密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：无线多模网关中视频传输拥塞控制机制的设计与实现

学　　号：2012110614

姓　　名：张新

专　　业：计算机科学与技术

导　　师：李文生

学　　院：计算机学院

2015 年　1 月　5　日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

无线多模网关中视频传输拥塞控制机制的设计与实现

摘 要

近年来，随着通信网络的迅速发展，具有多种通信手段、设施布置灵活的无线多模网关成为环境监测、应急通信等领域研究的重点。同时人们对视频服务的需求日益增大，而基于无线多模网关的网络与传统网络差别较大，因此在无线多模网关中对视频传输的拥塞控制是当前急需解决的问题。

本文通过对拥塞控制进行研究与分析，针对无线多模网关设计和实现了一种视频传输的拥塞控制机制。该机制通过对视频发送速率的控制和视频发送缓存的控制两方面来实现拥塞控制功能。本文改进了基于 TFRC 的速率模型，以此来实现对视频发送速率的控制，同时保证了 UDP 连接和 TCP 连接公平共享网络带宽，并且，通过对 UDP 缓冲区和视频传输跳帧数的调整来实现对视频发送缓存的控制。最后本文在基于 Linux 系统的 ARM 平台上实现了具有该拥塞控制机制的视频传输系统的基本原型，验证了该拥塞控制机制是可行的，对控制效果的评估结果表明该机制是有效的。

关键词：无线多模网关，视频传输，速率控制，缓存控制

DESIGN AND IMPLEMENTATION OF CONGESTION CONTROL STRATEGY IN VIDEO TRANSMISSIONFOR WIRELESS MULTI-MODE NETWORK

ABSTRACT

In recent years, with the rapid development of communication network, wireless multimode gateway which has a variety of means of communication and flexible facilities layout become a focus of research in the fields such as environmental monitoring and emergency communication. At the same time, people's demand of video services is increasing, while the networks based on multi-mode wireless gateways and traditional networks vary greatly. As a result, it is urgent to solve the congestion control problem in video transmission of the wireless multi-mode gateway.

This thesis designs and implements a congestion control strategy for the multi-mode wireless gateway that suits for video transmission based on the research and analysis of congestion control. We implement the congestion control function by two aspects, video transmission rate control and video transmission buffer control. We implement the video transmission rate control by using the improved TFRC model, while ensuring that the UDP connection and TCP connection are abled to fairly sharing the network bandwidth, and the video transmission buffer control that by adjusting the UDP buffer and video transmission skipped frames. Finally we establish the prototype video transmission system that based on the congestion control strategy in the Linux system on ARM platform to test the feasibility of the congestion control strategy and the effect on evaluation results shows the strategy is effective.

KEY WORDS： wireless multi-mode gateway, video transmission, rate control, buffer control.

目录

第一章 绪论

1.1 课题研究背景

随着互联网技术的迅速发展，网络的应用从传统的文字网页、电子邮件过渡到了以视频、语音为主体的多媒体应用[1]。而近年来以平板电脑，智能手机为代表的智能终端得到了迅速的推广，使得新一批的多媒体业务犹如雨后春笋般大量催生，例如视频会议、网络教学、远程医疗等，而其中视频流又占据了这些互联网多媒体业务流的绝大部分。这些视频多媒体业务对网络带宽、网络时延以及网络抖动有着非常严格的要求，因此对于视频数据网络传输的过程必须采取相应的拥塞控制策略。在有线网络中，数据传输误码率低，网络传输带宽有限制，而且拥塞控制机制比较成熟，因此以视频为主的多媒体在有线网络中已经得到了非常广泛的应用。

同时近年来，随着无线通信技术的高速发展，出现了许多不同类型的无线网络，例如卫星通信网络（如北斗、铱星）、3G 和 4G 蜂窝移动通信网络（如 GSM、WCDMA、CDMA2000 等）、无线个人局域网（WPAN，如 802．15）、无线局域网（WLAN，如 802．11）、无线城域网（WMAN， 如 802．16）、无线广域网（WWAN，如 802.20）、移动 Ad hoc 网络（MANET）、无线传感网（WSN）和无线 Mesh 网（WMN）等[2]。与此同时，由于近年来国内外自然灾害和突发事件的不断发生，应急通信逐渐成为了研究的焦点[3]。在应急通信领域，单纯以一种通信技术手段或者一种通信网络的通信方式已经不能为各类用户在情况复杂多样的应急环境中提供快速、灵活、可靠的通信服务[3]。因此，拥有多种无线网络、卫星网络等为通信手段的无线多模网关，成为各种应急突发情况下有效的通信保障。而在应急通信中，以视频为主的多媒体流依然是网络传输的主体，因此为了保证无线多模网关中视频的服务质量，有必要对视频传输进行拥塞控制。这不仅是无线多模网关中进行视频传输的迫切需求，也是今后无线多模网关发展的必然要求。

此外，当前互联网上的数据传输基本通过 TCP 和 UDP 这两种协议来实现。其中 TCP 是面向连接的传输协议，有超时重传、拥塞控制等机制来保证可靠性，但是却无法保证实时性，故对于实时性要求较高的视频数据，基本选择 UDP 协议来进行传输。但是 UDP 没有拥塞控制机制，因此当 TCP 连接和 UDP 连接同时占用网络带宽的时候，UDP 连接将占据大量的带宽，而 TCP 则由于具有拥塞控制机制导致其占据的带宽一步一步的缩小，最后可能导致拥塞崩溃。因此，保

证 TCP 数据正常传输是研究 UDP 拥塞控制机制的一个重要内容。对于传统的有线网络和传统的单一无线网络，经过近 20 年的发展，已经有不少成熟的 UDP 拥塞控制机制。但是对于无线多模网关来说，网关节点可能具有多种网络通信手段，因此能为用户带来灵活的通信方式。但是与此同时，网关节点具有多种通信方式的特点也导致了数据传输网络结构不稳定，因此传统的拥塞控制方法已经不再适用于该网络。无线多模网关作为将来重要的通信设备，而视频流又占据了网络多媒体流的大部分流量，可以确定是主要的多媒体流，因此研究一种专门针对在无线多模网关中的视频传输拥塞控制机制是当前的迫切需求，也是保证基于无线多模网关的网络整体效益的必然要求。

1.2 国内外研究现状

对 UDP 协议的拥塞控制机制的研究开始于二十世纪末，到如今已经有很多成熟的拥塞控制机制。这些拥塞控制机制按照网络负载的调整方式可分为两大类。第一类是以 TCP 的拥塞控制机制为参照，模拟 TCP 的 AIMD 机制，并在此基础上有一定的改进机制，典型代表如 GAIMD、BCCA。这类拥塞控制机制由于其调整策略，会导致数据传输波动较大，数据传输速率不稳定的缺陷，甚至可能产生突发的数据流。

第二类是国外的研究学者专门为多媒体数据的传输提出，是可以保证与 TCP 公平竞争网络带宽的拥塞控制机制。这类拥塞控制机制在传输多媒体数据时，在保证传输速率稳定平滑的基础上，同时保证了 UDP 数据流和 TCP 数据流能公平共享网络带宽。比较经典的算法包括 LDA+[4]、TEAR[5] 、RAP[6]和 TFRC[7]。LDA 协议采用 RTP 进行传输，能在减小发送速率抖动的基础上防止突发数据流，其缺陷是一旦发生丢包，就会降低数据的发送速率，因此会带来较大的抖动，影响数据传输的平稳性。TEAR 协议的策略是在接收端计算发送速率并反馈至发送端来实现拥塞控制，虽然该算法能很好地适应多媒体数据的传输，但是实现非常困难，而且大量的反馈数据会给接收端带来巨大负担。RAP 则采用适配算法来减少传输抖动，然而 RAP 对每个数据包都要进行反馈的规定大大的增加网络的额外数据量。TFRC 采用 TCP-Reno 协议中的吞吐量模型计算发送速率，复杂度低且易于实现，速率调整也较为平滑，但是由于 TFRC 不具有丢包区分机制，因此可能导致发送速率估计过低，浪费带宽的现象。

在国内，文献[8]提出多媒体传输都会有一个最低的传输速率，要求网络拥塞程度比较轻的时候以最低的传输速率发送数据，但是该策略对缓冲区、多媒体编码器等因素没有考虑充分。文献[9]提出基于显式速率的 UDP 拥塞控制机制，该机制通过发送端和路由器相互配合，使得 UDP 应用能够根据网络的反馈，以网

络的公平带宽为速率发送数据，但是估计实际连接到该网络的应用数目难度较高，不能保证公平性。文献[10]提出端系统和路由节点结合的方法，采用基于令牌桶的支持标记的自适应速率调节机制来对端主机进行参数配置，同时在路由器上采用对不同的流量进行相应处理的 RED 算法，但是实现比较困难。

1.3 研究意义及目标

根据无线多模网关的迅猛发展以及以视频为主多媒体服务的客观需求，本文希望设计并实现一种在无线多模网关中视频数据的拥塞控制机制。

无线多模网关由于具有诸多的优势已经成为业界研究关注的焦点，将来必将在应急通信领域发挥巨大的作用，而视频流作为网络数据流主体，随着将来视频会议、远程医疗的迅速发展，视频流在网络流量中的比例必将越来越大，必然也是无线多模网关中传输的主要数据流。本文针对无线多模网关中视频传输提出的拥塞控制机制不仅能保证视频数据在该网络中的传输质量，同时还能保证视频数据流与其他数据流公平共享网络带宽，保证了网络的整体效益。另一方面，本文将视频传输的拥塞控制思想应用到基于无线多模网关的网络中，这不仅是无线多模网关的发展必然要求，同时也为其他相关数据（如语音）在该网络中的拥塞控制提供了重要的理论参考。

本文的目标是设计并实现一种在无线多模网关中的视频传输拥塞控制机制。要求该机制能够根据网络延时、视频丢包率等网络参数来制定对视频数据发送速率和发送缓存的调整方法，以达到降低网络丢包率，减少网络拥塞的效果。针对以上功能，本文提出拥塞控制机制的设计与实现方案，对该机制的实现效果做出测试和评估，并针对可能测试中遇到的问题提出可行的解决方案。

1.4 主要工作及内容组织

本文的主要目标是研究并实现一种稳定、有效的拥塞控制机制，要求该机制能对无线多模网关中视频传输进行控制，以达到减少网络丢包率，降低网络拥塞程度的效果。该拥塞控制机制主要通过速率控制和缓存控制两方面来实现。

通过分析论文的目标，本文的具体任务有：

首先研究网络拥塞现象和其产生的原因，并对传统的拥塞控制进行研究与分析；研究基于无线多模网关下的网络结构特点和视频传输系统的相关知识；针对无线多模网关中视频传输设计并实现拥塞控制机制，同时对该机制搭建测试环境，对该机制进行性能评估和测试结果分析。

论文组织如下：

第一章为绪论，通过介绍本文的研究背景和国内外的研究现状，再结合对论

文研究意义及研究目标的分析，得出论文工作方向及论文组织结构。

第二章为相关技术介绍，先研究了拥塞产生的原因和基本思想，接着对有线网络和无线网络中的拥塞控制进行了细致分析，并研究了其中经典的拥塞控制算法。

第三章为无线多模网关中视频传输拥塞控制机制的设计，通过介绍基于无线多模网关的网络结构特点以及视频传输机制，结合关键的相关技术，给出了无线多模网关中视频传输拥塞控制机制的详细设计方案。

第四章为无线多模网关中视频传输拥塞控制机制的实现，结合上一章节的详细设计与相应的软硬件平台，给出了拥塞控制机制的具体实现方案。

第五章为对本文所述拥塞控制机制的测试和结果分析，先介绍了测试环境，接着在传统视频传输系统和具有本文所述的拥塞控制机制的视频传输系统进行了拥塞控制测试，通过对多组参数的测试结果对比，验证了本文所述机制的可行性与有效性。

第六章为结束语，对论文整体工作做出总结，结合测试结果，对本文所述的拥塞控制机制提出了改进建议。

第二章 相关技术介绍

2.1 拥塞控制的基本概念

随着互联网技术的迅猛发展，计算机网络的规模和数据传输的速率都得到了巨大的提升，造成了数据冲突、网络拥塞等问题的产生。为了避免造成网络资源的浪费，让网络负载适应网络带宽，网络拥塞控制技术成为当前网络领域的研究热点之一。

网络拥塞是指在数据传输过程中传送分组数目超出了存储转发节点的资源限度而造成的网络传输性能的降低[11]。美国劳伦斯伯克利实验室与加州大学伯克利分校于 1986 年 10 月最先观察到网络拥塞的现象[12]。网络拥塞现象的过程可分为三个部分，依次是拥塞避免阶段、拥塞恢复阶段和拥塞崩溃阶段。当网络负载比较少时，网络吞吐量随着网络负载的增加而增加，此时，网络拥塞程度很轻，整个网络处于拥塞避免阶段；而如果网络负载继续增加，网络吞吐量的增长速度会放缓，但是还是会增加，此时网络处于拥塞恢复阶段；如果网络负载进一步增加，则会导致网络丢包率大幅提升，响应时间延长，进而导致网络吞吐量急剧下降，此时整个网络处于拥塞崩溃阶段。

通过网络拥塞现象的分析可知，网络负载的需求超过了网络资源所能提供的网络带宽的最大值是网络拥塞发生的主要原因。最初的网络设计不是面向连接的，而是采用尽力服务的机制，不加区分的传输所有分组数据，缺乏必要的制约机制，是造成网络拥塞的根源。而具体来说，网络拥塞现象产生的主要原因有网络带宽容量不足、存储空间不足和网络节点处理速度与网络带宽不匹配等。

拥塞控制是指网络中的数据传输节点能根据网络的拥塞状态做出相应的措施来防止网络拥塞现象的发生。网络拥塞控制的目标是在检测到网络拥塞现象时，采取相应的控制机制对拥塞进行恢复，降低拥塞带来的数据包丢失、延时过大等网络性能下降的程度。根据网络拥塞产生的原因分析，单方面的去增加网络带宽或者存储空间，又或者是去增加整个网络节点的处理速度，都不能有效的减缓网络拥塞程度。而如果多个因素都考虑的话，所需要的开销费用十分昂贵。所以一般的拥塞控制机制都是根据网络检测因素，包括数据丢包率、网络往返延时和网络延时抖动等，去判断网络拥塞程度，然后有效的去控制网络数据包的发送速率和接收速率。

2.2 有线网络下的拥塞控制机制

最初的拥塞控制机制都是基于有线网络的，在其中 TCP 拥塞控制机制是最经典的拥塞控制算法，后续的拥塞控制算法基本都是基于TCP拥塞控制的思想。另一方面，尽管 UDP 协议本身并没有拥塞控制机制，但是为了保证网络的整体传输质量，有许多研究将拥塞控制的思想应用到了 UDP 传输中。

2.2.1 TCP 拥塞控制机制

拥塞控制中最典型的算法就是 TCP 协议中基于滑动窗口的拥塞控制机制[13]。TCP 拥塞控制的原理是通过控制数据发送端对传输网络发送的数据量，来避免网络负载超过其所能承载的极限而崩溃的后果。TCP 拥塞控制的整个过程就是不断地对网络有效带宽进行探测，然后进行有效的数据传输控制的过程。整个 TCP 拥塞控制的有四个算法，即“慢启动”、“拥塞避免”、“快速重传”和“快速恢复”。

(1)“慢启动”算法。“慢启动”(slow start)算法是 1988 年 Jacobson 针对 TCP 协议在网络拥塞控制方面的缺陷提出的。原始的 TCP 协议会在连接建立后，无限制地发送数据包，导致网络缓存急剧缩小甚至耗尽，进而发生网络拥塞。因此，“慢启动”的思想就是在建立 TCP 连接之后，不会立即发送大量数据包，而是从一个比较低的值（一般是 1 个最大报文段(MSS)）开始，根据网络情况逐步地增加每次发送的数据包。具体来说，发送端每次接收到一个数据报文段确认之后，即一个网络往返时延 RTT(Round Trip Time)之后，就会将原来数据发送速率翻倍。因此“慢启动”的数据发送速率是随着 RTT 成指数级增长。

(2)“拥塞避免”算法。“拥塞避免”(congestion avoidance)算法也是 Jacobson 在 1988 年提出的。由“慢启动”算法速率呈指数增长可知，数据包的传输速率会很快的增长，但是网络带宽有限，数据包传输速率不可能一直这样增长下去。因此，TCP 设置了一个“慢启动”的门限值(ssthreash)，当数据传输速率超过该门限值，“慢启动”阶段结束，进入“拥塞避免”阶段。此时，数据包的传输速率不再按指数级增加，而是以加法线性增加。直至数据传输速率增加到 TCP 认为网络发生了拥塞，此时 TCP 将“慢启动”门限值设为当前数据传输速率的一半，同时将数据传输速率设为 1，然后重复“慢启动”过程。这种算法延长了网络达到拥塞的时间，使数据传输速率能够保持在较大值传输较长的时间，而且最后会慢慢的调整至网络的最佳值。

(3)“快速重传”算法。“快速重传”(fast recovery)算法是专门针对数据包丢包或数据包乱序提出的改进算法。该算法的基本思想是当发送端连续接收到三个或者更多的来自接收端的重复 ACK，就认为网络中出现了拥塞现象，该 ACK 对应的数据包已经丢失，因此，“快速重传”的做法是不等待重传定时器超时而立

即重发该数据包。这种算法极大地减少了重传等待时间，从而降低网络节点由于丢包或乱序造成的网络拥塞的程度。

(4)“快速恢复”算法。“快速恢复”(fast retransmit)算法是 Jacobson 在 1990 年在 Tahoe 的基础上提出的改进算法，该版本称为 TCP Reno。主要目标是在网络拥塞程度较轻的情况下，避免由于“慢启动”造成的传输速率过渡减小的现象。相对于“拥塞避免”算法，“快速恢复”算法主要改进是在 TCP 认为网络发生拥塞时，先把“慢启动”的门限值设为当前数据传输速率的一半，但是不会重复“慢启动”的过程，而是将当前数据传输速率设置为“慢启动”的门限值，之后继续呈加法的线性增加。

TCP 拥塞控制机制是网络拥塞控制中最基础的算法，它采用基于探测的思想，不断地调整数据传输速率，最终达到适合网络数据传输的最佳状态。但是 TCP 拥塞控制算法有调整时间过长的缺点，之后也有许多算法研究对其进行了改进，典型的如 TCP-HACK、TCP-Real 和 JTCP 等，但是基本思想都与上文所述基本类似，本文由于篇幅限制，在这里不在复述。

