申请上海交通大学工程硕士学位论文

AFDX 网络的容错控制方法研究

学校代码：10248

作者姓名：任亚周

学　　号：1090379071

第一导师：胡飞

第二导师：李健

学科专业：软件工程

答辩日期：2012 年 1 月 5 日

上海交通大学软件学院

2012 年 1 月

A Dissertation Submitted to Shanghai Jiao Tong University

for Master Degree of Engineering

THE RESEARCH OF FAULT-TOLERANCE MECHANISM ON AFDX

University Code：	10248
Author：	Yazhou Ren
Student ID:	1090379071
Mentor 1：	Fei Hu
Mentor 2:	Jian Li
Field：	Software Engineering
Date of Oral Defense：	Jan 5, 2011

School of Software

Shanghai Jiao Tong University

Jan 5, 2011

AFDX 网络的容错控制方法研究

摘 要

随着高性能飞机的快速发展和空域环境的日趋复杂，飞机对航空电子系统的需求越来越多，依赖性越来越强，传统航空电子系统数据通信总线 ARINC429 和 MIL-STD-1553 等协议已经无法满足新一代航空系统对通讯系统的要求。因此，航空电子全双工分组交换以太网(Avionics Full Duplex Switched Ethernet)(简称 AFDX)应运而生。

为了满足航空系统对可靠性的苛刻要求，AFDX 通信链路采用物理上冗余的技术，一个 AFDX 系统中有 A 和 B 两个独立的交换网络。正常情况下，AFDX 终端系统同时在 A 和 B 两个网络上进行数据传输。因此，每个终端系统将会收到两份同样的数据包，所以需要在接收端冗余管理模块对数据包进行过滤，把两个数据流整合成一条数据流。但是，这个看似简单的工作，并不简单。冗余管理模块需需要保证以下情况时，网络能够接受正确的数据包，例如随机的不正常的帧、单个帧丢失、端系统重启、switch babbling 等情况。本文的主要内容是对 AFDX 网络和冗余管理机制进行研究，从而提高网络可靠性。

本课题主要研究内容和成果如下：

(1) 通过引入 jitter-EDD(Earliest-Due-Date)机制，控制端到端的延迟和延迟抖动，解决有效数据包丢失的问题，从而提高数据的安全性。

(2) 为了在 AFDX 中引入 jitter-EDD 机制，本文对端系统和交换机的架构进行了重新设计，满足了 jitter-EDD 机制的应用要求。原有的 jitter-EDD 机制并没有给出具体的端到端的 deadline 分配机制，因此本文对 deadline 分配机制也进行了研究，设计了基于权重的 deadline 分配机制。

(3) 现有的 AFDX 航空数据总线系统在可靠性和安全性方面，尚不能满足对于关键性飞行控制系统的苛刻要求。因此，研究网络节点上任务的可调度性就显得很必要，本文给出了多条虚拟链路可调度性的充分必要条件。

(4) 在研究端到端延时抖动的基础上，设计了新的冗余管理算法，对网络数据包及其冗余数据包进行安全可靠的冗余管理，确保网络可靠的工作。

(5) 使用 SimEvents 作为仿真工具，根据 A380 内部网络拓扑结构，搭建仿真平台。使用错误注入的方法，来验证新冗余管理机制对于各种网络错误的容错效果。经仿真实验证明，该冗余算法有效的提高了网络安全性。

关键词 AFDX，jitter-EDD，SimEvents，端到端延时，延时抖动控制，可调度性分析

THE RESEARCH OF FAULT-TOLERANCE MECHANISM ON AFDX

ABSTRACT

With the rapid development of high-performance aircraft and the increasing complexity of air environment, the requirement about avionics electronic systems is more and more, and the dependency about the avionics electronic systems becomes more and more strong. The traditional avionics data communication bus, such as ARINC429 and MIL-STD-1553, can't meet those requirements of the next generation of avionics communication system, so the Avionics Full Duplex Switched Ethernet (AFDX) has been designed.

In order to meet the critical reliable real-time communication demands of the avionic application, AFDX uses the network mechanism of redundant communication link, so there are two separate network, A and B in AFDX network. Normally, the AFDX end system will transmit data paralleling on the network A and network B. So, every end system will receive two pieces of same packets, and the end system need to filter the packets and merge the two data flow into one non-redundant stream in the redundancy management module. The redundancy management module must ensure that the network can accept the correct packets when there are some unexpected errors such as abnormal transmitted frame, loss of a frame, reset of end system, switch babbling, and so on. In this article, our main work is to do some research about the AFDX network and its redundancy management to increase the safety of the AFDX network.

The main content and result of the subject is as follows:

(1) The jitter-EDD mechanism has been added into the AFDX network to control the end-to-end delay jitter and solve the problem of the loss of valid packets and increase the data safety.

(2) The architecture of the AFDX end system and switch has been changed to meet the requirement of the jitter-EDD mechanism. In the original jitter-EDD scheme, there is no valid deadline distribute algorithm, so a method for deadline distribution which based on weight has been proposed.

(3) The existing AFDX network can't meet the demands for critical fly-control in reliability and safety Therefore, the study of the schedulability of the task set on the network nodes becomes very necessary. In this paper, a necessary and sufficient condition has been given out.

(4) A new redundancy management algorithm is designed to management the packet and its redundant packet to assure the reliability of the network.

(5) SimEvents is used to establish AFDX network simulation platform according to the topology of A380. And, we use fault injection methods to verify the redundancy management algorithm. It has shown that the new redundancy management algorithm was useful.

Keywords AFDX, jitter-EDD, SimEvents, end-to-end delay, jitter control, schedulability analysis

目录

1 绪 论

1.1 研究背景

随着高性能飞机的快速发展和空域环境的日趋复杂，飞机对航空电子系统的需求越来越多，依赖性越来越强。高新技术特别是信息技术的迅速发展及其在机载领域上的应用正推动着航空电子系统向综合化、模块化、智能化和实时化的方向发展。而实现这些发展目标的关键技术之一就是机载数据通信技术。因为大型飞机上有许多航空电子子系统，包括惯性平台、控制系统、传感器系统和通讯系统等。这些系统都需要高速、高可靠的信息传输。而飞控系统和燃油系统等尤其需要及时地从数据源向接收器发送不断更新的完整信息，可靠的实时通讯链路已经成为重要安全系统的核心部件[1]。

传统航空电子系统数据通信主要采用 ARINC429 和 MIL-STD-1553 等协议。ARINC429 规定数据传输速率为 100kbps，而 MIL-STD-1553 协议速率为 1Mbps。随着通讯数据量的增大，这些传统的飞机总线已经无法满足新一代航空系统对通讯系统的要求[1][2]。波音公司的解决方案是 ARINC629 数据总线，该协议采用是多通讯总线机制，能达到 2Mbps，120 个用户。然而，这个技术开发成本非常高，很难令人满意[3]。

在军事通信领域的驱动下，近年来商业计算机通讯工业在高速数据通信领域取得了飞速的发展，从 10M/100M 速率发展到 10G 以太网。基于商业计算机工业取得的巨大成功，将商业计算机通信模型应用于下一代航空电子系统已成为不可避免的趋势，因此，AFDX—航空电子全双工分组交换以太网（Avionics Full Duplex Switched Ethernet）应运而生[3]。空客 A380 率先使用了 AFDX，它创造性的以交换式网络取代了传统的分立式电缆连接或共享介质的总线，克服了后者的布线复杂，维护、改型困难的缺点[4]。

AFDX 是 ARINC664 协议第 7 部分定义的一个确定性网络，它具有带宽大、集成度高、实时性和可靠性好等特点，在航空电子领域有着巨大应用潜力，目前已用于空客 A380 和波音 787 中。AFDX 总线主要包含终端（End System），交换机（Switch），虚拟链路（Virtual Link）（简称 VL）。它是基于一种网络概念而不是通常所说的总线形式，在这个网络上有交换机和终端两种设备，终端之间的数据信息交换是通过 VL 进行的。VL 起到了从一个唯一的源端到一个或多个目的端逻辑上的单向链接的作用，且任意一个虚拟链路只能有一个源端。为了提高数据传输的可靠性，AFDX 基于通信链路物理上

冗余的交换网络原理，一个 AFDX 系统中有 A 和 B 两个独立的交换网络。AFDX 终端系统传输的每个数据包同时在 A 和 B 两个网络上进行发送。因此，正常情况下，每个终端系统将会收到两份同样的数据包。这样，即使某个网络内数据包传输失败或数据链路失效，AFDX 系统也可以提供可靠安全的数据传输。在 AFDX 中，接收端系统先进行完整性检查(Integrity Check)，然后再进行冗余管理(Redundancy Management)。完整性检查主要对序列号进行检查，查看收到的数据包是否合法。而冗余管理对冗余帧进行处理，对两条冗余链路上接收到的数据包进行过滤，消除冗余帧，保证一个正确的无冗余的数据包流传到应用层[5-7]。

目前我国航空总线的研究现状是已基本实现 ARINC429 通信协议，并在实际的飞行中表现出良好稳定的性能，但对于作为下一代航空飞行器的通信神经中枢的 AFDX 网络的研究和实现尚处于起步阶段。航空电子数据总线是航空电子系统的神经中枢，其质量和水平直接决定着航空电子综合化程度的高低和性能优劣。AFDX 航空数据总线通过硬件实现全双工技术，消除了传统以太网标准所固有的非确定性，为电传操纵(Fly-By-Wire)的终端航空电子设备提供了可靠的、无冲突的消息发送、接收机制。然而，现有的 AFDX 航空数据总线系统在可靠性和安全性方面，尚不能满足对于关键性飞行控制系统的苛刻要求。因此，研究 AFDX 网络，提高其安全性和可用性，对于我国研制具有自主知识产权的大型飞机具有积极作用。

本课题正是在分析系统端到端延时和延时抖动的基础上，对 AFDX 网络端系统和交换机进行建模，实现端到端的延迟和延迟抖动的控制，进而改进冗余管理机制，保证网络数据安全传输。

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器数据网络中强化安全级别的研究”，中国航空无线电电子研究所合作项目“AFDX 网络性能评估”项目中，希望对中国的大型飞机的研制起到积极的作用。

1.2 研究目标

对于航空数据通信来说，数据安全是一个尤为重要的问题，特别是一些关键的数据，例如油的的剩余量，飞机的位置等数据，必须确保能够安全可靠实时的传输[8]。传统的 802.3 以太网无法满足航空通讯总线的这些高可靠实时通讯的需求，所以航空全双工以太网 AFDX 对 802.3 以太网进行了改进，以满足航空通讯的需求。

以太网的缺点是它不能确保端到端的延时。这是由于以太网访问控制机制载波监听多路访问冲突检测(CSMA/CD)造成的。载波监听就是数据传输端能够检测到物理媒介是否空闲。多路访问就是多路数据传输终端能够同时接入同一个物理传输介质。冲突检测意味着数据传输终端能够检测到它的数据发送是否与其他终端的传输发生了冲突。在最初始的以太局域网中，它使用半双工的通讯模式，并且没有中央控制器。所以，当冲突

发生时，每个数据传输终端必须重新发送它的数据。这样有可能造成冲突的数据发送终端再一次同时的发送数据，从而造成再次的冲突。为了避免这种情况的发生，发送数据的终端在发生发送冲突时，会等待一个随机的时间 t，然后再重新发送数据。如果再次发生冲突，则等待 $2t$ 的时间，如此循环下去……这个策略也就是二进制指数退避策略。但是，这样造成的结果是数据包冲突的不确定性，从而导致数据包的延迟不确定。这个特性是航空网路无法忍受的特性[9]。

为了解决由此造成的问题，AFDX 使用全双工交换式以太网来消除二进制指数退避策略造成的延迟的不确定性。ARINC429 采用的是“总线节点”结构，各设备的接收机必须通过双绞线和惯性平台的发送机连接才能接收信号[10]。由于这种点到多的特性，航空电子系统必须为每个子系统的每个通信信道部署一条带有多个端点的 ARINC429 总线，这使得 ARINC429 总线系统的成本和重量都大幅度地增加。AFDX 采用 Ethernet 与 ATM 结合的技术，使用 Virtual Link (VL) 来代替 ARINC429 端到端的物理链路的链接。这样，就大大减轻了机载电缆的重量(波音 747 的使用 ARINC429 总线，电缆长度共有 250km，共有约 4500 个连接器，电缆和连接器占供货商电子产品价值的 30%[4])。AFDX 通过先进的排队管理和多带宽应用策略来实现对带宽的控制，并通过引入“带宽间隔分配”（BAG）的概念，最大帧长度等方式来进行带宽的分配。这些保障措施被应用到 AFDX 的虚拟链路方面，极大的提高了数据的完整性和数据传输在时间上的确定性[10]。为了增强系统的可靠性，AFDX 使用双冗余的网络。在端系统中，默认的情况下，每个需要发送的帧都要经过复制，然后分别发送到网络 A 和网络 B 上，最后帧和其副本都被传送到目的端系统中。在 AFDX 中，所有通过虚拟链路来传输的帧都提供了一个字节的序列号字段，这个序列号的范围是从 0 到 255，序列号 0 保留，用于端系统复位。当端系统在一定时间范围内，接收到两个序列号相同的帧时，就可以判断这两个帧为冗余帧。在接收端，使用第一个到达的帧有效（First Valid Win）的规则，也就是说如果两个副本都被正常接收，那么后到的帧被丢弃。根据帧的编号可以轻易判断同样编号的帧到达的顺序，避免帧的重复；如果其中一个帧出现传输故障，则可以用另一个相同的帧进行替代。接收端系统在每条虚拟链路内按顺序检查序列号，这就是“完整性检测”(IC)。对于无故障的网络来说，完整性检测的任务就是把帧传递到冗余管理部分；对于有故障的网络来说，完整性检测是要消除无效帧并且通知到网络管理部分。在完整性检测完成之后，端系统将根据帧的序列号消除冗余帧，这一过程就叫做冗余管理(RM)[9]。

但是调度器会引入抖动(jitter)，特别是在多跳网络中，两个冗余的网络 A 和 B 网络拓扑结构不同或者网络上的数据流相差很大时，两个冗余的数据到达的间隔比较大，造成当接收端在 B 网络接收到数据包之前，A 网络连续的接收到一个无效数据帧 A1 和一个有效的数据帧 A2，这时根据“先到的数据包有效”原则接收端系统就会把有效的数据帧 A2 提交给上层应用程序，这时即使 B 网络上的数据包 B1 有效的，也会造成也不

会接收 B1 的数据帧，这样就造成有效编号为 1 的有效数据帧被丢弃的问题。对于航空数据通信中一些对数据安全要求特别高的数据来说，这样的情况时无法容忍的。

因此，本文主要研究为：

AFDX 端到端的延时保证和 delay jitter 控制机制，在 AFDX 网络中加入时延抖动控制机制 jitter-EDD，确保端到端系统的延时，控制端到端系统 jitter，在此基础上提出新的冗余管理算法，降低冗余管理算法的复杂度，提高系统的可靠性。

在 AFDX 网络中，加入 delay jitter 控制机制，需要对 AFDX 网络节点的模型进行改进，建立配套的网络模型。另外，传统的交换机模型都是基于物理链路的调度，这样的结构无法保证每条虚拟链路的性能需求，无法保证虚拟链路对带宽的需求。因此，要改进传统的基于物理链路调度的交换机模型，建立基于虚拟链路调度管理的交换机模型，从而保证每条虚拟链路对于调度延迟和延迟抖动的要求，从而保证通讯网络的安全、高效、可靠。

现有的 AFDX 航空数据总线系统在可靠性和安全性方面，尚不能满足对于关键性飞行控制系统的苛刻要求。究其原因，当多个终端连接到一个交换机(Switch)上，并且在同一个终端上的多个子系统(subsystem)需要同时发送消息时，存在严重的资源竞争问题，往往导致系统性能的不确定性甚至系统崩溃。因此，研究网络节点上任务的可调度性就显得很必要。

使用网络控制系统仿真软件 SimEvents 作为 AFDX 网络仿真工具，建立起节点数≥8，交换机数目≥6，通讯速率≥10M 的 AFDX 网络，实现端到端时延的控制机制，使用错误注入的方法来验证新设计的 AFDX 网络冗余管理机制对于各种网络错误的使用效果。

