次.

⑨ 控制台接收到打点确认报文，重新向所有终端发送打点确认响应报文.

⑩ 终端 A 发送打点确认报文后超过 2000ms，打点控制命令交互结束.

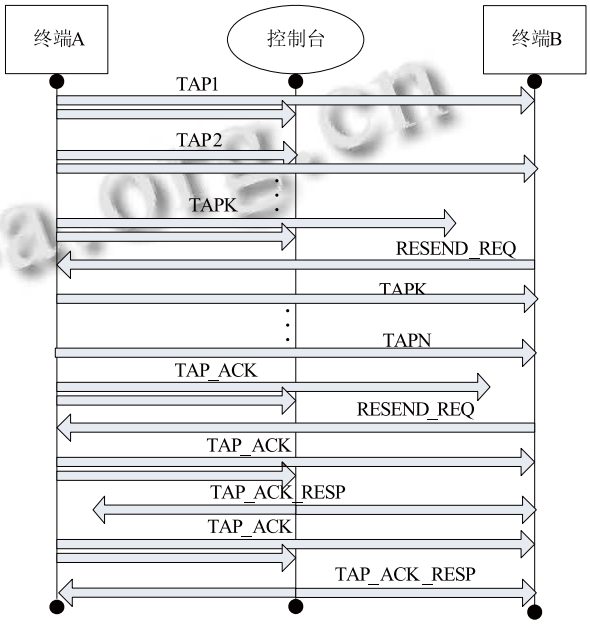

图 7　丢包情况下打点数据包交互

2.2.4 急停控制命令数据交互

急停控制命令比打点控制命令具有更高的实时性，当急停命令被执行时，所有的打点命令操作被终止.

急停控制命令数据交互主要包括发送急停命令和急停响应两个阶段, 数据交互过程如下:

① 终端 A 向所有终端发送急停报文, 如果没有取消发送, 则以 200ms 为间隔一直发送急停报文.

② 普通终端收到急停报文后不用进行回应, 控制台收到急停报文后, 如果没有取消急停, 同样会以 200ms 为间隔向所有终端发送急停响应报文.

急停控制命令不会发生冲突, 在同一时间可以有多个终端发送急停控制命令. 当有多个终端发送急停控制命令时, 控制台仍以同样的时间间隔发送急停响应报文.

3 协议的实现

该协议在中科院沈阳计算所网络与通信实验室原有协议栈的基础上采用 c++ 语言开发.

3.1 协议全局变量

协议中除打点确认重发请求和和打点控制命令重发采用单播方式发送, 其余的消息都采用组播发送.

多播地址: m_MultiCastIP, 239.0.0.3

多播端口: m_MultiCastPort, 2010

生存时间: m_TTL,1

多播回环: m_LoopBack, false

单播端口: m_SingleCastPort, 2009

3.2 协议的主要数据结构

① CTapPacket: 包括账号, 代理 Id, 序列号, 包类型, 是否结束等成员变量. 并提供 SetAccount(), SetDlgID(), SetSeqNo(), SetEndFlag(), SetDuration(),, Parse(), Encode()等成员函数, 这些函数分别用来设置账号, 设置代理 ID, 设置序列号, 设置结束标志, 设置持续时间, 解析数据包, 编码数据包等.

② TapEvent: 枚举所有可能添加的事件. 包括空闲超时, 打点键按下, 打点键抬起, 接收打点确认请求, 发送打点超时, 等待打点确认响应超时, 等待打点确认响应超时 K 次, 接受打点确认响应, 等待结束超时, 接受打点竞争, 等待打点确认超时, 等待打点去确认超时 K 次, 接收打点超时, 接收打点确认, 接收急停, 接受急停响应, 发送急停超时, 发送急停响应超时, 等待急停超时 19 种事件.

③ TapState: 枚举终端所有可能的状态, 包括空闲、打点、完成一点、发送打点确认、接收打点、接收端等待结束、发送端等待结束, 接收急停 8 种状态.

④ CTapEvent: 包括 TapEvent 类型成员变量和 LPEventParam 类型成员变量, 其中 LPEventParam 是由 CTapPacket 与发包地址组成的结构体. EventName() 函数用来获得不同事件的名称.

⑤ TapEventQueue: 类型为 CtapEvent 指针的队列, 定义如下: Typedef queue<CTapEvent*> TapEventQueue;

⑥ CTapState: 维护一个事件队列, 开启一个线程, 当接收到新的数据包时, 根据终端的当前状态, 向队列中添加相应的事件. 并在线程中处理事件队列, 直到线程终止. 其中函数 StartThread()用于开启线程添加相应事件, ProcEventQueue()用于获取事件队列的元素, 并根据终端当前状态和事件类型进行相应处理. Transmit()用于实现终端状态的转变. StartSendTap()函数负责设置数据包的账号, 序列号等参数并通过多播方式完成数据发送.