2.2.2 UDP 拥塞控制机制

UDP 是一种面向无连接的传输层协议，它本身并没有拥塞控制机制，但是为了保障 UDP 网络传输的服务质量，在 UDP 拥塞控制方面还是有大量的研究。总体来说可将其分为基于丢包率的拥塞控制机制和基于网络往返时延的拥塞控制机制。

基于丢包率的拥塞控制机制是基于反馈的思想。主要流程是数据接收端会在每个时间段内向发送端反馈一次网络丢包率，发送端会根据接收到的丢包率情况判断网络拥塞的程度，然后来调整发送端数据的发送速率，以此完成拥塞控制。比较典型的就是实时流媒体传输协议 RTP[14]。该协议通过在 UDP 包头之后加上了一个带有数据包序列号字段的 RTP 包头，接收端可以根据该包头信息计算丢包率信息。而后发送端会根据丢包率信息将网络分为三个状态，即未装入、装入和阻塞。未装入表示数据发送速率小于网络带宽，可以继续增大；装入表示数据发送速率正好与网络带宽相契合，无需调整；阻塞则表示网络中发生了拥塞现象，需要降低数据的发送速率。这类基于丢包率的拥塞控制机制需要传输路径的路由相对稳定，保证传输路径不会发生巨大变化，另外丢包率统计的时间间隔的大小对拥塞控制的效果直接相关联，。如果时间间隔太小，会导致反馈信息加重网络负担，数据发送速率调整频繁，波动过大；而如果时间间隔过大则会导致速率调整不及时，不能实时地对网络拥塞现象做出处理。

基于网络往返时延的拥塞控制机制是根据网络往返时延的变化来判断网络

的拥塞程度。这类机制的基本思想是在发送端不断的获取网络往返时延的值，然后计算网络往返时延的变化率来判断网络拥塞程度，在这基础上对数据发送速率进行调整。比较典型的就是 Vegas[15]。该算法的主要参考因素是网络往返时延 RTT。该算法根据网络往返时延的变化情况来判断网络中是否出现的拥塞现象，即如果网络往返时延的值大幅度的提升，该算法就认为网络处于拥塞状态，相应的便会减小数据传输的速率；相反，如果网络往返时延慢慢变小，该算法则认为网络中无拥塞现象的发生，便会相应的增加数据传输速率，以达到资源利用的最大化。由于 Vegas 的拥塞控制机制不考虑网络往返时延的大小，而关注的是网络往返时延的变化率，因此对网络带宽的预测更加准确，而且基于端对端的方案，也能对减缓网络拥塞程度有明显的效果。但是该算法完全忽略了由于网络拥塞产生和由于误码产生的丢包对网络状态的影响，故无法精确的判断网络拥塞程度，无法从根本上解决网络拥塞问题。

2.3 无线网络下的拥塞控制机制

传统的拥塞控制机制是基于有线网络的，数据的丢包率是有线网络中检测网络拥塞程度的主要根据，而在数据误码率高、网络拓扑不稳定的无线网络中，数据丢包率不再是唯一的依据。除了数据丢包率，网络往返时延、误码率等也是网络拥塞的判别依据。在无线网络中，要综合考虑各类因素，才能做到有效地对网络拥塞进行控制，避免造成网络带宽的浪费等网络资源利用率低的后果[16]。目前无线网络下的拥塞控制机制主要三种，即基于端到端的拥塞控制机制、基于跨层的拥塞控制机制和基于分段的拥塞控制机制。

2.3.1 基于端到端的拥塞控制机制

基于端到端的拥塞控制机制的思想与传统有线网络中的拥塞控制机制类似，都是通过发送端和接收端获取网络的参数信息来判别网络的拥塞程度，然后对数据的发送速率做出调整。但是在无线网络中不仅要考虑丢包率，还需综合考虑由于无线信道造成误码对于丢包的影响，才能准确地判断网络拥塞程度。这类拥塞控制机制中比较经典的是 Westwood[17]算法。

Westwood 算法的基本思想是通过带宽估计算法直接计算出网络的可用带宽。具体的估算流程是发送端先对接收端的反馈的确认报文进行速率统计，之后把统计的接收端的确认报文的发送速率当做剩余带宽，以此来调整发送端的数据发送速率。而当发生网络丢包的时候，就把 TCP“慢启动”的阀值赋值为统计的值，这样 TCP 在发生丢包现象之后数据发送速率会调整为与网络可用带宽相近的水平。该算法不会因为丢包而盲目地降低数据发送速率，保证了网络资源的利用率。

但是对确认报文的速率统计需要有一定的时间消耗，而且统计的准确性难以保证，因此提高对确认报文速率的统计算法的准确性直接关系拥塞控制的效果。

2.3.2 基于跨层的拥塞控制机制

基于跨层的拥塞控制机制的思想在考虑到在无线网络中联系紧密的层次关系，在本层的拥塞控制机制可能会对其它层次产生重大影响，因此这类拥塞控制机制希望通过多层次拥塞控制之间的相互作用来完成整个无线网络的拥塞控制，比较经典的算法是 TFRC-ARQ[18]算法。

TFRC-ARQ 算法是基于跨层的解决方案。该协议的主要工作原理是当检测出由于误码产生的丢包情况时，会对该数据包进行重传，也就是在数据到传输层之前完成恢复，从而传输层就不会有由于误码产生的丢包现象。ARQ 协议从链路层入手，采用重传的机制，能有效地降低由于误码产生的丢包率，进而使传输层的拥塞控制协议工作更加准确有效。同时由于 ARQ 协议的重传超时值难以确定，而且过多的数据包重传次数也会导致整个传输网络整体性能的下降，同时影响其它数据的正常传输，因此 ARQ 协议也有其局限性。

2.3.3 基于分段的拥塞控制机制

基于分段的拥塞控制机制的思想是利用网络中的有线网络和无线网络的连接处，将网络连接分为有线和无线两个部分。在有线网络部分采用有线的拥塞控制机制，在无线网络部分采用针对无线网络的拥塞控制机制。这类拥塞控制机制能够提升丢包区分算法的准确性，因此提高了对网络拥塞程度判断的准确度，比较经典的有 M-TCP[19]算法。

M-TCP 是基于分段连接的解决方案。基本思想是将一条 TCP 连接分割成两段，一段是无线基站和固定节点之间的有线链路，另一端是无线基站和无线节点之间的无线链路。在有线链路上采用标准的 TCP 传输控制协议，即使用传统的 TCP 拥塞控制机制，而在无线链路上采用改进的 TCP 拥塞控制机制，这类思想通过提升无线链路的性能来提升整个网络的性能，但是将 TCP 分解，破坏了其连通性。M-TCP 针对无线基站和无线节点进行了改进。只有当数据包到达固定节点之后，才会返回确认包给无线节点，保证了 TCP 端到端的特性，同时采用 Freeze-TCP 的零窗口机制来解决无线网络的丢包问题。M-TCP 虽然基于分段的思想来处理网络拥塞状况，同时保留了 TCP 端到端的特性，能够对无线链路进行有针对性的性能提升，但是由于无线信号高误码率的特征，M-TCP 需要一个性能高的链路层协议来配合，同时分段的处理思想需要中间节点有足够的缓存来储存转发状态控制信息和数据包，因此缓冲区的开销也将变大。

通过介绍无线网络中三种基于不同思想的拥塞控制机制以及其典型的算法，可知，与有线网络不同，无线网络中网络往返时延、高误码率才是衡量网络拥塞的关键因素，因此传统的拥塞控制机制必须要进行改进，才能有效地在无线网络中发挥作用。

2.4 本章小结

本章重点介绍了与拥塞控制机制的相关知识和主要研究技术。本章先分析了拥塞现象的定义和其产生的原因，接着分析了拥塞控制的基本概念。然后详细的介绍有线网络中最基本的拥塞控制机制—TCP 拥塞控制，接着继续介绍了 UDP 协议下两种不同思想的拥塞控制机制。最后本章介绍三种无线网络下的拥塞控制机制的基本思想及其相关的经典算法，为本文中拥塞控制机制的设计与实现建立了基础。

第三章 无线多模网关中视频传输拥塞控制机制的设计

3.1 无线多模网关及视频传输系统综述

本文的目标是在无线多模网关中针对视频传输设计并实现一种高效的拥塞控制机制，因此要针对基于无线多模网关的网络结构特征以及视频传输系统的流程来完成拥塞控制机制的设计。

3.1.1 基于无线多模网关的网络结构

无线多模网关的多模性主要是指无线多模网关能够在多种通信方式中有选择地进行网络切换，即可以有多种网络进行通信方式的选择。因此，无线多模网关会根据接入设备上传送数据的类型选择合适的网络进行传输，而当某条线路的网络突然意外中断时，无线多模网关能够选择其他连通的网络继续进行数据传输。

基于无线多模网关的网络结构图如图 3-1 所示。

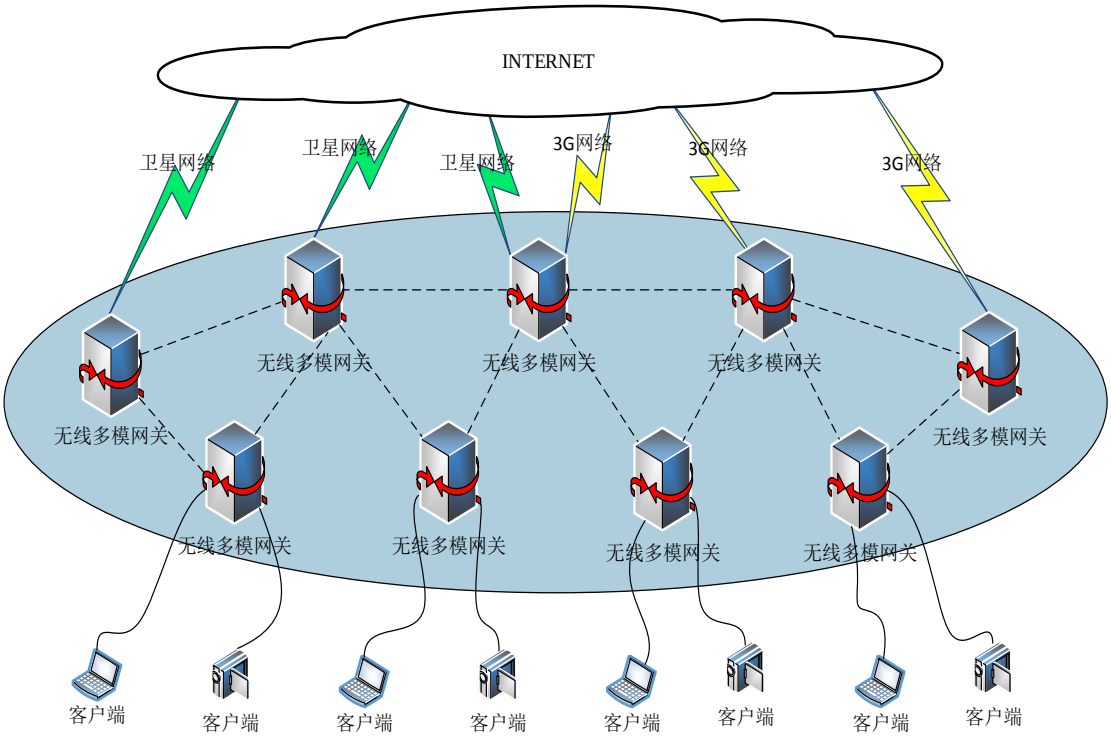


图 3-1 基于无线多模网关的网络结构图

从图 3-1 中可以看出，该网络的结构可以分为三层。最上层是卫星网络和 3G 网络，这两种网络覆盖范围广，传输距离远，而且能直接与 internet 相连通，可以直接的向互联网用户提供服务。卫星网络由于传输时延较大且费用比较昂贵，

一般用于提供文本数据的服务，而 3G 具有传输速率高而且资费合理，一般用于提供以视频为主的多媒体服务。中间一层是无线多模网关相互之间的连通网络。这种无线多模网关之间的连通网络以无线局域网（如 Wi-Fi）、mesh 无线网等为主要通信网络，相比于 3G 网络和卫星网络，无线多模网关之间的传输距离要近一些。这一层网络在数据传输过程中主要起转发作用，同时由于无线多模网关可移动性非常强，因此它的位置可以经常发生变化，这就直接导致了这一层的网络结构会随着无线多模网关的位置变化而变化，网络拓扑很不稳定。最底层是移动设备接入层，与之连接的是各类移动设备（如笔记本电脑、IP 摄像头等），这些设备是网络中传输数据的采集终端和接收终端。无线多模网关有多个移动设备的网络接口，这些移动设备可以通过网线直接与无线多模网关连接，再通过上两层网络完成数据的传输。

无线多模网关设备布置非常灵活，不受地理环境的影响，而且具有多种无线网络的接入，能为数据的传输提供多套传输方案，同时无线多模网关又可以根据数据的优先级选择合适网络来进行数据传输，因此在环境监测、应急通信、文化遗址监测中都能发挥很好的作用，而对于一些无信号覆盖的“死区”，还可利用无线多模网关群之间来完成数据的传输。因此，无线多模网关在数据传输中发挥的作用必将越来越大。

3.1.2 视频传输系统

当前的视频传输方式普遍采用的是流式传输，而不再是单纯的先下载后播放的方式。流式传输是一种边下载边播放的视频传输方式，因此对于视频数据包的实时性要求较高。传统的 TCP 对于数据传输由于有超时重传等机制，虽然能够保证数据的可靠性，却极大的破坏了数据的实时性。所以视频传输一般选择实时性更加优秀的 UDP 协议。UDP 协议是一种面向无连接的传输协议，它资源消耗极小，处理数据包的速度快，但是 UDP 也有其局限性。由于 UDP 传输没有拥塞控制机制，视频传输的速率不会受网络拥塞状态的影响，因此当网络拥塞发生时，还是会有大量的视频数据在发送，不仅视频传输的效果极差，还会导致网络拥塞程度的加重。

传统的视频传输系统一般包括发送端和接收端两部分。各部部分具体的模块图如图 3-2 所示。

从图 3-2 中可知，发送端包括视频采集模块、视频编码模块和视频发送模块。视频采集模块负责原始视频数据的采集，一般采用的 YUV 或者 RGB 编码，此时的视频数据一般数据量比较大，不适宜直接进行传输，因此视频采集模块会把采集的原始视频数据传递给视频编码模块。视频编码模块主要负责将原始视频数

据进行压缩编码，来生成适合网络传输的压缩视频帧数据。经过视频编码模块之后的视频数据一般只有原始视频数据的一半，甚至更少，因此视频编码模块极大的减轻了网络传输的负担。经过视频编码模块压缩编码后的视频数据传送至视频发送模块来完成视频数据的发送。这个过程主要是采用 UDP 协议来完成。视频发送模块会将视频数据帧切割为适合 UDP 传输的数据块，然后让后端网关发送视频数据。视频发送模块是控制着视频数据的发送速率，是视频传输拥塞控制的主体，但是传统的视频传输系统并没有拥塞控制功能。

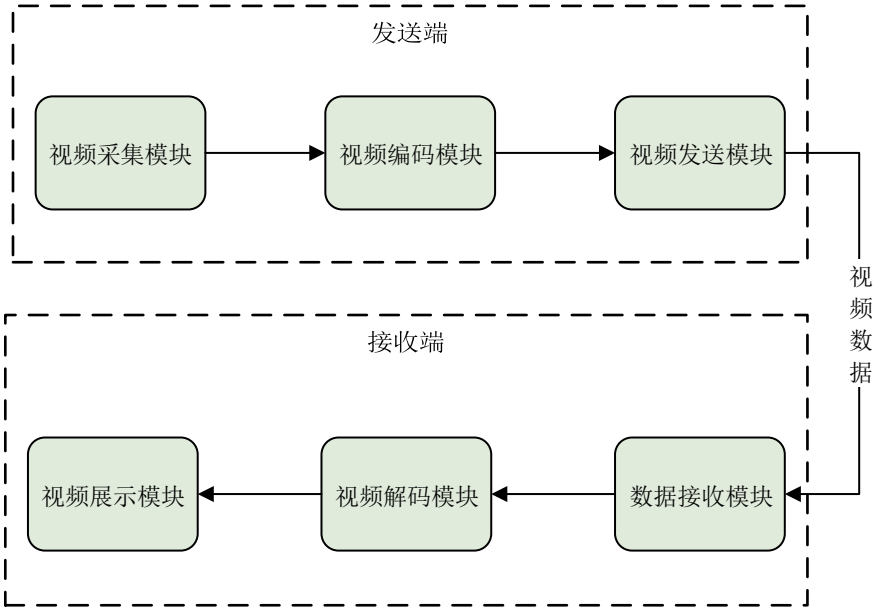


图 3-2 传统视频传输系统模块图

视频数据经过中间网络传输至视频传输系统的接收端。接收端主要包括数据接收模块、视频解码模块和视频展示模块。数据接收模块主要功能是接收经过 UDP 传输的视频数据包，并根据其中的数据包头信息进行视频数据帧的重组。若不考虑丢包误码等因素的影响，经过数据接收模块后，将会得到是一个完整的视频数据帧。接着视频数据进入视频解码模块进行解码。视频解码模块是视频编码模块的逆过程，功能是将压缩的视频数据帧解码成为原始数据帧，以方便视频数据帧的展示。解码后的视频数据最后进入视频展示模块进行视频的播放展示，由此可知，为了视频展示的流畅性和实时性，视频传输对于网络延时的要求很高。视频展示模块是体现视频服务好坏的最直接表现，是评估一个视频传输系统服务质量的最直接因素。

3.2 总体设计

本文的目标是针对无线多模网关中的视频传输设计一种高效的拥塞控制机制，该机制在保证视频传输的服务质量的基础上，不会抢占其他数据传输的带宽，同时可以根据网络拥塞程度，调整视频数据的传输速率，不加重网络的拥塞程度。

首先本文设计的拥塞控制机制的环境是基于无线多模网关的网络。该网络是集成了多种无线网络的网络，数据在其中传输时可能会经过不同的传输网络最终到达服务器。同时该网络本身的网络拓扑结构不稳定，网络往返时延是判别网络结构变化的关键因素，因此是判定该网络拥塞程度的重要因素。

