清 华 大 学

综 合 论 文 训 练

题目：基于机会路由协议的无线视频传输研究

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：冯晓君

指导教师：杨士强　教授

2009 年 6 月 19 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____导师签名：_____日　期：_____

中文摘要

多媒体无线网状网络在交通监测、城市监控等方面具有广泛的应用前景，但是无线网络传输信道的不稳定性、视频流较高的传输速率和质量要求也带来了多方面的挑战。为此，研究人员提出了机会路由协议。最新的机会路由协议 MORE，用网络编码对数据包随机混合，充分利用了无线网状网络的空间重用性，和传统路由协议相比能够获得更高的传输吞吐率。

本文在一个由六个节点组成的真实网络环境中，实现了一个基于 MORE 协议的，支持多视频流的无线视频传输系统。通过实验，本文分析了 MORE 协议的视频流传输特性，给出了能够较好的平衡传输吞吐率和传输延迟的 MORE 协议中参数的设定方式。为了提高视频传输的实时性，本文结合 MORE 协议的特点提出了一种基于网络速率预测和视频截止时间的自适应丢帧控制算法 SAFDA（Source Adaptive Frame Discard Algorithm）。该算法通过记录传输延迟的历史数据，估计当前的网络传输速率，并结合视频的截止时间进行判断，在源节点自适应的将预测超时的帧提前丢弃，以提高视频整体质量。模拟实验表明，SAFDA 算法能够大大降低视频流传输的丢帧率。在我们的测试中，传输视频最多得到了接近 15dB 的质量提高。

关键词：机会路由；MORE；视频流；视频质量

ABSTRACT

Wireless Multimedia Mesh Networks (WMMNs) which can be used in traffic and urban monitoring, have wide application prospects. But challenges arise due to the unstableness of wireless channel, high quality and high throughput requirements of video streams. To address these problems, Opportunistic routing protocols are proposed. The latest Opportunistic routing protocol MORE, which can fully exploit spatial reuse of WMMNs and randomly mix packets with network coding before forwarding them, performs much better than traditional protocols in throughput.

In this paper, a multi-stream video transmission system with six nodes based on MORE has been set up. Through experiments, the video transmission performance is evaluated and the parameter of MORE is tuned to balance between throughput and video transmission delay. To further enhance the real-time video quality, a Source Adaptive Frame Discard Algorithm SAFDA considering transmission speed and video deadline is presented. In SAFDA algorithm, the historical transmission delay times are recorded to estimate current network speed. Using video deadline, frames predicted to be delayed are discarded adaptively in the source node to get better overall video quality. The simulation results show that this algorithm can significantly reduce frame loss rate and improve video quality by up to 15 dB.

Keywords：Opportunistic Routing; MORE; video stream; video quality

目 录

第1章 引 言

1.1 研究背景

随着无线技术的发展，无线网状网络在越来越多的领域得到了运用，多媒体无线网状网络就是其中的一个研究热点。多媒体无线网状网络的探测节点具备采集视频、音频信息的功能，并将多媒体数据通过无线网络进行传输，在交通监察、城市监控等方面有着广泛的应用前景。但是，除了具有一般的无线网络所共有的，诸如网络传输速率不稳定的问题之外，多媒体无线网状网络还由于其传输的数据量大的特性，面临着更为严峻的无线带宽挑战。

为此，研究人员提出了多种旨在提高无线网络传输吞吐率的路由协议。结合无线网络的广播特性，人们提出了机会路由协议 ExOR[1]。ExOR 协议中，网络节点对于收听到的数据包，按照一定的概率转发，利用网络节点的空间重用性，提高传输吞吐率，与传统的固定路由相比，能获得 35%的吞吐率提高。

机会路由协议 MORE[3]是目前最新的机会路由协议。MORE 的全称是独立于 MAC 层的机会路由和编码协议（MAC-independent Opportunistic Routing & Encoding）。MORE 在进行传输时，首先将大小一致的一定数量的数据包（Packet），组成包组（Batch），然后将包组中的数据包通过网络编码（Network Coding）进行混合。广播时，发送编码之后的数据包。一旦接收点收到足够数量的网络编码数据包后，就可以将原始的包组解码，从而完成数据的传输。MORE 通过采用网络编码而非重传的机制，进一步利用了网络空间的特点和无线网络的广播特性，在丢包率较高的网络环境下能够获得相当高的传输吞吐率。同等情况下，MORE 的吞吐率能在 ExOR 的基础上提高 22%。

对于多媒体无线网状网络，仅仅提高传输吞吐率是不够的，视频流数据具有较高的实时性要求，每一帧图像的播放都有固定的时间限制。基于机会路由协议 ExOR，一种考虑视频截止时间的传输协议 TOR[7]被提出。TOR 通过计算视频的截止时间决定对数据包进行转发或者丢弃，从而减少重传次数，提高视频的传输吞吐率。受到 TOR 协议的启发，那么，我们能否将传输吞吐率更高的 MORE 协议用于视频流的实时传输？进一步的，如果使用 MORE 协议作为视频传输协议，

我们能否根据视频流的特点，通过优化 MORE 协议，使之获得更好的实时性传输性能？在本文中，我们将主要对这两个问题进行研究。

1.2 相关研究工作

1.2.1 多媒体无线网状网络的特点和应用

随着电子技术的发展硬件成本的降低，多媒体无线传感器网络（WMSNs）[9]成为了一个研究热点。在多媒体无线传感器网络中，传感器节点可以通过视频捕获设备对场景中的事件进行实时的拍摄，并经由无线网络进行数据传输。无线网络的使用大大减少了网络线路布设的成本，同时具有更高的灵活性和机动性，在军事、交通监察、城市监控等方面有着广泛的应用前景。

和无线传感器网络不同，在无线网状网络（WMNs）中，并不需要网络中的所有节点都具备视频捕获的功能，而只需要在特定的探测节点上设置视频传感器，中间的网络节点只起到网络传输的作用。另外，多媒体无线网状网络可以采用功能更为强大的处理节点作为中间节点，采用较为充足的电源供应形式，并不会面临传感器网络中常常出现的电池寿命短、存储容量小的限制。但是，和多媒体无线传感器网络一样，多媒体无线网状网络也面临着如下挑战[9]：

1) 不可靠的传输信道

在有线网络中，链路的容量是固定和可以预先确定的，但是在多跳无线网络环境中，网络带宽不仅更为有限，并且可能会受到其他无线信号，城市中的建筑物等的影响而具有多变性和不确定性。网络连接可能出现时断时续的现象，同时传输的误码率更高。

2) 应用层的不同 QoS 需求

在多媒体无线网状网络中，媒体文件的传输质量由特定的应用决定，不同的应用对应不同传输质量的需求。例如，视频流的传输要求视频能够做到实时；而快照应用（Snapshot）则只需要传输间隔一定时间的图像。

3) 高带宽需求

多媒体数据，特别是视频流，和其他数据的传输相比对于传输带宽有着更高的要求，高清视频流对于带宽的需求在几百 Kbps 到几 Mbps 左右。

4) 较高的视频编码技术要求

未经压缩的视频流的数据量是巨大的，分辨率为 176x120 的 QCIF 大小的一帧图像要求大约 21KB 的空间，按照每秒 30 帧（30fps）计算，视频流的码率可

以达到 5Mbps。视频流对带宽的苛刻要求也促使视频编码技术不断发展，力求提高压缩码率。

1.2.2 机会路由协议简介

在无线网状网络中，节点之间的丢包概率往往在 30%以上[2]。如果选择固定的下一跳节点，一次成功的传输可能需要经过多次重传。而采用机会路由协议，所有收听到数据包的节点都可以转发数据包，充分利用了无线网络的空间重用特性，能够很大的提高传输吞吐率。

2005 年，MIT 的 Biswas 和 Morris 提出了 ExOR[1]机会路由协议，这是一种结合了 MAC 和路由的机会路由协议。在 ExOR 协议中，节点向网络广播数据包，然后从所有收听到数据包的节点中选择一个距离目标最近的节点作为下一跳的转发结点。ExOR 对 MAC 层的控制能够保证，在某个时间段，只有一个节点在进行数据包的发送。ExOR 只保证 90%的数据包通过机会路由的方式传输到目的节点，剩下的 10%通过固定路径的传统路由方式发送。作者在一个含有 38 个节点的真实网络环境中进行的实验表明，ExOR 和传统固定路由相比，能够获得 35%的吞吐率提高。

1.2.3 MORE 简介

2007 年，MIT 的 Szymon Chachulski 等人，提出了机会路由协议 MORE[3]，它的全称是独立于 MAC 层的机会路由和编码协议（MAC-independent Opportunistic Routing & Encoding）。和 ExOR 相比，MORE 的显著优点是它的路由算法独立于 MAC 层，可以直接运行在 802.11 协议上，不需要对网络中的节点进行全局的控制和调度，进一步提高了对无线网络的空间重用。MORE 采用网络编码的方法，在进行转发前，将数据包随机混合。这种模式，能够保证节点不会传输完全相同的数据包。在一个由 20 个节点组成的室内无线网状网络中的实验表明，MORE 在单播的情况下，能够在 ExOR 的基础上获得 22%的吞吐率提高。同时，MORE 还支持 ExOR 所缺乏的多播传输方式。

图 1.1 是 MORE 的工作方式示例图，箭头上的数字表示节点之间的连接概率。在图中的情景下，传统路由方法会按照连接概率的高低选择“SRC→R→DST”的路由路径。在无线网络中，由于数据包采用广播的方式发送，有一定的概率，距离目的节点更近的节点也收听到了广播的数据包。例如，当 SRC 向 DST 广播 P1、

P2 两个数据包时，中间节点 R 接收到了 P1、P2，有可能 DST 也已经收听到了 P1。这种情况下，R 只需要向 DST 转发 P2 就可以完成 P1、P2 的传输。

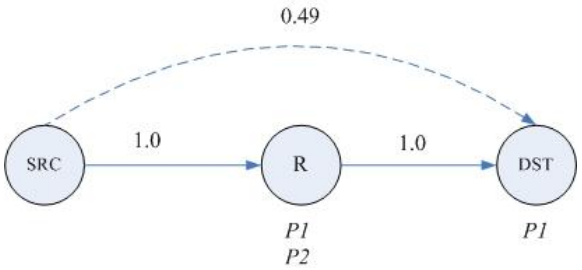

图1.1　MORE工作方式示意图

ExOR 使用全局节点调度的方式控制 R 只对 *P2* 进行转发，而 MORE 采用了网络编码的手段，不需要对节点进行全局调度。通过网络编码，R 向 DST 转发 *P1*、*P2* 的线性组合，例如“*P1+P2*”；DST 可以通过收到的“*P1+P2*”和 *P1* 解码得到 *P2*。事实上，在 MORE 中实际采用的是随机的网络编码系数，*P1*、*P2* 被编码为 $c1 \bullet p1+c2 \bullet p2$，其中 *c1*，*c2* 是随机数。

下面我们给出 MORE 通过网络编码进行数据包发送的简要过程。

源节点在进行文件发送时，首先将文件划分为数据包（packet），进一步，每 K 个数据包被组装成一个包组（Batch），这样的 K 个未经网络编码的数据包被称为初始数据包（native packet）。而经过编码后，数据包可以表示为：

$$p_j' = \sum_i c_{ji} p_i \qquad (1\text{-}1)$$

向量 $\overrightarrow{c_j} = (c_{j1},...,c_{ji},...,c_{jK})$ 被称为 p_j' 的编码向量（code vector）。

源节点还将 MORE 报头加到每个编码后的数据包上。MORE 报头中包含数据包的编码向量、包组 ID（Batch ID）、源节点 IP 地址、目的节点 IP 地址等信息。

当网络中的节点收听到包，将首先检查编码数据包中是否包含新信息。更确切的说，一个编码数据包中包含新信息，当且仅当包的编码向量和节点之前收到的所有包的编码向量线性无关。检查线性相关性可以用高斯消元法实现。含有新信息的包被称为新进包（innovative packet）。转发节点将新进包和已有的编码数据包进行又一次的随机网络编码，之后将编码后的包发送出去。值得注意的是，

由于按照线性编码的方式，重新编码后的编码数据包仍然是初始数据包的线性组合。

目的节点对于接收到的包，进行新信息检查，丢弃非新进包。目的节点收到的新进包的数目为 K 时，说明已有足够的信息可以解出初始数据包。目的节点通过高斯消元法解码出 K 个初始数据包。同时，目的节点通过最佳的固定路径向源节点返回 ACK。在 MORE 协议中，ACK 报文具有更高的传输优先级。源节点收到 ACK 后，停止对当前包组的发送，整个包组的传输完成。

MORE 协议中的另一个要点是对于网络节点转发次数的分布式控制。

我们之前提到，无线网络中，带宽是稀缺资源，而一种提高无线传输吞吐率的方式就是减少重传次数。MORE 的作者通过理论分析，给出了网络转发节点对数据包的最大理论转发次数，从而实现控制重传次数的目的。

假设网络中节点 i 比节点 j 距离目的节点更近，i 收听到从 j 发送过来的数据包后，对数据包的最大所需发送次数为 TX_credit，z_i 和 z_j 分别为 i 和 j 发送数据包的期望次数，e_{ji} 为节点 j 到节点 i 的丢包概率。那么对于节点 i，计算公式如下：

$$TX\_credit_i = \frac{z_i}{\sum_{j>i} z_j (1 - e_{ji})} \tag{1-2}$$

在 MORE 的传输中，设置了一个传输信用（TX_credit）计数器（credit counter），每当节点 i 收到一个上游传来的数据包时，在计数器上增加 i 所对应的 TX_credit，如果计数器的值为正，那么 i 创建一个网络编码包并进行发送，同时在计数器上减 1。由此，每个节点都能够通过计算传输信用独立的确定对每个数据包的转发次数。

1.2.4 机会路由协议在视频流传输中的应用

CUM 的 Mei-Hsuan Lu 等人，首次评测了采用机会路由协议进行视频流传输的性能。在他们 PV2007 的文章[15]中，提出了一种离散时间马尔科夫链模型用以描述机会路由传输行为。通过该模型，作者给出了采用机会路由进行传输时，一次成功传输的期望传输次数：

$$ETX_{OR} = \sum_{n=1}^{\infty} n \cdot \Pr(ETX_{OR} = n) \tag{1-3}$$

上式中，ETX_{OR} 表示一次成功传输所需要的期望次数，$\Pr(ETX_{OR}=n)$ 表示完成传输时，重传次数为 n 时的概率。

而采用传统的固定路由方式，例如传统的最佳路径路由算法[17]，一次成功的传输期望的传输次数为：

$$ETX_{SPP}=MIN(\sum_{l\in L}\frac{1}{P_l}) \qquad (1\text{-}4)$$

L 表示最优的路径，l 为最优路径上的点，P_l 为点 l 的连接概率。通过在一个真实网络中得到的连接概率数据集上的随机仿真测试表明，使用机会路由协议进行视频传输得到的视频质量能够在传输路由方式上提高 2.8 到 3.4dB。

结合机会路由协议 ExOR 和视频流传输的特点，Mei-Hsuan Lu 等人还提出了基于截止时间的机会路由传输协议 TOR[7]（Time-aware Opportunistic Relay scheme），按照在视频中的实际时间关系，每个数据包被赋予一个到达接收节点的截止时间。在中间节点，根据数据包中写入的截止时间计算转发截止时间。转发截止时间可用如下方法计算：

$$\text{relay\_deadline}=\text{packet\_recv\_time}+\text{time\_to\_relay}-\text{duration}$$

$$\text{time\_to\_relay}=\text{packet\_deadline}-\text{current\_time}$$

其中，packet_deadline 为数据包中写入的截止时间。通过计算，网状网络中的节点可以判断对于收听到的数据包应该转发或者丢弃。

考虑到多视频流的传输情境，Hulya Seferoglu 等人提出了 NCV（Network Coding for Video）和 NCVD 算法[18]。这两种算法的关键想法是将不同视频流的数据进行混合网络编码。

NCV 和 NCVD 算法假设网络中可以获得各个节点已经接收到的数据包的信息。传输的过程中，源节点维护一个缓存，存储待发送的数据包。进行传输时，源节点根据接收节点中已有数据包的信息，选择那些接收节点能够立即解码的编码方式，将不同视频流中的数据包混合编码，并进行传输。

作者进行的模拟测试表明，使用 NCV 和 NCVD 这两种算法前后，视频流传输吞吐率和视频的质量都获得了提高。但是，这两种算法在使用中需要获得整个网络中已完成传输的数据包的信息，在真实的无线网状网络中，这个条件很难被满足。

在本文中，我们将在 MORE 协议的基础上对多视频流的传输特性进行评测，并结合 MORE 协议的特点提出提高视频传输质量的方法。据我们所知，目前还没有关于 MORE 协议用于多视频流传输方面的工作。

1.3 论文工作

本文的工作主要分为三部分。

第一，实验系统的实现。我们组建了由 6 个节点组成的真实无线网络环境。在这个真实网络环境下，我们实现了基于 MORE 协议的支持多个源点同时传输视频文件和实时播放视频流的实验系统。

第二，MORE 协议在视频传输中的应用研究。我们在实验系统中在不同网络传输功率、不同码率的视频文件、不同源节点数目等多种情景下，视频的传输性能进行了详细的测试和分析，给出了 MORE 协议对视频文件传输的关键参数，例如帧的传输延迟、包组传输延迟、传输吞吐率、丢帧率等的影响。

第三，为了使视频的实时传输质量得到提高，我们设计了一种基于网络速率预测和视频截止时间的源节点传输码率自适应的丢帧控制算法：SAFDA（**S**ource **A**daptive **F**rame **D**iscarding **A**lgorithm），使得视频数据的传输实时性有了明显的提高，使用 SAFDA 算法后视频的 PSNR 与使用前相比，最多能够获得 15dB 的提高。

1.4 论文组织结构

本论文共分六章，各章的主要内容如下：

第一章 引言。分析了研究的背景，介绍了机会路由协议 ExOR、MORE 以及将机会路由协议应用于视频传输的几个已有工作。最后介绍了本文的工作内容和组织结构。

第二章 基于机会路由协议的视频传输系统实现。首先介绍如何在真实网络环境中实现 MORE 协议，同时介绍了在 MORE 协议的实现平台上搭建支持多视频流并发传输的实验系统的实现方法。实验系统的实现是后续工作的基础。

第三章 MORE 协议的视频传输性能评测。本章通过实验，介绍了使用 MORE 协议进行视频流传输的特性。包括包组大小的设定对视频流传输关键参数例如吞吐率、帧的延迟等方面的影响。以及在不同网络传输功率、不同码率的视频文件、不同源节点数目等多种情景下视频的传输性能。

第四章 源节点自适应丢帧算法 SAFDA 的设计。视频流和普通的数据文件不同，具有实时性的要求，在本章中，我们提出了一种基于网络速率预测和帧截止时间的源节点传输码率自适应的丢帧控制算法 SAFDA（**S**ource **A**daptive **F**rame **D**iscarding **A**lgorithm）。该算法使用截止时间作为丢帧判断的依据，并充分考虑了 MORE 协议的传输特点。

第五章 SAFDA 算法的性能评测。本章中，我们对 SAFDA 算法进行了实验评测。实验表明，通过使用 SAFDA 算法，在搭建的真实网络平台上，视频流的质量最多能够得到 15dB 的提高。

第六章 总结和进一步的研究工作。对全文进行总结，并提出了进一步研究的可能方向。

第2章 基于机会路由的无线视频传输系统

在本章中，我们将介绍基于机会路由协议 MORE 的视频传输系统的实现原理。我们的系统运行在真实的网络环境中，系统的主要功能包括三项：视频文件的传输，多视频流的传输、视频流的实时播放。我们还对在多流情况下的调度算法进行了分析和设计。

2.1 Base Code 介绍

我们的实验系统是在 MORE 的论文作者提供的基代码（Base Code）[4]的基础上实现的。基代码由 C++实现代码和 python 脚本代码两部分组成。C++实现代码运用 Click 软件路由器[5][6]函数库，实现了对网络数据包的接收、组装、发送等行为的控制以及数据包网络编码等功能。python 脚本代码提供了控制网络中的各个节点、测量节点间的连接概率、计算各个节点在机会路由中的传输参数、控制传输的起始和结束时间等功能。基代码的实现中，仅能够支持单数据流。

2.1.1 Click 软件路由器

Click 软件路由器是一套由 MIT 开发的，能够灵活创建路由功能的软件体系结构，可以在 Linux、Mac OS X 和 BSD 操作系统上运行。使用 Click 进行数据包路由，传输码率最大可达 35Mbps[5]。

一个 Click 软件路由器由一个或多个包处理模块组成，简单包处理模块被称为 Element。一个 Element 可以实现一种简单的包处理功能，例如数据包分类、调度、与网络硬件通讯等。每个 Element 由一个 C++类（Element class）实现。类中的参数可以使用配置字串（configuration string）设定。多个 Element 可以组成一个复合 Element，从而能够完成更为复杂的功能。Element 中定义了和其他模块通讯的端口（Port）。用户可以按照 Click 指定的语法，编写自定义的 Element。图 2.1 给出一个 Element 的示意图。

Click 软件路由器通过配置文件（Click scripts）指定 Element 之间的连接关系，配置文件按照 Click 定义的语法将 Element 按照数据包的处理顺序连接，组成一个模块连接图。图 2.2 是三个 Element 连接的示意。其对应的配置文件为：

```
FromDevice(eth0)
    → Counter
    → Discard;
```

上述脚本中指定了三个 Element，分别是 FromDevice、Counter、Discard，其中 FromDevice 使用配置字串“eth0”进行初始化。

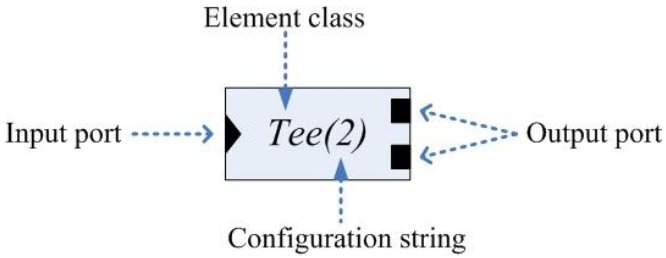

图2.1　Element的示意图

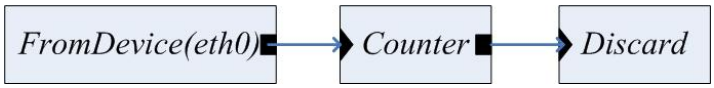

图2.2　三个Element连接的示意图

在 Click 中，C++的 STL 库不能兼容使用，Click 提供了一套类似 STL 的类库，提供了诸如 Vector、String、HashTable、HashMap 等常用类型。同时，Click 提供了一套机制，使得可以在编译阶段指定所需要的外部类库。

