摘 要

本论文的研究内容主要定位于在分组交换式局域/城域网范围内，从网络结构、协议的角度，利用开放软件仿真平台 NS2 对高速光分组式交换网络进行了分析和研究。论文内容包括四个部分：高速光分组交换网络概述，如何基于 NS2 开发仿真软件，以及对相关协议偏射路由算法和简化的周期预留多址接入协议的仿真研究与分析。

首先简单介绍了光通信领域的整体发展现状，描绘了光通信领域研究的基本情况，比较了电路交换式光网络、突发交换式光网络和分组交换式光网络的异同，在次基础上着重分析了分组交换式光网络的网络结构、协议及相关技术。

接着对开放软件仿真平台 NS2 进行了详细介绍。NS2 是面向对象的、事件驱动的网络仿真工具，主要关注于综合服务网络中协议的交互的研究，非常适合于包交换网络，主要用于小规模的排队算法、传送协议、拥塞控制和一些与多播相关的工作。它提供了多种 TCP、路由、多播、链路层、MAC 协议的支持，但对大规模（数千节点）的仿真受到限制。本论文还对 NS 各个基本模块及相互关系，NS2 的移植及基本理论，如何仿真现有的通信协议，如何添加新的协议等进行了描述。

然后探讨了在光分组网络中采用偏射路由算法来避免或缓解光分组网络对光缓存需求的问题。通过对偏射路由模型进行仿真，将偏射路由算法和传统的存储转发网络进行了比较，结果说明，尽管相对存储转发路由算法而言偏射路由会导致数据包在网络中更大的平均

跳转次数，网络的吞吐量也因此下降，但在光处理和光存储技术还很不成熟的情况下，通过架构规则网状网和使用偏射路由算法，光分组网络能够利用网状网具有迂回路由的特性在基本无需光存储器或极少数光存储器的条件下达到很高的网络性能。当前的许多研究偏射路由工作均以规则的 ShuffleNet 为模型，本论文在此基础之上，考虑到循环路由问题，提出了新的拓扑模型，得到了很好的效果。

最后，在折叠总线网络模型的前提下，为适应高速光分组交换的局域/城域网的要求，提出简化的 CRMA 协议（SCRMA），并对各个节点的接入控制方式进行了探讨和仿真分析。S-CRMA 协议根据主节点预留队列的长度，自适应地改变预留时隙的间隔，实现了接入时延和网络资源利用率之间的均衡，即保证了在满足接入时延的条件下，尽可能高的网络资源利用率。同时，S-CRMA 对分组网络的时隙结构重新进行了定义，取消了原有协议的 REJECT、CONFIRM 等命令，降低了整个协议的复杂程度，并保证了网络在接入时延和接入速率方面的公平性。通过仿真数据和曲线，验证了基于 S-CRMA 协议的分组网络能很好的保证各个本地节点的公平接入。

关键词： 光分组交换 光时分复用 NS2 光网络 偏射路由 折叠总线 周期预留多址接入

ABSTRACT

The main topic of this paper is focused on the scope of packet-switched LAN/WAN. It analyses and studies the high-speed optical packet-switched networks on the point of view of network structure and protocols using open software simulation platform, i. e., NS2. There are four subject matters serving for the topic: the general description of high-speed optical packet-switched networks, the method of building simulation software based on NS2, the simulation study and analysis of correlative protocols including deflection routing algorithm and simplified Cyclic Reserve Multiple Access protocol.

Firstly, overall development status and basic things of optical communication are depicted and the comparison among circuit-switched, burst-switched and packet-switched optical networks is introduced. On the above foundation, the network structure, protocols and correlative technologies of optical packet-switched networks are made future analysis.

Secondly, the open software simulation platform NS2 is detailed. NS2 is a object-oriented discrete-event simulator and focuses on protocol interaction issues in integrated services internet. At the time being, NS is well-suited for packets switched networks and is used mostly for small scale simulations of queuing algorithm, transport protocol congestion control, and some multicast related work. It provides support for various

implementations of TCP, routing, multicast protocol, link layer, MAC and etc., but has limitations in the face of large simulation. The basic modules as well as their each others' relations, the transplantation with correlative theories, the how-to of simulating existing communication protocols and building new protocols are too specified.

Thirdly, to avoid or alleviate the difficulty suffering from the deficiency of the optical buffer in optical packet-switched networks, the deflection routing algorithm is advanced. On the base of simulation and comparison of the two network models, deflection and store-and-forward model, the following conclusions are obtained: by comparison with store-and-forward routing algorithm, although deflection routing algorithm leads bigger average numbers of hops when data packets transporting through networks and corresponding lower throughput, it reaches excellent performance on the background that nowadays optical processing and storing technique is not well-rounded, through taking full advantage of redundant but roundabout route resources of regular mesh optical packet-switched networks.

At last, under the condition of folded bus network model, SCRMA protocol, i.e. simplified cyclic reservation multiple access protocol is advanced to meet the need of high-speed optical packet-switched LAN and WAN, and then the access control mechanism of nodes is discussed and simulated. SCRMA protocol adaptively adjusts the interval of reservation slot according to the length of the global reservation queue of

headend, thus achieves the equalization between access delay and link utility, i.e. satisfies the access delay as well as improves link utility at the same time. Also, the slot structure is redefined in SCRMA. The original REJECT and CONFIRM commands are canceled to reduce the protocol's complexity. The above measures guarantee the fairness of access delay and rate among all local nodes, which can be validated by simulation data and curve.

KEY WORDS: Optical Packet Switching OPS, Optical Time Division Multiplexing OTDM, NS2, optical networks, deflection routing, folded bus, Cyclic Reservation Multiple Access CRMA, SCRMA

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：李国庆　　日期：2004年3月15日

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）

本学位论文不属于保密范围，适用本授权书。

本人签名：李国庆　　日期：2004年3月15日

导师签名：[illegible]　　日期：2004.3.16.

第一章 高速光分组网络概述

在 20 世纪 70 年代，光通信技术在光源和光信号传输领域取得突破，半导体激光器和低损耗光纤的实用化开辟了光通信发展的重要里程。进入 80 年代，人们已经可以充分地利用光纤巨大的带宽资源进行光信号的长距离传输，单根光纤的传输速率逐年以指数方式增长，这不但消除了网络的传输瓶颈，扩展了网络的容量，而且实现了各种业务的“透明”传输。在 90 年代的十年间，光纤通信技术依然快速发展，人们对光信号进行信息处理的能力逐渐提高，在充分掌握了光传输技术之后，还进而实现了光通道（包括波长、波段，以及光纤等光通道资源）的复用和解复用技术，实现了光信号的交换和路由技术等等。这样，光纤通信技术突破了物理层的界线，逐渐渗入并扩展到数据链路层和网络层；它不但承担数据传输的功能，还具备了一定的交换能力和路由能力。

这期间，以窄带业务为核心的传统电信网络正处于深刻的变革之中，数据业务和多媒体业务成为技术变革的主驱动力，它不但要求传统电信网络能够支撑 IP 业务，而且要求在边缘接入、城域、核心交换，以及综合业务平台等各个方面适应 IP 技术的发展，并不断地推动着传统电信网络向下一代网络（NGN）演进。IP 业务的爆炸式增长给光通信的发展提供了新的机遇和挑战，一方面巨大的 IP 业务量刺激了高速光纤传输技术的应用和发展，另一方面如何使光网络技术适应 IP 业务和下一代网络的发展也成为关键性课题。

目前提出的实现技术有三种：线路交换/波长交换、光分组交换和光突发交换。线路交换采用双向资源预留方式设置光通路，中间节点不需要光缓存，可提供有保证的服务，但交换粒度粗，不能实现统计复用，带宽利用率低，不适于传输突发数据；对于长距离网络来说，其环回时间与延迟长；由于波长数目有限，还不能建立全连接的网络，导致网络中的负载不均衡。光分组交换能对光纤的巨大带宽进行更灵活、更有效的分配和利用，却对光子器件提出了很高的要求，还有很多关键技术（如快速严格同步、光缓存等）有待发展。

光突发交换技术是上述两种技术的过渡，即在较低的光子器件要求下，实现面向 IP 的快速资源分配和较高资源利用率。它是一种单向资源预留方案，其控制分组和数据分组在时间上是分离的。控制分组先于数据分组在特定的信道中传送，核心交换节点根据控制分组的信息和网络当前的状况为相应的数据分组建立全光通路。数据分组经过一段延迟后，在不需要确认的情况下直接在预先设置的全光通道中透明传输。不需要确认的单向预留方案减小了建立通道的延迟等待时间，提高了带宽利用率；而数据分组和控制分组的隔离、适合的颗粒及非时隙交换方式降低了对光电子器件的要

求和中间节点的复杂度，如中间节点可以不使用缓存，不存在网络内时隙同步的问题等。

但是，光突发交换仍不能最有效利用网络资源，带宽效率也不是最高的，因此，从长远考虑，光分组交换（Optical Packet Switching，OPS）是光交换技术的真正发展目标。

与电分组交换技术类似，光分组交换首先可以显著地提高各个光信道的资源利用率；另一方面，可以通过 OPS 技术，延伸光的透明性，提高光网络技术在整个通信网络中的比重，即尽量利用光网络技术，在保留现有网络功能的条件下，减少当前网络中电协议层的数目，进而提升通信网络的控制管理能力和工作效率。

根据光分组交换技术的应用场合，可以将其分为光分组交换核心骨干网络和光分组交换城域/局域网络。“分组交换式光网络的结构”部分将详细阐述此内容。

根据信号的处理方式，OPS 网络可以分为全光型和光电混合型两种网络模型。在全光分组交换中，分组头的产生和识别、分组的处理和交换全是在光域进行的。这样可利用光信号带宽，提高交换速度和信号透明性。但目前全光处理技术并没有突破性的进展，无论是从灵活性还是实现方式来看都不是很理想。而光电混合型方案中分组净荷以光的形式通过交换节点，控制信息或分组头却转换成电域信号，在电域完成分解、解释、控制、变换等过程以保证光分组净荷能有效地通过高速节点。

根据光分组格式，光分组交换又可分为定长时隙光分组网络（Slotted Network）和非定长时隙光分组网络（Unslotted Network）。前者采用固定长度的分组，每个光分组在给定时隙内交换，它能有效地降低光缓存处理的难度，缩短分组保护带的宽度。定长时隙光分组网络一般要求严格的同步关系。非定长时隙光分组网络中，光分组长度与适配到该分组的数据突发长度相适应，提高了信道的利用率，但光缓存处理难度增大，信号同步比较困难（要求比特级同步），分组保护距离也增大。“分组交换式光网络的协议”部分将详细阐述此内容。

对于光分组传送网络，在光网络中由于光缓存难以实现或大规模应用，因此需要利用光传送网络的光纤冗余度（指利用两个节点之间的闲置光纤解决碰撞的问题）、波长冗余度（指利用两个节点之间的闲置波长解决碰撞的问题）、连接冗余度（指利用两个节点之间的闲置连接解决碰撞的问题，如偏射路由算法）和时间冗余度（在时间域上解决碰撞问题，如光缓存技术）。当然，实际中解决 OPS 网络中的碰撞问题，需要综合应用以上多种解决方案。“分组交换式光网络的协议”部分将详细阐述此内容。

根据光分组交换的交换方式，光分组交换技术可以分为波分、光时分和光空分分

组交换三种基本形式。当然多数的光分组交换方案中，光分组网络采用了以波分复用通道为核心的交换方式。对于未来光网络，采用了三者之间的混合交换方式，充分利用各个通道资源，将是最佳的交换方式。“分组交换式光网络的关键技术”部分将详细阐述此内容。

展望 21 世纪，光通信技术将以人们难以想象的速度向前发展，逐渐成为信息通信产业的重要支柱和下一代网络（NGN）的核心支撑技术，预计在未来 10 年光通信技术还将继续保持持续增长的趋势。在未来的十年中，OTDM 技术、OCDMA 技术、OPS 技术和 OBS 技术将逐渐丰富现有的光传送网络，新的接入方式和业务模式也将随之出现，同时保证多种粒度接入和多种服务质量，结合光分组交换和光突发交换技术，将成为未来全光网络发展的主要方向。

第一节 分组交换式光网络的结构

从网络应用的角度对光纤通信网络进行分类，可以分为局域网、城域网和核心骨干网。一般地，城域网和局域网采用线性拓扑，如环状网络和总线型网络，这样的网络建设简单、灵活，易于扩展，属于单跳网络；而核心骨干网采用网状网络拓扑结构，通过冗余的网络连接提供丰富的保护策略和路由策略，属于多跳网络。它们针对不同的应用环境，具有各自非常鲜明的特点。构建线性局域网络的主要目的是实现局域或城域网络的高速互联，以及本地数据的接入控制，它采用简单的网络拓扑结构使得它具有易于管理和维护，适合于为企业或机构提供高速的接入或互联解决方案。网状网络的主要应用环境是核心骨干网络，它可以通过节点之间的丰富的连接度实现策略路由、保护恢复等功能。

1.1.1 分组交换式城域/局域光网络

在光纤城域/局域网络中，与电路交换式光网络相比，分组交换式光网络展示出了明显的优势。有一些文献对基于 WDM 技术或比特间插的 OTDM 技术的光纤城域/局域网进行了阐述和分析，但是这些电路交换式光城域/局域网不能满足城域或局域网高业务突发性的要求。而基于光分组技术或 OTDM 分组间插技术的网络可以支持高突发性的业务，有效地提高了网络带宽利用率，同时提供灵活的接入粒度。该类技术充分地实现了总线资源的统计复用功能，符合光分组局域/城域网络业务突发性的特点，因此分组交换式城域/局域光网络将是未来光局域/城域网络的发展方向。同时，分组交换式城域/局域光网络将成为现有通信网络引入光分组技术的切入点和突破口，即在出现一些特定的城域范围内出现超高速互联应用需求，如大型计算机互联、

战场实时数据互连等应用。基于以上原因，分组交换式光网络是本论文研究的焦点。

HLAN（Helical Local Area Network）网是结合 OTDM 技术的典型的单跳网络，它是由美国 ARPA 计划支持，AT&T Bell Lab、Digital Equipment Corp.和 MIT 共同开发的一种采用分组间插 OTDM 技术的单比特流 100Gbit/s 的单跳光网络。该网主要为高速视频服务器、Tbit 媒体库和巨型计算机服务，以提供从 10G~100Gbit/s 的按需分配带宽（BOD）和固定带宽（GBW）业务。其功能为：(1) 为高速网络间的互联提供骨干网；(2) 迅速传递大型数据块；(3) 实现多种业务间的交换；(4) 提供对于高速设备（如视频服务器）的低速访问。

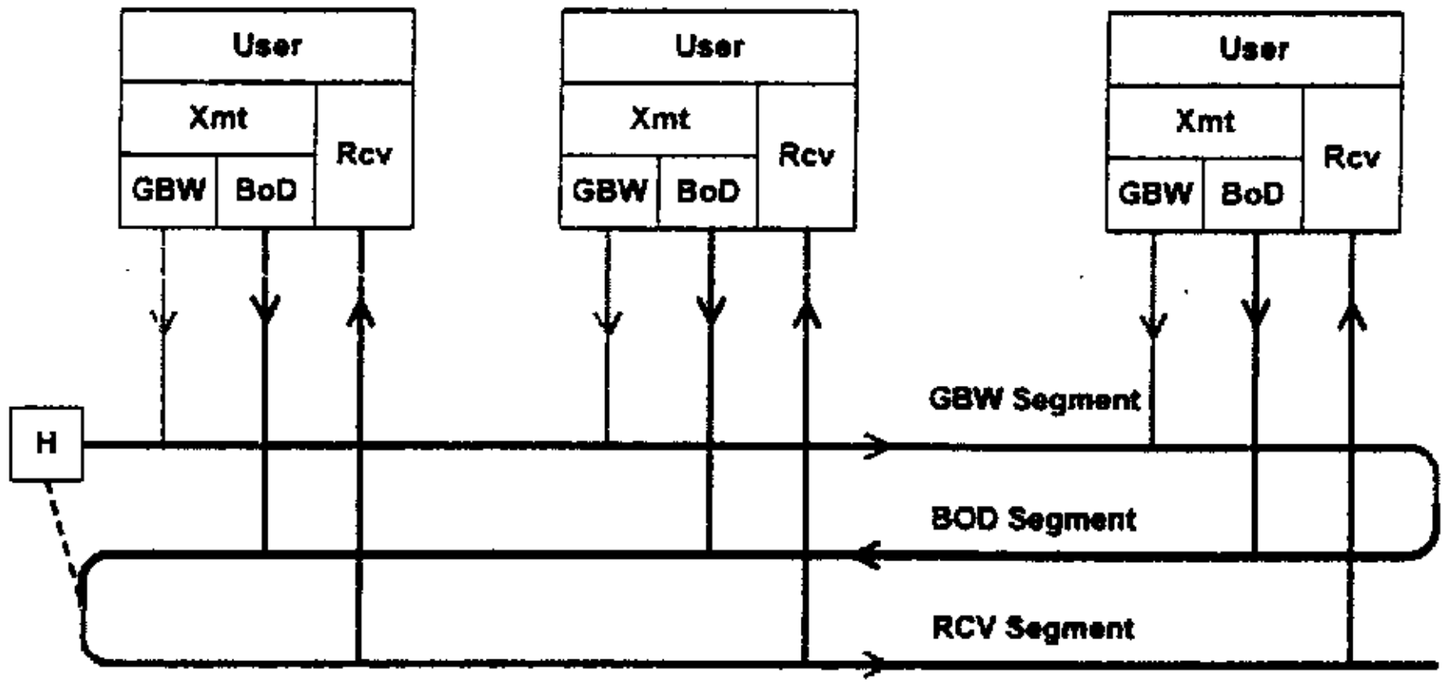


图 1-1 HLAN 网络结构

HLAN 采用改进的折叠总线结构，如图 1-1 所示，总线分为三个部分，即 GBW 段、BOD 段和 RCV 段，分别与各个本地节点的发送端 GBW、发送端 BOD 和接收端 RCV 相连；因此也可以称作 GBW 写总线、BOD 写总线和读总线。主节点（H）产生一系列含有空时隙的帧，并将其置于总线上。本地节点将本地的固定带宽业务在 GBW 段上传输，将按需分配带宽业务在 BOD 段上传输，所有的数据都在 RCV 段上接收。这样所有节点的接收机都位于所有节点的发射机的下行方向上。在该网络中采用一种空的时隙标记来通知节点可以在该时隙上写入数据。关于折叠总线的详细阐述和分析请参见第四章。

下面简要介绍一下基于分组间插的 OTDM 技术的光局域/城域网络的特点。

如果从技术实现的角度，可以将 OTDM 网络分为基于比特间插技术的 OTDM 网络和基于分组间插的 OTDM 网络。所谓 OTDM 分组间插的系统或网络是指每个用户的数据以分组的格式发送，用户分组彼此之间是间插的。与比特间插的 OTDM 系统相比，分组间插的 OTDM 网络具有以下优势：

- ✧ 统计复用的扩展，如果在本地采取电的缓存机制和本地的接入机制，将可以

实现本地节点间的统计复用；

- ✧ CoS 和流量控制机制的扩展，结合本地接入策略和主节点的时隙管理策略可以实现 CoS 的功能和流量控制的功能；
- ✧ 潜在的扩展能力，可以与 Photonic Slot Routing 技术结合，用波分复用技术进一步扩展 OTDM 局域网的吞吐量；
- ✧ 多速率接入的扩展，通过优化的帧结构的设计，可以实现本地用户多速率甚至全速率的接入解决方案，这是 OTDM 比特间插的方案无法比拟的。

一般地，分组交换式光局域/城域网络采用线性的网络结构。对于各种线性网络结构，包括星形结构、环形结构、总线式结构等等，都可应用于光纤 MAN/LAN 中。不同的网络结构通过采用不同的媒质接入控制机制，以实现信道的共享，并尽可能的提高信道利用率、对不同的节点做到公平的分配信道。

1.1.2 分组交换式核心骨干光网络

所谓多跳网络，是指数据包必须经过多个节点路由才能到达目的节点。通过这种方式，多跳网络提供了丰富的网络连接冗余度，有利于实现网络扩展和解决数据分组碰撞等问题。

在传统的电交换网络中，一般采用电缓存器和存储转发方式来解决路由冲突问题。但是在光网络中由于光缓存和光存储器件的不成熟，使得不但要考虑光纤冗余度、波长冗余度和时间冗余度来解决资源竞争问题，而且需要考虑通过规则网络的结构特殊性，利用网络连接冗余度通过偏射路由的方式解决资源竞争的问题。

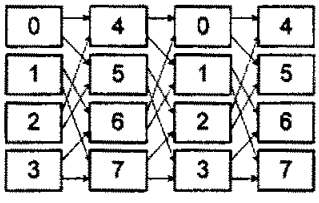


图 1-2 ShuffleNet 拓扑结构

目前提出的多跳网络结构主要有 ShuffleNet（混洗网）、Manhattan Street（MS，曼哈顿街区）、SuperCube（超立方体）、de Bruijn Graph（dBG）网等，其中又以 ShuffleNet 和 MS 结构具有更多的优点和更强的实用性，其它的普遍具有结构复杂、各节点业务量不平衡、链路利用率低等缺点，无法得到广泛应用。ShuffleNet 网络拓扑结构如图

1-2 所示，在第三章将对其进行详细分析。

图 1-3 是欧洲 KEOPS 项目（KEys to Optical Packet Switching）中描述一个 OPS 核心节点的通用功能框图。它包括输入接口、交换矩阵、输出接口、交换控制单元、同步控制单元和分组头重写等部分组成。

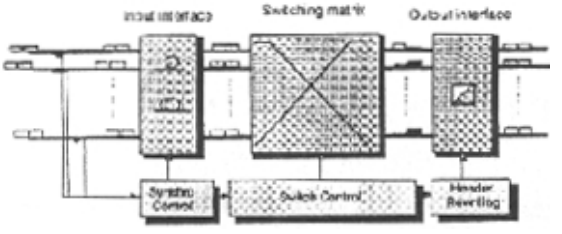


图 1-3 KEOPS 核心节点的通用功能框图

第二节 分组交换式光网络的协议

1.2.1 分组格式

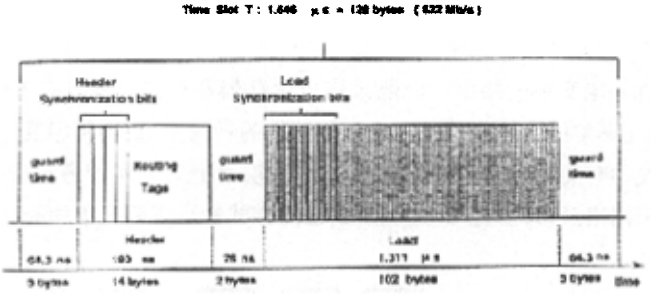


图 1-4 KEOPS 分组格式示意图

一般地光分组格式如图 1-4 所示，主要包括三个部分，分组头、分组净负荷和保护间隔。保护间隔用于实现分组之间，以及分组头与分组净负荷之间的信息保护，以防止由于抖动所造成的信息重叠；分组净负荷承载用户数据，占有分组长度的大部分；分组头的长度是一个最优化问题，因为它一方面要服务于所需的所有控制功能，另一方面它是一种开销，因此需要根据实际光信息处理能力和光网络功能要求合理设计分组头。另外，还包括分组头同步比特和净负荷同步比特，以实现分组头和分组净负荷的同步功能。如果采用相关的流量控制等协议，还需要设计相应的协议字节。