本课题在给出的网络模型的基础上，给出具体的可调度性分析方法，对于实际网络的实施具有一定的指导意义。

1.3 国内外研究现状

以太网最早是由 Xerox 等公司在上世纪 80 年代初期推出的局域网，1983 年被 IEEE802 委员会采纳，其 MAC 层协议由 IEEE 802.3 标准定义。上世纪 90 年代，交换式全双工网络互联技术从根本上屏蔽了传统以太网的 CSMA/CD 信道访问机制，使以太网的实时通信应用成为现实[4]。

在航空电子领域，最初波音 767-400 只在座舱显示系统采用了交换式以太网；在波音 767-400ER 的通信与任务管理系统中，采用了静态 TDMA 机制的交换式以太网。直到上世纪 90 年代中期，交换式以太网在空客 A380 型飞机中的应用逐渐明朗化。作为一种经过全面的航空电子适用性改造的交换式以太网技术，它不仅给出了航空电子网络的解决方案，而且结束了交换式以太网在航空应用中呼声很高但莫衷一是的局面，飞机数据网络(AND)的领域模型也清晰起来[4]。

AFDX 由 A380 项目中的产品演化为标准。2000 年前后，航空电子工程委员会（AEEC）暨 ARINC 公司发布了 ARINC664 第 7 部分规范草案，对“航空电子全双工交换式以太网”进行了规范化说明，而 AFDX 的物理层、数据链路层和应用层由 ARINC7-664 的其他部分定义[4]。

国外对于 AFDX 的研究始于 A380 项目，早在上世纪 90 年代初期德国和法国的相关高等院校就在 Linux 操作系统下构造 AFDX 的原型。对于 AFDX 的航空电子系统总体设计与工程应用研究，主要集中于法国图卢兹的空客研究所，以及法国高等航空工程学校，图卢兹“En7”学院等研究机构。20 世纪 90 年达中期，我国的相关的研究院所与高等院校也对交换式以太网的实时应用开展了研究。目前，交换式以太网已经在工业现场控制和舰船作战平台等领域获得了应用[4]。

Von Hanxleden and Gambardella 等人的对 AFDX 的冗余管理机制进行了研究，给出了详实的报告。报告共 75 页，包括 37 个推论和 86 幅图。这些图大部分都是针对不同的通讯场景的，具有实际的意义，因此从侧面反映了冗余管理的复杂性[11]。

Täubrich 利用形式化验证语言 TLA(Lamport’s Temporal Logic of Actions)来对 AFDX 冗余管理机制进行建模，采用 TLC 进行验证，对 Von Hanxleden 等人给出的 13 种冗余管理算法进行了细致入微的分析，给出了这 13 算法 Safety、Liveness、Quality、Availability 方面的评估[12]。

Madhukar Anand、Samar Dajani-Brown、Steve Vestal and Insup Lee 等人建立了形式化的 AFDX 帧管理模型来验证 AFDX 设计的可靠性。他们使用时间自动机来建立模型，用 UPPAAL 来进行模型检验 ，以此来分析精确的时序特性。结果发现，AFDX 帧管理策略不能解决有效数据包丢失的问题。他们建议添加 buffer 来解决这个问题[13]，但是 buffer 会引入队列延迟，导致端到端延迟的不确定性。

陈昕等人围绕系统中的冗余管理问题，针对冗余帧发送时间间隔 SkewMax，进行了较为深入的分析和研究，利用 Network Calculus 对 AFDX 网络进行研究，定量分析其数据包的延迟抖动，提出了较为有效的冗余管理算法 SKRM。然后，运用时序自动机模型的验证工具 UPPAAL，对所提出的 SKRM 冗余管理算法进行仿真验证。结果证明，SKRM 冗余管理算法是有效的，可以保证航电网络数据传输可靠性要求[14][15]。但是并未给出保证端到端延迟抖动的具体方案。

李哲等人对 ARINC 664 Part7 协议中关于 SkewMax 的内容进行了深层次剖析，并针对终端系统的接收端多个虚链路，多个 SN(Sequence Nubmer) 号余度帧之间的 SkewMax 的测量，提出了一种有效的解决办法，对于 AFDX 网络的设计和测试有一定参考价值[16]。

Hussein Charara 等人用 QNAP2 (Queuing Network Analysis Package)建立一个仿真模型用来描述 AFDX 网络，分析了 AFDX 的网络性能,并且与用 Network calculus 进行的分

析做了对比[3]。

Dinesh C.Verma 等人研究在包交换存储转发网络中，delay jitter 的控制方法。他们通过仿真的方法，验证了解决方案的正确性，并且研究了它的性能。结果表明，他们的方法可以显著的减少 jitter，并且在端到端的链路上,jitter 不累加。Jitter 控制可以减少所需 buffer 大小[17][18]。

Qin Zheng 等人给出了建立实时通道的数学基础，主要是：(1) 在同一个物理链路上的一组通道的可调度性的充分必要条件。(2)计算一个通道可调度的最小 delay bound。只要给定一组通道的 traffic 特性，就可以判断是否所有的数据包可以被有效的调度。这个结论对于普通的实时任务调度问题也具有重要的价值[19-22]。

Jean-Philippe Georges，Eric Rondeau 和 Thierry Divoux 使用缓冲器、复用器等基本组件构建网络模型，对不同的工业以太网中通信情景计算最大延迟，对网络规模以及周期性和非周期性的数据大小提出了建议和指导[23]。Frédéric Ri- douard，Jean-Luc Scharbarg 和 Christian Fraboul 使用随机网络演算对时延的分布进行了计算[24]，并且得出结论：相对于确定性的网络演算对最大时延的估计，随机网络演算所获得的值要更小，或者说更有效。

关于交换机上的调度算法，Christian Fraboul 等人对其进行了研究[25-27]，分析了先来先服务、固定优先级、加权公平排队等算法对端到端网络时延的影响。其中对网络节点的建模计算，得益于 Rene L. Cruz 对网络模型的研究[28]。同时，Marc Boyer 和 Christian Fraboul 在计算延时边界的时候，引入了物理通信链路的带宽对网络传输的影响[29]，使延时边界减小了 40%。针对多个数据流经过采用先来先服务调度的交换机的情形，进一步研究给出了时延边界。此外，还对“pay burst only once”现象的使用情况给出了说明。

贾秋玲等人对于航空总线通信的冗余设计进行了探讨，而朱大奇等人对于容错冗余基本原理、故障诊断技术和故障诊断的信息融合方法进行了介绍[30-33]。

1.4 论文结构

本文一共分为 7 章：

(1) 第 1 章 绪论。详细介绍本论文研究背景，研究的目标，国内外研究现状，概要的描述了研究的关键技术。

(2) 第 2 章 AFDX 网络架构与 jitter-EDD 机制。主要介绍 AFDX 网络的基本架构，以及 jitter-EDD 机制的原理。

(3) 第 3 章 AFDX 网络建模。主要对接收端系统、发送端系统和交换机进行了建模，详细介绍了实施 jitter-EDD 机制的关键细节，并给出了新的冗余管理算法。

(4) 第 4 章 AFDX 网络模型分析。首先对端到端的延时进行了分析，接着介绍了 deadline 分配的算法，给出了 AFDX 网络可调度性分析的充分必要条件，最后对新的冗

余管理算法进行了理论分析。

(5) 第 5 章 AFDX 仿真平台的搭建。首先对仿真工具 SimEvents 中的基本概念-实体、事件、实体端口、路径、数据和信号做了介绍，接着对 SimEvents 的产生器、属性、服务器、路由模块、实体管理、信号管理、子系统等主要模块进行了说明，最后详细介绍 AFDX 网络仿真中接收端系统、发送端系统、交换机和错误注入模块的架构和原理。

(6) 第 6 章 实验与仿真结果。通过 SimEvents 仿真工具搭建的 AFDX 网络仿真平台，研究 AFDX 网络端到端的延迟抖动和验证新设计的网络冗余管理算法的可靠性。

(7) 第 7 章 总结。对于全文的研究的总结。

1.5 本章小结

本章主要介绍了 AFDX 网络容错的研究背景和所要达到的研究目标，概要的介绍了论文研究的关键技术和所使用的工具，并对 AFDX 网络冗余管理、帧管理、端到端延时分析、AFDX 建模等国内外现状进行了描述，对论文的结构进行了总体上的描述。

2 AFDX 网络架构与 jitter-EDD 机制

2.1 AFDX 网络架构

AFDX 网络由以下三部分组成：

(1) 航空子系统：飞行器上的传统的航空电子系统有全球定位系统，航空计算机，压力监视系统等。为航空子系统提供计算环境的是航空计算机，航空子系统通过航空计算机内的 AFDX 终端系统连接到 AFDX 交换机，进而与其他航空子系统进行可靠的通讯。

(2) AFDX 终端系统：终端系统是航空子系统与 AFDX 分组交换机连接的接口。每个航空子系统对应的终端系统保证了航空电子系统之间数据交换的安全和可靠性。该接口导出的应用程序编程接口(API)使得各航空电子系统通过简单的报文接口进行通讯。

(3) AFDX 交换机：全双工分组以太网交换机，通过它可以快速的把以太网数据转发给适当的目的地。分组交换以太网技术与传统的 ARINC429 单向传输技术，和 MIL-STD-1553 总线技术是不同的。

2.1.1 端系统和航空子系统

如图 2-1 所示，航空计算机通过端系统(End System)与 AFDX 网络互联。通常情况下，一个航空计算机支持多个航空子系统。通过限制地址空间和 CPU 时隙把位于同一台航空计算机中的子系统进行隔离，这样防止位于不同分区(Partition)的航空子系统之间相互影响，提高了系统的安全性。

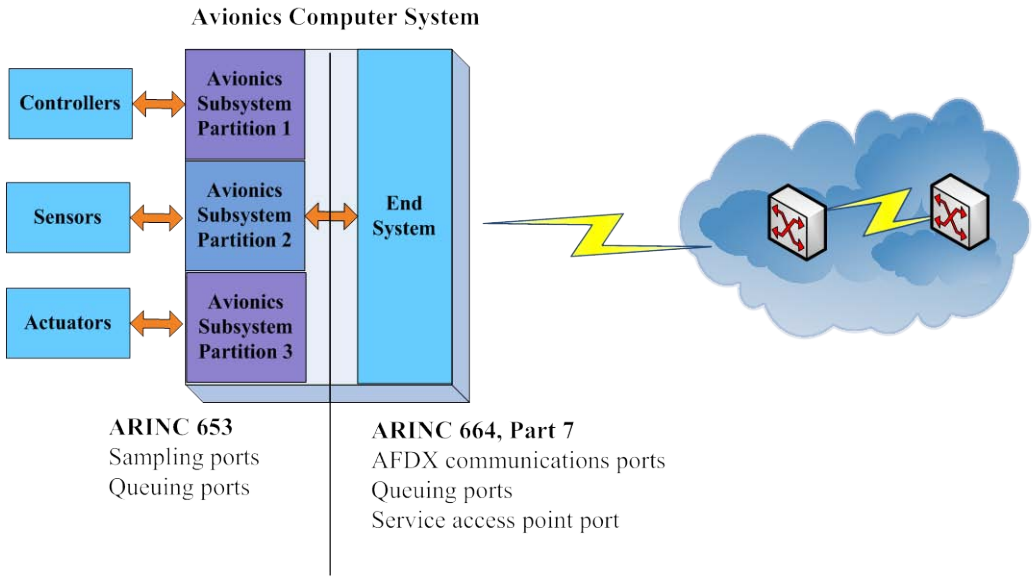

图 2-1 端系统和航空子系统

Fig.2-1 End System and Avionics Subsystems

航空应用程序之间通过消息来通讯，这些消息是基于通讯端口的。ARINC653 定义了两类通讯端口-采样端口(sampling port)和队列端口(queuing port)。ARINC664 part7，也就是 AFDX，定义了三类端口，采样端口和对列端口分别与 ARINC653 的采用端口和队列端口对应，除此之外，还定义了服务访问端口(service access port-SAP)，作为 AFDX 系统与非 AFDX 系统之间通讯的接口。

图 2-2 和图 2-3 分别表示采样端口和队列端口。每个采样端口仅包含一个消息的 buffer，当下一个消息到来时，就会重写消息缓冲区。读消息并不会清空消息缓冲区，因此可以重复的进行读消息的操作。当每个采样端口刷新对应的消息缓冲区时，给出某种提示以通知接收航空子系统是消息停止发送还是在发送重复的消息。而队列端口是一个长度可配置的 FIFO 队列，当消息被取走时，对应的缓冲区清空[9]。

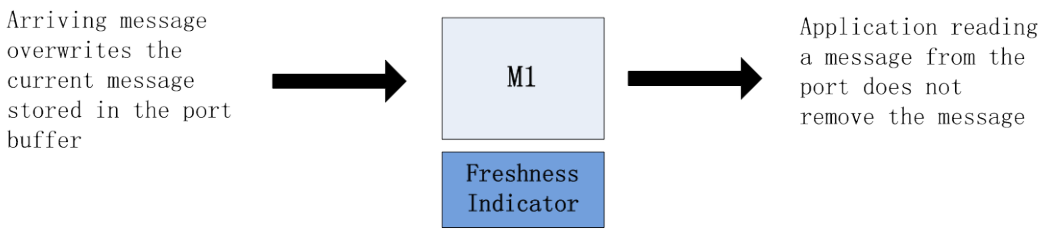

图 2-2 接收端的采样端口

Fig.2-2 Sampling Port at Receiver

图 2-3 接收端的队列端口

Fig.2-3 Queuing Port at Receiver

2.1.2 虚拟链路(Virtual Link)

在传统的以太网交换网络中，以太网帧基于以太网目的地址进行路由，而 AFDX 网络中使用 16 比特的 Virtual Link ID 进行路由。图 2-4 为 AFDX 网络中以太网帧格式。

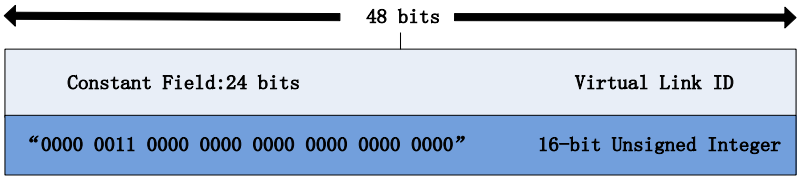

图 2-4 AFDX 网络中以太网目的地址帧格式

Fig.2-4 Format of Ethernet Destination Address in AFDX Network

在 AFDX 交换机中，根据配置的静态路由信息，进入的以太网帧被发送到一个或多条链路上，但是具有相同 Virtual Link ID 号的所有的以太网帧都来自同一个端系统。来自于同一个端系统的数据帧被发送到一组确定的端系统上，这类似于 ARINC429 多路总线。

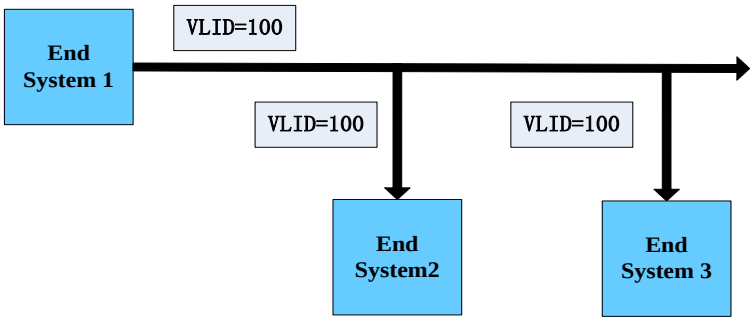

图 2-5 包路由实例

Fig.2-5 Packet Routing Example

如图 2-5 所示，当源端系统 1 发送 Virtual Link ID(VLID)为 100 的数据帧时，经过 AFDX 交换机后，发送到一组确定的目的端系统(2 和 3)。每个端系统包含多个虚拟链路，每个虚拟链路包含多个通信端口上的消息。

