申请上海交通大学工程硕士学位论文

基于共享内存的 AFDX 航空网络交换机设计

学校代码：　　10248

作者姓名：　　胡靖飞

学　　号：　　1100379037

第一导师：　　胡飞

第二导师：　　李健

学科专业：　　软件工程

答辩日期：　　2013 年 01 月 09 日

上海交通大学软件学院

2013 年 01 月

A Dissertation Submitted to Shanghai Jiao Tong University

for Master Degree of Engineering

SPACE PARTITIONED AFDX SWITCH DESIGN BASED ON SHARED MEMORY ARCHITECTURE

University Code：	10248
Author：	Jingfei Hu
Student ID:	1100379037
Mentor 1：	Fei Hu
Mentor 2:	Jian Li
Field：	Software Engineering
Date of Oral Defense：	2013-01-09

School of Software

Shanghai Jiao Tong University

Jan, 2013

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：胡靖飞

日期：2013 年 01 月 11 日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：胡王清飞　　指导教师签名：

日期：2013年01月11日　　日期：2013年01月11日

基于共享内存的 AFDX 航空网络交换机设计

摘 要

AFDX 已经成为新一代航空网络通信的标准，被应用在诸如 A380 等大型客机上。AFDX 网络相比上一代航空网络来说，有极其明显的优势，例如全双工、高带宽、低时延、低丢包率和低成本等。作为新型的航空数据总线，AFDX 为了满足航空数据传输的实时性和可靠性的需求，采用了传统以太网技术，并对传统以太网技术进行改造。在 AFDX 航空网络中，交换机相互连接组成的 AFDX 互连系统构成了 AFDX 网络数据通信的主体，交换机的性能对 AFDX 整体性能起着举足轻重的作用。在众多已有的交换机架构中，共享内存架构技术提出的时间非常早，发展至今已经非常成熟。然而共享内存架构本身固有的缺点，导致其可靠性差，不易扩展，究其原因主要是交换机所有输入和输出端口共享一块内存和一个调度器，从而导致严重的资源竞争、调度时延和抖动。

本文基于共享内存架构，对交换机所有输入和输出端口共享的内存采用空间分离技术，同时对调度器采用异步调度技术，针对 AFDX 网络提出一种新型架构的交换机——空分共享内存交换机（Space Partitioned Shared Memory Switch，简称 SPSMS）。

论文的主要工作和研究成果如下 ：

（1）研究 AFDX 协议的内容以及 AFDX 航空网络的组成、特性及其有别于其它网络的属性，包括航空电子子系统、AFDX 终端、AFDX 互连系统以及虚拟链路等。充分掌握了 AFDX 网络的静态特性，包括带宽静态性和路由静态性等，并意识到 AFDX 交换机在整个 AFDX 网络中起着不可或缺的作用。

（2）研究传统计算机网络中各种架构类型的交换机，并着重于共享内存架构，缘于其结构简单、硬件成本低和硬件利用率高的特点。总结出共享内存交换机的优点和缺点，并指出这些优缺点适合与不适合 AFDX 网络的原因。

（3）结合 AFDX 网络的静态性与共享内存架构的简单高效，设计出一种新型的空分共享内存交换机，即 SPSMS 交换机。SPSMS 交换机每一个输出端口都拥有一块内存和一个子调度器，去往某个输出端口的包由对应于该输出端口的子调度器进行调度，然后存放于对应于该输出端口的内存中，最后从该输出端口发送出去。不同输出端口的内存之间没有任何交集，这大大减小了内存资源的竞争和冲突，从而降低了数据包经过交

换机的时延。同时由于流经一个输出端口的总的网络带宽是确定的，所以设置该输出端口对应的内存大小不少于总的带宽乘以一个调度单元，从而在内存资源不浪费的情况下基本不会出现包溢出的情况，丢包率大大降低。

（4）通过 OPNET Modeler 对本文提出的 SPSMS 交换机和传统共享内存交换机进行网络模拟与仿真，发现 SPSMS 交换机的性能得到很大改善，在保证 100%的吞吐量的前提下，数据包进出交换机的时延、在缓冲区即内存中的排队时延以及缓冲队列大小都大大降低。

关键词 AFDX 航空网络，共享内存架构，交换机设计，网络模拟

SPACE PARTITIONED AFDX SWITCH DESIGN BASED ON SHARED MEMORY ARCHITECTURE

ABSTRACT

As a new generation of aviation network communication standard, AFDX has been applied to the A380 and other large civil aircrafts due to its advantages of full-duplex, high-bandwidth, low latency, low packet loss ratio and low cost. As the major components of an AFDX network, switches are key devices influencing the overall performance of an AFDX network. Shared memory architecture is widely used but it has imperfect reliability and scalability, because all the input and output ports share only one memory and one scheduler, which will lead to severe resource contention and scheduling latency.

This article employs space partition technique to isolate the memory space for each input and output port and asynchronous scheduling technique to ease the burden of the switch scheduler, and proposes a new space partitioned shared memory switch (SPSMS) architecture for AFDX networks with these two techniques.

The main work and research results are summarized as follows:

(1) Make a deep research into AFDX protocol and AFDX network, including its components, characteristics and properties which are different from other types of networks'. Avionics Subsystems, AFDX End Systems (ES), AFDX Interconnect and Virtual Links (VL) are four quite significant parts of an AFDX network. And one of outstanding characteristics an AFDX network owns is the stability of network capacity, bandwidth and routing tables of AFDX switches. It is realized that AFDX switches play a quite important role in an AFDX network.

(2) A broad and deep survey of various types of switches is conducted, and then the

shared memory architecture is emphasized due to its simple, low hardware cost and high hardware utilization ratio. The advantages and disadvantages of shared memory architecture are summarized. And it's pointed out that whether a property of shared memory architecture is suitable for AFDX networks.

(3) Combined with the stability nature of AFDX networks and the plainness and efficiency of shard memory architecture, a new space partitioned shared memory switch design is proposed, namely SPSMS switch. First, SPSMS switch reduces the competition for memory resources and avoid conflicts, thereby decreasing the latency of packets through the switch. Furthermore, packet overflow is reduced to a great extent so long as the size of the memory corresponding to some output port is larger than the product of its highest flow rate and scheduling unit, which can be achieved since the avionic network flows are predictable. Due to this, the packet loss ratio is also greatly decreased.

(4) Simulation of SPSMS and plain shared memory switch with OPNET Modeler shows significant performance improvement over a classical shared memory switch in terms of overall switch packet latency, queuing delay and queue size.

Keywords AFDX avionics network, shared memory architecture, switch design, network simulation

目 录

1 绪 论

1.1 引言

航空全双工交换式以太网（Avionics Full-DupleX switched ethernet，简称为AFDX）作为新型的航空数据总线，其目的是为高性能航空电子系统提供安全可靠的传输方案，满足复杂和恶劣条件下的飞行控制要求。早在上世纪末，世界航空业界的巨头波音和空客公司就开始研究下一代飞行控制总线，以适应多种多样的数据流，包括飞行控制数据流、传感器采集的飞行状态数据流以及为乘客服务的多媒体数据流等。鉴于传统以太网已经能很好地处理多级别多服务数据流，所以两大公司的科研重点转向了使用商用的以太网技术对下一代的飞行控制总线进行改造，研究结果导致了AFDX的诞生并立刻投入了应用。空客公司已经成功地将其应用于A380项目上，并且波音公司已经明确表示将在Boeing787上使用该技术。AFDX网络解决方案以IEEE 802.3为基础，针对航空电子系统的特性和需求进行相应适应性的修改，使得AFDX不仅保留有商业以太网的优良特性而且能满足航空网络的实时与可靠性需求。

作为航空以太网数据标准ARINC664的第七部分[1]，AFDX为航空电子子系统定义了电子电气相关的规定和数据交换的协议。AFDX协议在整个设计和研发过程中吸收了很多别的协议的技巧，使用了很多极具创新性的技术，比如对带宽控制采用了多带宽的应用策略，将复杂、健壮和鲁棒性强的异步传输模式（Asynchronous Transfer Mode，简称为ATM）协议和简单且高效的以太网链路进行有效的组合等。所以，相比于早先提出的ARINC429[2]和MIL-STD-1553[3]总线技术，AFDX网络在灵活性、扩展性、模块性、传输速率和效率上都有了很大的提高。

传统的航空数据通信协议中，ARINC429和MIL-STD-1553是研究和应用最为成功与广泛的。ARINC429协议采用的是总线节点拓扑，子系统与子系统之间必须通过双绞线连接才能收发数据，各个子系统之间数据通信速率仅为100kpbs，只为AFDX网络的千分之一，同时航空电子系统必须要给每个子系统的各个通信信道分配一条具有多个端点的ARINC429总线。这样的网络设计，使得整个ARINC429总线系统光双绞线的质量和成本就非常大。相比而言，AFDX采用的是交换式以太网架构，各个子系统直接与交换机相连接，通过交换机来相互通信，

新增加的设备只需要将其与交换机端口相连接即可。同时，在ARINC429的总线架构中，一个发送端最多可以发送至20个接收端，而在AFDX网络中，由于交换机的层次与数量可以按需增加，所以可用的接收端数量只取决于交换机端口的数目。所以，不管是从扩展性还是从传输速率上考虑，AFDX网络比ARINC429的总线架构都具有更明显的优势。与ARINC429类似，MIL-STD-1553协议也是采用总线架构，但是其传输速率过低，仅为12.5kbps。缺陷更明显的是，MIL-STD-1553协议采用半双工和异步传输模式，需要增加一个总线控制器对总线上挂载的所有设备之间的通信进行控制。这种设计在扩展性、可靠性以及传输速率上与AFDX都有不小的差距。

在AFDX网络中，最重要的组成部分是由AFDX交换机互连组成的AFDX互连系统（AFDX Interconnect），所有设备之间的数据通信都经由AFDX互连系统的路由转发，最后到达目的设备。对于航空网络这样一个对任务和数据处理具有强实时性要求且安全性要求极高的系统中，在特定资源的条件下，数据包能否在给定时延上限内到达目的设备，是至关重要的，数据包超过时延上限到达和没有到达一样会造成致命的错误。而AFDX互连系统主要由AFDX交换机组成，所以AFDX交换机的性能对于整个AFDX网络的性能起着举足轻重的作用。

传统的计算机网络例如ATM网络和以太网中，交换机技术已经非常成熟，有非常多的交换机架构模式，其中共享内存架构相比其它的架构，有硬件成本低、硬件利用率高、吞吐量大等优点。然而，共享内存架构也有明显的缺点，就是其容量不易扩展，数据包经过交换机的时延与抖动非常大。

本文主要基于共享内存架构，充分利用AFDX网络的特性，采用空间分离和异步调度技术，提出一种适合AFDX网络的交换机架构，并与传统的共享内存架构进行对比。其中主要借鉴与参考了一些国内外相应研究机构与研究人员对AFDX网络和交换机研究的成果来进行相应的研究。

该研究成果将应用于科技部中加合作项目“基于AFDX航空总线的下一代飞行数据网络中强化安全级别的研究”项目中。

1.2 论文研究目标

本论文研究目标是提出一种基于共享内存架构和空间分离的AFDX交换机架构。首先，分析AFDX网络的特性及其对交换机设计施加的影响，同时分析共享内存架构的优缺点；其次结合AFDX网络的特性，分析针对共享内存架构扬长

避短的方法，在此基础上提出一种新的适合AFDX网络的交换机架构；最后，通过软件模拟来验证新的交换机架构比其它交换机架构性能更优异。具体研究包括以下几点：

（1）AFDX网络的建模与分析。对AFDX网络中的网络实体（终端系统和交换机）进行概念和性质分析，例如时延需求、带宽需求等，尤其是交换机的各种性质。

（2）共享内存架构分析。对共享内存架构交换机的优缺点进行详细的剖析，总结其适合与不适合AFDX网络的各种原因并探讨可能的解决方案。

（3）结合AFDX网络的特性与共享内存架构的优缺点，提出一种新的适合AFDX网络的交换机架构，包括采用的方法与技术、性能分析与数值模拟等。

（4）使用网络仿真软件OPNET Modeler作为AFDX交换机仿真的工具。从构建基本的数据包格式开始，到数据发送节点和数据接收节点，进而到队列和交换机节点，最终形成一个完整的交换机。

（5）使用OPNET建模传统的共享内存交换机与本文提出的SPSMS交换机，在相同的数据包到达速率下，比较二者的性能，并研究不同端口数对交换机性能的影响。

1.3 国内外研究现状

AFDX 是为实时性和安全性极高的应用而提出的一种数据网络[1,2,3]，在只能利用专属带宽的情况下，AFDX 也能提供确定性的网络 QoS。AFDX 六个主要的特性是全双工、冗余性、确定性、高速、交换式网络和异构网络。由于基于 IEEE802.3 以太网技术，AFDX 显著地减少了需要的连接线，从而减少了整体的飞机重量，对客机来说就能提高载人数量。围绕 AFDX 已经展开了许多的研究。陈昕、王伟鹏[4,5]等人对 AFDX 端系统关键技术进行了深入研究，同时王辉等人[6]对于 AFDX 终端软件的测试策略和技术展开了研究，而李浩峰[7]对 AFDX 网络整体的测试进行了研究和分析。除了针对 AFDX 终端的研究之外，有大量学者和研究人员对通信链路以及 AFDX 的冗余管理机制进行了研究[8,9,10,11]。另外一个关于 AFDX 网络比较多的研究是计算 AFDX 网络的端到端时延，因为确定性是 AFDX 网络区别于其它类型的网络的最重要的属性之一。在计算时延方法上，网络微积分[12]应用最为广泛，出现了一系列与此相关的研究和文章[13,14,15]。以上研究集中于 AFDX 网络的冗余性和确定性等方面，对于 AFDX 网络的高速交换

技术研究的比较少，虽然 AFDX 标准鼓励使用 COTS 产品，然而 AFDX 网络对传统以太网做了许多改进，在设计交换机时许多必须考虑的因素和限制条件也跟着发生了变化，所以针对 AFDX 网络设计适合它的交换机还是很有价值的。

在传统网络例如 ATM 网络中，交换机技术已经发展得非常成熟了，Newman、Awdeh、Oie 等人总结了各种各样类型的交换机架构[16,17,18,19,20,21,22,23]，也针对它们进行了分类，总体说来有三种分类方法，第一种是根据交换机结构（Switch Fabrics）来分类，典型的有空分交换机和时分交换机，两者又可以细分为很多小类；第二种是根据缓存存放的位置，典型的有 Input-Queued 交换机、Output-Queued 交换机以及它们的各种组合和变种，所谓 Input-Queued 交换机就是缓存放置于输入端口处，而 Output-Queued 交换机就是缓存放置于输出端口处；第三种是根据冲突发生时的解决机制来分类，典型地策略有简单地丢弃数据包，或者将数据包重新定向至输入端口。在这些林林总总的交换机中，共享内存架构以其简单、硬件成本低、能源消耗低等优点获得了广泛的研究和应用。Kuwahara、Endo 和 Kozaka 等人可能是最早提出共享内存交换机并用 VLSI 实现的一批学者[24,25,26]，在他们提出的交换机架构中，输入与输出端口数均不是很多，一般为 8*8 或者 16*16，最多也就是 32*32，所以其许多的设计决策并未考虑扩展性，当输入端口数增加或者速率增大或者输入数据流优先级别增多时，交换机性能显著下降。为了解决容量和带宽受限的问题，Barri 等人提出以小的交换机来构建大的交换机的思想，他称这些小的交换机为交换机元素[27]。Shobatake 等人构建了一个非常小的能放在一块芯片上的交换机，并且这个小的交换机可以用于构建更多端口数能满足更高带宽和容量需求的交换机[28]。提出的这些解决方案都由于共享内存的本质而或多或少在某一方面局限性非常大，从而很多的学者都从理论上来分析共享内存的性能上限和下限。这些文章都利用了排队理论[34]的知识。Liew 等人针对异构交换机模块的缓存策略进行分析 [29]，Prycker 等人以分布式缓存来取代原来本地内存共享 [30,31]，Causey 等人分析不同共享策略例如全部共享、部分共享等对交换机整体性能的影响[32]，Choudury 等人分析层次共享内存架构的性能[33]。这些文章都从不同方面说明共享内存架构在特别的需求和引入特别的设计技术下都可以突破性能瓶颈。

1.4 论文结构

本文一共分为七章。

（1）第一章 绪论。详细介绍本文研究的目标、国内外研究现状、以及概括

本论文研究的关键技术与所使用的理论。

（2）第二章 背景介绍。主要介绍 AFDX 网络的架构以及共享内存交换机的架构，阐述主要的技术难题。

（3）第三章 系统模型。主要介绍本文提出的交换机架构，包括每个组成部分和利用 AFDX 网络静态特性解决共享内存架构性能瓶颈的方法等。

（4）第四章 分析。分析本文提出的架构的性能优劣，并从理论和数值模拟上分别给出相关的结果。

（5）第五章 实验平台介绍。主要介绍了 OPNET Modeler 软件的构成和使用。

（6）第六章 实验仿真与结果分析。本章讲述如何使用 OPNET Modeler 来仿真 AFDX 交换机，包括如何构建 AFDX 数据包、AFDX 链路和各种网络节点等。通过 OPNET Modeler 搭建不同类型以及不同端口数的交换机，测量每一种情况下的端到端时延、排队时延以及队列大小等，同时验证相应的结论。

（7）第七章 总结与展望。对于本文提出的交换机架构进行总结，提出不足与将来的工作。

2 背景介绍

本章主要介绍 AFDX 网络的发展过程、基本特性以及主要构成部分。同时，介绍共享内存架构的发展历史、基本特性和优缺点。

2.1 AFDX 网络

从上个世纪 90 年代开始，空客公司开启了一系列基于 A380 项目的技术研究，目的是为了评估飞行控制、动力管理和航空数据通信等领域内的新型技术是否能够更好地提升 A380 之类的飞机的性能。由于多方面因素包括商业因素和技术因素等的驱使，在航空网络系统数据通信方面，空客公司的计划是开发新型的数据总线系统来取代之前大型飞机上常用的 ARINC429[3]总线系统。在当时，以太网技术相对来说已经比较成熟，利用软硬件开发商提供的许多 COST（Commercial Off-The-Shelf）组件、硬件设备和软件，以太网可以方便快捷地部署在地面网络通信中。以太网以其设备完全独立于网络拓扑结构以及易于扩展等优点成为了空客公司的首选。最终，使用全双工交换以太网同时对其进行相应的修改以满足航空电子系统的需求导致了 AFDX 协议的诞生。2005 年 6 月 27 日，民用通用化航空标准化机构美国航空无线电设备通信公司和美国航空公司电子工程委员会飞机数据网络工作组正式规定 AFDX 协议成为一项确定性标准。此标准对航空电子子系统之间数据交换的协议和电气规范进行了明确的定义，AFDX 协议的内容为 ARINC664[1]的第七部分。ARINC664 的内容还包括了以太网在飞机上的机载通信系统的应用规范、以太网传输的特性以及向未来 IPv6 协议扩展等多方面规格说明。

AFDX 协议基于 IEEE802.3，充分利用和吸收了 TCP/IP 等协议的优点，是首个被广泛使用在航空电子子系统的开放式标准与网络协议。AFDX 协议在整个开发与设计过程中，引入了不少先进的理念，比如：

（1）带宽保障。通过使用多带宽应用方式和排队管理策略实现了对带宽的接入性控制。使用“最小包间隙（Bandwidth Allocation Gap，简称 BAG）”概念和限制最长数据帧长的方式来进行带宽的分配。将这些保障性措施应用到 AFDX 网络的虚拟链路数据流上，有效地提高了数据包传输的完整性和传输时延的确定性。

（2）虚拟链路与服务保证。在传统以太网中，数据包首先经过上层路由器的路由，然后到达交换机，交换机根据数据包网络目的地址例如 MAC 地址选择输出端口。然而在 AFDX 网络中，引入了异步传输模式（Asynchronous Transfer Mode，简称为 ATM）网络中“虚拟链路”（Virtual Link，简称为 VL）的技术。一个数据发送端，亦即一个 AFDX 终端系统使用唯一的一条 VL 进行数据的发送，每条 VL 都在一条物理链路上传输。在每个数据帧（本文将交替使用数据包和数据帧，代表 AFDX 网络中交换机的调度单元）的包头，都增加了一个 16 位的无符号整数代表虚拟链路 ID，然后 AFDX 交换机根据虚拟链路 ID 查找路由表，将数据帧发送到相应的输出端口上，通过一级一级传送，最后到达目的终端系统。AFDX 网络是第一个将 ATM 协议的复杂性和传统以太网链路的简便性巧妙结合到一起的航空电子标准。