Click 提供了动态控制 Element 行为的句柄（Handler），分为读句柄（Read Handler）和写句柄（Write Handler）两种。句柄的使用，使得外部程序能够动态读取 Element 的状态或者实时写入 Element 的控制参数。

在 MORE 协议的基代码中，主要用到了如下几个重要的 Element：

1) FromDevice.u

功能：从网络设备读取数据包。

2) Classifier

功能：按照指定的模式对数据包进行分类。可以利用这个 Element 提取自定义的报文头部，对报文分类。

3) PrioSched

功能：按照优先级顺序，从不同端口选择数据包。在实现 ACK 的报文的优先级高于数据报文时使用。

4) SetTXPower

功能：设定数据包的传输功率。

5) ToDevice.u

功能：将数据包发送到网络设备。

6) AverageCounter

功能：统计经过的数据包和吞吐率，用于测试统计结果。

7) Print

功能：打印出数据包中指定长度的数据，用于打印消息类型、时间戳。

8) More

功能：用户自定义模块，是实现 MORE 协议的中心模块，具有包括进行网络编码、数据包组织、转发、机会路由等关键功能。MORE 中有三个数据输出端口，分别用于发送 ACK 报文、发送数据报文、丢弃收到的报文。

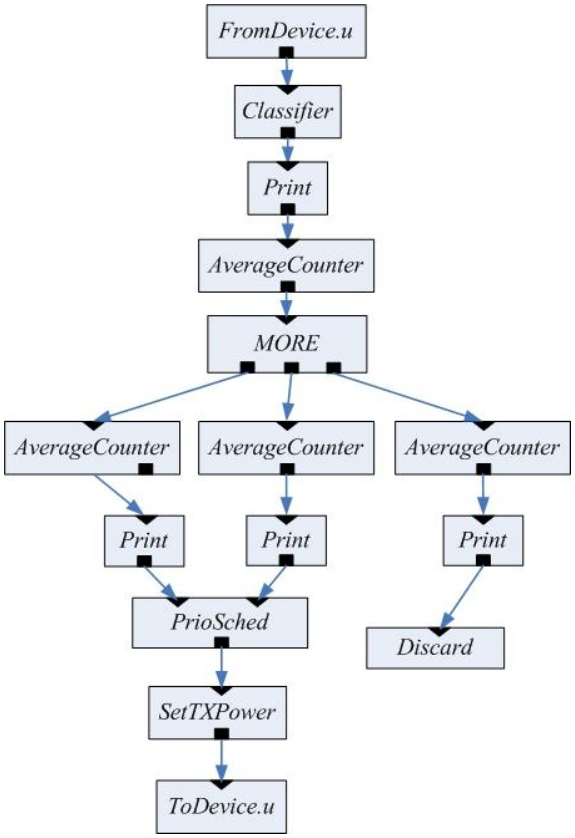

图2.3　MORE中主要Element的连接关系

MORE 中使用的主要 Element 的连接关系，可以用图 2.3 表示。

2.1.2 MORE 协议实现中的重要类介绍

在基代码中，MORE 被编写成一个 Click 的 Element，能够支持如下受限的功能：

1) 测试脚本只支持单播
2) 只支持单个数据流
3) 固定的路由链路
4) 数据包的载荷由 Click 自动生成

从以上受限功能我们可以看到，基代码是一个用于模拟机会路由数据传输的测试系统，不能用于视频文件和视频流的传输。同时，基代码中已经实现了机会路由协议的关键代码，包括网络编码和源点、转发节点、接收节点对包组的基本处理功能，主要的类及功能如下：

1) Packet 类

Click 提供的类。描述网络中的包，在各个元素之间传递。可以根据需要对数据包头和载荷进行空间分配和赋值。在 MORE 协议的实现中，接收和发送的数据都被封装成 Packet。

2) CodedBlock 类

MORE 中封装数据包网络编码的类，保存随机的网络编码参数和编码后的数据。

3) Matrix 类

MORE 中对网络编码系数矩阵进行封装的基类，它的三个继承类 MatrixMgr、PartialGEMatrix、FullGEMatrix 实现了高斯消元法和对包组的编解码。

4) MOREBatch 类

MORE 中封装包组的类，包含包组的序列号、包组发送的传输信用（TX_credit）等数据成员和相关函数。

5) MOREFLow 类

MORE 中封装数据流的类。最重要的功能是源点可以通过这个类实现初始数据包（native packet）的组合。

6) MORE 类

MORE Element 的实现类，成员变量中包含一个 MOREFlow。定义了对于接收到的数据包的处理函数 push()以及发送数据包函数 pull()。同时提供了设定节点传输参数的写句柄（Write handler）。

2.2 单流传输的实现

2.2.1 H.264 压缩算法简介

H.264[12]是由 ITU-T 和 ISO/IEC 的联合开发组共同开发的最新国际视频编码标准。在同等图象质量下的压缩效率比以往的标准提高了 2 倍以上[13]。在我们的实验系统中，将采用 x264 软件[16]压缩后的文件作为传输的视频文件，x264 是一款实现了 H.264 标准的免费软件。

H.264 文件的码流在第一层由 NAL Unit 组成，通过解读 NAL Unit，可以获得视频每一帧的数据。我们将以帧为单位对视频文件进行数据包划分。

2.2.2 视频文件打包发送和接收的实现

对于进行传输的 264 文件，我们通过对文件的解析，将视频以帧为单位进行数据包的拆分，我们采用了类似 RTP[10]的打包方式。由于在网络中传输的数据包大小通常为 1500 字节左右，并考虑到 MORE 的报头将占用一定的空间，我们将一帧的数据拆分成 1400 字节大小的数据块。我们在数据块前加入报头，报头结构如下：

```
struct PacketHeader
{
    uint32_t size;     // 数据量大小
    uint32_t sequence; // 数据包序列号
    uint32_t timeStamp;  // 帧时间戳
};
```

其中包含数据包的实际大小（size），数据包在一帧数据中的序列号（sequence），数据包所在帧的时间戳（timeStamp）。

这些数据块被保存在一个链表中，由 PacketSource 类封装。

使用 MORE 协议进行数据传输时，源节点从 PacketSource 类中取出 K 个数据块，组合成包组并发送。数据块长度不足 1400 字节的部分用 0 填补。

源节点对包组的大小 K 有两种设置方法。其一为固定的包组大小，无论一帧的数据被划分的数据包的数目如何，包组大小不变。在这种方法下，同一帧的数据有可能被拆分到两个或多个不同的包组中。其二是按照帧划分的数据包的数目组织包组，每个包组的大小可变。在这种方法下，可以通过一定的算法保证同一帧的数据不会被拆分到两个不同的包组中。在协议的实现中，对于第二种包组划分方式，我们采用了动态计算算法设定包组大小。算法流程如图 2.4 所示。

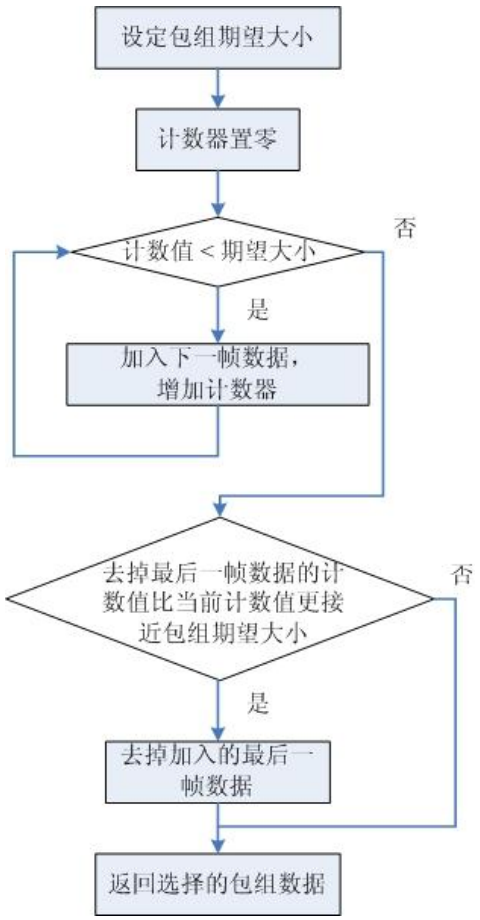

图2.4　动态包组大小计算算法流程图

具体实现方法如下：

第一步：设定包组大小的期望取值，比如 64。

第二步：将每帧的数据包循环加入包组，直到包组的总数据包数量超过设定的期望取值。

第三步：包组大小的动态调整。对加入最后一帧前后的包组大小进行比较，更为接近设定的期望取值的值作为包组的大小。例如，假设最后一帧的数据包加入包组之前包组中的数据包数目为 50，加入最后一帧的数据包后，如果包组中数据包数在 78 之内，则按照加入最后一帧后的包组数量作为包组大小。否则，取消最后加入的一帧数据，仍然将 50 作为最后包组的大小。

动态计算算法可以保证同一帧的数据不会被拆分到两个包组中，同时，每个包组的大小都接近包组期望大小。

源节点和中间节点对编码后的包组将按照 MORE 协议进行发送和转发。接收节点将收到的数据包解码后，解析数据包报头，通过 timeStamp 和 sequence 确定数据包在视频中所处的位置，进而按照数据包的实际顺序，写入 264 码流，从而实现视频文件的接收。接收视频文件的功能在系统中用类 PacketReceiver 封装。

2.3 多流传输的实现

2.3.1 多流的定义和识别

在基代码中，只支持单一数据流的传输。为了对多流传输进行支持，需要在 MORE 的报头增加流的标识。我们的系统中在 MOREFlow 类中增加一个流的标识变量：

uint32_t flowid

同时，在 MORE 类中，取消原有的单一 MOREFlow 定义，改为包含类型为 MOREFlow 的向量（Vector）：_flowVec，以及 flowid 到流的映射表：_flowidVec。映射表的对应位置，保存了相应的流的 flowid，这样，我们就可以通过给定的 flowid 在_flowidVec 查询并取出相应的需要处理的流。

2.3.2 多流传输的实现

源节点传输的文件信息是在 Click 的配置文件中指定的。为了支持多个数据流，我们在每个源点的配置文件中指定了流的 flowid 信息和源节点进行传输的文件名。

转发节点需要分别处理所有经过的流。不同的流具有不同传输参数，最重要参数的是传输信用（TX_credit）。这也需要在配置文件中针对每个流进行指定。另外，在 MORE 模块中，对于数据包的和 ACK 的转发是通过 pull()函数实现的。在多流的情况下，由于 pull()函数每次调用只能够选择一个流进行处理，因此，

多流之间的调度方式也是一个需要仔细设计的环节。我们在后面的内容中会专门讨论多流调度算法的设计。

接收节点对于不同的流，应该能够分别接收处理。对于不同源点发送的不同文件，接收点需要写入不同的接收文件。因此，我们需要在单流的接收封装类 PacketReceiver 中增加对多流的支持，对于接收到的数据包，首先通过头部的 flowid 信息，查询到对应的流，并对不同的流指定不同的接收文件。

2.3.3 多流网络控制的实现

MORE 通过 Python 脚本文件设定各节点的运行参数，并控制网络节点的运行。在多流的条件下。控制脚本文件需要进行如下的修改：

1) MORE 配置字串中增加对 flowid 的指定

一方面，需要在 MORE 的类实现中，增加对 flowid 参数的支持。另一方面，需要在脚本文件控制运行的部分，在 MORE 的配置字串中加入 flowid 参数。不同的源节点需要具有不同的 flowid 标识。

2) 网络节点的参数设定

MORE 协议的正确运行，对于每一个数据流，需要指定所有网络节点对于该流的传输参数。我们在 Python 脚本程序中加入了对于每个流的循环处理，分别计算每个流在网络中的所有节点具有的传输信用（TX_credit），从而实现多流的支持。

2.4 多流调度算法的设计

正如前面所提到的，在多流的情况下，每个节点每次只能处理一个流的数据，因此，需要考虑多流调度的算法。

在我们的系统中，实现了三种方法，分别是简单轮转调度法、优化轮转调度法、序列号优先级调度法。

2.4.1 简单轮转调度法

在 MORE 模块中，设置一个计数器 count，每次 pull()函数的调用，在 count 上增加 1。假设共有 n 个编号分别为 1~n 的流经过该节点，则每次 pull()函数处理编号为 count mod n + 1 的流。算法流程如图 2.5 所示。

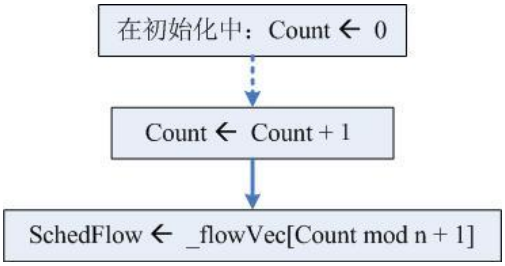

图2.5　简单轮转调度法流程图

简单轮转调度法实现简单，对于流的处理相对公平，但是在当前获得调度的流没有收到新进包或者没有 ACK 需要发送时，该次的函数调用就不会做任何实质处理，这会影响整体的传输效率。

由此，我们对简单轮转调度法进行了优化。

2.4.2 优化轮转调度法

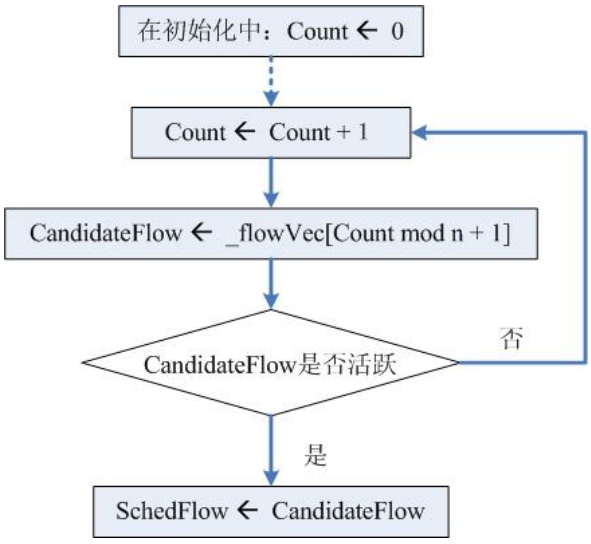

图2.6　优化轮转调度法流程图

优化轮转调度法针对简单轮转调度法可能出现的函数调用不做实质处理的问题进行了优化。

在优化轮转调度法中，同样设置了一个计数器 count，每当 pull()函数被调用的时候，系统从编号为 count mod n + 1 的流开始，对每个流进行测试，如果该流

有新进包或者需要进行 ACK 发送时，称该流为“活跃流”，并将该流设置为调度流，同时更新 count 为调度流的编号。算法流程如图 2.6 所示。

采用这样的优化方法，既保证了所有的流能够按照比较公平的顺序都能够获得 pull()函数中处理的机会，同时也避免了 pull()函数可能出现的无功而返的问题。

2.4.3 序列号优先级调度法

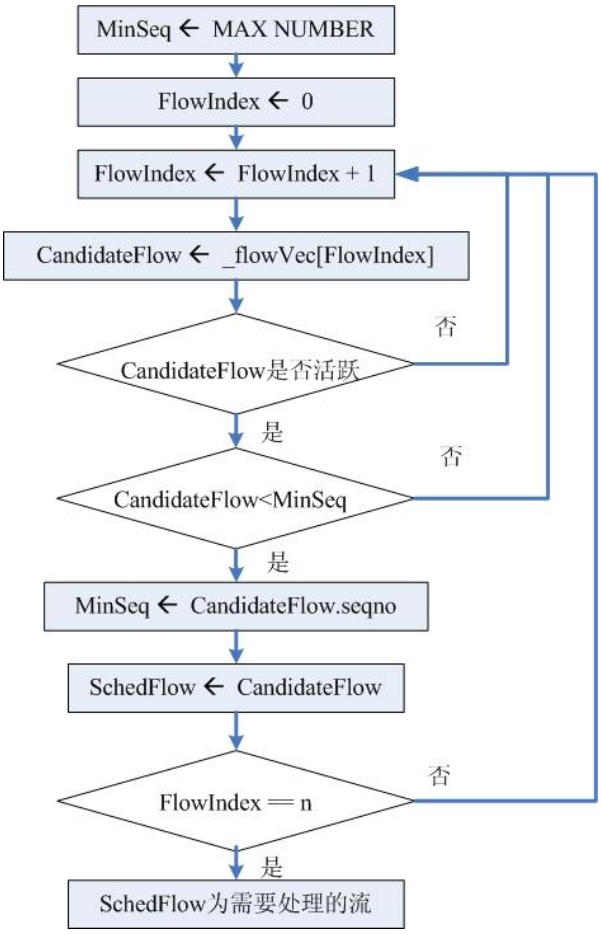

图2.7　序列号优先级调度法流程图

每个经过节点的流都会在当前节点留下当前的序列号信息，我们可以在 MORE 模块的成员变量的 MOREFlow 中获取。

很自然的，一种按照序列号优先级进行调度的方法可以按照下列的步骤进行。类似优化轮转调度法，我们希望保证每次 pull()函数的调用都不会无功而返。但是和轮转的流查询方式不同，我们按照流已经发送的序列号大小升序排列查找的

顺序。序列号低的流拥有更高的优先级。优先级高的流如果有新进包或者 ACK 的发送需要，我们就在当次的 pull()函数处理中，使该流得到调度。算法流程如图 2.7 所示。

某个流的序列号更低，意味着，比其他流发送的包组的数量少，为了能够对所有源节点的传输获得一个平衡，我们赋予序列号低的流更高的优先级。同时，这样做并不会影响通过节点的吞吐率，因为我们可以保证，每次 pull()函数的操作都会有一个流得到处理。因此，这个调度方法在吞吐率上应不会低于优化轮转调度法。

2.5 视频流实时传输和播放的实现

我们利用开源软件 VLC media player[11]实现了视频流实时传输和播放的功能。

VLC media player 是一款开源的支持多种操作系统、多种音视频格式的媒体播放器。最初是法国 École Centrale, Paris 大学的学生项目，后来发展成全世界爱好者共同维护的开源社区。目前已经发布 0.9 版本。

VLC 不仅具有播放器的基本功能，还能够将视频通过流化处理广播到网络中，同时也能接收和播放网络视频流。在系统中，我们将使用 VLC 流化后的视频数据作为传输数据块，并借助 VLC 播放。

源节点的视频流数据通过 VLC 播放器获取。处理过程如图 2.8 所示。

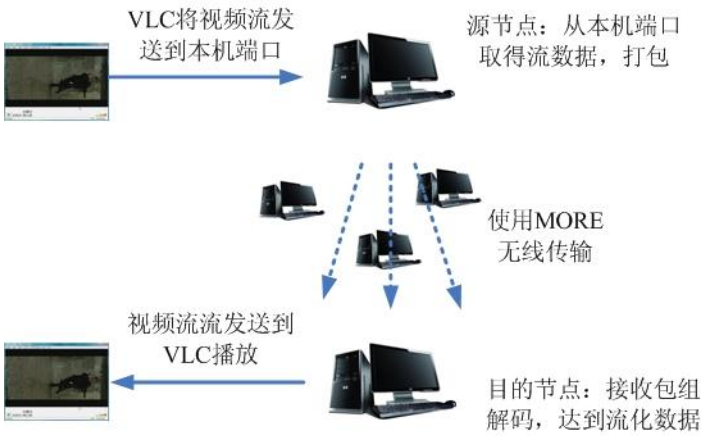

图2.8　实时播放处理流程图

第一步：在源节点打开 VLC 播放器，选择视频文件，进行流化处理，发送到本机的一个端口，默认端口号为 1234

第二步：启动源节点的传输，建立一个 StreamSource 类，该类维护一个线程负责建立到本机 1234 端口的 socket 连接。将该端口的数据接收并封装成固定大小的数据包，加上报头信息后，加入链表。

第三步：源节点在构建源数据包的过程中，从 StreamSource 类的数据包链表取出数据包，组合成包组并发送。

第四步：接收节点收到并解码包组后，将数据提取出来，根据 flowid，发送到本机的不同端口，默认端口设置为 2000+flowid。

第五步：打开接收节点上的 VLC 播放器，接收本机一个或多个端口上的视频数据并播放。当接收到足够的视频数据后，VLC 播放器就可以实时的播放视频。对数据包的接收和发送功能封装在系统的 StreamReceiver 类中。

实时播放的效果见图 2.9。我们分别采用两个源节点（图 2.9 左）和三个源节点（图 2.9 右）利用 MORE 传输和播放电影《Yes Man》。

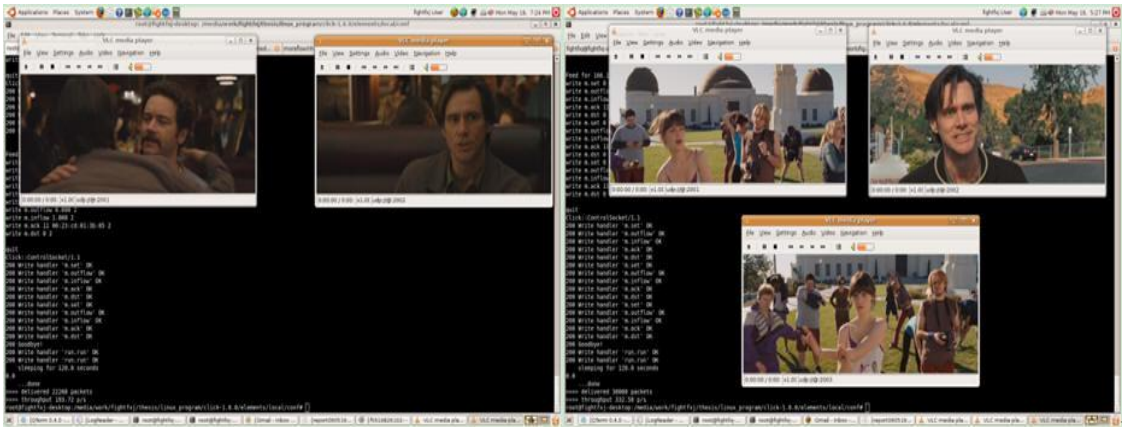

图2.9　实时播放效果图

2.6 本章小结

在本章中，我们在 Base code 的基础上，实现了基于机会路由 MORE 的多视频流传输系统。我们的系统通过 Click 软件路由器，实现了控制网络硬件发送和接收数据包的功能，从而使得我们的系统可以运行在真实的网络环境中。在系统中，我们传输的视频文件通过最新的视频压缩标准 H.264 压缩。系统可以支持多