当应用程序发送数据帧时，它会首先发送该消息到某个通信端口上，然后经过源端系统、交换机、目的端系统，最后到达目的通讯端口，完成数据包的发送。

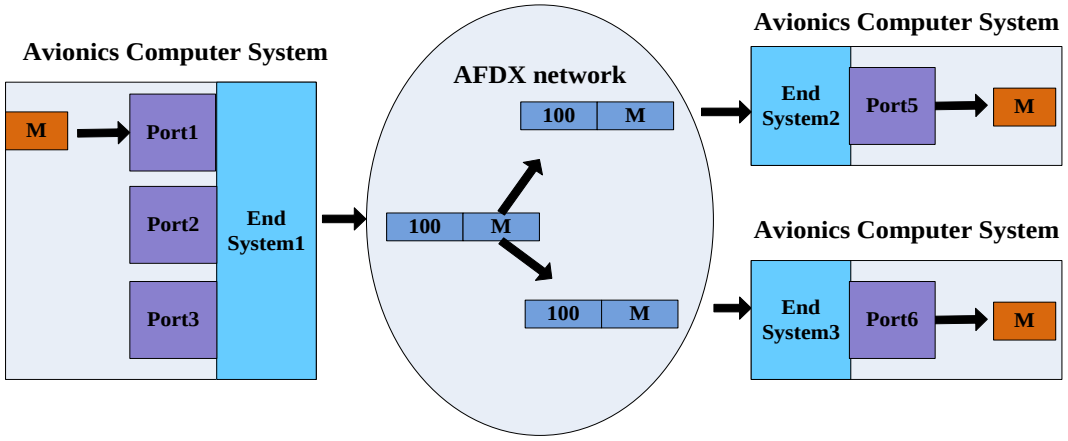

图 2-6 消息传递到端口 1

Fig.2-6 Message Sent to Port1

图 2-6 显示了消息 M 的传输过程。首先，消息 M 被发送到端口 1，源端系统把该消息封装成以太网帧，并把该以太网帧发送到 AFDX 网络 Virtual Link 100 上。AFDX 交换机根据配置好的静态路由表，将该以太网帧发送到目的端系统 2 和 3。目的端系统通过查看该以太网帧数据的头部中目的端口信息，将该帧发送到对应的通信端口，图中，端系统 2 把该以太网帧发送到端口 5，而端系统 3 把该以太网帧发送到端口 6。

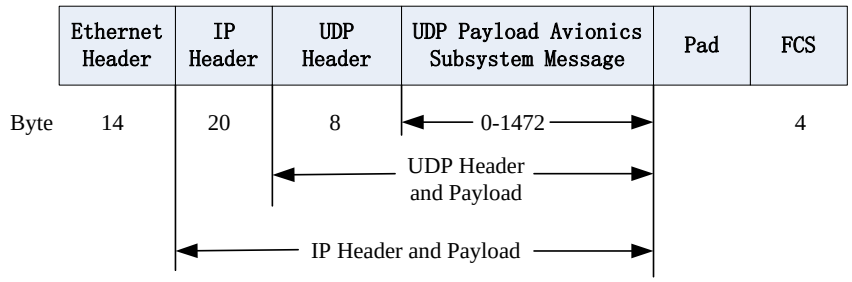

图 2-7 局域网帧

Fig.2-7 Ethernet Frame

图 2-7 显示了以太网帧的负载域内容，其为 IP 数据包(包括 IP 包头和 IP 负载域)。IP 负载域承载 UDP 包，其中封装了航空子系统发送的消息。只有当 UDP 负载域的长度小于 18 字节时，Pad 域才有效，这时，Pad 域和 UDP 负载域共占 18 字节。当 UDP 负载域大于等于 18 字节时，没有 Pad 域。当 UDP 包数据内容比较多时，UDP 数据包会被分到几个 IP 数据包中，此时 IP 头负责分片的控制功能。IP 头部为目的端系统标识和航空子系统标识，或者为一个多播地址。当 IP 头部为多播地址时，IP 目的地址包含虚拟链路 ID。UDP 包头包含源端口号和目的端口号。

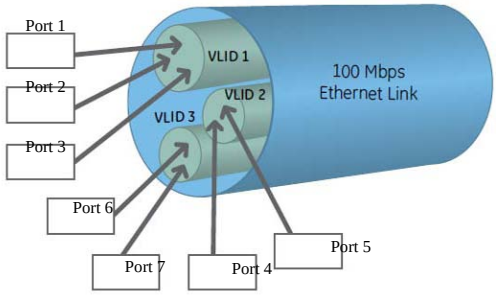

图 2-8 包含三个虚拟链路的物理链路

Fig.2-8 Three Virtual Links Carried by a Physical Link

端系统 100Mbps 的物理带宽能够支持多个虚拟链路，他们共享 100Mbps 的物理带宽。如图 2-8 所示，该物理链路包括三个虚拟链路。AFDX 通讯端口 1、2、3 在虚拟链路 1 上，通讯端口 4、5 在虚拟链路 2 上，通讯端口 6、7 在虚拟链路 3 上。

像航空子系统之间通过分区(Partition)实现隔离一样，虚拟链路之间也需要实现隔离。AFDX 网络中，虚拟链路的隔离是通过限制以太网帧发送的速率和以太网帧的大小来实现的，由源端系统来保证。

每个虚拟链路包含两个参数：

1) 带宽分配间隙(Bandwidth Allocation Gap-BAG)，范围为 1 到 128 毫秒，为 2 的指数倍(BAG 允许分配的值如表 2-1 所示)。

2)L_{max}，在虚拟链路上允许传输的最大以太网帧长度，单位为字节。

表 2-1 BAG 许可的值

Table 2-1 Allowable BAG Values

BAG(milliseconds)	Hz
1	1000
2	500
4	250
8	125
16	62.5
32	31.25
64	15.625
128	7.8125

BAG 代表虚拟链路上，帧与帧之间发送的最小时间间隔。如果一个虚拟链路的 BAG 为 32 ms，L_{max} 为 200 字节，则在该虚拟链路上，以太网帧发送速度最大 32ms/帧，最大带宽为 50000bps(200*8*1000/32)。

BAG 的选择是由通信的需求来决定的，例如，某个航空子系统在三个通讯端口上通讯，这三个通讯端口位于同一个虚拟链路上，三个通讯端口上的消息频率为 10Hz、20Hz、40Hz。那么，三个通信端口总共通讯频率为 70Hz，平均的消息发送周期为 14.4ms。

为了确保通讯的带宽，应该选择 BAG 小于 14.4ms，所以应该选择 BAG 为 8ms。

2.1.3 AFDX 交换机

交换机的基本功能是数据帧的接收和转发。按照 ARINC664 part7 的要求，如图 2-9 所示，交换机应该具有模块为：配置表(Configuration Tables)、交换功能(Switching Function)、终端系统(End System)、监控功能(Monitoring Function)、过滤仲裁(Filtering&policing Function)等。

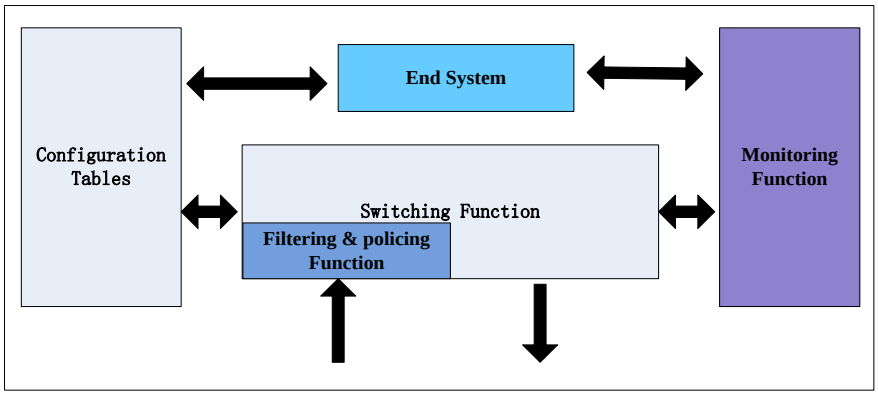

图 2-9 AFDX 交换机功能结构图

Fig.2-9 Functional Blocks of the AFDX Switch

过滤和仲裁功能分为几个不同的功能，包括帧的完整性，帧长度，流量预算，以及目的地址的合法性等。所有到达交换机的帧都要首先经过过滤与仲裁功能模块的处理，然后提交给交换功能模块。

交换功能执行交换的核心工作。经过仲裁过滤功能处理的帧被转发到合适的物理输出端口，通过这些端口，数据包再次离开交换机。

这些功能是由静态配置表中的配置数据项来控制。

终端系统模块提供了外部设备与交换机通信的方法（将接收的帧传送给交换机，并允许交换机向外发送帧）。这主要是用于数据加载以及监视功能的实现。

所有的操作都被监控功能模块监视，该模块记录事件（日志），诸如：某个帧的到达，或者某次 CRC 校验失败；并且还创建关于内部状态的统计量。因为交换机是网络的一部分，监控功能与网络管理功能通信，进行操作信息和有关于健康状态的信息的交互。

2.1.4 虚拟链路调度和 Jitter

AFDX 每个通信端口对应唯一的一个虚拟链路， UDP、IP 和以太网头部封装起来的消息被发送到通讯端口，最后发送到与之对应的虚拟链路缓存队列，等待发送。虚拟

链路缓存队列中的以太网帧数据是由端系统虚拟链路调度器来调度的。虚拟链路调度器负责同一个端系统上的所有虚拟链路的数据的调度。

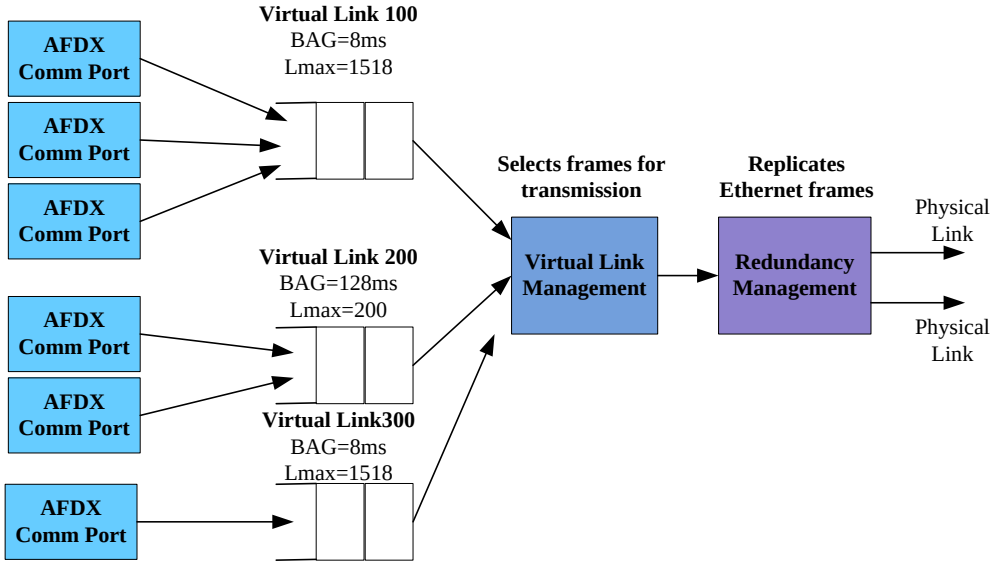

图 2-10 虚拟链路调度

Fig.2-10 Virtual Links Scheduling

图 2-10 显示了虚拟链路调度的过程。虚拟链路调度器的任务是确保每个虚拟链路的带宽限制。虚拟链路调度器不仅确保 BAG 和 L_{max} 限制，它亦负责将多个虚拟链路复合成一条链路。所以，复合所引入的延时抖动必须能够确保满足一定的界限。

发送到 AFDX 通讯端口的消息的时间由航空子系统和负载设备需求来决定的。例如，传感器发送的数据的频率为 10Hz。 如果消息在队列非空时到达，就会引用抖动。ARINC664 标准规定，端系统发送出的数据包的虚拟链路延迟抖动必须满足公式 2-1 和 2-2。

$$\max\_jitter \le 40us + \frac{\sum_{i\in\{SetOfVLs\}} (20 + L_{\max j})*8}{Nbw} \tag{2-1}$$

$$\max\_jitter \le 500\mu s \tag{2-2}$$

Nbw 代表链路带宽 100Mbps。

第一个公式表明端系统若具有较少的虚拟链路并且其中待处理的数据帧是短帧，则最大允许的时延抖动将较低。在所有的情况下，抖动被限制在 500us 的界限内，以限制对整个网络确定性的影响。从多个虚拟链路中选择了某个帧后，打上基于虚拟链路号的序列号，然后该数据帧被送到冗余管理模块进行复制，最后发送到链路上。

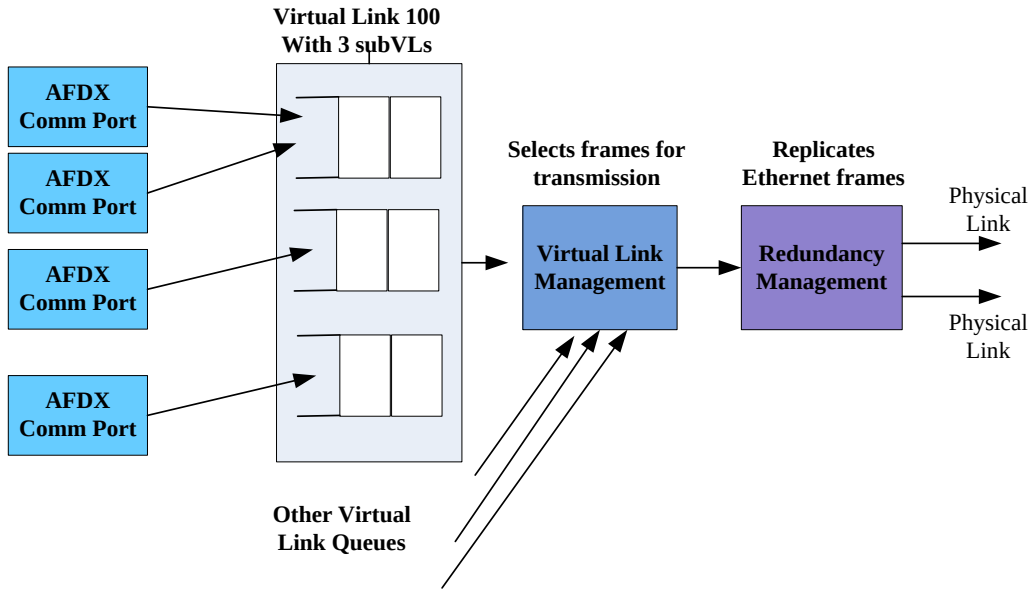

图 2-11 子虚拟链路调度

Fig.2-11 Sub-VLs Scheduling

图 2-11 描述了有三个子虚拟链路的处理过程。虚拟链路调度器将这三个子虚拟链路当做一个虚拟链路来调度，子虚拟链路中根据 round-robin 策略来选择数据包。显然，如果某个虚拟链路有多个子虚拟链路，在实际虚拟链路调度之前，以太网数据帧中不添加序列号标签的，只有在帧实际调度时，才打入帧序列号。

虚拟链路的调度包含两部分：包整流和复用。图 2-12 示意了整流器的工作过程，不规则的数据流经过整流后，输出没有延时抖动的数据流。虚拟链路调度器应该负责将整流器输出的数据流进行复用、整合，提交给冗余管理模块，进行数据的复制和发送到物理链路上。当整流器的输出多个数据流进行复用时，会经历队列延时，导致延时抖动。