（3）冗余管理。AFDX 网络是基于通信链路物理冗余的交换网络。与通常的逻辑冗余网络不同的是，在 AFDX 网络中，存在着 A 和 B 两个各自独立的物理交换网络。AFDX 终端系统的每个数据包都在 A 和 B 两个网络上传输。所以，在正常情况下，每个目的终端系统都会收到同样内容的数据帧，数据帧携带了标识其是通过哪个物理网络传输的信息，此时冗余管理机制只需选择某一数据帧，删除其附加信息，然后交给上一网络分层。通过这样的方式，即使某一个交换网络因为传输失败或者链路失效，AFDX 网络也能通过另外一个一模一样的物理网络有效地进行安全可靠的数据传输，这对于安全性要求极高的航空数据网络来说是必需的。

AFDX 网络基本的组成部分有航空电子计算机系统（Avionics Computer System）和 AFDX 互连系统（AFDX Interconnect），前者一般包括航空电子子系统（Avionics Subsystem）和终端系统（End System）。如图 2-1 所示。

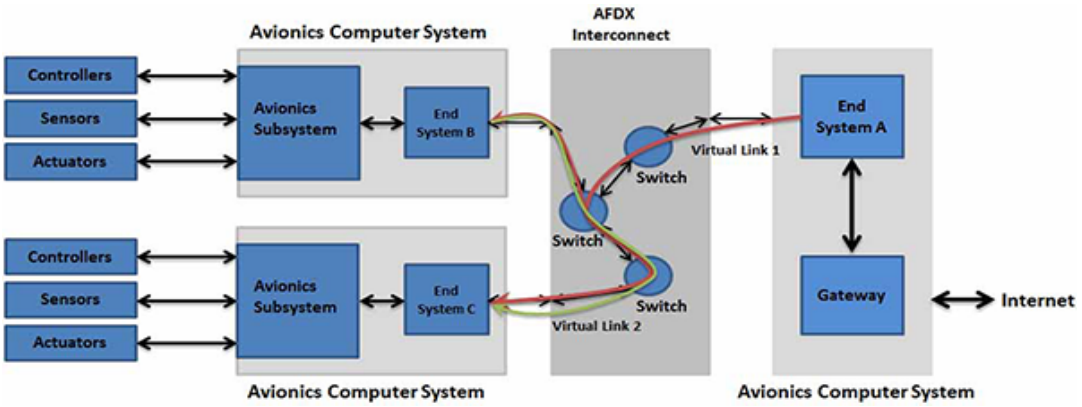

图 2-1 AFDX 网络系统结构图

Fig.2-1 AFDX network topology

（1）航空电子子系统。与传统飞机上的航空电子子系统一样，常见的有飞行控制计算机、全球定位系统、通信系统和轮胎压力监控系统等。航空电子计算机系统为航空电子子系统提供一个快速和可靠的计算环境。每一个航空电子计算机系统都包括一个嵌入的终端系统，航空电子计算机系统与外围其它设备的通信都通过终端系统来进行。

（2）AFDX 终端系统。它为航空电子子系统和 AFDX 互连系统提供了一个通信的接口。终端系统确保了航空电子子系统之间安全可靠的数据交换。通过终端系统开放的应用程序接口（Application Programming Interface，简称 API），不同的航空电子子系统可以以非常简单的消息来通信，而不用考虑底层通信的协议等细节。如图 2-1 所示，终端系统 B 和终端系统 C 为航空电子子系统提供数据通信接口。终端系统 A 为 AFDX 网络提供网关支持。该终端系统的作用是建立 AFDX 网络和外部网络例如 Internet 的通信信道，此举便于 AFDX 系统设计与开发期间数据的导入、导出和调试等。

（3）AFDX 互连系统。它是一个由 AFDX 交换机组成的全双工交换式以太网互连系统。交换机负责将数据帧转发至正确的目的终端系统，这是与传统的 ARINC429 单向且点对点的技术和 MIL-STD-1553 的总线结构都不同的。AFDX 互连系统构成了 AFDX 网络数据传输的主体，任何终端系统之间的数据通讯都经由 AFDX 互连系统的路由和转发。

（4）虚拟链路。AFDX 网络非常重要的一个概念是虚拟链路（Virtual Link，简称为 VL）。虚拟链路建模的是 AFDX 网络中终端系统（End System，简称为

ES）之间数据通信的逻辑关系。一条 VL 只能有一个源 ES，但是目的 ES 可以有一个或者多个。虚拟链路由 16 位虚拟链路 ID 来标识，该 ID 号表示了虚拟链路的唯一性，通过虚拟链路 ID 号，数据包能被发送至一个或者多个终端系统。这一点满足了多播或者广播的设计需求。如图 2-1 所示，存在一条从终端系统 B 到终端系统 C 的虚拟链路，也存在一条从终端系统 A 到终端系统 B、C 的虚拟链路。每条虚拟链路用两个参数来表示。

- 带宽分配间隙（Bandwidth Allocation Gap，简称为 BAG）。BAG 通常以毫秒为单位，取值为 2 的指数次方，范围为 1 到 128 毫秒。BAG 含义为虚拟链路上发送以太网数据帧的最短时间间隔。
- 数据包最大帧长（$L_{\max}$），表示了虚拟链路上能够发送的最长的以太网数据帧的字节个数。

虚拟链路最重要的特性是其最大带宽是确定的，不会出现突发情况，并且不随时间而变化。举个例子，虚拟链路的 BAG 为 32ms，$L_{\max}$ 为 540 字节，则虚拟链路的最大网络带宽为 $540*8bits/32ms=135000bit/s$。

2.2 AFDX 交换机

按照 ARINC667 第七部分的规定和要求，一个 AFDX 交换机应该具有仲裁过滤（Filtering & Policing Function）、交换功能（Switching Function）、监控功能（Monitoring Function）和配置表设置（Configuration Tables）的功能。仲裁过滤功能主要是防止非法的数据帧进入交换机，例如长度超出 $L_{\max}$ 或者由于噪声而导致校验失败或者不完整的数据帧等，仲裁过滤功能模块是数据包进入交换机第一个必须经过的模块。监控功能主要是监管整个交换机的工作状态是否正常以及创建不同状态变量的统计量，包括丢弃的分组、成功转发的分组、每个交换端口的吞吐量等参数。配置表设置允许离线设置 AFDX 交换机的路由表、路由表中的虚拟链路参数等。本文主要研究交换功能以提高交换机的整体性能。

2.3 共享内存交换机

作为数据链路层上的网络互联设备，交换机技术已经非常成熟，发展出了许多架构类型的交换机，架构类型不同，交换机的吞吐量、时延、抖动、丢包率等性能参数都会不同。交换机结构（Switch fabrics）是交换机最重要的组成部分之

一。如图 2-2 所示。

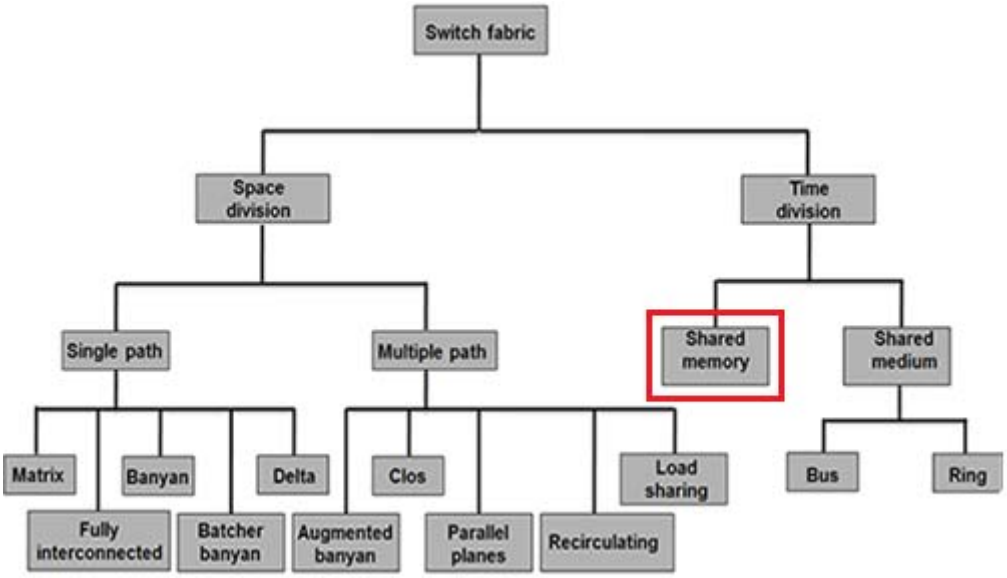

图 2-2 交换机结构分类

Fig.2-2 Classification of switch fabrics

首先交换机结构分为空分（Space division）和时分（Time division）两大类型。空分结构根据一个输入端口到一个输出端口之间是否存在多条路径分为单路径（Single path）和多路径（Multiple path）两种。然后每种子类型下面又会细分成许多类型。至于时分结构，根据共享的组件不同，划分成共享内存（Shared memory）和共享介质（Shared medium）两种。对于时分交换机来说，最明显的特点是由于共享硬件而有硬件成本低和硬件利用率高的优点。所以共享内存交换机一样地拥有上述优点，除此之外，共享内存交换机还有许多别的优点。

早在上个世纪八九十年代，共享内存交换机就已经被提出并应用于商业应用中，发展到今天，共享内存架构交换机已经相当成熟。相比于其它架构的交换机来说，共享内存交换机拥有以下非常明显的优势。

（1）所有的输入和输出端口都共享一块内存，所以理论上来说，每一个输出端口对应的缓冲区的容量都是整个内存的大小，从而很少出现缓冲区溢出的情况，亦即丢包率低。

（2）由于共享一块内存，而不是为每一个输出端口都配备一块内存，这一方面减少了硬件成本和资源浪费，另一方面也提高了硬件利用率。

（3）任何时候只要有数据包去往某个空闲的输出端口，它即刻可以从该输出端口出去，而不会受其它输出端口状态的影响，所以与 Output-Queued 交换机

一样，共享内存交换机的吞吐量也是 100%。

虽然具备上述优点，但是工业上共享内存交换机的应用相比其它类型架构的交换机来说优势并不明显，它的研究兴趣绝大部分来自于大学和研究机构中。上述优势的确使共享内存交换机成为一种性价比非常好的交换机，然而优势的另一面却成为限制共享内存交换机广泛应用的劣势。例如由于所有输入端口到达的数据包均写入同一块内存，所有输出端口出去的包均从同一块内存读取，所以共享内存的容量和带宽成为整个交换机容量和带宽的瓶颈。

典型的共享内存交换机结构如图 2-3 所示。

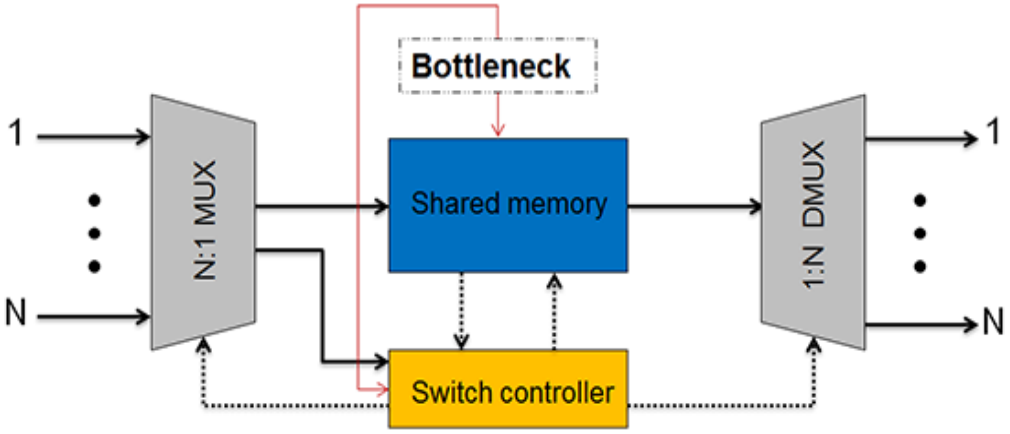

图 2-3 共享内存交换机结构

Fig.2-3 Shared memory switch fabrics

图中 N 代表输入端口和输出端口的个数，数据包首先经由 N 路复用器（N:1 MUX）处理，然后存储入共享内存（Shared memory），经过一定的排队时延，最后从内存中读取出来，通过分路器（1:N DMUX）发送出去，这一系列步骤均由交换机控制器（Switch Controller）或者调度器（本文将交叉使用控制器和调度器，二者均指 Switch Controller 这个部件）统一协调控制。共享内存和交换机控制器便成为整个共享内存交换机的性能瓶颈，成为共享内存交换机应用于 AFDX 网络的技术难题。

2.4 技术难题

由于共享一块内存和一个调度器，数据包经过交换机的时延非常长且抖动非常大，使得共享内存交换机很难满足 AFDX 网络实时性、可靠性和确定性的要求。

2.4.1 共享内存难题

第一个需要解决的技术难题是共享一块内存带来的非确定性。输入端口处的多路复用器（Multiplexer，简称为 MUX）将来自 1 到 N 号输入端口的并行数据串行存储进共享内存，如图 2-4 所示。

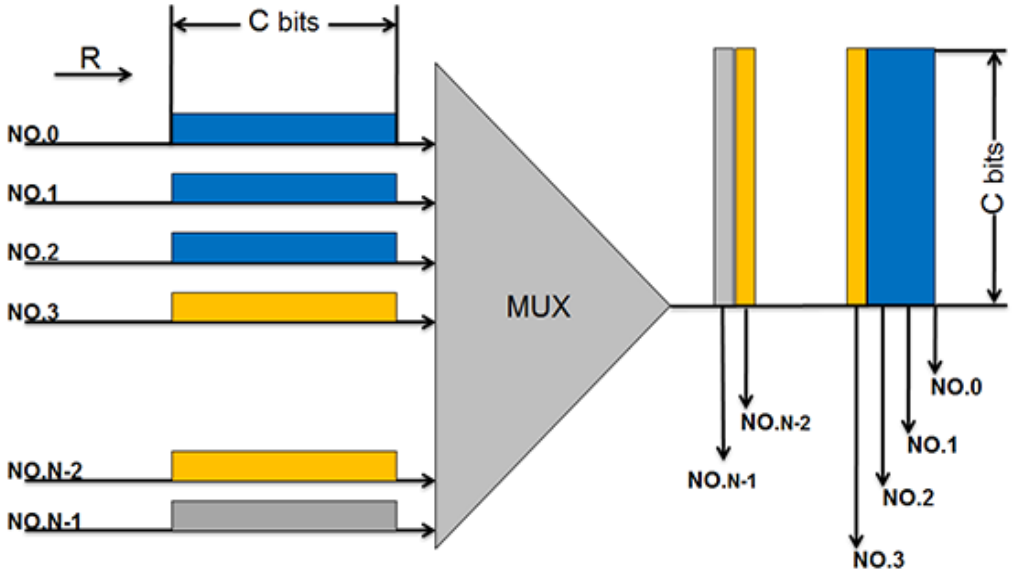

图 2-4 复用器的数据单元处理

Fig.2-4 Cell handling of MUX

图中 N 代表输入端口的个数，标号为 NO.0~N-1，R 代表每个输入端口的速率，C 代表输入端口到达的包被等长切割的长度。由于复用器并行转串行的作用，交换机在一个周期里面必须同时完成一次读操作和写操作，所以共享内存的带宽为 2NR，亦即共享内存每 $A_t = C/2NR$ 秒就要被访问一次，举个例子，这也是一般交换机的最低需求，当 C=64 字节，N=32 以及 R=10Gb/s 时，那么内存访问周期为 800ps，这个数值远远低于 SRAM 的访问周期，而相比 DRAM 的访问周期来说，更是低了两个数量级，所以需要特别的技术来构造共享内存以满足容量和带宽的需求。通常来说，有以下三种方法。

（1）只以 SRAM 来构造共享内存。虽然可以勉强达到带宽的需求，但是最大的 SRAM 也少于 2048MB，远远低于 AFDX 网络的需求。如果通过多个 SRAM 来堆积，那么交换机的空间占用率将急骤上升，同时这还会引发一系列的连锁问题例如交换机散热不良和硬件成本过高等等。

（2）只以 DRAM 来构造共享内存。虽然可以满足容量的需求，但是 DRAM

的动态随机访问特性使得其带宽远远低于所需的带宽。

（3）SRAM 与 DRAM 相结合。虽然可以满足带宽和容量的需求，但是其复杂度太高，使得硬件设计与实现的成本偏高。

本文将着重于改造第三种方案，然而 SRAM 只用来存储每一块内存的空闲地址，而着重以 DRAM 来存储数据包，这大大降低了设计的复杂度。同时充分利用 AFDX 网络的静态特性，那么构造一个高带宽、低丢包率、低时延和低抖动的共享内存 AFDX 交换机是可能的，这与算法分析设计中用空间换时间有异曲同工之妙。

2.4.2 调度器难题

除了因共享内存容量和带宽受限引起的时延和抖动增大之外，传统的共享内存交换机的调度时延与抖动亦非常大，在每一个时隙里面，交换机调度器必须同时完成两项任务。

（1）轮询每一个输入端口，如果有数据包，根据其输出端口将其放入相应的内存位置。

（2）轮询每一个输出端口，如果有数据包去往该输出端口，则从相应内存中读出数据包，放到该输出端口发送出去。

以上的调度策略会造成一个输出端口的数据包必须等待调度器处理完其它输出端口的数据包之后才能得到下一次处理的机会，如果其它输出端口相对空闲，那么调度时延就低，反之则高，这对于实时性和确定性要求很高的 AFDX 网络来说是不可接受的。解决方法是降低调度器的负担，将调度器从繁琐的访存操作中解脱出来，调度器只负责任务的分配和转发，从而很好地降低调度时延和抖动。

2.5 本章小结

本章简要介绍了 AFDX 网络的发展历史、组成部分及其每一部分的特性，集中阐述了对于交换机设计有重要帮助的因素，扼要地说明了 AFDX 规范中关于交换机设计的规格说明。然后阐述了共享内存交换机在所有类型交换机中所处的位置及其优缺点。最后详细说明了本文需要解决的两个技术难题。

3 系统模型与分析

本章主要介绍本文提出的空间分离共享内存交换机（Space Partitioned Shared Memory Switch，简称为 SPSMS）的架构，以及 SPSMS 架构中用于解决第二章中提出的技术难题的解决方案。然后对 SPSMS 架构的性能进行理论分析和数值模拟。

3.1 SPSMS 架构

图 3-1 是 SPSMS 的整体架构图，主要描述交换机组成模块之间的联系，并未涉及到底层的电路与电气连接与特性。交换机有 N 个输入端口、N 个输出端口和 N 块并行的内存，每个输入端口与输出端口均与一个直接存储器访问（Direct Memory Access，简称 DMA）相连，每一个 DMA 与对应的内存相连。这些部件均由交换机调度器(Switch Controller)来统一控制以相互协作完成数据交换任务。

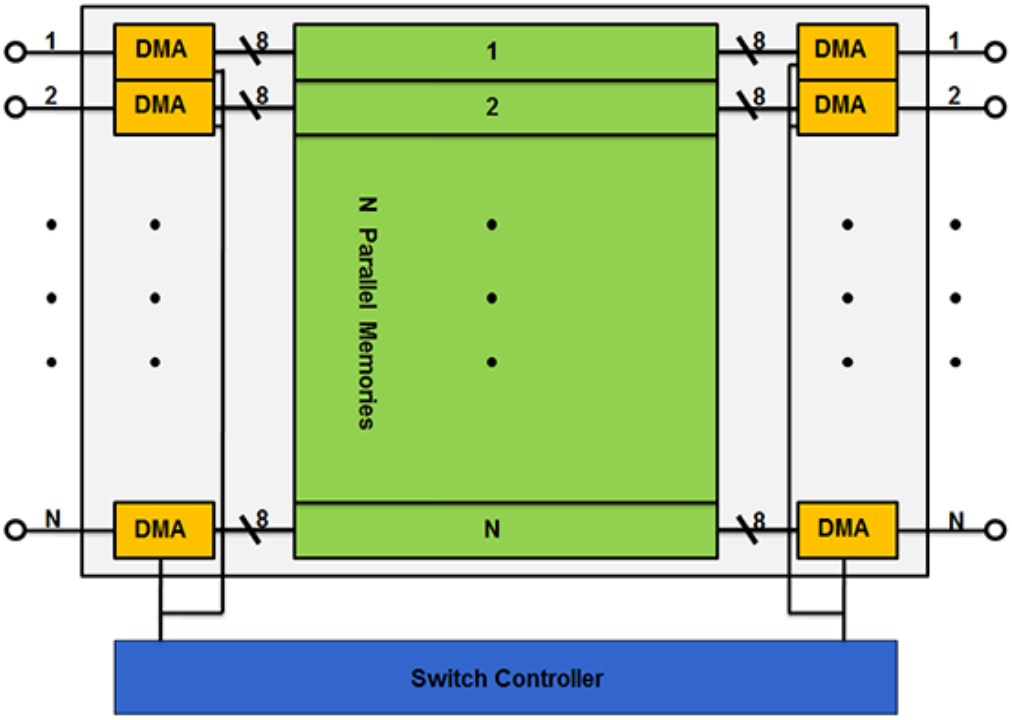

图 3-1 SPSMS 交换机架构图解

Fig.3-1 Architecture of SPSMS

图中每一个 DMA 通过一个 8 位的数据总线与相应的内存相连接，为了简化作图，此处 DMA 将集成 DMA 控制器（Direct Memory Access Controller，简称 DMAC）的功能。需要特别注意的是图中 N 块内存相互独立，亦即它们不共用任何数据传输总线，所以消除了数据总线的冲突。它们的大小并不是等同的，需要根据输出端口的带宽来动态配置。下面分别阐述 SPSMS 有别于其它架构的两个技术，空间分离与异步调度。这两种技术