个视频流的同时传输，并实现了三种多流间的调度算法。除此之外，借助 VLC 播放器的视频流化和播放功能，该系统还可以支持多个视频流的实时播放。

第3章 MORE 协议的视频传输性能评测

在本章中，将基于机会路由 MORE 的视频传输系统，对 MORE 协议中包组的不同划分方式、不同视频流数量、不同网络环境、不同调度算法等各个方面的视频传输性能进行评测。同时，本章将针对视频流的实时性要求，分析 MORE 协议在视频传输实时性上的不足之处。

3.1 实验平台介绍

3.1.1 网络布局

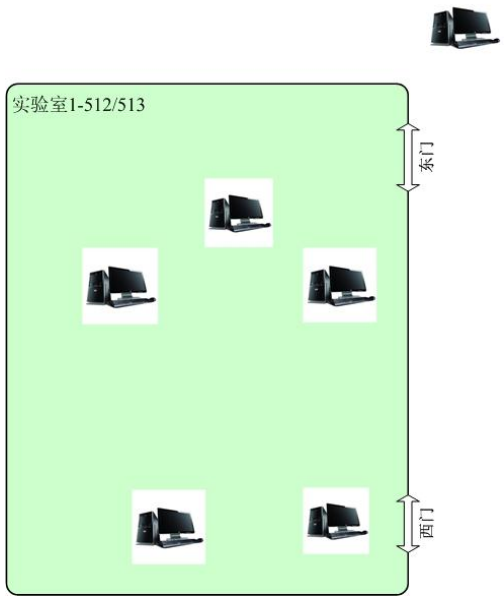

图3.1 网络拓扑图1

在清华大学 FIT 大楼 512、513 房间，我们用 6 台电脑作为网络节点构建了一个真实的无线网状网络环境。6 台电脑在实验室中的分布如图 3.1 所示。

3.1.2 硬件环境

我们的 6 台电脑上，其中 5 台的 CPU 为 Intel Pentium 4，内存 1GB，另 1 台的 CPU 为 Intel Pentium Dual，内存 3.48GB。

在每个节点上我们采用的无线网卡为 TP-Link 的 TL-WN550G 型台式机无线网卡。该无线网卡可以支持 1Mbps、5.5Mbps、11Mbps 传输速率，外置固定全向天线。在我们的所有实验测试中，均采用 5.5Mbps 传输速率。

通过调节无线网卡的发送功率，我们可以粗略获得不同质量的节点间连接概率。在较差的网络条件下，节点间的连接概率在 50%以下。

3.1.3 软件环境

各节点上运行的操作系统均为 Ubuntu8.04，其 linux 内核版本为 linux 2.26.24。在我们采用的 Ubuntu 操作系统上采用 madwifi[14]作为无线网卡驱动程序。

Click 在操作系统上以一个用户层次进程的方式运行，通过与无线网卡交互，直接处理网卡接收到的原始数据。

3.2 单流传输性能评价

对 MORE 的实验测试中，作者提到，包组的大小设置会对传输的吞吐率有影响，包组大小较大时，吞吐率较大。对于视频文件，一方面，我们希望获得更大的吞吐率，但是随着包组的大小的增加，包组中包含的帧的数目也随之增加，部分数据帧的传输延迟将相应增大，我们需要在总传输吞吐率和帧的延迟中权衡，力求达到最优的传输效果。我们将在单流传输实验中对包组的划分方式给视频文件传输性能带来的影响进行测试和比较。

3.2.1 包组的划分

假设包组的大小为 K，即在一个包组中包含 K 个数据包。K 有两种不同的取值方式：

1) 固定的 K 值

在整个传输过程中 K 的取值固定。在这种划分方式下，有可能同一帧的数据包被划分到不同的包组中。在计算帧的发送时间和接收时间的时候，帧中的第一个数据包的发送时间作为该帧的发送时间，帧中最后一个数据包被接收到的时间作为该帧发送完成的时间。

实验中测试的 K 固定的取值情况为：1，4，8，16，32，64，48。分别记为：K=1，K=4，K=8，K=16，K=32，K=48，K=64。

2) 可变的 K 值

即随着帧包含的数据包数量变化的 K 值。根据帧中包含的数据包数量设置每个包组的 K 值。在这种情况下，同一个帧的数据能够保证划分到同一个包组中。我们使用前文介绍的动态包组大小计算算法对包组大小进行实时计算。

实验中，测试的 K 可变的取值情况中设置包组的期望大小分别为 16、32、48、64。分别标记为：K=var16，K=var32，K=var48，K=var64。

3.2.2 实验测试数据

在实验中，我们使用两种不同码率的 264 文件作为传输的视频文件，测试文件信息如表 3.1 所示。在测试中，每个数据包的大小为 1400 字节。

表3.1　包组划分测试视频信息表

视频名称	视频类型	帧数	GOP 长度	码率（Kbps）	平均每帧数据包数
night.264	1280x720	90	30	3500	14.3
foreman_2000.264	352x288	300	25	2000	9.0

3.2.3 主要测试指标

1) 帧平均传输延迟（Average Frame Delay）

定义为发送端收到该帧的 ACK 的时间减发送端开始发送该帧的时间。

2) 整个文件的传输吞吐率（Overall Throughput）

定义为总的数据包的数目除以整个文件的传输时间。值得注意的是，按照这里的定义，传输吞吐率的计算单位为 Packets/s。假设吞吐率 100 Packets/s，那么换算为 bps 作为衡量单位，可以计算如下： 100Packets/s = 100x1400x8/1024 = 1093.75Kbps。也就是说，100Packets/s 的吞吐率大约对应 1000Kbps 的码率。

3) 平均延迟抖动（Average Transmission Jitter）

延迟抖动（T_{Delay}）定义为发送端开始发送该包组的时间（T_{Send}）和接收端开始解压的时间（T_{Decode}）之差，平均延迟抖动（T_{Jitter}）是这些延迟数据中最大的和最小的差对包组大小（$Average\_Batch\_Size$）的平均。具体计算为：

$$T_{Send} - T_{Decode} = T_{Delay} \tag{3-1}$$

$$T_{Jitter} = \frac{\max\{T_{Delay}\} - \min\{T_{Delay}\}}{Average\_Batch\_Size} \quad (3\text{-}2)$$

这个参数可以反应无线网络速率的不稳定性。

4) 视频丢帧率（Loss Rate）

在视频中，每帧的图像都有固定的播放时间。假设视频的帧率为每秒播放α帧，那么，视频中总第i帧图像的开始播放时间为i/a秒。

假如某一帧数据在播放时间之后才完成传输，那么我们认为该帧被丢弃。丢帧率即为所有被丢弃帧占视频总帧数的比例。这个参数可以反映视频传输的实时性。

3.2.4 测试结果和评价

我们在本小节中，将从四个测试参数分析单流测试的结果。

1) 帧平均传输延迟

我们分别在 K 取值固定和 K 取值可变的情况下，改变包组的大小，测试帧的平均传输延迟。图 3.2 是 K 取值固定情况下的结果，图 3.3 是 K 取值可变情况下的结果。

从图中可以看到，当包组大小小于平均每帧数据包数时（K<16），每个帧需要多个包组进行传输，这种情况下，帧的传输延迟随着 K 的增加而减小。当包组大小大于平均每帧数据包数时（K≥16），一个包组中可以包含 2 个以上的帧的数据包，这种情况下就出现了随着 K 的增加，帧的传输延迟增大的情况。

同时，在期望的 K 值相同的条件下，我们比较了两种 K 的取值对于帧传输延迟的影响。图 3.4 是针对 night.264 视频的结果，图 3.5 是针对 foreman_2000.264 的结果。从图中，我们可以看到，在期望的 K 值基本相同的条件下，固定的 K 取值方式下获得的传输延迟稍大于可变的 K 取值。以上实验结果和我们的理论分析相符。

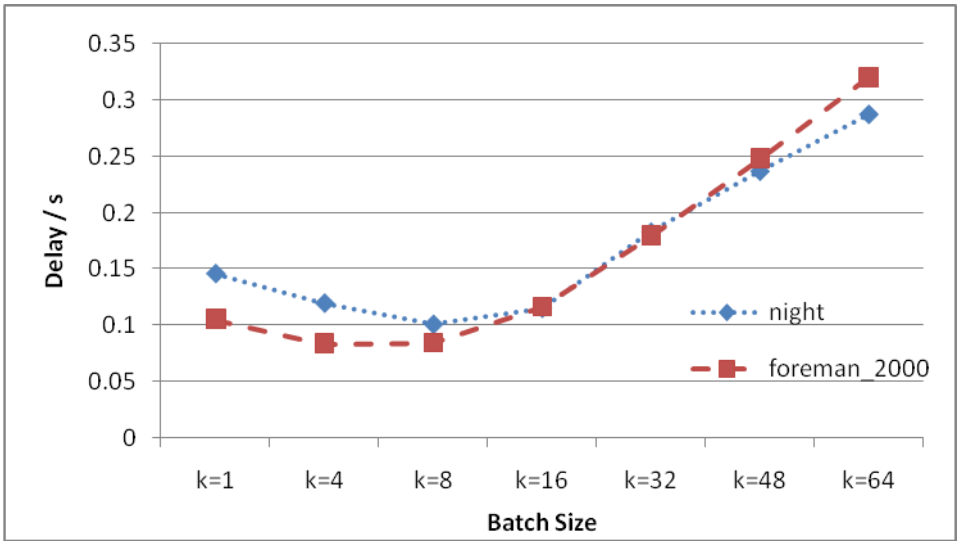

图3.2　K值固定条件下，K与帧平均传输延迟的关系

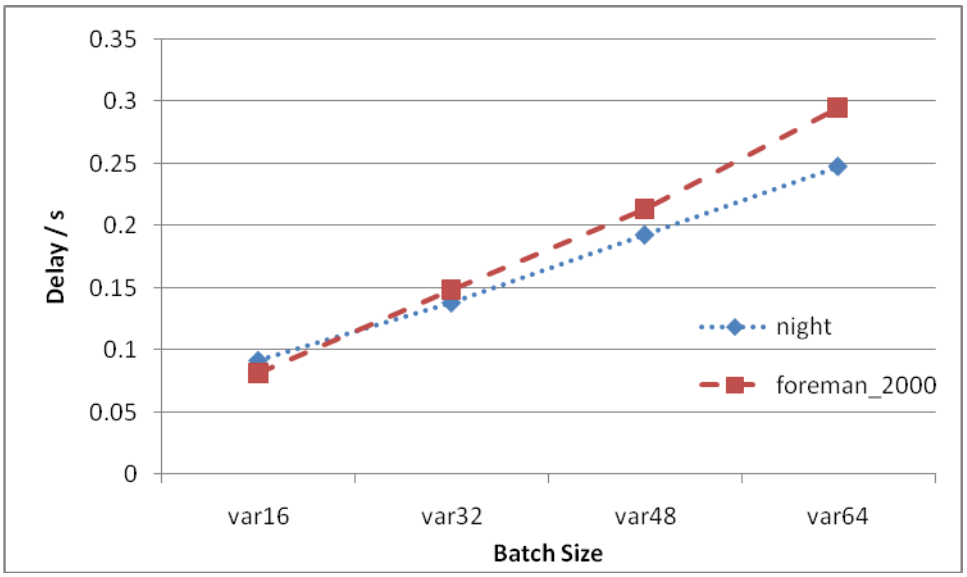

图3.3　K值可变条件下，K与帧平均传输延迟的关系

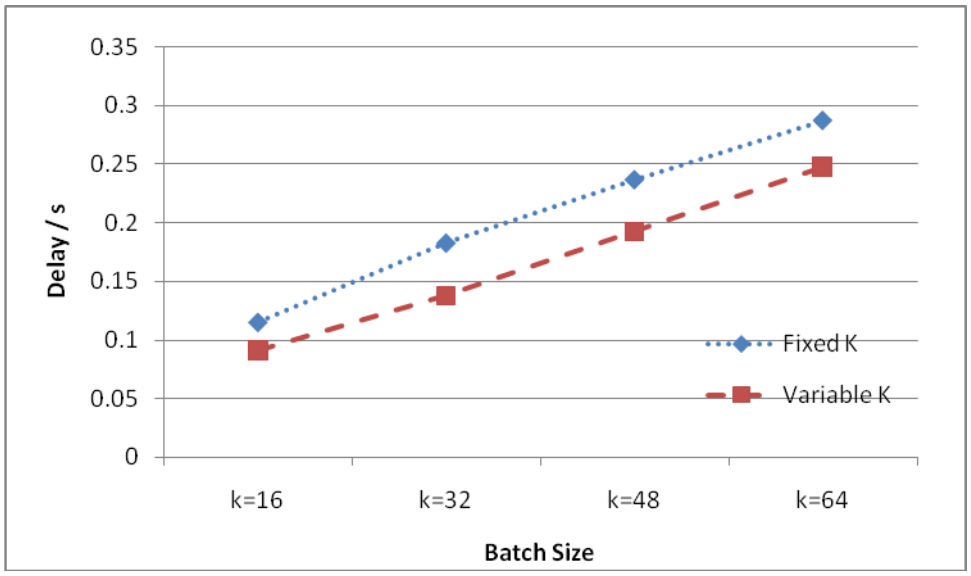

图3.4　night.264，固定K值和可变K值传输延迟比较

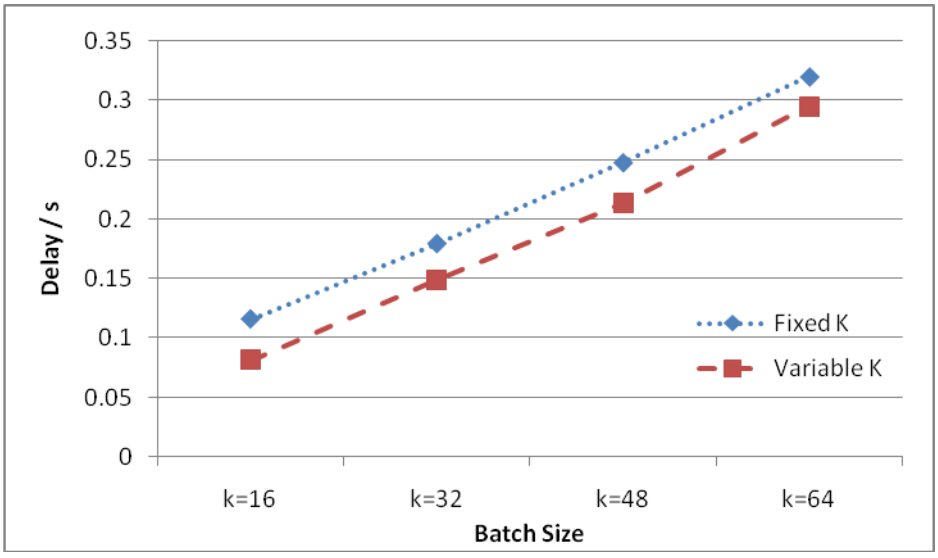

图3.5　foreman_2000.264，固定K值和可变K值传输延迟比较

2)　整个文件的传输吞吐率

对于各种 K 的设置方法，我们比较了视频文件的传输吞吐率，如图 3.6 所示。从图中可以看到，随着包组大小的增大，MORE 协议的传输吞吐率有所增加。结合对于传输延迟和吞吐率的考虑，我们认为，K 取值为 K=var32，K=var48，K=var64 这几种情况较为合理。

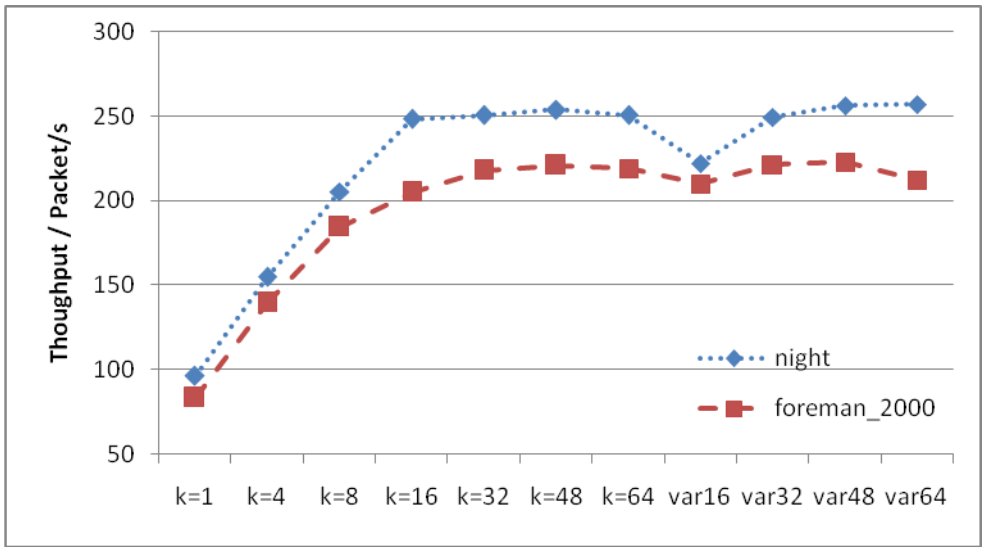

图3.6　K与视频传输吞吐率的关系

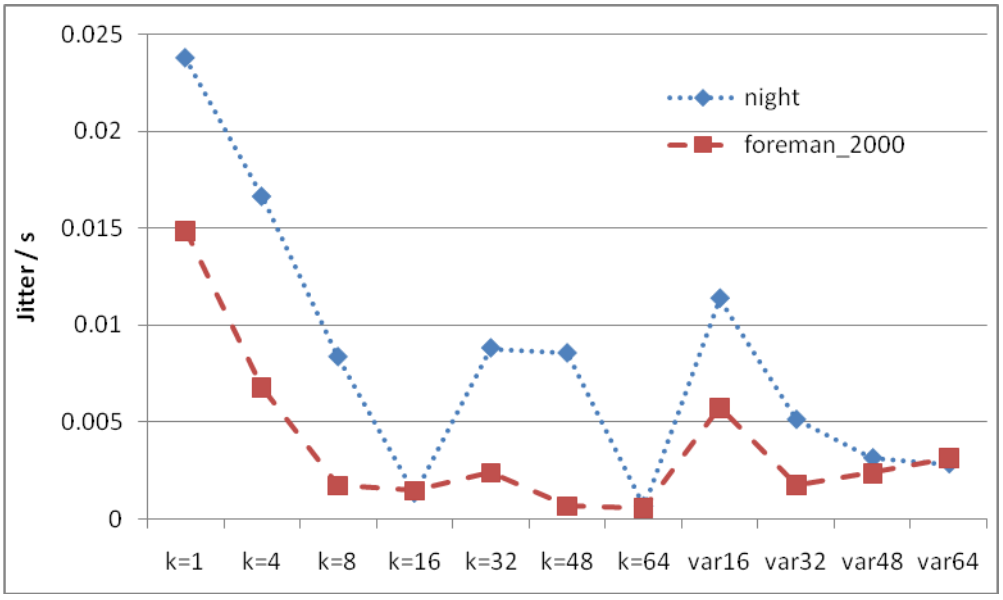

图3.7　K与延迟抖动的关系

3)　延迟抖动

我们对于包组平均延迟抖动的测试结果如图 3.7 所示。可以看到传输延迟的变化较为随机，不具有明显的规律性，这也说明了无线网络传输中的不稳定性。

4)　视频丢帧率

在包组大小较小的时候，由于传输吞吐率较低，丢帧率很高，我们在图 3.8 中，仅给出包组大小在 32 及以上的视频丢帧率测试结果。需要说明的是，在当时设置的网卡功率条件下，MORE 的最大吞吐率（约 250Packets/s）对应的视频码率和 foreman_2000.264 文件接近，而大大低于码率较高的 night.264 文件。从图 3.8 中我们可以看到，foreman_2000.264 的传输最好能够获得接近 10%的丢帧率，而对于 night.264 文件，丢帧率高达 90%以上。

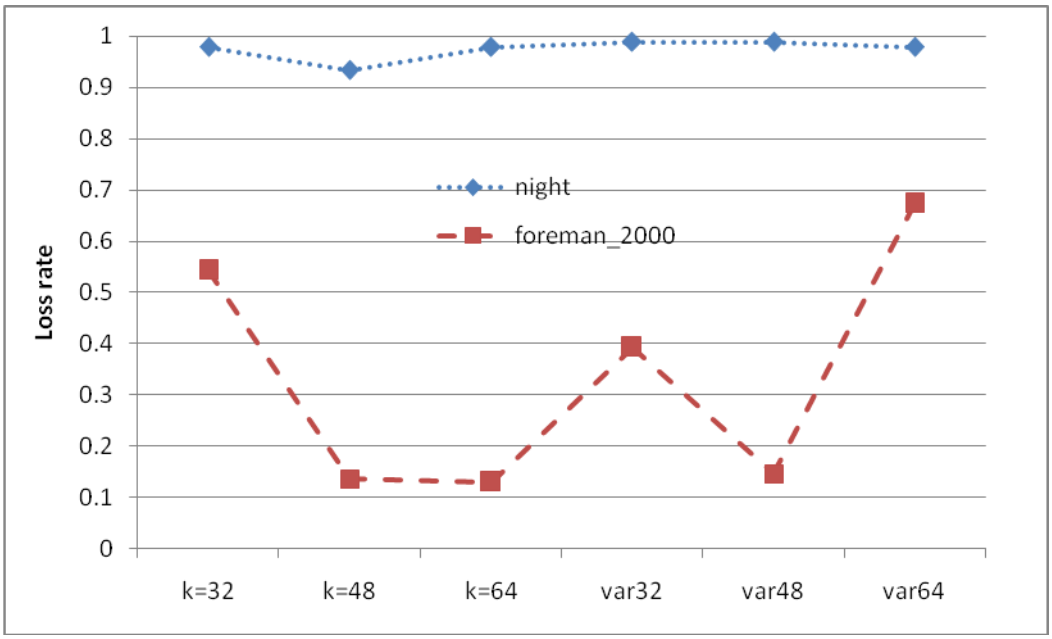

图3.8　K与视频丢帧率的关系

3.3　单流实时播放实验

在单流情况下，我们对于不同码率的视频进行了传输和实时播放测试。测试中，我们使用了不同码率的视频文件，表 3.2 给出了我们使用的视频信息。

表3.2　单流实时播放实验测试视频信息表

视频名称	视频类型	帧数	GOP 长度	码率（Kbps）
foreman_500.264	352x288	300	25	500
foreman_1000.264	352x288	300	25	1000
foreman_2000.264	352x288	300	25	2000
foreman_3000.264	352x288	300	25	3000
foreman_3500.264	352x288	300	25	3500
foreman_4000.264	352x288	300	25	4000