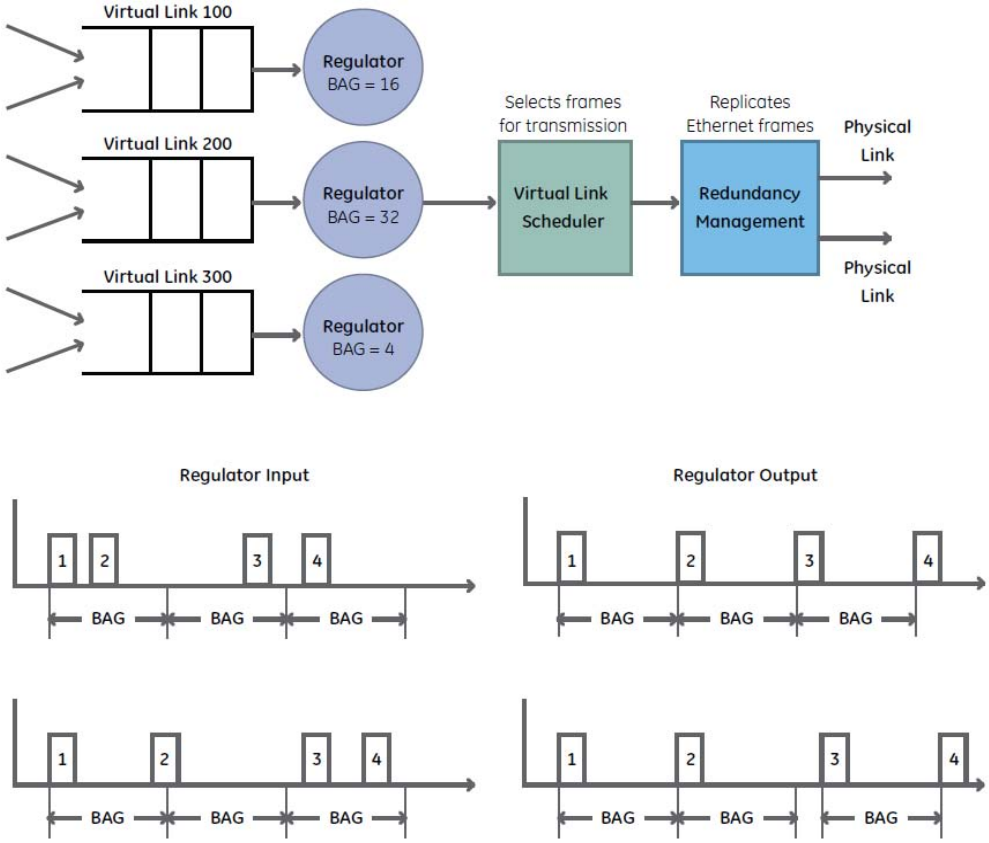

图 2-12 虚拟链路整流

Fig.2-12 Virtual Link Regulation

2.1.5 冗余管理

AFDX 网络中，端系统把数据包在两个相互冗余的网络 A 和 B 上发送，正常的情况下，每个接收端系统会收到两个同样的数据包(如图 2-13 所示)。

每个端系统需要某种方法来鉴别来自 A 和 B 网络的数据包。AFDX 中，每个传输的数据包有 1-byte 的序列号域。序列号域位于 FCS 域的前面，这意味着，AFDX 中，IP/UDP 负载域减少 1 个字节。这个序列号从 0 到 255，然后回滚到 1。序列号 0 是端系统重启帧[5][9]。

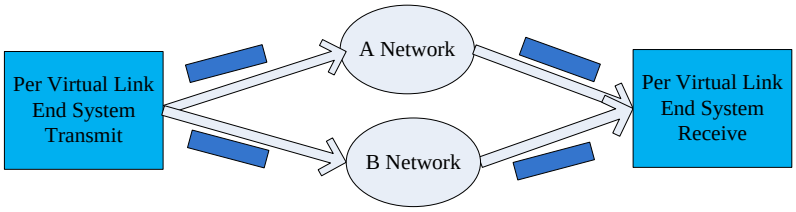

图 2-13 A 和 B 网络

Fig.2-13 A and B Networks

端系统检查后续帧的序列号是否顺序正确。这个操作就是完整性检查(Integrity

Checking)。然后，端系统根据 A 和 B 网络的数据包来决定数据包是否接收。这个操作就是冗余管理(Redundancy Management)，如图 2-14 所示。

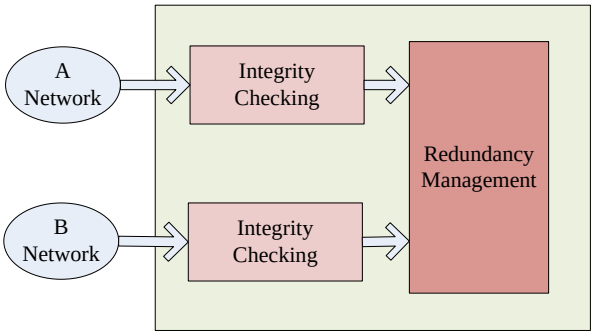

图 2-14 帧接收过程

Fig.2-14 Receive Processing of Frames

冗余管理期望的行为如图 2-15 至图 2-18 所示[5]。

例一：非正常发送帧(Abnormal Transmitted Frame)

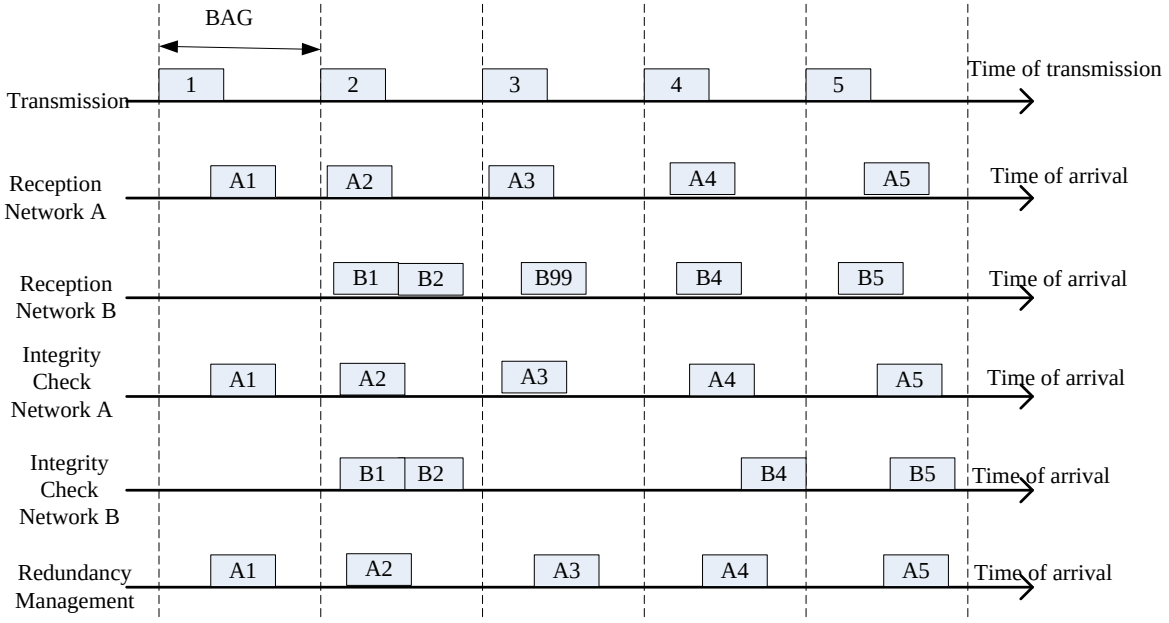

图 2-15 网络 B 发送一个不正常的帧

Fig.2-15 Network B Transmits an Abnormal Frame

如图 2-15 所示，当发生非正常发送帧的情况时，冗余管理结果是非正常帧不会被发送到 Partitions。

例二：丢失一个帧

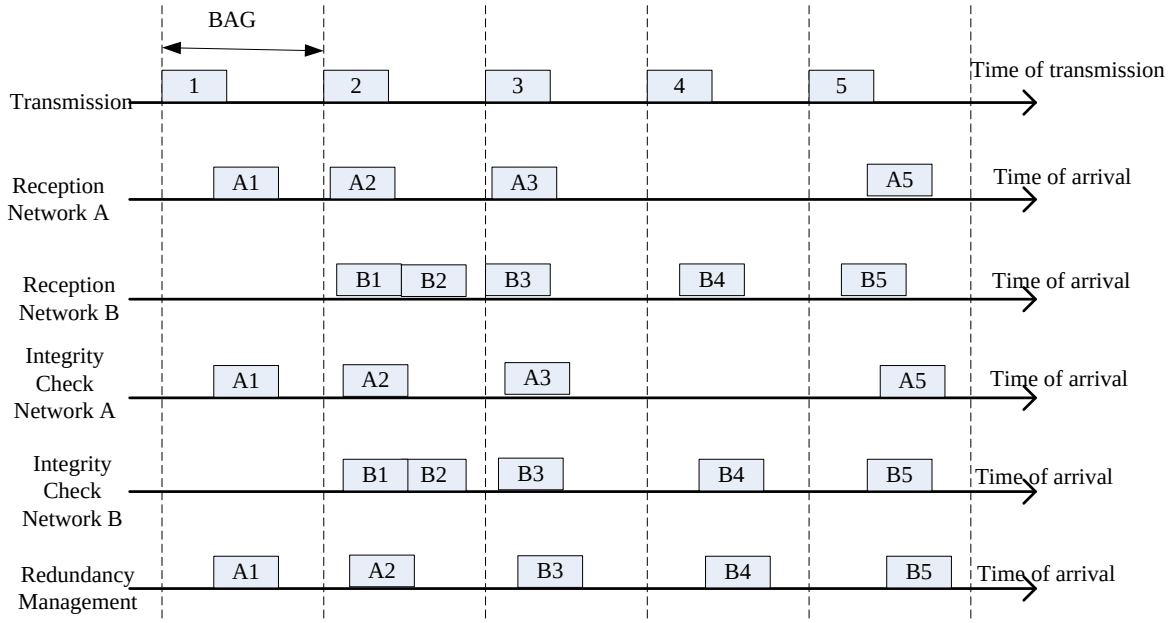

图 2-16 网络 A 丢失一个数据包

Fig.2-16 A Packet is Lost on Network A

如图 2-16 所示，由于比特错误，帧“A4”丢失，网络 B 上的到达的对应的数据包被接收。

例三：帧重启

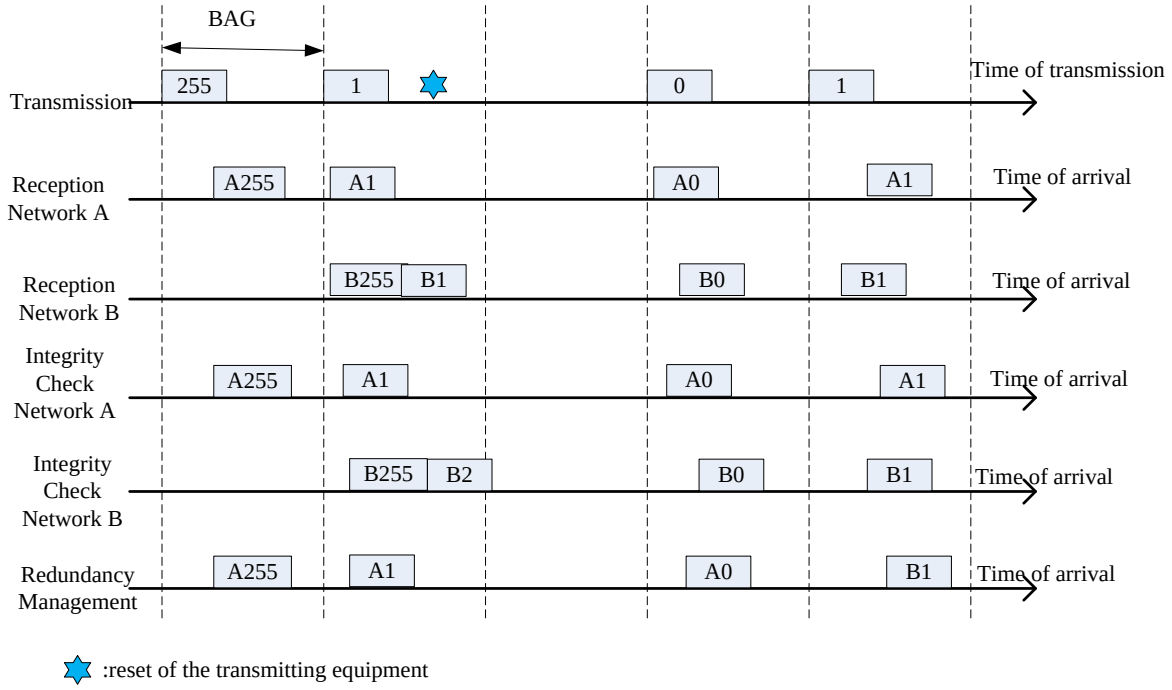

图 2-17 传输端系统重启

Fig.2-17 Reset of the Transmitting End System

如图 2-17 所示，当网络重启时，冗余管理结果是数据包不会丢失。

例四：交换机紊乱(Babbling Switch-stuck frame)

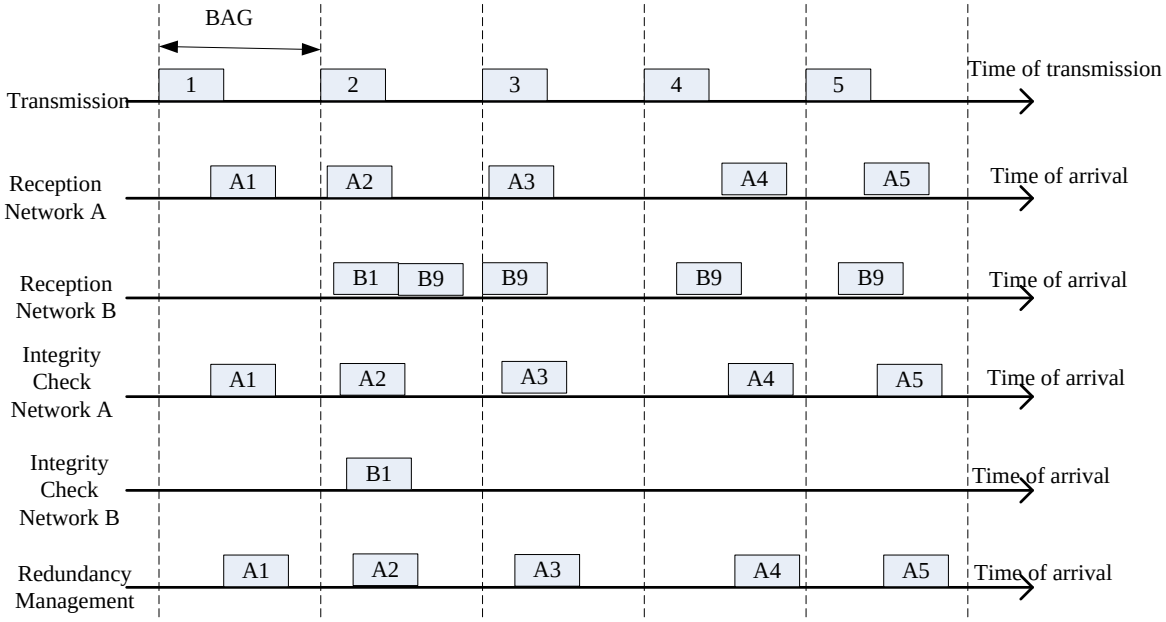

图 2-18 网络 B 混乱

Fig.2-18 Babbling on Network B

如图 2-18 所示，当发生交换机紊乱时，经过完整性检查后，帧被过滤掉，不会传给 IP 层。

2.2 Jitter-EDD 机制

Jitter-EDD(Earliest Due Date)是加州伯克利大学提出来的一种控制端到端延迟抖动的方法[34]，它基于简单的固定路由链接（实时通道）来连接，路由是在通道建立的时候确定的。为了确保通道的服务的质量，客户端建立连接时应该给出流特性和性能需求，如最小包间隙、最大包长、端到端的延迟界限等。下面对 jitter-EDD 进行介绍。

2.2.1 Jitter-EDD 网络模型

在这个网络中，每个节点有两个功能：整流和确保流经过调度后，流形变较小。 所以每个节点可看成两部分。整流器，负责将经过的数据流进行整形重构；延迟可限定的服务器(Bounded-delay server)，服务器确保流的形变在下一个节点整流器可接受的范围之内。网络节点模型如图 2-19 所示。

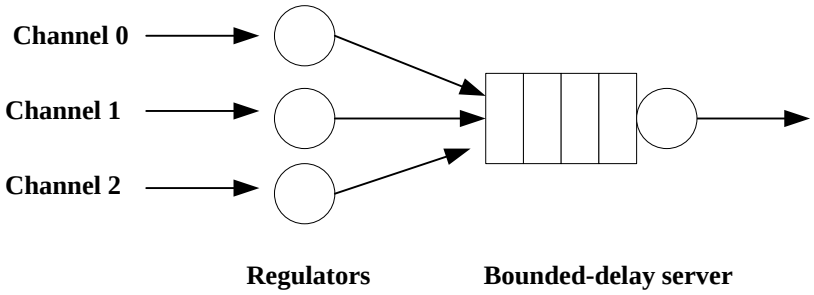

图 2-19 jitter 控制节点模型

Fig.2-19 The Node Model for Jitter Control

延迟可限定的服务器采用 EDF 调度算法，经过确定性测试（用来确保节点有足够的处理能力和充足的带宽），延迟界限测试（确保加入新的 channel 后，所有 channel 的 deadline 限定能够满足）和 缓存空间测试（确保加入新的 channel 后，有足够的缓存），保证端到端的延迟抖动。