3.2 空间分离

在图 2-3 中，传统的共享内存交换机是让所有的输出端口共享一块内存，但是 SPSMS 交换机针对每一个输出端口分配一块合适大小的内存，既保证内存资源不会出现严重浪费，又降低内存资源的共享和竞争。我们称之为空间分离（Space Partition，简称为 SP），这与图 2-2 中的 Space division 是不同的，二者针对的对象不同，前者是针对缓冲区亦即共享内存，后者一般是交换电路。空间分离并不是均匀地分割整块共享内存，甚至不是分割，而是根据输出端口的带宽为每个输出端口配备一块相应大小的各自独立的内存。

AFDX 网络最显著的一个特点是静态性，一个 AFDX 网络一旦部署成功后，总的终端数、交换机数、虚拟链路数等都是固定的，每一条虚拟链路的带宽上限是确定的，总的带宽上限也是确定的。假设一个 AFDX 网络中总的虚拟链路数为 N_{VL}，虚拟链路 $i(1 \sim N_{VL})$ 的带宽为 s_i，带宽上限为 S_i，则存在以下不等式：

$$\sum_{N_{VL}}^{i} s_i \le \sum_{N_{VL}}^{i} S_i \le M$$

上式中的 M 值为 AFDX 网络总的带宽，与 S_i 一样均为定值。记虚拟链路 i 的路由路径上的某个交换机为 W，输入端口数和输出端口数均为 N，任意选取一个输出端口 $x(1 \sim N)$，由于 AFDX 网络是静态的，通过查询路由表可以得知流经 x 的虚拟链路集合是固定的，设为 V_x，则 x 的带宽满足如下不等式：

$$\sum_{i \in V_x} s_i \le \sum_{i \in V_x} S_i$$

上式右边是一个定值，因此只要分配与 $\sum_{i \in V_x} S_i$ 相对应大小的内存，就能满足输出端口 x 带宽需求。这样的设计使得整个交换机的扩展性非常好。最后的空间分配如图 3-2 所示。

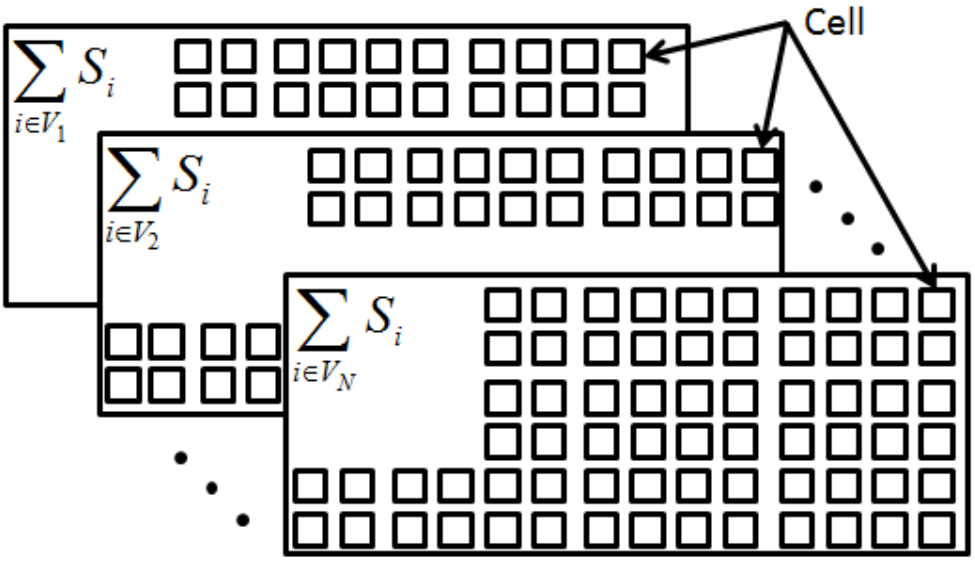

图 3-2 空间分离示意图

Fig.3-2 Space Partition Block Diagram

图中有 N 块内存，内存的大小由左上角的求和公式给出，Cell 为等长切割的包，亦即内存的存储单元。空间分离比一般的并行交换机更优异之处表现为，并行交换机一般是所有输入和输出端口共享很多并行的内存，对于输出端口和输入端口来说，它们看到的是一块虚拟出来的大内存，需要比传统共享内存交换机更加复杂的存储管理算法来完成数据包的存储和空闲空间的维护和更新。然而，空间分离将所有输出端口需要的存储空间组成的集合划分成互不相交的子集，亦即每一个输出端口拥有只属于自己的内存，这一方面简化了内存管理，包括可用空间的维护和更新，另外一方面也便于快速地读写数据包。

3.3 异步调度

在图 2-3 中，交换机只有一个调度器，所有内存访问操作都由调度器控制完成，当调度一个包存储进内存时，调度器必须等待访存操作结束，同时根据返回值判断是否存储成功，进而更新用于存储可用的存储空间地址的数据结构，通常来讲为一个列表，当这些一系列连续的步骤结束之后，调度器才前进到下一个端口。当从内存中读取一个包时亦是如此。由于内存访问的时间具有很大的随机性，同时一个端口的数据包必须等待其它端口的数据包处理完后才能得到下一次处理的机会，这造成一个包从进入交换机到离开交换机的时延和抖动均相当大。

SPSMS 交换机采用空间分离将共享内存变换为 N 个相互独立的内存，每块

内存只需要维护它自己的可用存储空间，并且有专属于它自己的 DMA，由 DMA 来完成具体的数据包存取操作，然后异步地将操作结果值返回给交换机控制器，并更新自己的可用存储空间。下面分别阐述输入和输出的情况。

当一数据帧到达输入端口 $i(1\sim N)$ 时，由调度器给 i 号 DMA 一个控制信号以及该数据帧的目的输出端口例如 $k(1\sim N)$，然后调度器立刻返回，并继续遍历下一个输入端口。i 号 DMA 负责将数据帧存储进 k 号内存中。如果该帧是一个多播或者广播的帧，则调度器给该帧的每一个目的 DMA 都发送一个控制信号，并经由复制模块进行复制，最后由相应 DMA 存储进相应内存中，并将相应结果返回给调度器。

当开始遍历输出端口时，如果 $j(1\sim N)$ 号输出端口有数据帧需要出去，则发送信号给 j 号 DMA，然后返回，继续访问下一个非空闲的输出端口。j 号 DMA 负责从 j 号内存读取数据帧，发往输出端口 j，并将相应结果返回给调度器。

在以上的调度策略中，由于每一次内存存储操作都由 DMA 来控制完成，调度器不必等到每一次内存访问操作结束才返回，这大大降低了调度引起的时延和抖动，同时调度器异步地处理 DMA 的返回结果，并根据处理结果执行相应的操作，这也是取名为异步调度的原因。总而言之，不同的输出端口拥有不同的内存以存储去往该输出端口的数据帧，相互之间没有交集，并且由不同的 DMA 控制，消除了传统共享内存交换机中对于内存资源的竞争，也解耦了输出端口之间的依赖，保留了吞吐量 100%的优点。

3.4 分析

本节开始阐述 SPSMS 性能的理论分析，包括存储访问周期、平均排队时延和平均队列大小，并给出数值模拟结果。

3.4.1 存储访问周期

SPSMS 交换机是用空间来换取时间，并列放置多个内存以平摊高速的存储带宽。设 N 表示交换机的输入端口和输出端口数，R 表示端口速率，C 表示以字节为单位的包长度，则存储器访问周期为

$$T=\frac{8*C}{2*R*N}=\frac{4C}{RN}$$

分母乘以 2 是因为在一个时隙里面，交换机必须同时完成一个读操作和写操作。

在本文提出 SPSMS 交换机中，每一个输出端口均有自己相对应的一块内存，并且相互之间不存在竞争和冲突，所有的内存能并行存取数据包。事实上每一块内存的存储访问速度是不改变的，但是将并行的内存作为一个整体并且忽略调度器发送信号给 DMA 的时间，那么在一个存储器访问周期里面，每一块内存都可以完成长度为C的数据包存储或者读取，此时存储器总体的带宽为$N*C$，相比传统共享内存交换机来说提高了N倍。存储和转发同样多的数据，SPSMS 只需要原来访问时间的$1/N$。

3.4.2 平均排队时延和平均队列大小

由于每一个输出端口对应的内存都配备有 DMA，交换机调度器只需要给有包出去的输出端口的 DMA 发送一个信号即可，然后由 DMA 来控制包的实际传送。这大大降低了交换机调度器的时延和抖动。除了调度时延之外，数据包流经交换机的时延的另外一个主要组成部分为排队时延。排队时延和队列长度均与交换机输入端口数据包到达的数学分布有关系，一般通过将交换机等价地映射为已有的队列模型，然后利用成熟的排队理论知识计算排队时延和队列长度。

在本文中，将传统共享内存交换机映射成一个 M/M/1 模型，即数据包到达速率和服务速率均符合指数分布，同时假设只有一块内存并假设该内存足够大，不会出现包溢出的情况，所以映射成一个服务能力无限的服务器。虽然实际情况会比 M/M/1 建模的结果差，但是给出的结果至少是一个上限。设 M/M/1 模型中数据到达速率和服务速率的均值分别λ和μ，平均队列长度和平均排队时延分别为L和W，则有：

$$L=\frac{\lambda}{\mu-\lambda}$$

$$W=\frac{1}{\mu-\lambda}$$

SPSMS 交换机有 N 个并行的内存，假设每个内存均不会出现包溢出的情况，即相当于有 N 个容量无限的服务器，所以映射成一个 M/M/N 模型，此时的平均队列长度和平均排队时延分别为：

$$L^{'}=\frac{\lambda}{\mu}+\frac{\lambda}{N\mu-\lambda}P_B$$

$$W'=\frac{L'}{\lambda}=\frac{1}{\mu}+\frac{1}{N\mu-\lambda}P_B$$

其中 P_B 为所有服务器亦即内存都处于忙的概率，式中 $\rho=\frac{\lambda}{N\mu}$

$$P_B=\frac{1}{1+N!(1-\rho)\sum_{n=0}^{N-1}\frac{(\frac{\mu}{\lambda})^{N-n}}{n!}}$$

可以明显看出 P_B 是关于 N 的递减函数，且 N 是大于 1 的正整数，当 N=2 时，

$$P_B=\frac{1}{1+2(1-\frac{\lambda}{2\mu})\sum_{n=0}^{1}\frac{(\frac{\mu}{\lambda})^{2-n}}{n!}}=\frac{\lambda^2}{2\mu^2+\lambda\mu}$$

此时平均排队时延差为

$$W-W^{'}=\frac{1}{\mu-\lambda}-\frac{1}{\mu}-\frac{1}{2\mu-\lambda}\frac{\lambda^2}{2\mu^2+\lambda\mu}=\frac{\lambda(4\mu-\lambda)}{(\mu-\lambda)((2\mu)^2-\lambda^2)}$$

由于 $\mu>\lambda>0$，所以上述差值大于 0，从而 $W>W^{'}$，$L>L^{'}$。可见 SPSMS 交换机在平均排队时延和平均队列长度上均比传统共享内存交换机要小。

3.5 数值模拟

下面针对平均排队时延和平均队列长度做数值模拟，数值模拟对于粗略地衡量解决方案最终的效果非常有用。

3.5.1 平均排队时延数值模拟

对于端口数 N 一定的交换机来说，平均排队时延 W 只与平均数据到达速率的值 λ 和平均服务速率的值 μ 相关。本文采取单变量数值模拟，先固定一个变量不变，然后计算平均排队时延随另外一个变量变化的值。

当 $\mu=100$ 时，平均排队时延随平均数据到达速率的数值模拟如图 3-3 所示。

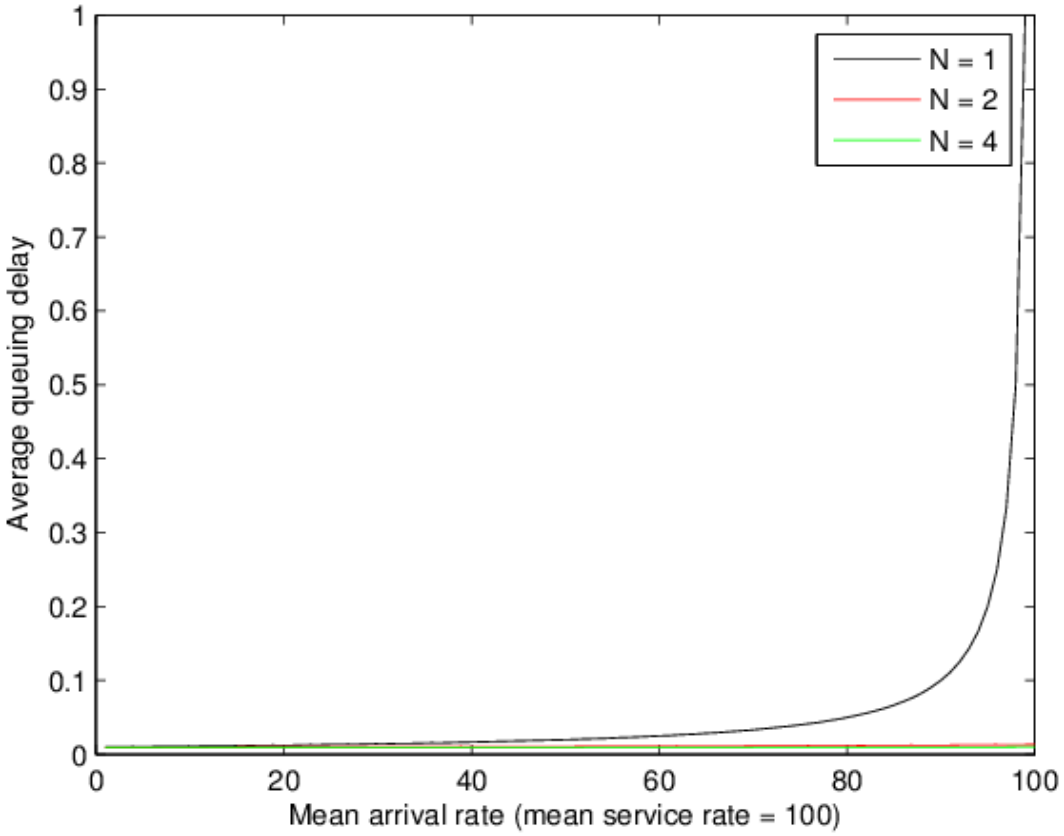

图 3-3 平均排队时延随平均数据到达速率的数值模拟

Fig.3-3 Numerical simulation of average queuing delay with mean arrival rate varying

当 N=1 时，表征的是传统共享内存交换机的平均排队时延随平均数据到达速率的变化情况。当 N>=2 时，表征的是 SPSMS 交换机的平均排队时延随平均数据到达速率的变化情况。N=2 与 N=4 的情况下平均排队时延并没有明显地变化，当 N 更大时亦是如此。然而可以明显看出，随着平均数据到达速率增加，传统共享内存交换机的平均排队时延剧增，SPSMS 的平均排队时延一直维持在一个稳定水平。

固定平均数据到达速率 $\lambda = 50$，平均排队时延随平均服务速率的数值模拟如图 3-4 所示。

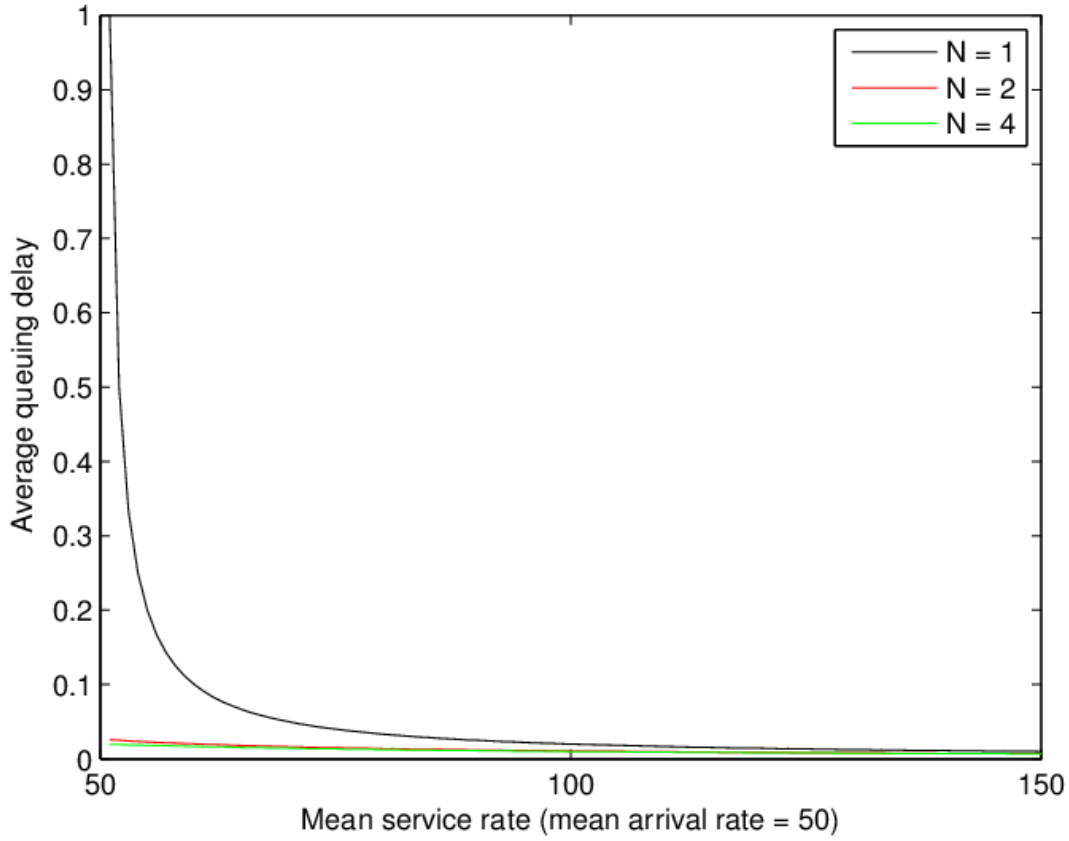

图 3-4 平均排队时延随平均服务速率的数值模拟

Fig.3-4 Numerical simulation of average queuing delay with mean service rate varying

当 N=1 时，表征的是传统共享内存交换机的平均排队时延随平均服务速率的变化情况。当 N>=2 时，表征的是 SPSMS 交换机的平均排队时延随平均服务速率的变化情况。N=2 与 N=4 的情况下平均排队时延并没有明显地变化，当 N 更大时亦是如此。在平均服务速率接近平均数据到达速率的情况下，SPSMS 的平均排队时延明显优于传统共享内存交换机。但是当平均服务速率远大于平均数据到达速率，即交换机服务能力非常强时，二者趋于相同的水平。

3.5.2 平均队列大小数值模拟

对于端口数 N 一定的交换机来说，平均队列大小 L 亦只与平均数据到达速率的值 λ 和平均服务速率的值 μ 相关。本小节亦采取单变量数值模拟，先固定一个变量不变，然后计算平均队列大小随另外一个变量变化的值。

当 $\mu=100$，平均队列大小随平均数据到达速率的数值模拟如图 3-5 所示。

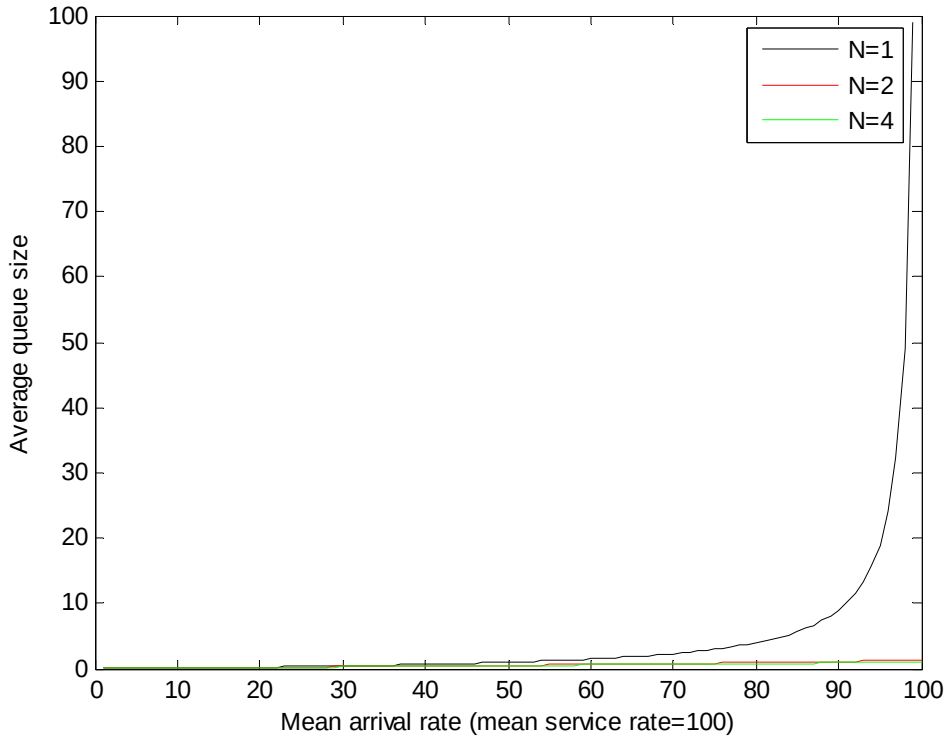

图 3-5 平均队列大小随平均数据到达速率的数值模拟

Fig.3-5 Numerical simulation of average queue size with mean arrival rate varying