1.2.2 分组交换式光 MAN/LAN 中的媒体接入控制协议

在前面 HLAN 的例子中，由于上游节点（Upstream Node）要比下游节点先发现来自主节点 H 的分组，因此将造成各个节点占用总线资源的不公平性。因此，需要采用相关的媒体接入控制 MAC 协议，以保证各个节点的公平接入。在 HLAN 网络中，针对 GBW 和 BOD 业务分别应用了基于预留和基于“Credit”的媒体接入控制方式，虽然可以解决公平接入的问题，但是网络结构、节点结构都很复杂。本文在后续章节中，提出了基于周期预留的媒体接入控制协议，不但保证了折叠总线中各个节点的公平接入，还提供四种类型的服务，相关内容将在第四章中介绍。

1.2.3 解决资源竞争的相关策略和协议

当处在两个不同输入端口的两个不同分组要同时被送往同一输出端口时，两个分组将在输出端发生碰撞。一般地，利用光纤（空间）冗余度、时间冗余度、波长冗余度和网络连接冗余度可以解决资源竞争的问题。在第三章，将采用偏射路由算法作为这种资源竞争问题的一种解决方案，然后通过软件仿真对其性能进行了分析，并与普通存储转发网络进行了比较，详见第三章。

第三节 分组式光网络的关键技术

目前，不论是结合 WDM 技术，还是结合 OTDM 技术构建的光分组交换网络，都受到众多单元技术的限制，如光存储和光缓存技术不成熟等等。这里简要介绍一下相关技术内容：光分组交换技术、分组头处理技术、3R 再生技术、网络同步技术。

光分组交换技术的核心是交换矩阵，交换方式大体可分为三类：波分的交换结构、空分交换结构，以及时分交换结构。为了能在分组级上实现吉比特交换，交换速度必须达到纳秒量级。目前光分组交换中的各种交换结构都是在光域上实现分组的存储和交换，在电域上完成路由选择和存储控制。

光时分分组交换是指结合光时分复用技术的一种交换方案，也是非常重要的一种光分组方案，其原理类似于传统电信网络中程控交换机的时隙交换方式，其示意图见图 1-5。首先，当光分组进入交换节点，首先对光分组进行分组压缩，压缩后的光分组根据电路控制经历不同的时延，通过排序后依次进入交换矩阵，各个输出端只需要按照顺序将压缩后的分组读出，然后进行分组的解压缩，也就是可变写入、顺序读出的工作方式。当然，也可以考虑顺序写入、可变读出的方式，但是考虑分组写入比较容易（仅需要精确的时延控制），而分组读取需要利用分组光开关，因此建议采用可

变写入、顺序读出的工作方式。

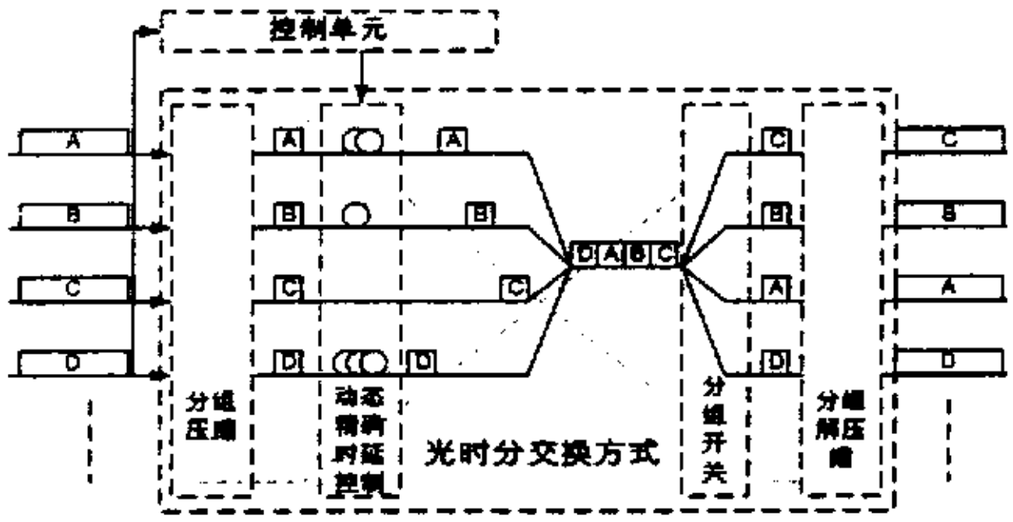


图 1-5 光时分交换方式

分组头的处理包括分组头的识别、分组头的分离和分组头的逻辑分析等内容。目前，分组头的识别和分组头的分离都可以在光上进行，但是分组头的逻辑分析，如地址分析等等，还需要在电域上进行。

为了实现分组头的分离和识别，可以将分组头与净负荷通过不同的波长或偏振进行传送，但是由于波长之间走离的影响，在较长距离内难以实现。在 University of California Davis 的相关实验中，采用了副载波调制技术（SCM），实现分组头的识别与分离。分组头和分组净负荷复用在同一个光载波上，在调制激光器的电流中，负载是基带编码，分组头比特调制在合适的副载波频率上，以较低的速率编码。因为激光器与光电探测器的频率响应必须达到副载波频率，因此，副载波频率尽量低，只需要是分组头比特率的两倍即可。

随着网络规模的扩大，光信号通过节点数量和光开关的数量越来越多，为了实现全光的数据传送，需要对网络中光信号进行再放大、再定时、再定形的处理，即 3R 再生技术。

它的基本原理框图如图 1-6 所示，主要包括位时钟提取和光开关两部分。光开关可以采用非线性光学环路镜（NOLM）或 MZI 实现，而位时钟的提取可以采用注入锁定的位时钟提取技术。

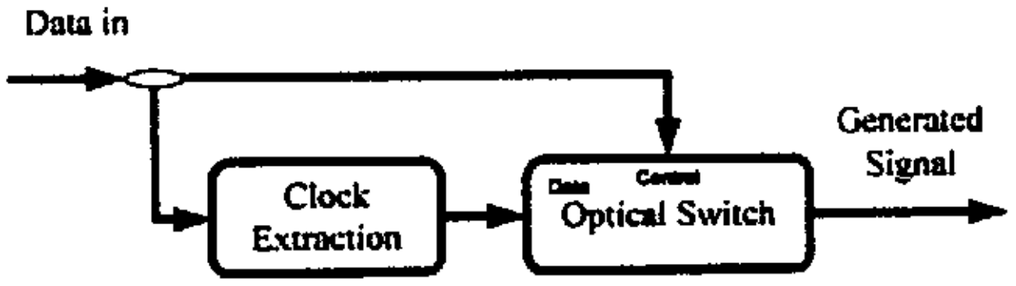


图 1-6 3R 再生技术的基本原理

由于到达 OPS 节点的分组可能来自不同源头，经历了不同的光纤路径和处理，也可能是不同的波长，各个分组因温度变化、色散和路径不同，必然有不同的传播时延；另外，交换矩阵的光分组处理也会恶化光分组的定时特性。因此，虽然在光分组帧格式的设计上考虑了保护间隔，在 OPS 节点的输入端，还必须对到达的各个光分组进行同步处理。

时钟提取是实现网络同步的有效途径，对于 Slotted 光分组网络来说，需要考虑帧时钟的提取和位时钟的提取（也就是比特时钟）。但是对于 Unslotted 光分组网络来说，在光域上直接实现光分组的同步还是非常困难的，这也是 Unslotted 光分组网络应用的瓶颈之一。

第四节 本论文的主要工作和成果

本论文针对光通信研究的最新进展，将焦点集中在基于光分组交换的局域/城域网络范畴之内，通过设计和实现光纤局域/城域网的网络架构，分析了光纤局域/城域网中的接入控制问题和资源竞争问题，并进而开展了相应的理论研究、仿真研究。

论文包括四个部分：高速光分组交换网络概述，如何基于 NS2 开发仿真软件，以及对相关协议偏射路由算法和简化的周期预留多址接入协议的仿真研究与分析。

本论文首先简单介绍了光通信领域的整体发展现状，描绘了光通信领域研究的基本情况，比较了电路交换式光网络、突发交换式光网络和分组交换式光网络的异同，在次基础上着重分析了分组交换式光网络的网络结构、协议及相关技术。

接着对开放软件仿真平台 NS2 进行了详细介绍。NS2 是面向对象的、事件驱动的网络仿真工具，主要关注于综合服务网络中协议的交互的研究，非常适合于包交换网络，主要用于小规模的排队算法、传送协议、拥塞控制和一些与多播相关的工作。它提供了多种 TCP、路由、多播、链路层、MAC 协议的支持，但对大规模（数千节点）的仿真受到限制。本论文还对 NS 各个基本模块及相互关系，NS2 的移植及基本理论，如何仿真现有的通信协议，如何添加新的协议等进行了介绍。

然后探讨了在光分组网络中采用偏射路由算法来避免或缓解光分组网络对光缓存需求的方案。通过对偏射路由算法进行仿真，将偏射路由算法和传统的存储转发网络进行了比较，结果说明，尽管相对存储转发路由算法而言偏射路由会导致数据包在网络中更大的平均跳转次数，网络的吞吐量也因此下降，但在光处理和光存储技术还很不成熟的情况下，通过架构规则网状网和使用偏射路由算法，光分组网络能够利用网状网具有迂回路由的特性在基本无需光存储器或极少数光存储器的条件下达到很高的网络性能。当前的许多研究偏射路由工作均以规则的 ShuffleNet 为模型，本论文

在此基础之上，考虑到循环路由问题，提出了新的拓扑模型，得到了很好的效果。

最后，在折叠总线网络模型的前提下，为适应高速光分组交换的局域/城域网的要求，提出简化的 CRMA 协议（SCRMA），并对各个节点的接入控制方式进行了探讨和仿真分析。S-CRMA 协议根据主节点预留队列的长度，自适应地改变预留时隙的间隔，实现了接入时延和网络资源利用率之间的均衡，即保证了在满足接入时延的条件下，尽可能高的网络资源利用率。同时，S-CRMA 对分组网络的时隙结构重新进行了定义，取消了原有协议的 REJECT、CONFIRM 等命令，降低了整个协议的复杂程度，并保证了网络在接入时延和接入速率方面的公平性。通过仿真数据和曲线，验证了基于 S-CRMA 协议的分组网络能很好的保证各个本地节点的公平接入。

参考文献

[1] Gambini, P.; Renaud, M.; etc., "Transparent optical packet switching: network architecture and demonstrators in the KEOPS project", IEEE Journal on Selected Areas in Communications, Vol.16 Issue7, Sep 1998, pp1245 -1259

[2] Guillemot, C.; Renaud, M.; etc. "Transparent optical packet switching: the European ACTS KEOPS project approach", Journal of Lightwave Technology, Vol.16 Issue12, Dec 1998 pp2117–2134

[3] S.W. Seo, B.Y. Yu, and P.R. Prucnal, Bit-Level Packet-Switching All-optical Multihop Shuffle Networks with Deflection Routing, Appl. Opt., Vol. 36, No. 14, 1997, pp.3142-3146

[4] Christian Guillemot, etc.; "Transparent Optical Packet Switching: The European ACTS KEOPS Project Approach", IEEE Journal of Lightwave technology, Vol.16, No.12, DEC. 1998, pp2117-2134

[6] 雷震洲，"核心网中的光分组交换"，电信工程技术与标准化，No. 1，2003

[7] 戴翌清等，"折射路由法在光分组交换网络中的应用"，光纤与电缆及其应用技术，No. 3，2003

[8] 镇桂勤等，"分组交换技术与光分组交换技术"，现代通信，No. 10，2001

[9] 李新碗，叶爱伦，陈建平，"光突发交换试验网的可行性分析"，中兴通讯技术，2002 年第五期

[10] 左鹏，"分组交换式光网络一结构、协议和相关技术"，北京邮电大学博士研究

生学位论文，2003 年五月

[11] 吴龟灵，陈建平、李新疏，“采用部分共享光缓存和共享波长转换器的关分组交换节点性能分析”，电子学报，No. 11，2003 年 11 月

[12] 周建华，蔡茂国，杨淑雯，“关分组交换技术研究”，深圳大学学报，第 18 卷，第四期，2001 年 12 月

[13] 杨震，“光分组交换技术在未来通信网中的应用”，通讯世界，2001 年 10 月

[14] 隋志成，姜希军，吴志坚，“光分组交换网络结构”，现代有线传输，第 1 期，2002 年 3 月

[15] 纪越峰，柏琳，徐大雄，“光分组交换系统的结构研究”，激光技术，第 25 卷，第 1 期，2001 年 2 月

[16] 张正线，“全光网络中的全光分组交换技术”，电讯技术，1999 年第 6 期

[17] 殷洪玺，王勇，徐安士，“光分组交换网”，电信科学，2000 年第 11 期

[18] 谢世钟，陈明华，董毅，陈向飞等，“光分组交换网络技术展望”，清华大学学报，2002 年第 42 卷第 7 期

[19] 戴礼森，洪佩琳，“光分组交换技术”，现代电信科技，1998 年 10 月，第 10 期

[20] 郑福生，张劲松，“光分组交换技术的研究”，光通信研究，2002 年第 4 期

第二章 基于NS2开发光网络协议仿真软件

第一节 NS2简介

NS是面向对象的、事件驱动的网络仿真工具。NS1最初是由Lawrence Berkeley National Laboratory (LBNL)的Network Research Group开发出来的，NS2是VINT project的一部分。该项目的目的是建立一套具有创新意义的网络仿真工具，主要关注于综合服务网络中协议的交互的研究。它不是要设计一套全新的网络仿真工具，只是想把所有网络仿真领域的工作成果统一起来。目前，大多数网络仿真工具都只关注于单一的一种协议，多种协议之间是无法联系在一起的，与别的工具的组件也无法交互，因而，其努力是有局限性、较初级、孤立的，仿真结果之间缺少比较与联系，各种工具之间有很多重复性的工作。

而且，随着技术的进步，网络的规模、协议的数量、协议之间的交互、硬件等等都变得越来越复杂，很难设计出一种评估工具以适应这种变化，而每次都重新设计的代价又太高了，而且如今的大多数工具都对可视化、结果的分析的支持较弱。

NS还提出了增加仿真工具之间协同性的思路，所有研究者都可以互相交流来改善仿真的框架。这个项目借鉴了MIT的NETSIM、University of Maryland的MARS、UC Berkeley的REAL、 Columbia的NEST和LBNL的NS的经验，在NS和NAM的基础上构建（NAM是一种动画工具，用来观察仿真结果和跟踪包的数据）。它旨在评估大规模互联网络从路由层协议到会话层协议各个层次的正确性和性能。它提供一种可重构的仿真模型，这种模型模拟Internet的各模块，可以支持不同人发布的组件模块。它还包括大量可扩展库，支持各种各样的抽象的技术、工具以及可视化。

2.1.1 应用环境

目前，NS非常适合于包交换网络，主要用于小规模的排队算法、传送协议、拥塞控制和一些与多播相关的工作。它提供了多种 TCP、路由、多播、链路层、MAC协议的支持，、但对大规模（数千节点）的仿真受到限制。

NS2本来是在Unix或Linux的环境下运行的，所以要想在win32平台下、使用大家非常熟悉的MVC6这样的集成化、可视化的开发工具，就必须进行移植工作。

2.1.2 语言支持

NS2 的仿真代码是由两种语言写成：C++和 OTcl，这是因为仿真器有两方面的需要：

- 一方面，对协议细节的仿真需要一种比较系统的编程语言，它能高效地对字节、数据报头进行操作，能高效实现运行在大量数据集上的算法。对于这些任务，运行时间（run- time）是我们最关心的，要尽可能少，而转换时间（turn-around time，即运行、调试时间）可以长一些。
- 另一方面，对网络研究会涉及到对网络实体的参数或配置的修改，或是对网络事件的修改。在这些情况下，反复时间（iteration time，即变换模型、重运行）更重要。由于配置仅运行一次（在仿真刚开始时），因此这一部分任务的运行时间可不用考虑。

C++与 Otcl 解决了这两种需要。C++程序的运行时间很短，但转换时间很长,正适合用于实现具体的协议；Otcl 运行得很慢，但可以很快的转换(或是交互)，用来进行仿真的配置最适合不过。然后，通过 tclcl 模块将出现在两种语言里的变量、对象胶合起来。

当然以上的分法并不是绝对的，例如，许多路由算法是由 Otcl 实现的，但其核心算法——Dijkstra 算法是用 C++来实现的。通常，如果需要在一秒钟内调用许多次的模块，最好还是用 C++代码实现。

第二节 NS2 的体系结构和主要软件包

NS2 平台的体系结构比较复杂，其主要的模块包括：tcl、tk、otcl、tclcl、ns、nam、itm、GenApps、Xgraph、awk 等，每一个模块均由对应的一个软件包来管理，位于相应的目录之下，构成一个相对独立的功能模块；而所有这些软件包则全部包含于 NetSim 目录之下，共同组成完整的软件仿真平台，可以用来仿真各种现有的网络协议，也可以根据需要自己手工修改已有协议和添加新的协议。下面简单介绍各模块功能。

Tcl 模块是一个解释器，完成对 tcl 脚本的解释执行。Tcl 脚本语言为集成应用提供了强有力的平台，可以将各种不同的应用、协议、设备和框架集成在一起，当配合 tk 小工具（见稍后说明）一起使用时，能为运行于各种平台（win32、Unix、Macintosh）上的 GUI 应用提供最快最好的解决方法，特别适合于各种网络相关的应用。

Tk 模块是用 tcl 脚本语言实现的 X11 工具，完成所需的图形部分的功能。

Otcl，在此项目中具体为 MIT 版本的 Object Tcl，是 tcl/tk 面向对象编程的扩展，并且是完全基于 tcl 的语法和概念扩展的，同 tcl 一样，功能强大且具有很大的灵活性。

所有 tcl 脚本的解释与执行，就是由上述三个模块共同来完成的（在实际仿真中，全部采用面向对象的 tcl，即 Otcl，但由于二者语法、概念完全一致，所以以后提到的 tcl 一般指 Otcl，几乎不会产生误解）。

Tclcl 模块为 tcl/C++提供接口，是 tcl 脚本和 C++之间的“胶合剂”，供 ns、nam 等调用。对于绝大多数的类和对象，在 tcl 脚本中和 C++中都是一一对应的，或者说用 tcl 脚本语言和 C++编译语言同时实现了两组相同的类和对象，那它们之间是如何实现统一与同步的呢？这就是 tclcl 模块的主要功能，把两组对象关联起来，在任何时候都达到一致。

NS 模块是 NS-2 网络仿真的核心部分，它利用了上述模块，并包含各种协议、算法等，所有新功能的添加也在此完成。后面章节将对其进行详细叙述。

其他的模块主要是完成对仿真得到的数据进行处理。Nam 模块是专为 ns 设计的图形化工具，它利用仿真得到的数据（存在*.tr, *.out 文件中），自动生成动画图像，形象的演示仿真的状态和过程，如链路、节点、拓扑状态和入队列、出队列、丢包动作等等，但并非是仿真过程的精确描述，后面章节将对其进行进一步说明。Xgraph 是 X-Windows 应用程序，则可用来对生成的数据进行统计分析，并可得到图形化的结果。Gt-itm 是 GT Internetwork Topology Models 的简称，用来产生互连网络结构的拓扑。

第三节 移植

在将系统移植到 win32 平台下时，最主要的问题就是重新组织项目结构、设置系统参数，如路径、版本号、与 MSVC 语法习惯一致等，和一些细节的修改，这里以 tcl 包的安装为例，说明移植是如何实现的，其它的包的安装过程与此类似。

2.3.1 准备工作

（1） 下载上述各软件包（均为开放软件，可免费下载作为研究使用），并按要求组织好结构，此处以所有包均置于 D:\NetSim 下为例说明。

（2） 安装 MSVC6.0，此处以安装在 C:\Progra~1\Micros~3（是 C:\Program Files\Microsoft Visual Studio 的缩写，与旧版中目录名不超过 8 个有效字符兼容,具体写法可通过实验获得）为例。

（3） 为 VC 设置环境变量，在 DOS 提示符下执行下述命令：

```
D:\NetSim>path=%path%;C:\Progra~1\Micros~3\VC98\bin↙
D:\NetSim>vcvars32↙
```

2.3.2 tcl 包的安装

在子目录 win 下，有 windows 下 VC 特有的 makefile 文件 makefile.vc，对 tcl 的安装最主要的工作就是对此文件的配置，详细的配置见 D:\NetSim\tcl8.3.2\win\makefile.vc 文件，这里只对主要的环境参数配置进行说明。

仿真平台所在目录，即 NetSim 的目录定义，ROOT=D:\NetSim;

VC 工具包所在的目录 tools32=C:\Progra~1\Micros~3\VC98;

tcl 的安装目录的定义，INSTALLDIR=D:\NetSim\tcl8.3.2;

tcl 版本信息的定义 VERSION=8.3;

调试信息的定义 NODEBUG=1,即不调试;

运行时库文件*.lib 文件、动态连接库文件*.dll 文件的指定;

windows 下的的生成新目录、删除新目录、删除文件等命令的定义;

与 VC 习惯相一致的编译、连接命令的定义

```
cc32      = "$(TOOLS32)\bin\cl.exe"
link32    = "$(TOOLS32)\bin\link.exe"
rc32      = "$(TOOLS32_rc)\bin\rc.exe"
include32 = -I"$(TOOLS32)\include"
libpath32 = /LIBPATH:"$(TOOLS32)\lib"
lib32     = "$(TOOLS32)\bin\lib.exe";
```

各种依赖关系的定义，包括如何生成各种目标文件，如何连接等等。

在各种参数配置完成后，使用下述命令编译并安装包：

```
cd D:\NetSim\tcl8.3.2\win↙
nmake /f makefile.vc
```

```
nmake /f makefile.vc install
```

然后就完成了 tcl 包的安装。此时在 tcl 目录下生成新文件夹 bin，内包含有刚刚编译生成的解释 tcl 脚本的动态库文件和可执行文件：

```
tcl83.dll tclpip83.dll tclsh83.exe
```

此时，可以编写自己的 tcl 脚本进行测试，运行时在命令提示符状态下敲入 tclsh83↙，然后就可编写自己的脚本，或者直接敲入

```
tclsh83 mytest.tcl ↙
```