在实验设定的网络状况下，MORE 的传输吞吐量最大约为 350Packets/s。

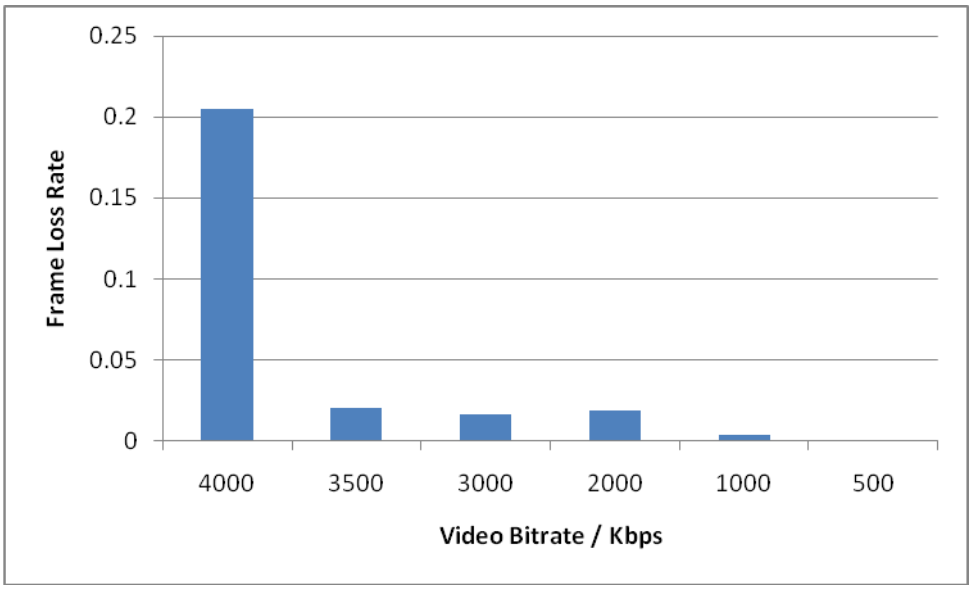

图3.9　视频码率和丢帧率的关系

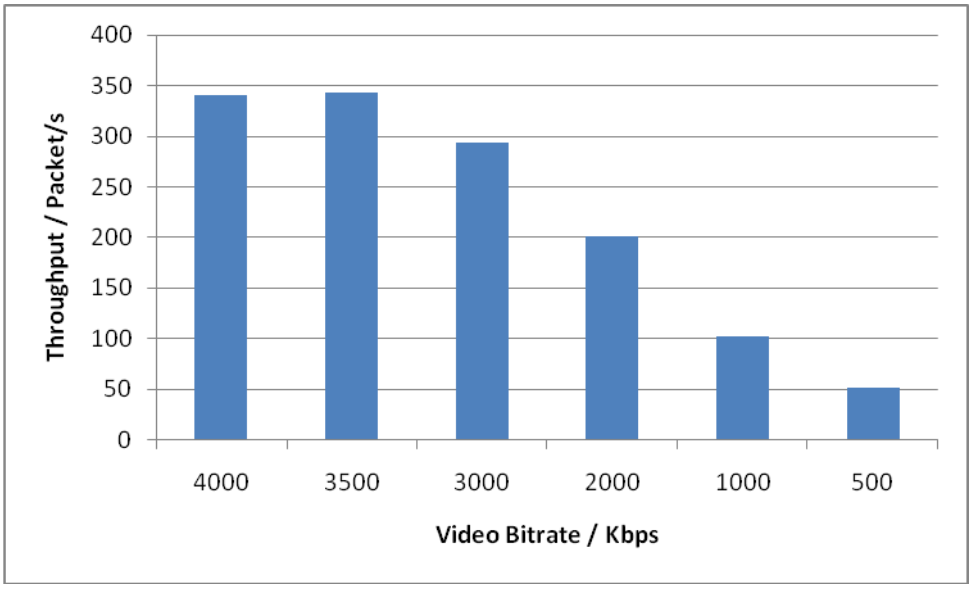

图3.10 视频码率和吞吐率的关系

VLC 播放器中提供了实时查看网络传输码率和丢帧数量的功能。图 3.9 显示了不同码率传输下，丢帧率和码率的关系。图 3.10 统计了不同码率传输线接收节点获得的传输吞吐率情况。在视频码率低于 350Kbps 的情况下，MORE 的吞吐率能够做到与视频码率相当，同时视频播放的丢帧率远不到 5%，能够获得比较理想流畅的播放效果。而当视频码率达到 4000Kbps 时，由于超过了 MORE 传输的吞吐率限制，丢帧率达到了 20%以上，视频在播放中也出现了严重的停滞现象，无法正常观看。

3.4 多流传输性能评价

3.4.1 不同数目背景流对于关键传输参数的影响

为了探究不同数目背景流对于视频传输的关键参数的影响，我们选择两个流作为基础流，并在基础流的基础上逐渐增加背景流的数目。

基础流对应的网络结构如图 3.11 所示。接下来，分别将其他的三个节点也逐个设置为源节点，网络中总的流数目最多达到 5 个，其中三个为背景流，如图 3.12 所示。

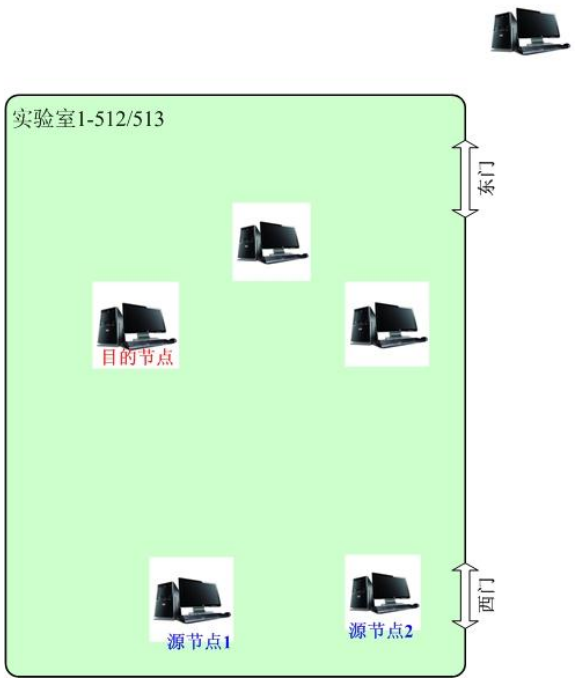

图3.11 网络拓扑图2-1

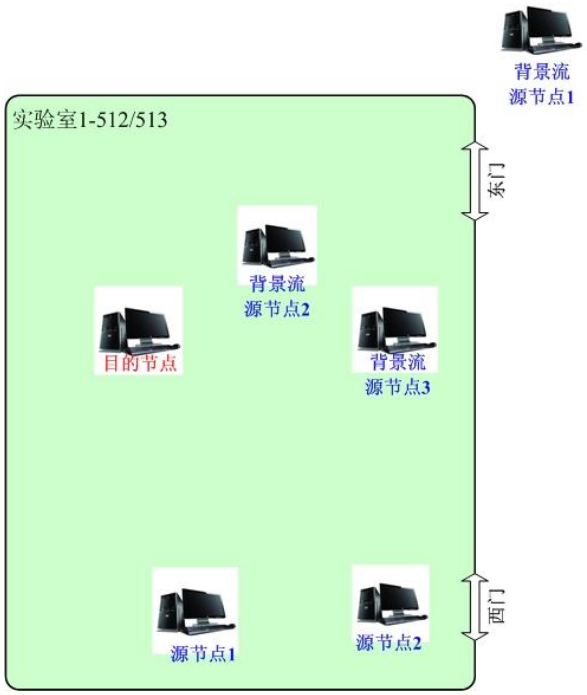

图3.12 网络拓扑图2-2

我们在不同背景流的数目下，测量“源节点 1”和“源节点 2”到目的节点的帧延迟、延迟抖动、吞吐率、包组延迟、包组传输抖动、包组的传输有效性这 6 个参数以及接收节点所有流量总吞吐率，共 7 个参数。

其中包组的传输有效性参数的定义为：包组的实际包含数据包的数目除以包组的总传输次数。进行传输的文件为 night.264（见表 3.1）。

包组的划分方式采用下列 6 种：k=32，k=48，k=64，k=var32，k=var48，k=var64。

我们在各种背景流条件下，对“源节点 1”和“源节点 2”两个流按照 6 种包组划分方法得到的测试参数的实验数据计算平均值，从整体的均值角度比较不同背景流数目对于传输的影响。

源节点 1”和“源节点 2”到接收点的帧延迟、延迟抖动、吞吐率、包组延迟、包组传输抖动、包组的传输有效性参数均值。如表 3.3 所示。

表3.3　不同背景流数目关键参数测试结果表

流数目	帧延迟（s）	帧延迟抖动（s）	包组延迟（s）	数据包传输抖动（s）	传输有效性	吞吐率（Packets/s）
2flows	0.3123	0.3557	0.2742	0.0033	0.9171	171.45
3flows	0.4342	0.5078	0.3819	0.0053	0.9050	123.38
4flows	0.5739	0.6815	0.5031	0.0072	0.8493	93.38
5flows	0.7306	0.8719	0.6421	0.0093	0.8322	73.90

为了按照统一的尺度比较这 6 项参数的变化规律，我们在表 3.3 的每个数值上除以对应列的最大值，将所有数值规范到(0, 1]范围内，得到表 3.4。按照规范化后的参数作图，得到图 3.13。

表3.4　规范化后不同背景流数目关键参数测试结果表

流数目	帧延迟（s）	帧延迟抖动（s）	包组延迟（s）	数据包传输抖动（s）	传输有效性	吞吐率（Packets/s）
2flows	0.4274	0.4080	0.4271	0.3585	1.0000	1.0000
3flows	0.5943	0.5823	0.5947	0.5696	0.9868	0.7196
4flows	0.7854	0.7816	0.7835	0.7818	0.9260	0.5447
5flows	1.0000	1.0000	1.0000	1.0000	0.9073	0.4311

从图 3.13 可以看到，帧和包组的延迟、帧和包组的延迟抖动这四个与传输时间相关的参数随着背景流数目的变化有接近线性增加的关系。而吞吐率和传输有效性则随着流的数目的增加而下降。吞吐率下降的速度略低于线性。

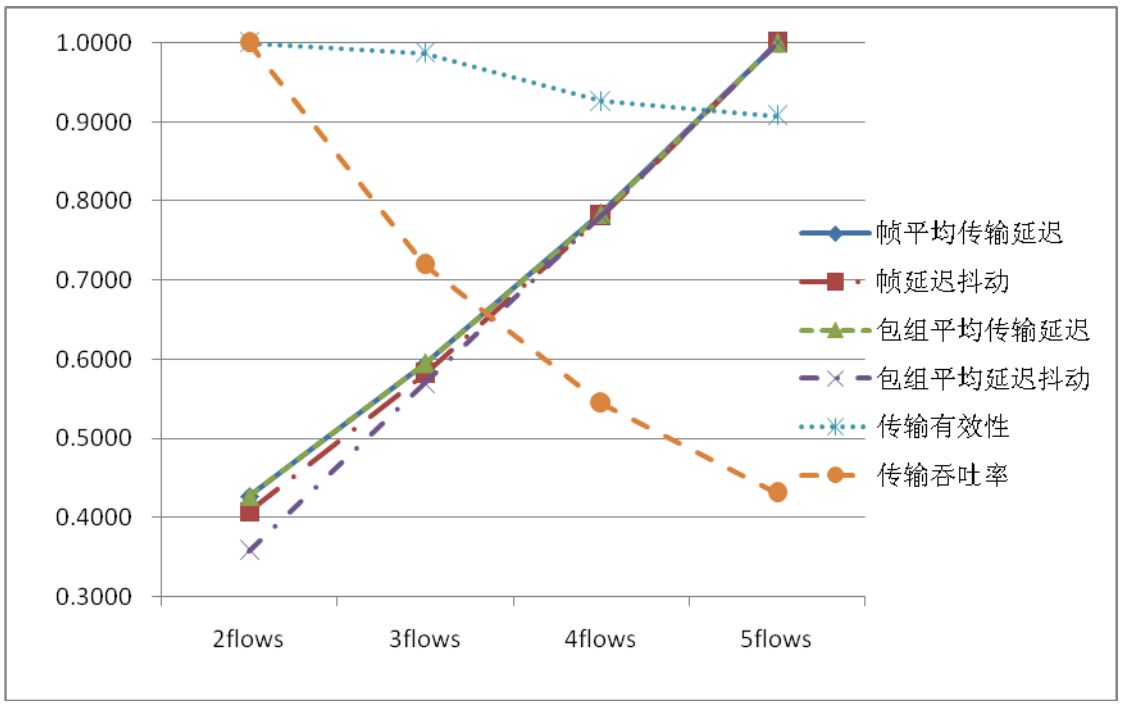

图3.13 背景流数目对于多流传输关键参数的影响

接收点所有流量总吞吐率测试结果如表 3.5 所示。

表3.5 不同背景流数目接收节点总吞吐率测试结果表

包组划分方式	2flows 吞吐率（Packets/s）	3flows 吞吐率（Packets/s）	4flows 吞吐率（Packets/s）	5flows 吞吐率（Packets/s）
32	342.48	351.81	342.1	359.53
48	349.09	361.22	357.36	358.91
64	352.22	363.9	342.01	366.95
var32	325.43	338.63	345.39	351.57
var48	344.75	355.99	342.47	365.98
var64	352.61	359.85	349.18	362.8

平均值	344.43	355.23	346.42	360.96
最大值	352.61	363.9	357.36	366.95
最小值	325.43	338.63	342.01	351.57

我们在每种背景流条件下计算了吞吐率的平均值，最大值和最小值。按照流的数目和吞吐率均值作图，得到图 3.14。

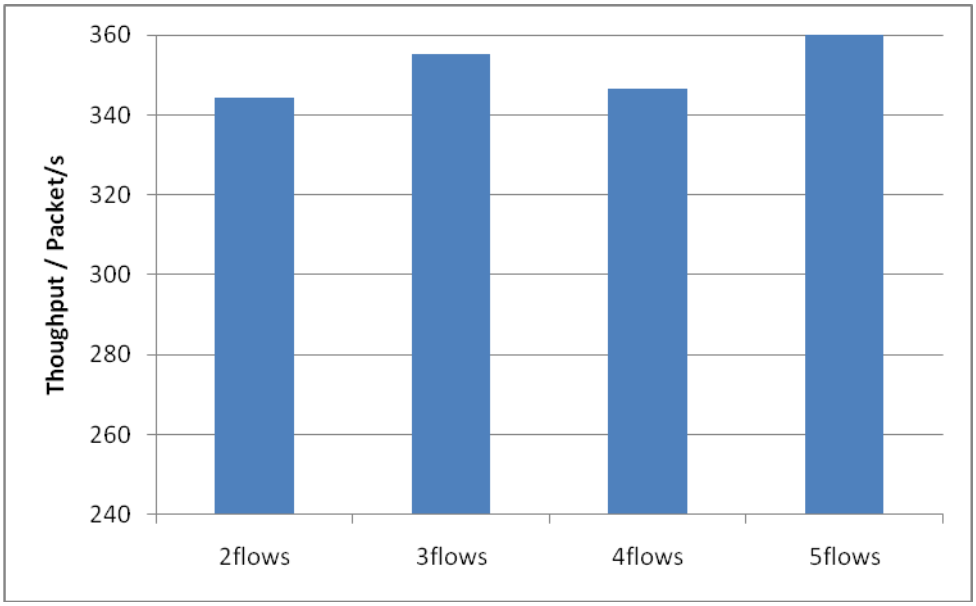

图3.14 不同背景流条件下接收节点总吞吐率比较

从图 3.14 中我们可以看到，随着背景流数目的增加，接收点的总吞吐量基本不变，随着流数目的增长略有增加。因此，可以认为，传输的瓶颈主要在于接收节点，接收节点的吞吐率决定了能够支持的流的数目。

3.4.2 不同网络情况对传输吞吐率的影响

前文我们提到，通过设置网卡的传输功率，我们可以获得不同的节点间连接概率。在传输速率恒定为 5.5Mbps 的情况下，节点间的连接概率越大，成功发送包组所需的传输次数越少，从而能够获得更大的吞吐率传输效果。

我们在实验中，对三种不同的网卡传输功率的情况下进行了实验，由于源节点和接收节点分别是数据包组和 ACK 的发送发起者，我们在进行链接概率调节

的时候主要通过控制源节点和目的节点的无线网卡传输功率实现。三种网络情况对应的网卡功率如表 3.6 所示：

表3.6　不同传输功率条件下网卡功率设置表

网络状况评价	源节点和接收节点传输功率（mW）	转发节点传输功率（mW）
良好（high）	16	16
一般（mid）	4	16
较差（low）	2	8

我们采用两个源节点进行传输，网络拓扑结构如图 3.15。进行传输的文件为 night.264（见表 3.1）。包组的划分方式为下列 6 种：k=32，k=48，k=64，k=var32，k=var48，k=var64。

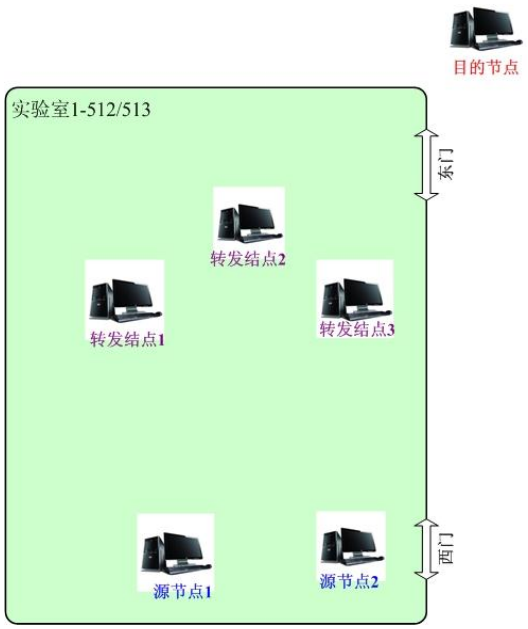

图3.15　网络拓扑图3

我们在不同网络连接条件下，测量“源节点 1”和“源节点 2”在接收节点的总吞吐率参数。不同网络连接条件下，总吞吐率结果如表 3.7 所示。

表3.7　不同传输功率条件测试结果表

包组划分方式	high 条件下吞吐率（Kbps）	mid 条件下吞吐率（Kbps）	low 条件下吞吐率（Kbps）
32	342.48	264.06	189.37
48	349.09	255.31	217.46
64	352.22	298.89	203.83
var32	325.43	239.96	236.18
var48	344.75	284.32	213.45
var64	352.61	307.3	294.31
平均值	344.43	274.97	225.77
最大值	352.61	307.3	294.31
最小值	325.43	239.96	203.83

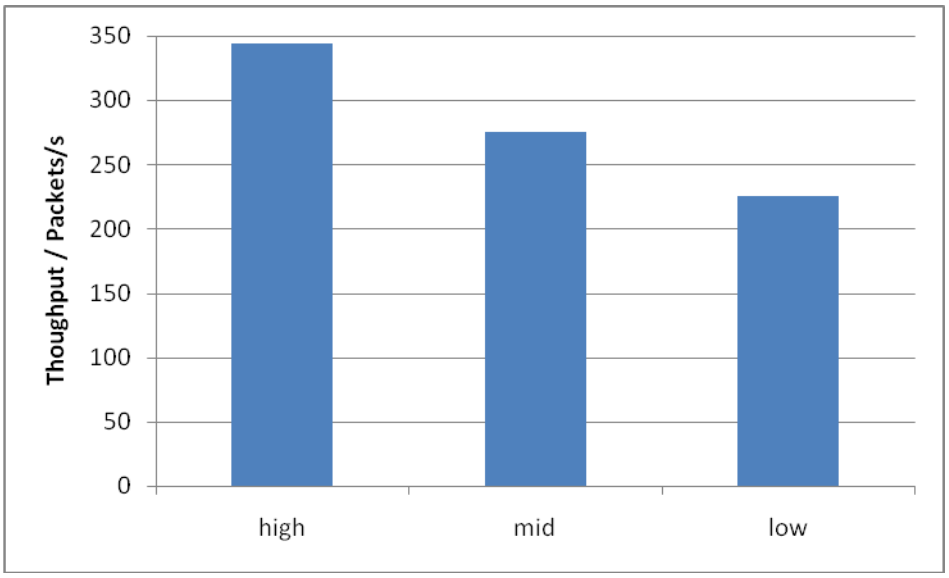

图3.16　不同网络传输条件对总吞吐率的影响

按照每种网络条件下各种包组划分方式的平均值作图，得到图 3.16。从图中可以看出，网络连接情况对于传输效果有较为明显的影响。但是限于我们的研究范围，在本文中不会对无线网卡的传输功率对传输吞吐率的影响做定量分析。今

后在没有特殊说明的情况下，我们的实验均在网络情况良好（high）的情况下进行。

3.5 多流调度算法对传输性能的影响

3.5.1 三种调度算法简介

1) 简单轮转调度法（noschd）

在这种方法中，节点对于经过的所有流，按照顺序选择流进行传输。如果选择的流不具备发送的条件，例如传输的阈值低于 1，或者没有足够的包组能够形成发送 ACK 的条件则该轮的发送就会无功而返。

这种方法实现简单，但是有可能会出现轮空的情况。

2) 优化轮转调度法（count）

这种方法对简单轮转调度法进行了改进，按照流编号顺序依次选择，直到找到一个具备发送条件的流为止。

3) 序列号优先级调度法（prio）

这种方法在流的调度上按照已经经过节点的包组的序列号升序的顺序进行查找，也就是说，成功发送包组的数目少的流的优先级更高。按照这个顺序，直到找到一个具备发送条件的流为止。