其次本文设计的拥塞控制机制针对的是视频应用程序，因此本文所设计的拥塞控制机制是基于应用层的，是专门针对视频传输系统的。而在应用层进行视频传输拥塞控制不需要对现有的传输协议进行改变，而是在应用层对视频发送速率及发送缓存等方面做出调整，因此具有更大的灵活性，也是当前视频领域研究的重点。同时视频数据由于其传输数据量极大的特点，丢包率是衡量其网络拥塞程度的重要参考因素。另外对于无线网络而言，误码会极大的影响丢包率的值，因此有必要对数据误码进行区分处理，降低误码对于网络拥塞判别的影响。

最后，本文的拥塞控制机制首先是基于最基本的视频发送速率的控制。对视频发送速率的控制要求在公平共享带宽的基础上，拥塞控制机制能随着网络结构的变化，有效地控制视频数据传输速率，降低网络拥塞程度。其次，本文还针对是视频数据的发送设计了视频发送缓存控制，二者相互结合来完成本文设计的拥塞控制机制。

根据以上需求分析，本文为视频数据在无线多模网关中的传输设计一套拥塞控制机制，具有该机制的视频传输系统总体设计图如图 3-3 所示。

从图中可以看出，本文的拥塞控制机制在视频传输系统的发送端和接收端都有涉及。发送端是拥塞控制的关键，接收端则是反馈数据的来源，为拥塞控制提供决策数据。

在发送端，主要通过两种途径来实现对于视频数据的拥塞控制。第一个方面是视频数据发送速率的控制。由前面的分析可知，网络拥塞控制的关键就是传输数据速率的控制，因此视频数据发送速率的控制必然是本文拥塞控制机制的核心，本文通过改进 TCP 友好速率控制模型[20]来完成视频数据速率的控制。另一个方面是视频发送缓存的控制，是针对网络存储不足造成网络拥塞提出的控制机制。在视频发送速率控制的基础上，作为有效的补充，能够辅助视频发送速率控制完成本文的拥塞控制机制。本文所述的拥塞控制机制就是将这两种控制机制相结合，来完成对视频数据的拥塞控制。除此之外，在发送端，还有网络参数反馈模块，这个模块的主要功能是获取网络参数，包括网络往返时延和视频数据丢包率，这些参数是判断基于无线多模网关的网络的拥塞程度的关键，也是为本文拥塞控制机制控制决策的基础。

在接收端，涉及的主要是视频数据的统计，然后把视频数据丢包率统计信息反馈至发送端。这个视频丢包率统计主要在视频传输系统的数据接收模块之后完

成，本文通过有效的处理，有效地降低了误码对于视频丢包率的影响，使本文的拥塞控制机制对于网络拥塞程度的判定更加准确。

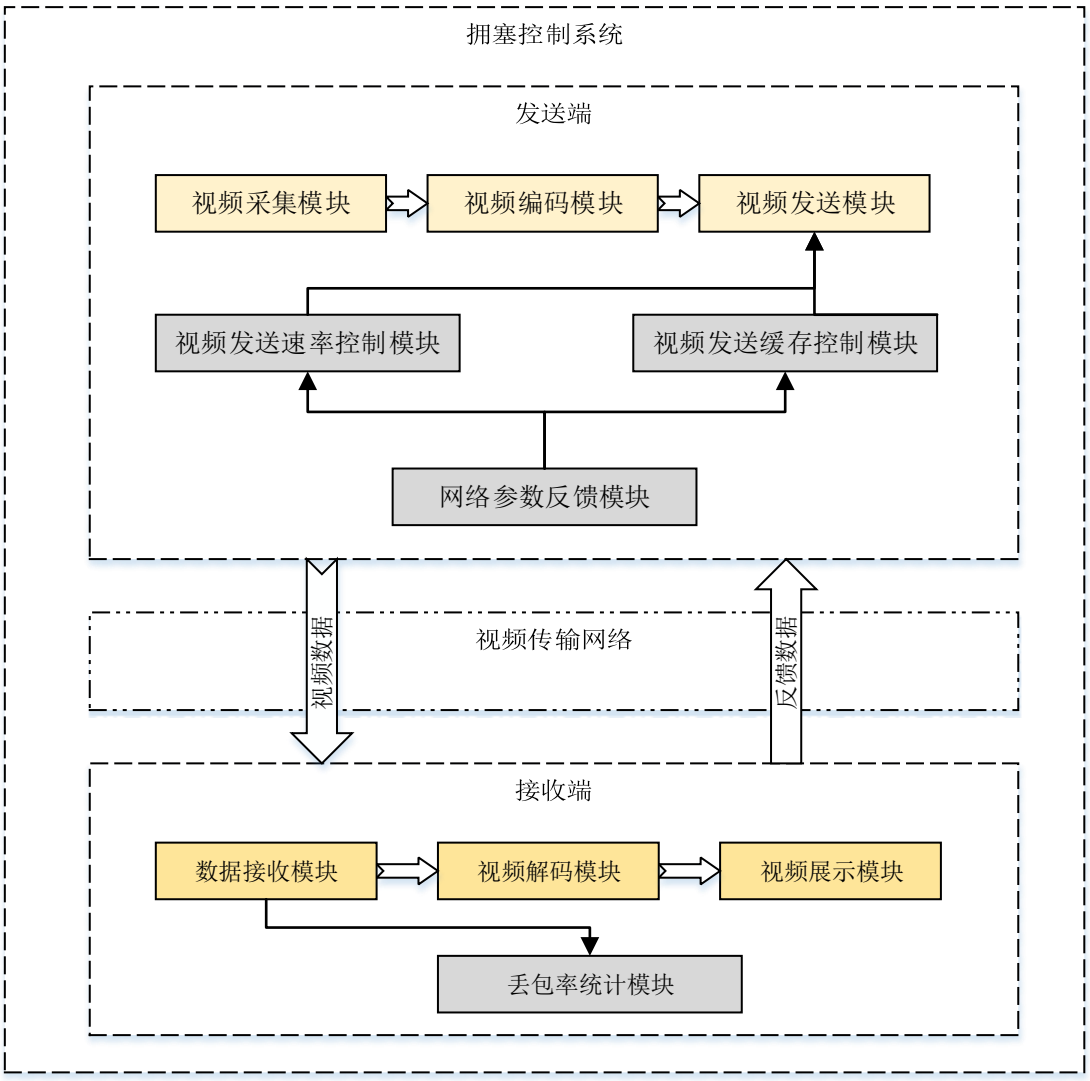


图 3-3 具有拥塞控制机制的视频传输系统总体设计图

3.3 详细设计

3.3.1 网络参数反馈模块的设计

在本文中，网络参数是衡量一个网络传输服务质量的各类参数集合，能够为网络状态变化提供依据，是判断网络拥塞程度的关键因素。网络参数主要包括数据丢包率、网络往返时延、数据延时抖动等。类型不同的网络会根据不同的网络参数判断网络拥塞程度，如判断有线 TCP 传输的拥塞程度主要依靠的数据丢包率。考虑到无线多模网关的网络结构特点，而且本文的应用目标是数据量极大的视频数据传输，因此所关注的网络参数主要有视频数据丢包率和网络往返时延。

网络参数反馈模块的功能就是获取这两种网络参数的值。

（1）视频数据丢包率

视频数据丢包率是指所丢失的视频数据包数量占所发送的数据包的比例。具体来说，假设发送端为 S，接收端为 R，在 t 时间内，发送端 S 发送的视频数据包的数量为P_s，接收端 R 接收到的视频数据数量为P_r，则整个网络在 t 时间内的视频数据丢包数P_l如公式 3-1 所示。

$$P_l = P_s - P_r \tag{3-1}$$

在 t 时间内的视频数据丢包率r公式 3-2 所示，

$$r = \frac{P_l}{P_s} = \frac{P_s - P_r}{P_s} \tag{3-2}$$

视频数据丢包率是衡量网络拥塞状况的重要参数，视频数据丢包率越高，则说明网络拥塞程度越严重。

根据上面对视频数据丢包率的分析可知，视频数据丢包率获取模块主要在视频传输系统的接收端统计，然后以反馈数据的方式发送给发送端。视频数据丢包率获取模块具体的设计图如图 3-4 所示.

从图 3-4 中可知，在接收端，主要有丢包率统计模块。丢包率统计模块的工作原理是在接收视频数据包时，会统计所接收到的视频数据包的数量和接收到的最大视频数据包的分片号。假设本时段内视频数据包的最大分片号 CurSMax,前一时段内的视频数据包的最大分片号为 PreSMax，那么本时段内发送端所发送的视频数据包数就为（CurSMax－PreSMax），再通过与该时段内接收端统计接收的视频数据包数计算商值，与 1 求差即可得出丢包率的值。这样的好处是不管接收到的视频数据包是否完整，只要接收到了，就会计入接收的视频数据包数量中，有效地降低了误码对丢包率的影响。通过该方法计算得出丢包率之后，丢包率统计模块会将丢包率发送给视频传输系统的发送端。

在发送端，主要有丢包率接收模块和网络参数反馈模块。丢包率接收模块主要负责丢包率的接收，同时会对本次接收到的丢包率与上一次的丢包率进行平滑操作，操作原理根据如公式 3-3 所示。

$$r_n = a \times r_n + (1 - a) \times r_{n-1} \tag{3-3}$$

其中r_n代表第 n 时段的丢包率，r_{n-1}为第 n-1 时段的丢包率，a是平衡因子，值在 0~1 之间，一般取 0.75。视频数据丢包率经过平滑处理之后，可以减少突发事件引起丢包率产生的较大抖动的程度。最后，视频数据丢包率信息将送往网络参数反馈模块。

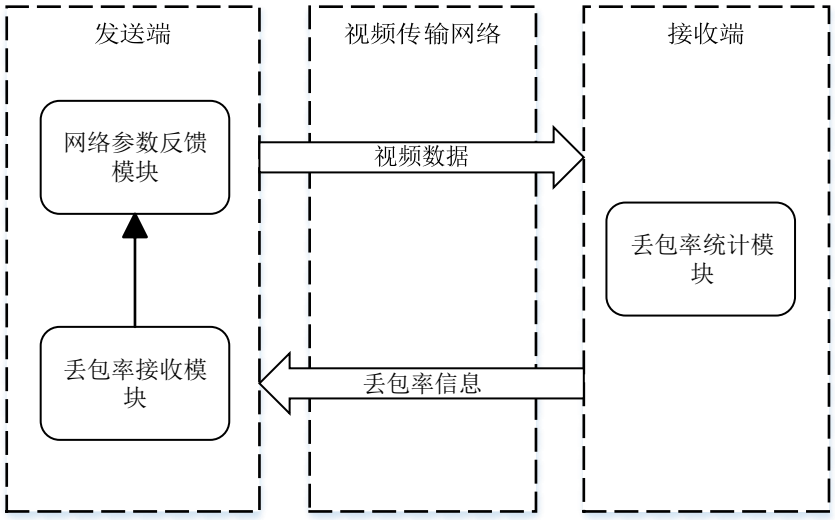


图 3-4 视频数据丢包率获取模块图

（2）网络往返时延

网络往返时延是指数据包从发送端发送开始到接收到来自接收端的确认所经过的全部时间。假设 P 为接收端 R 接收到的数据包，T_1 为 P 从 S 端发送开始的时刻，T_2为 S 接收数据包 P 的确认包的时刻。设网络往返时延为T_D，则有公式3-4。

$$T_D = T_2 - T_1 \tag{3-4}$$

根据网络往返时延可以估计该网络上的拥塞情况和信息传输的状况，同时也可以估算网络 S 端到 R 端的距离。

根据对网络往返时延的分析可知，得出网络往返时延计算模块具体的设计图如图 3-5 所示。

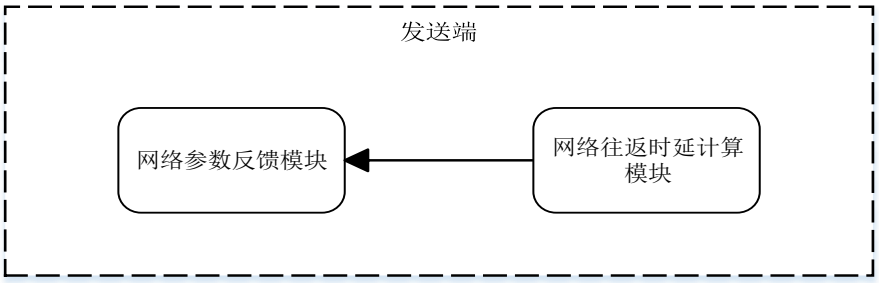


图 3-5 网络往返时延计算模块图

从图 3-5 中看出，整个网络往返时延计算模块都工作在视频传输系统的发送端。网络往返时延计算模块的功能是获取当前的网络往返时延，利用网络层的ICMP 协议来实现。ICMP 协议规定了网络中任何一台具有 TCP/IP 协议的设备都可以将该报文转发或者回应该报文。如果网络情况正常则回应查询报文，如果网络状况出现异常则回复携带错误编号的错误报文。ICMP 协议的报文格式如图 3-6 所示。

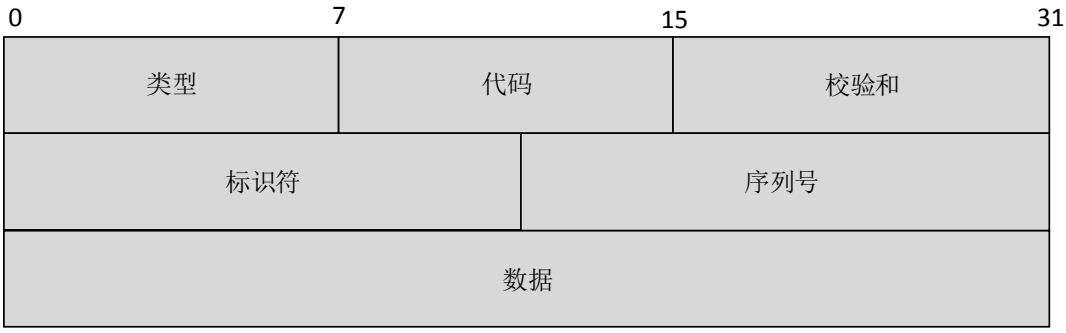


图 3-6 ICMP 数据报文格式

其中类型字段和代码字段是用来区分 ICMP 报文类型，校验和字段是用来检验 ICMP 报文的损坏情况，标识符标示该 ICMP 报文的进程号，序列号是区分查询报文和回应报文的顺序。数据字段则可以是用户自定义的数据。获取网络往返时延的一般步骤是先构建 ICMP 报文，包括类型、标识符和序列号等，其中数据字段携带的是当前的时刻信息。其次就是 ICMP 报文的发送。最后是接收 ICMP 应答报文，通过解析报文数据段中的时刻信息，与接收应答报文的时刻信息计算差值，就可以得出网络往返时延的值。为了防止突发情况的发生，一般采用发送多次 ICMP 报文来获取多次的网络往返时延的值，最后求它们的平均值当做本次的网络往返时延的值。得到网络往返时延的值后，会将网络往返时延进行平滑操作，依据如公式 3-5 所示。

$$T_n = a \times T_n + (1 - a) \times T_{n-1} \quad (3\text{-}5)$$

其中T_n代表第 n 次的网络往返时延，T_{n-1}为第 n-1 次的网络往返时延，a是平衡因子，值在 0~1 之间，一般取 0.75。网络往返时延经过平滑处理之后，可以减轻突发事件引起网络往返时延产生较大抖动的程度。网络往返时延最后发送至网络参数反馈模块。

反馈模块通过上文所述的两种不同的方式分别获取视频数据丢包率和网络往返时延。经过平滑处理之后，使这两个参数的数据更加平滑，也降低了突发事件的影响。获取的网络参数会传送至视频传输速率调整模块和视频发送缓存调整模块，这两个模块会根据网络参数判断网络的拥塞程度，做出相应的调整策略，以完成降低网络拥塞程度的目标。

3.3.2 视频发送速率调整模块的设计

由于视频数据传输要求较高的实时性，一般采用 UDP 来进行传输。然而，UDP 协议是个面向无连接的简单协议，缺乏拥塞控制机制，没有任何服务质量的保证机制。同时，在 UDP 连接与 TCP 连接共享同一个瓶颈网络的情况下，当 TCP 连接检测到网络拥塞时，它会使用拥塞控制机制不断地降低数据发送速率，而 UDP 连接由于没有拥塞控制机制，将会挤占大部分带宽乃至所有带宽，会导

致 TCP 连接发生拥塞甚至停止[21]。因此，要对视频数据的传输进行拥塞控制首先要对视频数据的发送速率进行控制，同时要保证网络对 TCP 和 UDP 的公平性。

在对视频发送速率控制的方法里，总体可分为两类：基于探测的速率控制方法和基于模型的速率控制方法。

基于探测的速率控制方法的基本思路类似于 TCP 协议的拥塞控制 AIMD 算法。该方法在视频传输系统的发送端和接收端都维持一个拥塞窗口，每个数据包会占用拥塞窗口的一个槽，而当接收端接收到一个数据包时会发送一个应答包，发送端接收到应答包时，就会释放一个槽，只有发送端有空槽时才会发送数据包。当网络拥塞程度很轻的时候会增加拥塞窗口，而当网络拥塞程度严重时则会减少拥塞窗口。

采用 AIMD 算法进行速率控制时，根据公式 3-6 计算视频数据发送速率，

$$T = \frac{\sqrt{2-b}\times\sqrt{a}}{\sqrt{2b}\times RTT\times\sqrt{p}} \qquad (3\text{-}6)$$

其中 p 表示视频数据丢包率，a表示拥塞窗口的调节增量常数，一般设置为 1，b 是拥塞窗口调节的倍减参数，一般值为 0.5，T 视频数据发送速率。根据该公式，拥塞窗口的调整机制是经过一个网络往返时延的时间后，有视频数据包丢失时，拥塞窗口变为原来的（1-b）倍，而如果没有视频数据包丢失，则把拥塞窗口增加a。

基于探测的速率控制算法有一些典型的改进算法，如二项式拥塞控制算法、平方增加积式减少控制算法和速率自适应算法等。这种类型的算法本质上都是基于对 TCP 拥塞控制算法的模拟，虽然可以极大程度上利用网络带宽，并且速率调整平滑，但是需要数据包确认应答，对于数据量巨大的视频数据来说，加重了网络负担，显得多余。另外调整时间过长，积式的速率减少机制也会带来视频传输速率的大幅波动，而且这类方法不适用于网络结构不稳定的网络。