当 N=1 时，表征的是传统共享内存交换机的平均队列大小随平均数据到达速率的变化情况。当 N>=2 时，表征的是 SPSMS 交换机的平均队列大小随平均数据到达速率的变化情况。除了纵坐标标度不一样之外，图 3-5 与图 3-3 的变化情况是基本一致的。

固定平均数据到达速率 $\lambda = 50$，平均队列大小随平均服务速率的数值模拟如图 3-6 所示。

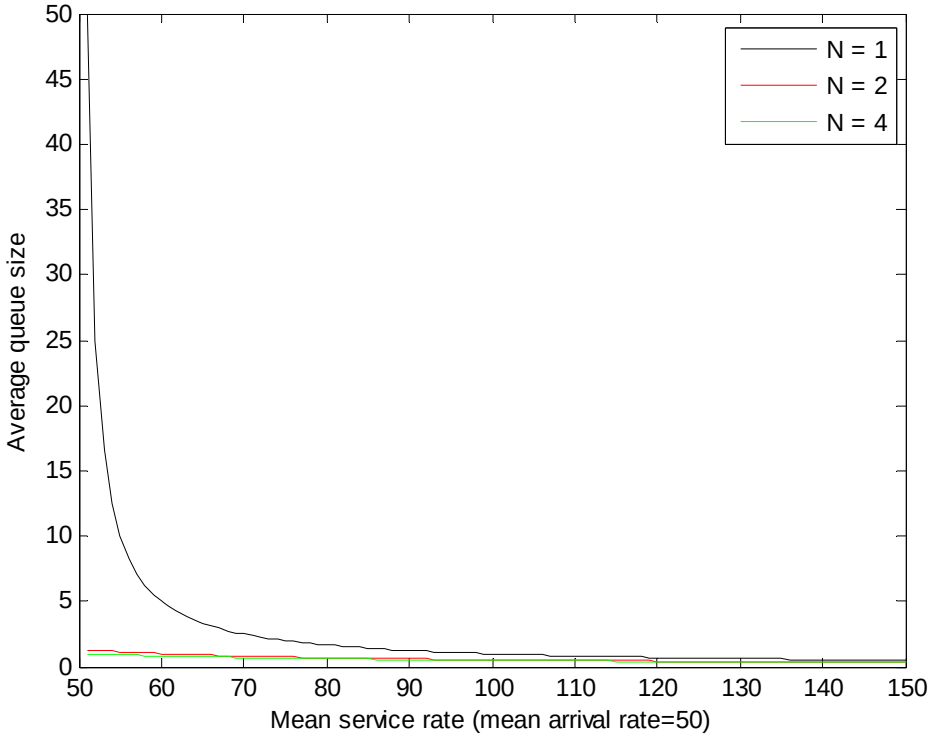

图 3-6 平均队列大小随平均服务速率的数值模拟

Fig.3-6 Numerical simulation of average queue size with mean service rate varying

当 N=1 时，表征的是传统共享内存交换机的平均队列大小随平均服务速率的变化情况。当 N>=2 时，表征的是 SPSMS 交换机的平均队列大小随平均服务速率的变化情况。除了纵坐标标度不一样之外，图 3-6 与图 3-4 的变化情况是基本一致的。

从以上的数值模拟结果粗略地看出，SPSMS 交换机的性能例如平均排队时延、平均队列大小等均比传统共享内存交换机有所提升。

3.6 本章小结

本章针对第二章的技术难题提出相应的解决方案，最终形成了 SPSMS 交换机架构，接着从理论上和数值模拟上对 SPSMS 交换机的性能进行了分析，得出了 SPSMS 交换机的性能相比传统共享内存交换机的性能有很大提升的结论。

4 实验平台介绍

本文前三章分析了 AFDX 网络的特性和共享内存的优缺点，并详细阐述了解决共享内存的缺点以适应 AFDX 网络的技术，并从理论上和数值模拟上进行了分析。本章介绍如何使用 OPNET Modeler 网络模拟软件来进行网络仿真，首先是实验平台的需求，其次是 OPNET Modeler 的介绍，再次是用 OPNET Modeler 进行网络建模与仿真的流程，特别是仿真、分析和重构形成的反馈循环。

4.1 实验平台需求

为了对交换机进行模拟仿真，仿真软件必须具备定制交换机输入端口数据到达速率的数学分布、设计交换机调度器的行为、设置交换机队列的个数和大小以及数据收集和统计等功能。在众多网络模拟仿真软件中，OPNET 很好地满足了上述需求。除此之外，OPNET 还可以自定义底层的数据包格式以使之符合 AFDX 网络虚拟链路的设计需求。在网络模拟仿真领域，OPNET 功能是最强大的，拥有的客户与研究人员数也是最多的。

OPNET 全称为 Optimized Network Engineering Tools，最初是 Alain Cohen 就读于麻省理工学院时一门网络课程的结题项目。1986 年，Alain Cohen 和他的兄弟 Marc 还有他的同学 Steven Baraniuk 一起创立了 OPNET 公司。1987 年，OPNET 公司发布了它的第一个商业化的网络性能仿真软件 OPNET Modeler，提供了意义重大的网络性能优化功能，使得具有预测性的网络性能管理和仿真成为可能。至今 OPNET Modeler 已经发行到了 14.5 以上版本。在网络建模和仿真方面，OPNET 公司的产品线除了 Modeler 之外，还包括 IT Guru、SP Guru、OPNET Development Kit 和 WDM Guru 等。IT Guru 和 SP Guru 着重于自动分析和规划由多种技术、多个供应商构建的复杂的企业级网络，让客户能为网络升级、扩张、更新、技术迁移和部署新应用等做精确的规划，例如能不能升级、升级的成本、升级需要解决的关键技术问题、升级后网络的性能如何、升级后网络的性能瓶颈等等。典型的客户包括大型企业、教育机构和政府部门等。OPNET Development Kit 用于定制 Modeler 的用户界面、创建产品插件和应用以及集成 Modeler 和其它软件。用户可以使用偏好的编辑器、自定义的配置文件、C 或 C++或 Python 等编程语言编写的应用程序接口以特定方式来定制 Modeler 界面以及处理和显

示结果的流程。WDM Guru 是一个高级的多层次网络规划解决方案，通过它可以设计弹性高和性价比高的光纤网络。

本文主要使用 OPNET Modeler 进行网络建模、仿真和分析。下面将简要介绍 OPNET Modeler 的使用。

4.2 OPNET Modeler 介绍

OPNET Modeler 在网络建模和仿真领域功能是最强大的，可以模拟国际标准化组织提出的 OSI 模型的各个层面。然而其使用也最为复杂，前期学习需要花费较长时间。OPNET Modeler 为建模各种通信网络和分布式系统提供了一个综合的环境。通过离散事件模拟，被建模的系统的行为和性能都可以被精确定义和获取。OPNET Modeler 集成了模型设计、仿真、数据收集以及分析等工具。

4.3 OPNET Modeler 结构

如图 4-1 是用 OPNET Modeler 进行网络建模、仿真和分析的常见流程。

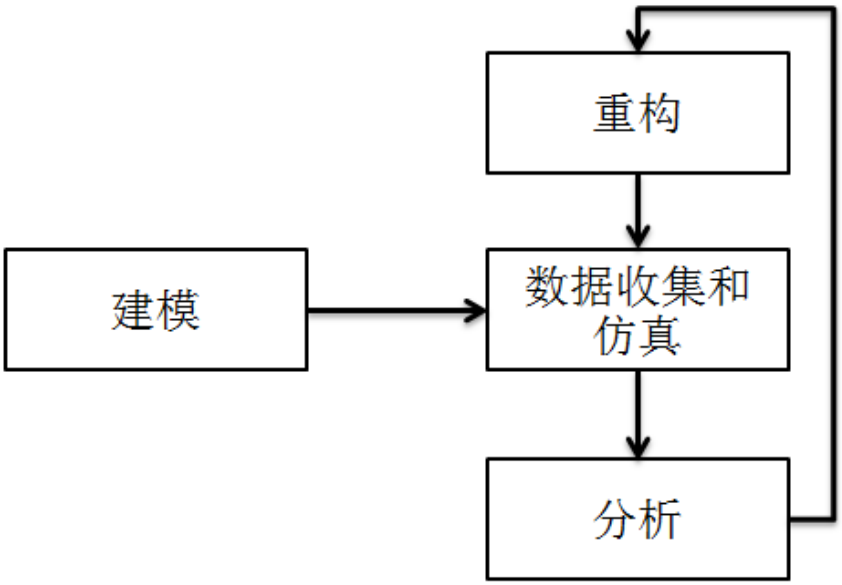

图 4-1 OPNET Modeler 建模仿真的循环过程

Fig.4-1 The cycle of modeling and simulation with OPNET Modeler

建模的最初阶段，依据应用需求建模一个初始的模型，然后设定好需要收集的数据并进行仿真，最后分析收集好的数据。如果结果不符合期望，那就需要重新修改模型，即重构，如果符合期望，就可以发布模型和数据的报告。本文也将

采用这么一个循环来设计交换机。

4.3.1 建模与重构

在图 4-1 中，建模与重构是整个循环中最为重要的部分，其好坏将很大程序上影响最终的结果和分析。OPNET Modeler 中提供了许多的工具，称之为编辑器，来完成一个模型的各个方面设计，其中最重要的是项目编辑器、节点编辑器和进程编辑器。一个模型是由网络、节点和模块三者构成的层次关系。如图 4-2 所示。

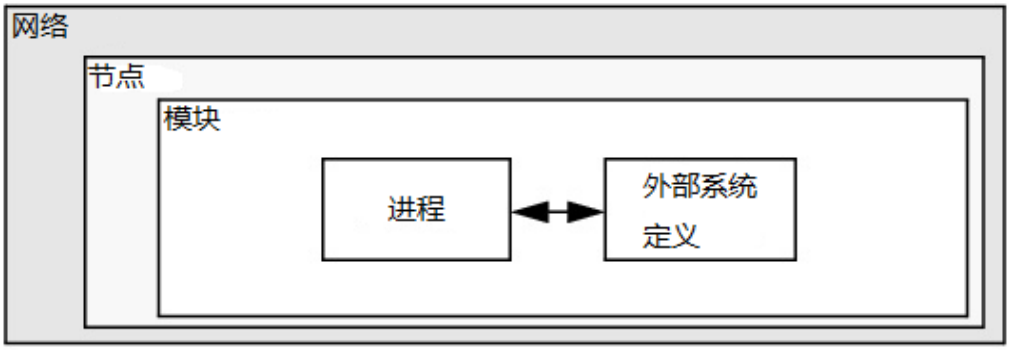

图 4-2 模型中各个部分的层次关系

Fig.4-2 Hierarchical relationship of levels in a network model

模型首先是一个网络，这里的网络与一般意义上的网络不同，这里的网络纯粹上是节点的互连，节点之间可以通过另外一个编辑器——链路编辑器创建的链路相连。节点就与一般意义上的网络中的节点类似，可以接收和发送数据帧，可以有队列存储数据帧，可以有处理器来完成逻辑上的行为。而其中的队列和处理器就是模块，OPNET Modeler 中模块的行为是通过进程来描述的，进程是所有逻辑行为实现的地方，通过有限状态机来描述，当然也可以通过外部系统的定义来实现。

与之对应的，OPNET Modeler 提供了一个层次关系的编辑器。如图 4-3 所示。

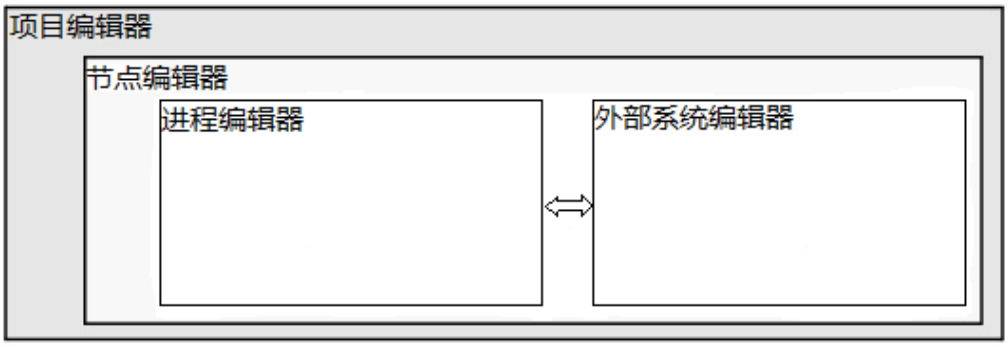

图 4-3 层次关系的编辑器

Fig.4-3 Hierarchical relationships of editors

下面将详细阐述项目编辑器、节点编辑器和进程编辑器。

（1）项目编辑器

在项目编辑器里面，设计的是网络领域的元素以及网络拓扑结构，包括子网、节点、链路和地理上下文。此处子网与传统以太网中的子网完全是不同的概念，并没有子网掩码、网关以及 IP 地址划分等概念，它只是一些节点和链路组成的集合。当一个网络中包括大量的节点与链路时，以子网为单位便于项目编辑器的管理。项目编辑器除了构建和编辑网络通信模型之外，还提供最基本的仿真、数据获取和分析设置选项。如图 4-4 是典型的项目编辑器界面。

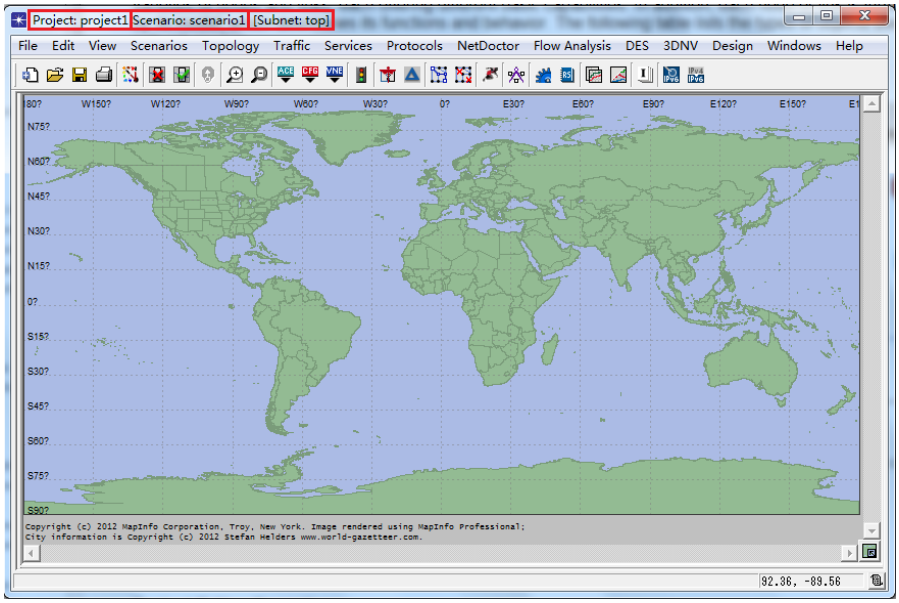

图 4-4 项目编辑器

Fig.4-4 GUI of project editor

最值得注意的是一个项目里面可以有很多场景（Scenario），而每一个场景里面，最顶层的就是名为 Top 的子网，然后在 Top 子网里面，可以添加新的子网，子网里面还可以继续添加子网，如此构建一个复杂的分层的网络拓扑结构。子网里面除了子网之外，还有节点和连接节点的链路。场景是为便于复用已构建好的网络拓扑结构而设计的，在项目编辑器中，可以通过 Duplicate Scenario 功能直观方便地根据一个已有的网络拓扑结构复制一个一模一样网络拓扑结构出来。除了场景、子网、节点和链路之外，项目编辑器另外一个显著的特点是其网络拓扑结构可以建模在实际的地理位置信息上，如图 4-4 所示，背景是整个地球的平面结构图，任何两个节点之间的距离就是实际两个地理位置上的距离，而距离对结果会产生一定的影响。但是 OPNET Modeler 也可以提供另外一种不计较距离的仿真，本文就是采用该种类型的仿真。

项目编辑器将子网、节点、链路等都当成对象（Object）来对待，当需要往一个新建的子网里面添加对象时，是通过对象调色板（Object Palette）组件来完成的。如图 4-5 所示，是一个通用的对象调色板树状结构。

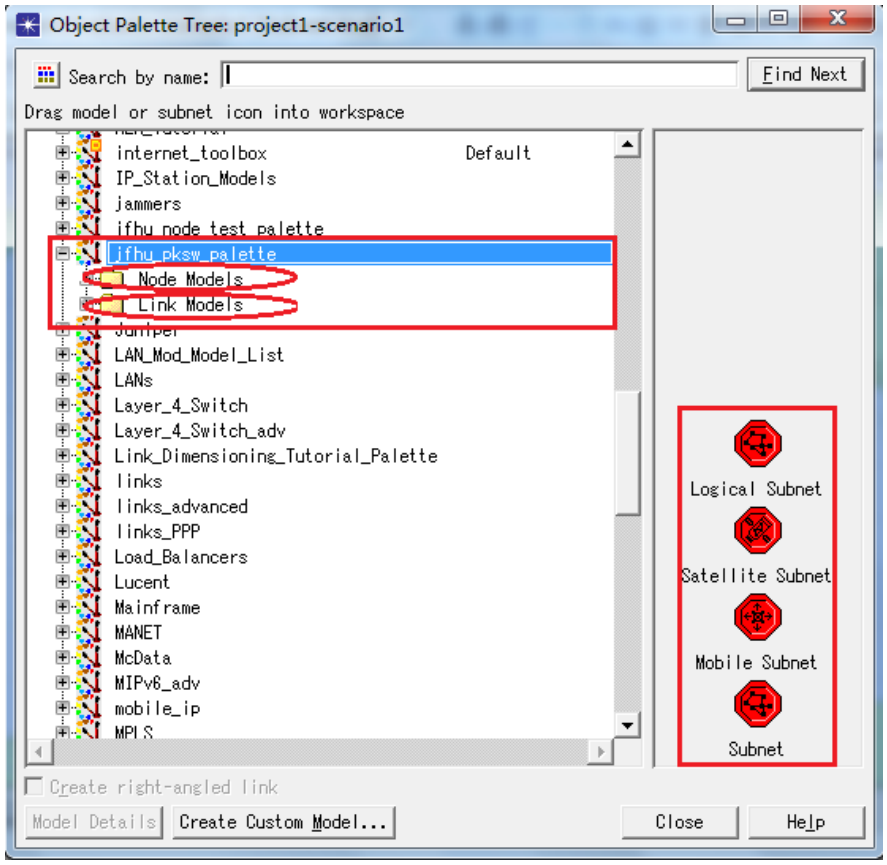

图 4-5 对象调色板树状结构

Fig.4-5 Object Palette Tree

在图 4-5 中，在最右边可以选择不同类型的子网，选择子网后，双击子网就可以编辑子网的内容。左边的列表里面列出了许多的对象调色板，每个对象调色板下面都有节点模型和链路模型，通过这些模型就可以构建各种各样的网络了。

当模型构建完成之后，项目编辑器可以验证模型的底层连接属性是否正确，例如节点里面发射器的速率与链路的传输速率是否匹配，节点里面接收器的速率与链路的传输速率是否匹配等等。如果验证没有不过，就不能进行仿真，这样做的好处是避免由于这些结构上的错误影响最终的仿真结果。

（2）节点编辑器

节点编辑器就是构造设备模型的结构，每一个设备模型在上层的网络领域里面都会被实例化为一个节点对象。除了可以定义设备模型的结构之外，节点编辑器还为每一个设备模型定义一个接口，接口规定了设备模型的哪些方面对用户及上层对象来说是可见的，包括节点的各种属性和统计量。节点编辑器的界面如图 4-6 所示。