3.5.2 实验结果

进行测试的网络结构采用图 3.15 所示的网络拓扑图 3。进行传输的文件为 night.264（表 3.1）。

表3.8 不同调度算法测试结果表

网络条件	调度算法	总吞吐率 (Packet / s)	吞吐率标准差 (Packet / s)
high	noschd	332.8	5.43
high	count	342.4	4.98
high	prio	343.68	4.60
mid	noschd	2.95	1.39
mid	count	197.91	18.80
mid	prio	203.32	0.35

包组的划分方式为：k=64。在网络传输条件良好（high）和一般（mid）两种网络条件下进行了测试。

在每种网络条件下，对于每种调度方法我们进行 4 次测试，统计每个流的在相同时间内（high link 为 25 秒，mid link 为 65 秒）的传输吞吐率均值，并计算所有流传输吞吐率的标准差。流之间吞吐率标准差，可以反应不同流之间传输速率不平衡程度，从而比较各种调度方法对于流传输的公平性的影响。

每种调度方法下，总吞吐率和每个流吞吐率标准差如表 3.8 所示。从表中可以看出，在良好的网络传输条件下，三种调度方法在吞吐率和吞吐率标准差方面差别不大，但是在中等的网络连接条件下，简单轮转调度法因为网络的不稳定和拥塞，基本上不能够正常传输，而优化轮转调度法在总的吞吐率方面和序列号优先级调度法差别不大，但是序列号优先级调度法更好的平衡了不同流之间的流量，流的吞吐量标准差远小于优化轮转调度法的标准差。

3.6 不同码率文件传输效果的比较

在之前的测试中，我们采用的视频文件多数是 night.264（表 3.1），这个文件的码率在 3500Kbps 左右，而在良好网络条件下，目的节点的最大接收吞吐率为 350Packets/s 左右，大约对应 3600Kbps 的视频码率，在多个源节点情况下，这样码率的文件难以做到实时传输。我们在接下来的测试中，将对更低码率的视频进行测试，并着重分析传输的实时性效果，为设计改进实时性的传输算法提供参考。

3.6.1 网络条件

我们将在两种网络拓扑结构中进行测试，第一种网络结构包含两个源节点，同图 3.15 所示的网络拓扑图 3。

第二种网络结构包含三个源节点，如图 3.17 所示：

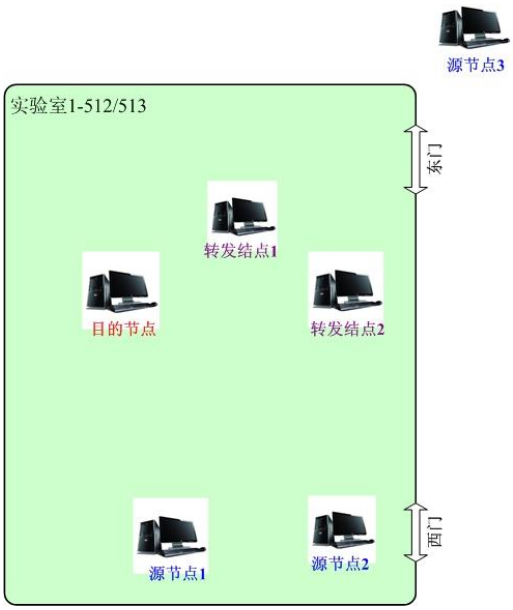

图3.17 网络拓扑图4

针对两种不同的网络拓扑结构，我们在中等（mid）和良好（high）两种网络条件下进行了测试，其中：对于两个源点的情况，在中等和良好传输条件下测试；对于三个源点的情况，只在良好传输条件下的测试。

3.6.2 实验测试和分析

测试文件 foreman_1000.264 和 foreman_2000.264。表 3.9 列出了测试文件的相关信息。

表3.9 不同码率测试文件信息表

视频名称	视频类型	帧数	GOP 长度	时长（s）	码率（Kbps）	平均帧数据报数
foreman_1000.264	352x288	300	25	12	1000	9.0
foreman_2000.264	352x288	300	25	12	2000	5.2

对于每一种文件、流的数目、网络设置的组合，我们采用一个三元组标记为：（文件名、流数量，网络情况），例如 foreman_1000.264 文件在中等网络条件下，对 2 个数据流传输的情况，标记为（foreman_1000，2flows，mid）。这样，我们分别有六种测试情况：

1) （foreman_1000，2flows，mid）
2) （foreman_1000，2flows，high）
3) （foreman_2000，2flows，mid）
4) （foreman_2000，2flows，high）
5) （foreman_1000，3flows，high）
6) （foreman_2000，3flows，high）

在之前的实验中，我们可以看到，在良好传输条件下，在接收点的最大接收吞吐率为 3600Kbps 左右。对于两个源节点，应该能满足码率为 1000Kbps 左右的 foreman_1000.264 的实时传输，但是在传输码率约为 2000Kbps 的 foreman_2000.264 的时候可能会出现帧不能按时到达的情况。在三个源节点的情况下，在高传输功率的情况下应基本能支持 foreman_1000.264 的传输，但是对于 foreman_2000.264 的传输会出现非常严重的帧不能按时到达的现象。

包组的划分方式为以下六种：k=32，k=48，k=64，k=var32，k=var48，k=var64。在实验结果的表示中，我们将区别不同流的对各种包组划分方式进行标记，标记方式为：按照包组划分方式和流的序号进行编号。例如，第二个流，按照 k=32 划分包组，标记为 K=32_2。又比如，第三个流，按照 k=var64 划分，标记为 var64_3。

实验中，我们测试了对于每个源点的如下 5 个参数：到接收点的帧延迟、吞吐率、包组延迟、包组中每个数据包的平均延迟抖动和丢帧率。同时，着重分析在相对低吞吐率的网络环境的视频丢帧率关系，为进一步设计传输实时性算法提供实验依据。

由于测试情况较多，在这里，以（foreman_1000，2flows，mid）的传输效果为例，给出实验结果，帧延迟、吞吐率、包组延迟、包组中每个数据包的平均延迟抖动等 4 个传输参数的实验结果如表 3.10 所示。

分别对于这四个参数作图，我们可以得到图 3.18、图 3.19、图 3.20、图 3.21。

从这些图中我们可以看到，对于多流中的某一个流，在帧延迟、包组延迟、传输吞吐率方面，包组的划分方式带来的参数变化趋势在多流方面和之前获得的单流的结论是一致的。但是和单流传输中看不到的是，在多流条件下，每个流的性能趋势可能不尽相同，例如在上面的情况中，flow1 的传输吞吐率大于 flow2；类似的，flow1 在帧和 Batch 的延迟方面小于 flow2。但是 flow1 在数据包的平均传输抖动方面大于 flow2。

表3.10 不同码率文件测试文件表

流和包组划分标识	帧延迟（s）	吞吐率（Packets/s）	包组延迟（s）	数据包延迟抖动（s）
K=32_1	0.284	123.242	0.249	0.005
K=48_1	0.402	128.249	0.365	0.013
K=64_1	0.506	130.555	0.476	0.004
Var32_1	0.231	133.269	0.230	0.005
Var48_1	0.321	147.059	0.317	0.006
Var64_1	0.441	140.279	0.442	0.006
K=32_2	0.277	126.314	0.244	0.003
K=48_2	0.454	112.896	0.417	0.002
K=64_2	0.567	116.648	0.534	0.003
Var32_2	0.243	127.741	0.241	0.003
Var48_2	0.388	123.256	0.382	0.006
Var64_2	0.474	133.470	0.466	0.006

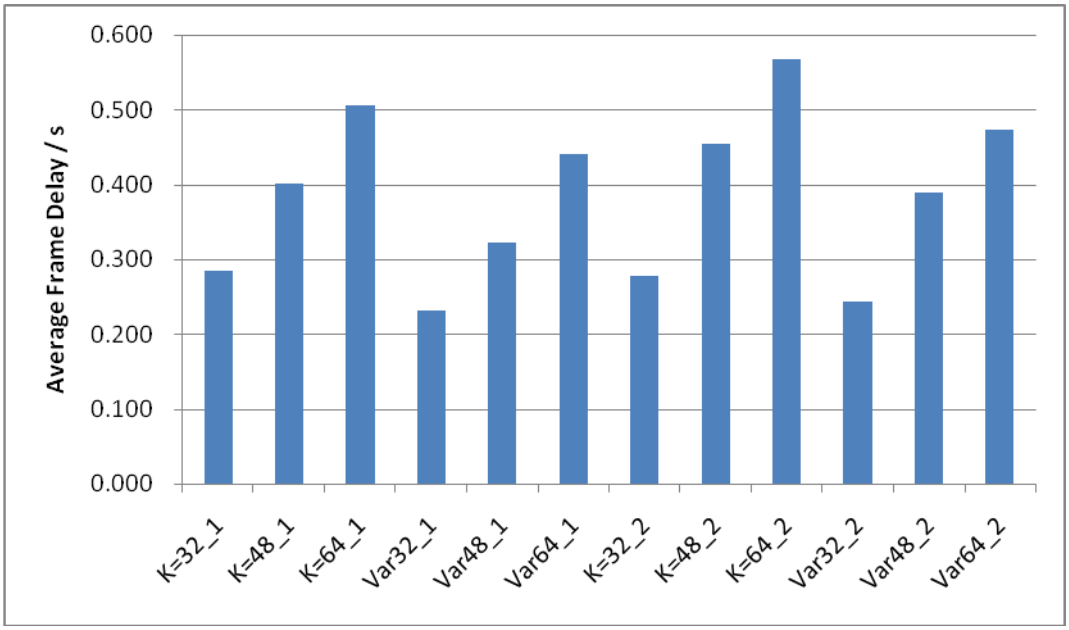

图3.18 foreman_1000.264平均帧延迟

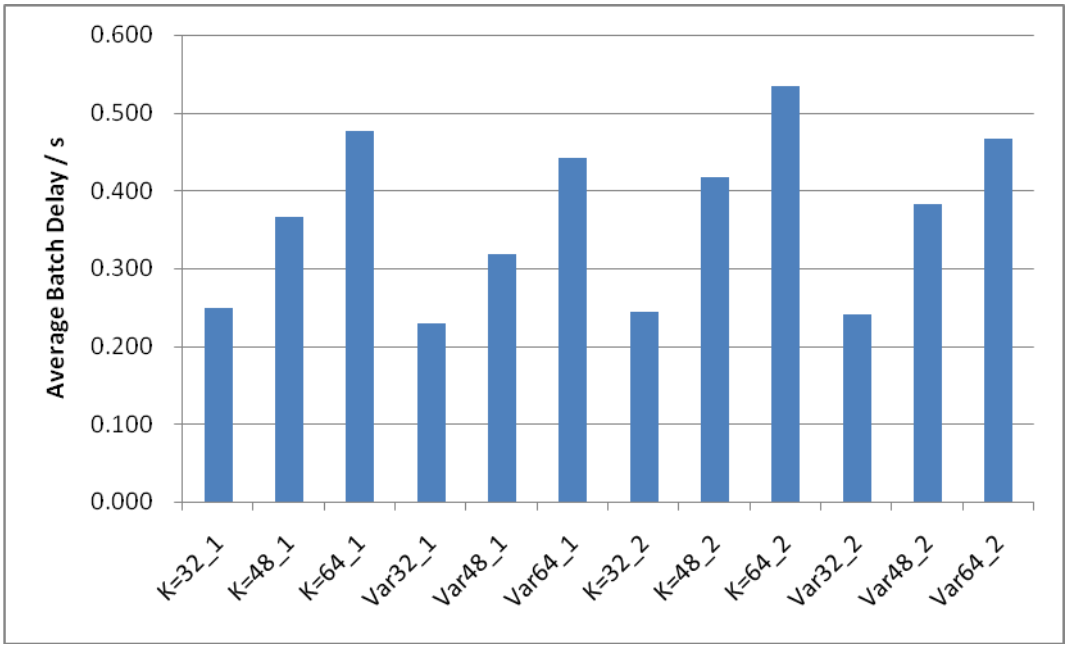

图3.19 foreman_1000.264平均包组延迟

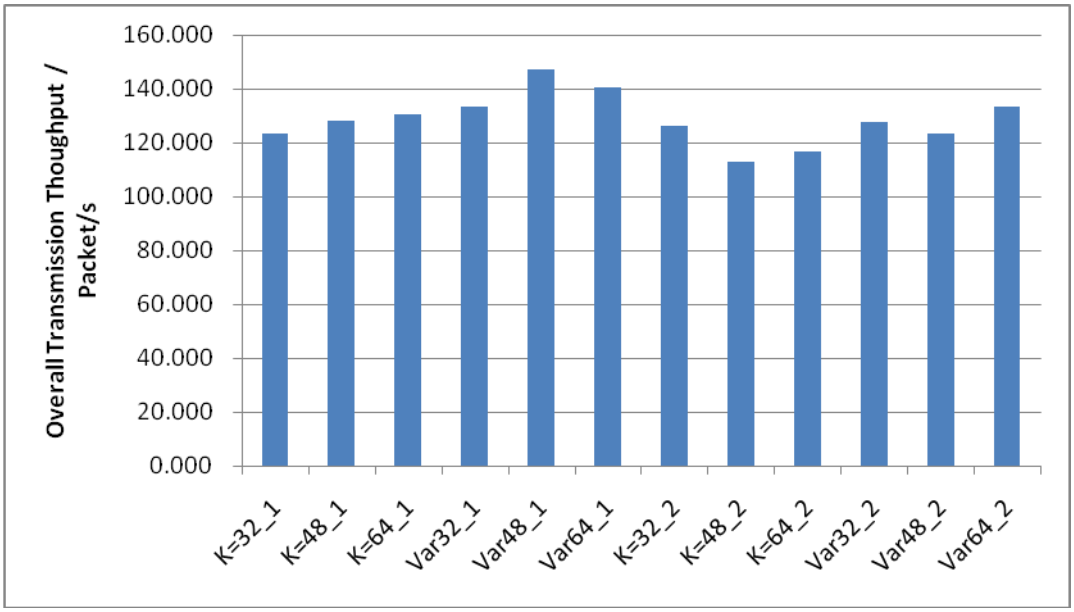

图3.20 foreman_1000.264接收节点总吞吐率

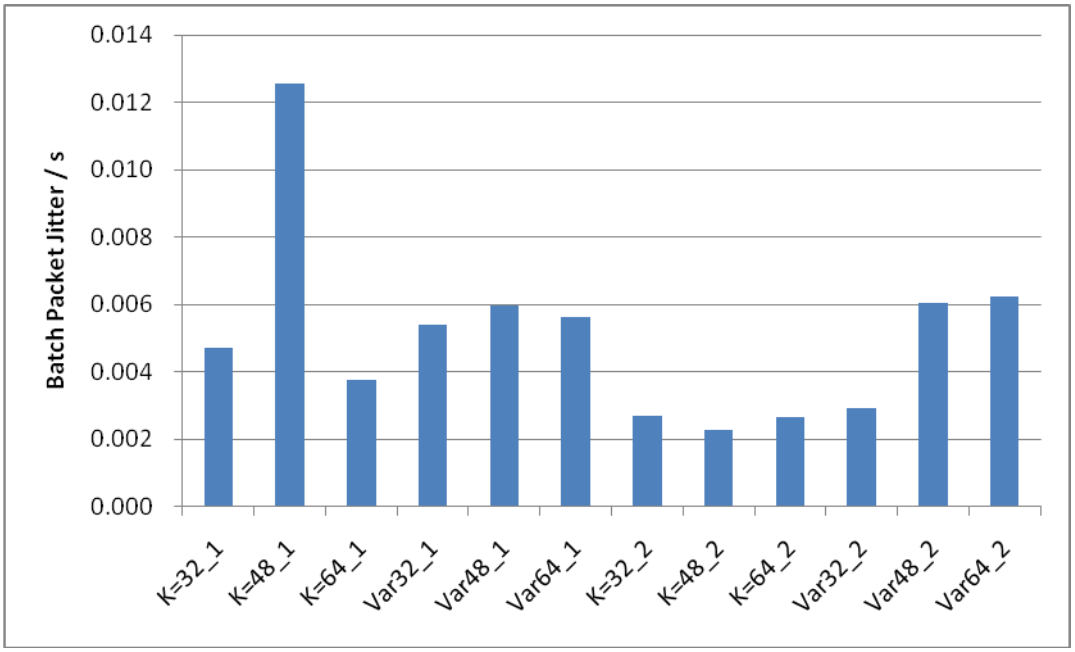

图3.21 foreman_1000.264数据包平均延迟抖动

为了衡量不同码率的视频流的实时性传输效果，我们引入了“理想丢帧率”的概念。

未在视频播放时间之前传到的帧，我们认为该帧被丢弃（Loss）。假设文件的实际传输时间为T_a，视频时长为T_0，则在理论上，可以获得的理想丢帧率，记为L_{ideal}，实际丢帧率记为L_{actual}。

则有：

$$L_{ideal}=\frac{T_a-T_0}{T_a}=1-\frac{T_0}{T_a} \tag{3-3}$$

这相当于如果我们丢弃那些部分超时帧，使得剩余帧的传输时间和视频时长相等，可以认为其余帧能在截止时间之内传到。这是一个比较粗略的分析模型。我们按照上述公式，将所有六种测试组合和包组划分组合中的丢帧率L_{actual}和L_{ideal}绘制在图 3.22 中。

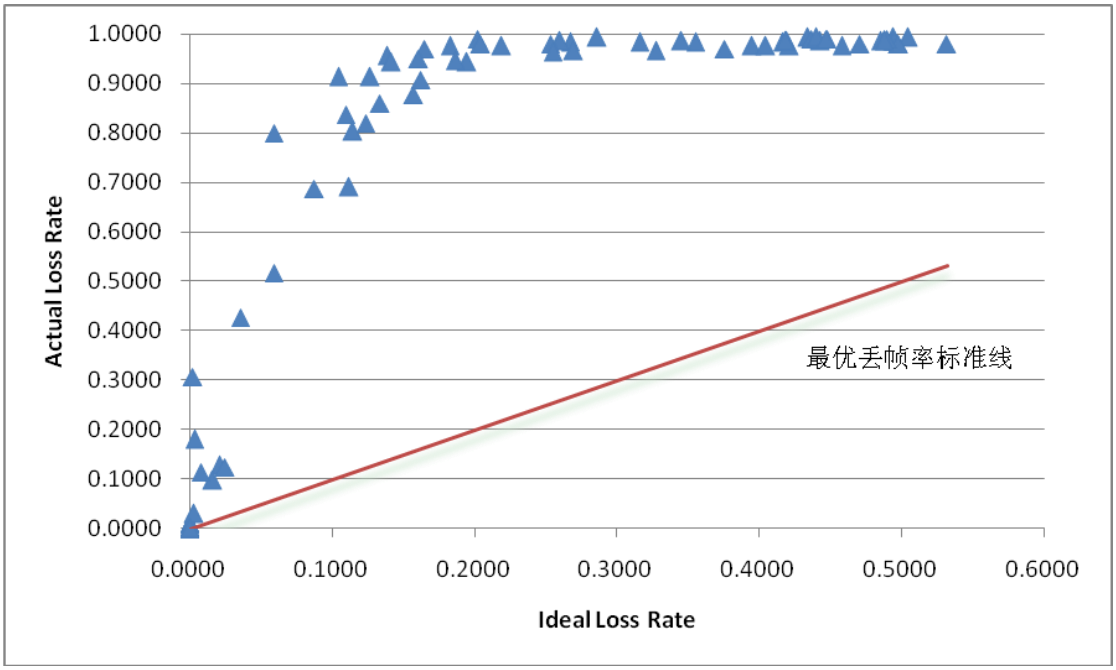

图3.22 实际丢帧率和理想丢帧率关系

图 3.22 中的红线是理想情况下能够期望得到的丢帧率，蓝点为实际丢帧率。从图上可以看到，实际的丢帧率远远大于理想情况，丢帧率在多数情况下接近 100%。这样的丢帧率情况远远不能满足我们对于视频流的实时性要求。在实际传输码率小于视频码率的情况下，我们仍然希望通过设计一种自主控制的丢帧算法，降低视频传输的实际丢帧率，使得实际丢帧率接近理论上的最优丢帧率。

3.7 多流实时播放实验

和单流的实验类似，我们同样在多流的情境下进行了视频实时播放测试。测试中，我们采用了三个源节点。源节点的分布和图 3.17 所示的网络拓扑图 4 相同。

在测试中，我们采用码率约为 1000Kbps 的《Yes Man》电影视频作为传输视频流。每次测试时间设定为 120s，进行了 6 次实验，在接收点的传输吞吐率见表 3.11。

随着视频的码率波动，不同测试时间得到的吞吐率有所变化。测试中，最大传输吞吐率达到了 342.59Packets/s。这和我们之前对于文件传输最大能够获得 350Packets/s 左右的传输吞吐率的结果相符。从播放的效果来看，对于码率

1000Kbps 左右的实验视频，播放较为流畅，偶尔在播放中会出现一定的跳帧、画面短时间停滞的现象。

表3.11 多流实时播放吞吐率表

测试编号	1	2	3	4	5	6
Thoughput（Packets/s）	266.43	302.63	311.21	342.59	332.57	287.7

3.8 本章小结

本章对 MORE 协议的视频传输性能进行了多方面的评测，并通过传输系统的实时播放功能，较为直观的评价了视频实时播放的功能。本章中，为了衡量传输的实时性，提出了理想丢包率的概念。实验表明，直接使用 MORE 协议进行视频传输在实时性效果上仍不够理想，视频的传输质量有一定的提升空间。

第4章 源节点自适应丢帧算法 SAFDA 的设计

和普通数据的传输不同，视频数据具有实时性的要求。假如某一帧数据在播放时间之后才完成传输，那么我们认为该帧被丢弃。为此，我们需要设计一种丢帧控制的算法，提高视频传输的实时性质量。在本章中，我们将首先介绍根据视频截止时间控制实时性质量的 TOR 算法，并由此提出基于 MORE 协议的源节点自适应丢帧算法 SAFDA 的设计和算法复杂度分析。