基于模型的速率控制方法的基本思想是通过该网络的一些网络参数，对其建立精确的速率模型来估计在当前的网络环境下视频数据所应该占有的带宽，发送端根据该模型估计出来的带宽来完成对视频数据发送速率的调整。这类方法虽然建立模型会有一定的时间开销，而且若频繁建立速率模型，则会导致速率波动过大，但是这类方法可以明确地估计网络可利用带宽，调整发送速率到最佳状态的时间短，对网络结构变换处理及时，适用于网络结构经常变换的网络，因此是被讨论的最多的，也是最具发展潜力的速率控制策略。比较典型的 TCP 吞吐量模型。

由于本文面向的是基于无线多模网关的网络，网络结构是不稳定的，故选择基于模型的速率控制方法。同时为了保障视频数据流与其他 TCP 数据流公平地竞争网络带宽，本文引入了“TCP 友好（TCP-Friendly）”[22]的概念。所谓“TCP

友好”是指在相同的网络条件中，若一个非 TCP 流在很长时间内，吞吐量始终小于或等于同等条件（相同的传输路径、丢包率、包的规模、网络往返时延）下的 TCP 流，那么该非 TCP 流就能被判定是 TCP 友好的。本文采用的是基于模型的 TCP 友好速率控制机制——TFRC(TCP Friendly Rate Control)[23,24]。 TFRC 根据“TCP 友好”的概念，得出视频数据的发送速率的上限应该是同期 TCP 数据流的发送速率，即可以使用 TCP 吞吐量计算公式来计算 TCP 友好的视频数据的发送速率[23,24]。计算公式见公式 3-7。

$$T_s = \frac{s}{RTT\times\sqrt{\frac{2}{3}p}+RTO\times\left(3\times\sqrt{\frac{3bp}{8}}\right)\times p\times(1+32p^2)} \quad (3\text{-}7)$$

其中T_s为发送速率；s 为数据包大小；RTO 为重传超时时间，一般设为 RTT 的 4 倍；RTT 为网络往返时延，本文中所提到的重要网络参数；b 为每个 ACK 确认的分组个数，通常该值为 1；p 为网络丢失事件率，在本文中为视频数据丢包率。

TFRC 采用 TCP 连接的吞吐量模型，通过测量当前的网络参数，包括丢包率和网络往返时延，得出适合当前网络状态的数据发送速率，是一种比较完善的速率控制算法。但是 TFRC 频繁的模型建立会花费大量的时间，导致速率调整的延迟和数据发送速率频繁的抖动。

为了防止视频发送速率频繁地抖动，在视频发送速率调整模块引入速率微调机制和速率最佳机制。同时，引入这两种机制减少了 TFRC 速率模型建立次数，因此也降低了速率调整的延时性。根据以上分析知，速率调整模块工作在视频传输系统的发送端，具体模块设计图如图 3-7 所示。

在本文所述的视频传输系统中，发送端分为六大模块，除了传统的视频采集模块、视频编码模块、视频发送模块，还增加了网络参数反馈模块、速率调整模块和缓存调整模块。视频采集模块主要负责原始视频数据（YUV 视频帧）的采集，而视频编码模块则采用 H.264 编码形式对原始视频数据进行压缩编码，视频发送模块负责将压缩编码后的视频数据以 UDP 连接的形式通过多模无线网络发送至视频接收端，网络参数反馈模块负责网络参数的获取，本文上一小节已具体描述，缓存调整模块将在下一小节进行具体描述。

速率调整模块是核心模块之一。速率调整模块首先会进行参数获取，获取的参数主要包括视频数据丢包率和网络往返时延。参数获取具体是由网络参数反馈模块来完成，其中视频数据丢包率主要通过接收来自视频接收端的统计信息来获取，而网络往返时延则通过发送端接收来自接收端反馈的 ICMP 应答包来计算获取。在获取完参数之后，将进入调整判定模块对参数进行分析，其分析流程如下图 3-8 所示。

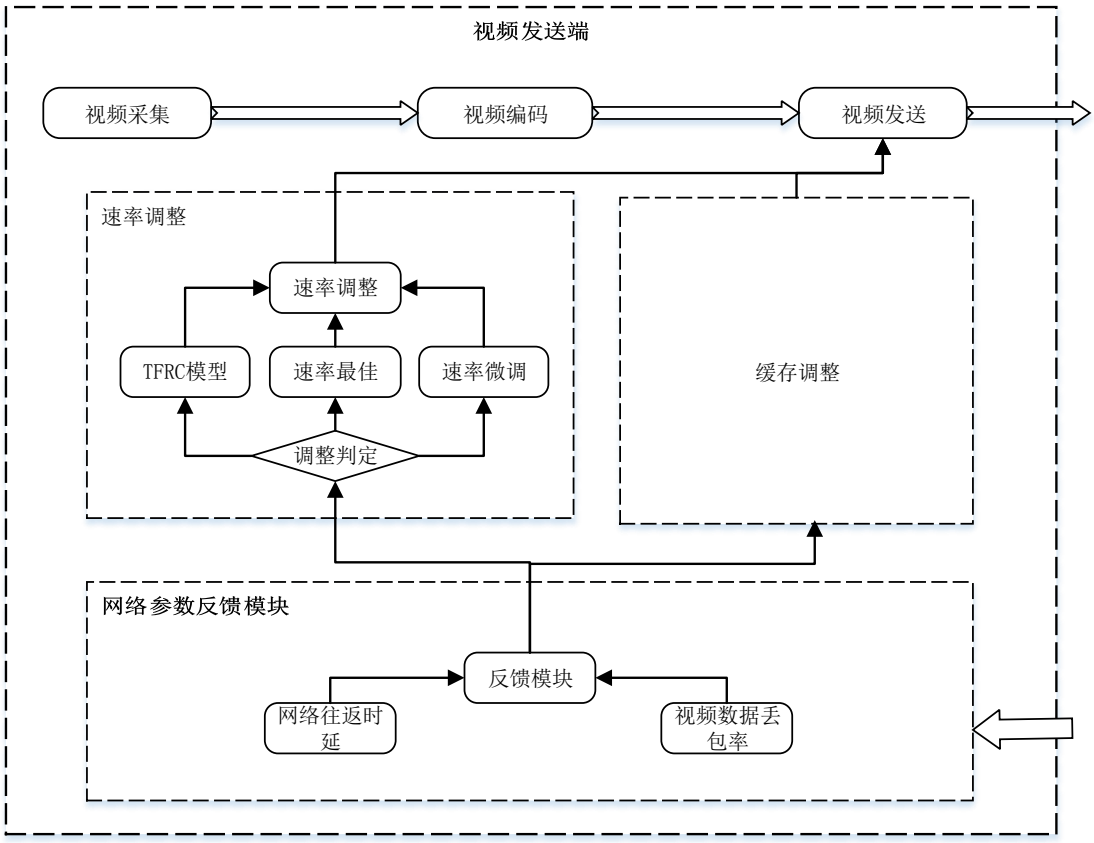


图 3-7 具有速率调整的发送端模块图

本文引入了三个参数 k，m 和 n，它们的值都在 0 到 1 之间，其中 m < n。k 代表根据网络往返时延变化率判别网络拥塞程度的阀值，m 和 n 代表根据视频数据丢包率来判别网络拥塞程度的阀值。

从 3-8 图中可以看出，如果网络往返时延变化率不超过 k 时，本文认为网络结构没有发生变化，此时在这个基础上，对视频数据丢包率进行如下分析：（1）如果视频数据丢包率高于 n，则认为网络拥塞程度严重，视频发送速率不合理，速率调整模块需根据新的网络参数进行 TFRC 模型重建，重新计算视频发送速率；（2）如果视频数据丢包率在 m 和 n 之间时，则认为 TRFC 速率模型合理，但是视频发送速率有些偏高，将进入速率微调模块，稍微降低视频发送速率；（3）如果视频数据丢包率低于 m 时，则认为视频发送速率此时最佳，将进入速率最佳模块，视频发送速率保持不变。而当网络时延变化率超过 k 时，则认为网络结构发生变化或者网络拥塞程度十分严重，此时为了保证视频发送速率的合理性，不论视频丢包率为何值，都会重新建立 TFRC 模型，重新计算视频发送速率。

经过上述的调整判定模块之后，速率调整模块都会得到一个新的视频发送速率调整策略，以此策略得到一个新的视频发送速率。总结各调整机制功能及判定条件如表 3-1 所示，

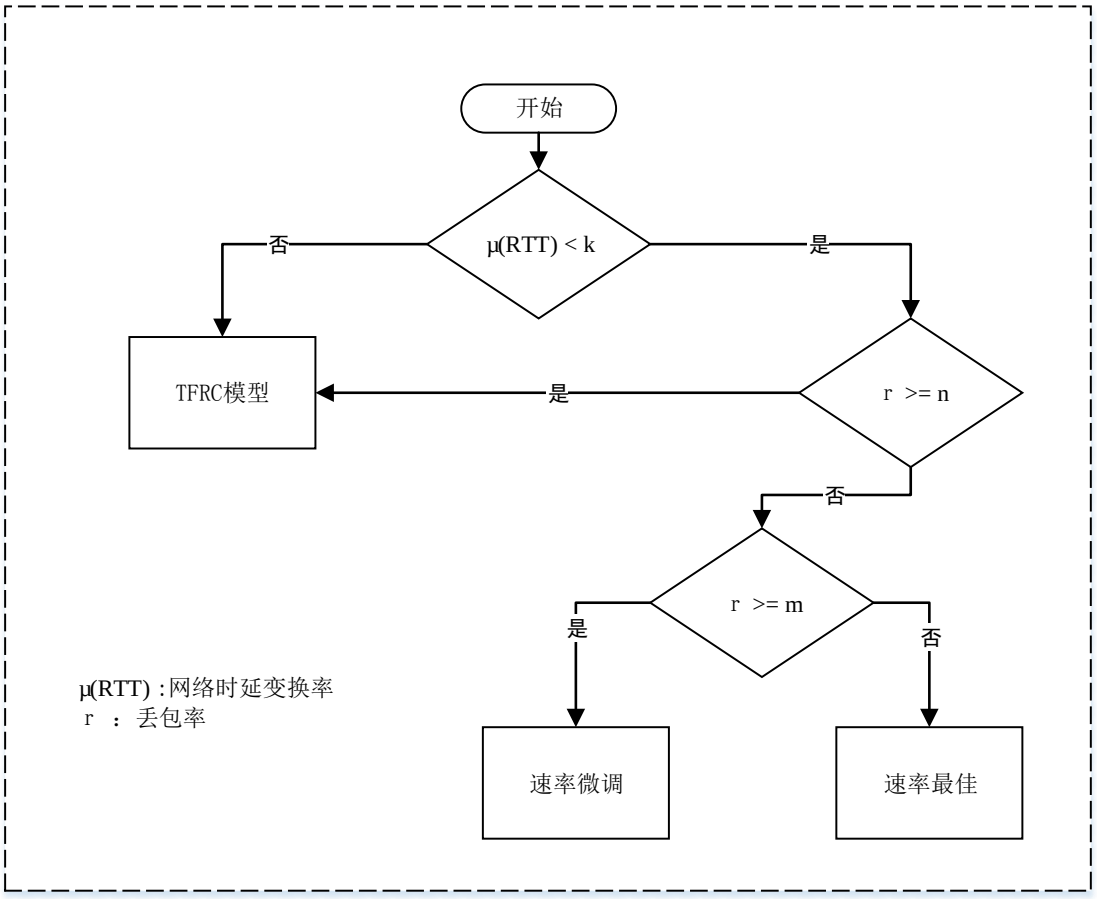


图 3-8 速率调整判定流程图

表 3-1 速率调整机制功能与机制判定条件

调整机制	功能	机制判定条件
TFRC 模型	根据 TCP 吞吐量公式重新计算视频发送速率	（网络往返时延变化率大于 k）或（网络往返时延变化率小于 k 且视频数据丢包率大于 n）
速率微调	视频发送速率减 1	网络往返时延变化率小于 k 且视频数据丢包率在 m 和 n 之间
速率最佳	视频发送速率保持不变	网络往返时延变化率小于 k 且视频数据丢包率小于 m

此后会对视频发送速率进行平滑操作，依据的公式 3-8

$$V_n = a \times V_n + (1-a) \times V_{n-1} \quad (3\text{-}8)$$

其中V_n代表第 n 次的视频发送速率，V_{n-1}为第 n-1 次的视频发送速率，a是平衡因子，值在 0~1 之间，一般取 0.75。视频发送速率经过平滑处理之后，进一步减

轻视频发送速率抖动的程度。最后速率调整模块将经过调整的视频发送速率向视频发送模块反馈，来完成视频发送速率调整。

本文通过引入 TFRC 速率控制模型作为视频速率控制机制的基本方法，分析 TFRC 模型的不足，并针对该模型不足的地方进行了改进。本文根据网络往返时延和视频数据丢包率这两个网络参数判别网络拥塞程度和基于无线多模网关的网络结构变化，并提出了速率微调机制和速率最佳机制，来弥补 TFRC 速率控制模型的不足，最终将 TFRC 模型状态和这两个状态相结合，完成本文的视频发送速率控制机制。

3.3.3 视频发送缓存调整模块的设计

本文所述的无线多模网关中的视频传输拥塞控制机制在传统的视频发送速率控制机制之外，还增加了视频发送缓存调整机制，二者结合来完成视频的拥塞控制。

在网络拥塞产生的三大因素中，网络带宽和网络节点的处理能力都是在网络设施布置的时候就已经确定了，要进行更改提升只能人为替换网络硬件设施，而无法从软件方面入手。而另外一个因素存储空间是可以从软件角度进行调整。本文的视频发送缓存机制就是基于存储空间这个因素来实现拥塞控制。

视频发送缓存调整机制的基本思想就是当网络拥塞程度严重时，增加视频发送缓存；而当网络拥塞程度减轻时，会减小视频发送缓存，一方面是减少缓存的浪费，另一方面也是一些轻量级视频采集设备的资源紧张的必然需求。本文的视频发送缓存调整机制从 UDP 发送缓冲区的调整和视频传输跳帧数的调整两方面入手。

首先，本文所述的视频数据的传输是依靠 UDP 协议来进行的，因此对 UDP 发送缓冲区的大小进行调整是直接对视频发送缓存的调整。其次，在视频传输中，通常不是摄像头采集的所有视频数据帧都会进行传输，因为相连的视频数据帧往往差别很小，没必要全部发送；另外所有的视频数据帧都发送会加重数据的传输量，增加网络的负担。因此视频传输一般都会设置视频传输的跳帧数。假设跳帧数为 f，则视频发送模块会每隔 f 个视频数据帧后选择一个数据帧发送。可知在单位时间内，如果视频传输跳帧数越大，则要传输视频数据量变少，而视频传输跳帧数越小，则视频传输的数据量越大。因此可以通过视频传输跳帧数来控制视频传输数据量的大小，从另一个角度完成视频发送缓存的调整。

发送端增加缓存调整模块之后，具体模块流程图如图 3-9 所示。

从图 3-9 中可知，缓存调整模块会得到的网络参数反馈模块的网络往返时延和视频数据丢包信息。之后进入调整判定模块进行参数判定，判断网络拥塞程度，

然后通过从 UDP 发送缓冲区和视频传输跳帧数两方面制定缓存调整策略，反馈给视频发送模块完成视频发送缓存调整机制。

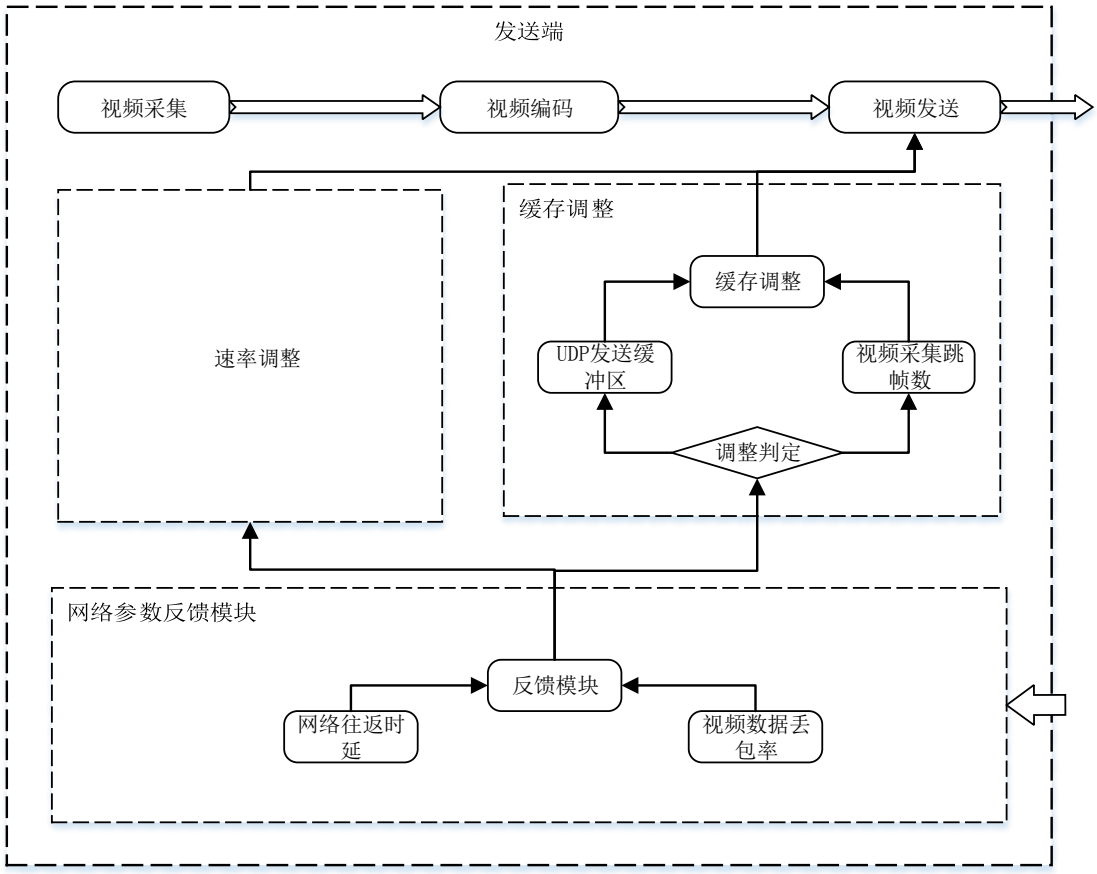


图 3-9 具有缓存调整的发送端模块图

调整判定模块中各参数与调整状态关系如图 3-10 所示，其中 k，m，n 是本文上一小节所述的根据视频丢包率和网络往返时延判定网络拥塞程度的阀值。

从图 3-10 中知，当网络往返时延变化率超过 k 时，本文认为多模网络结构发生变化或者网络拥塞程度严重，此时会同时对视频传输跳帧数和 UDP 发送缓冲区进行调整。具体做法是先判断网络往返时延是增长还是减少。当网络往返时延增大，则视频传输跳帧数增加一帧，UDP 发送缓冲区增加一个单位（一般以视频采集原始数据帧的大小为一个单位）；当网络往返时延变小，则视频传输跳帧数减少一帧，UDP 发送缓冲区减小一个单位。而当网络传输变化率小于 k，本文认为多模网络结构没有发生变化，此时根据视频丢包率的情况进行调整。如果视频丢包率大于 n，本文认为网络拥塞程度非常严重，则把视频传输跳帧数增大一帧，减少视频数据发送量。而视频丢包率在 m 和 n 之间时，本文认为拥塞程度情况不严重，视频发送缓存无需调整，而通过视频发送速率调整来实现拥塞控制。而当丢包率小于 m 时，本文认为网络没有出现拥塞状况，会将 UDP 缓冲区减少一个单位，减少内存浪费。