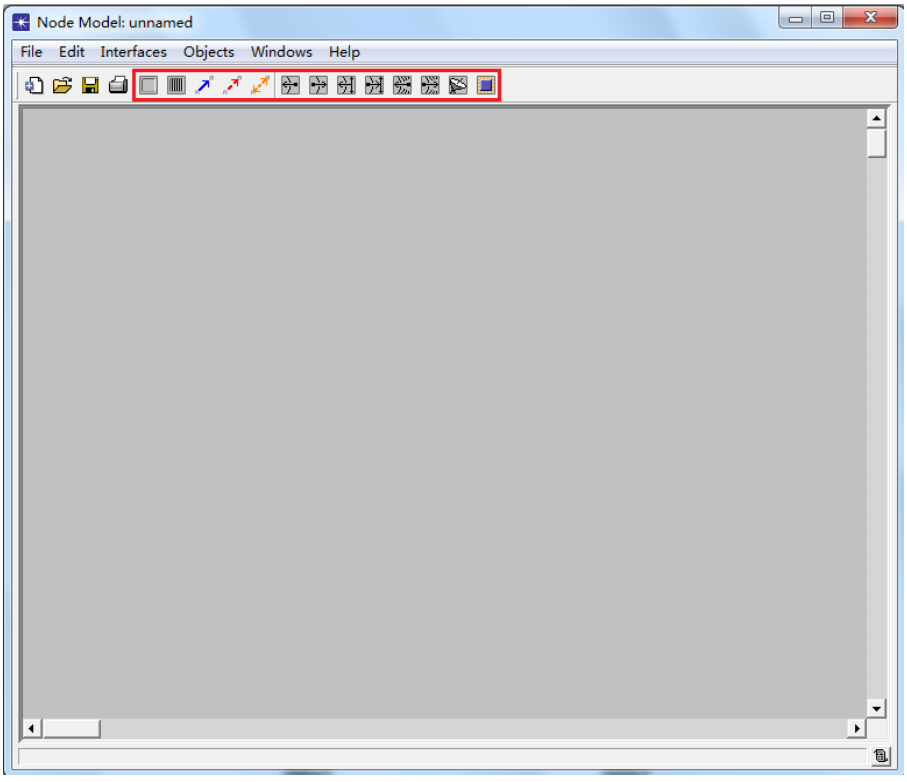

图 4-6 节点编辑器

Fig.4-6 GUI of node editor

在图 4-6 中，能被添加到设备模型中的模块如红色方框标识，从左到右依次

为处理器、队列、包流（Packet Stream）、静态连线（Static Wire）、逻辑连线（Logical Wire）、发送器和接收器，后面的与总线型网络和无线网络仿真相关，此处不再赘述。一个设备模型被实例为节点之后，如果要在网络领域访问节点的属性和统计量，就必须在节点模型里面对这些属性和统计量进行提升操作（Promotion）。

（3）进程编辑器

在节点编辑器建模的节点模型中，最重要的两个模块是处理器和队列，二者均是可编程定制行为的，而这些行为就是由进程编辑器构建的进程模型来描述的。一个进程模型可以被实例化为处理器模块或者队列模块中的进程，进程之间相互独立地运行通信函数或者数据处理函数，这些函数代表能在实际硬件或者软件中实现的功能。与节点编辑器提供一个节点接口一样，进程编辑器也会提供一个进程接口，指定进程模型的哪些方面与属性对用户及上层对象可见，包括进程模型的属性和统计量。

进程模型使用有限状态机（Finite State Machine，简称为 FSM）来描述进程行为，行为依赖于当前状态和触发因素。FSM 通过状态转移图（State Transition Diagram，简称为 STD）来表达。进程状态以及状态之间的转移都是通过图形来表示。进程编程器的界面如图 4-7 所示。

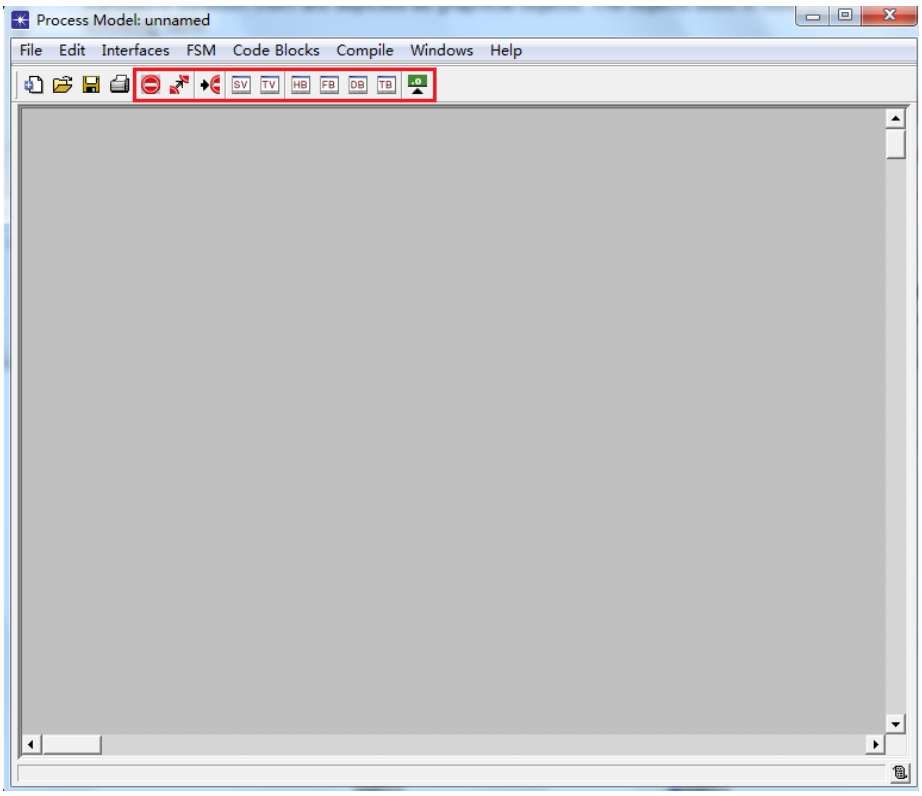

图 4-7 进程编辑器

Fig.4-7 GUI of process editor

一个进程模型包括状态和状态转移，状态可以被设置为初始状态。状态代表的是进程通过前一次触发和相应的决定之后获得的模式。状态包括一些代码，可以指定这些代码是在进程刚进入这个状态时执行还是将要退出时执行。状态可以是强制的或者非强制的。对于非强制的状态，当一个进程进入并执行完相应的代码后，就阻塞并等待下一个中断。状态转移指示的是一个进程从源状态到目的状态的变化。一个状态可以是任意状态转移的源或者目的状态。一个状态转移有一个条件语句说明进程必须满足才能完成这个转移，还有一个函数指定如果进程完成这个转移就必须执行的动作。除了这些之外，进程编辑器可以定义全局变量、数据结构、状态变量、临时变量、自定义的函数以及当进程结束时执行的代码等等。如图 4-8 所示是定义全局变量和函数的编辑器。

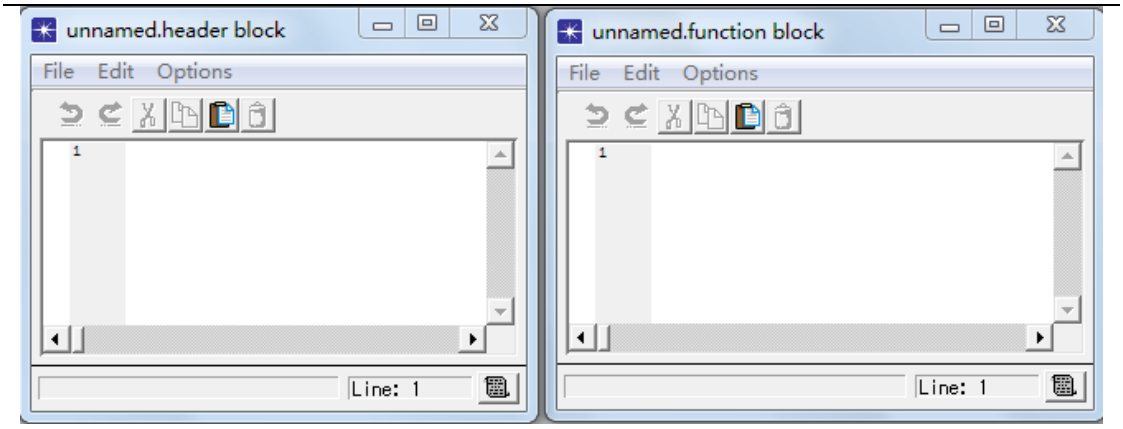

图 4-8 全局变量和函数编辑器

Fig.4-8 Editor of global variables and functions

当构造了所有的状态、状态之间的转移、状态和状态转移执行的代码之后，就可以编译进程模型，只有当所有图形元素与后面的代码都符合语法并正确地建立对应关系之后，编译才能通过。编译成功的代码就是进程模型实例化为处理器或者队列中的进程之后执行的代码。

在一个状态转移图里面，状态及状态转移的示例如图 4-9 所示。

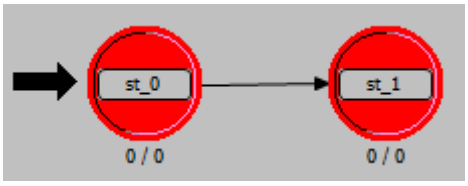

图 4-9 状态示例

Fig.4-9 State samples

左边的状态是初始状态，双击一个状态图标的上半部分可以添加进入状态执行的代码，双击下半部分可以添加退出状态执行的代码。

（4）其它编辑器。其它编辑器中跟本文最相关的就是包格式编辑器和链路编辑器，通过包格式编辑器，可以自定义任何类型的数据包，包括数据包由哪些字段组成，每个字段的名称、类型、长度以及默认值等等。通过链路编辑器，可以定义链路的传输速率、错误处理模型等属性。

4.3.2 数据收集与仿真

网络建模与仿真的最终目的就是为了获取被建模系统的性能参数。通过仿真来模拟系统的性能和行为，然后收集数据进行分析。仿真一般可以输出三种类型

的数据：向量输出、标量输出和动画。

（1）向量输出

网络仿真中最常收集的数据就是一个输出向量，输出向量由一系列条目组成，每个条目都是一个实数对。实数对的第一个通常就是自变量，第二个就是因变量。大多数情况下，自变量就是仿真时间，因变量就是网络模型中固有的或者自定义的统计量，随着仿真的进行，仿真时间单调递增，最后就能获得因变量随时间变化的曲线。有些仿真中自变量不是时间而是仿真序号。当然因变量也可以有很多个。

（2）标量输出

标量输出一般是通过向量输出来构造，只有得到了向量输出，才能通过一定的计算得到标量输出。常见的标量有平均值、概率值、最大值和最小值等。一般情况下，单独的标量值没有什么意义，多次仿真并且每次改变仿真参数得到不同的标量值是最常见的做法。

（3）动画

顾名思义，动画就是以最简单的图形的移动来表征系统运行过程中事件的发生、数据的流动和状态的转换等等，并且在回放动画过程中，可以交互地控制动画的播放以查看系统的各种状态数据。

4.3.3 分析

在建模并且仿真结束之后，数据都以向量形式保存下来。OPNET Modeler 提供了一个结果浏览器，可以以图形来显示经过数值处理的数据，包括向量数据和标量数据，并且可以通过创建过滤器来生成新的结果图形。本文中的结果大量应用了过滤器。

4.4 本章小结

本章主要介绍了 OPNET Modeler 的功能、结构以及仿真流程，重点阐述了仿真流程的每个步骤以及该步骤完成的功能。

5 实验仿真

第四章详细阐述了 OPNET Modeler 的功能、使用方法和建模步骤。本章将使用 OPNET Modeler 来建模传统共享内存交换机和 SPSMS 交换机，然后仿真，收集数据。

5.1 数据帧格式

OPNET Modeler 使用包格式编辑器建模 AFDX 数据帧格式，本文并不打算在 AFDX 数据帧格式上做太多文章，因为 AFDX 数据帧本质上是一个以太网数据帧，但是不同点在于目标地址的格式。AFDX 网络中，目标地址长度与以太网数据帧长度一样，均是 6 字节，但是 AFDX 只使用其中的 2 字节，即 16 个比特，其余 32 位是常量。如图 5-1 所示，是本文实验中用到的 AFDX 数据帧格式。为便于后面的描述，命名为 rtio_afdx_frame。

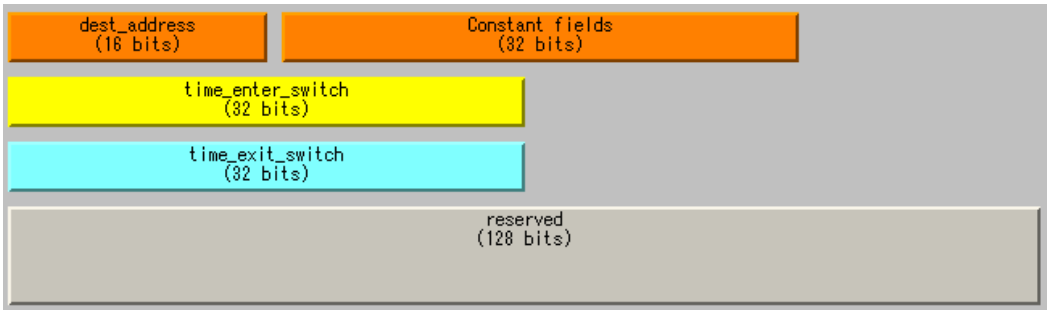

图 5-1 AFDX 数据帧格式

Fig.5-1 AFDX frame format

为了简便，在图 5-1 中，省略了与本文并无太大关系的其它字段，例如 IP 头、TCP 或者 UDP 头等字段。然而为了调试和统计一个数据帧经过交换机的时延，额外增加了两个域，分别表示进入交换机的时刻和离开交换机的时刻，当实际以硬件实现时，这两个字段是要删除的。具体每个字段的名称、长度、类型和描述见表 5-1。

表 5-1 AFDX 数据帧各字段细则

Table 5-1 Details of fields of an AFDX frame

名称	长度（bits）	类型	描述
dest_address	16	无符号整数	目标地址，实际上就是虚拟链路 ID
Constant fields	32	无符号整数	常量
time_enter_switch	32	浮点数	进入交换机的时刻
time_exit_switch	32	浮点数	离开交换机的时刻
reserved	128	结构类型	保留字段

需要注意的是所有字段在创建时可以不赋值，也可以赋一个默认值。

5.2 链路格式

用链路编辑器建模链路格式时，最重要的是链路格式的类型、数据传输速率和兼容的数据帧格式。表 5-2 描述了本文建模的链路的参数，取名为 rtio_afdx_link。

表 5-2 AFDX 链路参数

Table 5-2 Parameters of an AFDX link

名称	类型	数据传输速率	兼容数据帧格式
rtio_afdx_link	点到点全双工	提升（仿真时设置）	rtio_afdx_frame

表 5-2 中，数据传输速率虽然是在仿真时设置，但是必须与链路两端连接的节点的发送速率和接收速率一样，否则建模完成后不能通过验证。

5.3 数据生成节点

节点编辑器组合不同的模块以构建完成某一功能的节点。数据生成节点按照一定的数学概率分布来生成符合 rtio_afdx_frame 格式的数据帧。如图 5-2 所示是数据生成节点的模块连接图。

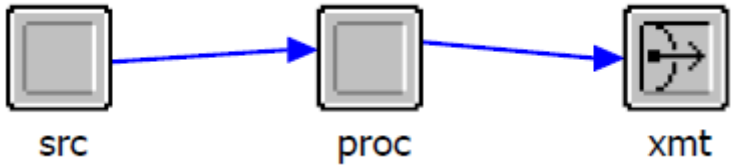

图 5-2 数据生成节点模块图

Fig.5-2 Module interconnection of data generation node

图 5-2 中，src 模块负责生成数据，最重要的三个参数为包格式（Packet format）、包相互到达时间间隔（Packet interarrival time）和包大小（Packet size）。后两者均通过提升（Promotion）操作以便在仿真时可以更改。

src 模块生成的数据帧经过连接 src 和 proc 的数据流到达 proc 模块，proc 模块必须给每一个生成的数据帧设置一个目的地址，即虚拟链路 ID，设置完成之后通过 xmt 发射器发送出去。proc 的进程模型的有限状态机如图 5-3 所示。

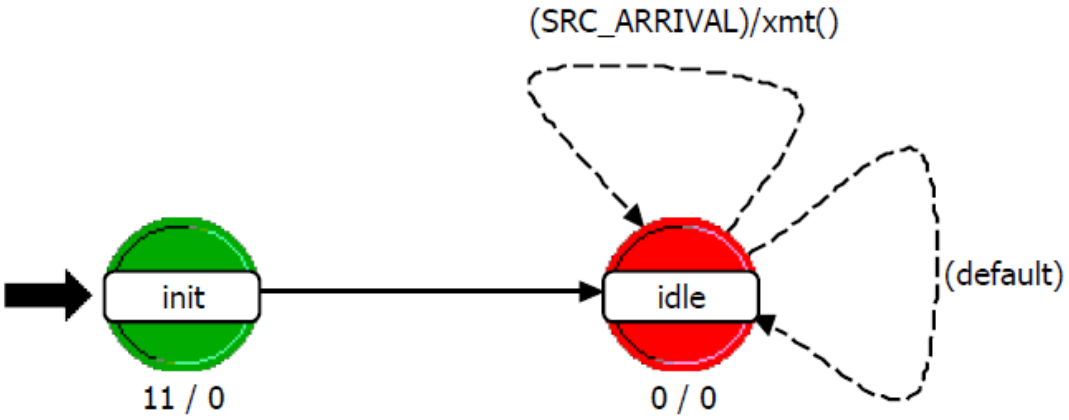

图 5-3 proc 模块的进程模型有限状态机

Fig.5-3 FSM of process model within proc module

进程首先进入 init 状态，完成初始化工作。进入 init 状态执行的代码如图 5-4 所示。

```
op_ima_sim_attr_get_int32("number of output ports", &numberOfOutputPorts);

address_dist = op_dist_load("uniform_int", 0, numberOfOutputPorts - 1);
sent_packet_count = op_stat_reg(
    "sent packet count",
    OPC_STAT_INDEX_NONE,
    OPC_STAT_GLOBAL);
```

图 5-4 init 状态入口执行代码

Fig.5-4 Code of init state enter executives

该代码首先获取交换机的输出端口数 numberOfOutputPorts，然后以均匀分布生成 0~numberOfOutputPorts - 1 之间的虚拟链路 ID，最后是创建一个统计发送数据包数量的统计量。

由于 init 状态是一个强制状态，所以 proc 的进程模型执行完 init 的状态代码之后，立刻转移到 idle 状态，进入一个等待-事件触发-等待的无限循环当中，其中的事件就是帧到达事件 SRC_ARRIVAL，每当一个帧到达时，就执行 xmt 函数，设置虚拟链路 ID。如图 5-5 所示是 xmt 函数的代码。

```
static void xmt(void)
    {
    Packet *pkptr;
    FIN(xmt());
    /* Get the pointer of the packet */
    pkptr = op_pk_get(SRC_IN_STRM);
    /* Assign the packet a randomly generated destination address */
    op_pk_nfd_set_int32(
        pkptr,
        "dest_address",
        (int)op_dist_outcome (address_dist));
    /* Send the packet out */
    op_pk_send(pkptr, XMT_OUT_STRM);
    /* Increment the number of sents packets */
    ++sentPacketCount;
    op_stat_write(sent_packet_count, sentPacketCount);
    FOUT;
    }
```

图 5-5 帧到达事件处理函数

Fig.5-5 Handler of frame arrival event

上述代码首先获取数据包的指针，然后设计数据包的目标地址。设置好的数据帧经由另外一个信道发送出去。

5.4 数据接收节点

数据接收节点除了接收经过交换机转发的数据之外，还会进行很多的计算和统计，例如计算一个数据包进出交换机的时延、端到端的时延、收到的包的个数等等。如图 5-6 是数据接收节点的模块连接图。

图 5-6 数据接收节点模块连接图

Fig.5-6 Module interconnection of data receive node

rcv 接收器将接收到的数据包传送给 sink 模块。sink 模块的进程模型的有限状态机如图 5-7 所示。

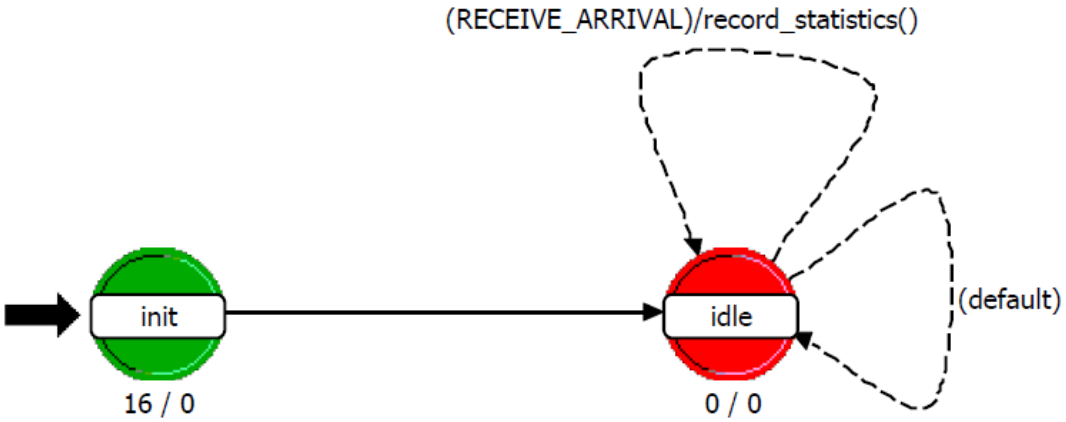

图 5-7 sink 模块的进程模型有限状态机

Fig.5-7 FSM of process model within sink module

sink 模块的进程模型在 init 状态里完成初始化工作后，立刻转移到 idle 状态并阻塞，当有数据包到达事件 RECEIVE_ARRIVAL 发生时就执行 record_statistics 函数进行各种计算与统计，结果均以向量形式保存在文件里面，然后重新阻塞等待下一个事件。