其中 mytest.tcl 为已编写好的脚本。

其它软件包的安装与上述过程类似。

最后，为整个系统设置环境变量，包括添加新的环境变量 NS-2 和修改系统路径 Path。

到此为止，整个系统的移植工作基本结束，可以在提示符状态下运行 NS2 自带项目脚本（在 D:\NetSim\ns-2.1b8a-win\tcl\ex 下）。

但要想利用 MSVC 这样的集成开发工具进行开发和调试，还要做一些改动，比如在项目的环境设置中填写与实际情况一致的参数，主要是工作目录、要执行的脚本和参数等。

第四节 make 工具

在介绍 NS2 平台的详细构架之前，有必要首先对 make 工具和 makefile（在 MSVC 下为 nmake 和 makefile.vc）文件做简单介绍，因为整个项目就是用 makefile 文件进行构架和管理的。

在小型项目的开发中，make 工具不能带来太多的方便，但在大型的开发项目中，通常有几十到上百个的源文件，如果每次均手工键入编译、连接命令的话，不仅非常不方便，而且在调试时很容易出错。因此，人们通常利用 make 工具来自动完成编译工作，这些工作包括：如果仅仅修改了某几个源文件，则只重新编译这几个源文件及相关文件；如果某个头文件被修改了，则重新编译所有包含该头文件的源文件；如果项目加入了新的模块，则只编译跟该模块相关的部分文件。这样，就大大简化了开发、调试工作，避免了不必要的重新编译。

实际上，make 工具通过 makefile 文件来自动完成编译维护工作。Makefile 文件

需要按照某种语法进行编写，其中说明了如何编译各个源文件并连接生成可执行文件，并定义了源文件之间的依赖关系。当修改了其中某个源文件时，如果其他源文件依赖于该文件，则也要重新编译所有依赖该文件的源文件。Makefile 可用于许多环境，在 Win32 平台下，利用 MSVC 可以通过友好的界面生成和修改 makefile 文件，只是其语法习惯跟 Unix 和 GNU 下的有些细微不一样，可以参考 GNU 下的 makefile 规范和 MSDN。

第五节 NS2 仿真流程、基类和功能模块

2.5.1 仿真流程

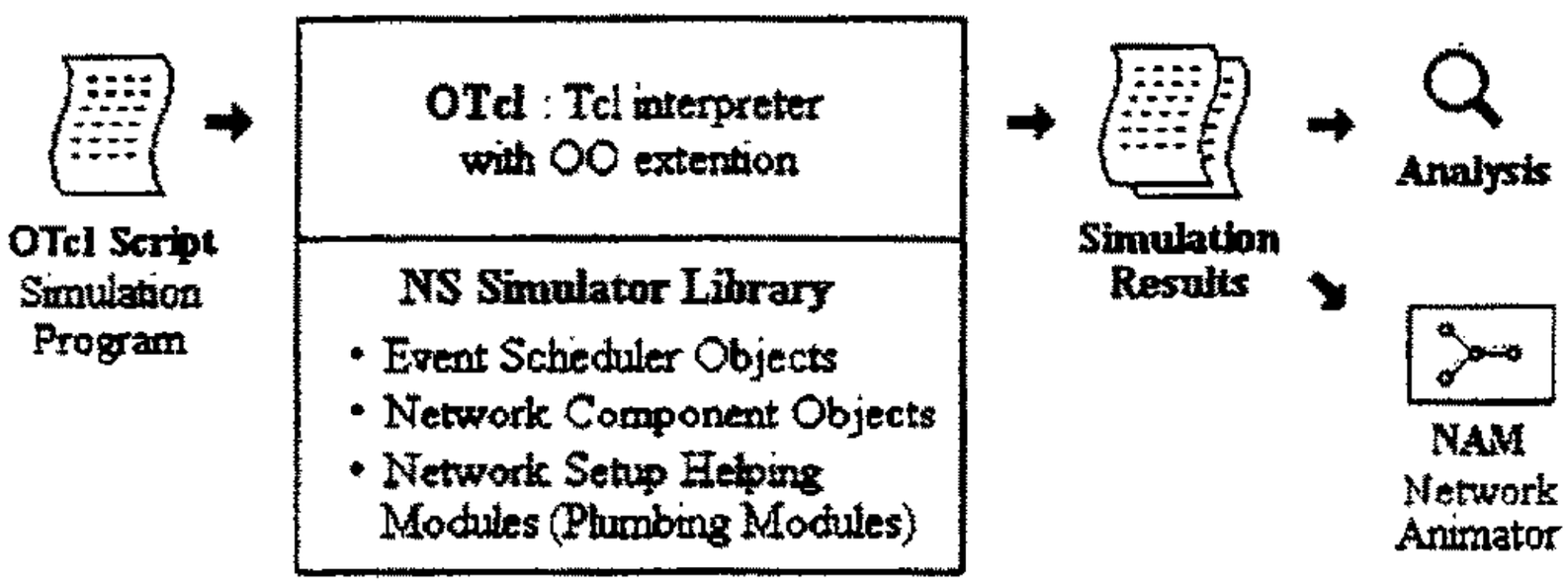


图 2-1 NS2 仿真流程

利用 NS2 进行仿真的流程如图 2-1 所示。对于已有的协议，先编写 Otcl 脚本，定制网络拓扑、业务模型等；NS 进行解释，其中要调用内置模块库，如网络组件等；NS 输出仿真结果，包括 tr 文件、nam 文件、out 文件，以及计算所得统计数据；最后可以对统计数据以及各种输出文件进行分析，也可利用 Nam 工具观看网络动态运行效果。

2.5.2 基类介绍

NS2 有六种基类，所有的 NS 组件都是由它们派生出来的，最终构成完整的 NS2 仿真平台。这 6 种基类是：Tcl 类、TclObject 类、TclClass 类、TclCommand 类、EmbeddedTcl 类、InstVar 类，下面对它们的最重要的功能作简单介绍。

2.5.2.1 Tcl 类

Tcl 类的主要功能有：

1）获得访问 Tcl 实例的入口：C++代码将整个 Tcl 部分视为一个类，必须首先获取访问入口，即用 Tcl& tcl=Tcl::instance()实现。

2）通过解释器调用 Otcl 过程，主要是调用四种 eval 方法实现。

3）实现 C++方法和脚本解释器二者间结果的交换，即 C++变量和脚本字符串之间的互相转换；

4）报告出错，并以统一方式退出，这通过 error 方式实现。

5）存储、查找 TclObject 类对象。NS 中的功能模块几乎都是由 TclObject 类派生来的（参见 2.5.3），它们之间的功能、结构不大一样，为了管理方便，NS 把每一个TclObject对象的入口都存储在hash表中，并以对象名为关键字对hash表进行操作。

6）获取解释器的句柄。

2.5.2.2 TclObject 类

TclObject 类是 NS 中决大多数类的基类，封装了绑定、跟踪和对相关命令的调用机制，主要功能包括：

1）生成（create）和释放（delete）该类对象。create 有一系列过程，实现生成相应的解释类、调用解释类的构造函数、初始化、入 hash 表等操作，delete 是其逆过程。

2）实现变量的绑定，以使解释变量和编译变量同步，包括显示和隐式两种方法，隐式绑定是在初始化对象时由编译对象的构造函数完成的。

3）变量的跟踪。

4）定义解释类的 cmd 方法自动调用编译类 command 方法的方式。

2.5.2.3 TclClass 类

TclClass 类是一个纯虚基类，派生层次简单，几乎所有的子类都不再有派生类，只有 PacketHeaderClass 派生出许多子类，以提供各种包格式。从 TclClass 派生的类需实现两个成员函数：构造函数，构造解释类层次来镜像编译类层次；create()函数，生成与之相应的 TclObject 对象。

2.5.2.4 TclCommand 类

TclCommand 类的作用是为解释器提供全局命令，它是一个纯虚类，需要派生类实现两个成员函数：构造函数和 command()函数。

2.5.2.5 EmbeddedTcl 类

用户对脚本~tclcl/tcl-object.tcl 进行修改，或是修改、增加 tcl/lib 的文件来对 ns 进行扩展。对于新文件的装载是由 EmbeddedTcl 类的对象来完成的。Tcl 脚本其实就是由 char 类型数据组成的文本文件，所以类 Embedded 的构造函数可以用 char*型指针指向脚本代码，并将此指针值赋于成员变量 code_。EmbeddedTcl 的定义中（~tclcl/tclcl.h）有一 load()成员函数。Load()用 Tcl::instance()获得解释器句柄，然后调用 Tcl::evalc，调用 Tcl 命令 eval，完成对代码的装载。

2.5.2.6 InstVar 类

类InstVar定义了显式实现绑定机制的方法。绑定过程bind()通常需要指定解释变量名和编译对象的成员变量的地址。基类InstVar的构造函数在解释器里创建一个实例变量，建立陷阱例程（trap routine），捕捉所有的通过解释器对变量的访问。当解释器有对变量的读操作发生时，触发陷阱例程，陷阱例程调用适当的get函数，从对应的编译对象获取成员变量的当前值，并给解释变量赋值。对写操作，只是触发时间略有不同，在解释器写完解释变量之后，触发陷阱例程，陷阱例程调用适当的set函数，将改变后的值写回编译对象。

2.5.3 功能模块介绍

NS 的组件体系结构如图 2-2 所示。

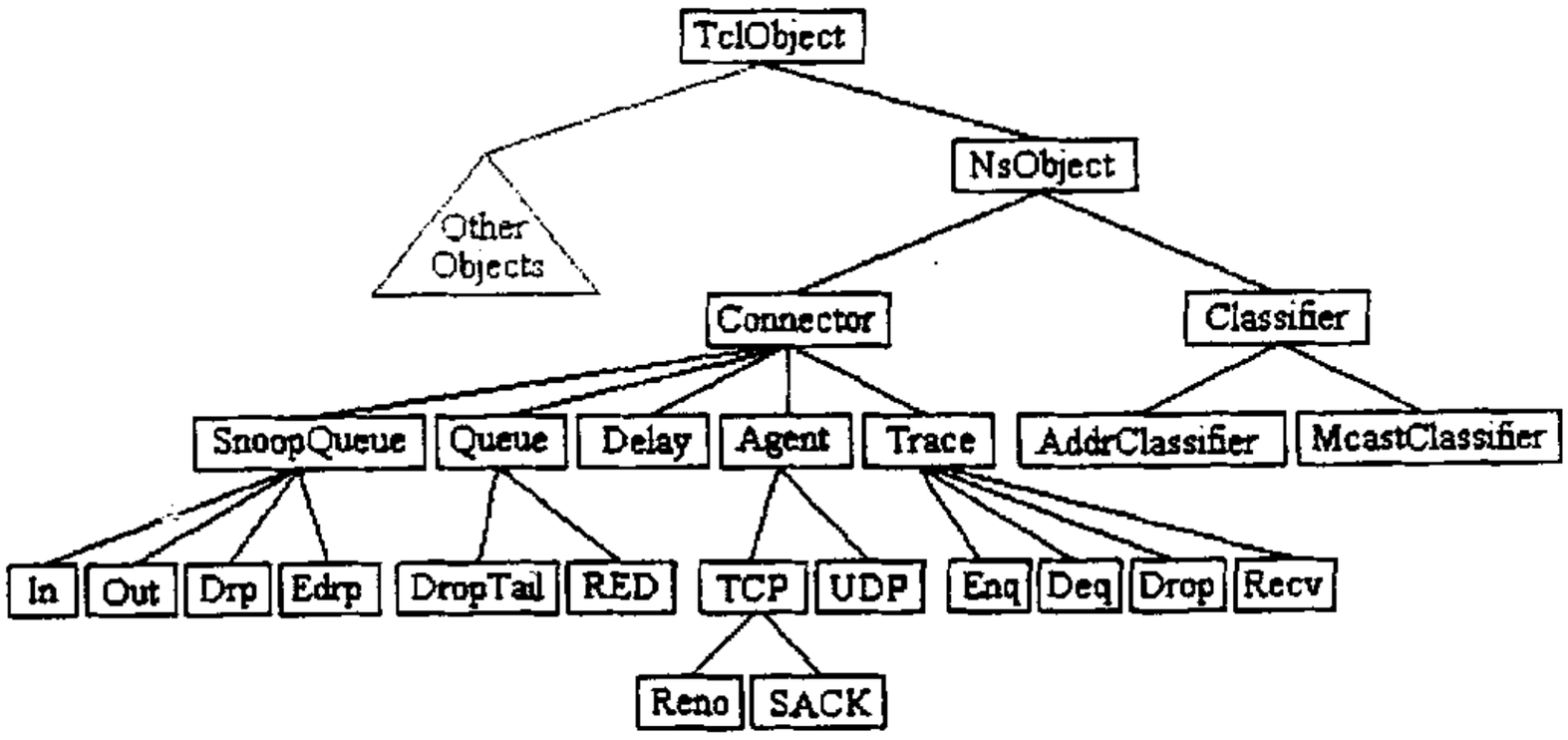


图 2-2 NS2 组件体系层次结构

2.5.3.1 Simulator 类—仿真的开始

Simulator 类是一个解释类，没有相应的编译类，但组成它的许多子类有相应的编译类。从 NS 外部看来，整个的仿真过程可以看成对 Simulator 的操作，因此，仿真工作从创建一个 Simulator 的实例对象开始，之后，通过这个 Simulator 调用各种方法生成节点，进而构造拓扑图，对仿真的各个方面进行配置，定义事件，然后，根据定义的事件，模拟整个网络活动的过程。

总的说来，Simulator 提供了三类方法：创建、管理拓扑(通过管理节点，和链路来实现)，跟踪，协调函数与 scheduler。

仿真器封装了许多功能模块，最基本的是节点、链路、代理、数据包格式等等。下面分别是这些模块的解释。

2.5.3.2 Node 类

节点是 ns 最重要的模块，其实是由分类器（classifier）构成的。节点也是一个由 OTcl 实现的解释类，也没有相应的编译对象。节点分为单播节点和多播节点，它们的内部构造如图 2-3 所示。

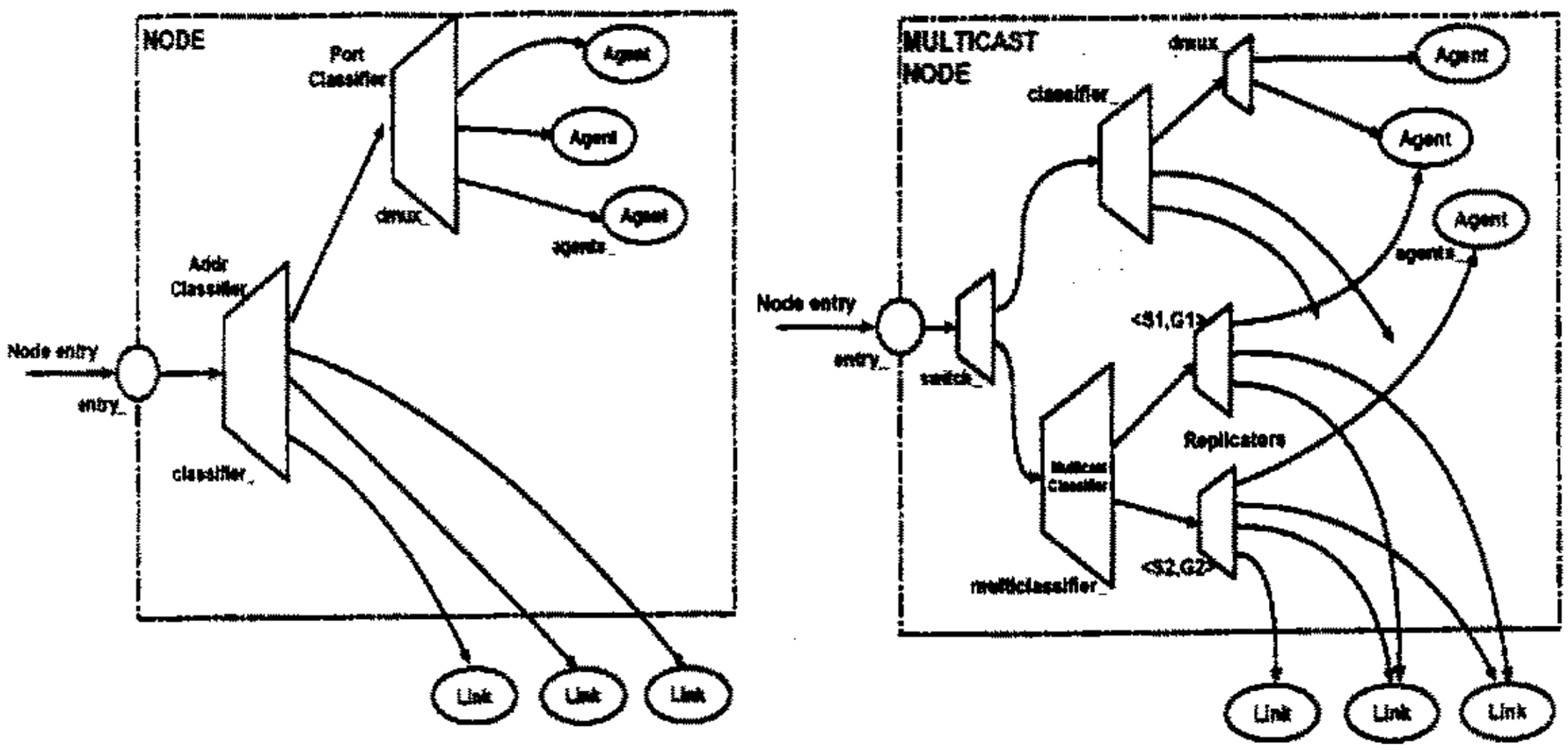


图2-3 单播和多播节点结构

节点的功能是接受包，检查它然后映射到相应的出口。一个node的组成包括几个classifier（分类器）对象，每个classifier完成特定的功能（比如处理包的特定部分），然后转发到下一个classifier。agent（代理）是node的另一类组件，它位于网络的端点，是产生和吸收包的模块。使用者用agent类建立业务的源和宿，当前NS支持各种TCP、CBR、UDP业务和RTP、RTCP、SRM协议，但不支持ATM协议。

2.5.3.3 Classifier 类

node接收到一个包后，要检查特定的域，通常是目的地址（还有可能是源地址），决定如何处理这个包，如果是转发，则还要决定转发到哪里。这项任务由分类器Classifier对象来完成，不同类型的分类器对象检查包不同的域以决定如何转发，共有五种分类器：base，address，multicast，multipath，replicator。

实质上，节点其实是Classifier的集合，最主要的功能是路由。最简单的单播节点，仅包含一个地址分类器和一个端口分类器。在节点内加入更多的分类器，就能扩展节点的功能，多播节点就是个例子。当节点中的分类器越来越多，就需要一个统一的接口来组织这些分类器，也就形成一个路由模块。路由模块由三个功能子块组成：

- **Routingagent** ：与相邻节点交换数据包；
- **Routelogic** ：利用routing agent收集到的信息确定路由，对于静态路由，还可以根据全局拓扑进行路由计算；
- **Classifiers** ：根据路由计算的结果，实现数据包的转发。

目前，ns实现的路由模块有以下几种：

- **RtModule/Base** ：面向单播协议，提供最基本的功能，如添加/删除路由，连接/断开agent；
- **RtModule/Mcast** ：面向单播协议，只建立multicast classifiers；
- **RtModule/Hier** ：多层次的路由，可以和其他的路由协议相结合，比如ad hoc路由；
- **RtModule/Manual** ：Manual路由。
- **RtModule/VC** ：用虚拟classifier取代vanilla classifier。
- **RtModule/MPLS** ：实现MPLS功能。

2.5.3.4 Link 类

link是与网络时延和带宽相关的类，它由一系列connector对象构建，包括head、queue、TTL和drop。connector接受一个包，进行相应的处理，然后发送到下一个connector或者是交给drop处理。NS支持点到点、广播、无线等各种链路类型。链路的构造如图2-4所示。

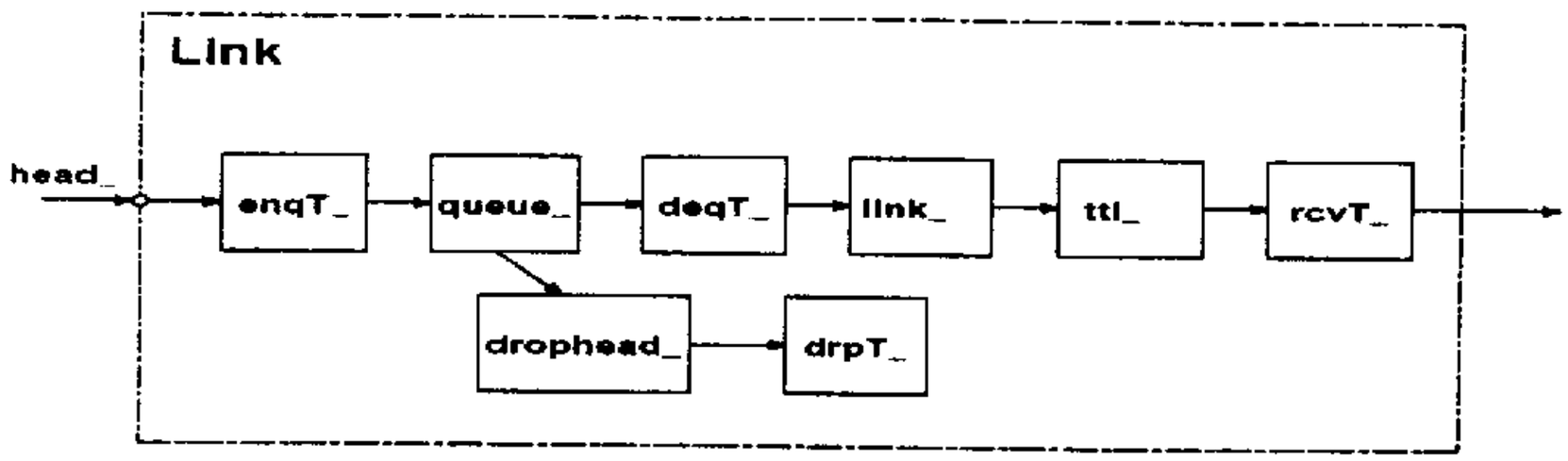


图2-4 链路结构

2.5.3.5 Connector 类

连接器Connector的主要功能是：接收数据包，进行一些处理，然后转发给下一个connector，或者drop处理。Ns中有不同类型的连接器，每一种有它自己独特的功能：

- ✧ networkinterface：为接收到的包标注上接口id；
- ✧ DynaLink：根据链路状态（连上/断开），决定是否传输数据；
- ✧ DelayLink：模拟链路的延迟、带宽。如果不是动态链，则在接收到数据包后，经过一定时间后，发给下游节点；若是动态链，将接收到的数据包加入排队过程，然后发送出去。若在某一时刻链路断开了，调用reset()，丢弃所有的数据包；
- ✧ Queues：模拟链路的输出缓冲；
- ✧ TTLChecker：检查数据包的ttl值，若是正值，发送到链路的下个元素。