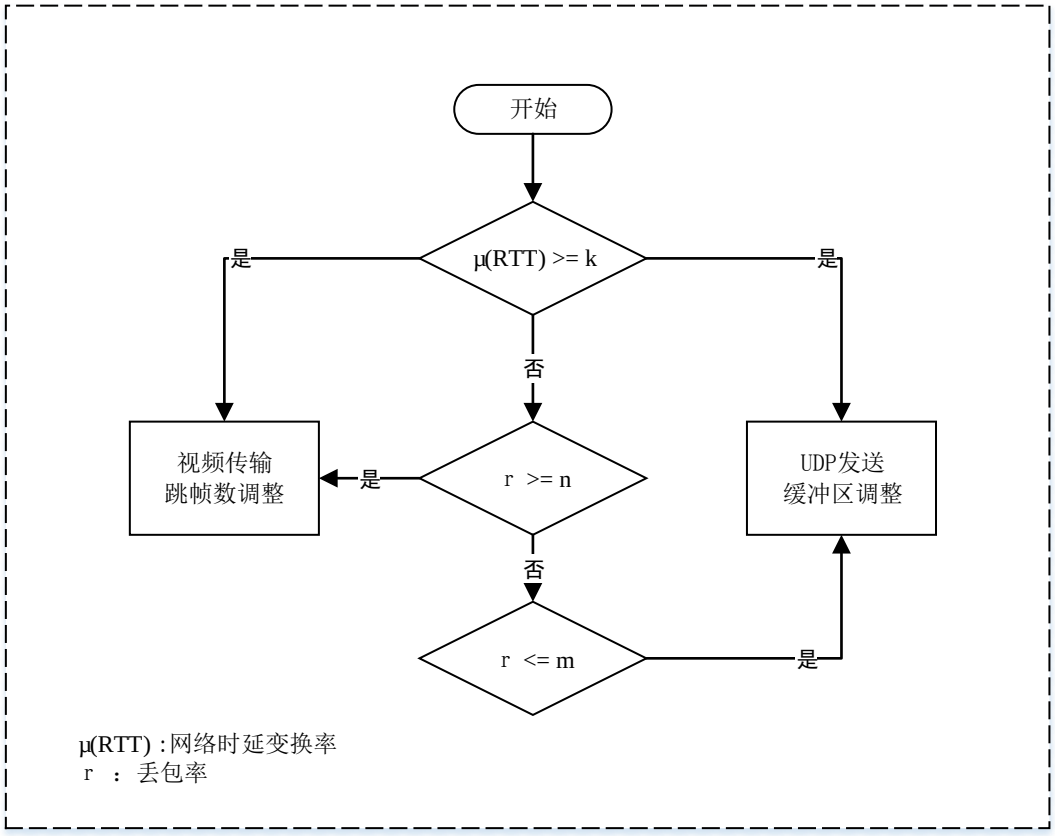


图 3-10 缓存调整判定流程图

经过上述的调整判定模块之后，缓存调整模块都会得到一个新的视频发送缓存调整策略。总结各调整机制功能及判定条件如表 3-3 所示。

表 3-2 缓存调整机制功能与机制判定条件

调整机制	功能	机制判定条件
UDP 发送缓冲区调整	UDP 缓冲区增加或减少一个单位	（网络往返时延变化率大于 k）或（网络往返时延变化率小于 k 且视频数据丢包率不大于 m）
视频传输跳帧数调整	视频跳帧数增加或减少一帧	（网络往返时延变化率大于 k）或（网络往返时延变化率小于 k 且视频数据丢包率不小于 n）

最后缓存调整模块将经过根据网络参数制定的调整策略向视频发送模块反馈，来完成视频发送缓存调整。

3.4 本章小结

本章通过介绍基于无线多模网关的网络结构特征以及视频传输系统的基本

知识，给出了无线多模网关中视频传输拥塞控制机制的总体设计，提出了从视频发送速率控制和视频发送缓存控制两个方向着手的方案。同时本章还对视频发送速率控制和视频发送缓存控制分别给出了详细的模块设计，为本文的拥塞控制机制的实现建立基础。

第四章 无线多模网关中视频传输拥塞控制机制的实现

4.1 开发环境介绍

本文所述的拥塞控制机制是应用在无线多模网关上的视频传输系统上。无线多模网关上运行的是嵌入式 Linux 系统。

Linux 是基于 Unix 的操作系统，具有支持跨平台、网络接口全面、外设管理简洁、多任务、免费等特点。同时，由于其开源的特性，可以进行用户私有化定制与裁剪，非常适合作为轻量级嵌入式操作系统。作为系统硬件与应用软件之间的桥梁，嵌入式 Linux 将外设视作文件，与其它类型文件管理方式相似，同时，Linux 极为灵活，可搭载多种文件系统[25]。

由于嵌入式 Linux 应用广泛、性能稳定、接口全面、可定制可裁剪，本文使用 Linux 架构作为软件开发和运行环境，能够满足目前视频传输拥塞控制的功能需求。

其次，程序源代码需要编译、链接后，才能被执行。对于嵌入式开发而言，程序需要被嵌入式平台的处理器执行，所以编译的目标文件应当是与目标平台相匹配，即可被嵌入式处理器执行的二进制可执行文件。由于嵌入式设备的处理性能、存储性能、人机交互环境与传统 PC 开发环境差异很大，开发时具有很大局限性。所以通常使用个人电脑的系统作为开发环境，使用交叉编译器进行程序跨平台的开发与编译，这一过程称为交叉编译。编译过后生成的二进制文件无法在宿主机（个人电脑）上运行，必须将其移植到目标机（嵌入式平台）上运行。本文选用的是 ARM 提供的 eabi 交叉编译器，此编译器通常被使用于 Linux 系统平台。

4.2 网络参数反馈模块的实现

本文所述的拥塞控制机制所涉及的网络参数主要由两个，视频数据丢包率和网络往返时延。下文具体叙述这获取两种参数的反馈模块的实现。

4.2.1 视频数据丢包率获取的实现

本文统计视频数据丢包率的信息主要在视频传输系统的接收端进行，依据的是视频数据包的包头信息。视频数据包的结构如图 4-1 所示，由图可知视频数据包由包头和视频数据组成，视频数据位于视频数据包尾部，大小固定，一般是 1024 字节。包头大小也固定，是由固定的结构体携带记录该视频数据包的信息，

图 4-1 视频数据帧结构图

包头的结构体信息如下：

```
struct frame_header
{
   int camera_no;                  //节点 ID
   int frame_no;                   //帧 ID
   long slice_no;                  //视频数据包 ID
   int frame_type;                 //帧类型
   int T;                          //最大视频数据包长度
   int K;                          //实际视频数据包长度
   int esi;                        //视频数据包序号
   unsigned char* frame_data;      //视频数据包大小
};
```

结构体中 frame_header 定义了一个视频数据包的包头结构，其中 camera_no 字段表示发送端编号信息，frame_no 字段表示视频数据帧的帧号，slice_no 字段表示视频数据包的编号信息，而 frame_type 用于指明帧类型，规定 I 帧用“1”表示，P 帧用“2”表示。T、K、esi 表示当前视频数据包的基本信息。frame_data 是一个数组指针，指向当前视频数据包的视频数据大小。

根据视频数据包的包头信息可知，slice_no 字段是记录视频数据包编号，一般是从 0 开始。因此在 t 时间段内，每当接收到视频数据包后，会在当前时段的视频数据包数量加 1，同时把接收到的视频数据包的 slice_no 与当前时段的最大 slice_no 比较，若大于当前时段的 slice_no 的最大值，则用该值替换，否则不做处理。直到时间段结束时，根据当前时段 slice_no 的最大值和前一时段 slice_no 的最大值求差值，得出本时段内发送端发送的视频数据包数量，再通过与接收端接收到的视频数据包数求商。最后通过 1 减去所得的商，得出丢包率的值，再通过 socket 发送给发送端。接收端丢包率统计模块的关键代码如下：

```
socket 初始化；
while (1)
{
      recvfrom();//接收视频数据包；
      readPacket();//解析视频数据包；
      SumOfPacket++；//视频数据包自加 1
```

```
    if (frame->slice_no > curSliceNoMax) //替换当前最大的视频数据包编号
        cur_slice_no_max = frame->slice_no;
    if(timeout()) //时间段结束
    {
        //计算视频数据丢包率
        LossOfPacket = 1 - SumOfPacket / (curSliceNoMax - preSliceNoMax) ;
        sendto(); //反馈丢包率给发送端
        //下一时段参数初始化
        preSliceNoMax = curSliceNoMax；//替换时段最大视频数据包编号
        curSliceNoMax = 0；//
        SumOfPacket = 0; //当前视频数据包数量清零
        timebegin(); //计时器重新计时；
}
}
```

发送端接收到反馈数据之后会进行平滑处理。同时，为了预防反馈数据包额丢失导致发送端接收数据阻塞问题，本文利用 select 机制来实现非阻塞的 socket 接收机制。select()函数具体形式如下所示：

```
int select(int maxfdp, fd_set *readfds, fd_set *writefds, fd_set *errorfds, struct timeval *timeout);
```

第一个参数表示最大的文件标识符，之后三个参数分别为可读、可写和异常的文件标识符集合。timeout 参数的作用是设置 select 的超时时间，通过设置该参数不同的值，可以将 select 置于三种不同的状态。第一种是将该参数的值设置为 NULL，就是不传入时间结构，此时 select 处于阻塞状态，只有等到监视的文件描述符集合中某个文件描述符发生变化才进行下一步操作；第二种是将时间值设为 0 秒 0 毫秒传给 timeout 参数，此时 select 成为一个纯粹的非阻塞函数，不管监视的文件描述符是否有变化，都立刻返回继续执行，文件无变化返回 0，有变化返回一个正值；第三种是 timeout 的值设置为大于 0，此时 timeout 的值是等待的超时时间，即 select 在 timeout 值表示的时间内阻塞，如果在超时时间之内有事件到来就返回，而一旦发生超时则立即返回。发送端的视频丢包率获取模块的核心代码如下

```
float getPacketLoss(float factor, float lastPL, char *IP)
{
    初始化 socket；
     select(); //超时设置；
     //判断是否超时
```

```
    if(FD_ISSET()) //未超时
   {
        recvfrom(); //接收视频数据丢包率信息
        curPL= atof(recvbuff); //解析视频数据丢包率信息
    }
    else//超时
        curPL = PACKETLOSS_DEFAULT;//超时一般设为默认值

   //平衡调整，curPL 为当次丢包率，lastPL 为上次丢包率
    curPL = (1 - factor) * lastPL + factor * curPL;
   return curPL;
}
```

从核心代码中知，本文通过 select 机制设置了超时机制，本文中一般超时时间为 20s。如果没有超时则直接读取视频数据丢包率的值，而发生超时时，就把本时段的视频数据丢包率设为默认值。发生超时，说明拥塞程度比较严重，丢包率值较高，所以默认值一般设为 75%。

4.2.2 网络往返时延获取的实现

通过网络往返时延获取的详细设计可知，本文获取网络往返时延的方法是通过 ICMP 请求应答数据报文来获取。因此，第一步要先构建 ICMP 报文，这是个构建过程中需要原始套接字来完成。

套接字（socket）是计算机操作系统提供给应用程序来完成与底层协议进行数据交换的接口，比较常见的有流套接字（SOCK_STREAM）、数据报套接字（SOCK_DGRAM）和原始套接字（SOCK_RAM）[26]。流套接字适用于面向连接、提供可靠数据传输的服务，一般 TCP 协议传输采用流套接字，保证数据的有序，无差错的传输。数据报套接字适用于面向无连接的服务，UDP 协议传输采用这种套接字。数据报套接字实现简单，实时性高，但是不能保证可靠性，一般视频传输就采用这种套接字。

原始套接字与以上两种套接字有本质的区别，在系统核心实现，使用户进程可以直接使用 ICMP 协议构建应用程序，而不必向内核中添加额外编码。创建原始套接字需要 socket 函数，具体形式如下：

```
int socket(int domain, int type, int protocol)
```

其中第一个参数指定应用程序使用的通信协议的协议簇，原始套接字采用 AF_INET；第二个参数指定要创建的套接字类型，原始套接字采用 SOCK_RAW；

第三个参数指定应用程序使用的通信协议，本文中设置为 IPPROTO_ICMP，表明采用 ICMP 协议。ICMP 报文头结构体如下：

```
typedef struct _icmphdr
{
   unsigned char i_type; //8 位类型，报文类型，8 表示请求，0 表示应答
   unsigned char i_code; //8 位代码, 表示报文代码
   unsigned short i_cksum; //16 位校验和,
    unsigned short i_id ; //识别号（一般用进程号作为识别号），用于匹配请求和应答报文
   unsigned short i_seq ; //报文序列号, 用于标记请求报文顺序
   unsigned int timestamp; //时间戳
}ICMP_HEADER;
```

其中依靠 i_id 匹配请求和应答报文，而 timestamp 字段会记录发送 ICMP 报文的时刻，接收到应答报文时则可利用接收时刻与 timestamp 计算差值，求出网络往返时延，同时为了防止偶然性，本文采用获取 5 次网络往返时延求平均值的方法来最终确定网络往返时延的值，关键伪代码如下：

```
// 获取网络往返时延的值
float getRTT(float factor, char *IP, float preRTT)
{
   定义原始套接字;
   地址绑定;
   for(int i = 0; i < 5; i++) //求 5 次网络往返时延的和
   {
      获取发送时刻;
      构建 ICMP 报文;
      发送 ICMP 报文;
      if(超时)
            curRTT += RTT_DEFAULT ;
      else
       curRTT +=  接收时刻  -  ICMP 数据报文内时间戳;
   }
   curRTT = curRTT / 5;//求平均值
   //值平衡处理
   curRTT = (1 - factor) * preRTT + factor * curRTT;
   return curRTT;
```

```
}
```

从伪代码中可以看出，本文为 ICMP 应答报文设置了超时机制。超时时间通过调用 setsockopt()函数实现，具体参数设置如下：

```
struct timeval timeout={3,0};//定义超时时间
//设置超时时间
setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &timeout, sizeof(timeval));
```

当发生超时，会将本次的网络往返时延设为默认值，本文中将该默认值设置为超时时间。而在网络往返时延经过平均求值后会通过平衡因子（factor）与上次的网络往返时延进行值平滑处理，进一步降低由于突发事件带来的网络往返时延波动过大的影响。

4.3 视频传输速率调整模块的实现

在完成视频数据丢包率和网络往返时延这两个参数的获取完成之后，就可以制定视频传输拥塞控制机制的具体实施方案。第一个方面就是视频传输速率的调整控制。

4.3.1 视频发送速率调整的方法

视频发送速率的调整主要可以通过调整发送视频数据帧的速率或者调整视频采集分辨率来实现[27]。

视频采集分辨率是决定视频清晰度的关键因素，分辨率越高，视频越清晰，同时视频采集模块设备所采集的数据量就越大；而分辨率低则视频数据量越小。因此，当视频采集分辨率越高，视频传输系统在单位时间所需要发送的数据量就越大，视频发送速率就越高。也就是说，通过调整视频采集分辨率的大小就可以调整视频发送速率。但是，视频分辨率的调整比较麻烦，因为视频采集分辨率参数是视频传输系统启动时，就设置完成，要想重新设置，只能是人为的重启系统重新设置，而且视频采集分辨率往往和视频编解码相关联，因此要调整视频采集分辨率则需要同时调整视频采集系统发收两端的编解码参数设置。

发送视频数据帧的速率调整则相对简单一些，只需要控制发送视频数据帧的时间间隔就可以。通过 TFRC 模型计算公式可知，本文计算出来的是每秒的视频数据包的数量，而不是每秒发送视频数据帧的数量。而为了考虑视频在接收端解码播放的流畅性，一般的视频传输系统都是发送完整个视频数据帧之后再完成时间间隔。因此必须要对本文的视频数据帧进行数据包的统计。

通过之前的介绍可知，本文所述的视频传输系统是通过 H.264 技术来完成视频的编解码。H.264 是国际标准化组织(ISO)和国际电信联盟(ITU)共同提出的新

一代数字视频压缩格式[28]。通过对一小段时间内图像的统计结果可知，在相邻的几幅图像画面中，有差别的像素一般只有 10%，亮度差别的变化也不超过 2%，而色度差值的变化更是在 1%以内。H.264 编码的原理就是对于这一段连续变化不大的图像画面，首先编码出一个完整的图像帧，而之后的图像帧则不会编码全部图像内容，而只写入和该完整图像帧的差别数据。经这样处理之后，后面的图像帧的数据大小只有完整帧的 1/10 或者更小。然后重复以上过程，直到当某个图像帧与之前的图像变化很大，无法参考前面的帧来生成，就结束这段图像帧的编码，开始下一段图像编码。根据该编码原理，H.264 编码定义了三种帧，完整图像数据编码的帧叫 I 帧，它包含图像全部独立的信息，对 I 帧进行单独解码后的内容就可以直接播放；参考之前的 I 帧生成的帧叫 P 帧，P 帧只包含差异部分图像数据的编码，所占用的空间更小，方便传输；还有一种参考前后的帧编码的帧叫 B 帧，B 帧的解码不仅要取得之前面数据帧的画面，还要取得后面数据帧的画面，通过前后图像画面与本帧数据的叠加形成最终的图像画面，这种帧的压缩率最高，但是解码时对 CPU 耗费极大，因此 B 帧一般不使用。