4.1 视频数据包截止时间和 TOR 算法简介

[8]中提出了和 GOP、帧的类型相关的数据包传输截止时间。对视频的相关参数有如下定义：λ 表示两个帧之间的传输间隔。α 表示一个 GOP 中包含的帧的数量。$F_{i,j}$ 表示第 i 个 GOP 的第 j 个帧。$P_{i,j}^{k}$ 表示 $F_{i,j}$ 的第 k 个数据包。

定义 $M(F_{i,j})$ 为与 $F_{i,j}$ 相关的所有帧的个数，称之为 $F_{i,j}$ 影响到的帧。例如 I 帧的影响大于 P 帧，大于 B 帧。定义 $R(P_{i,j}^{k})$ 为 $F_{i,j}$ 影响到帧所包含的时间段，则有：

$$R(P_{i,j}^{k}) = \lambda(M(F_{i,j}) + 1) \qquad (4\text{-}1)$$

这个时间段包括了影响到的帧和本身的帧的包含的时间段。

通过以上定义，作者给出了传输截止时间 $D(P_{i,j}^{k})$ 的计算方法：

$$D(P_{i,j}^{k}) = \Delta + \lambda(\alpha(i-1) + (j-1)) + R(P_{i,j}^{k}) \qquad (4\text{-}2)$$

Δ 是调节截止时间控制而引入的一个变量。从以上公式可以看到，一个帧的所有数据包的传输截止时间都是相同的。

除此之外，作者还假设发送端到接收端的延迟为一个 GOP 的时间，而在传输的过程中，缓存的大小也为一个 GOP。

如果当前时间小于截止时间 $D(P_{i,j}^{k})$，对应的包会被进行发送和重发。

[7]在[8]的基础上，基于 ExOR 机会路由协议，结合视频实时传输的情景，提出了基于截止时间的机会路由传输协议 TOR（Time-aware Opportunistic Relay scheme）。TOR 通过限制转发节点的重传时间，控制重传的次数，达到提高传输吞吐率的目的。数据包中写入截止时间 $D(P_{i,j}^{k})$，在中间节点，根据数据包的截止时间（packet_deadline）计算节点转发截止时间。转发截止时间（relay_deadline）可用如下的公式计算：

$$\text{relay\_deadline} = \text{packet\_recv\_time} + \text{time\_to\_relay} - \text{duration} \quad (4\text{-}3)$$

$$\text{time\_to\_relay} = \text{packet\_deadline} - \text{current\_time} \quad (4\text{-}4)$$

其中，packet_deadline 为数据包中写入的截止时间。duration 为发送数据包需要的时间。如果超过了转发截止时间，网状网络中的节点直接将其丢弃。

4.2 SAFDA 算法概述

TOR 算法和 ExOR 类似，是数据包粒度上的机会路由协议。TOR 考虑了视频数据包的截止时间，网络节点数据包转发的控制按照转发截止时间计算，某个数据包的丢失，会影响该数据包对应的帧。

MORE 协议的传输以包组为单位，一个包组中可以包含多个帧的数据。这一特点给我们的实时性控制算法的设计带来了两方面的困难。一方面给包组截止时间的设定带来了困难，同一个包组中的数据包将可能具有不同的截止时间。另一方面，包组作为传输单位，在包组的传输完成之前，接收节点收到的数据是不能解压出原始数据包的。如果我们在包组传输完成之前就停止包组的传输和转发，会影响包组中的所有帧，使得其中的多个帧都被迫丢弃。

为了适应 MORE 协议的传输特点，我们设计了一种基于网络速率预测和帧截止时间的源节点传输码率自适应的丢帧算法 SAFDA（**S**ource **A**daptive **F**rame **D**iscarding **A**lgorithm）。SAFDA 算法主要有五个方面的设计要点：

1) 未完成发送的包组不提前丢弃。
2) 帧的截止时间的控制在源节点进行。

源节点对历史发送的包组传输延迟的记忆，实时估计当前的网络传输速率并下一个包组的可能传输时间，同时根据帧的截止时间判断该帧是否可以进行传输，被判超时的帧在源点即进行丢弃。

3) 源节点丢帧算法考虑不同类型帧的优先级，尽可能保留 I 帧。

4) 转发节点按照每个流的传输信用（TX_credit）控制包组的转发次数。

5) 接收节点设置一定大小的数据包缓存

4.3 SAFDA 算法的设计

4.3.1 一些基本的变量定义和假设

和[8]中的标记一致，我们采用 λ 表示两个帧之间的传输间隔； α 表示一个 GOP 中包含的帧的数量； $F_{i,j}$ 表示第 i 个 GOP 的第 j 个帧； $P_{i,j}^{k}$ 表示 $F_{i,j}$ 的第 k 个数据包。另外， B_i 表示第 i 个包组。

假设一个 GOP 对应一秒钟的视频长度。每个 GOP 中，第一个为 I 帧，之后的 $\alpha-1$ 个为 P 帧。

4.3.2 源节点

在源节点，我们尽可能保持发送的码率和视频的平均码率一致。第 i 个包组的发送，在收到 B_{i-1} 的 ACK 后开始。

源节点对网络传输速率的预测通过保留前几个包组的传输时间的方式实现。在传输开始前，源节点先发送 n 个空包组，记录发送的总时间 T_{nB}^{0} 和数据包数目 P_{nB}^{0}。传输开始后，保留前 n 个包组发送总时间 T_{nB}^{i} 以及这 n 个包组的总数据包个数 P_{nB}^{i}。包组 B_i（i≥1）中数据包的选择，依据数据包的截止时间，选择方法如下。

第一步：计算数据包的截止时间。

假设在接收节点进行缓存处理，缓存的长度为 $N_{buf}\alpha$，即包含 N_{buf} 个 GOP 的数据，给传输增加的延迟为 $N_{buf}\alpha\cdot\lambda$。收到多出缓存的部分，才开始播放视频，视频开始部分缓存的帧数的截止时间应为无穷大。

在上如缓存的设置下， $F_{i,j}$ 中的各个包 $P_{i,j}^{k}$ 的相对截止时间为：

$$D(P_{i,j}^{k})=\begin{cases}N_{buf}\alpha\cdot\lambda+\lambda(\alpha(i-1)+(j-1)) & ,i>N_{buf}+1\\ \infty & ,i<=N_{buf}+1\end{cases} \quad (4\text{-}5)$$

每一帧中数据包的绝对截止时间t_D可以表示为视频开始传输时间（t_{start}）和相对截止时间$D(P_{i,j}^k)$的和：

$$t_D = t_{start} + D(P_{i,j}^k) \tag{4-6}$$

也就是说，在$(N_{buf}\alpha+1)\cdot\lambda$时间范围内的帧没有截止时间，这些帧一定会等待传到。缓存范围之后的帧按照公式进行截止时间的计算，数据包的截止时间将写入数据报头。

第二步：根据截止时间判断是否丢帧。

首先我们可以根据保存的历史状态对当前的传输速率进行估计，设定的包组数据包期望数目记为P_B，则当前包组B_i的延迟估计为：

$$\frac{P_B T_{nB}^{i-1}}{P_{nB}^{i-1}} \tag{4-7}$$

当前时间为t_{now}，数据包的截止时间为t_D，如果这一帧是 P 帧，则当满足下列条件时，选取该数据包作为包组B_i中的一部分：

$$t_{now} + \frac{P_B T_{nB}^{i-1}}{P_{nB}^{i-1}} <= t_D + \Delta t \tag{4-8}$$

如果这一帧是 I 帧，为了尽可能的保证 I 帧能够传输，源节点将丢弃 I 帧后面的若干个 P 帧。丢弃的 P 帧的数量为：

$$N_{discard} = \frac{t_{now} + \frac{P_B T_{nB}^{i-1}}{P_{nB}^{i-1}} - (t_D + \Delta t)}{\lambda} \tag{4-9}$$

上式中，Δt是调节变量，控制截止时间计算的严格程度。我们将在实验中对该参数的不同设置进行评测。

4.3.3 转发节点

转发节点对于包组的发送次数，按照 MORE 协议中设置传输信用的方式（TX_credit）[3]确定。传输信用描述了对数据包转发次数的限制，可以按照启发式的方法计算。

假设网络中共有 N 个节点，对任意两个节点 i 和 j，我们用 $i < j$ 表示节点 i 到接收节点的距离小于节点 j，所谓距离就是传输到接收节点所需的重传次数的期望，表示为 ETX。z_i 表示转发节点 i 对数据包的转发次数。e_{ji} 为节点 j 到节点 i 的丢包概率，我们可以在传输开始前提前测定。

我们集中考虑一个数据包的发送过程。对于转发节点 j，从 ETX 更大的节点收到数据包的次数为：

$$\sum_{i>j} z_i(1-e_{ij}) \tag{4-10}$$

此时节点 j 需要转发该数据包，当且仅当没有比节点 j 具有更小 ETX 值的节点收到该数据包，出现的概率是 $\prod_{k<j} e_{ik}$。那么，我们节点 j 需要发送数据包的数量（记为 L_j），计算如下：

$$L_j = \sum_{i>j}(z_i(1-e_{ij})\prod_{k<j} e_{ik}) \tag{4-11}$$

可以注意到，初始条件下，源节点的 L 值应该为 1。

当 ETX 小于节点 j 的节点收到数据包后，节点 j 就可以停止传输，因此，每一次传输后，节点 j 将停止传输的概率为 $1-\prod_{k<j} e_{jk}$。从公式 4-8，我们知道了节点 j 需要转发的数据包数据，那么节点 j 的期望转发次数可以计算如下：

$$z_j = \frac{L_j}{1-\prod_{k<j} e_{jk}} \tag{4-12}$$

通过公式 4-8 和 4-9，我们可以而从源节点开始，采用递推的方法计算网络中所有节点的期望转发次数 z。

那么每当节点 i 收到一个数据包后，增加的传输信用 TX_credit，计算方法为：

$$TX\_credit_i = \frac{z_i}{\sum_{j>i} z_j (1 - e_{ji})} \quad (4\text{-}13)$$

通过传输信用，我们无需全局调度就可以实现控制转发次数的目的。在多个流的情况下，每个流在初始阶段分别计算传输信用，设置节点中的相应参数。并在每个流的传输中，分别采用各自的传输信用计算转发节点转发次数。

4.3.4 接收节点

由于视频的帧丢弃在源节点处理，目的节点只需要按顺序处理接收到的包组和帧，并按照帧中给定的顺序进行解码。如果出现丢帧情况，该帧图像使用前一帧的数据填补。

同时，接收节点维护一个数据包缓存，缓存中保留帧的数目为：

$$N_{buf}\alpha \quad (4\text{-}14)$$

在接收到的数据包超出缓存的范围后，开始进行视频数据的解码播放。

4.4 SAFDA 算法复杂度和算法中参数的设定

4.4.1 算法时间复杂度

从上述算法描述，我们可以看到，在源节点中，只需要对截止时间进行简单的计算，时间复杂度为常数级。在转发节点，需要提前按照迭代的算法计算每个节点的传输时间，假设网络节点的数目为 N，按照 MORE[3]中给出的结果，计算复杂度为 $O(N^2)$，对于多流的情况，假设流的数量为 F，则计算复杂度为 $O(FN^2)$。

4.4.2 算法空间复杂度

算法对空间的额外需要来源于两个方面，一方面是源节点在估计当前网络速率时，保留 n 个包组的历史延迟数据。另一方面，是接收节点开辟的缓存空间，大小为 $N_{buf}\alpha$。

4.4.3 算法中参数的设定

在 SAFDA 算法中，主要有三个参数：

1) 在进行数据截止时间判断是用到的参数 Δt

我们进行截止时间计算的公式会对帧是否丢弃的判断起到极其关键的作用，需要使用这一变量对时间判断的严格程度进行控制。一般来说，Δt 越大，对于丢弃的时间判断越宽松。需要注意的是，Δt 的取值可以为负值。

2) 接收节点保留的缓存的 GOP 的长度 N_{buf}

直观来看，缓存的长度越长，丢帧的概率越小。因为即使传输速率不足，提前发送到缓存中数据包，可以在一定程度上弥补。

3) 估计当前网络状态时使用的包组数目 n

n 的取值，反映对网络情况的估计。n 的取值越大，在估计当前传输速率时考虑的历史传输情况越多，在网络变化较为平稳的情况下，可能会更为准确。n 的取值越小，在估计当前传输速率时，考虑的历史传输情况越少。

我们将在下一章对这些参数对于算法的影响进行实验评测。

4.5 本章小结

在本章中，我们根据 MORE 协议的特点，结合视频的截止时间，提出了基于网络速率预测的源节点传输码率自适应的丢帧控制算法 SAFDA。本章给出了 SAFDA 算法在源节点、接收结点、转发结点计算和处理流程的形式化描述，并分析了算法的复杂度。由于采用了源节点主导的方式，SAFDA 算法具有良好的可扩展性。

第5章 SAFDA 算法的性能评测

本章中，我们将通过实验对 SAFDA 算法进行评测。为了在实验测试和比较中具有统一的网络速率数据，我们采用了模拟测试的方式。在我们模拟测试中，将采用第 3 章介绍的实验中得到的包组延迟数据，模拟每个包组的网络传输延迟。

在我们的实验中，我们将主要回答如下两个问题：

第一：SAFDA 算法是否能使得视频传输的丢帧率和视频质量得到提高，能得到多大程度的提高。

第二：在我们的算法中，使用的三个参数 Δt 、 N_{buf} 、 n 的取值如何影响算法的性能。

5.1 SAFDA 算法对于视频实时传输质量的提升

5.1.1 单流传输评测

我们选取了 foreman_2000.264（见表 3.9）作为测试视频文件：

模拟传输速率采用单流测试实验中上述视频的包组传输延迟数据。测试视频的码率约为 2000Kbps，当时的网络传输速率与此相当。但由于网络的不稳定，在某一时刻后，大部分的帧都出现了晚于截止时间到达的情况。

包组的划分采用三种不同的方式：k=var32，k=var48，k=var64

在实验中，我们对于使用和不用 SAFDA 算法的两种情况进行了模拟，SAFDA 算法的参数设置为：$\Delta t = 0$ 、 $N_{buf} = 0$ 、 $n = 1$ 。可以注意到，缓存长度为 0，即除了加入网络预测和丢包处理之外，其他参数和不用 SAFDA 算法完全一致。

264 文件经过传输后，保留按时到达的帧，未能按时到达的帧采用之前的帧数据补全。最后比较传输后视频解码后的 yuv 与原始 yuv 文件的 PSNR。在结果的表示中，对于没有采用丢帧算法的结果，我们标记为 Old，采用了丢帧算法的结果，标记为 SAFDA。

实验结果如表 5.1 所示。

表5.1　SAFDA算法单流测试结果表

算法名称	var 32 情况下 PSNR（dB）	var 48 情况下 PSNR（dB）	var 64 情况下 PSNR（dB）
Old	25.47	39.31	21.26
SAFDA	39.34	42.44	36.26
提高	13.87	3.13	15.00

从上表可以得到图 5.1。

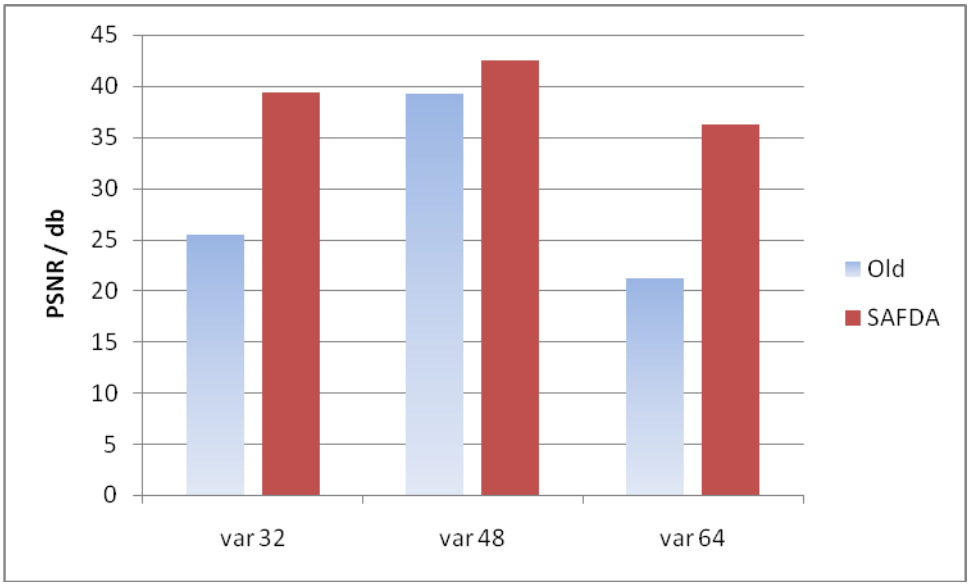

图5.1　采用SAFDA算法后视频PSNR的提高

从图 5.1 中，我们可以看到，采用 SAFDA 算法，可以改善接收视频的 PSNR。在等情况下，采用了 SAFDA 算法得到了最多达 15dB 的质量提升（包组划分为 var64 的情况）。图 5.2 是在 var64 情况下的视频质量对比。左边的两个截图是 SAFDA 的结果，右边的截图是 Old 的结果。上面两个图是 foreman_2000 序列的第 100 帧，可以看到，SAFDA 的效果明显好于 Old；下面两个图是序列的第 250 帧，不采用 SAFDA 算法的情况下，视频最后部分的帧都被丢掉，在 SAFDA 算法下仍可以保证有部分的帧按时传到。

图5.2　foreman_2000传输效果比较

5.1.2 多流传输评测

对于多流的评测，我们采用本文 3.6 节中的六种多流情况下实验中得到的延迟结果，作为模拟数据，在三种包组划分方式中，选择了其中一种 var64。SAFDA 算法的参数设置为：$\Delta t = 0$、$N_{buf} = 0$、$n = 1$。我们将比较使用 SAFDA 算法前后的丢帧率以及理想丢帧率L_{ideal}的关系，L_{ideal}的计算按照前文公式的定义计算。实验结果如图 5.3 所示。

图中的紫色点表示采用 SAFDA 算法后视频的丢帧率，蓝点表示未采用 SAFDA 算法的丢帧率结果。而红线表示按照理想丢帧率关系计算的丢帧率结果。从图上我们可以看到，使用 SAFDA 算法后，视频的丢帧率能够接近或者达到理想丢帧率。因为理想丢帧率计算过程中直接使用了实数的时间，而丢帧率的计算按照帧为单位，因此，甚至出现了实际丢帧率低于同样情况下理想丢帧率的数据点。丢帧算法的使用，使得每个流的传输质量都得到了很大的提高。

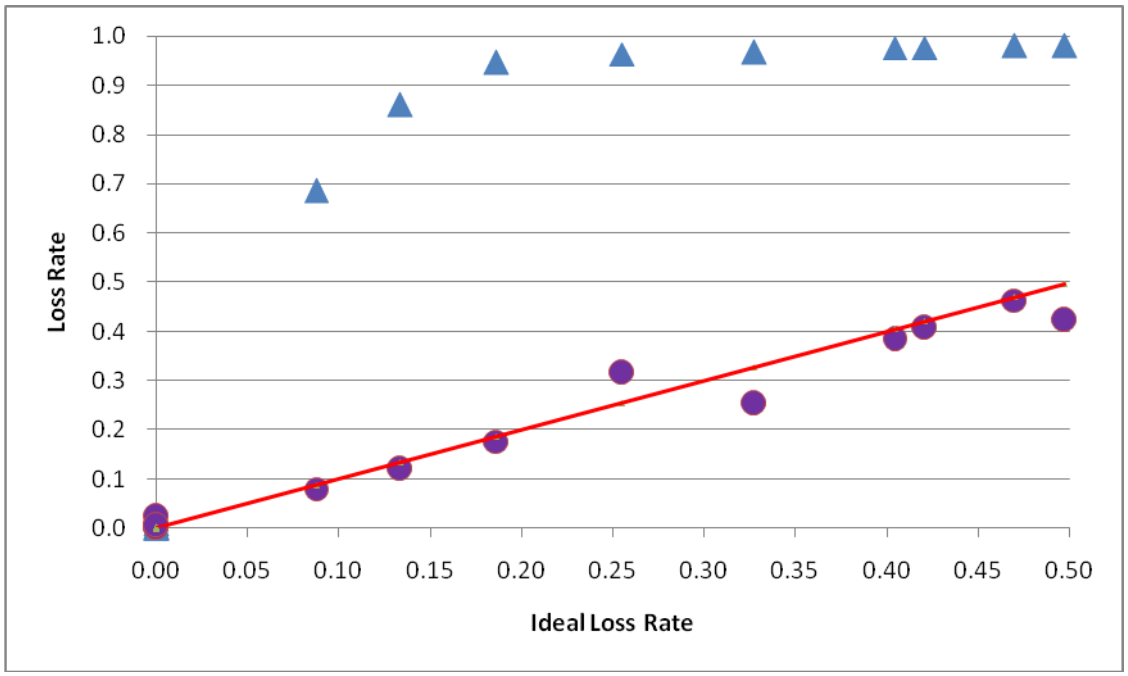

图5.3　采用SAFDA算法前后丢帧率和理想丢帧率的关系

5.2　Δt 的设置对于传输效果的影响

在进行丢帧判断条件的计算中，Δt 是我们用以条件丢帧条件紧密程度的变量，在计算公式中，与缓冲区的长度 N_{buf} 相关，因此，我们将这个两个变量结合起来进行测试

我们仍然采用与 5.1 节中相同的测试条件和模拟传输数据。

对于 N_{buf} 的取值，我们取为从 0 到 0.6，每隔 0.1 取一个值，共 7 个数值。

对于每一个 N_{buf} 取值，Δt 从-1.00~1.00 间隔 0.01 取值，共 201 个数值。n 的取值固定为 $n=1$。我们对 N_{buf} 和 Δt 的组合情况，测量视频丢帧率 P_{loss}，按照前文的设定，一旦帧不能按时到达，我们就认为该帧被丢弃。