2.2.2 Jitter-EDD 原理

Jitter-EDD 机制是面向包交换网络而提出的，设定每个在线上的数据包包含一个域，这个域的值是包的 deadline 与实际完成时间之间的差值。Jitter-EDD 整流器会保持包直到 eligibility time 才会被整流器调度。 每条 channel 都有一个 jitter-EDD 调度器，它负责把到达的包进行整流，图 2-20 示意了包通过两个连续的节点时的情况。

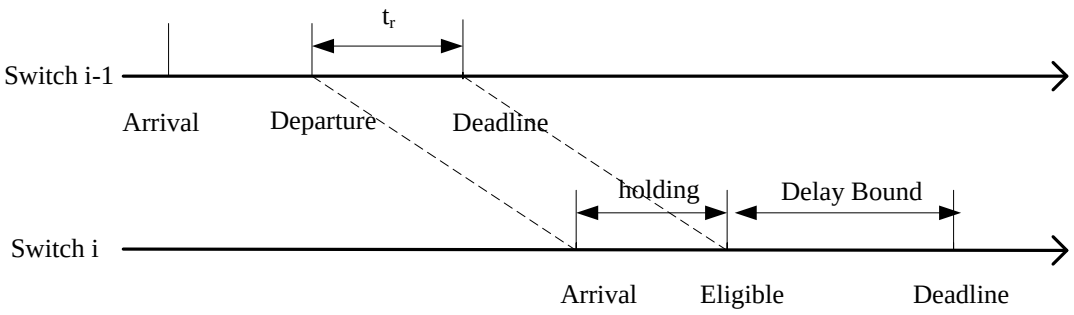

图 2-20 jitter-EDD 机制

Fig.2-20 The Jitter-EDD Mechanism

在第一个服务器(Switch i-1)上，包在它的 deadline 之前 t_r 秒被服务。所以在第二个服务器(Switch i)上，jitter-EDD 确保这个包被保持 t_r 时间段后，才能被调度。所以，任意两个服务器间的包有效时间间隔都是确定的，这样数据流就有一个确定的抖动界限。端到端的路径可以被看做是多个这样的节点对，所以端到端的延迟抖动也是确定的。

2.2.3 Jitter-EDD 问题与改进

为了在 AFDX 网络中实现端到端的延迟抖动控制，根据 AFDX 特有的网络特点，本文需要对原始的 jitter-EDD 机制进行了改进。

首先，原始的 jitter-EDD 机制中，channel 的建立每次都要申请，并进行确定性和延迟限定测试，但 AFDX 网络是静态路由网络，虚拟链路的数目是确定的，并且每条虚拟链路的带宽是提前设定的，所以只需要离线计算虚拟链路的的调度性，这样省去了 channel 建立的过程，节省大量处理器资源。

其次，原始给出的延迟界限测试的条件为充分非必要条件，这样就造成一部分可以运行的网络不能运行，所以本文根据最新的理论研究成果，给出了充分必要条件。

再次，在原始的 jitter-EDD 方案中，在目的接收端系统中并没有 jitter-EDD 整流器，所以端到端的延迟抖动为最后一跳得延迟界限。为了确保数据的安全传输，本文在 AFDX 的接收端系统中也加入了 jitter-EDD 整流器。

最后，原始的 jitter-EDD 方案中，对于全局 deadline 的分配没有给出有效的算法，本文提出了基于权重的 deadline 分配方案。

在 AFDX 网络中加入 jitter-EDD 机制会增加包的延迟，减少节点的输出流量，但是它提供了更好的延迟抖动控制。在本文的方案中，最好的情况下，甚至能够使端到端的延迟抖动减少到 0。

2.3 本章小结

本章首先对 AFDX 网络架构进行了介绍。AFDX 主要由端系统和交换机组成，端系统负责与航空子系统通信，多个端系统之间通过交换机进行消息传递。为了确保通信链路的质量，AFDX 使用 ATM 中虚拟链路的概念，对包大小和包间隙进行限制。AFDX 网络中，数据包在相互冗余的 A 和 B 网络上发送，以保证数据的可靠传输，因此需要在接收端进行网络冗余管理，消除冗余的数据包。其次，对 jitter-EDD 的网路模型和网络原理进行了介绍，并根据 AFDX 网络的特点，对原始的 jitter-EDD 机制进行了相应的改进。

3 AFDX 网络建模

AFDX 网络中，包括两种物理元素，AFDX 端系统和 AFDX 交换机。AFDX 端系统负责从航空子系统获得数据包和把收到的数据包发送给 AFDX 交换机。AFDX 交换机负责将数据进行路由，将源目的端口数据发送到目的端口。但是在这里，为了更好的说明端系统的功能，将端系统逻辑的划分为发送端系统和接收端系统。在实际中，需要将发送端系统与接收端系统整合到同一个物理介质中。

3.1 发送端系统

发送端系统将来自于航空应用软件的数据发送到 AFDX 物理链路上去，如图 3-1 所示。

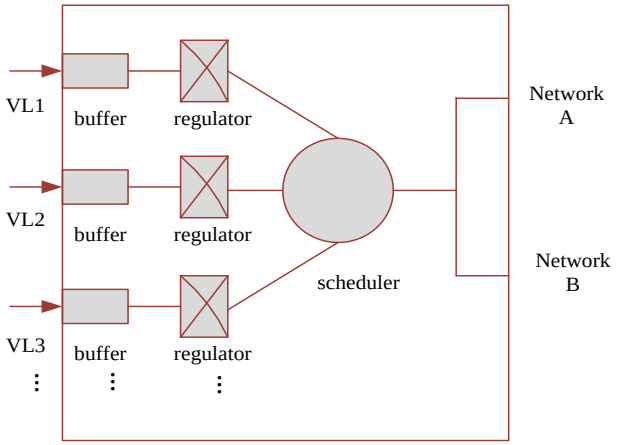

图 3-1 发送端系统

Fig.3-1 The Sending End System

在本文的模型中，AFDX 发送端系统中的调度器是 EDF(earliest deadline first)调度器，它负责将来自不同的虚拟链路的数据进行复合，并发送到 AFDX 物理链路上。为了确保 AFDX 网络的 QOS，AFDX 在发送端系统中添加一个整流器。该整流器负责保证同一个虚拟链路上的数据包之间的时间间隔最小是 BAG。它是一个特殊的缓存，每经过 BAG 时间，就可以从这个特殊的缓存中取出一个数据包，并允许调度器对该数据包进行调度。

当虚拟链路 i 上某个数据包可以被调度时，设定该时刻为 $t_{i,u}^{e}$ (e 代表端系统，u 代表数据包可被调度器调度)，也就是发送端系统上数据包可用时间。当这个数据包经过

EDF 调度器调度，将要被发送到 AFDX 物理链路上离开发送端系统时，这个时刻为 $t_{i,o}^{e}$(e 代表端系统，o 代表数据包离开，i 代表虚拟链路 i)。发送端系统的配置表中已经保存了每条虚拟链路的数据包在该节点的 deadline 时间信息 d_i^e（e 代表端系统，i 代表虚拟链路 i，d_i^e 代表虚拟链路 i 在发送端系统上分配的 deadline），定义 $d_i^e-(t_{i,o}^{e}-t_{i,u}^{e})$ 为 $t_{i,r}$（r 代表时间戳，$t_{i,r}$ 代表虚拟链路 i 在端系统上打入的时间戳）。当数据包将要离开发送端系统时刻，将时间戳 $t_{i,r}$ 加入到 AFDX 以太网包头中。然后，该数据包经过复制，在冗余的两个网络 A 和 B 上同时发送。

3.2 AFDX 交换机

为了更好的进行流量与延迟控制，本文中采用输出缓存型交换机(output buffer switch)[35]，如图 3-2 所示。由于路由模块的速度足够的快，输入缓存只需要一个包大小即可。当数据包到达交换机对应端口缓存(buffer)时，路由模块根据虚拟链路标识号将数据包转发入对应的 jitter-EDD 整流器缓存中。然后，数据包经过每个物理端口处的 EDF 调度器调度后，发送到物理链路上。

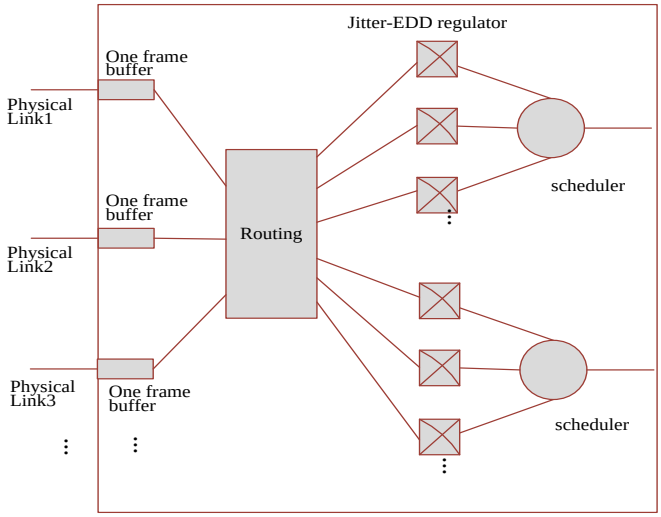

图 3-2 交换机

Fig.3-2 The Switch

定义数据包到达交换机 k 的时刻为 $t_{i,a,k}^{s}$（s 代表交换机，a 代表数据包到达该节点 k，i 代表数据包所属虚拟链路为 i），这个数据包被调度后将要离开该交换机的时刻为 $t_{i,o,k}^{s}$（s 代表交换机，o 代表数据包离开交换机 k，i 代表数据包所属虚拟链路为 i），定义该数据包所在的虚拟链路在该交换机上的 deadline 为 $d_{i,k}$(代表虚拟链路 i 在交换机 k

上分配的 deadline)。

Jitter-EDD 整流器是一种特殊的缓存器，它允许调度器在有效时间段(eligible time)内才能调度数据包，即时间间隔$[t^{S}_{i,a,k}+t_{i,r,k-1},t^{S}_{i,a}+t_{i,r,k-1}+d_{i,k}]$（i 代表虚拟链路 i，r 代表时间戳，k-1 代表第 k-1 个节点，$t_{i,r,k-1}$代表虚拟链路 i 的数据包在 k-1 个节点打入的时间戳）。jitter-EDD 整流器与发送端整流器的作用不同，它负责将到达的数据包进行重构，使数据流回到经调度器调度之前的数据流状态。交换机路由模块足够的快，比 AFDX 网络物理链路发送数据包的速率快很多，因此相对于数据包在输出缓存队列(这里指 jitter-EDD 整流器)中的等待时间来说，路由花费的时间可忽略不计。

交换机中数据包的处理过程为：首先，当数据包到达交换输入缓存模块后，经过高速路由模块，该数据包会到达对应 jitter-EDD 整流器，然后，jitter-EDD 整流器检查该数据包中的时间戳$t_{i,r,k-1}$(这个时间戳是上一跳网络节点加入的)，根据该时间戳，数据包在 jitter-EDD 整流器中停留$t_{i,r,k-1}$时间。最后，经过$t_{i,r,k-1}$时间后，对应的调度器拥有了调度该数据包的权利，可以根据 EDF 调度策略，对数据包进行调度，将新的$t_{i,r,k}=d_{i,k}-(t^{S}_{i,o,k}-(t^{S}_{i,a,k}+t_{i,r,k-1}))$添加到数据包的包头中，替换上一跳节点中添加的时间戳，并将该数据包发送到物理链路上。

如果忽略链路延迟和交换机高速路由模块引入的延迟，在任意连续的两个网络节点上，从第一个节点上数据包可被调度的时刻到第二个节点上数据包可被调度的时刻之间的时间差是一个常量值，就是该数据包所在的虚拟链路在第一个节点上做分配的 deadline。从调度器的角度来看，jitter-EDD 整流器对数据流进行了重构，在该虚拟链路上，任意两个数据包之间的时间间隔为 BAG(如果调度器能够有能力处理这些数据包)。本文的 AFDX 每条物理链路最多支持 128 条虚拟链路，因此每个物理端口对应 128 个 jitter-EDD 整流器。

3.3 AFDX 接收端系统

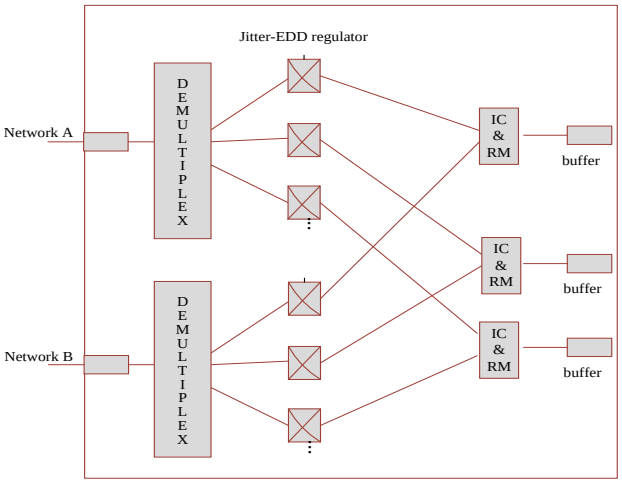

图 3-3 接收端系统

Fig.3-3 The Receiving End System

如图 3-3 所示，当一个数据帧到达接收端系统时，经过解复用器，根据该数据包的虚拟链路标识，查找配置表，将该数据包发送到对应的 jitter-EDD 整流器。接收端系统的 jitter-EDD 整流器与交换机上的 jitter-EDD 整流器是不同的，这里数据包的 eligible 时间为$[t_{i,a}^{e}+t_{i,r,n-1},+\infty]$(e 代表端系统，a 代表数据包到达端系统，i 代表数据包所属虚拟链路为 i，r 代表时间戳，n-1 代表接收端系统的上一跳交换机编号为 n-1，$t_{i,a}^{e}$代表属于虚拟链路 i 的这个数据包到达端系统的时间，$t_{i,r,n-1}$代表属于虚拟链路 i 的这个数据包在第 n-1 个交换机上打入的时间戳)。然后，数据包经过完成性检查和冗余管理之后，有效的数据包提交给应用层软件。高速解复用模块速率很快，数据包经过解复用的时间可以忽略不计。

每条虚拟链路都有两个完整性检查模块，分别进行网络 A 和网络 B 的数据包的完整性检查，然后将数据包传递给冗余管理模块，消除冗余的数据帧。完整性检查是基于序列号的，当完整性检查模块发现错误时，就会丢掉错误的数据包，并通知网络管理模块。对于每个数据包，完整性检查模块检查该数据包的序列号是否在 PSN“+”1 与 PSN“+”2 之间。PSN 代表该完整性检查模块先前收到的数据包的序列号。操作符“+”代表循环加(wrap-round add)，例如，如果 PSN=254,则 PSN”+1”为 255，PSN”+”2 为 1[9]。

像[12]中定义一样，我们定义：

$$S1"-"S2=((S1-S2+128)\bmod 256)-128$$

$$S1"<"S2\Leftrightarrow(S1"-"S2)<0$$

$$S1"="S2\Leftrightarrow(S1"-"S2)=0$$

$$S1">"S2\Leftrightarrow(S1"-"S2)>0$$

在冗余管理模块，采用轮询的策略来进行两个完整性检查缓存的数据包。完整性检查的缓存队列仅仅需要一个先进先出的队列，因为冗余管理速度很快，消耗时间可以忽略不计。冗余管理模块算法为：

Wait(*frame*)
if (((*SN*(*frame*)"＞"*PSN*(*frame*))*or*(*t* ＞ *SkewMax*)*orSN*(*frame*) == 0)
then
Re*ceive*(*frame*);
else
Discard(*frame*);
endif

SN(frame)代表当前正在被冗余管理模块处理的数据包的序列号，PSN(frame)代表冗余管理模块处理后已经被应用层软件接收的数据包的序列号。

3.4 本章小结

本章对加入 jitter-EDD 机制的 AFDX 网络进行建模，对发送端系统、接收端系统、交换机进行抽象，并详细的描述了数据包转发的过程以及 jitter-EDD 机制的具体实现细节，最后提出了新的冗余管理算法解决有效数据包丢失的问题，提高 AFDX 的容错能力。