5.5 交换机节点

本节将建模两种交换机节点，分别用来实现传统的共享内存交换机和 SPSMS 交换机。第一种交换机节点命名为单进程单队列交换机节点，用来传统的共享内存交换机的功能和行为。

5.5.1 单进程单队列交换机节点

本小节以输入端口和输出端口均为 32 为例来阐述单进程单队列交换机节点的模块连接图。

如图 5-8 所示是单进程单队列交换机节点的模块连接图。

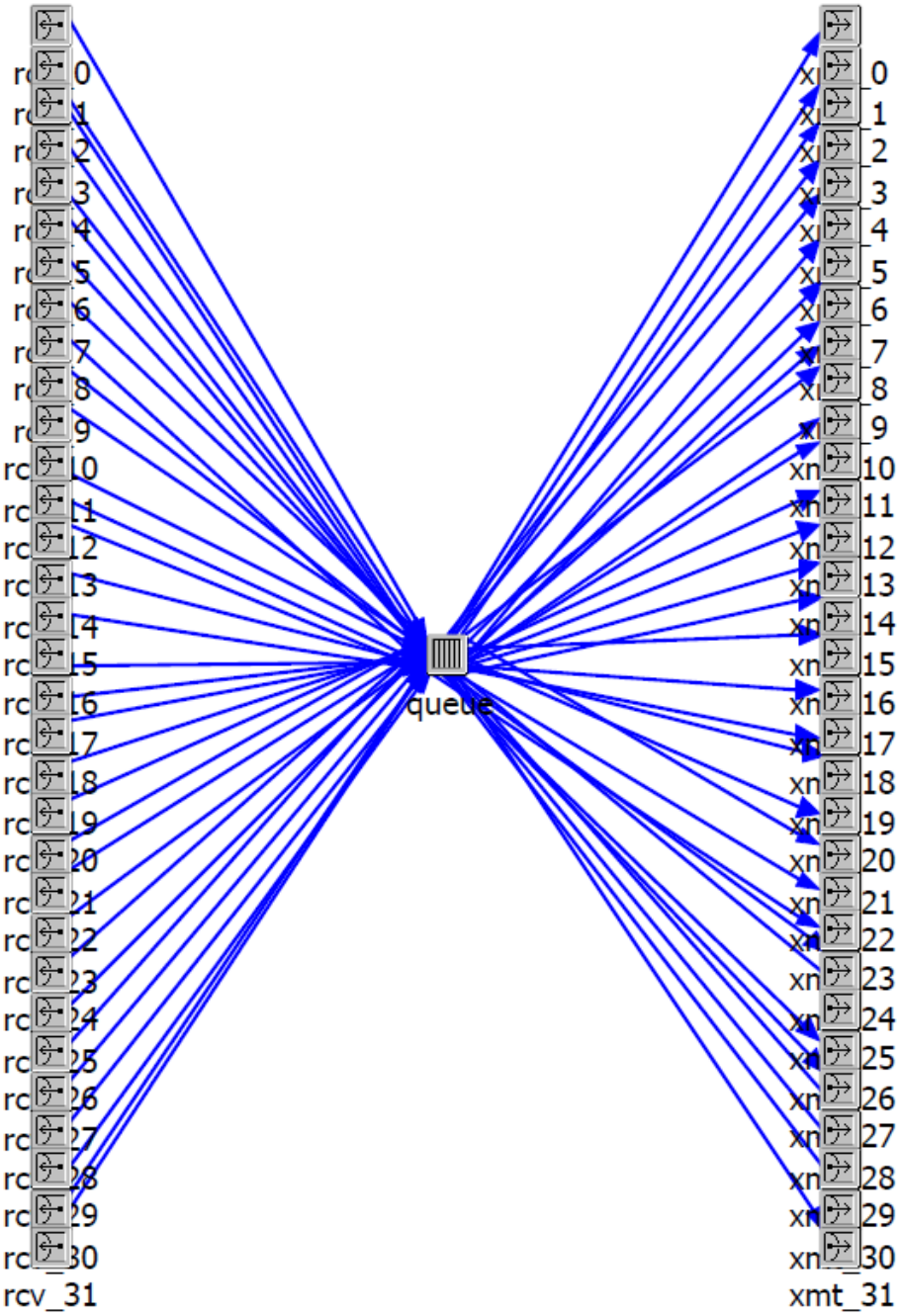

图 5-8 单进程单队列交换机节点的模块连接图

Fig.5-8 Module interconnection of single process and single queue switch node

图 5-8 中左边是标号为 rcv0~31 的接收器，右边是标号为 xmt0～31 的发射器，最重要的是中间的 queue 模块，在单进程单队列交换机节点中，queue 模块只有一个队列和一个进程，所有的输出端口都共享这一个队列，所有的数据包转发工作都由这一个进程完成。queue 模块的进程模型的有限状态机如图 5-9 所示。

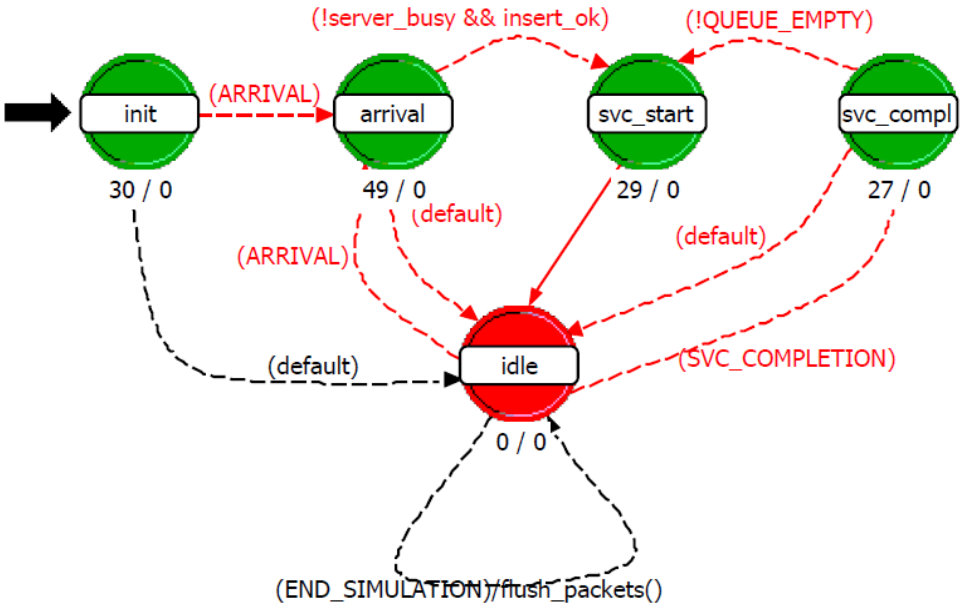

图 5-9 单进程单队列交换机节点 queue 模块的进程模型有限状态机

Fig.5-9 FSM of process model within queue module of single process and single queue switch node

进程经过 init 状态之中的初始化后进入空闲状态，如果在 init 状态结束时刚好有帧到达，或者在空闲状态阻塞时有帧到达，那么就进入 arrival 状态，执行 arrival 状态的入口执行代码。执行完后，如果此时进程不处于服务状态并且成功地将数据包压入队列，就进入 svc_start 状态开始服务队列中的下一个包，不管什么情况进程最后都重新回到空闲状态。当一个帧服务结束时，进程进入 svc_compl 状态，将刚服务完的数据包从队列中删除，并检查队列，如果非空，则弹出下一个数据帧并开始服务，否则回到空闲状态。

arrival 状态是最重要的状态之一，它的入口执行代码如图 5-10 所示。

```
/* acquire the arriving packet                    */
/* multiple arriving streams are supported. */
pkptr = op_pk_get (op_intrpt_strm ());
op_pk_nfd_get_int32(pkptr, "dest_address", &destination_address);
cellIndex = getFreeIndex();

if (cellIndex) {
    int queueSize = op_q_stat(OPC_QSTAT_PKSIZE);
    if ((*cellIndex) < queueSize) {
        // Defer delay, now being executed
        op_subq_pk_remove(0, *cellIndex);
    }
    if (op_subq_pk_insert(0, pkptr, *cellIndex) != OPC_QINS_OK) {
        insert_ok = 0;
    } else {
        insert_ok = 1;
    }
} else {
    insert_ok = 0;
}

if (!insert_ok) {
    /* the inserton failed (due to to a          */
    /* full queue) deallocate the packet.        */
    op_pk_destroy (pkptr);

    /* set flag indicating insertion fail        */
    /* this flag is used to determine            */
    /* transition out of this state              */

    insert_ok = 0;
} else {
    double t;

    /* insertion was successful                  */
    insert_ok = 1;
    op_prg_list_insert(port_queue[destination_address], cellIndex, OPC_LISTPOS_TAIL);
    t = op_sim_time();

    op_pk_nfd_set_dbl(pkptr, "time_enter_switch", t);
    op_stat_write(duration_before_switch, t - op_pk_creation_time_get(pkptr));
}
```

图 5-10 arrival 状态的入口执行代码

Fig.5-10 Code of arrival state enter executives

这段代码每一行的作用此处不再赘述，最重要的就是要为每一个输出端口维护一个逻辑队列，这些逻辑队列共享一个物理队列。除了数据包的查找、插入和删除操作之外，还必须设置包进入交换机的时间以及完成其它一些统计量的操作。

5.5.2 多进程多队列交换机节点

多进程多队列交换机节点建模了 SPSMS 交换机的行为，其模块结构图如图 5-11 所示。

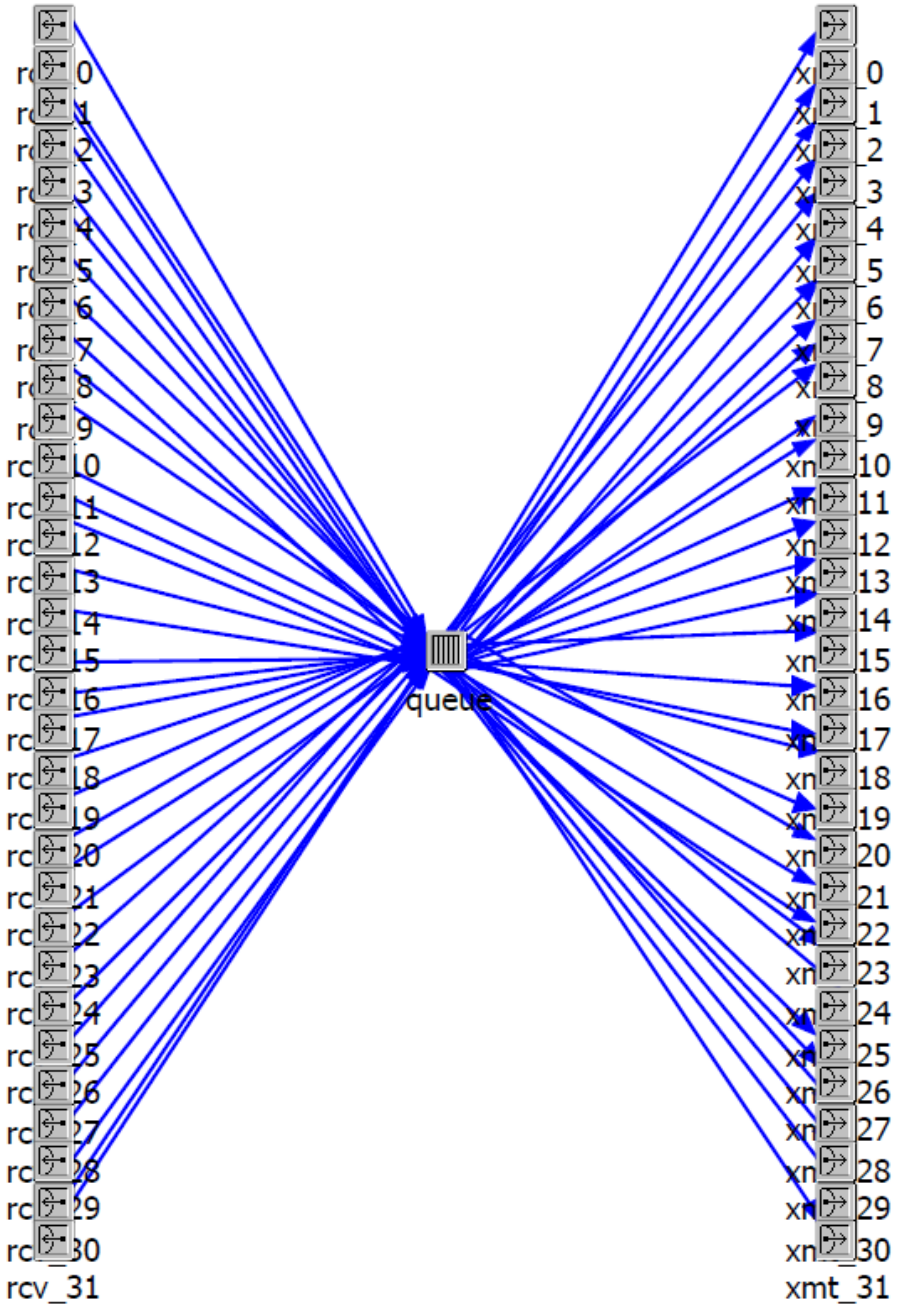

图 5-11 多进程多队列交换机节点的模块连接图

Fig.5-11 Module interconnection of multi-process and multi-queue switch node

与图 5-8 所示的模块连接图一样，二者在拓扑结构图上并没有什么区别，但是区别在于 queue 模块的不同，表现在两个方面。

（1）queue 模块的进程模型

单进程单队列交换机节点中，queue 模块的进程模型由图 5-9 所示的 FSM 描述。但是在多进程多队列交换机节点中，queue 模块的进程模型并不完成例如将

帧压入队列、从队列中取帧以及转发帧等工作，而只完成简单的分派工作。

（2）queue 模块的队列个数

单进程单队列交换机节点中，queue 模块只有一个物理队列。但是在多进程多队列交换机节点中，queue 模块有 32 个物理队列，每一个物理队列与相应输出端口相关联，以存储去往该输出端口的数据帧，各物理队列之间相互独立没有交集。

除了上述两点不同之外，多进程多队列交换机节点更重要的是利用 OPNET Modeler 的动态进程来模拟 DMA 的功能，逻辑上一个动态进程管理一对输入端口和输出端口。所以多进程多队列交换机节点需要 32+1 个进程，我们称之为 1 个主进程和 32 个子进程。主进程只做简单的分派任务，其它 32 个子进程完成具体的存储与转发工作。

5.5.3 主进程模型

多进程多队列交换机节点的 queue 模块的进程模型如图 5-12 所示。

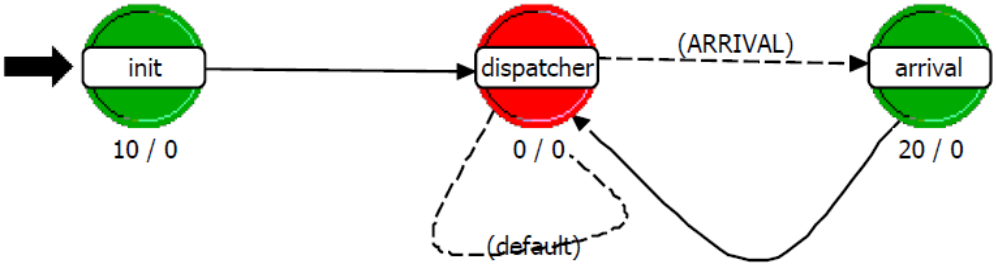

图 5-12 多进程多队列交换机节点 queue 模块的主进程模型有限状态机

Fig.5-12 FSM of master process model within queue module of multi-process and multi-queue switch node

在图 5-12 中，最值得阐述的是 init 状态的入口执行代码，如图 5-13 所示。

```
int i;
op_ima_sim_attr_get_int32("number of output ports", &NUMBER_OF_OUTPUT_PORTS);
for (i = 0; i < NUMBER_OF_OUTPUT_PORTS; i++) {
    subProcesses[i] = op_pro_create("rtio_subprocess", OPC_NIL);
}

duration_before_switch = op_stat_reg(
    "duration before switch",
    OPC_STAT_INDEX_NONE,
    OPC_STAT_GLOBAL);
```

图 5-13 init 状态的入口执行代码

Fig. 5-13 Code of init state enter executives

图 5-13 中的代码完成一个很重要的初始化工作，即创建 32 个子进程，并指定这些子进程的进程模型，这个进程模型的名字为 rtio_subprocess。主进程完成子进程的初始化工作之后，进入 dispatcher 状态，此时一旦有帧到达事件 ARRIVAL 发生，就转移到 arrival 状态，在 arrival 状态里面主进程根据帧的目的端口进行调度任务的分发，然后返回到 dispatcher 状态。

5.5.4 子进程模型

在 5.5.3 节中，通过 init 的入口执行代码，创建了 32 个子进程，同时指定了它们的进程模型，其有限状态机如图 5-14 所示。

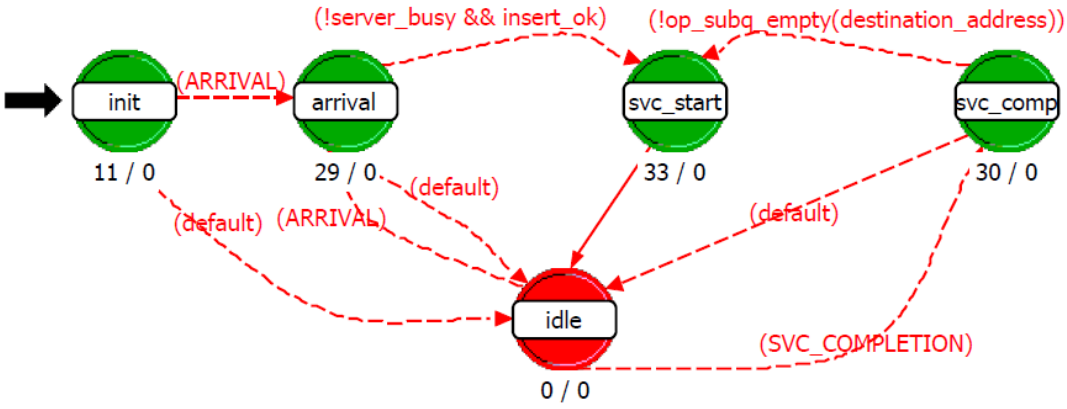

图 5-14 多进程多队列交换机节点 queue 模块的子进程模型有限状态机

Fig. 5-14 FSM of sub process model within queue module of multi-process and multi-queue switch node