测试结果如图 5.4 所示，不同颜色的曲线代表不同 N_{buf} 的结果。

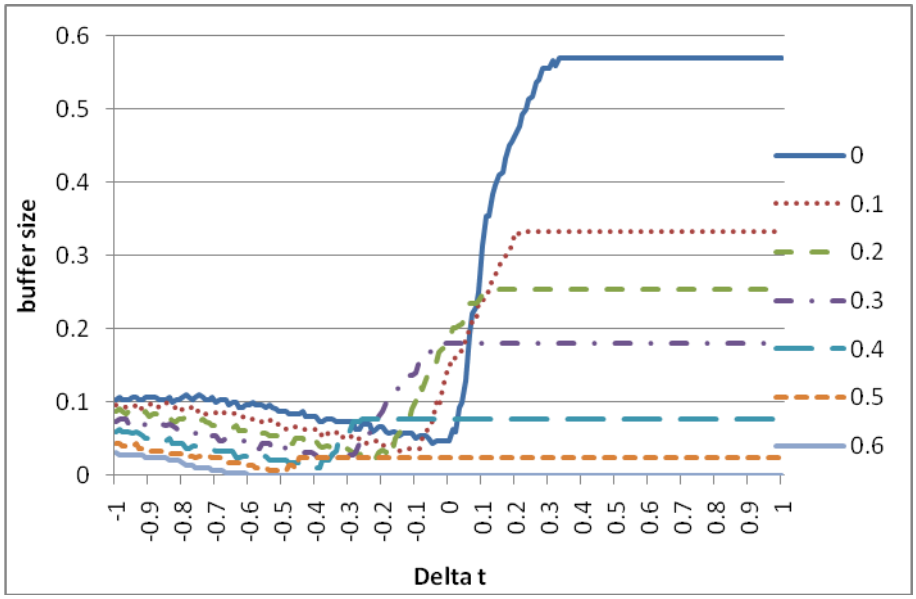

图5.4　N_{buf} 和 Δt 对于丢帧率的影响

图 5.4 中，最低点的数值就是获得最低丢帧率时对应的 Δt 的取值，我们将这些取值提取出来，可以得到下表所示的 N_{buf} 和 Δt 的对应关系。

表5.2　N_{buf} 和 Δt 的对应关系表

N_{buf}	0	0.1	0.2	0.3	0.4	0.5	0.6
Δt （s）	-0.05	-0.15	-0.25	-0.39	-0.4	-0.54	-0.6

对以上数据点进行线性拟合，得到图 5.5。

从图 5.5 中可以看到，　N_{buf} 和 Δt 呈现很高的线性相关关系，大致的方程为：

$$N_{buf} = -\Delta t \qquad (5\text{-}1)$$

我们最初的丢弃判断时间公式（4-3）（4-4）（4-5）在我们的情境中 $\alpha \cdot \lambda = 1$，因此，我们的延迟判断公式（4-5）可以进行简化为：

$$t_{now} + \frac{P_B T_{nB}^{i-1}}{P_{nB}^{i-1}} <= t_D \qquad (5\text{-}2)$$

同时， $D(P_{i,j}^k)$ 的计算公式（4-3），可以相应简化为：

$$D(P_{i,j}^k)=\begin{cases}\lambda(\alpha(i-1)+(j-1)) & ,i>N_{buf}+1\\ \infty & ,i<=N_{buf}+1\end{cases} \quad (5\text{-}3)$$

公式说明，我们对于传输延迟的判断，可以不同考虑接收节点的缓存长度。今后如不作特殊说明，在对 SAFDA 算法进行模拟测试时，我们将采用上述简化公式进行截止时间计算和丢帧判断。

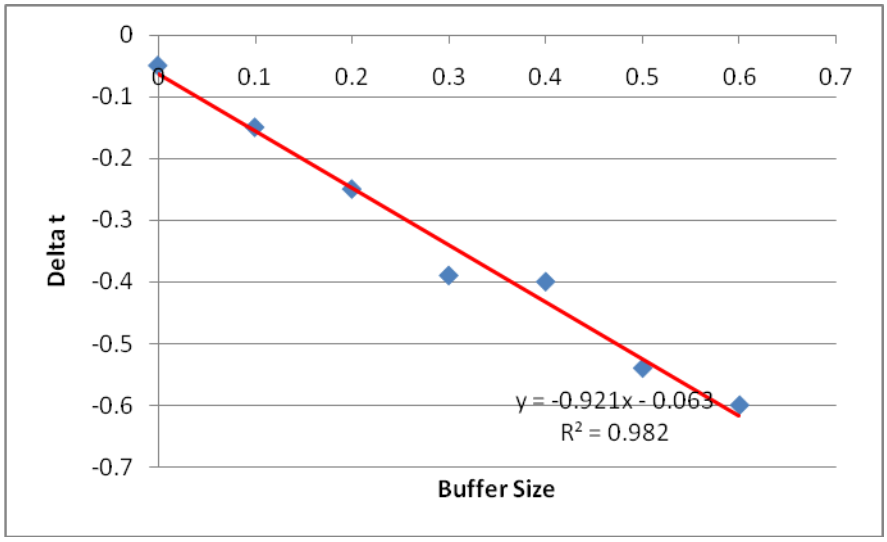

图5.5　N_{buf} 和取得最小丢帧率的 Δt 的关系

5.3 N_{buf} 的设置对于传输效果的影响

我们采用第本文 3.6 节中的多流测试中的两组情况的包组延迟作为模拟数据：（foreman_2000，3flows，high）以及（foreman_2000，2flows，mid）。包组划分方式为 k=var64。

为了评价 N_{buf} 对于每一个流的影响，我们将 N_{buf} 取值从 0 到 2，每隔 0.2 取一个值，共 11 个取值。在 N_{buf} 进行变化的同时，固定 n 的取值为 1，即，使用前一个包组的传输速率估计当前的网络传输速率。对传输后的视频，我们计算视频的丢帧率。

数据组合为（foreman_2000，3flows，high）时，三个流的丢帧率变化如图5.6所示，数据组合为（foreman_2000，2flows，mid）时，两个流的丢帧率变化如图5.7所示。

从图5.6和图5.7可以看出，随着N_{buf}的增长，丢帧率具有线性下降趋势。通过拟合数据点，我们发现各个流的丢帧率下降斜率大致相等，均在0.013~0.016左右。

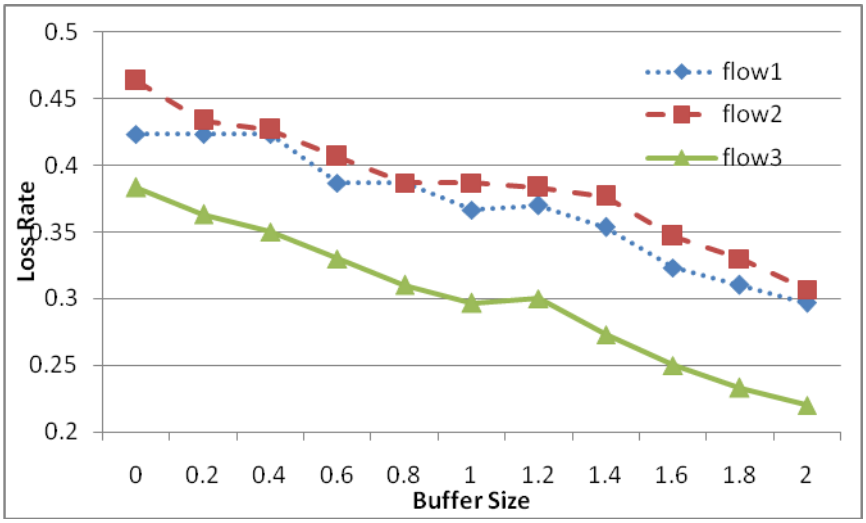

图5.6 （foreman_2000，3flows，high），丢包率与N_{buf}关系

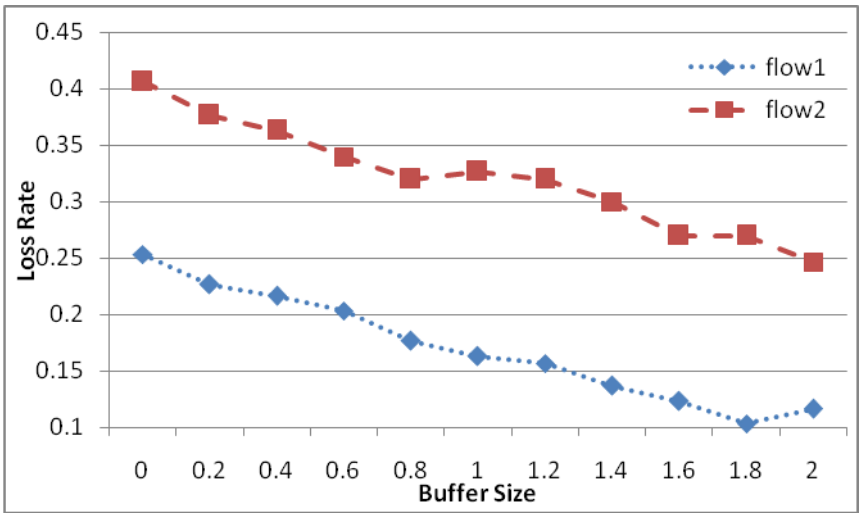

图5.7 （foreman_2000，2flows，mid），丢包率与N_{buf}关系

5.4 n 的设置对于传输效果的影响

n 的取值大小，将影响我们对当前网络条件的估计。在对 n 的实验中，我们仍然采用 3.6 节的多流测试中的包组延迟作为模拟数据。包组划分方式为 k=var64。其中（foreman_1000，2flows，high）由于传输码率大于视频码率的关系，丢帧率为 0，因此，我们考察了其他五种情况：

1) （foreman_1000，2flows，mid）
2) （foreman_2000，2flows，mid）
3) （foreman_2000，2flows，high）
4) （foreman_1000，3flows，high）
5) （foreman_2000，3flows，high）

为了全面的评价 n 在不同 N_{buf} 设置下对于丢帧率的影响，N_{buf} 取值从 0 到 2，每隔 0.2 取一个值，共 11 个取值。n 的取值在 1 和 8 之间的整数值变化。测试时采用 N_{buf} 和 n 的取值组合。

以（foreman_2000，2flows，mid）和（foreman_2000，3flows，high）为例，我们在图 5.8 到图 5.12 中给出各个流的视频丢帧率随着 n 的变化规律。

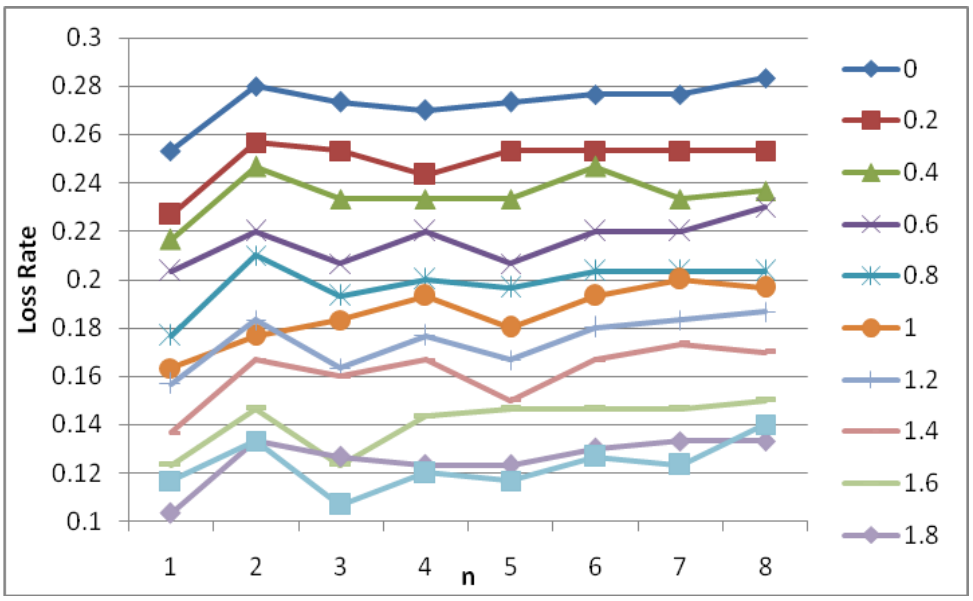

图5.8 （foreman_2000，2flows，mid）Flow1

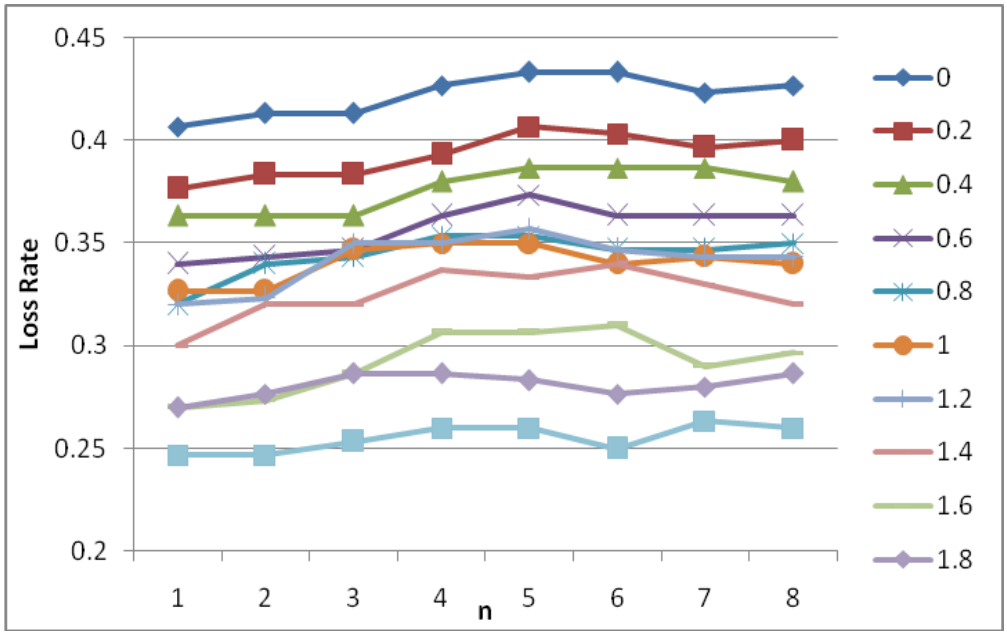

图5.9 （foreman_2000，2flows，mid）Flow2

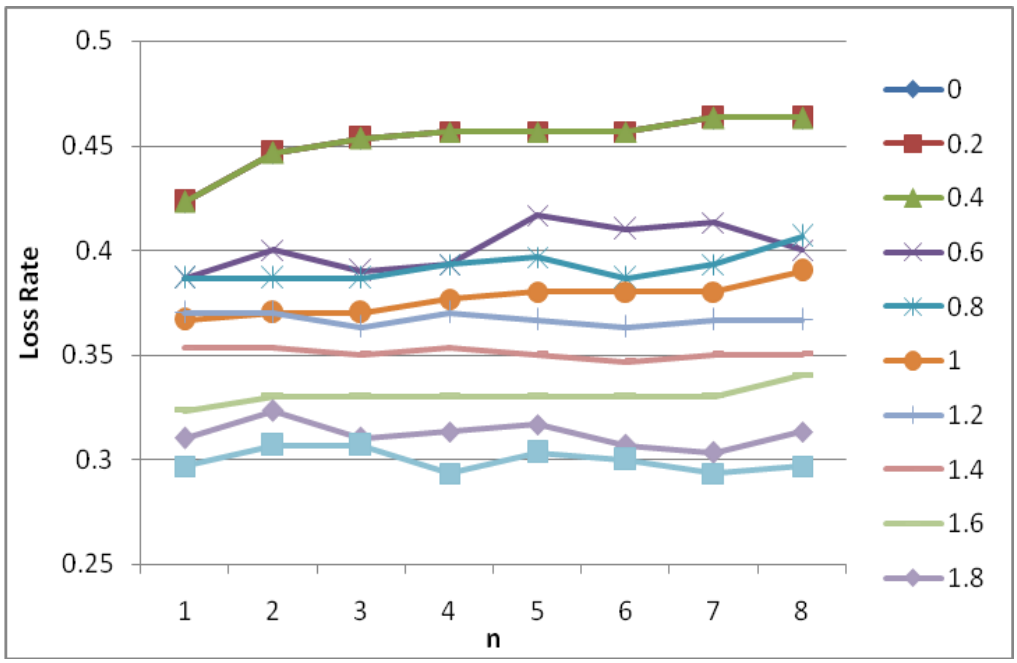

图5.10（foreman_2000，3flows，high）Flow1

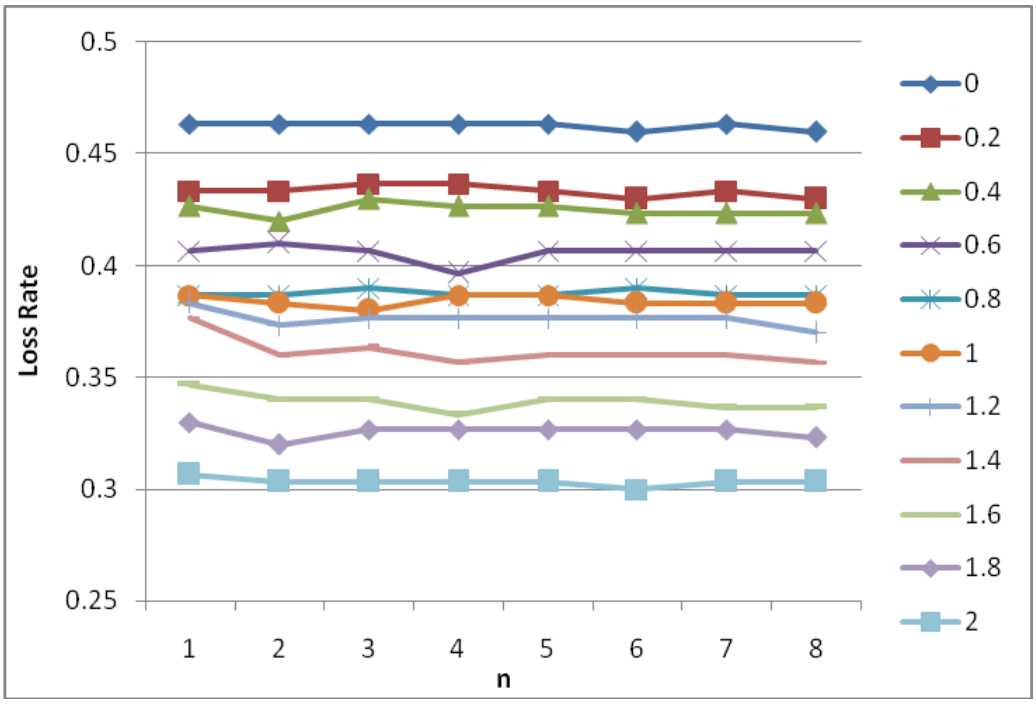

图5.11（foreman_2000，3flows，high）Flow2

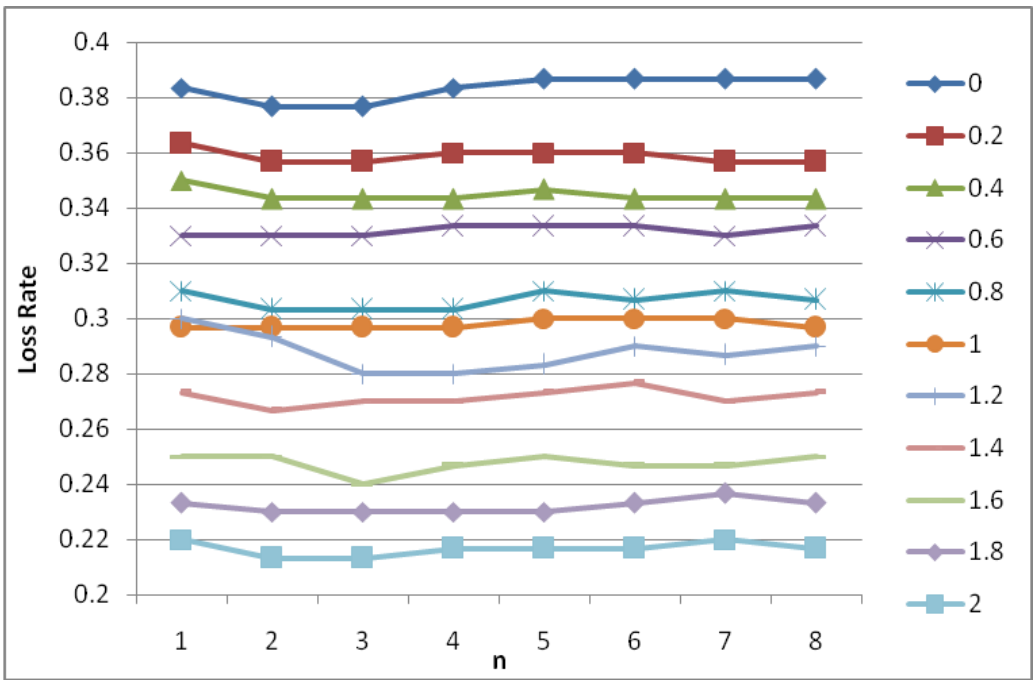

图5.12（foreman_2000，3flows，high）Flow3

从图中我们可以看到，n 的取值对于帧按时到达的比例是有一定影响的。但是，在相同的 N_{buf} 条件下，n 的取值对丢帧率的影响并不大，不超过 5%。不同的 N_{buf} 条件下，取到最低丢帧率的 n 的取值变化情况较大，并没有一定的规律性。这是因为 n 的取值和网络情况相关度较大，具有一定的随机性。

5.4.1 n 与缓存长度的关系

我们在表 5.3 中列出多流测试中的在所有五种情况下，初次取到最优的丢帧率的 n 值在每一个 N_{buf} 下的平均，记为 avg_n(buf)。

表5.3　缓存长度和 n 关系表

缓存长度	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
avg_n(buf)	2.7	2.8	2.1	2.2	1.3	1.6	2.3	2.3	1.9	1.8	1.9

在表中，最佳的 n 的均值在 1.3 到之间变化。取值分布具有一定的随机性。

5.4.2 n 与网络变化情况的关系

图 5.13 和图 5.14 给出了以（foreman_2000，3flows，high）和（foreman_2000，2flows，mid）两种组合情况为例的包组延迟的变化图，从图中可以看到，在不同的网络传输功率条件下，包组的平均延迟情况具有较大的不同，但是都具有较大的不稳定性。