4 AFDX 网络模型分析

本文已经在第 3 章中，对 AFDX 的网络进行了建模，这章中主要对加入了 jitter-EDD 的 AFDX 网络进行延时分析、deadline 分配、AFDX 网络的可调度性分析和容错管理算法的理论分析。

4.1 端到端的延迟分析

设定虚拟链路 i 的接收端系统和发送端系统之间共有 n-1 个交换机，如图 4-1 所示。

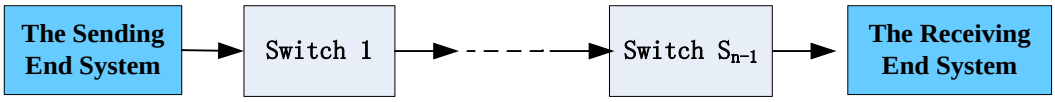

图 4-1 数据流的流动过程

Fig.4-1 The Transmitting Process of Data Flow

数据包的从发送端系统到接收端系统经历的延迟由以下部分组成：

(1) 发送端系统的发送延迟

根据第 3 章发送端系统模型，发送端系统的延迟为 $t^e_{i,o}$-$t^e_{i,u}$。

$t^e_{i,u}$ 代表虚拟链路 i 上，数据包在端系统上可被调度的时间。发送端系统有整流器，可对进入发送端系统缓存的数据包进行整流，为虚拟链路的带宽提供保证，本文从经整流器整流后数据包可被发送端系统调度器调度的时刻，开始计算端到端的延迟。

$t^e_{i,o}$ 代表虚拟链路 i 上的该数据包经过发送端系统调度，将要离开发送端系统的时刻。

(2) 交换机上引入的延迟

根据第 3 章建立的 AFDX 网络模型，在 AFDX 交换机 k (1=<k<=n-1)上引入的延迟主要包括 3 部分：

交换延迟：把输入链路的数据包转移到对应的虚拟链路的 jitter-EDD 整流器消耗的延时。

Jitter-EDD 整流器延时：由 jitter-EDD 整流器引入的延时。

调度延时：由发送端系统对应端口上的调度器引入的延时。

1) 交换延迟：交换机的延时主要由交换机路由模块引入。由于当前硬件高速的发展，中间高速路由交换模块速度很快，由高速路由交换模块的引入的时间，相对于

jitter-EDD 整流器引入的延时和调度器引入的延时来说，可以忽略不计，对端到端延时几乎没有影响。这样，为了简化延时分析，突出主要问题，因此对该延迟忽略。

2) Jitter-EDD 整流器延迟：由于 jitter-EDD 整流器使数据包的 eligible 时刻为 $[t^{S}_{i,a,k} + t_{i,r,k-1}]$，所以 jitter-EDD 整流器延迟为 $t_{i,r,k-1}$。$t^{S}_{i,a,k}$ 代表虚拟链路 i 上的数据包到达交换机 k 的时间，$t_{i,r,k-1}$ 代表在节点 k-1 上加入的时间戳。当 k-1 等于 0 时，代表这个节点是发送端系统。

3) 调度延迟：根据第 3 章建立的 AFDX 网络模型，调度引入的延迟为 $t^{S}_{i,o,k} - t^{S}_{i,u,k}$。

$t^{S}_{i,o,k}$ 代表在交换机 k 上，虚拟链路 i 的数据包将要离开发送到物理链路上的时刻。

$t^{S}_{i,u,k}$ 代表在交换机 k 上，虚拟链路 i 的数据包经过 jitter-EDD 整流器延迟，可被调度器调度的时刻。

所以在 n-1 个交换机上总共引入的延迟之和为：

$$\sum_{k=1}^{n-1} t_{i,r,k-1} + \sum_{k=1}^{n-1} (t^{s}_{i,o,k} - t^{s}_{i,u,k}) \tag{4-1}$$

$t_{i,r,o}$ 代表发送端系统在数据包头中加入的时间戳。

(3) 接收端系统延迟

完整性检查和冗余管理的延迟的时间可以忽略，所以接收端系统延迟为 $t_{i,r,n-1}$(代表第 n-1 个交换机上在虚拟链路 i 上的数据包中打入的时间戳)。

(4) 线路延迟(propagation delay)

线路延迟指数据包在物理链路上传输引入的延迟，忽略不计。

综上所述，虚拟链路 i 上的端到端的延迟为：

$$D = t^{e}_{i,o} - t^{e}_{i,u} + \sum_{k=1}^{n-1} t_{i,r,k-1} + \sum_{k=1}^{n-1} (t^{s}_{i,o,k} - t^{s}_{i,u,k}) + t_{i,r,n-1}$$

$$= d^{e}_{i} + \sum_{k=1}^{n-1} d_{i,k} \tag{4-2}$$

$d_{i,k}$ 代表虚拟链路 i，在网络节点 k 上分配的局部 deadline。D 代表端到端的延迟。

d^{e}_{i} 代表虚拟链路 i 在发送端系统上分配的局部 deadline。

通过这个公式，可以得出，在 AFDX 加入 jitter-EDD 机制后，从发动端系统数据包可调度开始，到接收端系统将数据包提交给应用层软件结束，端到端的延迟是一个常量值 D。这也就意味着，在实际的网络中，端到端的数据包的延迟抖动 jitter 非常的小。

AFDX 网络中，同一个数据包在网络 A 和网络 B 上，同时发送数据包。这样的话，

如果两个网络不对称，则会造成两个网络的数据包到达时间不同，这样造成冗余管理有效数据包丢失的问题。通过上述公式，可以得知，如果我们设定两个网络的端到端的延迟相等，这样就可以确保两个网络的数据包到达时间相同，从而避免有效数据包丢失的问题。

4.2 Deadline 分配和可调度性分析

上一节，我们对 AFDX 端到端的延时进行了详细的分析，如果我们设计虚拟链路 i 端到端的延时为 D，需要将 D 在虚拟链路上经过的节点上进行分配。在原始的 jitter-EDD 机制中[34]，作者只是简单的用平均分配延时的方法，进行端到端延时的分配。但是，虚拟链路经过的每个节点的数据吞吐量是不同的，也就是每个节点的忙碌情况是不同的，因此清闲的节点对于平均分配的节点来说，deadline 时间过剩，而对于忙碌的节点来说，deadline 的时间不足，造成这样的网络无法建立。

基于负载愈重，所需调度时间愈长的原理，我们设计一种基于权重的 deadline 分配方案，将端到端的延迟 D 分配到虚拟链路 i 所经过的 n 个节点:

$$d_{i,k} = \frac{\sum_{m \in M_{i,k}} L_m / Bag_m}{\sum_{k=0}^{n-1} \sum_{m \in M_{i,k}} L_m / Bag_m} * D \qquad (4\text{-}3)$$

$M_{i,k}$ 代表与虚拟链路 i 经过的节点 k 共用同一个调度器的所有虚拟链路的集合，0=<k<n，k 为整数。节点 0 代表发送端系统，$d_{i,0}$ 即为 d_i^e（虚拟链路 i 在发送端系统上分配的延时），节点 1 到节点 n-1 为交换机。L_m 代表虚拟链路 m 的最大包长，Bag_m 代表虚拟链路 m 的 BAG。

在实际进行 deadline 分配时，根据上面公式，在两条相互冗余的虚拟链路上的值不相等，但是由于物理的限制，每条虚拟链路的发送端系统上分配的延时必须唯一。这时，我们选取两条虚拟链路上得出的的较大值作为发送端系统上的局部 deadline，然后把剩余的延迟时间重新进行分配。

为了确保端到端延迟和抖动，必须对调度器进行可调度性分析。在原始的 jitter-EDD 机制中，deadline 保证条件为充分非必要条件，这样对网络带宽造成浪费[34]。在 AFDX 网络中，数据包的调度是非抢占式调度，[20-22]等对非抢占式调度进行了大量的研究，根据[22]中的理论，我们推出 AFDX 网络可调度的充分必要条件为：

同一个调度器上的虚拟链路组可调度的充分必要条件为：

(1) $\sum_{m \in M_{i,k}} \frac{\breve{L}_m / 100Mbps}{Bag_m} \le 1$

(2) $\forall t \in S, \sum_{m \in M_{i,k}} \lceil \frac{t - d_m}{Bag_m} \rceil * \frac{L_m}{Bag_m} + C_p \le t,$

$S = \bigcup_{m \in M_{i,k}} S_m,$

$S_m = \{d_{m,k} + j * Bag_m : j = 0,1,...,\lfloor \frac{t_{\max} - d_{m,k}}{Bag_m} \rfloor\},$

$$t_{\max} = \max\{\{d_{m,k} : m \in M_{i,k}\} \cup \frac{C_p + \sum_{m \in M_{i,k}} (1 - \frac{d_{m,k}}{Bag_m}) * \frac{L_m}{100Mbps}}{1 - \sum_{m \in M_{i,k}} \frac{L_m}{100Mbps} * Bag_m}\},$$

$C_p = \max\{L_m / 100Mbps : m \in M_{i,k}\}$

$M_{i,k}$ 代表与虚拟链路 i 经过的节点 k 共用同一个调度器的所有虚拟链路的集合，0=<K<n，k 为整数。节点 0 代表发送端系统，$d_{i,0}$ 即为 d_i^e，虚拟链路 i 在发送端系统上分配的延时，节点 1 到节点 n-1 为交换机。L_m 代表虚拟链路 m 的最大包长，Bag_m 代表虚拟链路 m 的 BAG，$d_{m,k}$ 代表虚拟链路 m 在节点 k 上分配的 deadline。

在设计一个 AFDX 网络时，首先对端到端延时进行设定（主要根据实际的需要，例如对于延迟要求比较严格的数据流，端到端的延迟设定小一些，对于延迟要求比较低的应用，延迟设定大一些，以满足实际的性能要求为准），然后进行 deadline 的分配，最后进行可调度性分析。对于不满足可调性分析的节点，分析其瓶颈，并对相应的 deadline 进行一定的调整。

4.3 冗余管理分析

AFDX 基于通信链路物理上冗余的交换网络原理，一个 AFDX 系统中有 A 和 B 两个独立的交换网络。AFDX 终端系统传输的每个数据包同时在 A 和 B 两个网络上进行发送。因此，正常情况下，每个终端系统将会收到两份同样的数据包。这样，即使网络内传输失败或数据链路失效，AFDX 系统也可以提供可靠安全的数据传输。在 AFDX 中，接收端系统先进行完整性检查(Integrity Check)，然后再进行冗余管理(Redundancy Management)。完整性检查主要对序列号进行检查，查看收到的数据包是否合法。而冗余管理对冗余帧处理，对两条冗余链路上接收到的数据包进行过滤，去除冗余帧，保证一个正确的无冗余的数据帧传到应用层。

但是当网络 A 和网络 B 中速度快的网络先后到达一个无效数据包和有效数据包后，慢速的网络的两个有效的数据包才到达时，端系统首先接收第一个到达的有效数据包，即快速网络的第二个数据包，当慢速网络的第一个有效数据包到达时，根据冗余管理先到先有效原则，这个有效的数据包被丢弃。这样就造成有效数据包丢失的问题。如图 4-2

所示，在网络 A 上，数据包 A2 丢失，A3 首先到达端系统。根据先到先有效原则，冗余管理接收 A3 数据包，丢失 B2 数据包。

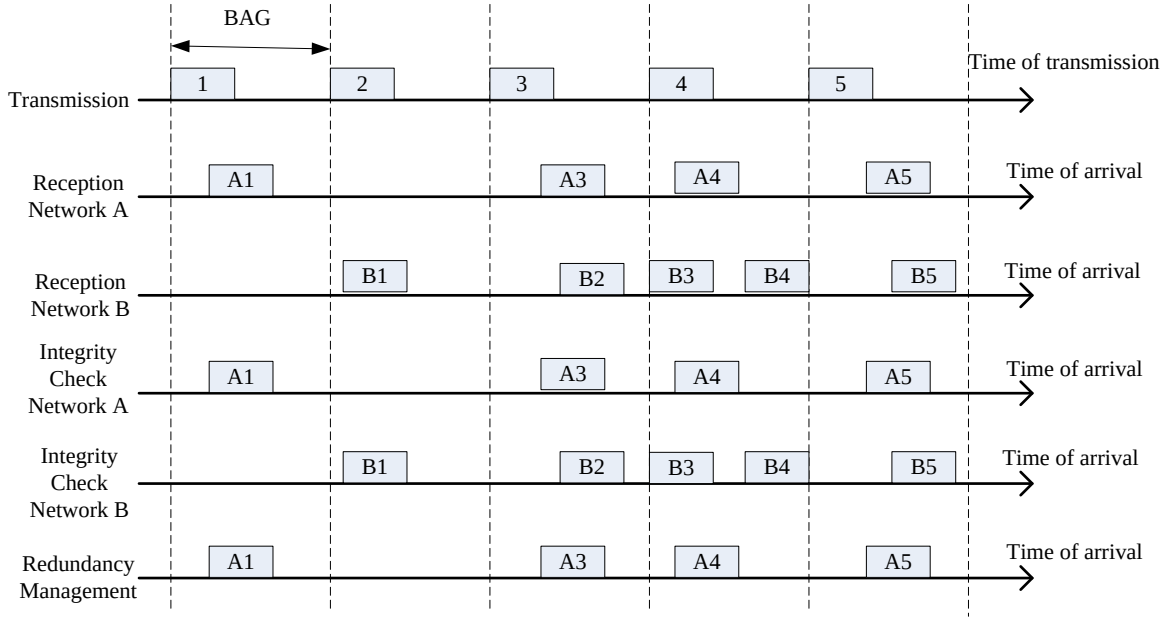

图 4-2 网络数据包丢失(网络 A 数据包 “A2”

Fig.4-2 A packet is lost on the network (frame “A2” of network A)

AFDX 网络中加入 jitter-EDD 机制后,端到端的延时抖动得到有效的控制，这样避免出现有效数据包丢失的问题，如图 4-3 所示，可见数据包 B2 能够正确的接收。

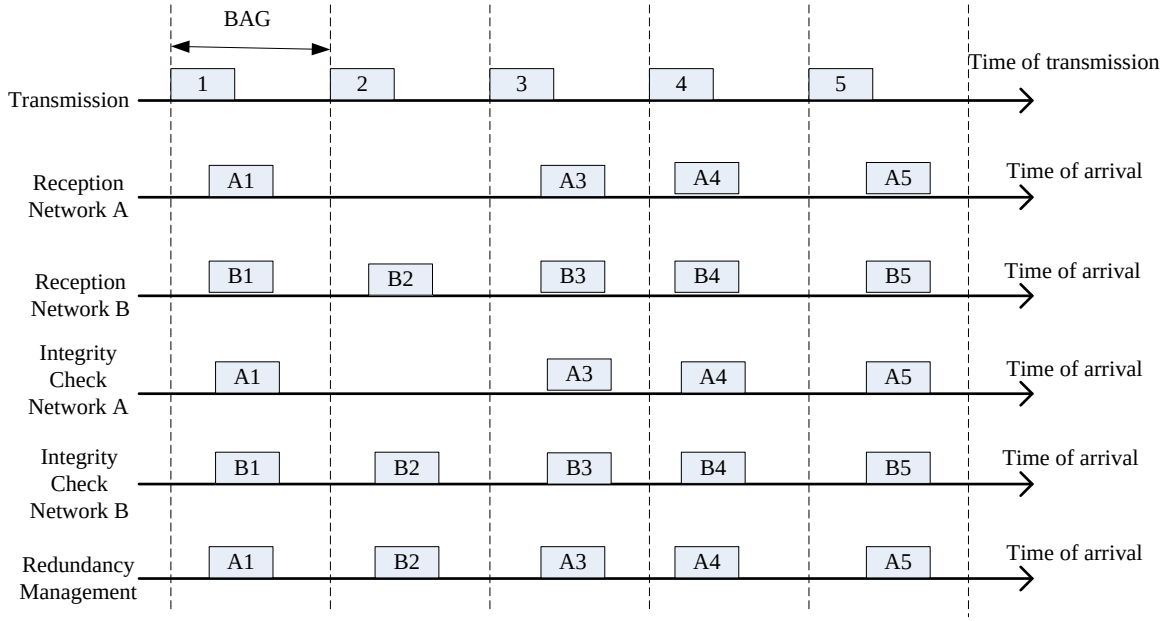

图 4-3 网络数据包丢失-jitter-EDD

Fig.4-3 A packet is lost on the network- jitter-EDD

4.4 本章小结

本章主要在第 3 章建立的模型的基础之上进行了端到端的延时的分析，并给出了基于权重的 deadline 分配方案。为了确保端到端的延时，必须确保每个 EDF 调度器上的虚拟链路的可调度性，因此我们给出了一组虚拟链路可调度的充分必要条件。本章还对第 3 章提出的冗余管理算法进行了理论上的分析。