2.5.3.6 Queue 类

在NS中，实际路由器中与链路相关联的输出缓存被模型化为queue(队列)，它是link的一部分。NS支持各种包调度算法和缓冲管理的仿真。包的调度指决定哪个包被服务、哪个包被丢弃的过程。缓冲管理指控制管理队列的规则，C++类提供了包括drop-tail(FIFO)队列、RED(Random Early Detection)缓存管理、CBQ(Priority and round-robin)，WFQ(Weighted Fair Queuing)、SFQ(Stochastic Fair Queuing)和DRR（Deficit Round-Robin）。

2.5.3.7 Delay 类

延迟类计算发送一个数据包所需要的时间，一些特殊的派生类甚至还封装了对链路失效的处理。发送一个数据包的基本延迟计算式为：s/b+d，其中s为数据包的长度（从IP包头部得到），b代表链的速率，d为链路的传输时间。

考虑 Delay 类时，需要解释一下动态链与非动态链的区别：假设某一时刻链路

接收到一个数据包 p，则它将在适当时间触发两个事件，E1 和 E2，E1 为对上游节点的回叫，E2 为下游节点接收数据包，E1 先发生。非动态链中，一个时刻最多只有一个数据包传输，所以 E2 与 E1 的时间间隔为定值。但在动态链中，一个时刻可能会有几个数据包等着发送，此时，后到达的数据包将加入队列，等待发送，因此 E1 与 E2 的时间间隔不定。这一点可以参见在第三章中关于如何判断两个包同时到达的方法。

2.5.3.8 Packet 类

类Packet是仿真对象间交换数据的基本单元，它提供了足够的信息，包括包头和数据缓冲。不同的协议采用不同的数据包格式，因此包头会不一样。要引入新的数据包，需要根据数据包的格式定义一个C++结构，定义一个静态类来提供与Otcl的连接，然后修改仿真器的初始化代码。当Otcl解释器初始化仿真器时，用户可以只激活部分内置的数据包格式，以节省内存。对各种包格式的管理，是通过一个特殊的“数据包头管理员”对象实现的。

2.5.3.9 Scheduler 类—仿真引擎

仿真引擎是可扩展、可配置、可编程的，当前的实现是单线程的（同时只能执行一个 event），也不支持 event 的部分执行或者预占机制。

event（事件）是用启动时间和处理函数来描述的。驱动仿真的 event scheduler 的类型可以是现有的以下四种类型之一：简单链表、堆、日历队列和特定的实时类型，每一种都对应着自己的数据结构。下面分别予以简单介绍。

- 链表调度器提供了一列时间列表，按时间先后顺序排列。这就需要扫描链表来找出合适的入口（entry）以决定插入或者删除。位于头部的 entry 总是首先被执行，要求同一时间仿真的所有 entry 将根据它们在链表中的顺序来提取。
- 堆调度器基本功能与链表调度器一致，只是在处理大量事件时性能要好一些，表现为所用时间更少。
- 对日历队列，只不过是堆调度器中一些年份的相同月和日的事件被记录在同一天中等待处理。
- 实时调度器是链表调度器的一个子类，它将以较低速率到达的事件重新组织好，然后实时执行。

第六节 新协议的添加

在 ns 包的 conf 子目录下有针对 windows 平台和 VC 环境的特定配置文件 makefile.win，里面对适合 windows 平台和 VC 的路径、环境变量以及大量的宏做了定义，并在 makefile.vc 文件中用包含指令包含了进去，在 VC 下对该文件调用 make 命令，便可得到在 windows 平台下、可利用 VC 进行调试的 NS2 仿真平台。

NS2 平台原本已经包含有一些常用的网络协议，如 mac 协议、mpls 协议、ip 协议等，在编译完成之后，即可得到 ns.exe 文件，编写 tcl 脚本进而运行与这些协议相关的仿真模型。

也可以添加上自己的协议，重新编译项目后就可以象自带的协议一样运行。下面简要说明添加一个新协议的基本思想。

添加一个新的协议包括对头文件的定义、C++和 Tcl 模块的建立，以及其他更改和编译。头文件的定义部分主要完成对新协议的包头格式和数据结构的定义，以及对数据源 Agent 的定义。在 C++模块中，主要定义数据包的初始化、产生、接受、处理、控制等功能组件，同时还要定义 C++和 Tcl 之间的接口。Tcl 模块除了定义建立网络模型所需的接口外，还可以进行部分运算，如统计等。然后还需要将新的协议号及信息添加到 packet.h 文件中，并在 ns-default.tcl 文件中为 tcl 对象赋默认值和访问入口。最后更新 makefile.vc 文件，编译后即可得到自己的新协议。

较为复杂的协议的添加过程的基本方法与此类似，可以参见已完成的 DEFLECTION 和 CRMA 部分。

参考文献

[1] Kevin Fall，Kannan Varadhan, "The ns Manual", April 14，2002

[2] Bo Wen, Nilesh M. Bhide, Ramakrishna K. Shenai, and Krishna M. Sivalingam, "Optical Wavelength Division Multiplexing (WDM) Network Simulator(OWns): Architecture and Performance Studies"

[3] 李国庆，"NS2 在 Windows2000 及 MFVC6.0 下的编译和调试"，2003 年 5 月

[4] Richard M. Stallman, Roland McGrath, "GNU Make-A Program for Directing Recompilation, GNU make Version 3.79", April 2000

[5] "The GNU Make Manual", last updated 08 July 2002, documents GNU make Version 3.80,

http://www.gnu.org/manual/make.html

[6] 李培源，“Linux/Unix 环境下的 make 和 makefile 详解”，2000/10/23

[7] Kongming，“Linux 下的编程”，bbs.tsinghua.edu.cn, Aug 2,1999

[8] Bruno Poupier, “Make-a Tutorial”, http://www.eng.hawaii.edu/Tutor/Make/FrenchMake/make.htm

[9] Nicolas Christin, “Building ns-2 on Cygwin [versions 2.1b9a and 2.26]”

[10] John K. Ousterhout, “Tcl and the Tk Toolkit”, Computer Science Division Department of Electrical Engineering and Computer Sciences University of California Berkeley, CA 94720, published in 1994 by Addison Wesley

[11] Paul Raines, Jeff Tranter, “Tcl/Tk Reference Guide”

[12] David Wetherall, Christopher J. Lindblad, “Extending Tcl for Dynamic Object-Oriented Programming”, Telemedia Networks and Systems Group Laboratory for Computer Science Massachusetts Institute of Technology

[13] David Wetherall, “Otcl – MIT Object Tcl, the FAQ & Manual (Version 0.96, September 95)”, MIT Lab for Computer Science

[14] Marc Greis, “ns Tutorial”, ftp://ftp.tns.lcs.mit.edu/pub/otcl/README.html

[15] 陈文革，钟雪慧，梁洁，刘志华，伍春华，“仿真技术及其应用”，广东省电信科学技术研究院

[16] Lee Breslau, Polly Huang, ect., “Advances in Network Simulation”, edited by VINT project

[17] Shkumbin Hamiti, “Building and evaluating network simulation systems”, Nokia Research Center, HUT 06.02.2001

[18] Atika COHEN, Radouane MRABET, “Generic Simulation Models of Communication System”, University Libre de Bruxelles

[19] Sandeep Bajaj, Lee Breslau, etc., “Improving Simulation for Network Research”, USC Computer Science Department Technical Report 99-702b, March 1999(revised September 1999)

[20] Anu Maria, “Introduction to modeling and simulation”, Proceedings of the 1997 Winter Simulation Conference, State University of New York at Binghamton, Department of Systems Science and Industrial Engineering

[21] Victor Frost, Benjamin Melamed, “Modeling and simulation for telecommunications networks”

[22] Thierry Ernst, “Notes about network simulation”, 30/10/1997

第三章 偏射路由算法的仿真与分析

第一节 偏射路由算法

在全光分组网络中，当多个的数据包要同时发往同一个出口时，就会出现竞争而导致丢包。其解决方法有：①增加光缓存器（optical buffer），采用类似电分组网络的存储转发方式。可以用光纤延迟线作为光缓存器件，此方案最为有效，网络吞吐量也较高，但目前光信号处理和光存储技术还不成熟，控制硬件复杂，并且由于各种限制，延迟时间最大只有几十微秒，对于长的突发分组不能适用。②增加波长路由器的波长转换（wavelength conversion）能力，但会带来很大的开销，并且波长变换只能在一定程度上改善系统的性能。③使用偏射路由算法（deflection routing）。这是目前较为可行的解决方案，只需要少量的存储器，甚至不需要存储器，使光缓存困难的问题得到缓解，其实质是将整个光网络作为一个大的光缓存器（OB）。该方案比较适合于有较充足链路资源的网状光网络，利用网状网具有迂回路由的特点在一定程度上降低了硬件的复杂性、提高链路利用率、降低网络拥塞。但是，偏射路由算法要求网络物理拓扑和逻辑拓扑是多径的且规则的（如 Shuffle Net 和 Manhattan Street Net），以降低网络吞吐量为代价，网络性能不高。本章的主要工作就是对光分组交换网络中偏射路由算法进行研究和分析。

偏射路由算法的基本原理是这样的：当无出口竞争时，按照 SPF（Shortest Path First）规则选择路由，所有的包均走最佳路由；当多个分组竞争同一个端口时，与存储转发路由算法不同，使用偏射路由算法时，只有一个数据包能从端口输出，走最佳路由，其它数据包则偏射到其他的空闲端口通过空闲链路输出，寻求尽可能短的次佳路由到达目的地，即并非所有的数据包都选择最佳路由。其实质是把整个网络看作一个大的光缓存器，通过空闲的链路来缓存受阻的数据包，增加数据包的平均跳数，以网络容量为代价，换取硬件的简单性。所以，在这种路由网络中，数据包所走的路由是随机的，数据包从源节点到目标节点过程中所经历的跳转次数也是随机的，其平均跳转次数高于存储转发路由算法，因此也导致了网络吞吐量的下降。

对于采用偏射路由算法的光分组网络而言，一般都要求有特定的拓扑结构，譬如每个节点的入端口数目要和出端口数目匹配，每个入端口进来的数据包都可以通过任意一个出端口到达目的节点（尽管不会都是最短路径），以保证从不同端口到达节点的数据流发生端口竞争时不会因缺少光域的存储而发生丢包现象。论文所研究的偏射路由模型只针对同步定长的光分组交换，即要求整个 OPS 网络的数据包的产生与接收都是严格同步的，节点才可以准确判断两个数据包的“同时”到达，从而进行偏射

路由的决策。

现有的一些用于偏射算法的拓扑模型包括Manhattan Street Network（MS）和ShuffleNet（SN）两种，其设计思想都满足上述要求，ShuffleNet由$N=kp^k$节点组成，这N个节点排成k列，每列有p^k个节点，最后一列又连回第一列，在逻辑上网络形成了一个圆柱型结构，这里p是每个节点的连接度，对于连接度为 2 的ShffuleNet，$p=2$。Manhattan Street Network由$N=n^2$个节点组成，这N个节点排成n行和n列，每一行和每一列都由单向的链路连接而成，相邻行和列之间链路的方向相反，这里的n为偶数。这两种网络都是规则网络，即从网络的任一节点看去，网络都是一样的，而且在任意两节点间有多条路径，这就保证了被偏射的数据包能最终到达目标节点。图 3.1 和图

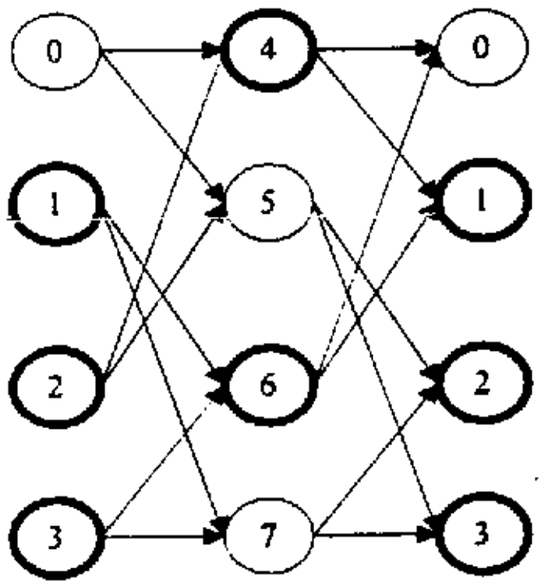


图 3.1 ShuffleNet

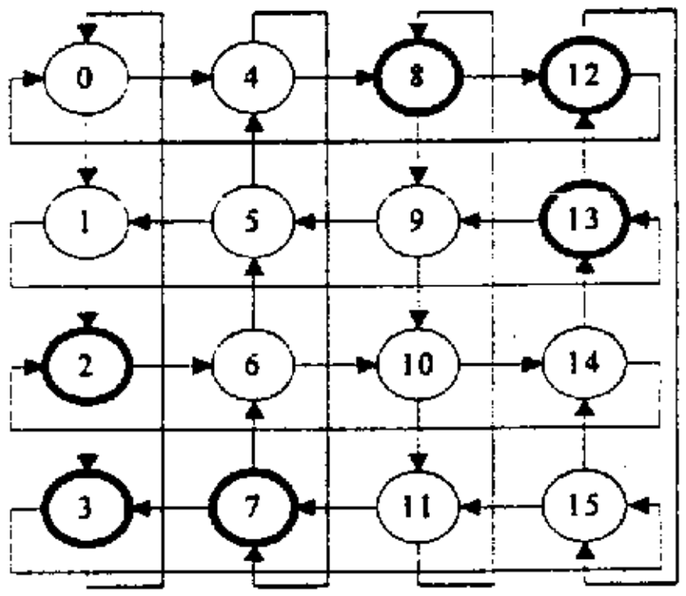


图 3.2 Manhattan Street Net

3.2 分别是连接度为 2（每个节点两入两出）的 8 节点SN与 16 节点MS网络。

对于采用偏射路由算法的网络，有几个网络结构参数对网络的性能有很大的影响：

1）网络半径：在网络的每对节点中，都有一个最短路径，这些最短路径的最大值称为网络半径。这个参数反映了网络的紧凑程度，网络半径越小，网络越紧凑。ShuffleNet的网络半径为 $2k-1$，当N很大时，趋近 $2log_2N$，Manhattan Street Network的网络半径为n。

2）偏射代价：一个数据包由于一次偏射所增加的最大跳转次数（每经过一个节点转接称为一次跳转）为偏射代价。ShuffleNet 的偏射代价为 k，是一个随着网络规模的增大而增大的量，而 Manhattan Street Network 的偏射代价为 4，是不随着网络规模的变化而变化的。

3）“无关”节点：网络中存在一些节点，这些节点的两条输出链路到目标节点的距离相等，被称为“无关”节点，别的节点称为“相关”节点。“无关”节点的存在，大大减少了数据包被偏射的概率（在无关节点，数据包从任何一个输出链路输出的效果是一样的，因此不存在偏射的问题）。图 1、图 2 中画粗线的节点就是节点 0 的“有

关”节点，其中 SN8 中的无关节点数是 2，占中间节点数（不包含节点 0）的 28.6%，MS16 的是 9,占 60%。

综上所述，偏射路由有以下特点：

网络总容量与平均光通道跳数以及流量负荷成反比，与节点数、平均链路容量成正比；

无关节点越多，则网络直径和偏射成本越小；

易造成额外的旁路流量，使网络吞吐量下降。

因此有必要确定通道跳数的上限，如果引入有限的 buffer，会取得较好效果；多用于规则拓扑，不适用于异步 OPS（异步则竞争概率大，则偏射流量多，吞吐量急剧下降，甚至网络瘫痪）。

偏射路由协议实际上可以看作是同步 OPS 网络的具体实现方式，而 OPS 本身就是 OTDM 网络的一种，因此对偏射路由机制的研究正是对 MESH 网结构下 OTDM 网络协议的研究。

第二节 偏射路由算法的仿真实现

这部分首先对 ShuffleNet 模型进行分析，然后由此提出改进的拓扑模型。对核心节点数为 8 的 ShuffleNet 网络，连接度为 2，拓扑结构规则，每个节点都有两条输入链路和两条输出链路。在仿真中，采用源节点和边缘节点分离的网络结构，其逻辑拓扑和实际拓扑分别如图 3.3 和 3.4 所示。

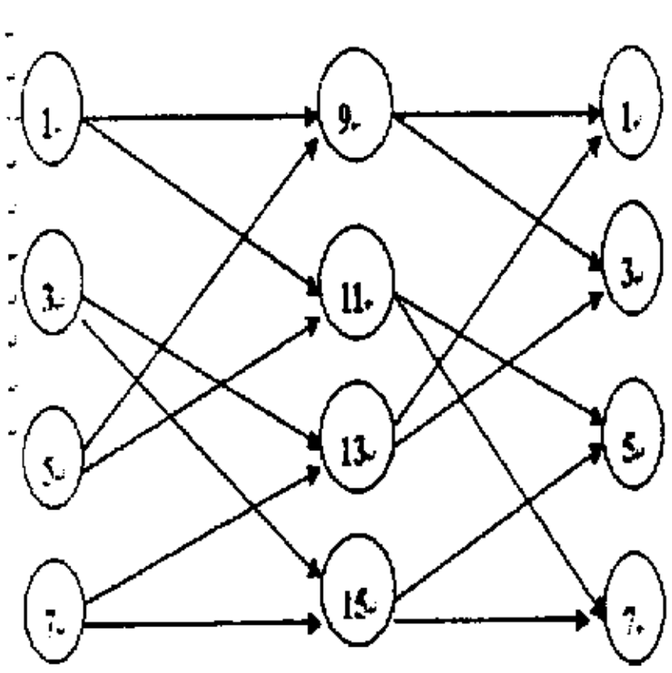


图 3.3 逻辑拓扑

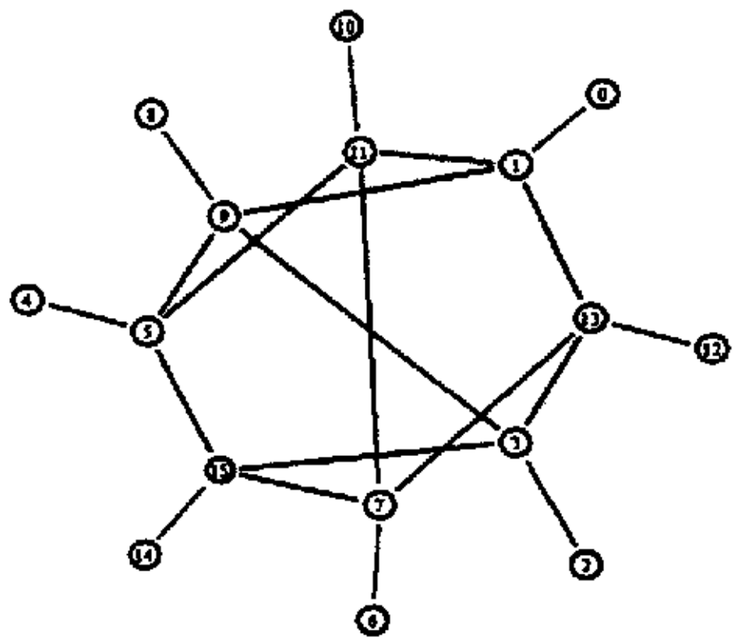


图 3.4 实际拓扑

编号为偶数的节点为源节点，完成的功能仅限于数据包的产生和吸收，构成接入网部分；奇数节点为核心节点，不产生数据，仅仅完成数据包的转发，构成核心网部分。

3.2.1 节点结构和仿真流程

基本原理

仿真模型中最重要的是节点结构的实现，从原理上，其基本功能模块包括四部分，如图 3.5 所示：

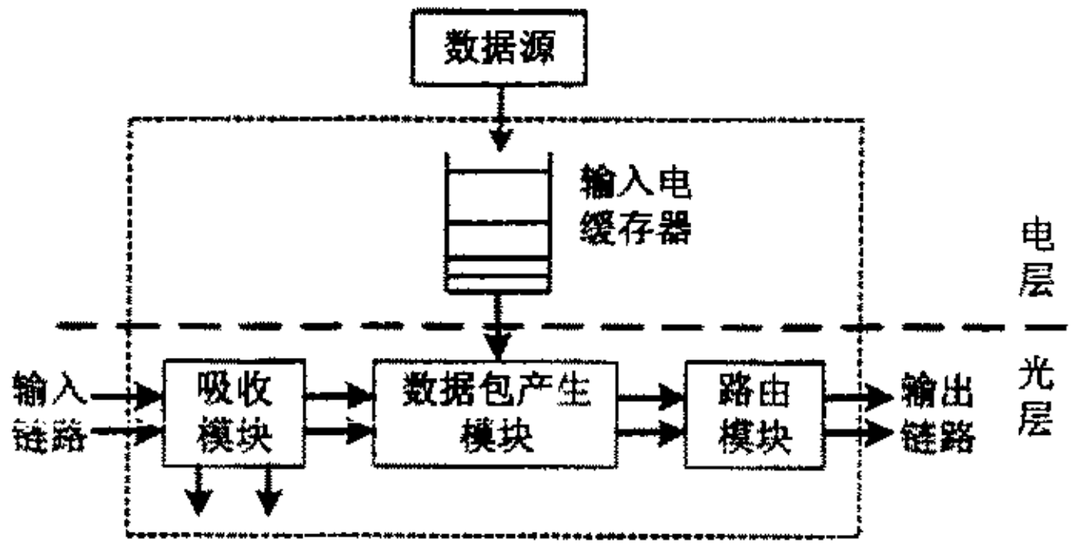


图 3.5 网络节点的结构框图

1）吸收模块：发往该节点的数据包被吸收模块吸收或向后转发。

2）数据包产生模块：如果在输入存储器中有新数据包准备发送，而数据包产生模块中有空闲链路，则数据包可以插入。从接入策略来看，这是一个拒绝系统。

3）路由模块：数据包根据规定的路由算法从相应的链路输出，如果两个输出数据包竞争同一输出链路，则执行偏射路由算法。

4）输入存储器：本地产生的流量先存储在输入存储器中，直到数据包产生模块判断有空闲链路后，数据包才能依次插入网络。

仿真模块

在实际仿真实现时，吸收模块、数据包产生模块和路由模块都合并到一个核心处理模块中了，如图 3.6 所示。

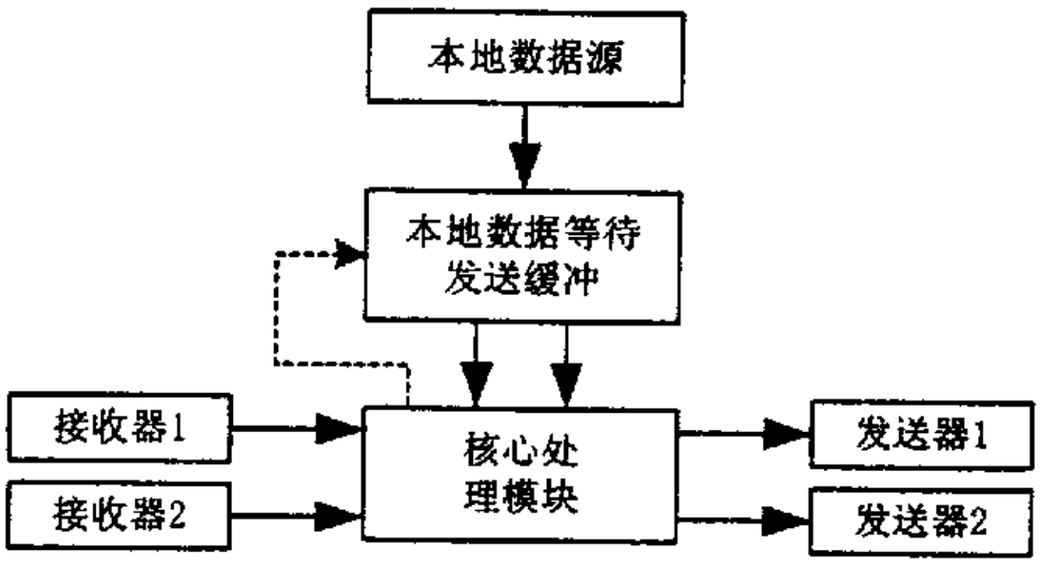


图 3.6 仿真节点结构图

在 NS2 平台上，核心处理模块是由分类器（classifier）构成的，原理图如图 3.7。

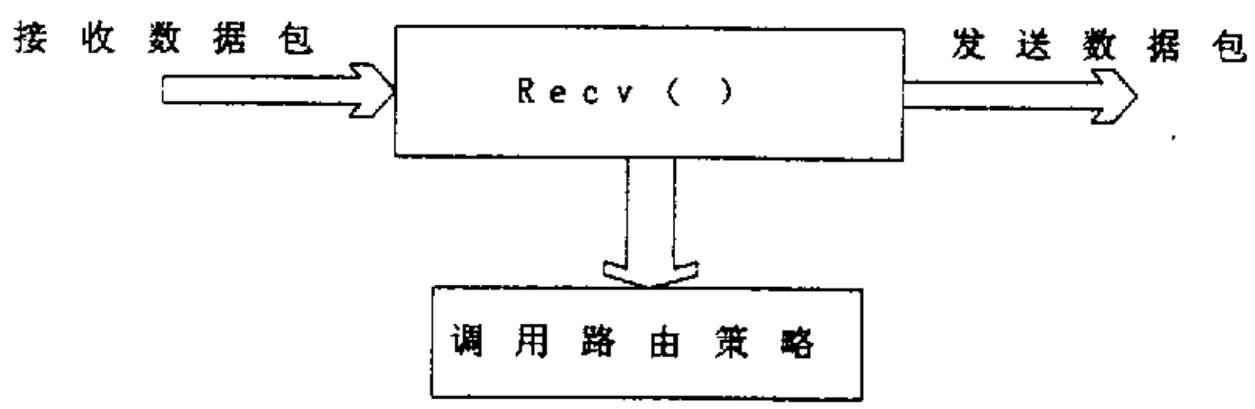


图 3.7 classifier 的接收机制

在这个节点模型中，本地数据由数据源产生，其产生速率、分布模式和分组格式都可以灵活设定，产生的数据在本地缓存中缓存起来，由核心处理模块决定数据分组是否可以进入光层网络。