可以看出图 5-9 与图 5-14 很类似，因为子进程完成的任务与原来单一进程完成的任务很类似，但是有区别，表现在两方面。

（1）当一个包进入时，子进程只需简单地将其插入到它自己的队列当中，然后检查当前子进程的服务状态，并依据检查结果转移到下一个状态。

（2）当一个包服务结束时，子进程检查的是它自己的队列，而不是整个队列亦或别的子进程的队列。

5.6 交换机结构图

5.1～5.5 节阐述了为完成一个完整交换机设计需要的所有部件，本节将利用这些部件构造一个可以仿真运行的交换机。如图 5-15 所示是传统的共享内存交

换机网络结构图。

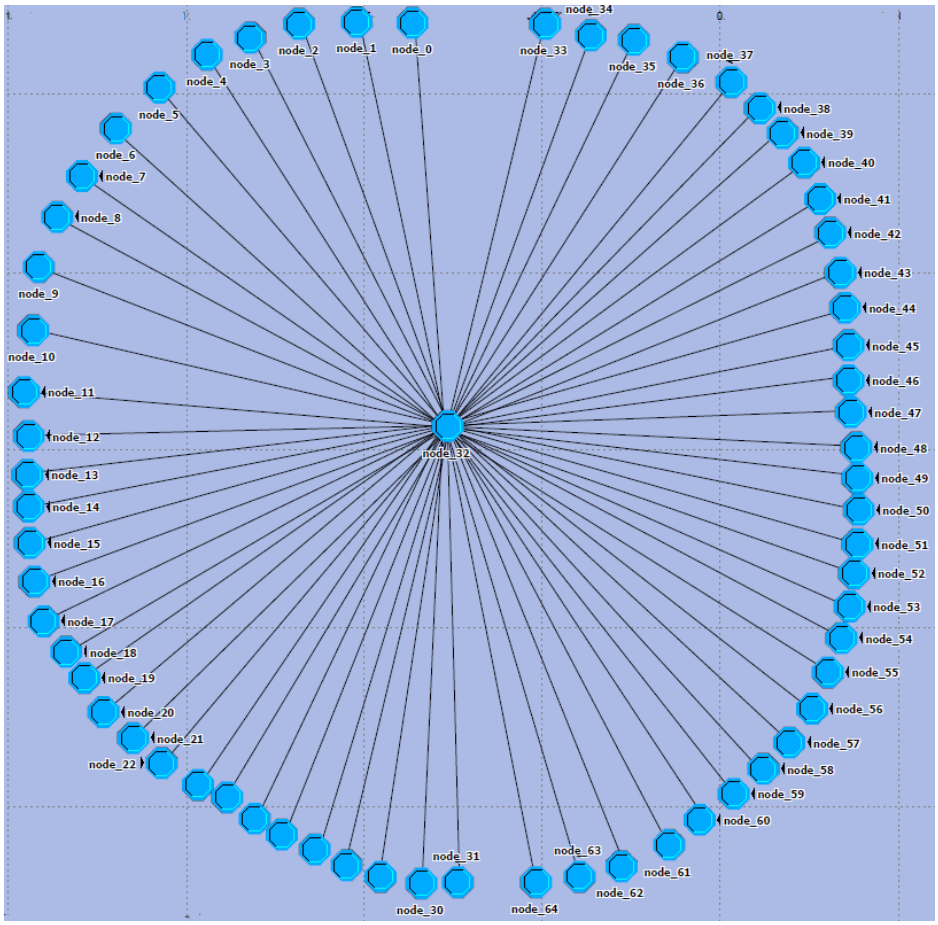

图 5-15 传统共享内存交换机的网络结构图

Fig.5-15 Network topology of shared memory switch

图 5-15 中，有 32 个数据生成节点和 32 个数据接收节点，中间的节点是单进程单队列交换机节点。数据生成节点与交换机节点之间以及数据接收节点与交换机节点之间均通过全双工链路相连接。

SPSMS 交换机结构图如图 5-16 所示。

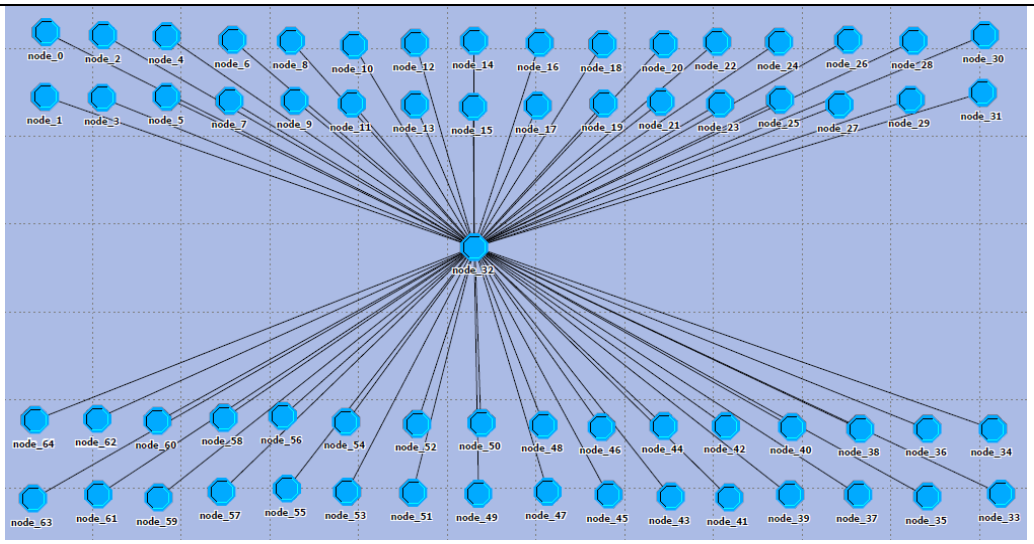

图 5-16 SPSMS 交换机的网络结构图

Fig.5-16 Network topology of SPSMS switch

图 5-16 的拓扑结构与图 5-15 的拓扑结构是一样的，只是布局不一样，这基本不会引入误差。地理上的距离对结果只产生微弱的影响。虽然拓扑结构一样，但是中间的交换机节点是不同的，图 5-16 中中间节点是多进程多队列交换机节点。为了清楚地看出 SPSMS 交换机与传统共享内存交换机架构上的不同，此处针对 SPSMS 交换机的整体层次架构作一个示意图。如图 5-17 所示，可以明显看出每个子进程负责一个输出端口的交换工作。

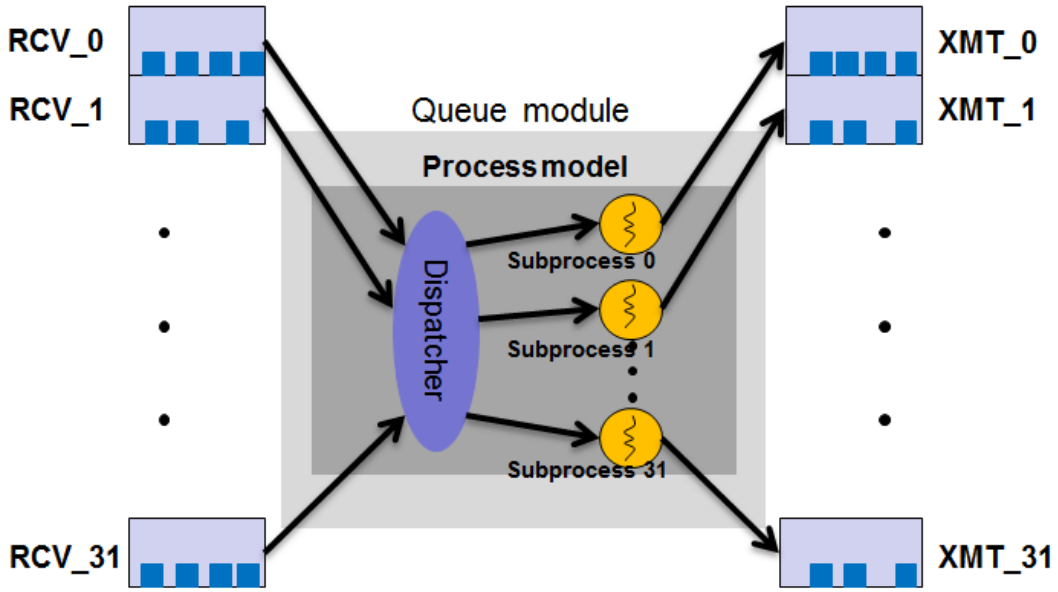

图 5-17 SPSMS 交换机的全局架构图

Fig.5-17 Overall architecture of SPSMS switch

5.7 本章小结

本章系统完整地介绍了使用 OPNET Modeler 建模交换机的整个流程，囊括了交换机的所有细节，小到数据帧格式和链路格式，大到数据生成节点、数据接收节点和交换机节点，最后构建了完整的交换机。

6 结果分析与讨论

第 5 章详细阐述了使用 OPNET Modeler 建模的传统共享内存交换机和 SPSMS 交换机，从最底层的数据帧格式开始，进而到链路格式，然后到数据生成节点、数据接收节点和交换机节点，最后构建一个完整的交换机网络模型。本章将对通过正确性验证的交换机网络模型进行仿真和收集数据，最后进行分析。

表 6-1 是试验中统计的一些参数的解释及计算方法。

表 6-1 统计参数表

Fig. 6-1 Statistics variables

名称	解释	计算方法
ETE Delay	包的端到端时延	当包到达接收端时，用当时的时刻减去生成包的时刻
queuing delay	排队时延	这是 OPNET Modeler 队列模块内建统计量
queue size	队列大小	这是 OPNET Modeler 队列模块内建统计量

6.1 端到端时延的分析

端到端时延表征着交换机调度器的性能，即一个包从进入交换机到离开交换机的时延。图 6-1 是端到端时延随时间的平均值的比较。从图中可看出，传统的共享内存交换机（Shared Memory Switch，简称 SMS）和 SPSMS 交换机的端到端时延平均值均趋向于一个稳定值，但是 SPSMS 交换机的端到端时延值要低。

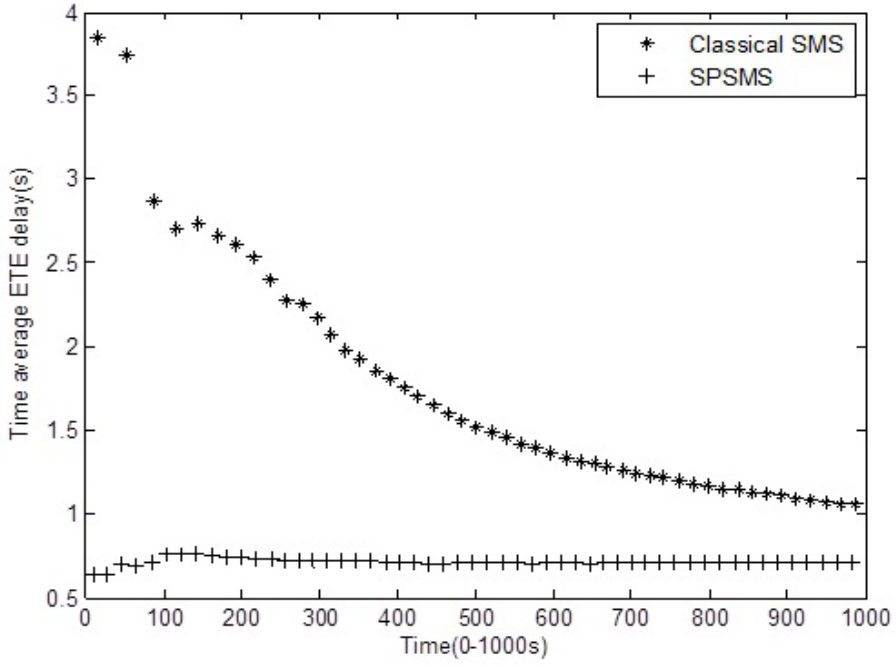

图 6-1 传统共享内存交换机和 SPSMS 交换机的端到端时延时间平均值比较

Fig.6-1 Comparison of time average ETE delay between classical SMS and SPSMS

6.2 排队时延的分析

排队时延主要衡量的是共享内存性能的好坏，值越大说明数据帧滞留的时间越长，性能越差。图 6-2 是在发送同等数量和接收同等数量的帧的前提下，SMS 和 SPSMS 的排队时延的比较。可以明显看出，SPSMS 的排队时延相比 SMS 来说，下降得非常明显。

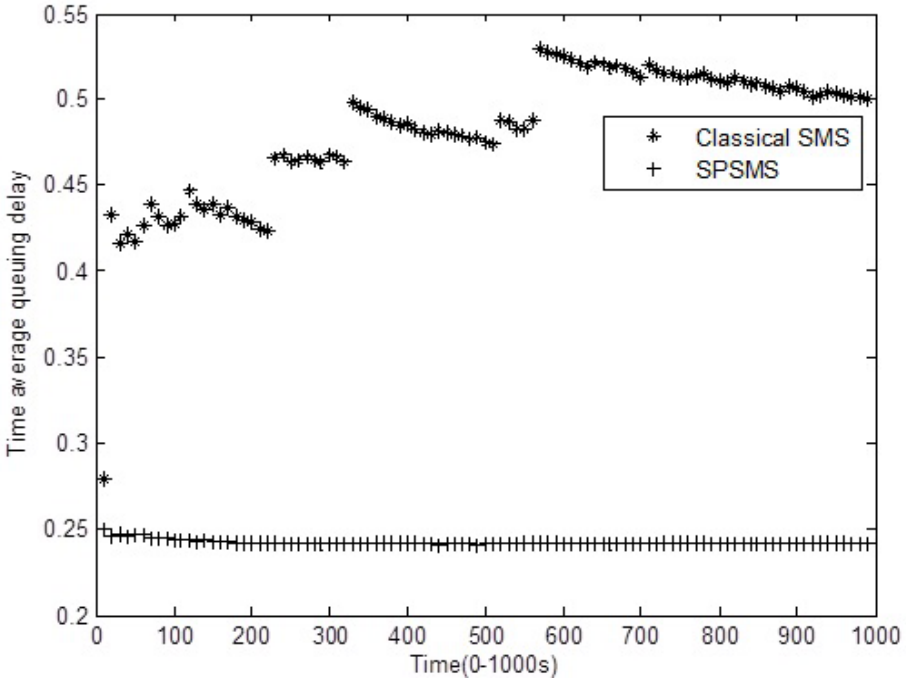

图 6-2 传统共享内存交换机和 SPSMS 交换机的排队时延时间平均值比较

Fig.6-2 Comparison of time average queuing delay between SMS and SPSMS

6.3 队列大小的分析

与排队时延一样，队列大小也是用来衡量共享内存性能的好坏的。从图 6-3 可以看出，最后二者均能稳定在一个水平，但是 SPSMS 的队列大小明显地少于 SMS 的队列大小。

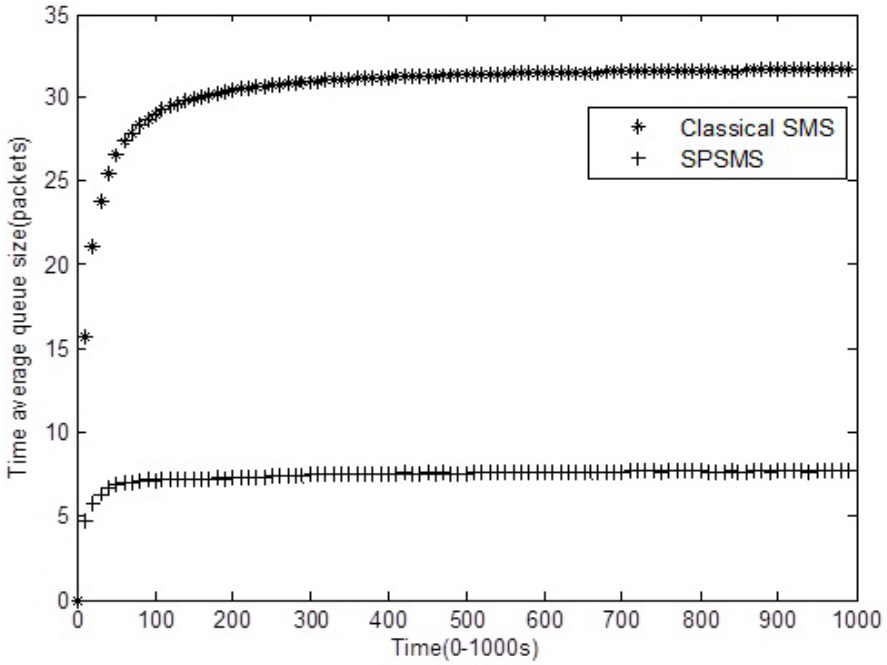

图 6-3 传统共享内存交换机和 SPSMS 交换机的排队大小时间平均值比较

Fig.6-3 Comparison of time average queue size between SMS and SPSMS

6.4 SPSMS 交换机的队列大小细则

图 6-3 中的 SPSMS 交换机的队列大小是指 32 个队列的和，为了看出每个队列变化的具体情况，需要分别对每个子队列大小进行统计。在本文的实现中，交换机有 32 个输出端口，所以也就有 32 个子队列。图 6-4 中展示了其中三个子队列的大小以及总和的变化曲线，总和曲线与图 6-3 中下方的曲线是一致的。

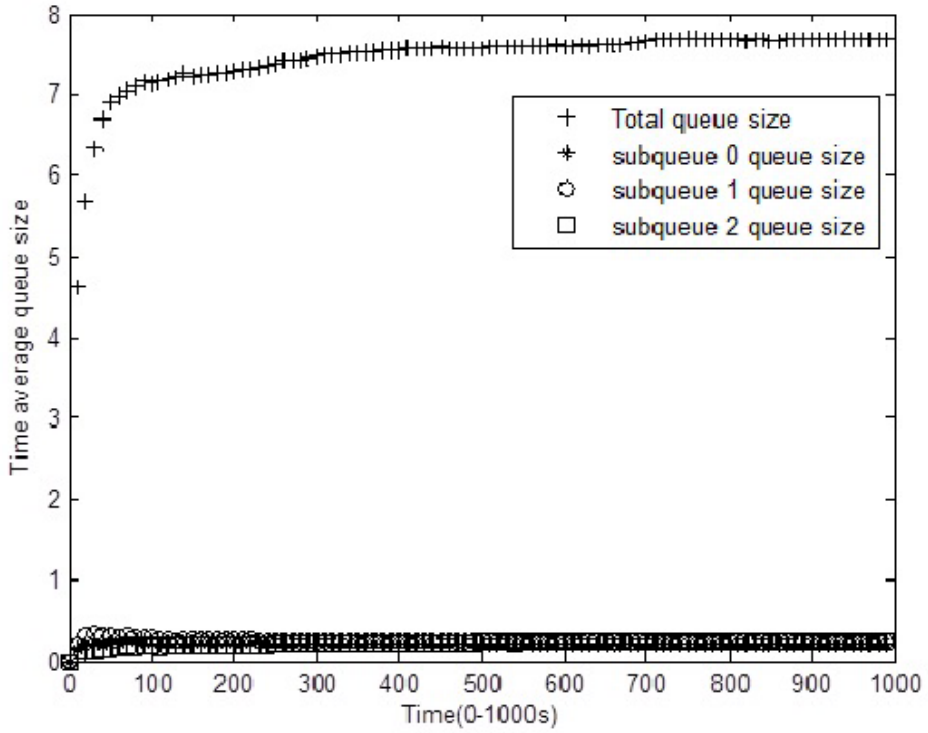

图 6-4 SPSMS 交换机队列大小细则

Fig.6-4 Details of queues of SPSMS switch

6.5 SPSMS 交换机端口数对性能的影响

本章前半部分阐述了 SPSMS 交换机与传统共享内存交换机性能的对比，从图中可以看出，SPSMS 交换机的性能相比传统共享内存交换机的性能来说，提升还是比较明显。本节将对 SPSMS 交换机做更多的仿真，以考察 SPSMS 交换机在端口数变化的情况下各种性能参数的变化情况。

（1）端口数对端到端时延的影响

端口数越多，代表对内存总的带宽需求更大，交换机调度器也需要处理更多的数据包，虽然相应的内存块数和子调度器的个数也会增加，但是端到端时延的变化并非线性增加或者减少。如图 6-5 所示，是不同端口数下端到端时延时间平均值的变化情况。

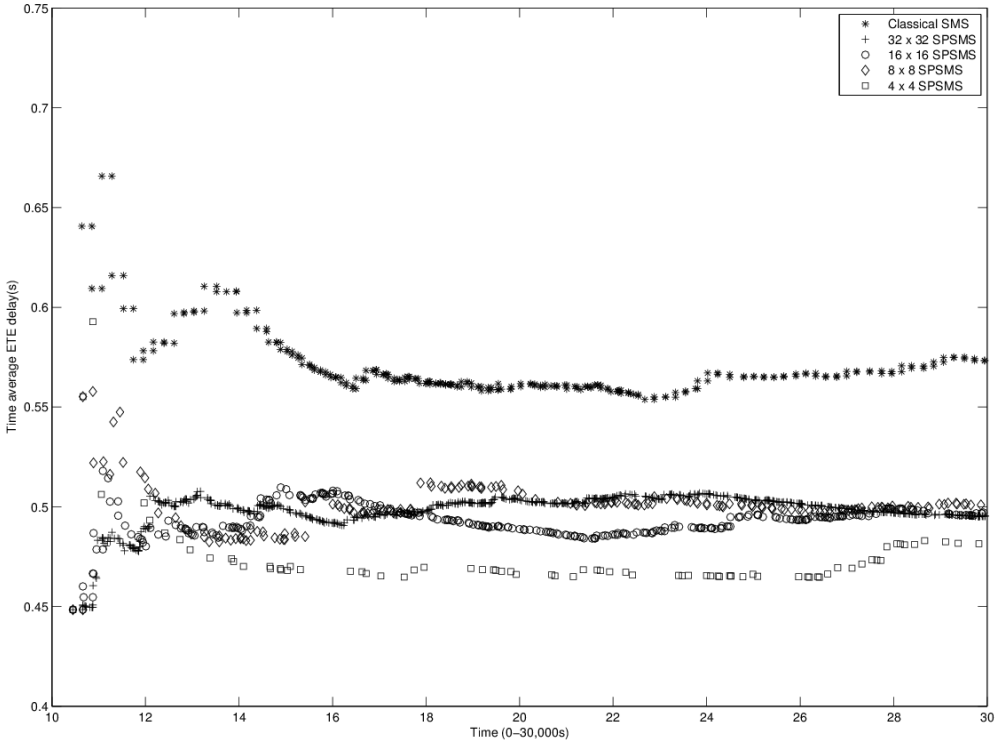

图 6-5 不同端口数下端到端时延时间平均值的变化情况

Fig.6-5 Curves of ETE delay with different number of ports

图 6-5 中，最上面的是传统共享内存交换机的变化曲线，下面分别是端口数为 32*32、 16 * 16、 8 * 8 和 4 *4 的 SPSMS 交换机的变化曲线。很意外地是，端到端时延并没有随端口数增加而减少，反而略微增加。因为端口数越多，主调度器的负担就越重，此时主调度器给 DMA 发送信号的时间足以影响到整体端到端时延的值，那么在计算时就不能忽略了。