5 AFDX 网络仿真平台的搭建

本论文研究了 AFDX 网络容错的问题，在 AFDX 中加入 jitter-EDD 机制，对数据流进行整流，对 AFDX 端到端的延迟抖动进行控制，从而解决冗余管理有效数据包丢失的问题。我们采用 SimEvents 搭建仿真平台进行仿真，本章主要对 SimEvents 进行介绍，并讲解如何使用 SimEvents 搭建交换机、端系统和如何进行错误注入以检验 AFDX 冗余管理等。

5.1 SimEvents 介绍

SimEvents[36]是 Mathworks 公司推出的最新的离散事件仿真工具。离散事件通常指受事件驱动、系统状态跳跃式变化的动态系统。SimEvents 利用队列和服务器模型来对离散系统建模，实现离散系统的仿真。SimEvents 和 Simulink 为模拟包含连续时间、离散时间和离散事件成分的混杂动态系统提供了一个综合环境。

5.1.1 SimEvents 一些基本概念

(1) 实体（entity）

实体代表离散时间仿真系统中要处理的项目，模型处理的对象不同，实体也不同。在仿真的过程中，实体能够通过队列(queues)、服务器、门(gates)和交换机。实体能够承载被称为属性的数据(attributes)。表 5-1 是一些实体的例子。

表 5-1 实体实例

Table 5-1 Entity Examples

应用场景	实体
飞机场中，一队飞机等待飞机跑道起飞	等待飞机跑道的飞机
通讯网络	等待传输的数据包、帧、消息
电梯	坐电梯的乘客
传送带	被传送的零部件
计算机操作系统	计算任务

在我们的 AFDX 网络模型中，实体指以太网数据包。

(2) 事件(event)

引起系统状态变化的行为称为事件，它是某一时间点的瞬时行为，从某种意义上来说，系统是由事件来驱动的。事件不仅用来协调两个实体之间的同步活动，还用于各个实体之间传递信息。在 SimEvents 模型中，事件产生的实例包括：

1) 实体从一模块(block)进入到另一个模块

2) 实体在某个服务器(server)上完成服务

3) 零交叉(zero crossing)信号连接到零交叉反应模块。这些事件也称为 trigger edges。

4) 函数调用(A function call)。函数调用是 Stateflow blocks 和 Simulink blocks 响应异步状态变化的所推荐的方法。

事件间的关系：

1) 某个事件必然导致另外一个事件的发生。例如：第一个实体的到达导致队列长度从 0 变为 1

2) 在一定条件下，某个事件的发生会导致另外一个事件的发生。例如：在某个服务器上，一个实体完成了服务，准备离开服务器。然而，只有当后续的模块能接收这个实体时，实体离开的事件才能发生。这个例子中，某个事件的发生可能会导致另外一个事件的发生，但是不是绝对的。

(3) 实体端口(entity ports)和路径(path)

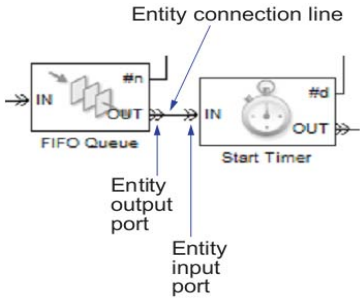

图 5-1 端口和路径

Fig.5-1 Ports and Paths

实体通过输入端口(input port)和输出端口(output port)进入和离开，路径为实体前进的路线。如图 5-1 所示,先进先出队列模块(FIFO Queue block)与开始时间模块(Start Timer block)通过路径(path)相连，进入队列的实体通过实体输出端口(Entity output port)和开始时间模块的实体输入端口(Entity input port)进入开始时间模块。仿真过程中，实体离开先进先出模块，同时进入开始时间模块。

(4) 数据(data)和信号(signals)

在基于时间的动态系统中，信号代表动态系统的输出，基于时间的模块能处理信号。实体的属性就是数据，基于时间和基于事件的系统可以通过数据来进行交互。

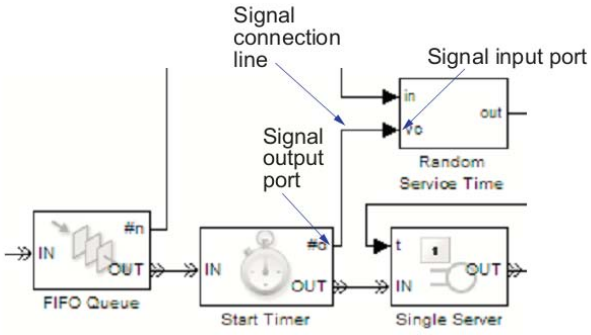

图 5-2　数据和信号

Fig.5-2 Data and Signals

从图 5-2 中，我们可以看到随机服务时间(Random Service Time)模块不仅有实体输入输出端口，还有信号输出端口。该信号端口与随机服务时间(Random Service Time)相连。

5.1.2 SimEvents 主要模块

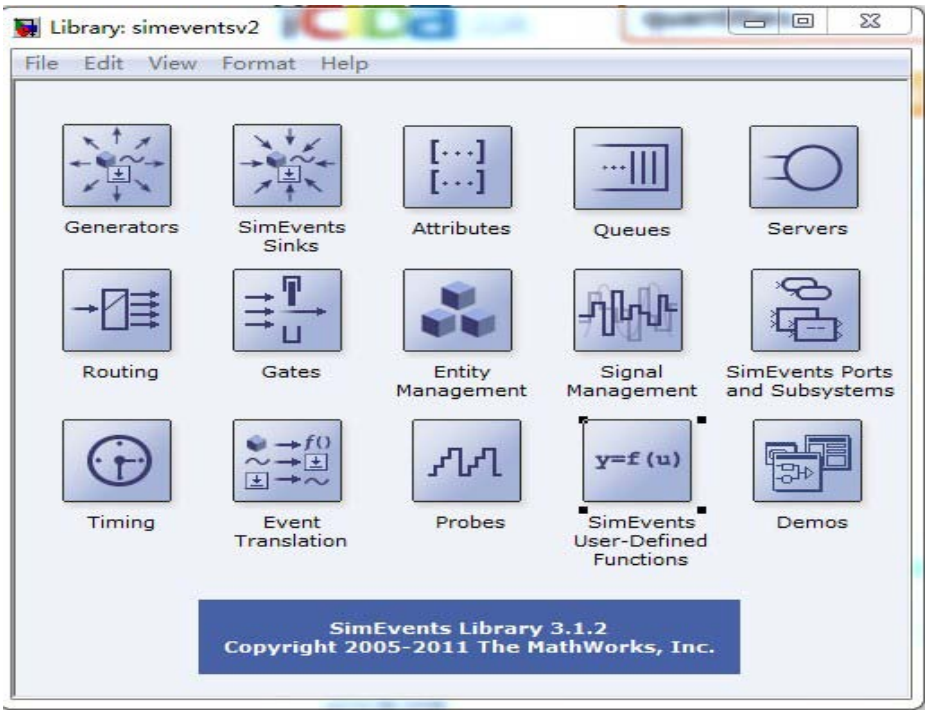

图 5-3 SimEvents 库

Fig.5-3 SimEvents Library

如图 5-3 所示，SimEvents 主要包括产生器(Generators)、SimEvents Sinks、属性、服务器、路由模块、门、实体管理、信号管理、 SimEvents 端口和子系统、时间模块、事件变换、事件探查模块(Probe)、用户自定义函数等模块。

(1) 产生器(Generator)与属性模块

产生器包括实体产生器、事件产生器、信号产生器。实体产生器分为基于事件的产生器和基于时间的产生器。基于事件的产生器，指当某个事件发生时，产生一个实体；基于时间的产生器，指当某个时间时产生一个实体。事件产生器包括基于信号的函数调用事件产生器和基于实体的函数调用事件产生器。信号产生器包括基于事件的随机数和基于事件的序列数信号产生器。

属性：实体所携带的信息，如数据包大小、deadline 信息、目的地址、虚拟链路 ID 等信息。实体与属性是进行离散仿真建模的基础。SimEvents 通过属性模块来获取和添加属性。

(2) 队列和服务模块

队列和服务模块一起为传输的实体提供存储空间和时延。利用这两个模块，我们可以做如下建模：

1) 设置存储器的大小（如交换机输入输出缓存，CPU cache、内存等）

2) 设置任务处理时间(如数据包的发送时间、job 处理时间等)

3) 实体的传输控制

4) 实体的优先级控制

如图 5-4 所示，实体经过队列和服务器进行传输。可以设定队列的长度、服务的时间和事件的优先级，这样可以对传输进行控制，对于网络拥塞等状况进行仿真。

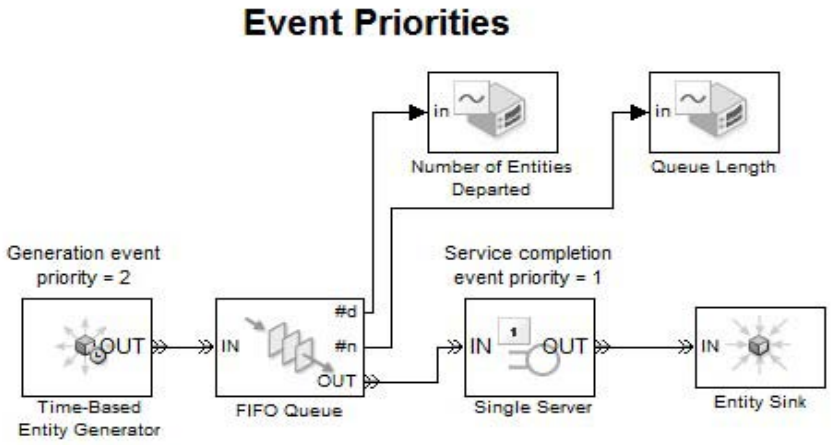

图 5-4 队列和服务器

Fig.5-4 Queue and Server

(3) 路由选择和门模块

路由选择模块包括输出交换机、输入交换机、实体复制、路径合并(Path Combiner)四个模块。输出交换机实现将单条路径上的实体发送到不同的路径上，输入交换机和路径合并模块实现实体传输路径的合并。复制实体模块实现实体的复制。门模块包括使能门(Enable Gate)和释放门(Release Gate),用来对实体的传输进行许可控制。如图 5-5 所示，

来自三个不同的通道的实体，经过输入交换机模块的路由选择之后，合并为一个通道输出。可以对路由选择策略进行控制(Round-robin、Equiprobable 等策略),实现建模仿真。

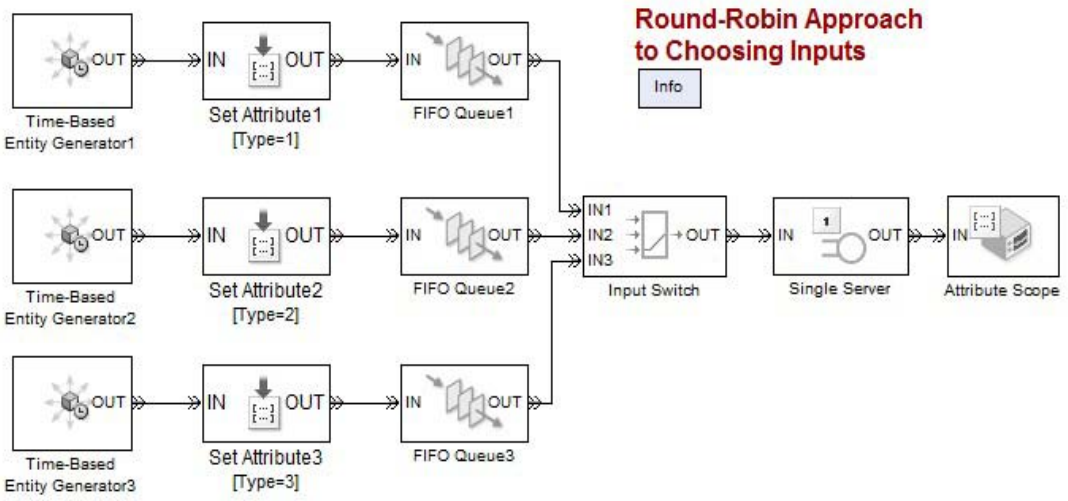

图 5-5 路由和门实例

Fig.5-5 Routing and Gate Example

(4) 事件探查模块(Probe)、SimEvents Sinks 和时间模块

这两个模块主要用于记录相关的数据信息如:

1) 端到端的延迟

2) 实体的传输时延和平均时延

3) 模块中的实体数量

4) 离开的实体数量

SimEvents 中这些模块几乎可以采集仿真过程中的各种数据，因此进行仿真调试和数据分析提供了极大的便利。

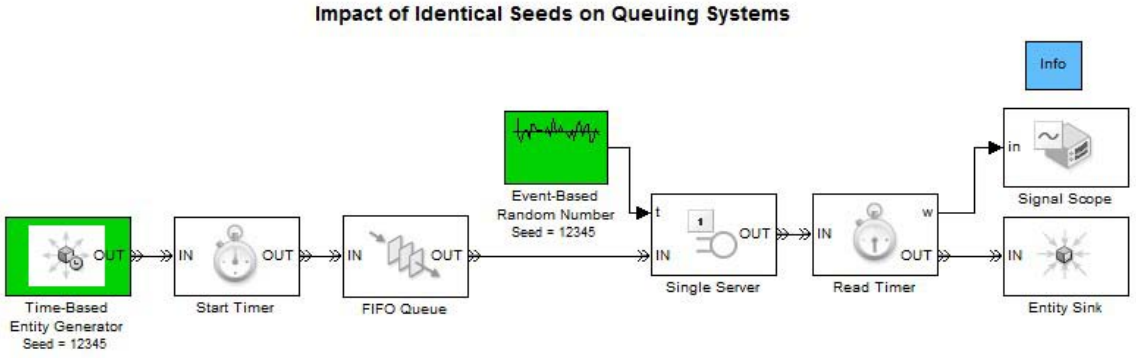

图 5-6 时间和 Sinks 模块实例

Fig.5-6 Timer and Sins Example

例如图 5-6 所示，利用 StartTimer、ReadTimer 和 Signal Scope 就可以对数据包传输的平均时延做分析。

5.2 AFDX 仿真平台搭建

根据第三章建立的 AFDX 网络模型，AFDX 主要包括发送端系统模型、接收端系统模型和 AFDX 交换机三个系统，所以需要使用 SimEvents 搭建发送端系统模型、接收端系统模型、交换机。为了验证新的冗余管理算法，需要设计错误注入仿真模块。下面将对这四个主要仿真模块的实现做详细的介绍。

5.2.1 发送端系统模型

发送系统模型主要由发送和整流器模型(Transmit and Regulator)、EDF 调度器和冗余帧复制(Frame Replicate)模块组成，如图 5-7 所示。

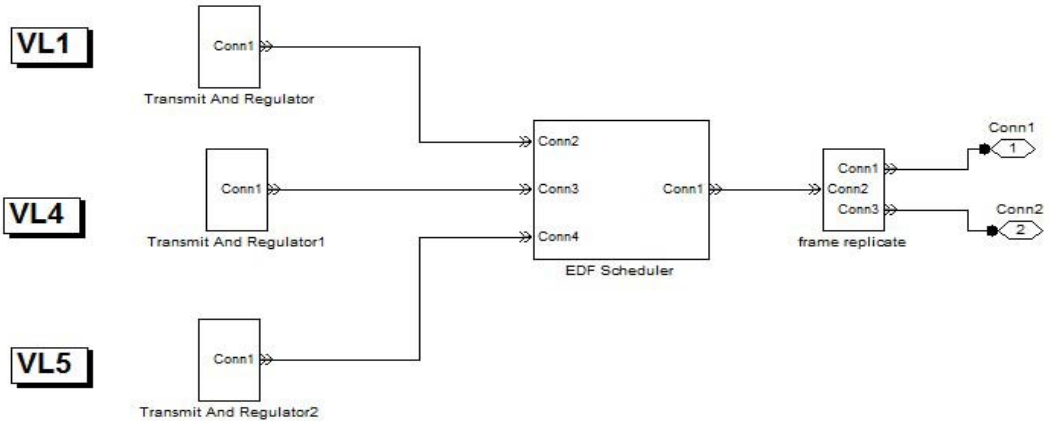

图 5-7 发送端系统 1 仿真架构图

Fig.5-7 The Simulation Structure of The Sending End System 1

端系统 1 上面有 3 个虚拟链路，VL1、VL4、VL5，传输和整流模块(Transmit and Regulator)发送 BAG 分别为 1ms、1ms、2ms 的数据包，经由 EDF 调度器进行调度。冗余帧复制（frame replicate）模块把要发送的数据包进行复制，并加入网络 A 和网络 B 的标志，将数据包发送到网络上。(为了简化仿真，对于不进行冗余网络包数据发送的端系统没有添加冗余帧复制模块)