网络同步与包同时到达

在仿真过程中，最重要的是通过一定的机制保证仿真网络能模拟实际网络中的同步机制，这一点在 NS2 平台上是通过本地节点的 scheduler 类和引入相应的时延来实现的。为了使问题简化，在仿真中，数据包的大小、节点处理时间、链路传输延迟均设为一样，这样只要所有节点的数据包发送能达到同步（这通过 scheduler 控制实现），那么所有的包到达下一个节点时也能达到同步。

当节点收到数据包时会检测包头的目的地址域，然后将此值映射某输出端口，对应于数据包路由的下一个节点。要实现两个数据包竞争时的偏射机制，首先要使节点能够判断两个包的同时到达。NS2 最基本的机制是事件驱动，数据包到达的事件触发实现接收功能的函数（Recv()），并且在接收函数中完成到目的对象的转发，即便是仿真时间中同时到达的包也以一定的顺序先后处理。在不能修改接收函数触发机制的前提下，要让其接收到数据包但却不会立刻转发，而要等到下一个接收功能被数据包触发后，将得到的两数据包信息进行比较运算后同时转发。所以在一个接收函数中

将触发的事件延迟十分之一时隙长度后，写入 Scheduler，这样当节点需要做决策进行数据包的转发时，两个接收函数均已发生，需要的数据包信息也已被存储，此时调用路由策略来完成数据包的转发。

路由策略

在路由模块中，所有的路由信息只是到达某节点的包的下一跳信息，先忽略本地吸收和产生包的部分，那么每个节点就都是两进两出。与普通路由不同，偏射路由每到一个节点都有可能由于出端口竞争发生偏射，从而不得不重新选路，所以对于每一个有源地址和目的地址的包而言，每到达一个节点，就可以将该节点看作新的源节点，目的节点不变。对于这个新的源节点，到达目的节点按照 dijkstra 算法，有一条最短路由，在每条链路情况一致的条件下，就是跳数最少的路由，另外一条出口则是该包的备选路由也就是次佳路由。在确认两个包同时到达之后就可以调用偏射路由算法，首先判断两个包的下一跳地址是不是一样，也就是是否存在冲突，不存在冲突就可以使用普通的 classifier 转发，存在出口竞争时，考虑在一些节点，它们的两条输出链路到目标节点的距离相等，被称为“无关”节点，别的节点称为“有关”节点。从而可以确定相对于到达的两个包的目的节点，所处的当前节点是否为无关节点。两个包中只要存在一个或两个无关节点的包，就可以将那个无关节点的包偏射，而不会影响总跳数。如果两个包都处在有关节点，则可以比较分别用偏射方案时，执行偏射算法之后的总跳数之和，取较小的一种情况作为最终偏射方案，然后即可在 classifier 中进行相应的转发了，而这种判决准则正是对原偏射路由协议的优化与改进。

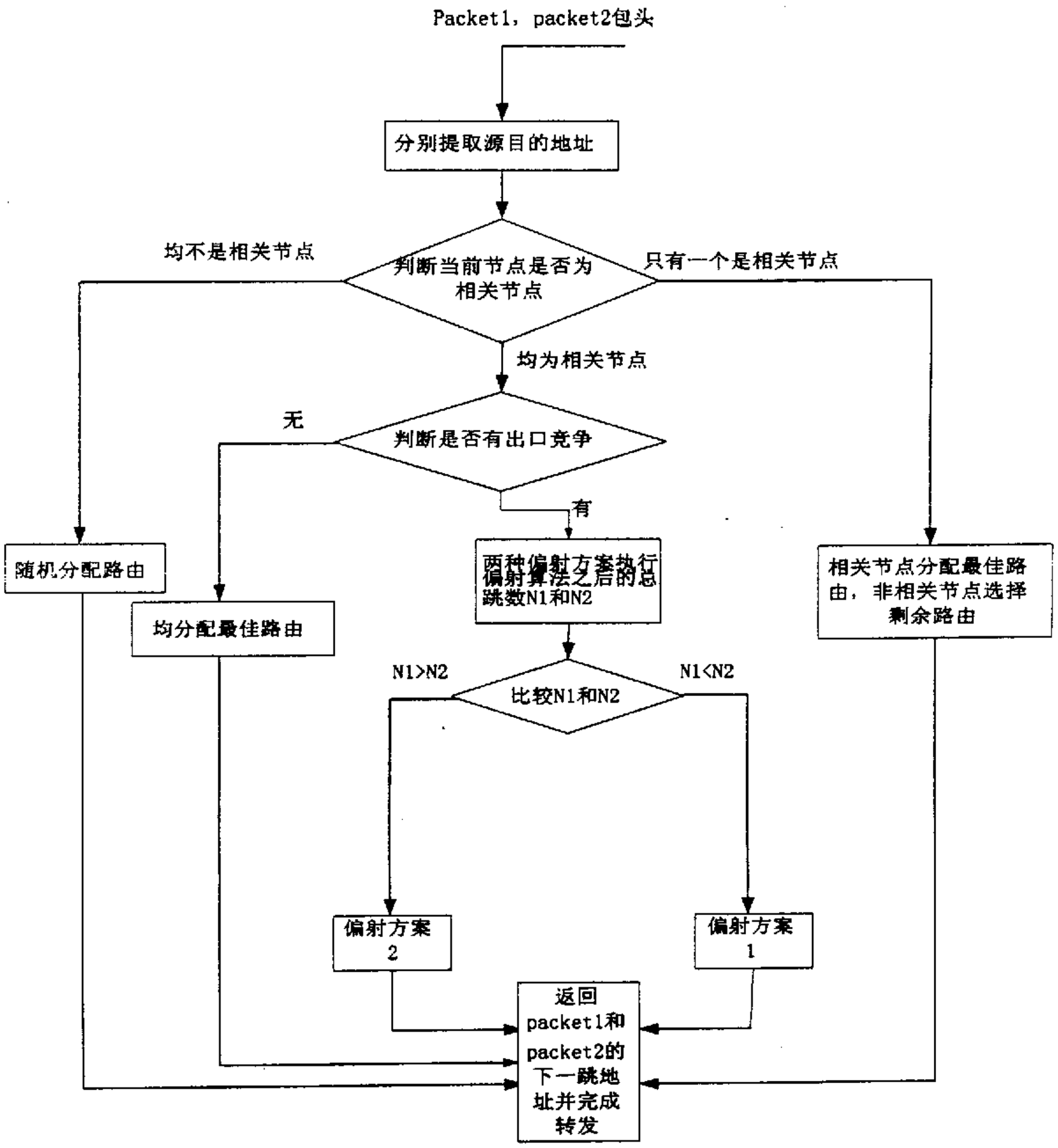


图 3.8 偏射路由决策流程图

在上图中，N1 代表包 1 输出到端口 1、包 2 输出到端口 2 时偏射（方案 1）的总跳数，N2 代表包 1 输出到端口 2、包 2 输出到端口 1 时偏射（方案 2）的总跳数。

其他模块

除节点设计外，还要对包头、链路、业务代理（Agent）、业务流（Application）、业务类型等进行设置，具体的在拓扑模型部分说明。

3.2.2 ShuffleNet 模型的仿真分析

图 3.9—3.13 分别给出了偏射与存储转发的总跳数比较、偏射与存储转发平均跳数比较、偏射与存储转发链路利用率比较、偏射总的包延迟和偏射平均包延迟。图中，每个数据点代表一种业务模型(即某时间段内由一些节点向另一些节点发送数据包的过程的统计值构成一个坐标点)。

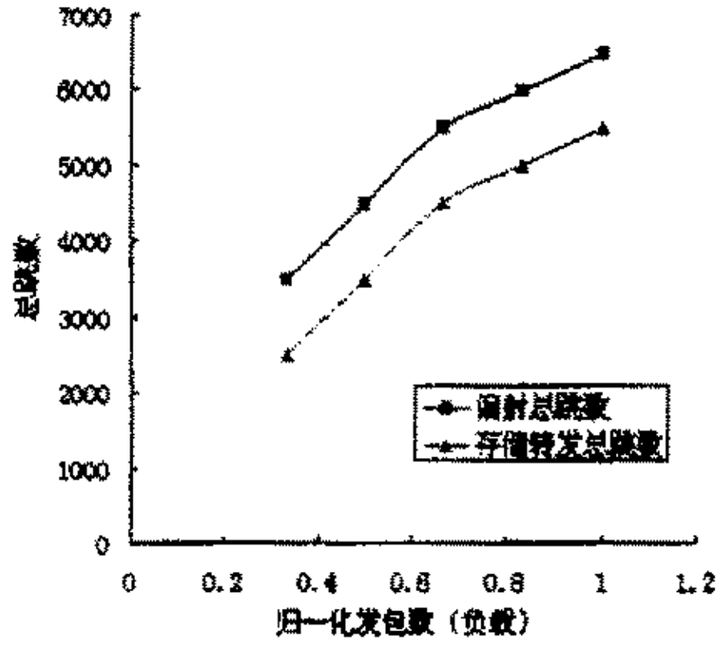


图 3.9 偏射与存储转发的总跳数比较

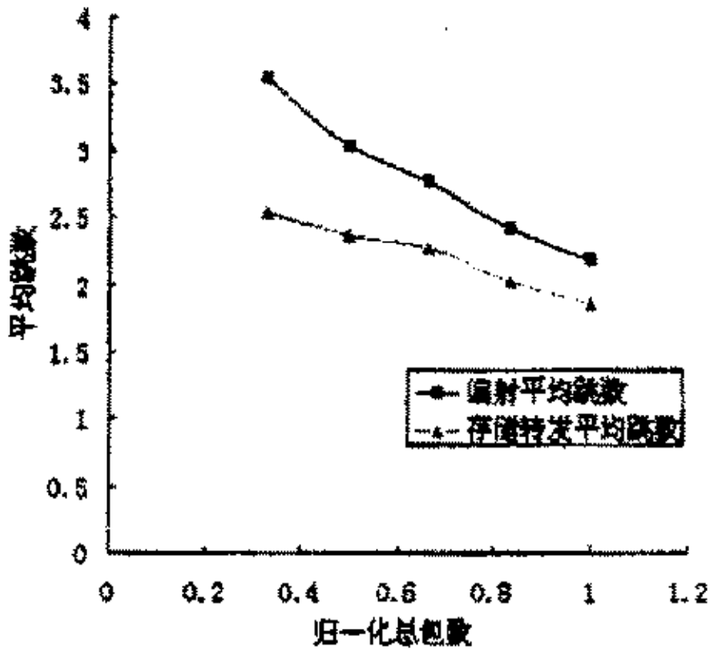


图 3.10 偏射与存储转发平均跳数比较

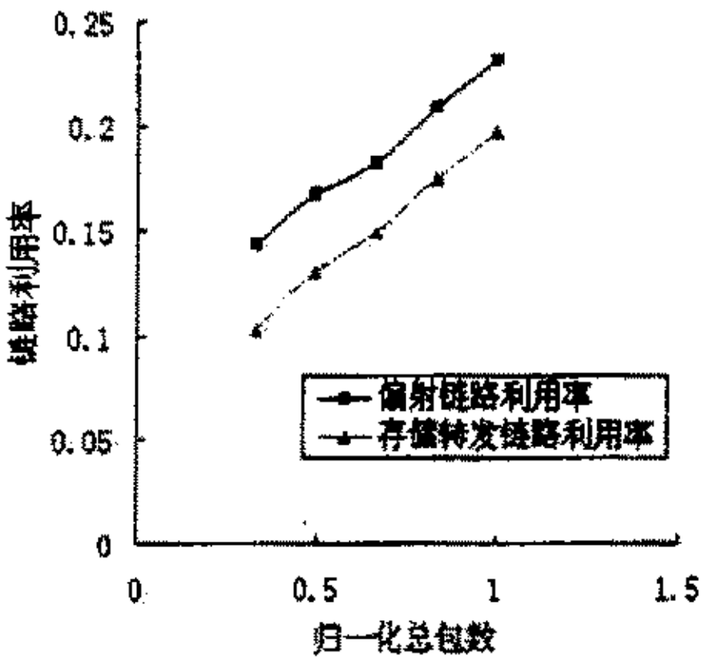


图 3.11 偏射与存储转发链路利用率比较

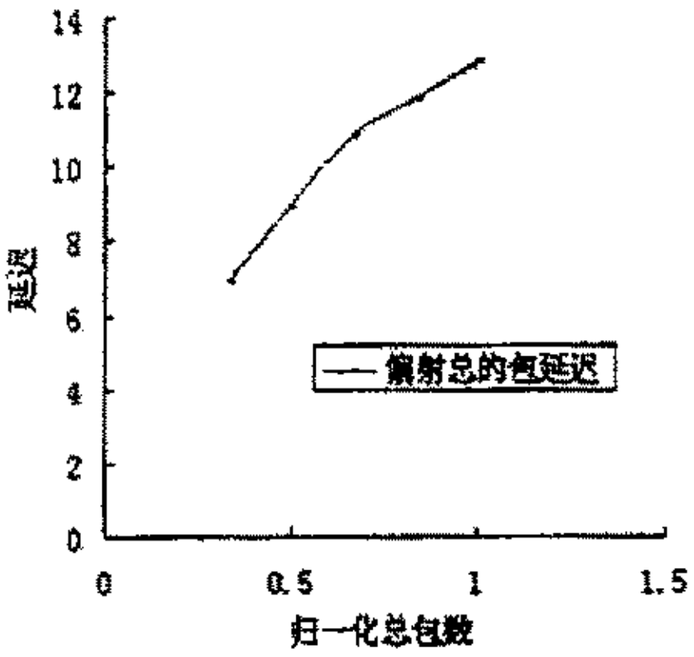


图 3.12 偏射总的包延迟

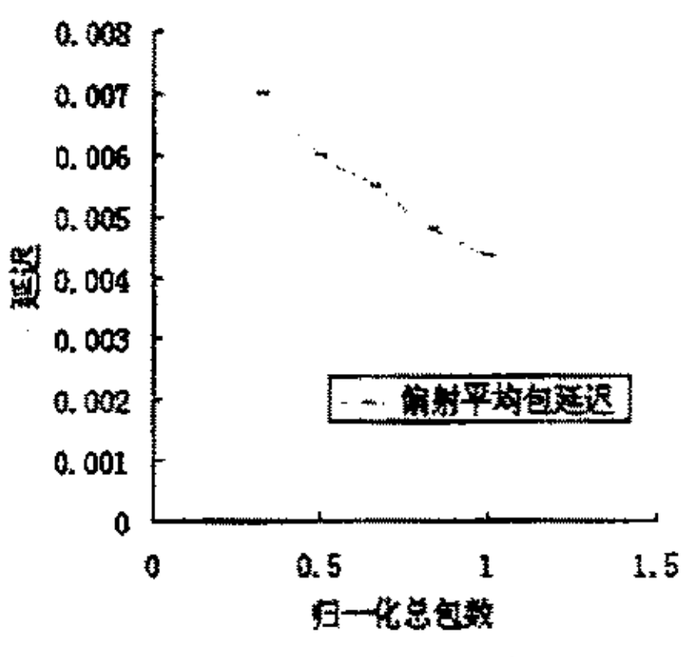


图 3.13 偏射平均包延迟

由统计结果可以看出，偏射情况下数据包的总跳数、平均跳数都较存储转发模式

下的要高一些，进而网络的链路利用率得到提高。这是因为当遇到输出链路冲突时，存储转发模式坚持走最佳路由，直到一个包转发完后另一个包才又继续由该冲突链路转发，而偏射算法此时则会将其中一个数据包从另一个空闲的链路输出（具体偏射方式由偏射路由策略决定），这就可能造成数据包的迂回，跳数增加，但这却利用了空闲的链路，使得链路利用率得到提高，一定程度上避免了一条链路上过于拥塞而另一条链路又空闲的情况。另一方面随着发包数的增加，包的总延迟增加了，但平均延迟却下降了。总延迟的增加是因为总的被偏射的包的数量增加了，而平均延迟降低则说明总的包延迟的增加慢于总包数的增加，即负载在一定范围内增加时，偏射机制能很好的均衡网络各部分的负载。

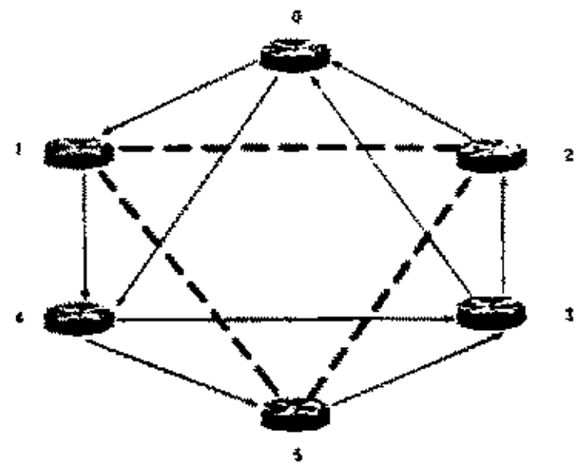
循环路由现象

在某些业务模型下，SN8 偏射时出现了循环路由现象：被偏射的包的路由出现死循环，反复回到发生偏射的节点，无法到达目的节点。参看图 3.3、3.4 对此现象作简单说明。

例如如果节点 6 和节点 2 如果同时给节点 5 发包（相当于节点 3 和节点 7 同时向 5 发包），必然在节点 15 发生冲突，其中一路数据又被偏射回 7 节点，此后只要 15 节点仍存在冲突，此数据流就会往返于节点 7 与节点 15 之间，造成路由的循环，当数据包中的 TTL 值到零时，会发生丢包现象，这一点在 NAM 图中可以被观察到。事实上，由于是全同步网络，被偏射回 7 节点的包必然与从节点 6 直接发出的包在 7 节点发生冲突，从而在 7 节点也会发生偏射，被偏射的数据选择节点 13 为偏射后的下一跳地址，为到达节点 5，再下一跳又为节点 3，从而在节点 3 与节点 13 之间再次出现循环路由现象。经分析原因是两方面的：一是采用延迟来达到两个包同时到达的机制有关；二是由于 SN8 中“有关”节点太多，比例太大，且两个节点就可以构成一个循环，即很容易就造成循环。为克服这个问题，论文提出了新的拓扑模型（见下一小节），用于 OPS 网络中的偏射路由算法，并对新的拓扑模型进行了详细的仿真分析。

3.2.3 改进网络拓扑后仿真与分析

改进后的新拓扑模型如图 3.14 所示：在该拓扑中，“无关”节点的比例非常大，并且要构成一个路由的循环至少要三个节点参与，这就从概率上很大程度的减少了发生循环路由的可能性，克服了 SN8 的困难。下面就以此拓扑为基础说明偏射算法的仿真情况。



DEFLECTION TOPOLOGY

图 3.14 改进后的拓扑图

1、topology 和 sessions

topology：两个正四棱柱扣在一起，每一条棱为一条单向链路；每一个顶点为一个两入两出的节点；所有链路均为带宽 8Mb，延迟 0.9ms；节点处理延迟是 0.1ms；

sessions：　cbr session,DropTail queue；

packet size：1000byte；

interval：　0.001s；

rate：　8Mb。

2、业务说明

0.01 秒：4-0：4—3—0，用黑色表示；

5-0：5—2—0 or 5—3—0，用红色表示，两条均为最佳路由；

0.02 秒：4-2：4—3—2 or 4—5—2，用蓝色表示；

根据业务模型，将在 4、5、3 连续发生偏射并重新选择路由。

3、偏射现象说明

0.01 秒时开始发包：4—3—0，5—2—0；

0.02 秒时 4 开始给 2 发包，4—3 链路被占用，将发生偏射，结果是原来经 4—3—0 的路由被偏射为经过 5 了；接着在节点 5，原本经 5—2—0 的路由被偏射为经过 3 了；最后在节点 3 再次发生偏射，只不过从现象上刚好跟不偏射情况下的一样（即执行了偏射算法但看不出现象）。最后，执行偏射算法后的路由情况为：