（2）端口数对排队时延的影响

端口数增加导致需要缓存的数据包增多，但是存在与输出端口数一样的内存，并且每一个数据包只需要等待跟它有一样的目的输出端口的包，所以排队时延不会有非常明显的变化。从图 6-6 中也可以看出这一点。

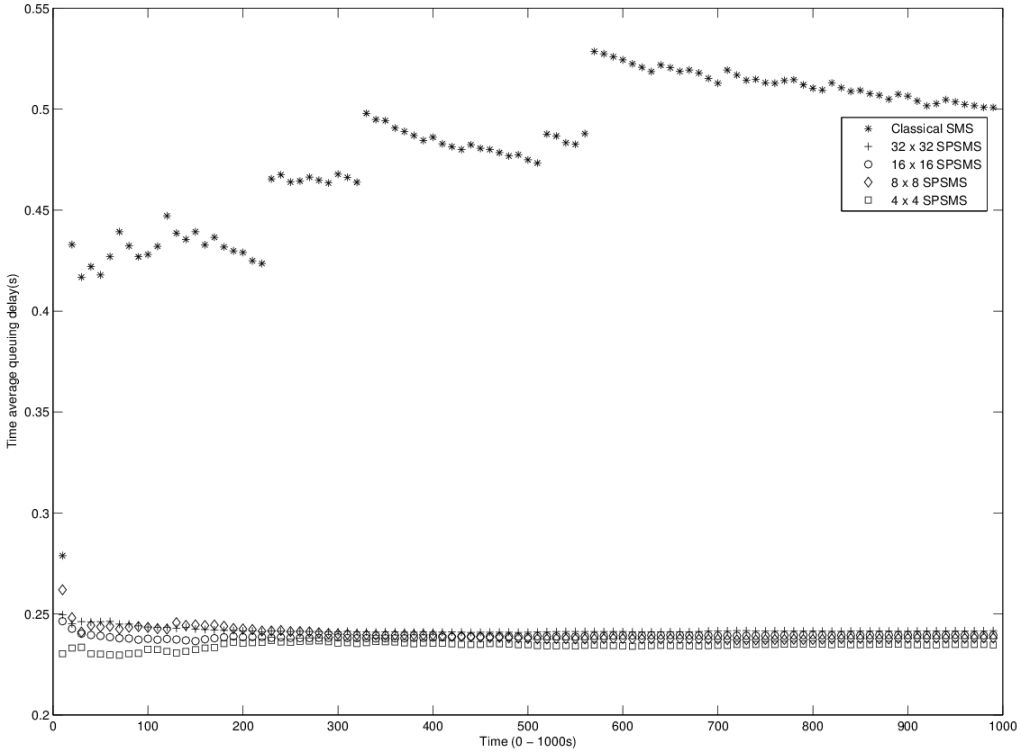

图 6-6 不同端口数下排队时延时间平均值的变化情况

Fig.6-6 Curves of queuing delay with different number of ports

（3）端口数对队列大小的影响

端口数增加导致需要缓存的数据包增多，虽然存在与输出端口数一样的内存，并且每一个数据包只需要等待跟它有一样的目的输出端口的包，然而数据包需要等待更多的数据包，从而队列大小就会相应增加。如图 6-7 所示。

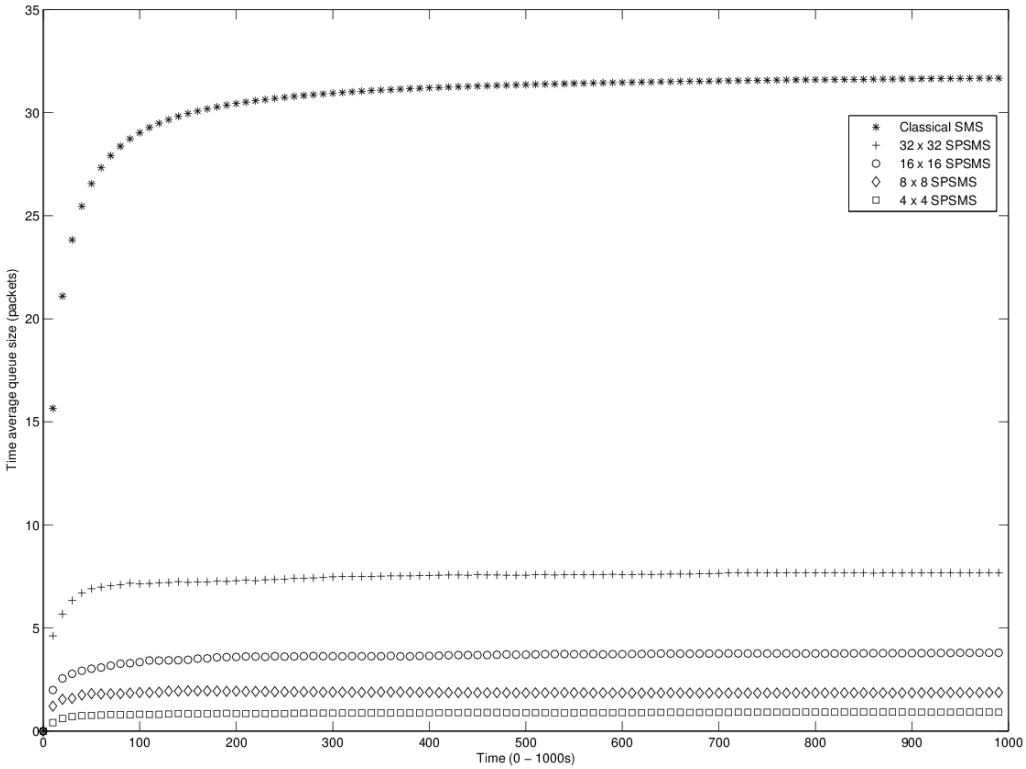

图 6-7 不同端口数下队列大小时间平均值的变化情况

Fig.6-7 Curves of queue size with different number of ports

6.6 本章小结

本章通过对交换机在不同参数下进行仿真运行，收集了大量数据并进行了各种分析，比较了 SPSMS 交换机和传统共享内存交换机的不同性能参数，发现 SPSMS 交换机的性能有显著提升。同时建模并仿真了不同端口数对 SPSMS 交换机性能的影响。

7 总结与展望

本文主要针对 AFDX 航空网络中的高速交换技术进行了研究，以共享内存架构为基础，通过空间分离和异步调度技术，充分利用 AFDX 网络的静态特性，提出一种新型的空分共享内存交换机——SPSMS 交换机，并通过 OPNET Modeler 进行网络模拟和仿真，验证本文提出的交换机架构性能上确实有所提升。

首先，对 AFDX 航空网络进行进行透彻的分析，总结了 AFDX 航空网络区别于其它类型的网络的特性。AFDX 航空网络最大对本文也最有价值的特性就是静态性。AFDX 网络借鉴了 ATM 网络中的虚拟链路概念，并进行了强化，使得每一条虚拟链路的带宽均不会超过限定值。同时，AFDX 网络的总带宽也是静态的，不会随时间而改变。除了带宽的静态性之外，AFDX 网络还具有路由静态性。即 AFDX 网络中的路由拓扑结构是确定的，所有虚拟链路的源终端和目的终端均事先指定，每一个 AFDX 交换机的路由表也是事先指定，并且不会在运行时改变。以上特性极大地方便了本文的设计与实验。

其次，对共享内存交换机进行了充分地研究，总结了它的优缺点，并分析指出适合与不适合 AFDX 航空网络的属性。首次提出共享内存交换机的时间非常早，经过长达十多年的研究，共享内存架构已经非常成熟，但是其本身固有的缺点使得其应用范围不是非常宽广。共享内存架构最明显的缺点就是不易扩展，包括容量和带宽，并且由于共享一块内存和一个调度器，使得时延和抖动非常大。

再次，通过对 AFDX 网络和共享内存交换机的充分研究之后，AFDX 网络的静态特性使得共享内存交换机非常适合 AFDX，本文很好地将二者融合在一起。第一，通过空间分离技术，使得交换机的每一个端口都拥有只属于自己的内存，并且既能保证每一个输出端口不会出现包溢出的情况，又不会造成内存资源的浪费，同时大大减少了对于内存资源的竞争；第二，为每一个内存增加一个 DMA，从而减轻了主调度器的负担，主调度器只需要给相关 DMA 发送一个信号，然后即刻返回。这大大降低了调度时延、排队时延和队列大小。

最后，通过 OPNET Modeler 进行网络模拟和仿真，并收集数据进行分析，本文提出的 SPSMS 交换机的性能参数确实比传统共享内存交换机的性能参数都要好。由于主调度器的负担大大降低，并且解耦了所有的输出端口，所以端到端时延、排队时延和队列大小都大大减少，从最后的结果图来看这很好并成功地验

证了本文提出的交换机设计。

本文很好地针对 AFDX 网络中的高速交换技术进行了比较深入和有益的探索与研究，提出了一种新型的交换机架构。但是仍旧存在很多的工作要做。

第一，在结果分析一章中，已经看到当端口数增加时，主调度器的负担会越来越大，甚至会影响到整个端到端时延的结果，所以必须在现有的 SPSMS 交换机上提出进一步的改进。目前的想法是消除主调度器的存在，亦即每一个输入端口都有一个控制器或者处理器来处理到达的数据包，每一个输出端口也都有一个控制器或者处理器来处理到达的数据包。可以尝试将 DMA 替换成 GPU 来达到上述目的。

第二，本文采用的是网络模拟和仿真。然而真正能说明 SPSMS 交换机具有价值的是用 FPGA 或者硬件来实现这个交换机，并构造真实的 AFDX 网络，再进行实际的运行、收集数据和分析，最后通过结果来说明 SPSMS 交换机的优势和劣势。

第三，在现有 OPNET Modeler 建模的模型基础上，进行功能添加和性能优化等工作，使得模型更加符合 AFDX 航空网络的需求。同时进行更多的统计，构建完整的交换机性能参数模型。

由于时间的关系以及本文作者理论水平有限，论文难免会存在不足与疏漏之处，恳请读者给予批评和指正。

参考文献

[1] Aircraft Data Network Part 7: Deterministic Networks, ARINC664 Std., 2003.

[2] AFDX/ARINC 664 Protocol Tutorial, Std., G. F. Embedded, 2007.

[3] ARINC specification 429, Aeronautical Radio Inc. Std., 2001.

[4] 陈昕，周拥军，万剑雄，AFDX 端系统关键技术的研究与实现，计算机工程，2009，35(5).

[5] 王伟鹏，张延园，姚进华，基于 FPGA 的 ARINC664/AFDX 端系统设计，微处理机，第 2 期，2009 年 4 月.

[6] 王辉，陈卓，刘宁，AFDX 网络终端软件测试策略的研究与应用，航空电子技术，2006.

[7] 李浩峰，AFDX 网络测试研究现状分析，计算机应用，2009.

[8] 罗杰，霍曼，AFDX 通信链路技术及其在航空电子系统的应用，全国第十届信号与信息处理和第四届 DSP 应用技术联合学术会议论文集，北京，2006，p16-21

[9] Hanxleden, R. V., Gambardella, AFDX Redundancy Management, 2001.2.

[10] Jana Tubrichvon, Hanxleden Reinhard, Formal Specification and Analysis of AFDX Redundancy Management, SpringerLink Date, 2007

[11] Yazhou Ren, Fei Hu, Jian Li, End-to-End Jitter Control on AFDX Network, International Conference on Transportation and Mechanical & Electrical Engineering, 2011

[12] J.Y.L. Boudec and P. Thiran, Network Calculus: A Theory of Deterministic Queuing System for the Internet, Springer-Verlag, 2001

[13] H. Charara and J.L. Scharbarg and J. Ermont and C. Frabou, Methods for bounding end-to-end delays on an afdx network, ECRTS, 2006, vol18, 102-202.

[14] Xiaoqiang Ji, Huanzhong Li, et al., Analysis of Deterministic End to End Delay in multi-hop AFDX Avonics Network System, PECCS, 2011, 434-440

[15] 胡光宇，李健，胡飞，姚建国，刘学，朱谷川，AFDX 网络端到端延迟的优化估计，中国科技论文在线

[16] P. Newman, “Atm technology for corporate networks,” Communications Magazine, IEEE, vol. 30, no. 4, pp. 90 –101, april 1992.

[17] R. Y. Awdeh and H. T. Mouftah, “Survey of atm switch architectures,” Comput.

Netw. ISDN Syst., vol. 27, no. 12, pp. 1567–1613, Nov. 1995. [Online]. Available: http://dx.doi.org/10.1016/0169-7552(94)00081-4
[18] J. Garcia-Haro and A. Jajszczyk, “Atm shared-memory switching architectures,” Network, IEEE, vol. 8, no. 4, pp. 18 –26, july-aug. 1994.
[19] Y. Oie, T. Sueda, M. Murata, D. Kolson, and H. Miyahara, “Survey of switching techniques in high-speed networks and their performance,” in INFOCOM ’90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. ’The Multiple Facets of Integration’. Proceedings., IEEE, jun 1990, pp. 1242 –1251 vol.3.
[20] H. Ahmadi and W. Denzel, “A survey of modern high-performance switching techniques,” Selected Areas in Communications, IEEE Journal on, vol. 7, no. 7, pp. 1091 –1103, sep 1989.
[21] T. Banniza, G. Eilenberger, B. Pauwels, and Y. Therasse, “Design and technology aspects of vlsis for atm switches,” Selected Areas in Communications, IEEE Journal on, vol. 9, no. 8, pp. 1255 –1264, oct 1991.
[22] A. Pattavina, “Nonblocking architectures for atm switching,” Communications Magazine, IEEE, vol. 31, no. 2, pp. 38 –48, feb. 1993.
[23] M. Karol, M. Hluchyj, and S. Morgan, “Input versus output queueing on a space-division packet switch,” Communications, IEEE Transactions on, vol. 35, no. 12, pp. 1347 – 1356, dec 1987.
[24] H. Kuwahara, N. Endo, M. Ogino, T. Kozaki, Y. Sakurai, and S. Gohara, “A shared buffer memory switch for an atm exchange,” in Communications, 1989. ICC ’89, BOSTONICC/89. Conference record. ’World Prosperity Through Communications’, IEEE International Conference on, jun 1989, pp. 118 –122 vol.1.
[25] T. Kozaki, Y. Sakurai, O. Matsubara, M. Mizukami, M. Uchida, Y. Sato, and K. Asano, “32 * 32 shared buffer type atm switch vlsis for bisdn,” in Communications, 1991. ICC ’91, Conference Record. IEEE International Conference on, jun 1991, pp. 711 –715 vol.2.
[26] N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara, and S. Gohara, “Shared buffer memory switch for an atm exchange,” Communications, IEEE Transactions on, vol. 41, no. 1, pp. 237 –245, jan 1993.
[27] P. Barri and J. Goubert, “Implementation of a 16 to 16 switching element for atm exchanges,” in Global Telecommunications Conference, 1990, and Exhibition. ’Communications: Connecting the Future’, GLOBECOM ’90., IEEE, dec

1990, pp. 1615 –1627 vol.3.

[28] Y. Shobatake, M. Motoyama, E. Shobatake, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue, “A one-chip scalable 8*8 atm switch lsi employing shared buffer architecture,” Selected Areas in Communications, IEEE Journal on, vol. 9, no. 8, pp. 1248 –1254, oct 1991.

[29] S. Liew and K. Lu, “Comparison of buffering strategies for asymmetric packet switch modules,” Selected Areas in Communications, IEEE Journal on, vol. 9, no. 3, pp. 428 –438, apr 1991.

[30] W. Fischer, O. Fundneider, E.-H. Goeldner, and K. Lutz, “A scalable atm switching system architecture,” Selected Areas in Communications, IEEE Journal on, vol. 9, no. 8, pp. 1299 –1307, oct 1991.

[31] M. De Prycker and M. De Somer, “Performance of a service independent switching network with distributed control,” Selected Areas in Communications, IEEE Journal on, vol. 5, no. 8, pp. 1293 – 1301, oct 1987.

[32] J. Causey and H. Kim, “Comparison of buffer allocation schemes in atm switches: complete sharing, partial sharing, and dedicated allocation,” in Communications, 1994. ICC ’94, SUPERCOMM/ICC ’94, Conference Record, ’Serving Humanity Through Communications.’ IEEE International Conference on, may 1994, pp. 1164 –1168 vol.2.

[33] A. Choudhury and E. Hahne, “Buffer management in a hierarchical shared memory switch,” in INFOCOM ’94. Networking for Global Communications., 13th Proceedings IEEE, jun 1994, pp. 1410 –1419 vol.3.

[34] Gross, Donald, Carl M. Harris. Fundamentals of Queueing Theory, 1998

[35] 周强，熊华钢，张晓林，张彦仲，AFDX 交换机在强实时条件下的分组调度，北京航空航天大学学报，2009 年第 4 期

[36] 张勇涛，黄臻，熊华钢，保证速率的 AFDX 交换机实时调度算法，北京航空航天大学学报，2010 年第 12 期

致 谢

值此毕业之际，回想自己的研究生生涯，感慨良多。除了感叹“逝者如斯，不舍昼夜”之外，更多的是满怀感恩之情。从当初本科毕业设计到研一期间的学习到研二研三期间的研究进而到工作的寻找最后到毕业论文的写作，这一路上的艰辛与拼搏、失落与成功，都让我受益匪浅。能取得不错的成绩和完成论文的写作，需要感谢的人很多。

首先，我要感谢我的导师胡飞老师和李健老师。胡飞老师和李健老师忘我的工作精神，一丝不苟的治学态度，博大精深的专业知识，都深深地震撼到了我，能作为他们的学生，我感觉非常地自豪。在学习上，两位老师素来严格要求我们，基础知识必须扎实牢固，要做到举一反三，要时刻反省自己的行为；在研究上，两位老师循循善诱，耐心开导我们，不是授之以鱼，而是授之以渔；在论文阅读与写作上，两位导师也是别具匠心，以每周例会的形式向我们传授阅读论文的方法，并且安排每个同学都必须上去做演讲，这不仅锻炼了我们的演讲和表达能力，同时也最大化地暴露出我们的缺点，并能得到两位老师精辟的指导；在生活上，两位老师也是关怀备致，体贴入微。在明白我家境困难的时候，他们为我全力争取每一个当助管和助教的机会，不仅提高了我的沟通和交际能力，也极大地巩固了我的专业知识。尤其是感谢李健老师，在我小论文的写作过程中给予了莫大的帮助，每每当我陷入一些局部章节和琐碎段落的描述时，他总能给我全局上的提示，让我有种醍醐灌顶的感觉，思路顿时豁然开朗。在此，我向胡老师和李老师致以最崇高的敬意和最真诚的感谢。

其次，我要感谢已经出站的周海瑞博士后，他在网络研究领域丰富的经验和深刻的见解都让我非常敬佩，同时他也教会了我们许多为人处事和做学问的方法。他以他自己的亲身经历教导我们，避免我们重蹈覆辙，少走了许多弯路。在此，向他表示最衷心的感谢。还有已经毕业了的学长，此处就不枚举他们的姓名了，也向他们表示最诚挚的谢意。

再次，我要感谢同届的实验室同学王威震、李晓波、鲁晓圆、王强和刘文化，能和你们成为一个实验室的同学真的非常荣幸。在研一期间，我们一起完成软件测试课的大作业，一起承担老师们布置的任务，互帮互助，你们在我失落时的鼓励和犯错时的包容，都是我研究生生活中最浓墨重彩的一笔。感谢他们。

最后，我要感谢我的女朋友欧阳玉媛，她博爱的胸怀让我每次受挫时都能很快找回应有的理智与冷静。她跟我一起分担忧虑与担心，跟我一起分享成功与喜悦，是她让我充满无限拼搏的激情与奋斗的动力。感谢缘分让我们走到一起。我还要感谢我的家人，他们为我倾其所有，一直给我温暖的呵护和关心，不管任何艰难困苦，都给我一个温暖的家。他们健康的身体和开心的笑容是我最大的心愿。

十多年的求学之路即将画上圆满的句号，再一次向所有关心、支持和帮助我的人表示最诚挚的感谢。

攻读学位期间发表的学术论文目录

[1] 胡靖飞，李健，胡飞， 基于共享内存的 AFDX 航空网络交换机设计[OL]，中国科技论文在线，已发表