通过上面的分析知 H.264 的数据帧一般可分为 I 帧和 P 帧（B 帧不使用）。假设经过统计，平均每个 I 帧是分为 i 个视频数据包发送，平均每个 P 帧是分为 p 个视频数据包发送，而平均每隔 k 个 P 帧就有一个 I 帧，则可以得出公式 4-1。

$$v = \frac{T_s \times (k+1)}{i+p \times k} \tag{4-1}$$

其中T_s由表示 TFRC 模型公式计算所得的每秒发送的数据包，v 表示每秒发送的视频数据帧数量。假设程序发送视频数据包时间忽略不计，则可求出发送视频数据帧的间隔时间，如公式 4-2 所示。

$$s_w = \frac{1}{v} \tag{4-2}$$

其中s_w表示发送视频数据帧的时间间隔，单位是毫秒（ms）。通过以上分析，可以通过调节s_w的大小来控制视频发送数据帧的速率，进而控制视频发送速率。这种方法实现简单，对速率控制直接有效，不需要重新设置系统其他参数，但是需要对视频数据帧有一定的统计。本文采用的就是这种方法来时实现视频发送速率的控制。

4.3.2 视频速率调整的三个机制的实现

在本文的详细设计中可知，速率调整根据参数不同有三个调整机制，分别为 TFRC 模型机制、速率微调机制和速率最佳机制[29]。

TFRC 模型机制主要是根据 TCP 吞吐量公式来计算视频发送数据包速率，而本文为了方便其他两个机制的调整，会根据公式 4-1 和公式 4-2 将视频数据包的

发送速率转换为视频帧的发送间隔（wait_ms），关键代码如下。

```
//根据 TFRC 模型获取视频发送速率（视频发送间隔（wait_ms））
//参数 R 为网络往返时延，参数 P 为视频数据丢包率
int buildTCPfriendlyModel(float R, float P)
{
   //利用 TCP 吞吐量公式计算视频包发送速率
   int rate = TFRC(R,P);
   //将视频数据包的发送速率转换为视频帧的发送间隔
   int wait_ms = RateToTime(rate);
   return wait_ms;
}
```

速率微调机制是在拥塞程度不是特别严重的情况下，对视频发送速率稍微降低，以降低网络拥塞程度。速率微调机制的进行需要通过之前的速率作为依据，关键代码如下。

```
//速率微调机制下的速率调整
int littleChange(int wait_ms)
{
      wait_ms = wait_ms + 1;//发送视频数据帧间隔增加 1ms
      return wait_ms;
}
```

速率最佳机制是指在网络几乎没有出现拥塞现象的情况下，保持视频发送速率不变。此机制速率调整十分简单，只需保持速率不变，关键代码如下。

```
//速率最佳机制下的速率调整
int stay(int wait_ms)
{
      return wait_ms;//保持视频发送速率不变
}
```

4.3.3 视频速率调整机制的实现

本文所述的视频速率调整机制是根据视频视频丢包率和网络往返时延来控制视频发送速率在 TFRC 模型机制、速率微调机制和速率最佳机制之间进行转换。另外，为了防止参数变化缓慢而导致的拥塞调整不及时，本文还设置了定时更新 TFRC 模型的机制，具体流程如图 4-2 所示。

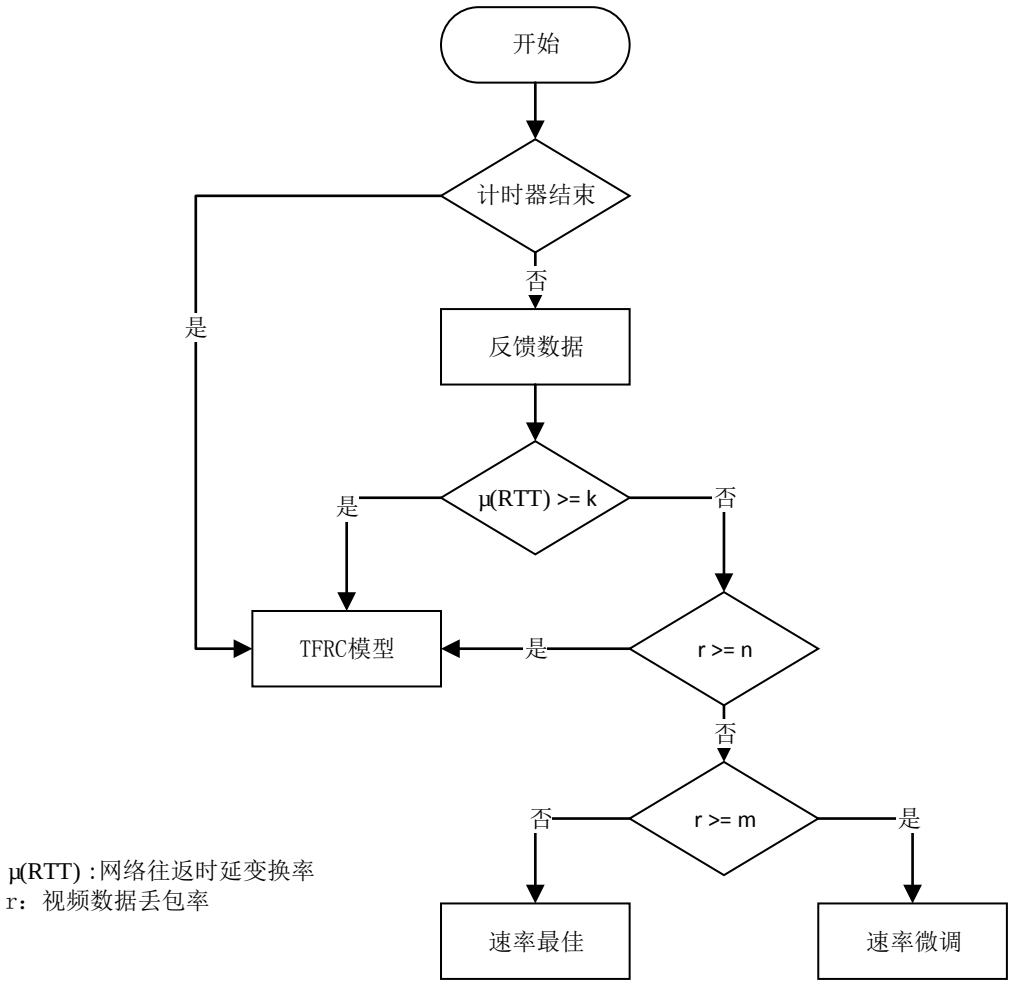


图 4-2 视频发送速率调整机制流程图

从图 4-2 中可知，本文先会判断计时器是否结束，如果结束就直接利用 TFRC 模型更新视频发送速率；如果计时器未结束，就会根据参数的情况制定相应的速率调整策略，具体调整情况在速率调整设计中已详述，在此不做复述。

另外本文为了在获取参数制定网络拥塞控制机制的同时，不影响视频的传输，采用了多线程的来实现。线程（Thread）是进程中的多个执行线路的一个，每个线程都有自己独立的局部变量，但是与该线程所属的进程的所有线程共享全局变量，文件句柄和信号等数据。Linux 系统所采用的 POSIX 线程标准。

Linux 系统下线程间通信方式有很多种，包括共享内存、套接字，消息队列、管道和信号量等[30]。套接字通信方式与网络套接字类似，也是基于 TCP/IP 协议，在线程之间进行数据交互，由于是在本机之间进行，通常采用 UNIX 域套接字。消息队列是采用一个链表来管理线程间的通信内容。管道是一种数据结构，像一个序列化文件一样访问，形成了两个进程间的一种通信渠道。

共享内存是在进程内部映射一段能被所有线程访问的内存空间。共享内存是在程序开始时，由主线程创建，可以被其它线程访问。共享内存是针对其它通信方式效率低、开销大而设计的，也是最简单也是最快的线程间通信方式。通常共

享内存在程序中采用全局变量的方式，配合信号量机制一同使用，以实现多线程间的通信和同步。本文中视频发送速率调整模块将调整信息反馈至视频发送模块就采用共享内存的方式。

采用共享内存进行线程间数据交互时，还需处理数据同步的问题。Linux 线程间同步方式主要有锁机制、信号量机制等，本文选用锁机制作为保护共享内存的同步机制。锁机制包括互斥锁、条件变量、读写锁等机制。其中互斥锁使用排他方式防止数据被多个线程并发修改；条件变量则是以原子操作的方式阻塞某个进程，直到特定条件被满足前，不会重启线程。通常特定条件都是在有互斥锁保护的条件下进行的，所以使用条件变量时必须带有互斥锁；读写锁用来管理临界资源，限制写操作，允许读操作。

Linux 提供了 pthread_mutex_lock( )和 pthread_mutex_unlock( )函数组来实现线程同步的锁机制。pthread_mutex_lock( )用来为互斥锁上锁，接受互斥对象的指针并将其锁定，如果在这之前互斥锁已经被锁定，则调用者将会被转入睡眠状态。函数成功返回，调用者被唤醒。pthread_mutex_lock( )函数成功返回“0”，否则返回相应的错误代码。pthread_mutex_unlock( )执行与前者相反的操作，打开互斥锁，释放临界资源。根据以上分析和视频发送速率调整流程图，本文得到如下关键代码。

```
while(1)
{
    if(first) //视频程序启动，第一次获取参数
    {
        curRTT = getRTT(ip); //第一次获取视频数据丢包率
        RTT = curRTT;
        PacketLoss = getPacketLoss(1, 0, ip); //第一次获取网络往返时延
    }
    else
    {
        preRTT = curRTT; //记录上一次网络往返时延
        curRTT = getRTT(ip); //获取下次网络往返时延
       //获取网络往返时延变化率
        RateOfRTT = fabs(preRTT - curRTT) / curRTT;

        prePacketLoss = PacketLoss; //记录上一次视频数据丢包率
       //获取下一次视频数据丢包率
```

```
        PacketLoss = getPacketLoss(factor, PacketLoss, ip);
    }

    pthread_mutex_lock(&work_mutex); //数据同步通信，上互斥锁
    if(9 == timer) //计时器结束，定时更新 TFRC 模型
    {
        wait_ms = buildTCPfriendlyModel(RTT, PacketLoss);
        timer = 0; //重新计时
    }
    else
    {
        if(first) //第一次视频发送速率调整，引用 TRFC 模型调整
        {
            wait_ms = buildTCPfriendlyModel(RTT, PacketLoss);
            first = false;
        }
        else
        {
        //网络往返时延变化率超过 k，引用 TFRC 模型进行速率调整
            if(RateOfRTT >= k)
                wait_ms = buildTCPfriendlyModel(RTT, PacketLoss);
            else //网络往返时延变化率低于 k
            {
              //视频数据丢包高于 n，引用 TRFC 模型调整
                if(PacketLoss >= n)
                    wait_ms = buildTCPfriendlyModel(RTT, PacketLoss);
                else //视频数据丢包低于 n
                {
                //视频数据丢包高于 m，引用速率微调状态模型调整
                    if(PacketLoss >= m)
                        wait_ms = littleChange(wait_ms);
                    else //视频数据丢包低于 m，引用速率最佳状态调整
                        wait_ms = stay(wait_ms);
                }
```

```
                }
            }
        }
        pthread_mutex_unlock( &work_mutex); //释放互斥锁
        sleep(60); //每隔 60 秒进行一次视频发送速率的调整
        timer++; //计时器自加
}
```

从上面的关键代码中知，本文不仅针对视频传输系统的初始化阶段进行的特殊处理（直接引用 TFRC 模型对视频发送速率进行赋值），还启用了定时更新 TFRC 速率模型的机制，更好的实现对视频发送速率的调整控制，同时采用多线程同步通信机制，使程序在制定拥塞调整机制的同时，不影响视频的采集传输。

4.4 视频发送缓存调整模块的实现

本文所述的视频传输拥塞控制机制，除去最基本的视频发送速率控制，还增加了视频发送缓存控制作为辅助机制，二者共同作用一起完成拥塞控制的目标。

4.4.1 视频发送缓存调整的方法

视频发送缓存调整主要包含两个方面，第一方面是 UDP 发送缓冲区的调整，另一个方面是视频传输跳帧数的调整。

本文中视频发送是通过 UDP 来进行传输，具体的实现是通过套接字（socket）来完成。因此 UDP 发送缓冲区的调整主要套接字的发送缓冲区的调整，主要通过 setsockopt()函数实现，该函数的具体形式如下

```
int setsockopt(int sockfd, int level, int optname, const char* optval, int optlen);
```

其中 sockfd 表示一个打开的套接字文件描述符；level 表示指定选项代码的类型，有 SOL_SOCKET（基本套接口）、IPPROTO_IP（IPv4 套接口）、IPPROTO_IPV6（IPv6 套接口）和 IPPROTO_TCP（TCP 套接口）等可选参数；optname 表示选项名称，有 SO_SNDBUF（发送缓冲区）、SO_RCVTIMEO（接收超时）等可选参数；optval 是一个指向变量的指针类型，可以指向整型，字符数组型等；optlen 是表示 optval 大小的整型变量。根据本文之前的详细设计可以得出 UDP 发送缓冲区调整机制的关键代码如下。

```
//UDP 发送缓冲区调整
//flag 是调整辨别因子，为 0 时表示减小缓冲区，为 1 时增大缓冲区
void UDPBufferChange(int flag)
{
```

```
    if(1 == flag)//增大缓冲区
    {
        //当小于 UDP 发送缓冲区的最大值，则增大缓冲区
        if(sndbuf < yuv_frame_buf_size*8)
            sndbuf += yuv_frame_buf_size;
    }
    else //减小缓冲区
    {
        //当大于 UDP 发送缓冲区的最小值，则减少缓冲区
        if(sndbuf > yuv_frame_buf_size*2)
            bufsize_addr -= yuv_frame_buf_size;
    }
    buflen = sizeof(sndbuf);
    //UDP 缓冲区调整
    setsockopt(sockfd, SOL_SOCKET, SO_SNDBUF, (void *)&sndbuf, buflen);
}
```

从上面代码可知，本文中 yuv_frame_buf_size 表示的是视频采集设备采集一帧原始数据帧的大小，故将它作为 UDP 缓冲区调整的增量单位。另外本文还设置发送缓冲区的最大值和最小值，防止 UDP 缓存调整越界。

而另一个方面视频传输跳帧数的调整主要是通过减少视频传输帧的频率来控制发送视频数据量，以此来控制发送缓存，关键代码如下。

```
//视频传输跳帧数调整
//flag 是调整辨别因子，为 0 时表示减小跳帧数，为 1 时增大跳帧数
void SndRateChange(int flag)
{
    if(1 == flag)//增大跳帧数
    {
        //当小于规定跳帧数的最大值，则增大跳帧数
        if(send_rate< send_rate_max)
            ++send_rate;
    }
    else //减小跳帧数
    {
        //当大于跳帧数的最小值，则减少跳帧数
```

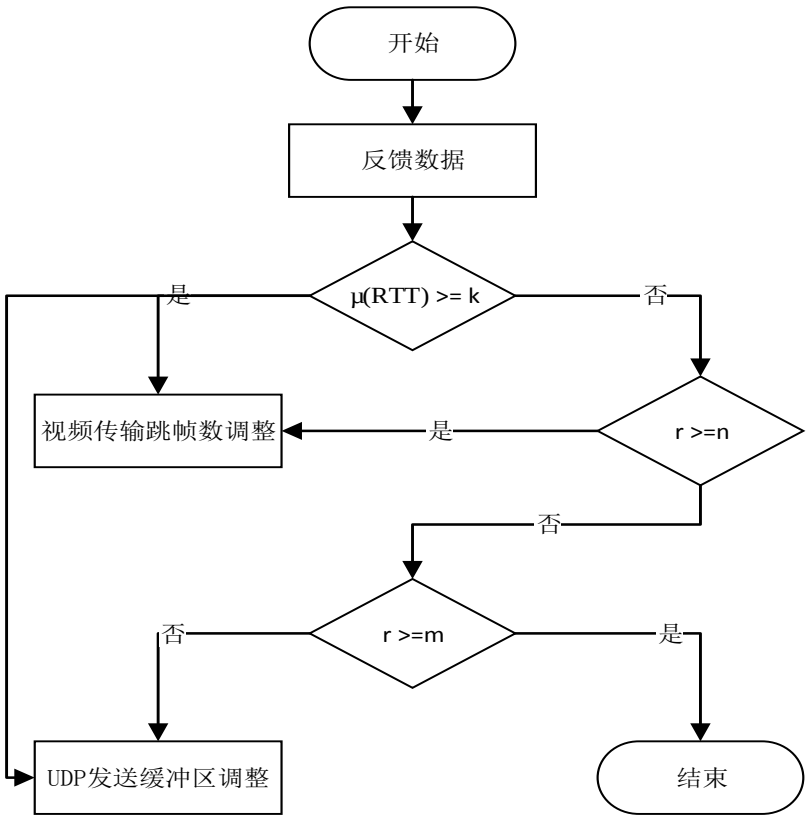
```
        if(send_rate > send_rate_min)
            --send_rate;
    }
}
```

通过上面的代码可知，本文分别设置的视频传输跳帧数的最大值和最小值，一方面是为了防止数据发生越界的现象，另一方面也是从跳帧数本身考虑。如果跳帧数过大，则会导致视频展示卡顿严重，画面不连续，而过小（如跳帧数为 0 或 1）则视频传输系统会传输一些没必要的视频数据。而本文通过设置了跳帧数的最大值和最小值解决这个问题。

4.4.2 视频发送缓存调整的实现

本文是根据网络往返时延和视频数据丢包率来实现对视频网络的拥塞程度判别。根据视频发送缓存模块详细设计的状态转移图，可以得到如下的视频发送缓存调整机制流程图，如图 4-3 所示。