⑦ TapTransporter: 提供的成员函数包括 StartTapStream(), StartTap(), StopTap(), StartEmergStop(), ProcRecvBuffer()等, 其中 StartTapStream()函数用于创建套接字, 加入组播组, 并调用 CTapState 类中的 StartThread()函数. StartTap(), StopTap(), StartEmergStop(), 分别用于向事件队列中添加打点键按下事件, 打点键释放事件, 接收急停事件. ProcRecvBuffer()函数用于当终端接收到数据时, 解析得到的数据包, 并根据收到的数据包的类型进行相应处理.

3.3 协议的状态转移

打点空闲状态为状态转移的初始状态, 手机和梭车控制台的初始状态都为打点空闲状态, 打点急停结束后, 他们的终止状态也都是打点空闲状态.

每种状态除了可以实现一种状态到另一种状态的转换, 还可以实现自身到自身状态的转换.

空闲状态下, 当打点键按下时, 终端发送打点, 播放打点音, 转变为打点状态; 当终端接收到打点包时, 播放打点音, 空闲状态转变为接收打点状态; 当接收到急停包时, 转换为接收急停状态;

打点状态下收到确认重发请求数据包或自动重发定时器超时, 重新发送打点数据包. 打点状态下按键抬起, 停止播放打点音, 停止发送打点消息. 进入完成一点状态. 打点状态下收到其他账号打点消息, 发送竞争错误, 打点终止, 进入空闲状态.

完成一点状态下接收打点确认重发请求消息, 重发最后一次打点消息. 收到其他账号打点消息, 发送竞争错误, 打点终止. 再次按下打点按钮, 发送打点播放打点音, 重新回到打点状态. 完成一点超时 1000 ms, 发送打点确认消息, 进入打点确认状态.

打点确认状态下收到打点确认重发请求, 重发打

点确认消息，收到打点确认响应，进入发送端等待结束状态. 超过 500ms 没有收到打点确认响应消息，重发打点确认消息，超过三次，表示发生错误，进入发送端等待结束状态.

发送端等待结束状态，收到打点确认重发请求，重发打点确认消息，发送端等待结束状态下延时2000ms,进入空闲状态.

接收打点状态下收到其他账号打点消息或收到竞争错误消息，打点终止，进入空闲状态；点击打点按钮，播放错误提示音；接收打点消息或打点重发消息，播放打点音； 1500ms 内没有收到打点消息和打点确认消息，则发送打点确认重发请求；接收打点确认消息或发送三次打点确认请求失败三次，终端进入接收端等待结束状态.

接收端等待结束状态延时 2000ms，进入空闲状态.

4 协议的测试

本文联合矿用运输控制系统梭车控制台对所实现的控制协议进行了测试. 程序实现了测试终端，通过在测试终端上选择 Tap Client 按钮指定本机作为手机终端运行矿用运输控制协议，测试了测试终端和梭车控制台的通信. 结果如图 8、9、10 所示:

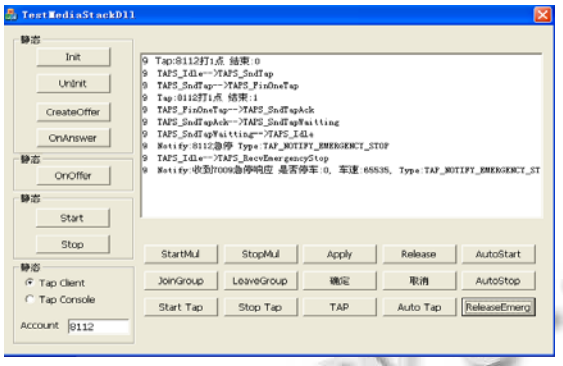

图 8　测试终端的测试结果

图 9　梭车控制台急停测试结果

图 10　梭车控制台打点测试结果

从图中可以看出测试终端发送的打点急停信息，控制台可以成功收到. 该协议实现了梭车控制台与通信终端的通信，满足了矿用运输控制系统的需求.

5 结语

本文在阐述矿用运输控制系统的基础上，提出了矿用运输控制协议的设计方法，并对该方法进行了实现. 通过在实际环境中测试，证明该协议很好的完成了控制台和测试终端通信的功能. 目前上述协议已经成功应用到矿用运输设备网络管理控制系统项目中，在系统项目中取得了良好效果.

参考文献

1 章壮新,吴桂义.矿井大巷矿车运输系统模拟.矿业安全与环保,2001,28(2):21–23.

2 彭可,陈志盛,陈岚,陈际达.一种面向控制应用的总线以太网改进协议.小型微型计算机系统,2005,26(5):771–774.

3 李莹,贾彬.一种基于状态机的串口通信协议的设计与实现.电子设计工程,2012,20(7):100–104.

4 杨云,张洁.Jabber 无线环境可靠传输协议的设计与实现.微电子学与计算机,2010,27(6):134–136,141.

5 Information Sciences Institute University of Southern California.Transmission Control Protocol. IETT RFC 793,Sep1981.

6 Postel J.User Datagram Protocol.IETF RFC768,Aug.1980.

7 Frederick R.A Transport Protocol for Real-Time Applications.IETF RFC3550,July 2007.