4—0：4—5—2—0

5—0：5—3—0

4—2：4—3—2

对 NS2 生成的统计文件用 NAM 进行处理可直观的观察到偏射动态效果图，

如图 3.15 示意。

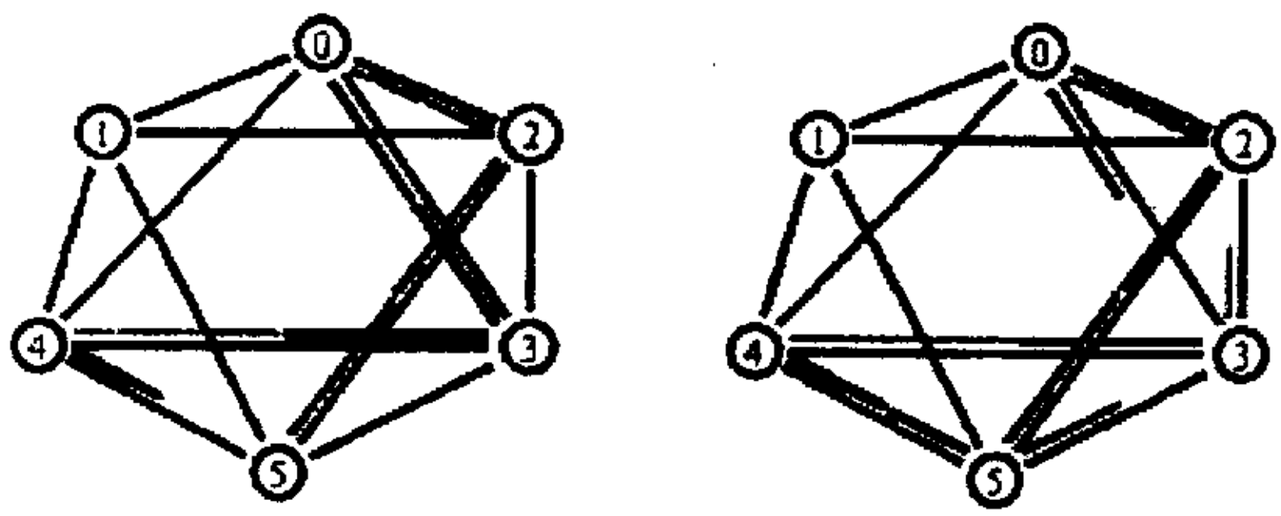


图 3.15 中间一次发生偏射的效果图

4、统计结果

仿真得到的统计结果如下：

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Total pkts  =2946
dropn       =0
Drop rate.  = 0.00000000
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Total hops with deflection:                 deflh=6890
Total hops with store-and-forward:          undeflh=5920
Average hops with deflection:               hops = 2.33876443
Average hops with store-and-forward:        hops = 2.00950441
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
delay=13.732000
Average pkt delay = 0.00466124
        total pkts = 2946,  total delay = 13.73200000
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Link utilization = 0.21453539
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

由结果可以看出在偏射情况下数据包的总跳数和平均跳数都将增加，在该业务模型下，平均每个包比存储转发模式增加了 0.329 跳。

第三节 本章小结

当两个数据包竞争同一个输出链路时，不同于存储转发模式，偏射路由算法将其中一个包偏射到另一输出链路，由于该链路可能不是最佳路由，就可能造成数据包的跳转次数多于存储转发算法的情况。这缓解了目前光缓存技术不成熟的困难，同时使得网络的链路利用率得到提高，均衡了网络中各部分链路的负载。

本章先对 8 个节点的 ShuffleNet 模型进行分析，对偏射和存储转发两种算法下包的跳数、链路利用率进行比较得到：偏射时包的跳数（平均跳数）有所增加，链路利用率也得到了提高；随着网络负载的增加，虽然包的总跳数增加了，但包的平均跳数却减少了。其中对原来的偏射路由算法进行了一些优化。然后考虑到 SN8 的路由循环问题，提出了新的拓扑，并对偏射算法应用于此拓扑时的现象作了简单的分析。

参考文献

[1] F. Forghieri, A. Bononi, P. R. Prucnal, "Analysis and comparison of hot-potato and single-buffer deflection routing in very high bit rate optical mesh networks", IEEE trans. Commun. vol.43, no.1, pp.88-98, Jan. 1995

[2] A. Bononi, F. Forghieri, P. R. Prucnal, "Analysis of One-Buffer Deflection Routing in Ultra-Fast Optical Mesh Networks", IEEE trans. Commun.

[3] Zheng Chen, Toby Berger, "Continuous time performance analysis for a hierarchical routing algorithm in Manhattan street networks", IEEE TENCON'93/Beijing

[4] Ben Y. Yu, Ivan Glesk, Paul R. Prucnal, "Analysis of a Dual-Receiver Node with High Fault Tolerance for Ultrafast OTDM Packet-Switched Shuffle Networks", Journal of lightwave technology, vol. 16, No. 5, May 1998

[5] S. H. Gary Chan, Hisashi Kobayashi, "Packet Scheduling Algorithms and Performance of Buffered Shufflenet with Deflection Routing", Journal of lightwave technology, vol. 18, No.4, April 2000

[6] Ching-Fang Hsu, Te-Lung Liu, Nen-Fu Huang, "Performance Analysis of Deflection Routing in Optical Burst-Switched Networks", IEEE 2002

[7] Alberto Bononi, Paul R. Prucnal, "Analytical Evaluation of Improved Access Techniques in Defection Routing Networks", IEEE/ACM transactions on networking, vol. 4, No. 5, October 1996

[8] Ender Ayanoglu, "Signal flow graphs for path enumeration and deflection routing analysis in multihop networks", 1989, IEEE

[9] C. Y. Li, P. K. A. Wai, X. C. Yuan, Victor O. K. Li, "Multicasting in deflection-routed all-optical packet-switched networks"

[10] Michele Baresi, Stefano Bregni, Achille Pattavina, Gianluca, "Deflection Routing Effectiveness in Full-Optical IP Packet Switching Networks",2003 IEEE

[11] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, Christophe Diot, "An Approach to alleviate link overload as observed on an IP backbone", infocom2003

[12] Xi Wang, Hiroyuki Morikawa, Tomonori Aoyama, "A Deflection Routing Protocol for

Optical Bursts in WDM Networks”,

[13] 戴翌清等，“折射路由法在光分组交换网络中的运用”，光缆与电缆及其应用技术，2003 年第 3 期

[14] 温锋，“光层组网的几个受限问题的研究”，北京邮电大学博士研究生学位论文，2003 年 7 月

第四章 S-CRMA 协议的仿真与分析

随着光纤通信技术的发展和用户对高速应用的不断需求，目前的超高速局域网和城域网要求：1、网络具有非常高的接入速率，通常达到几个 Gb/s；2、网络范围超过 100km。目前，FDDI 令牌环技术是主流的解决方案，但只能满足百兆速率和数十千米范围的要求，并且当网络规模扩大时，令牌传输时间和节点保持令牌的时间将使网络的吞吐量和接入时延严重恶化。为此，人们提出了分布队列双总线协议（Distributed Queue Dual Bus Protocol, DQDB, IEEE 802.6），它在具有两条单向总线的传输系统中采用了资源预留机制，确保了网络吞吐量和接入时延性能的要求，但又有如下不足：1、重负载下各节点接入的不公平性，节点的性能跟位置相关并相互影响；2、一个周期只能预约一个时隙，将数据包的各个片断置于不连续的时隙中，造成目的节点重新组装的困难。为克服上述问题，M. Medhi Nassehi 提出了 CRMA 协议。

周期预留多址接入（Cyclic Reservation Multiple Access，CRMA）协议是高速（几个 Gb/s）光局域/城域网中，基于时隙的折叠总线和双总线结构，具有高性能的网络协议。它引入新的预约机制，保证了各节点吞吐量的公平性（不保证时延的公平性），并且一个周期可以预约一个完整的包所需的时隙，但这增加了平均时延；因此同时引入背压机制，保证最坏情况下的时延在一定范围内。本章先对 CRMA 协议对进概略介绍，然后为适应光子时隙路由的要求（Photonic Slot Routing, PSR）,在此基础上进行改进，提出了简化的 CRMA 协议（Simplified CRMA Protocol，S-CRMA），并利用 NS2 平台对该协议在折叠总线型光网络上的运行进行仿真与分析。

第一节 CRMA 协议介绍

CRMA 是基于分时（slotted）单向总线（折叠总线或者双总线）系统的网络接入协议，它通过周期预留接入（cyclic reservation access）和预留取消背压（reservation-cancellation backpressure）两项新机制，保证网络在高速和长距离时高性能；为方便包的重组，它提供了为一个包预约连续时隙的功能。周期预留接入机制能提供高效的吞吐、灵活的容量配置，同时满足公平性要求；背压机制缓解最坏情况下的接入时延。下面分别就折叠总线对这两项机制进行说明。

4.1.1 基本接入机制

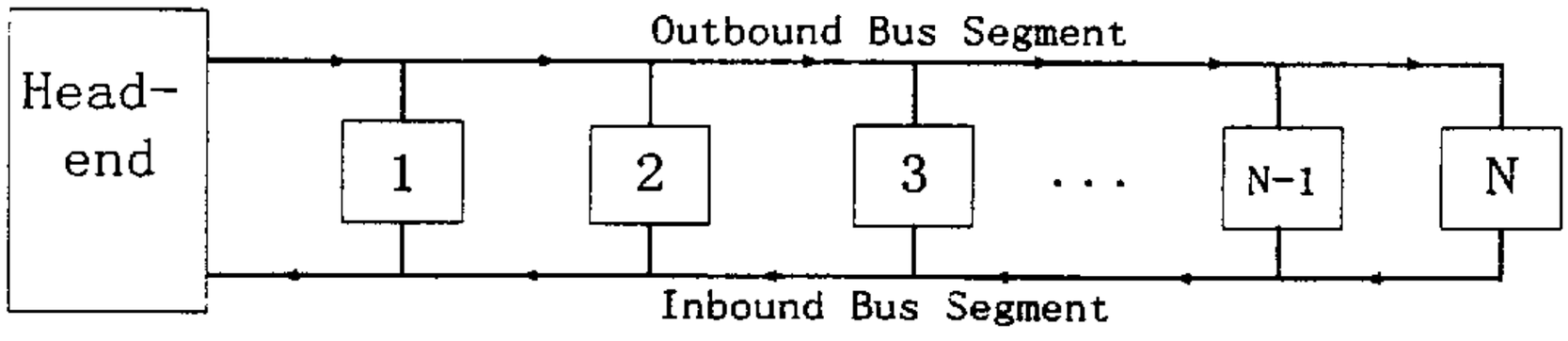


图 4.1 折叠总线结构

CRMA 折叠总线结构如图 4.1 所示，具有一个头节点（Headend）和 N 个本地节点（Local node）。头节点控制着整个总线的接入功能，从物理结构上讲，它是节点 1 的一部分，并且如果要求整个网络具有很好的可重构性（如增减节点等），它也应是其他节点的物理部分。总线由头节点开始，在写总线（Outbound bus segment）上依次经过节点 1 到节点 N，在节点 N 处折回到读总线（Inbound bus segment）上，再由节点 N 逐节点经过节点 1 最后回到头节点。数据包在写总线上发送并在读总线上接收，因此，在写总线上，节点应具有读取和修改数据位的功能，而在读总线上，节点只需读取和吸收数据位。

总线上数据的传送是分时隙的，时隙结构如图 4.2 所示。时隙包括接入控制域（access control field, ACF）和数据域（segment field, SF）。SF 可能为空或者包含某数据包的一部分。ACF 又分为两个子域：忙/闲标识位（busy/free bit, B/F）,指示 SF 是否为空；接入指令，节点根据该指令进行相应动作。一般 SF 都短于数据包的平均长度；如果数据包不长于 SF，则该包可装载在一个时隙中，否则该数据包将会被分割为一系列连续的小的片断（除最后一个片断可能小于 SF 外，其它的大小都等于 SF），在目的节点，再从总线上拷贝出所有的片断进行重组。

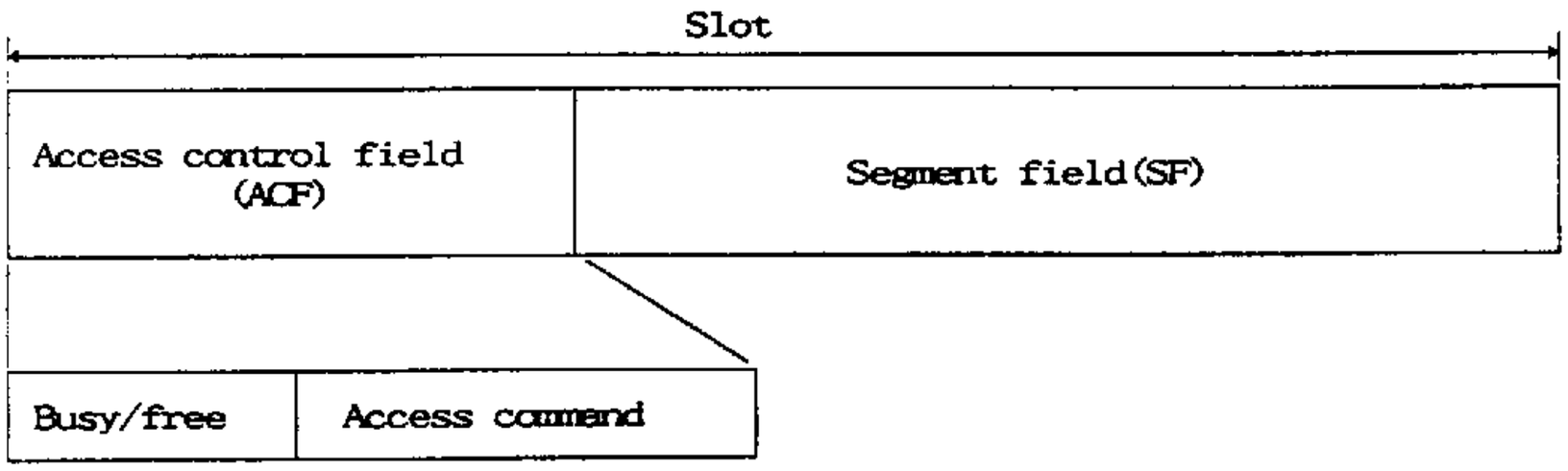
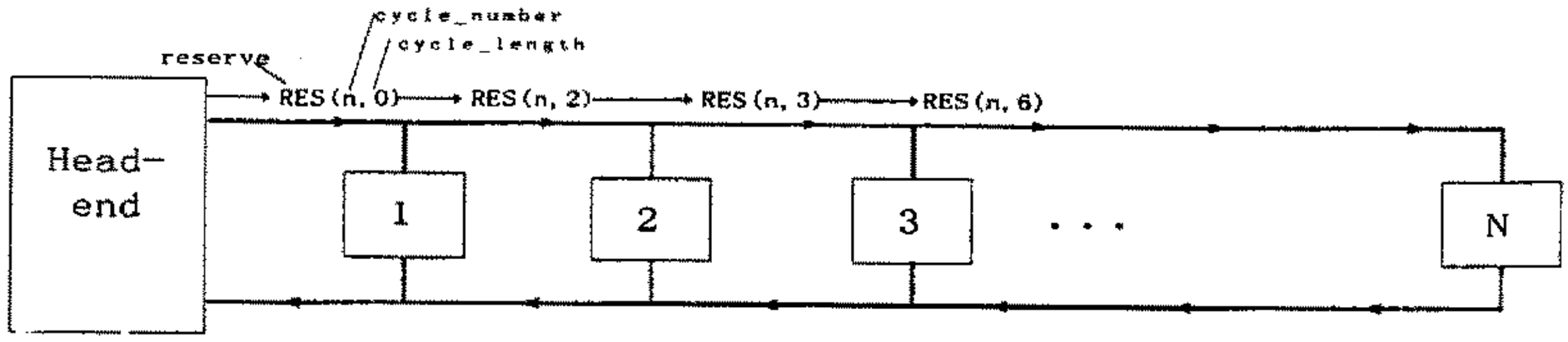


图 4.2 时隙结构图

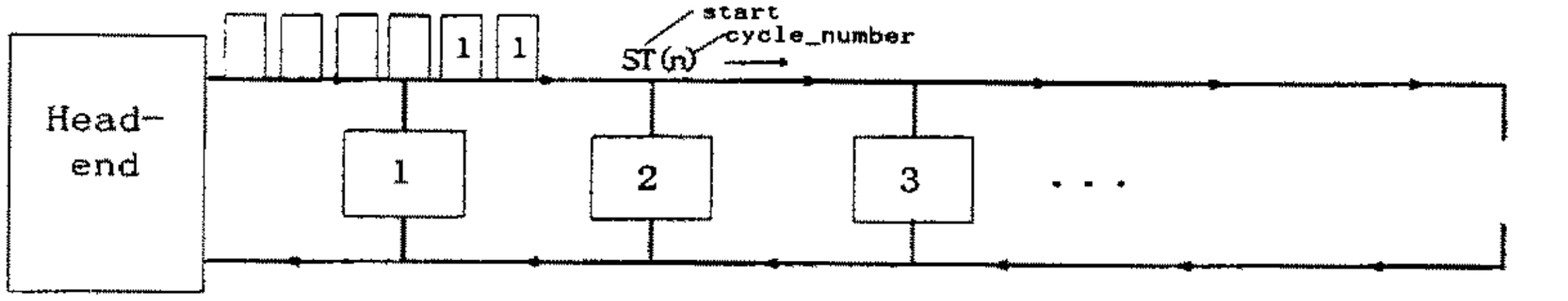
节点根据时隙的周期序号（cycle number）读取总线。周期号从零开始递增至某一最大值（比如 49）后复零，如此循环。在每个周期，本地节点预约一定数量的时隙，头节点据此产生足够长的周期满足此预约请求，因此每个周期的长度不是固定的，而是由本地节点的需求确定的；如果在本地节点没有预约（无数据包），头节点就不

产生相应的周期了。预约和数据周期(data cycle)的产生的命令分别是 reserve 和 start，它们由头节点发出，属于同一个周期，参数为周期序号（cycle_number）,reserve 命令还包括周期长度（cycle_length）参数。

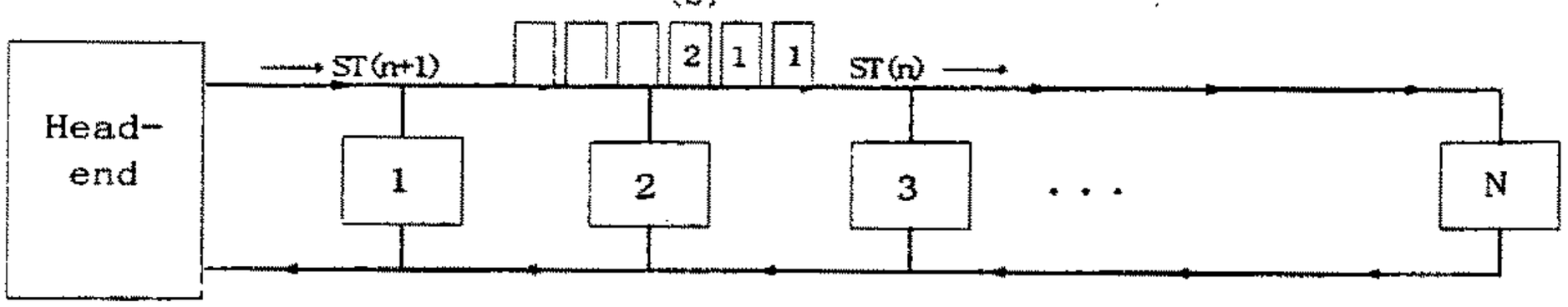
图 4.3 描述了基本的周期预留接入机制。头节点周期性发出 reserve 命令，初始化 cycle_length 参数为 0。在写总线上，当 reserve 命令经过节点时，节点预约时隙：将自己所需时隙数累加到 cycle_length 参数上，因此，该周期中预约的时隙数即为该参数值；并且，节点将此 cycle_length 连同 cycle_number 一起存储到本地预约队列中。经过写总线上最后一个节点时，cycle_length 即为该周期节点预约的总时隙数。在读总线上，节点不对 reserve 命令作任何修改，直接返回到头节点；头节点将该 reserve 命令及参数存储到全局预约队列中，然后按 FIFO 原则的服务。



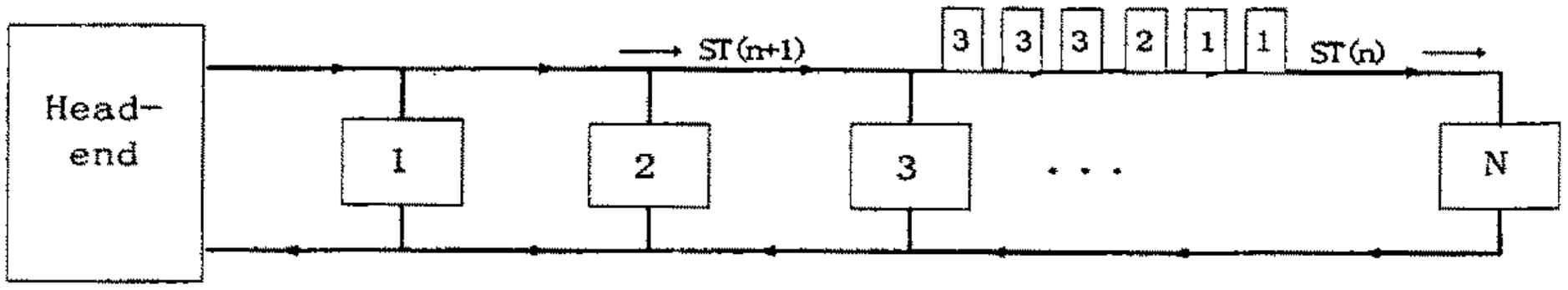
(a) 在周期n中，reserve命令在写总线上经过节点1预约2个时隙，节点2预约1个时隙，结点3预约3个时隙，cycle _length逐步累加



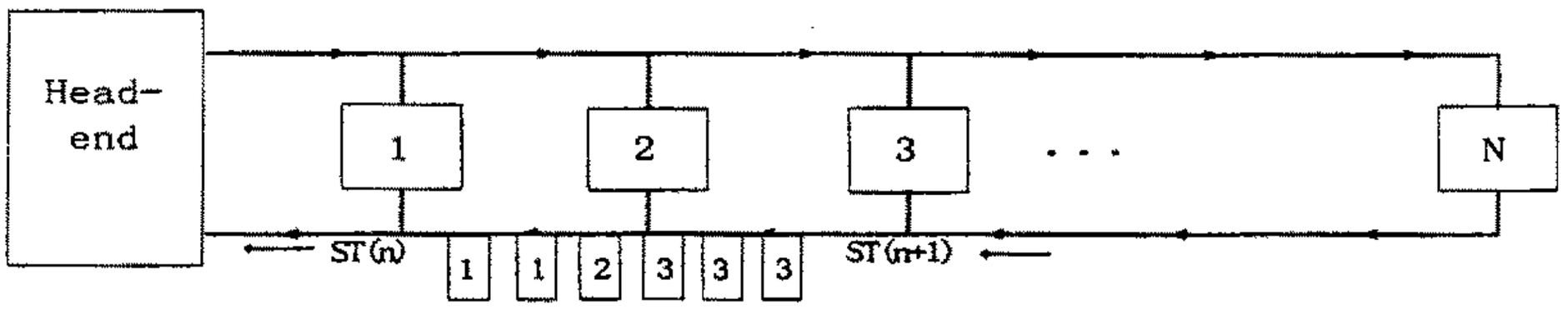
(b)