从图中可知当网络往返时延变化率超过 k 时，会对视频传输跳帧数和 UDP 发送缓冲区两方面进行调整；而当网络往返时延变化率低于 k，会根据视频丢包率的情况进行调整，即当视频数据丢包率大于 n，调整视频传输跳帧数，视频数据丢包率小于 m，调整 UDP 发送缓冲区。另外，与视频发送速率调整机制类似，为了在获取参数制定视频发送缓存调整机制的同时，不影响视频的传输，采用了多线程的来实现。根据以上分析，可以得出以下关键代码。



μ(RTT) :网络往返时延变换率
r ： 视频数据丢包率

图 4-3 视频发送缓存调整机制流程图

```
while(1)
{
    preRTT = curRTT; //记录上一次网络往返时延
    curRTT = getRTT(ip); //获取本次网络往返时延
    //获取网络往返时延变化率
    RateOfRTT = fabs(preRTT - curRTT) / curRTT;

    prePacketLoss = PacketLoss; //记录上一次视频数据丢包率
    //获取本次视频数据丢包率
    PacketLoss = getPacketLoss(factor, PacketLoss, ip);
    if(preRTT > curRTT) //判断网络往返时延是增大还是变小
        flag = 0;
    else
        flag = 1;
```

```
    pthread_mutex_lock(&work_mutex); //数据同步通信，上互斥锁
   //网络往返时延变化率超过 k，在两方面进行缓存调整
    if(RateOfRTT >= k)
   {
     UDPBufferChange(flag);
     SndRateChange(flag);
   }
    else //网络往返时延变化率低于 k
    {
        //视频数据丢包高于 n，调整视频传输跳帧数
         if(PacketLoss >= n)
              SndRateChange(1);
         else
            {
             //视频数据丢包低于 m，调整 UDP 发送缓冲区
                 if(PacketLoss < m)
                     UDPBufferChange(0);
              }
    }
    pthread_mutex_unlock( &work_mutex); //释放互斥锁
    sleep(180); //每隔 180 秒进行一次视频发送缓存的调整
}
```

从上面的关键代码中知，本文通过判断网络往返时延是增大还是缩小和视频丢包率的大小，来确定视频传输跳帧数和 UDP 发送缓冲区是增大还是缩小。同时视频发送缓存调整不宜像视频发送速率调整这么频繁，所以本文中视频发送缓存的调整频率要低于视频发送速率调整的频率，设置为 180s。

4.5 本章小结

本章首先介绍了本文的开发环境，接着对网络参数反馈模块进行了具体的实现描述，分别介绍了视频数据丢包率和网络往返时延的具体实现方法。在此基础上，本章对视频发送速率控制模块和视频发送缓存控制模块的实现的进行描述。本章都是先把调整的具体方法实现描述完整，之后先通过流程图描述整个控制模块的具体实现流程，再通过关键代码的进一步描述整个拥塞控制机制的实现过程。

第五章　视频传输拥塞控制机制的测试与分析

5.1 测试环境的简介

本文在第四章中实现了具有本文所述的拥塞控制机制的视频传输系统，用于验证评估该拥塞控制机制的可行性和控制效果。该视频传输系统的发送端运行于无线多模网关的视频采集开发板上，接收端则运行于 PC 机上。

5.1.1 M-Link 无线多模网关简介

本文所述的无线多模网关是 M-Link 无线多模通信网关。M-Link 无线多模通信网关是北京邮电大学计算机学院物联网技术中心自主研发的一种具有多种网络接入的网络设备。与传统的网关相比，M-Link 无线多模通信网关能进行多种网络的接入或者进行多种网络协议的转换。图 5-1 是 M-Link 无线多模通信网关的外观图。

图 5-1 M-Link 无线多模网关外观图

从图 5-1 中可以看到，M-Link 无线多模通信网关不仅有多个终端设备的网络接口，还有多个无线网络的接口。主要的有以下几种无线网络组成。

主要的有以下几种无线网络组成。

(1)卫星通信网络[31]。卫星通信网络是指利用人造地球通信卫星作为中转站转发无线电信号，以完成地面上多个卫星基站的通信网络。目前应用的最为广泛的卫星通信系统有铱星(Iridium)系统和中国的北斗卫星通信系统，在 M-Link 无线多模通信网关中，这两种卫星系统的通信模块都有内置，是无线多模网关连接 Internet 的接口。

(2)3G 通信网络。3G 是第三代移动通信技术，是能够支持高速数据传输的蜂窝移动通信技术[32]。目前主流的 3G 标准有 WCDMA、CDMA2000 和 TD-SCDMA。3G 也是无线多模网关连接 Internet 的接口。

(3)Wi-Fi 网络。Wi-Fi 是一种将移动设备（如笔记本电脑、手机）的通信终端以无线方式相连接的技术，是一种高频的无线电信号，是 3G 通信网络和有线网络非常好的补充。除此之外，还有 mesh 无线网和 340MHz 无线局域网等无线网络，它们主要用于无线多模网关之间的数据交换。

此外，本文所述的视频传输系统发送端运行于 M-Link 无线多模通信网关的视频采集 ARM 开发板上。该开发板使用 32 位 ARM11 S3C6410 CPU 800Mhz，扩展总线最大频率 133MHz，256MB DDR 内存；采用 1GB NANDFLASH，其中 256K NORFLASH；使用 OV3640 数字摄像头作为视频采集设备；支持无线网络和有线网络。接收端运行于 PC 机上。该 PC 机使用基于 32 位 Intel Core2 Duo E7500 处理器的 PC 机，主频为 2.93Ghz；内存为 1.96G DDR2；采用 Windows 7 操作系统。

5.1.2 测试环境网络部署

同时本文专门为测试搭建了具有多个无线多模网关的网络环境，该网络的网络部署图如图 5-2 所示。

在图 5-2 中，节点 A、B、C、D、E 分别表示无线多模网关，它们之间可以通过 Wi-Fi、无线 mesh 网或者 340MHz 无线网等进行网络连通。其中节点 A、B、C 不仅有与其他无线多模网关互通的无线网络，还有 3G 网络，可直接与 Internet 连通。而节点 D、E 没有 3G 网络，只有与其他无线多模网关连通的无线网络，不能直接与互联网连通。另外，在节点 D、E 上分别有视频采集开发板，测试的视频传输系统就运行在该开发板上。在另一端是直接与 Internet 连接的视频接收端，运行于 PC 上。

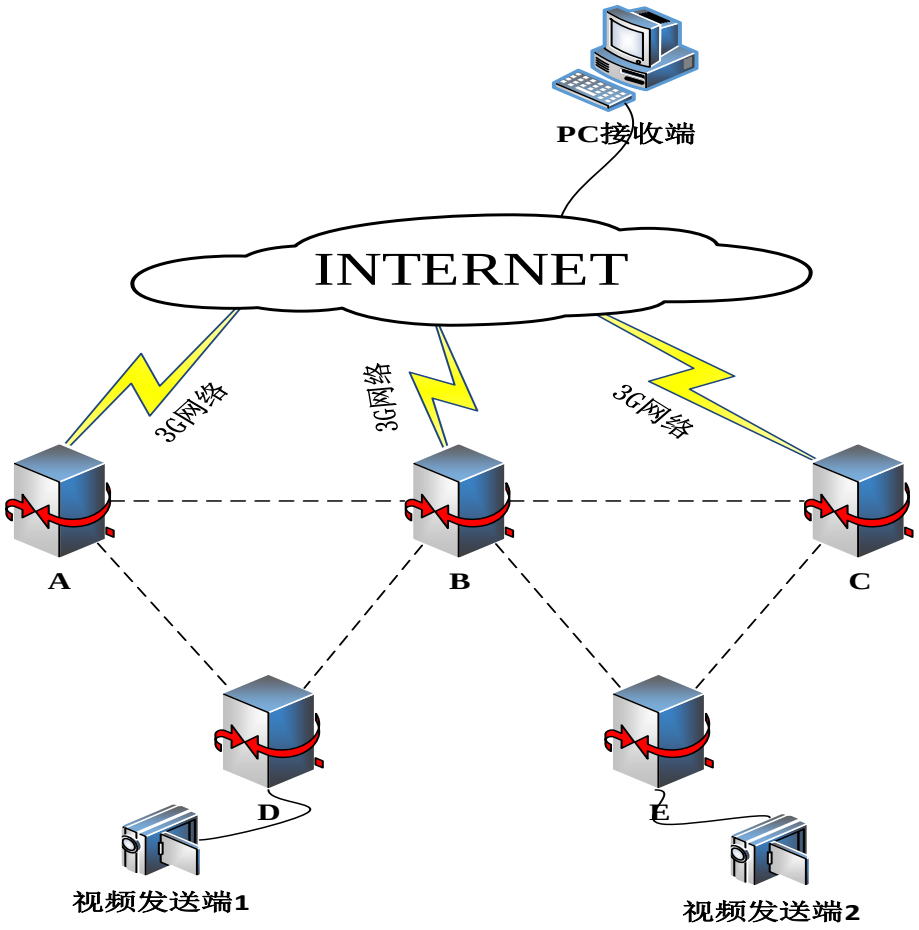


图 5-2 测试环境网络部署图

5.2 拥塞控制机制测试

本文的视频传输拥塞控制机制的测试是基于如图 5-2 所示的网络环境，视频发送端的效果展示图如下图 5-3 所示。

其中 wait_ms 表示视频帧发送间隔时间，单位是 ms（毫秒），用来代替表示视频发送速率；而 sndbuf 则表示 UDP 发送缓冲区，以 framesize（视频数据帧的大小）为单位；send_rate 表示视频传输跳帧数，单位是帧。

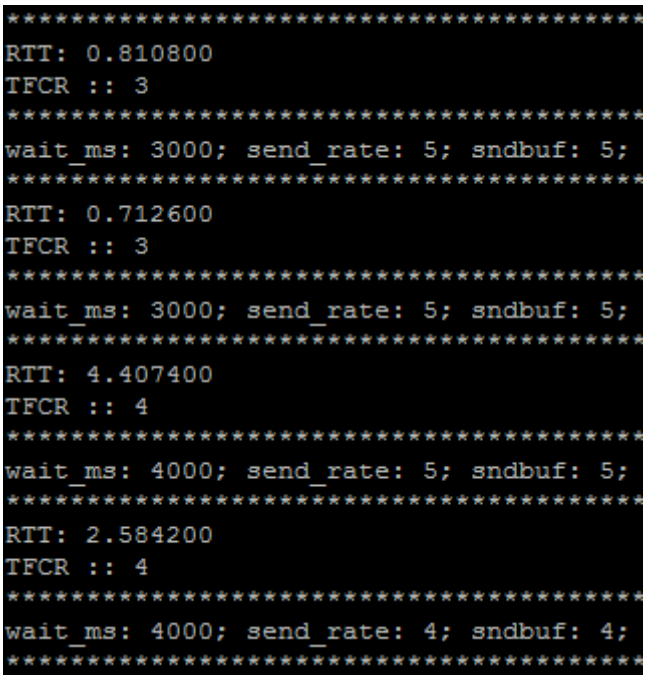


图 5-3 发送端效果展示图

接收端的效果展示图如图 5-4 所示。

图 5-4 接收端效果展示图

从图 5-4 中可以直接看到丢包率的统计信息。

本文在测试环境内的节点 D 上先后运行了传统的视频传输系统和具有本文所述的拥塞控制机制的视频传输系统，然后每隔 1min 统计了两个系统的视频帧间隔、UDP 缓冲区、视频跳帧数和丢包率信息。另外，本文为了测试，专门制造了拥塞现象，将视频发送速率值增大（发送帧间隔为 3ms）。根据测试结果，把发送端的视频帧发送间隔、UDP 发送缓冲区、视频传输跳帧数与接收端的丢

包率分别为 Y 轴，时间为 X 轴，得出以下四幅数据对比图。

从图 5-5 中看出，在传统视频传输系统中，由于没有拥塞控制机制，所以视频发送速率会保持不变，如图 5-5 中所示视频帧间隔一直保持在 3 ms。而在具有拥塞控制机制的视频传输系统中，在 0~3 min 时，如图 5-5 中所示视频帧间隔迅速从 3 ms 增大至 12 ms，说明视频传输系统检测到拥塞状况，建立 TFRC 模型，重新调整了视频发送速率；而在 3~6 min 内，视频发送速率从 12 ms 增加至 16 ms，增涨缓慢，说明视频速率控制模块进入速率微调状态；而在之后的 4~8 min 内，视频数据发送速率保持不变，说明视频发送速率控制进入速率最佳状态，视频发送速率此时最好。

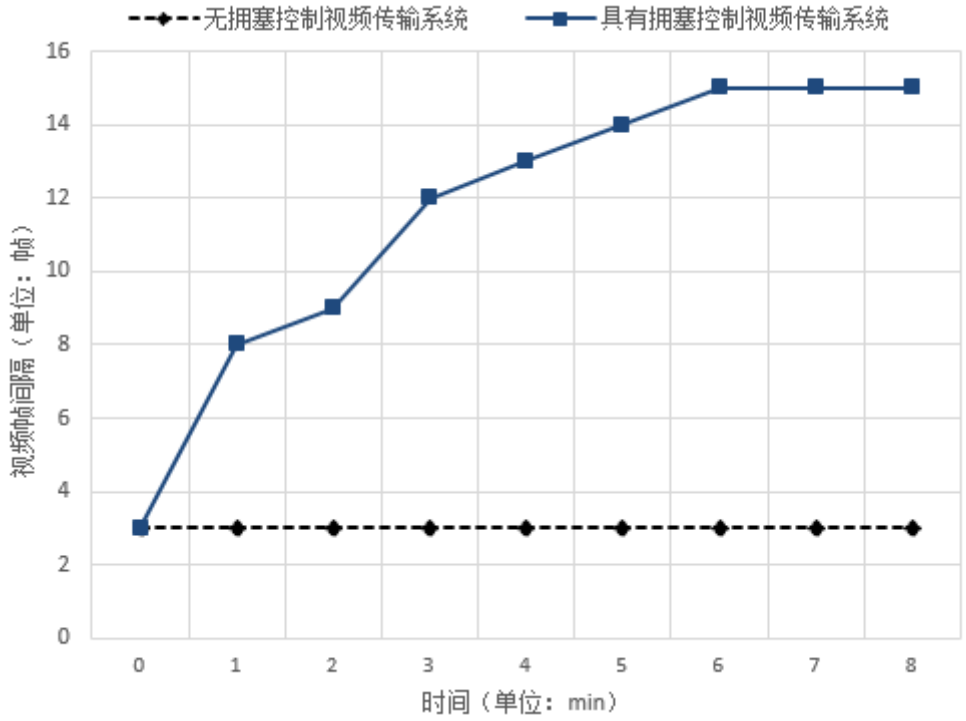


图 5-5 视频帧间隔对比图

从图 5-6 中看出，在传统视频传输系统中，同样由于没有拥塞控制机制，所以 UDP 缓冲区会保持不变，如图 5-6 中所示 UDP 缓冲区一直保持在 2 framesize。而在具有拥塞控制机制的视频传输系统中，在 0~3 min 时，如图 5-6 中所示 UDP 缓冲区迅速从 2 framesize 增大至 6 framesize，说明视频传输系统检测到拥塞状况，会迅速增加 UDP 缓冲区来缓解拥塞程度；而在 3~6 min 内，UDP 缓冲区保持不变，即使拥塞状态还存在，但是 UDP 缓冲区已经增加最大值；而在之后的 6~8 min 内，UDP 缓冲区减少至 4 framesize，说明拥塞程度降低，此时减少 UDP 缓冲区是为了减少内存的浪费。

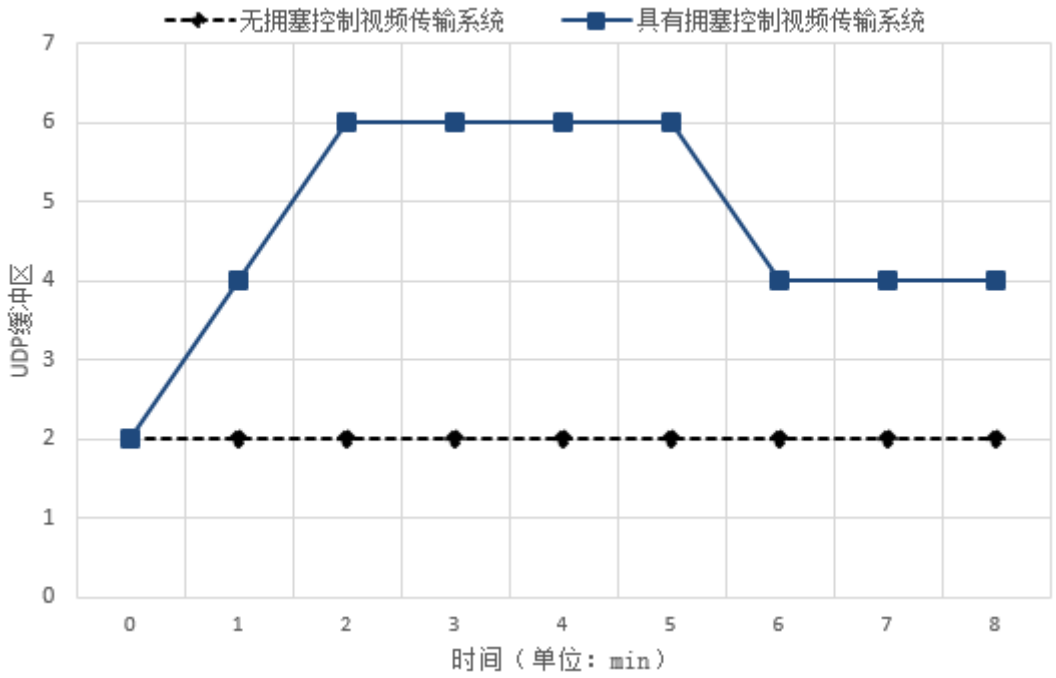


图 5-6 UDP 缓冲区对比图

从图 5-7 中看出，在传统视频传输系统中，由于没有拥塞控制机制，所以视频跳帧数会保持不变，如图 5-7 中所示视频跳帧数一直保持在 1 帧。而在具有拥塞控制机制的视频传输系统中，在 0~4 min 时，如图 5-7 中所示视频跳帧数迅速从 1 帧增大至 4 帧，说明视频传输系统检测到拥塞状况，拥塞控制机制为了减轻拥塞程度会对视频跳帧数进行调整；而在 4~8 min 内，视频跳帧数缓慢从 4 帧减少至 2 帧，说明网络拥塞程度慢慢减轻，拥塞控制机制为了保证视频的展示效果稍微降低的视频跳帧数，增加了视频发送数据量。