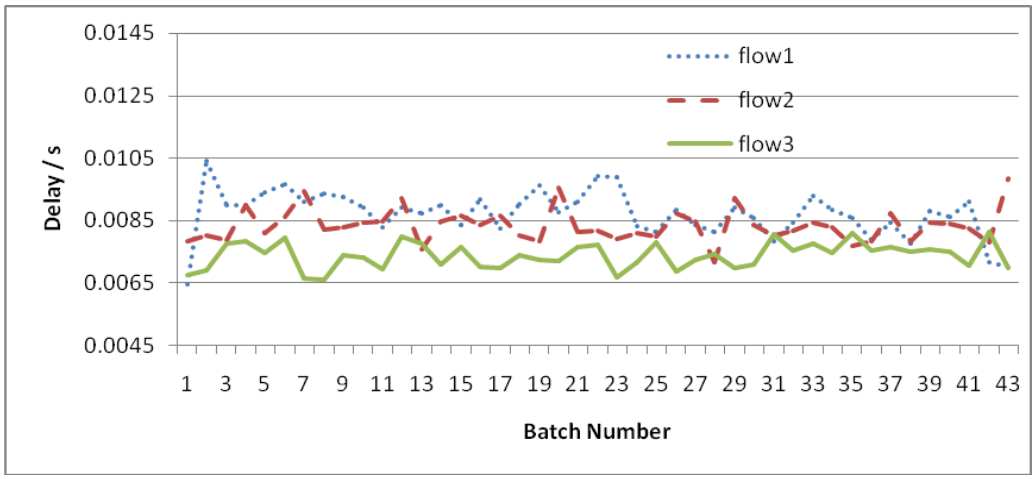

图5.13（foreman_2000，3flows，high）包组延迟

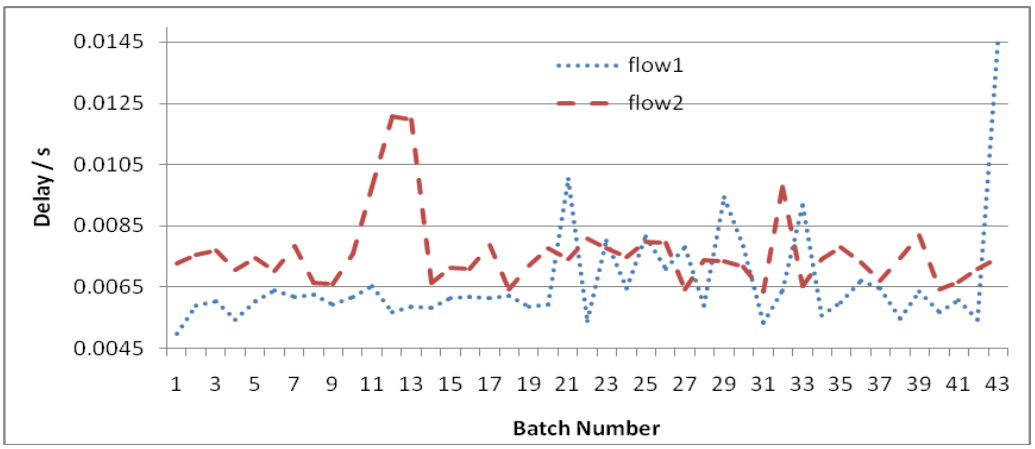

图5.14（foreman_2000，2flows，mid）包组延迟

为了描述网络的变化剧烈情况，我们计算包组平均延迟的标准差。为了使不同网络情况下的测试数据在公平的情况下比较，我们把计算的延迟标准差除以延迟的平均值进行规范化（记为 stdev）。同时，对于每一个单独的流，我们对各种缓存长度下取得最优丢包率的 *n* 值进行平均（记为 avg_n(net)），这样得到可以得到一定网络情况与最优 *n* 值的关系（表 5.4）。

表5.4　最小丢帧率下 *n* 的均值和传输延迟标准差关系表

测试名称	流标识	stdev（s）	avg_n(net)
2000_3flow_high	1	0.087604	2.45
2000_3flow_high	2	0.065207	4.18
2000_3flow_high	3	0.056095	2.00
1000_3flow_high	1	0.064427	2.55
1000_3flow_high	2	0.058169	2.36
1000_3flow_high	3	0.064427	1.00
2000_2flow_high	1	0.056993	3.55
2000_2flow_high	2	0.063608	2.18
2000_2flow_mid	1	0.249483	1.18
2000_2flow_mid	2	0.161564	1.00
1000_2flow_mid	1	0.344810	1.00
1000_2flow_mid	2	0.160121	1.27

按照表 5.4，做出 stdev 与 avg_n(net)的关系图，可以得到图 5.15。

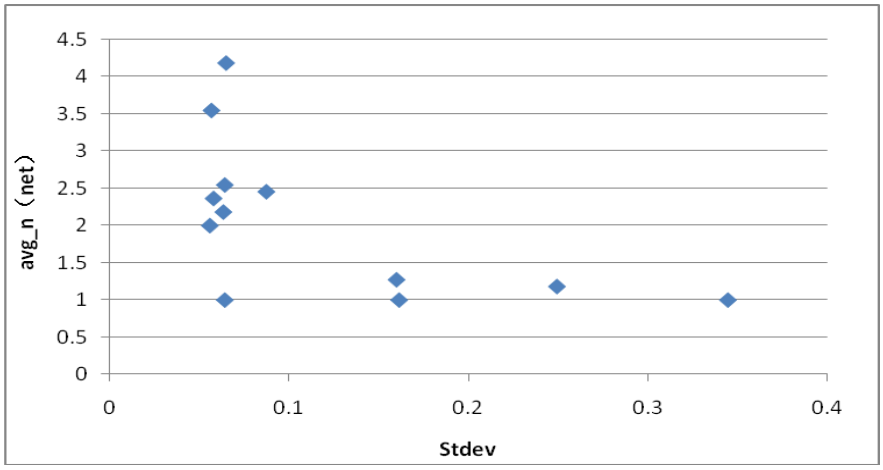

图5.15 网络波动和 n 均值关系图

从图中我们可以看到，随着包组发送延迟的 stdev 的增大，最佳的 n 值平均 avg_n(net)有变小的趋势，在网络传输变化较大的情况下，n 的取值集中在 1 附近。同时，在 stdev 较小的时候（图 5.15 的左侧的点），网络的传输延迟变化较小，这种情况下，最佳 n 的取值有较大的不确定性，从 1 到 4 之间都有分布。

5.4.3 测试结论

综合考虑 5.4.1 和 5.4.2 两小节的分析，并且考虑到 n 的变化对视频丢帧率的影响较小（不到 5%），我们认为， n 的取值为 1 是较为合理的。也就是说，我们可以直接用前一个包组的传输速率估计当前包组传输速率。

5.5 本章小结

在本章中，我们对 SAFDA 算法进行了评测，实验表明，SAFDA 可以给视频质量带来最多 15dB 的提高。另外，本章对 SAFDA 算法中的几个关键参数：Δt 、N_{buf} 、n 的设置对算法效果的影响进行了实验。实验表明，Δt 的适当取值可以简化截止时间的计算公式； N_{buf} 的提高可以接近线性的降低视频的丢帧率； n 的取值为 1 较为合理，即可以使用前一次的网络延迟数据作为当前网络延迟情况的估计。

第6章 总结和进一步的研究工作

6.1 论文主要工作总结

本文将机会路由协议研究中的最新进展——MORE 协议和无线网状网络中视频实时传输的应用结合起来，从传输吞吐率、帧传输延迟、包组传输抖动、视频帧按时到达的比例等多个方面，全面分析了 MORE 协议在视频文件的传输中的效果。采用机会路由协议 MORE，能够使得视频文件的传输达到比采用传统路由方法更高的传输吞吐率，同时，由于 MORE 中采用了包组作为传输单位，在取得较高吞吐率时，需要设置较大的包组大小，这可能会增大帧的传输延迟。

为了进一步提高视频文件传输的实时性和传输质量，我们提出了一种基于网络速率预测的源节点传输码率自适应的丢帧算法 SAFDA（**S**ource **A**daptive **F**rame **D**iscard **A**lgorithm），在网络的传输速率低于视频码率的情况下，通过提前丢弃一些重要程度较低的数据帧，显著的提高了实时视频的效果，在丢帧率和视频质量方面都取得了较大的提高，使用丢帧算法前后，视频 PSNR 的提升最高达到 15dB。

我们对于 MORE 协议的详细测试，有助于今后对机会路由实时视频传输的进一步研究。同时我们在此基础上提出的 SAFDA 算法结合了 MORE 传输的特点，取得了很好的效果。是该研究领域中的一个创新，对于今后的研究能够提供一定的参考。

6.2 进一步的研究工作

目前，我们对 SAFDA 算法进行的测试主要基于我们在真实网络环境中得到的传输延迟数据，并且测试主要采用了模拟的方式。在今后的工作中，我们可以进一步完善系统实现，在以下方面开展研究：

1) SAFDA 算法在真实系统中的实现。

我们可以将 SAFDA 算法加入到我们的实验系统中，在源节点对于传输的视频文件进行丢帧控制，以达到降低视频丢帧率，提高视频传输质量的目的。

2) MORE 协议在大规模网络环境中的测试。

限于我们现有的实验条件，我们的真实网络系统仅由 6 个节点组成。在今后的工作中。我们可以通过网络模拟软件例如 Omnet++、NS2 等在大规模的网络环

境下对 MORE 协议的实时视频传输效果进行模拟，为今后在大规模网状网络中的应用提供更多的参考。

另外，在本文的工作中，由于毕设的时间限制，对于 MORE 协议的视频传输性能，主要通过实验的方式进行评测，在进一步的理论抽象和分析方面仍然存在一定的不足。针对这一点，今后我们的工作还可以着眼于建立一个基于 MORE 协议的视频传输模型，结合无线网络传输的性质和 MORE 协议的特点，加以抽象和分析。例如，本文在 SAFDA 算法的设计中，对于主要参数 n ——即用以预测当前网络延迟的历史包组数据个数——主要通过实验和统计的方式给出。在未来的工作中，我们希望通过对 MORE 协议的视频传输模型的建立和分析，给出 n 取值的形式化描述。

插图索引

表格索引

參考文獻

[1] Sanjit Biswas, and Robert Morris. “ExOR: Opportunistic MultiHop Routing for Wireless Networks.” *SIGCOMM* , 2005.

[2] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, Robert Morris. “Link-level Measurements from an 802.11b Mesh Network.” *SIGCOMM* , 2004.

[3] Szymon Chachulski, Michael Jennings, Sachin Katti, Dina Katabi. ” Trading Structure for Randomness in Wireless Opportunistic Routing.” *SIGCOMM* , 2007.

[4] Szymon Chachulski. "How to build and run MORE." http://people.csail.mit.edu/szym/more/README.html .

[5] Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. “The Click modular router.” *Operating Systems Review* 34(5):217–231, December 1999.

[6] Bart Braem, Michael Voorhaen. “Click Modular Router Concepts”. http://www.pats.ua.ac.be/software/click .

[7] Mei-Hsuan Lu, Peter Steenkiste, and Tsuhan Chen. “Time-aware Opportunistic Relay for Video Streaming over WLANs”. *IEEE International Conference on Multimedia and Expo (ICME)*, July 2007.

[8] Mei-Hsuan Lu, Peter Steenkiste, and Tsuhan Chen. “Video Streaming Over 802.11 WLAN with Content-aware Adaptive Retry.” *IEEE International Conference on Multimedia and Expo (ICME)*, July 2005.

[9] Ian F. Akyildiz, Tommaso Melodia, Kaushik, R. Chowdhury. “A survey on wireless multimedia sensor networks.” *Computer Networks* 51(2007):921-960, 2006.

[10] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer. “RTP Payload Format for H.264 Video.” RFC3984, February 2005.

[11] VLC media player. http://www.videolan.org/vlc/.

[12] Joint Video Team of ITU-T and ISO/IEC JTC 1. “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 |ISO/IEC 14496-10 AVC).” *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050*, March 2003.

[13] Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. “Overview of the H.264/AVC video coding standard.” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 560–576, July 2003.

[14] MADWiFi: Multiband Atheros Driver for WiFi. http://madwifi.org.

[15] M. Lu, P. Steenkiste, and T. Chen. “Video Transmission over wireless multihop networks using opportunistic routing.” *Packet Video Workshop (PV2007)*, Nov. 2007.

[16] x264: a free h264/avc encoder. http://www.videolan.org/developers/x264.html.

[17] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. “Architecture and evaluation of an unplanned 802.11b mesh network”. *MOBICOM*, 2005.

[18] H. Seferoglu and A. Markopoulou. “Opportunistic network coding for video streaming over wireless.” *Packet Video Workshop (PV2007)*, Nov. 2007.

致 谢

衷心感谢杨士强老师和孙立峰老师，两位老师在我毕业设计的过程中言传身教，给了我很多的帮助。和孙老师一次次的讨论使我对研究课题的认识一步步加深。

感谢实验室的师兄师姐和同年级的同学们。感谢肖谋师兄、苗银军师兄、王智师兄、丁锡锋师兄、庞一师姐、杨春博师姐的热心指导和无私帮助。感谢胡伟栋同学，和他的讨论让我受益良多。

感谢 Click 开发社区给我的帮助，解答了我在系统开发过程中的诸多疑问。

感谢 MIT 的 Szymon Chachulski，和他的交流加深了我对 MORE 协议的理解。

最后衷心感谢我的家人，在他们的支持下，我的毕设工作才能顺利完成。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____　　日　期：_____

附录 A 外文资料的调研阅读报告

Survey for the Thesis Proposal

1. Introduction

These years the improvement of Wireless Multimedia Sensor Networks (WMSNs) i.e., networks of wirelessly interconnected devices that allow retrieving video and audio streams, still images, and scalar sensor data has enables many kinds of applications [1]. More recently, Real-time Video Streaming Service over WMSNs has caught significant attention in both research and industrial domains. But due to the inherent nature of WMSNs such as poor link quality caused by urban structures and the many interferers including local WLANs [3], there are still many Challenges. In the talk of Jongryool Kim in 2008.3.28[6], he stated three representative challenges, which are

- Limited and time-varying network available bandwidth
- Large delay due to frequent disconnection and change of video path
- Higher bit error rate (BER)

In order to transport high-quality video to end user over WMSNs efficiently, many technologies has been proposed such as opportunistic routing [2,3,5] and network coding for video [7]. In this report, a general view of these technologies is presented.

2. Opportunistic Routing

Unlike traditional routing schemes which choose the nexthop before transmitting a packet, opportunistic routing opportunistic routing allows *any* node that overhears the transmission and is closer to the destination to participate in forwarding the packet.

In [2] ExOR is proposed. ExOR is an integrated routing and MAC protocol that increases the throughput of large unicast transfers in multi-hop wireless networks. ExOR broadcasts each packet, choosing a receiver to forward only after learning the set of nodes which actually received the packet. Moreover, in ExOR only a single ExOR node forwards each packet at a time. The source includes the forwarder list in priority order based on the expected cost of delivering a packet from each node in

the list to the destination. ExOR attempts to schedule the times at which nodes send their fragments so that only one node sends at a time. ExOR only guarantees to deliver 90% of a batch, the destination requests the remaining packets via traditional routing.

In [3] MORE (MAC-independent opportunistic routing protocol) is introduced to overcome the problem of global scheduler in ExOR. MORE randomly mixes packets before forwarding them with network coding. This randomness ensures that routers that hear the same transmission do not forward the same packets. MORE does not need a special scheduler; it runs directly on top of 802.11. In MORE, a code vector is used to perform network coding. After network coding, the native packets are turned into coded packets. When a node hears a packet, checks whether the packet contains new information, in which case it is called an innovative packet. A packet is innovative if it is linearly independent from the packets the node has previously received from this batch. For each packet it receives, the destination sends an acknowledgment after destination receives all innovative packets. MORE can be used in multicast and achieve better throughput compared with traditional path routing protocol and ExOR.

ExOR and MORE both base on the unit of packets, in contrast, [4] describe a method of partial packet recovery for wireless networks.

The scheme is called PPR, which is a Partial Packet Recovery system that improves aggregate network capacity by greatly reducing the number of redundant bits transmitted. The key insight is to use information from the physical layer to improve error resilience. PPR incorporates the following two novel techniques: the SoftPHY interface and postamble decoding. PPR use PP-ARQ, a link-layer retransmission protocol in which the receiver compactly requests the retransmission of only the select portions of a packet where there are bits likely to be wrong. SoftPHY hints are tied to the details of the PHY. In PPR SoftPHY hints are PHY-dependent. Experimental results showing that Hamming distance is a good SoftPHY hint. In [4], the authors present the approach to synchronizing on packets without an intelligible preamble: to add a postamble to the end of each packet on which a receiver can also synchronize. In PP-ARQ, dynamic programming is used to find the best feedback strategy. It may chooses chunks such that each chunk “covers” all the bad runs in the packet, and may cover some good runs, if they are short

enough. The receiver encodes the feedback set in its reverse-link acknowledgement packet.

Take both the ideas of opportunistic routing and PPR, a Symbol-level Network Coding for Wireless Mesh Networks is presented in [5].

In [5], the authors describe MIXIT, a system that improves the throughput of wireless mesh networks. MIXIT routers use physical layer hints to make their best guess about which bits in a corrupted packet are likely to be correct and forward them to the destination. The core component of MIXIT is a novel network code that operates on small groups of bits, called symbols. MIXIT's symbol-level network code also incorporates an end-to-end error correction component that the destination uses to correct any errors. MIXIT incorporates two additional techniques to improve performance: increased concurrency and congestion-aware forwarding.

3. Video Transport Using Opportunistic Routing

Streaming video is inherently more challenging than normal data. The tight time constraints of video data and the predictively-coded property of video data results in different levels of importance across video packets [8] are not considered in the above opportunistic routing protocols.

To address these two problems, [8] propose TOR, a Time-aware Opportunistic Relay scheme that operates in the media access control (MAC) layer. To relay video packets in a time-aware manner, a *relay deadline* is computed for each packet, which is used by relay nodes to determine whether to relay or discard the packet. The authors of [8] assume a scenario where only one node (the video server) is the traffic source so the overheads are not considerable and they introduce a new MAC header field, namely *time_to_relay.* This field is initialized by the source and updated by each relay node to reflect how much time remains before a packet should be dropped. A retransmission deadline which is computed as a function of startup delay, frame type, and GOP structure is used to calculate *time_to_relay* as follows:

time_to_relay = relay_deadline – current_time

Whenever a relay node overhears a video packet that is not followed by an ACK, it assumes the transmission failed. Candidate relay nodes need to be ranked so that the ones with higher delivery ratios to the destination node get to retransmit first.

In [9] a similar scheme that considers multiple video streams is introduced. The authors develop schemes NCV (Network Coding for Video) and NCVD (Deeper) for network code selection and packet scheduling that take into account both (i) the importance and deadlines of video packets and (ii) the network state and the received/overheard packets in the neighborhood. In the scheme the intermediate node maintains a *transmission (Tx) queue* with incoming video packets and primary packets which can be thought as the main packet we try to transmit during a time slot are selected. Algorithm for choosing the best combination of primary packets and side packets are designed. But the schemes have shortage that nodes need to know the contents of the virtual buffers of all their neighbors, in order to code.

4. Conclusions and Future Work

In this report, the state-of-the-art opportunistic routing protocols and the issue of Video Transport Using Opportunistic Routing are surveyed.

Opportunistic routing takes advantage of the broadcast nature of wireless communication and can improve transfer throughput significantly. It is suitable for video stream transportation in WMSNs. But the original protocol must be modified to support the time constraints of video data. TOR and NCV/NCVD are two modifications but still have some drawbacks.

In the future, we will start from these two protocols and try to develop a new scheme for opportunistic video transmission in WMSNs

參考文獻

[1] I.F. Akyildiz et al.. “A survey on wireless multimedia sensor networks.” *Computer Networks* 2006.

[2] Sanjit Biswas, and Robert Morris. “ExOR: Opportunistic MultiHop Routing for Wireless Networks.” *SIGCOMM* 2005.

[3] Szymon Chachulski, Michael Jennings, Sachin Katti, Dina Katabi.” Trading Structure for Randomness in Wireless Opportunistic Routing.” *SIGCOMM 2007*.

[4] Kyle Jamieson, and Hari Balakrishnan. “PPR: Partial Packet Recovery for Wireless Networks.” *SIGCOMM* 2007.

[5] Sachin Katti, Dina Katabi, Hari Balakrishnan, and Muriel Medard.. “Symbol-level Network Coding for Wireless Mesh Networks.” *SIGCOMM* 2008.

[6] Jongryool Kim. “High-Quality Video Transport Using Opportunistic Routing Over Wireless Mesh Networks.” Networked Media Laboratory Dept. of Information & Communications GIST, http://nm.gist.ac.kr, 2008.3.28.

[7] Hulya Seferoglu, Athina Markopoulou. “Opportunistic Network Coding for Video Streaming over Wireless.” *Packet Video* ,2007.

[8] Mei-Hsuan Lu, Peter Steenkiste, and Tsuhan Chen. “Time-aware Opportunistic Relay for Video Streaming over WLANs.” *IEEE ICME* 2007.

综合论文训练记录表

<table>
<tr><td>学生姓名</td><td>冯晓君</td><td>学号</td><td>2005011312</td><td>班级</td><td>计 54</td></tr>
<tr><td>论文题目</td><td colspan="5">机会路由协议在流媒体传输中的应用研究</td></tr>
<tr><td>主要内容以及进度安排</td><td colspan="5">本毕设主要任务分为三部分。第一：基于机会路由协议 MORE 的多视频流传输系统的实现。 第二：在实验平台上测试 MORE 的视频传输性能。第三，根据 MORE 的特点提出改善视频传输实时性的方法。

进度安排如下：
第 4~8 周 调研和实验系统的开发，初步测试 MORE 协议传输性能。
第 9 周前后 准备中期检查和答辩
第 10~12 周 实现网络中多流传输的功能，并在多流条件下进一步测试 MORE 的视频传输性能。
第 13~15 周 根据视频截止时间设计视频丢帧控制算法，并进行算法的理论分析和实验测试。
第 16~18 周 撰写毕设论文，准备毕设答辩，并对毕设进行总结

指导教师签字：_____
考核组组长签字：_____
年 月 日</td></tr>
<tr><td>中期考核意见</td><td colspan="5">考核组组长签字：_____
年 月 日</td></tr>
</table>

指导教师评语	指导教师签字：_____ 年　　月　　日
评阅教师评语	评阅教师签字：_____ 年　　月　　日
答辩小组评语	答辩小组组长签字：_____ 年　　月　　日

总成绩：_____

教学负责人签字：_____

年　　月　　日