(c)



(d) 头节点收到返回的reserve命令后，发出相应的start命令，其后跟随着与cycle_length数一致的空时隙，每个节点将本地预约队列中相应数据包写到这些空时隙中



(e) 在写总线上，目的节点复制数据

图 4.3 周期预留接入机制

头节点通过产生 start 命令来服务每个预约，该命令包括一个 cycle_number 参数，后面则跟着 cycle_length 个空时隙以用来写入数据。写总线上，本地节点收到 start 命令后，检查本地预约队列，如果发现其 cycyle_number 与 start 命令的该参数一致，就等待紧跟的空时隙，然后将数据包写入数据周期。注意每个节点依序使用连续的时隙，

并且一个包的所有片断在同一个周期内传输，这样简化了接收时的重组。读总线上，包的目的节点复制数据后传向下一个节点，其它节点则直接转发，最后数据环回到头节点后被吸收。

基本接入的状态一时间动态效果如图 4.4 所示。

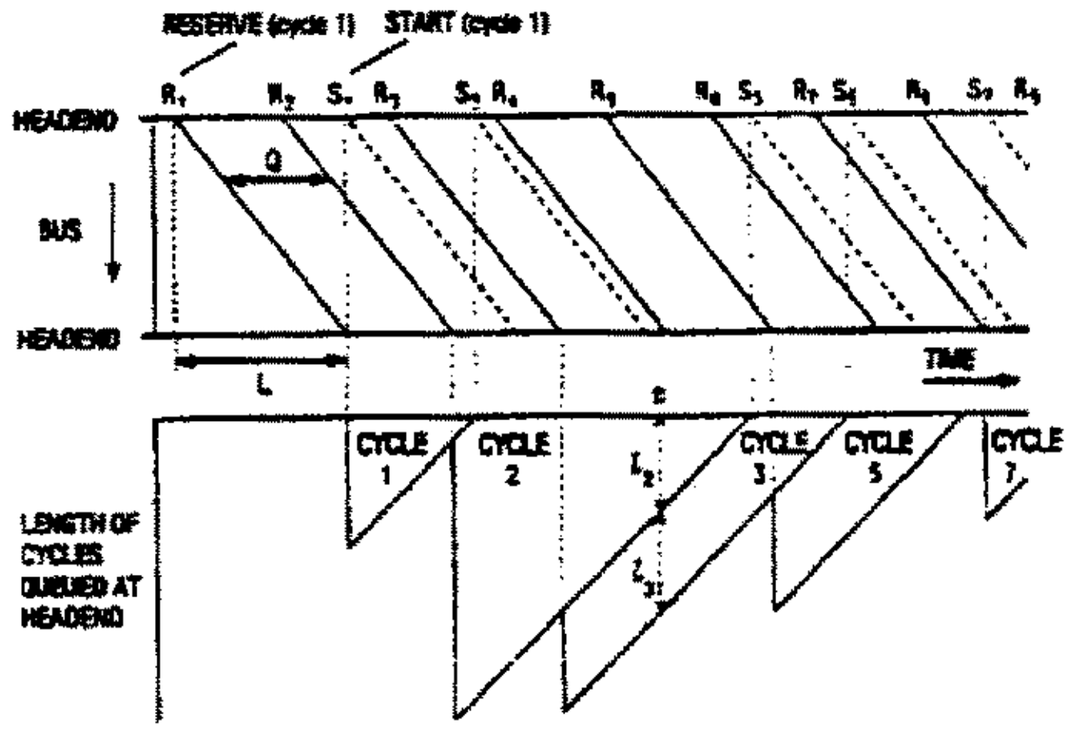


图 4.4 基本周期预留接入的时间一状态图

图中 R_i、S_i 分别表示 reserve、start 命令，L 表示命令在总线上的传输时延，Q 表示 reserve 命令的周期。上图是总线上命令传输的距离一时间关系，下图表示头节点中 cycle 队列的变化。t 时刻，cycle2 在队列的顶端，其剩余长度为 l_2，cycle3 排在 cycle2 之后，其长度为 l_3。假设头节点发 R_1 时，队列中预约为空，当收到 R_1 后，头节点发出 S_1 开始 cycle1；R_2 返回时，cycle1 还未完成，因此 cycle2 插入到全局预约队列中 cycle1 之后；当 cycle1 完成之后，开始 S_2；接着 R_3 返回，cycle3 入队；R_4 预约长度为零，不增加队列长度；cycle3 结束之后，cycle5 即开始...

这种管式的周期预留处理方法为网络提供了高的吞吐效率，并且跟网络的速度和规模无关。适当选择 reserve 命令的发送频率，可以提高网络利用率。如果该频率是 data cycle 频率的足够大倍数，则可以保证头节点中预约队列周期的连续性（即 cycle1、cycle2、cycle3 等依次入队，中间无空周期），因此选择较高的发送频率，可以获得较高的利用率。通常，reserve 发送周期设置为系统最大数据包的传输时间。

周期特性还为节点间的容量分配提供了很好的手段，这可以通过为每个节点设置在每个 cycle 中预约和接入的时隙数限制来实现，而这种限制既可以是节点类型也可以是周期序号。并且由于周期性的接入，每个节点的吞吐量是与在总线上的位置无关的，但接入延迟是与位置相关的。

4.1.2 背压机制

虽然选择足够高的 reserve 命令产生频率提高了吞吐效率，但同时却又恶化了接入时延。为减小在最差情况下的接入时延，发展了预留取消背压机制。这种机制时刻监视头节点中全局预约队列的总时隙数（包括那些即将发出的 cycle 的时隙数），如果大于某一阈值，则暂停发新的 reserve 指令，并且已经发出但还没返回的 reserve 也被取消，知道总时隙数降到低于阈值为止。最优化的阈值选择是总线来回一周的延迟除以时隙传输时间，即总线上传播的时隙总数。

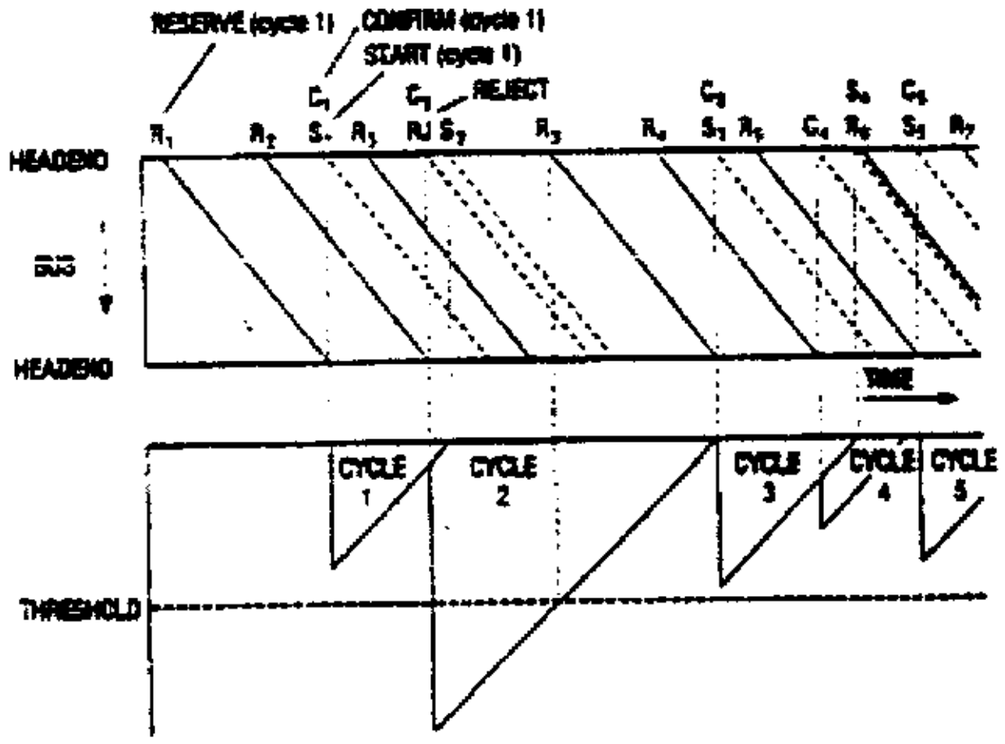


图 4.5 使用背压机制时的状态一时间图

背压机制需要两个指令：confirm 和 reject，它们的作用机制如下：当队列小于阈值时，头节点周期性的发出 reserve，在 reserve 返回后，头节点发出 confirm 确认相应的 cycle 并将预约加入全局预约队列；如果大于了阈值，就发出 reject，取消所有的已发出但未返回的 reserve，直到队列恢复到阈值以下才重新发 reserve。使用背压机制时的状态一时间动态效果如图 4.5 所示。

第二节 S-CRMA 协议对 CRMA 协议的改进

已有的 CRMA 协议只适用于传统网络，在未来的 10G 以上超高速网络和现有光纤通信技术条件下，其通信规程尤其是背压机制还是显得过于复杂：（1）CRMA 协议包括四种指令，reserve、start、confirm、reject，它们的参数不完全一样，时隙结构不一致，造成处理复杂、速度降低；（2）通过预留取消背压机制通知本地节点是否预约成功和根据业务状况调节预约周期的长度，客观上增加本地数据的接入时延，包括 reserve 指令网络环回时间、confirm 等待和处理时间以及数据传输时间；（3）reserve 指令的发送周期固定，无法根据网络状况进行自主的动态调节，造成业务量较大时接

入时延的增加。因此需要为适应超高速光分组交换网络作进一步简化和改进，即简化的 CRMA 协议（Simplified CRMA Protocol，S-CRMA 协议）。

S-CRMA 协议取消了背压机制及 confirm 和 reject 指令，而是采用由主节点根据全局预约队列的长度来确定 reserve 指令的发送间隔的方式来优化资源的预约，即当队列长度超过阈值时，增大 reserve 指令发送间隔，小于阈值时，恢复正常的发送间隔。

S-CRMA 协议同样采用统一的固定长度、固定间隔的时隙作为承载数据分组的基本单位，但只有 reserve 指令时隙和数据时隙两种，时隙结构如图 4.6 所示。

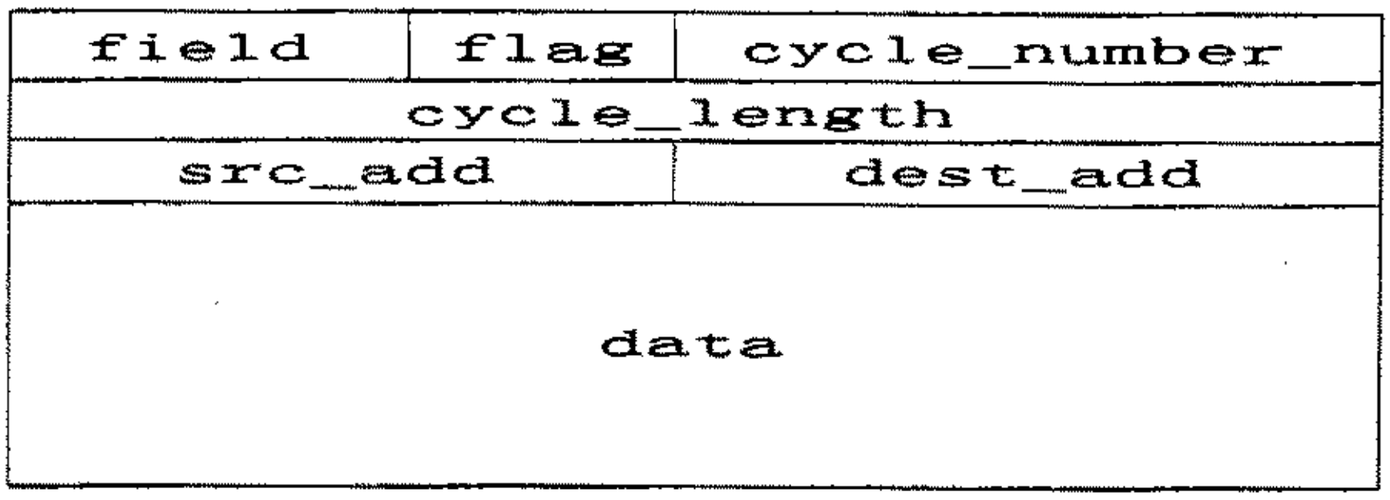


图 4.6 S-CRMA 时隙结构

各部分的意义如下：

field：初始值为 0，写总线上每经过一个节点值加 1，读总线上不变。field 值用来标识分组是在读总线上还是在写总线上：如果 field<N，包位于写总线上，节点可以对数据进行读写操作；如果 field>=N，则包位于读总线上，节点只能对数据包进行读操作。

flag：用于标识时隙类型，长度为 2 位。Flag 为 0 代表 reserve 指令时隙，cycle_length 初始化为 0，data 域为空；值为 1 和 2 代表数据时隙，1 表示空时隙，等待节点写入数据片断，然后值改写为 2，表示已写入数据；值为 3 则在 data 域填入无效数据，用于网同步。

src_add 和 dest_add：分别表示源节点和目的节点的 MAC 地址。

其它域的定义与 CRMA 中一致。

基本的周期预留接入机制也与 CRMA 中相似，只是多了对 field 和 flag 的判断和操作。另外，为了降低最坏情况下的接入时延，头节点在发送预留时隙之前查看全局预留队列的长度，当其超过阈值时，增大 reserve 指令时隙的间隔，直到小于阈值。本章的后续部分将基于折叠单向总线对 S-CRMA 协议进行仿真与分析，主要从节点间接入公平性、接入时延以及网络吞吐量等方面考查该协议的性能。

第三节 仿真设计

仿真使用的物理拓扑和逻辑拓扑如图 4.7 所示，包括头节点和 4 个本地节点，总线的速率是 40G，本地节点的接入速率可变；本地节点间的时延是 0.1us；包大小为 125 字节。

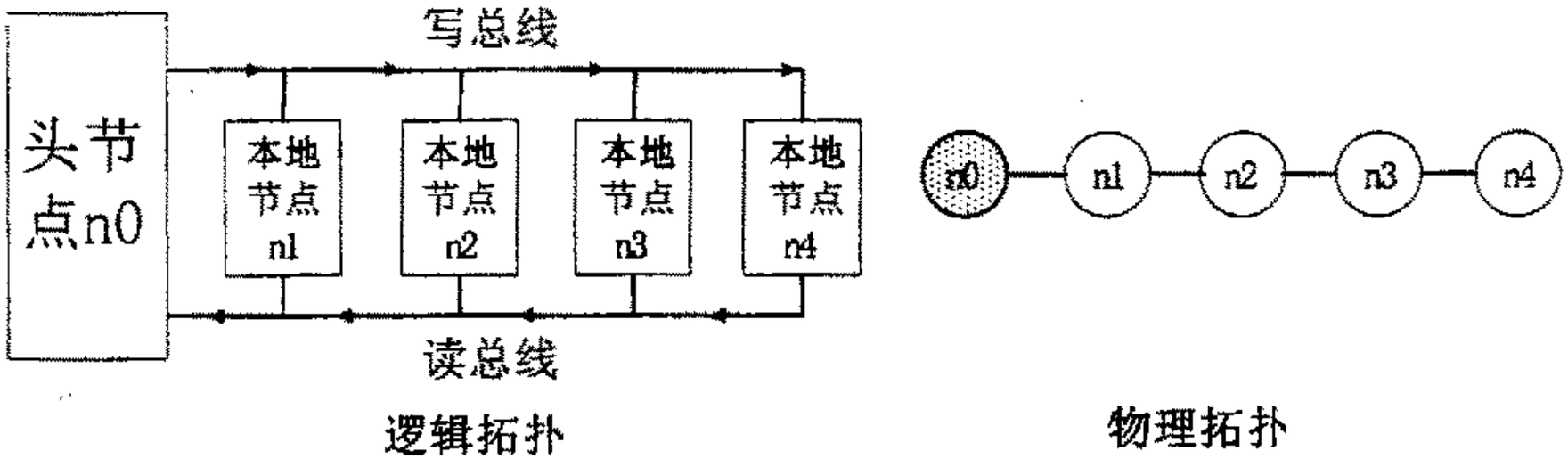


图 4.7 仿真 S-CRMA 协议的拓扑

无论是头节点的设计还是本地节点的设计，相对于偏射路由算法而言，都不需要路由模块，因为分组会自动环回于总线上，节点不需要为分组决定下一跳地址，而是只有唯一的下一跳。这样，利用 NS2 仿真时，只需要考虑节点的控制机制即可。在实际的网络中，头节点的功能是每个本地节点功能的一部分，以使网络具有很好的可升级性和重构性，但在仿真中为了简化问题，对头节点和本地节点分别设计，并且这样不会影响仿真结果。仿真中，针对头节点和本地节点在折叠总线型光网络中的不同作用，分别设计各自的 classifier 模块，下面对二者的工作流程作详细说明。

4.3.1 头节点

头节点的处理流程如图 4.8 所示。头节点周期性的产生 reserve 指令时隙；头节点内部包含一个全局预留队列，并设有阈值。头节点的 classifier 的工作流程是：头节点每产生一个新时隙之前，首先察看 reserve 指令时隙的周期是否到来，如果到来且全局预留时隙总数小于阈值，则发送 flag 值为 0 的 reserve 分组；如果 reserve 指令时隙发送周期尚未到来，则判断由上一个 reserve 指令时隙确定的数据 cycle 是否已经结束，如果没有，则继续发送 flag 值为 1 空闲数据分组以满足预留需求，如果已经发完，则看全局预留队列是否为空值，如果是空值，则表示当前时隙既不用发送 reserve 指令时隙也不用发数据时隙，则把 flag 置为 3，开始发送同步时隙，以保持全网同步；如果预留队列不是空值，则根据下一个预留周期的请求开始一个新的数据 cycle，进行空闲数据 cycle 的发送。当头节点收到环回的分组时，如果 flag＝0，则在全局预留队列中增加 cycle_length 个时隙，如果 flag 为其它值，则直接释放掉。仿真中设置全