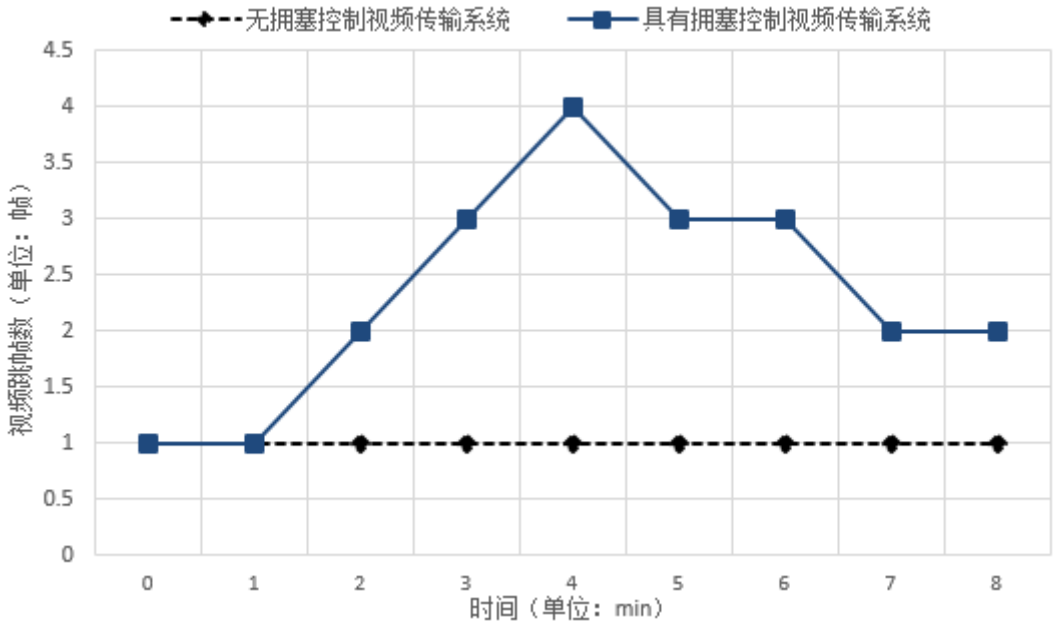


图 5-7 视频跳帧数对比图

从图 5-8 中看出，在传统视频传输系统中，由于没有拥塞控制机制，而且为了测试，本文专门加快了视频发送速率，所以视频丢包率会迅速增加，同时由于拥塞程度一直没有减弱，所以丢包率一直保持在较高的情况，如图 5-5 中所示视频数据丢包率一直保持在 55%左右。而在具有拥塞控制机制的视频传输系统中，在 0~3 min 时，如图 5-8 中 0~3min 内丢包率从 0 增大至 45%，这是由于视频传输系统虽然检测到拥塞状况，但是拥塞控制机制还未完全起作用，所以丢包率与传统视频传输系统中的一样，迅速增加；而在 3~6 min 内，经过拥塞控制机制的调整，网络拥塞程度明显减轻，丢包率从 45%降低至 21%，说明拥塞控制机制效果显著；而在之后的 6~8 min 内，丢包率进一步缓慢降低，保持在 15%左右，说明网络拥塞现象已经得到有效的控制。

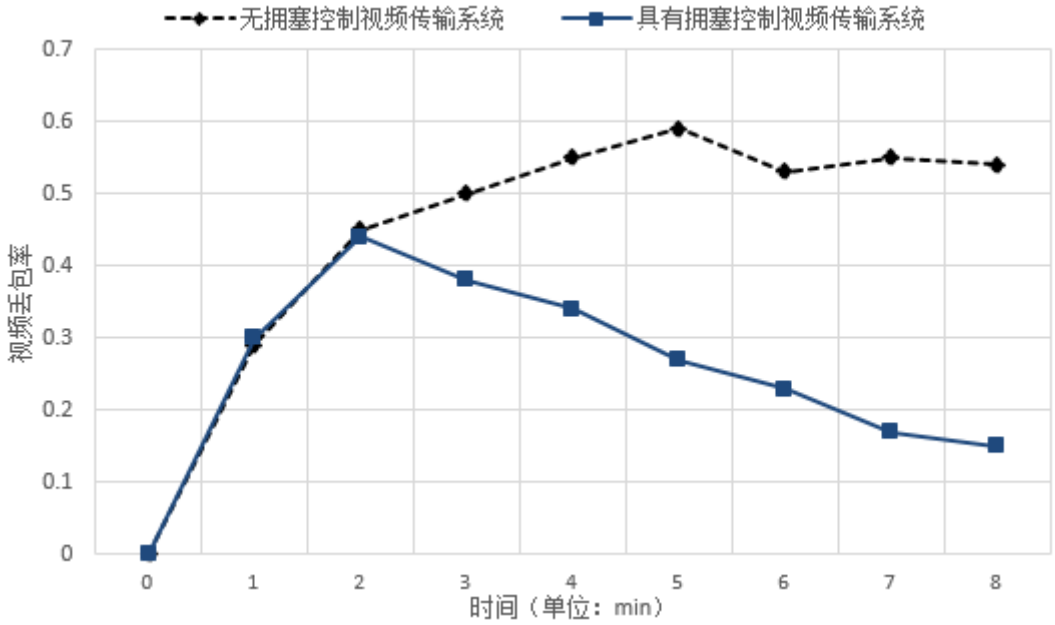


图 5-8 视频丢包率对比图

5.3 网络带宽公平性测试

本文所述的拥塞控制机制不仅实现了基本的对视频传输的拥塞控制功能，同时还保证了 TCP 连接和 UDP 连接公平共享带宽。

本文在节点 E 上同时运行了多个基于 TCP 连接的数据发送应用程序和具有本文中拥塞控制的视频传输系统，并记录统计这两种连接的程序的数据发送量，统计情况如图 5-9 所示。

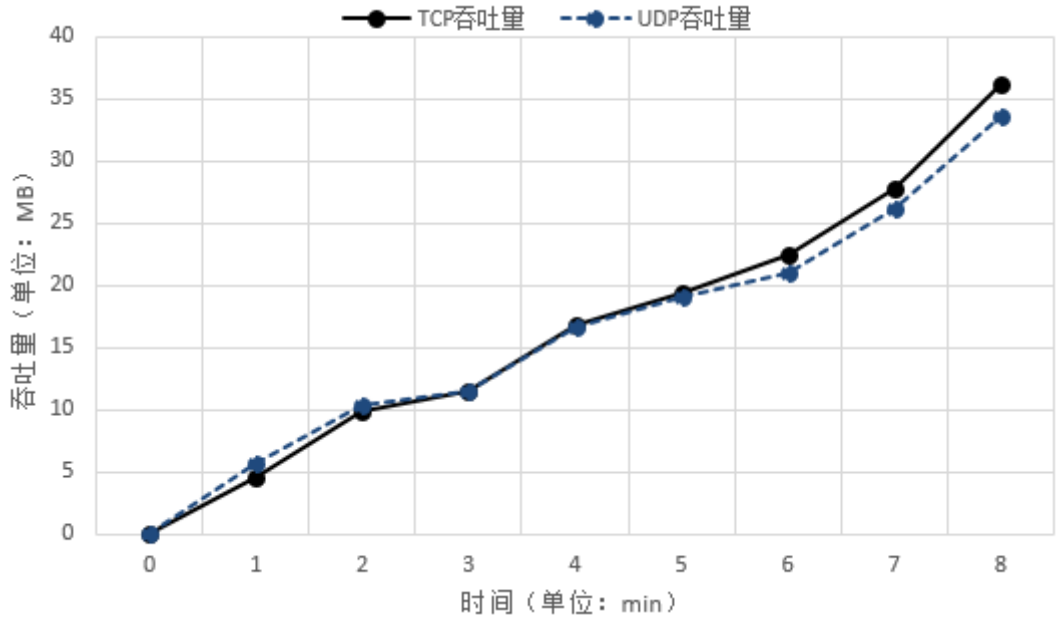


图 5-9 数据量统计图

从图 5-9 中看出，在 0~2min 时，视频传输系统的数据发送量超过了 TCP 的数据发送量，这是由于此时拥塞控制机制还未达到最佳状态；而在 3~4min 内，视频传输系统的数据发送量与 TCP 连接的数据发送量持平，说明拥塞控制机制已开始有效工作；之后在 4~8min 内，视频数据发送量一直保持与 TCP 连接的数据发送量相差不多。以上测试数据说明了该视频传输系统具有 TCP 友好性，保证了与 TCP 的共享网络带宽的公平性。

至此本文的测试工作全部完成。

5.4 本章小结

本章先对测试环境进行了简要描述，接着对视频传输拥塞控制机制的控制效果和网络带宽公平性两个方面进行了测试，测试结果表明本文所述的拥塞控制机制是有效的。

第六章 结束语

6.1 论文工作总结

本文首先对无线多模网关和视频传输的发展背景和研究现状进行了概述，说明了拥塞控制在无线多模网关中视频传输中的重要性，并借此提出研究无线多模网关中视频传输拥塞控制机制的目标。接着本文对拥塞现象及其产生的原因，有线网络和无线网络下的经典拥塞控制算法进行了重点研究，为制定本文所述的拥塞控制机制奠定了理论基础。

其次，通过对基于无线多模网关的网络结构特征和视频传输系统的研究，本文给出了无线多模网关中视频传输拥塞控制机制的设计方案。该方案基于网络往返时延和视频数据丢包率的网络参数对无线多模网关的拥塞程度进行判断，并从视频发送速率控制和视频发送缓存控制两方面来完成拥塞控制功能。在视频发送速率控制方面，本文比较了多种速率控制算法，最后选用 TFRC 算法作为基本的速率控制算法，并针对该算法的不足，提出了速率微调和速率最佳的思想。另一方面，与传统的拥塞控制算法不同，本文还对视频发送缓存进行了调整控制。针对视频传输的特点，从 UDP 发送缓冲区和视频传输跳帧数两方面同时完成对视频发送缓存的控制。

最后，本文在基于 ARM 平台和 Linux 系统的环境下开发实现了具有本文所述的拥塞控制机制的视频传输系统，用于验证评估该拥塞控制机制的可行性和控制效果。同时，在基于 M-Link 无线多模网关的网络环境下进行了测试和拥塞控制效果的评估，实验结果表明，本文研究的拥塞控制机制在视频传输系统的拥塞控制中取得了较好的效果，能在适应该网络结构变换，并在多模网络结构稳定的基础上，保持网络往返时延稳定，降低视频数据丢包率，为视频传输拥塞控制的进一步研究奠定了基础。

6.2 改进建议

本文完成了无线多模网关中视频传输拥塞控制机制的设计和实现，并对该机制进行了测试和评估。但是由于现有设备条件的限制，设计依然存在不足之处需进一步改进。

首先，在对视频数据丢包率统计的设计上，本文采用基于视频数据包的包头信息来完成，虽然降低了无线网络中误码对丢包率的影响，但是并未完全去除。

而在无线网络中误码对丢包的影响巨大，因此在后续的工作中，希望对丢包率统计进行改进，进一步降低误码对丢包统计的影响。

其次本文的视频发送速率调整机制是采用了基于 TCP 友好的 TFRC 算法作为速率调整的基本，并针对其不足提出了改进的方案。改进的速率控制机制不仅能保证 TCP 连接和 UDP 连接的公平性，还能很好的控制网络拥塞状态。但是在不同的环境下，网络拥塞判别参考不同，因此在实际应用中需要对根据参数判定网络拥塞程度做出适当的调整。

最后视频发送缓存调整机制在本文中是作为一个试探性的控制策略，并未像视频发送速率一样进行高频率调整。因此在后续工作中，希望进一步研究视频发送缓存的调整方法，以进一步提高对无线多模网关中视频传输拥塞控制的效果。

参考文献

[1]肖甫，王汝传，孙力娟，王华顺. 基于 TCP 友好的无线网络拥塞控制机制研究[J]. 计算机科学, 2010, 07: 50-53.

[2]王海涛，付鹰. 异构网络融合——研究发展现状及存在的问题[J]. 数据通信, 2012, 02: 18-21.

[3]王海涛. 应急通信网络中的异构网络互联问题及实现机制研究[J]. 电信科学, 2011, 03: 65-70.

[4]Sisalem D, Wolisz A. LDA+: A TCP-friendly adaptation scheme for multimedia communication[C]. 2000 IEEE International Conference on IEEE, 2000, 3: 1619-1622.

[5]Rhee I, Ozdemir V, Yi Y. TEAR: TCP emulation at receivers-flow control for multimedia streaming[J]. Dept. Computer Science, North Carolina State Univ, Raleigh, 2000.

[6] Rejaie R, Handley M, Estrin D. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet[C]. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, 1999, 3: 1337-1345.

[7] Padhye J, Kurose J, Towsley D, et al. A model based TCP-friendly rate control protocol[C]. Proceedings of NOSSDAV'99, 1999.

[8]李方敏，叶澄清，李仁发. 支持最少速率保证的 UDP 拥塞控制机制[J]. 计算机研究与发展, 2001, 08: 988-993.

[9]王彬，吴铁军. 基于显式速率的 TCP 友好的 UDP 拥塞控制策略[J]. 电路与系统学报, 2003, 05: 43-46.

[10]陈明，王东. 基于阈值限定的媒体流 TCP 友好拥塞控制协议[J]. 计算机工程, 2004, 11: 84-86.

[11]魏星，高振中. 网络拥塞控制概述[J]. 桂林航天工业高等专科学校学报, 2008, 01: 35-36.

[12] Jacobson V. Congestion avoidance and control[C]. ACM SIGCOMM Computer Communication Review, ACM, 1988, 18(4): 314-329.

[13]罗万明，林闯，阎保平. TCP/IP 拥塞控制研究[J]. 计算机学报, 2001, 01: 1-18.

[14]Busse I, Deffner B, Schulzrinne H. Dynamic QoS control of multimedia applications based on RTP[J]. Computer Communications, 1996, 19(1): 49-58.

[15] Brakmo L S, Peterson L L. TCP Vegas: End to end congestion avoidance on a global Internet[J]. Selected Areas in Communications, IEEE Journal on, 1995, 13(8):

1465-1480.
[16]刘拥民, 蒋新华, 年晓红, 鲁五一. 无线网络拥塞控制最新研究进展[J]. 计算机工程与应用, 2007, 24: 24-28.
[17]Casetti C, Gerla M, Mascolo S, et al. TCP Westwood: end-to-end congestion control for wired/wireless networks[J]. Wireless Networks, 2002, 8(5): 467-479.
[18]Jung I M, Karayiannis N B, Pei S, et al. The cross-layer adaptation of TCP-Friendly rate control to 3G wireless links[C]. Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on IEEE, 2010: 283-288.
[19]Brown K, Singh S. M-TCP: TCP for mobile cellular networks[J]. ACM SIGCOMM Computer Communication Review, 1997, 27(5): 19-43.
[20]许凡, 曾致远. 基于 TCP 友好速率控制和前向纠错的 MPEG-2 视频传输[J]. 微计算机应用, 2005, 05: 556-559.
[21]何建新, 杨格兰. 有线无线网络 TCP 友好流媒体拥塞控制机制[J]. 计算机系统应用, 2008, 05: 32-35.
[22] Jin S, Guo L, Matta I, et al. TCP-friendly SIMD congestion control and its convergence behavior[C]. Ninth International Conference on. IEEE, 2001: 156-164.
[23]Handley M, Floyd S, Padhye J, et al. TCP friendly rate control (TFRC): Protocol specification[J]. 2003.
[24]Floyd S, Kohler E. Tcp friendly rate control (tfrc): the small-packet (sp) variant[J]. 2007.
[25]张莫. 面向目标检测的视频传感器节点软件设计与实现[D]. 北京: 北京邮电大学, 2014.
[26]刘孟轩. 无线多模网关传输机制的设计与实现[D]. 北京: 北京邮电大学, 2014.
[27]王继先. 视频传输控制方法的研究[D]. 成都: 电子科技大学, 2003.
[28]车晋. 数字电视音视频压缩编码技术分析与研究[J]. 广播电视信息, 2013, 12: 96-99.
[29]张新, 李文生. 基于多模无线网络的速率可控视频传输系统的设计与实现[J]. 中国科技论文在线, 2014, 12.
[30]顾瑞春. 基于 Linux 的高效包过滤技术研究[D]. 包头: 内蒙古科技大学, 2007.
[31]甘仲民, 张更新. 卫星通信技术的新发展[J]. 通信学报, 2006, 08: 2-9.
[32]赵海阔, 赵宇杰, 石蕊. 3G 技术综述[J]. 甘肃高师学报, 2010, 05: 49-52.

致谢

在本人毕业论文完成之际，向我的母校北京邮电大学表示由衷的感谢，感谢这么多年的培育，感谢一切帮助我的老师和同学。

首先感谢我的导师李文生老师。在这两年多的研究生学习中，李老师严谨的治学态度，对工作高度认真的精神一直激励着我，是我获益匪浅。同时在论文工作过程中，李老师对我悉心指导，严格要求，为论文工作提出了诸多宝贵意见，使论文工作得以顺利完成。衷心祝福李老师身体健康，工作顺利。

同时感谢实验室的马华东教授。感谢马老师给我提供的良好的实验环境和实验设施，感谢马老师对我的论文的悉心指导和宝贵建议，为论文的最终成稿提供了巨大帮助。祝马老师工作顺利，身体健康。

感谢实验室的刘亮老师和段鹏瑞老师。两位老师在我参与的项目技术研究中，给予了我很多指导和帮助，同时在生活中也对我关怀备至。另外，两位老师认真专注，孜孜不倦的科研精神，也是我今后工作中学习的榜样。祝愿两位老师身体健康，再创佳绩。

同时还要感谢实验室里同学们的无私帮助，感谢胡金宇、徐盈盈、殷晓林、于韶峰、赵帅。感谢你们的帮助与配合，推进项目和成果日趋完善，也使我感受到团队工作的快乐。

最后感谢我的父母和朋友在我成长道路上给予的养育和支持。

感谢百忙之中抽出时间进行论文评审的老师们！

作者攻读学位期间发表的学术论文目录

[1] 张新，李文生.基于多模无线网络的速率可控视频传输系统的设计与实现.中国科技论文在线.2014 年 12 月.论文编号 201412-215.