局预约队列的大小为 400 个包的时隙数。

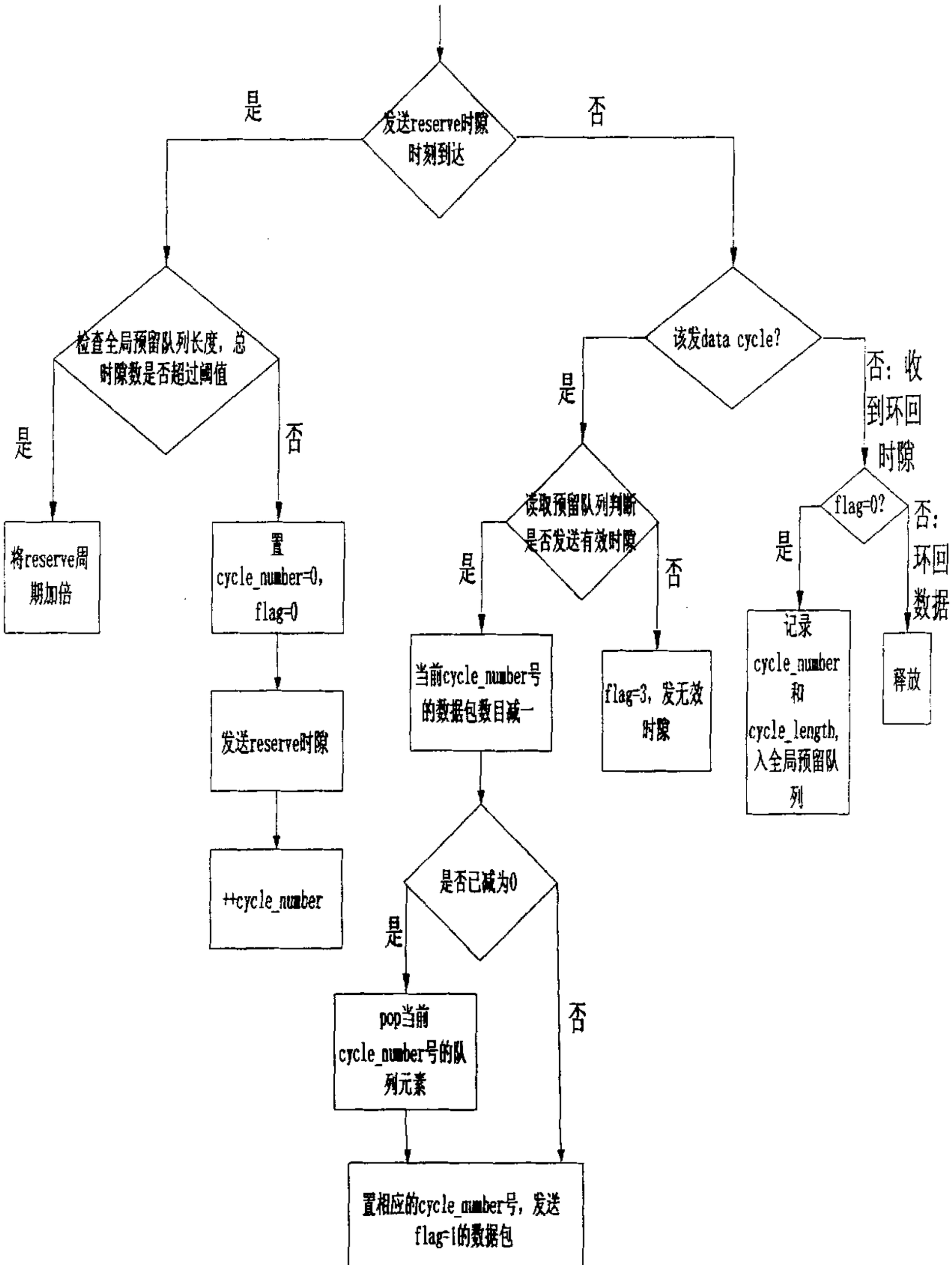


图 4.8 头节点处理流程

4.3.2 本地节点

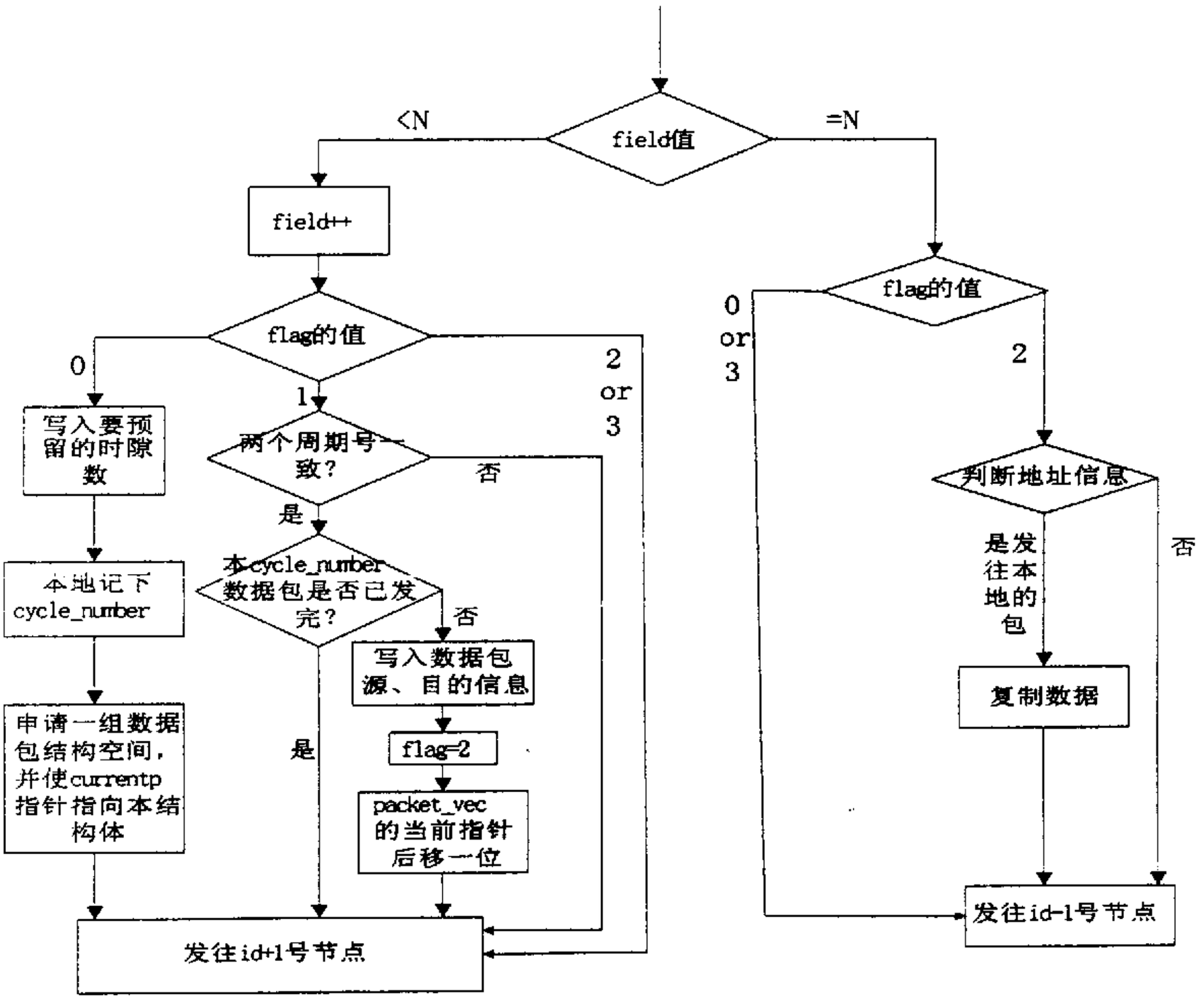


图 4.9 本地节点处理流程

本地节点处理流程如图 4.9 所示。本地节点根据 field 域的值判断分组是在写总线上还是在读总线上。在写总线上，本地 classifier 分别根据分组 flag 的不同值进行操作，flag 为 0，说明是 reserve 指令时隙，则统计本地存储队列中上一个 reserve 指令时隙之后新产生的本地数据包的个数，记录本周期 reserve 指令时隙的 cycle 值，并与本次预留的本地数据包进行关联；flag 为 1，说明是空闲数据分组，根据分组的 cycle 值将本地队列与这一 cycle 值相关联的数据包写入空闲分组，包括 src address 域，dest address 域，data 域，并修改 flag 值为 2；如果写总线上分组的 flag 是其他值，说明是占用数据分组或同步分组，不对此分组进行任何修改操作，直接发送到下游节点。在读总线上，如果分组的 flag 值是 2，则根据分组的目的地址 dest address 域判断本节点是否是目的节点，从而决定对数据分组进行复制或者直接发送到上游节点；如果分组的 flag 值是其他值则简单地将该分组递交到上游节点。仿真中设置本地预约队列（buffer）的大小为 100 个包。

4.3.3 业务模型说明

仿真中各个本地节点加载的是开关业务（ON_OFF 业务，为 NS2 内置的一种业务模型），ON 状态和 OFF 状态的保持时间，均符合 Pareto 分布，图 4.10 中的 T_{ON} 和 T_{OFF} 分别为二者的保持时间。Pareto 分布是描述自相似性常用的分布函数，其概率密度及参数如下：

$$f(x)=\frac{cT^c}{x^{c+1}},\quad E(x)=\frac{cT}{c-1},\quad \delta_x^2=\frac{cT^2}{\left[(c-1)^2(c-2)\right]},$$

式中 T 确定开关时间，c 确定开关时间的自相似程度。

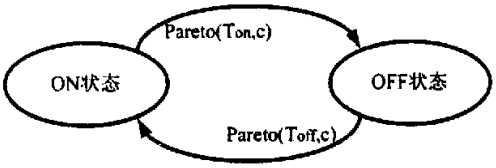


图 4.10 开关业务状态转移图

第四节 结果及分析

这部分是利用上述模型在 NS2 平台上仿真的结果及分析，图例中 Tr0.9A15G 表示 Traffic=0.9，节点接入速率 A＝15G，其中 Traffic 的计算式为：$Tr=\frac{T_{ON}}{T_{ON}+T_{OFF}}\times\frac{A}{10G}$

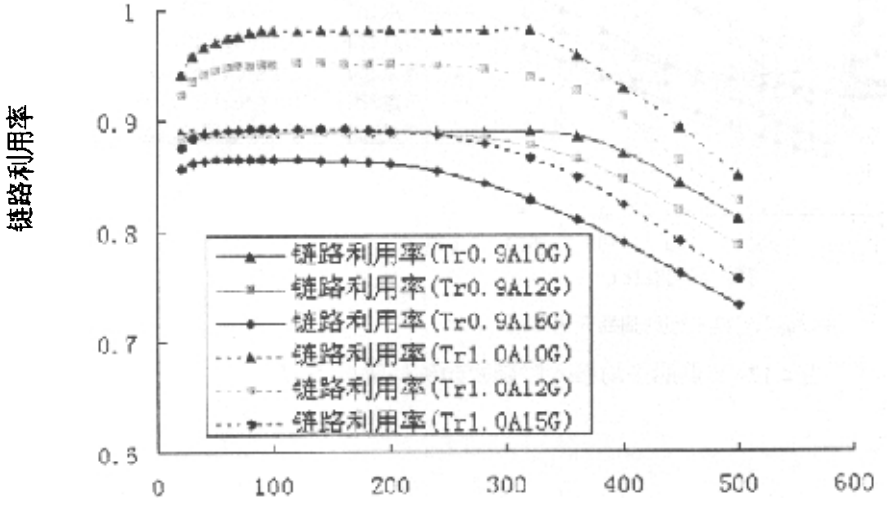
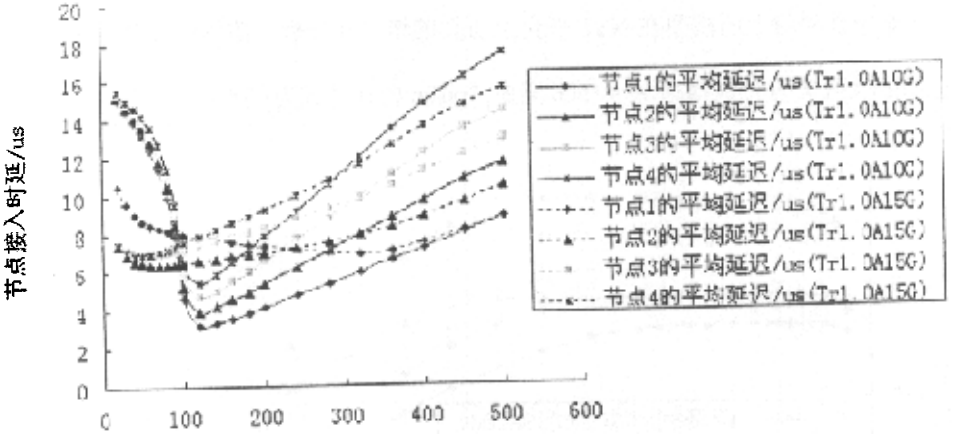


图 4.11 链路利用率随预约间隔的变化

图 4.11 描述了链路利用率与预约指令时隙间隔的关系。可以看到，当 reserve 指令时隙间隔很小（小于 50slots）时，链路利用率有急剧恶化的趋势，而在 50－350slots 之间，基本保持稳定，再增加又开始降低，并且负载越大降低越快。这说明，reserve 指令的频率必须足够高，以获得较高的链路利用率，但必须与数据包的传输时间可比拟。同时可以看出，链路利用率随节点接入速率的增加而减小，随流量（traffic）的增大而增加；后者容易理解，前者主要是因为在负载由中度变为重度的时候，丢包率增加的更快，参看图 4.14。

图 4.12 描述了中度负载和重负载情况下 4 个节点的平均接入时延随预约间隔的变化情况，两种情况下，都存在一个最优的 reserve 间隔使时延最小。在中度负载情况下，这个最优值为 100－140slots，这刚好与理论值一致：仿真中一个包的传输时间为 2.8us（包括接入延迟和链路延迟），恰好是一个 slot 的时间（为 0.025us，即总线上一个包的处理时间）的 112 倍，即 reserve 间隔约等于一个包的传输时间时性能最佳。在重负载情况下，最佳 reserve 间隔有所前移，为 50－70，这说明重负载下 reserve 频率不宜太高。对于图 4.13 描述的最大接入时延与 reserve 间隔关系，同样有上述结论。



预约周期/slot

平均接入时延－预约周期变化关系

图 4.12 节点的平均接入时延随预约间隔的变化

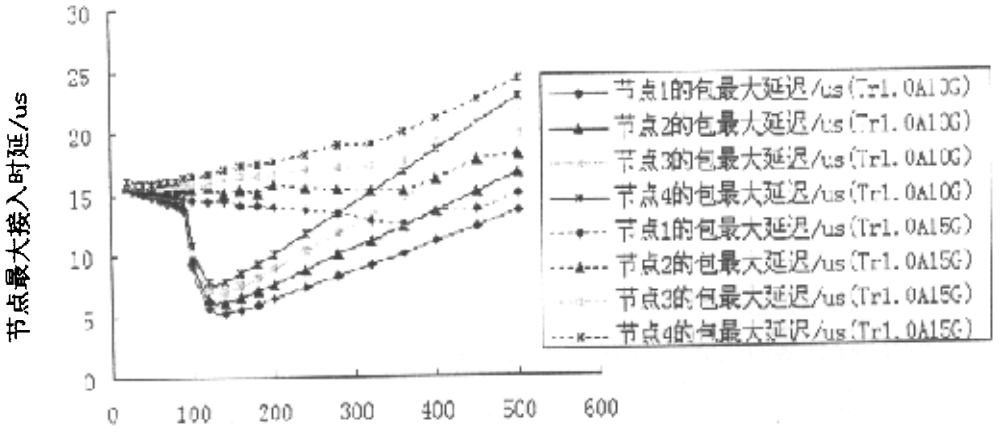


图 4.13 中度负载和重负载时节点的最大接入时延

从图 4.12 和图 4.13 还可以看出节点间接入时延的不公平性（无论平均时延还是最大时延）：本地节点越靠近头节点，接入时越具有优先性；但是，各节点的差别非常小（在最忧情况下不到 5%），这同时说明了改进后的算法对接入公平性的提高，后面将进一步看到这一点。还非常值得注意的是，平均时延跟最大时延差不多在同一量值，说明了系统的高度稳定性。

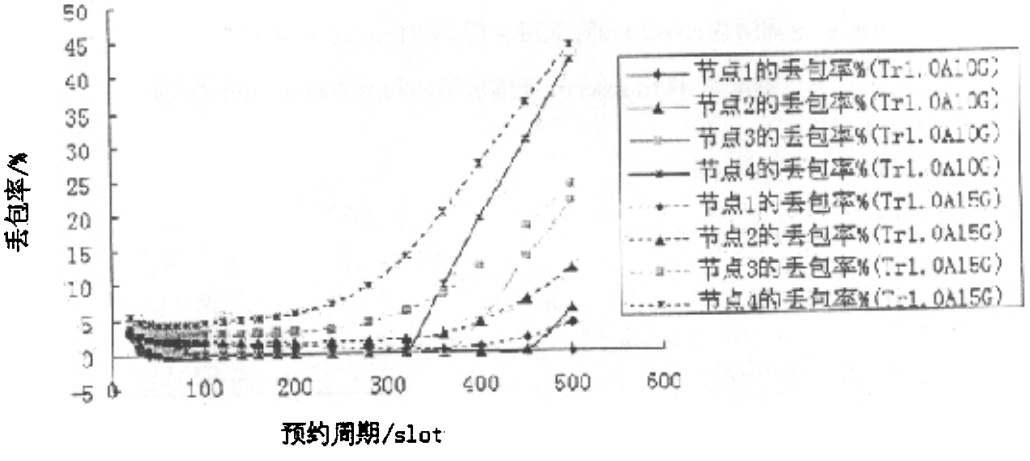
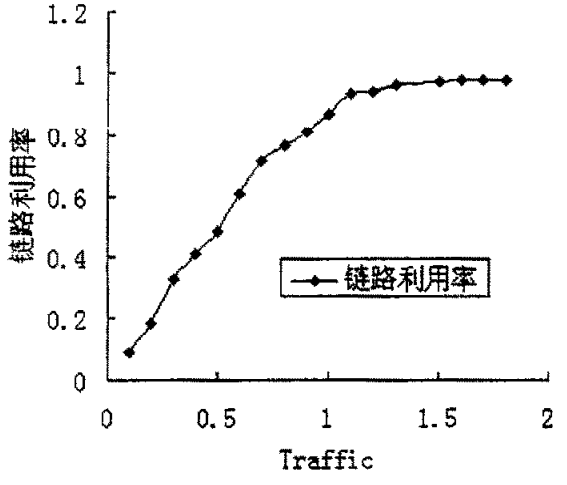


图 4.14 各节点丢包率的变化

不同负载下各节点的丢包率曲线（图 4.14）同样显示出 reserve 间隔存在最优值

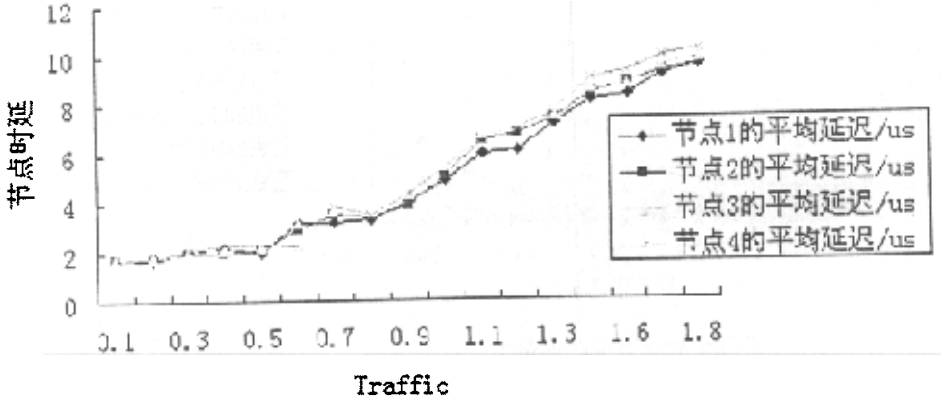
区间，在该区间各节点丢包率很小，并且各节点之间以及不同负载情况下的差别很小，显示了较好的公平性。

图 4.15 和图 4.16 是优化 reserve 间隔后的链路利用率和各节点接入时延与 Traffic 的变化关系。可以看出，最初链路利用率随着 Traffic 的增加线性增加，但当 Traffic 增大到 1.2－1.3 时，链路利用率趋于 1，平稳下来。这要求在一定的网络延迟范围内，选择合适的 Traffic。各节点的延迟随 Traffic 增大逐步增加，但是，节点之间的差别非常小，这说明在取消了原来 CRMA 的背压机制而采用根据全局预约队列大小自适应改变 reserve 间隔的算法，能很好的提高网络中节点的接入公平性。



reserve周期为100slots时链路利用率随Traffic的变化

图 4.15 优化 reserve 间隔后链路利用率随 Traffic 的变化



优化预约周期后时延与Traffic关系

图 4.16 优化 reserve 间隔后各节点时延随 Traffic 的变化

第五节 本章小结

针对光分组交换局域/城域网络，本章在提出应用折叠总线的网络模型的前提条件下，首先对 CRMA 协议进行了介绍，包括基本的周期性时隙预约机制和为控制最坏情况下接入时延的背压机制，然后对其进行了简化和改进，提出了简化的 CRMA（S-CRMA）协议。它对时隙结构重新进行了定义，取消了原有协议的 REJECT、CONFIRM 等命令，通过自适应地改变预留时隙的间隔，实现了网络在接入时延和接入速率方面的公平性。

在突发业务的条件下，对提出的 S-CRMA 协议进行了研究和仿真。通过仿真数据和曲线，说明了基于 S-CRMA 协议的网络在能很好的保证了各个本地节点的公平接入，实现接入时延和网络资源利用率之间的均衡，即实现了在满足接入时延的条件下，尽可能高的网络资源利用率。

参考文献

[1] Hamdi, M., "CRMA: a high-performance MAC protocol for fiber-optic LANs/MANs", IEEE Communications Magazine, Mar 1997

[2] J.O. Limb, "A Simple Multiple Access Protocol for Metropolitan Area Networks", Proc. ACM'90 Sigcomm conf., September 1990

[3] Watson, G.C.; Tohme, S.; "S++-a new MAC protocol for Gb/s local area networks", Selected Areas in Communications, IEEE Journal on , Volume: 11 Issue: 4 , May 1993 Page(s): 531 –539

[4] Finn, S.G; "HLAN - An Architecture for Optical Multi-Access Networks", LEOS 1995, Page(s): 45 -46

[5] Marsan, M.A.; Casetti, C.; Grasso, S.M.; Neri, F.; "Slot reuse in MAC protocols for MAN's", Selected Areas in Communications, IEEE Journal on , Volume: 11 Issue: 8 , Oct 1993 Page(s): 1290 -1301

[6] Muller, H.R.; Nassehi, M.M.; Wong, J.W.; Zurfluh, E.; Bux, W.; Zafiropulo, P.; "DQMA and CRMA: new access schemes for Gbit/s LANs and MANs", INFOCOM '90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. 'The Multiple Facets of Integration'. Proceedings., IEEE , 3-7 Jun 1990 , Page(s): 185 -191 vol.1

[7] Nassehi, M.M.; "CRMA: an access scheme for high-speed LANs and MANs", SUPERCOMM/ICC '90, Apr 1990, vol.4, Page(s): 1697 -1702

[8] Sharon, O.; Segall, A.; "Schemes for slot reuse in CRMA", Networking, IEEE/ACM

Transactions on , Volume: 2 Issue: 3 , Jun 1994 Page(s): 269 -278

[9] Marsan, M.A.; Neri, F.; Perino, M.; Taricco, G.; "Simulation and Analysis of the Crma Protocol", Advanced Communications and Applications for High Speed Networks, 1992. Proceedings., International Workshop on , 16-19 Mar 1992 Page(s): 261-272

[10] Sharon, O.; Segall, A.; "Schemes for slot reuse in CRMA II", Local and Metropolitan Area Networks, 1993., Proceedings of the 6th IEEE Workshop on , 14-16 Oct 1996 Page(s): 0_40 -0_40

[11] Ching-Chih Han; Chao-Ju Hou; Shin, K.G; "On slot allocation for time-constrained messages in dual-bus networks", Computers, IEEE Transactions on , Volume: 46 Issue: 7 , Jul 1997 Page(s): 756 –767

[12] Huang, E.Y.; "Analysis of cyclic reservation multiple access protocol", Local Computer Networks, 1994. Proceedings., 19th Conference on , 1994 Page(s): 102 –107

[13] 左鹏，"分组交换式光网络"，北京邮电大学博士研究生学位论文，2003 年 5 月

[14] Peng Zuo, Jian Wu, Jintong Lin, "Cyclic Reservation Based Photonic Slot Routing Architecture"

致谢

衷心感谢我的硕士研究生导师林金桐教授在科研和学习中对我的指导和关怀。林老师严谨求实的工作态度、公正民主的为人风范和高瞻远瞩的处事方式给我非常深刻的感触，必将影响我一生的学习、工作和生活。身为一校之长，林老师在承担繁重的行政事务的同时，不但自己的研究工作从未间断过，还时刻关心着实验室的建设、课题的进展以及研究生们的学习和生活；而且还坚持每学期都要开一到两门课程，这是非常难能可贵的。最让人感动的是在二零零三年非典期间，林老师带着学校的领导班子，坚持每天都和同学们一起在学生食堂吃饭、聊天，在那个特殊时期，这种举动为稳定大家的心态，为北京邮电大学胜利度过难关，起到了十分重要的作用。庆幸自己能遇到如此无私敬业的导师，其精神将永远激励、鞭策我人生的各个层面不断前进！

衷心感谢叶培大院士。叶先生渊博的学识、对科学孜孜以求的态度、超俗的大师风范和以九十高龄仍兢兢业业于讲坛的精神，给我留下难以磨灭的印象。

衷心感谢伍剑副教授。伍老师丰富的理论知识和扎实的科研功底，让我受益匪浅。正是伍老师的悉心指导，我才能顺利完成我的学业。

感谢左鹏博士、温锋博士、左超博士、马文华博士对我研究工作的真诚指导和帮助。

感谢高屹然同学，他十分活跃的思维往往能给我们的工作带来创新性的解决方案，再加上他那十分友好的为人方式，使得我们的合作总是无限愉快的。

感谢林毅恒同学，薛青同学、郭艳梅、李玉辉、张娇、王瑞麟等师弟师妹们，正是大家的共同努力，才使得我们项目组能够不断成长，最后取得好成果，同时愿这个组继续为实验室做出更大贡献；也感谢所有的实验室的兄弟姐妹们的热心帮助和支持。

感谢我敬爱的父母，是他们给予了我一切，我人生道路上的每一次进步和所有的荣耀，都与他们的理解、鼓励和支持分不开的。

感谢所有资助过我的亲朋好友及同学，是他们帮我度过了难关。

感谢所有关心我、鼓励我、支持我的良师益友！