(1) 发送和整流模型，如图 5-8 所示。

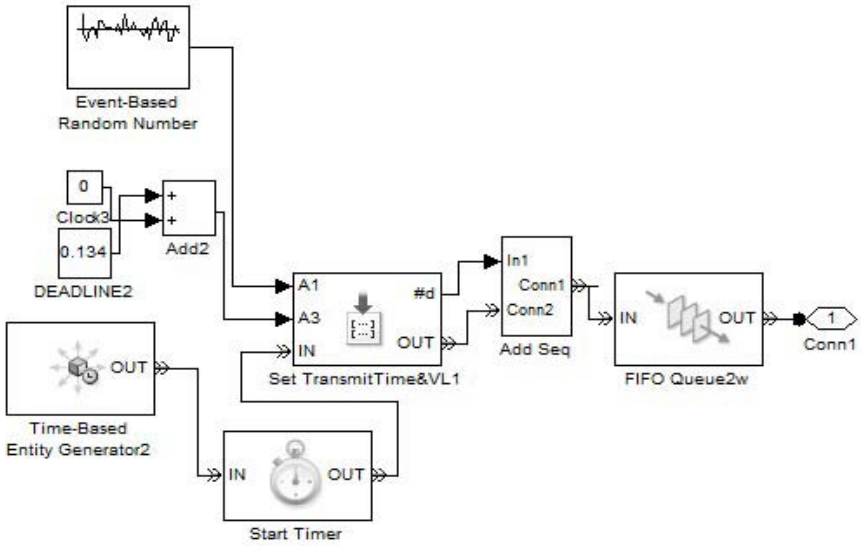

图 5-8 VL1 发送和整流模型

Fig.5-8 Transmit and Regulator

我们使用 Time-based Entity generator 来产生规整的数据流，然后添加虚拟链路 ID、节点上该虚拟链路分配的 deadline、数据包大小 (数据包大小由 Event-Based Random Number 添加) 等。

发送和整流模型中的 Add Seq 模块用于添加数据帧的序列号，具体实现如图 5-9。

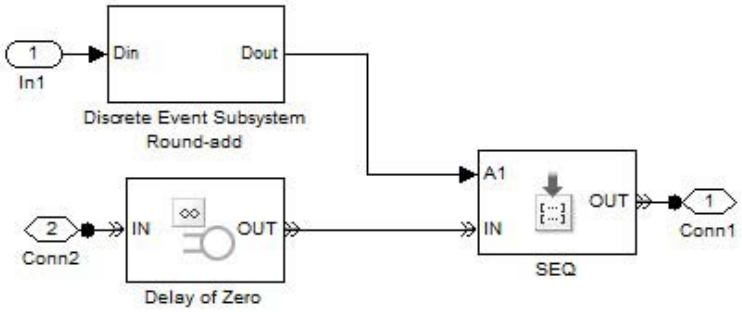

图 5-9 Add Seq 模块

Fig.5-9 Add Seq

当数据包离开 Set TransmitTime&VL 模块时，就会触发 Add Seq 模块中的离散事件子系统循环加(Discrete Event Subsystem Round-add)，利用设置属性(Set Attribute)模块 SEQ 将数据包序列号加入到数据包中。

Add Seq 模块中的离散事件子系统循环加(Discrete Event Subsystem Round-add)的内部结构如图 5-10 所示。

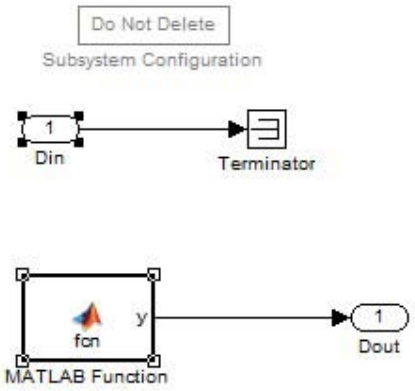

图 5-10 离散事件子系统循环加模块

Fig.5-10 Discrete Event Subsystem Round-add

当信号触发 Add Seq 模块中的离散事件子系统循环加(Discrete Event Subsystem Round-add)时，就会执行 MATLAB Function 模块。MATLAB Function 的代码为代码 5-1 所示。

代码 5-1 MATLAB Function 代码

Code 5-1 MATLAB Function

```
function y = fcn()
%#codegen
persistent a;
if isempty(a)
    a=0;
end
a=a+1;
if a>255
    a=1;
end
y=a;
```

首先定义一个静态局部变量 a，用来保存序列号的当前值。当数据包离开 Set TransmitTime&VL 模块时，就会触发触发离散事件子系统循环加(Discrete Event Subsystem Round-add)模块，然后进入 MATLAB Function 函数模块，对序列号进行循环加。序列号的值为 1 到 255,然后重新从 1 开始。

(2) EDF 调度器

EDF 调度器的具体实现如图 5-11 所示。

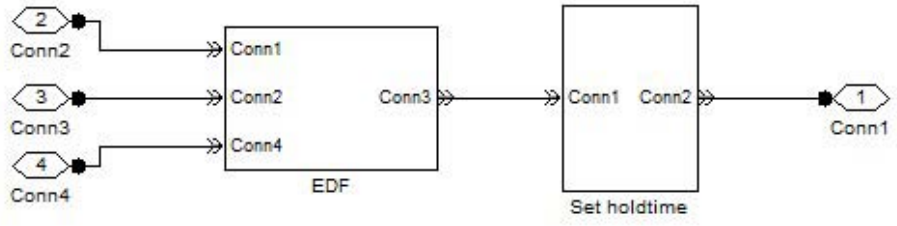

图 5-11 EDF 调度器

Fig.5-11 EDF Scheduler

EDF 调度器由 EDF 调度模块和 Set holdtime 模块组成。EDF 调度模块主要作用是实现 EDF 调度，确保数据包根据数据包 deadline 进行调度，具体实现如图 5-12 所示。

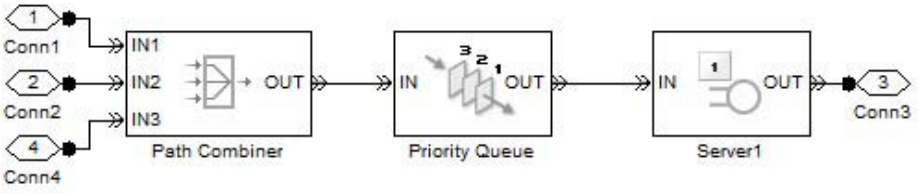

图 5-12 EDF

Fig.5-12 EDF

EDF 模块的实现过程如下：使用 Path Combiner 把数据流进行合并，使用优先级队列(Priority Queue)来实现 EDF 调度，Server1 的服务时间为数据包的传输时间。优先级队列的属性设定如图 5-13 所示，根据 deadline 进行排序，这样就可以实现 EDF 调度。

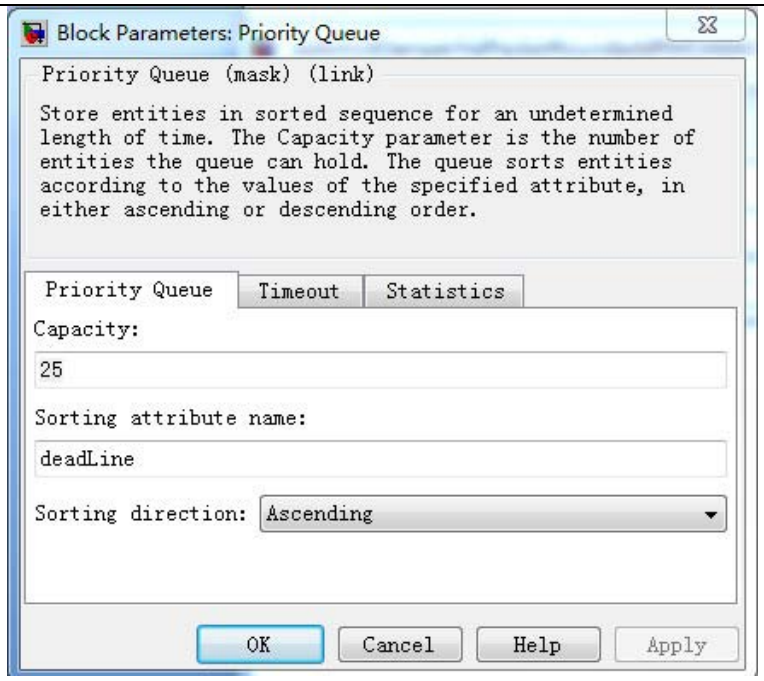

图 5-13 优先级队列属性

Fig.5-13 The Parameter of Priority Queue

EDF 调度器中 Set holdtime 模块具体实现如 5-14 所示。

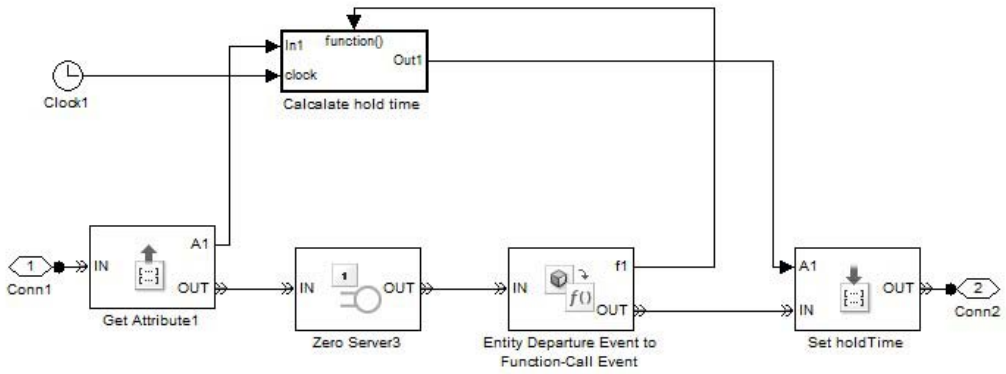

图 5-14 Set Holdtime 模块

Fig.5-14 Set Holdtime

该模块首先获取数据包的绝对 deadline 时间，然后与完成调度的时间作比较，计算 holdtime 时间，利用 Set holdTime 模块将该时间加入到数据包中。Entity Departure Event to Function-call Event 模块的配置如图 5-15 所示。

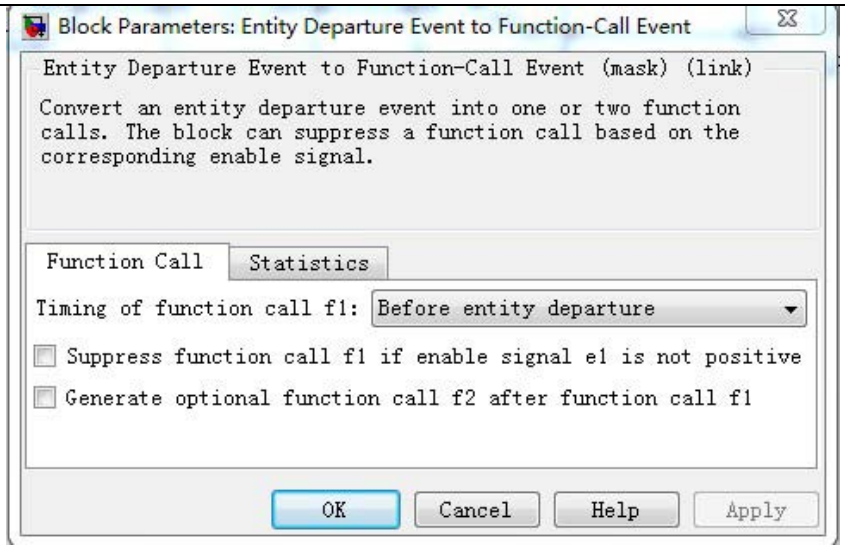

图 5-15 Entity Departure Event to Function-call Event 模块

Fig.5-15 Entity Departure Event to Function-call Event

这样，当数据包实体(Entitiy)离开 Entity Departure Event to Function-call Event 模块之前，就可以利用 calcalate holdtime 模块，计算 holdtime 时间，将该时间戳加入到数据包中。

(3) 帧复制模块

帧复制模块的具体实现如图 5-16 所示。

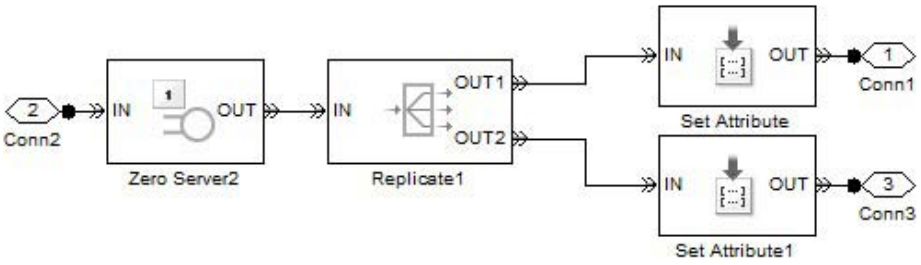

图 5-16 帧复制模块

Fig.5-16 Frame Replicate

主要是利用 SimEvents 的路由模块中的 Replicate 块实现数据包的复制，然后添加网络标志，如图 5-17 所示，network 属性，1 代表网络 A，2 代表网络 B。

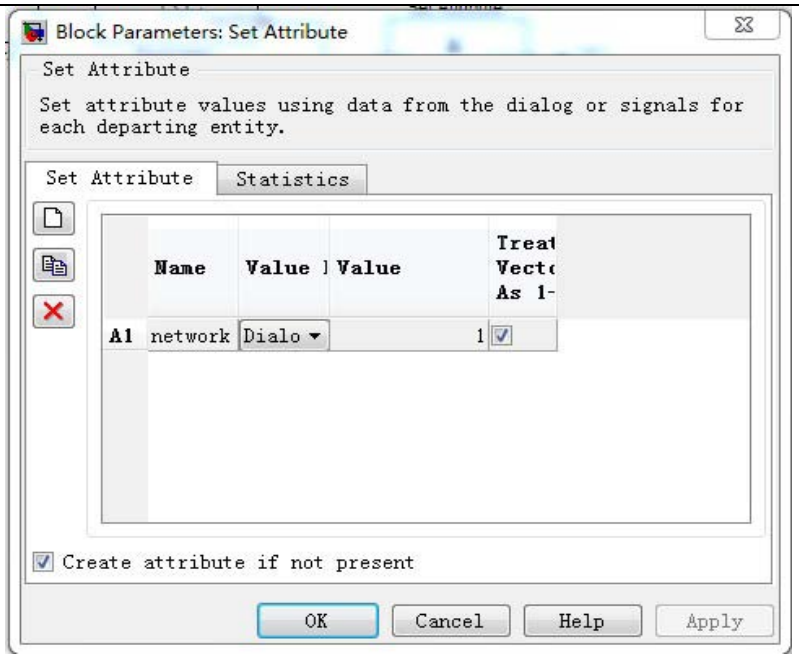

图 5-17 设置属性模块

Fig.5-17 Set Attribute

5.2.2 交换机系统模型

交换机模块主要由路由模块(Routing)、jitter-EDD 整流器模块和 EDF 调度器模块组成。路由模块负责将接入交换机的数据包经过路由转发到对应的输出端口 port 相应的 jitter-EDD 整流器中。Jitter-EDD 整流器负责将数据流进行整流，对数据流进行重构。每个端口上的 EDF 调度器实现 EDF 调度，保证端到端的延时。整体结构如图 5-18 所示。

为了仿真的简化，根据每个交换机流经的数据流的情况，对交换机网络进行微调，以减少不必要的仿真模块，加快仿真速度，所以其他交换机可能与交换机 3 稍微不同。

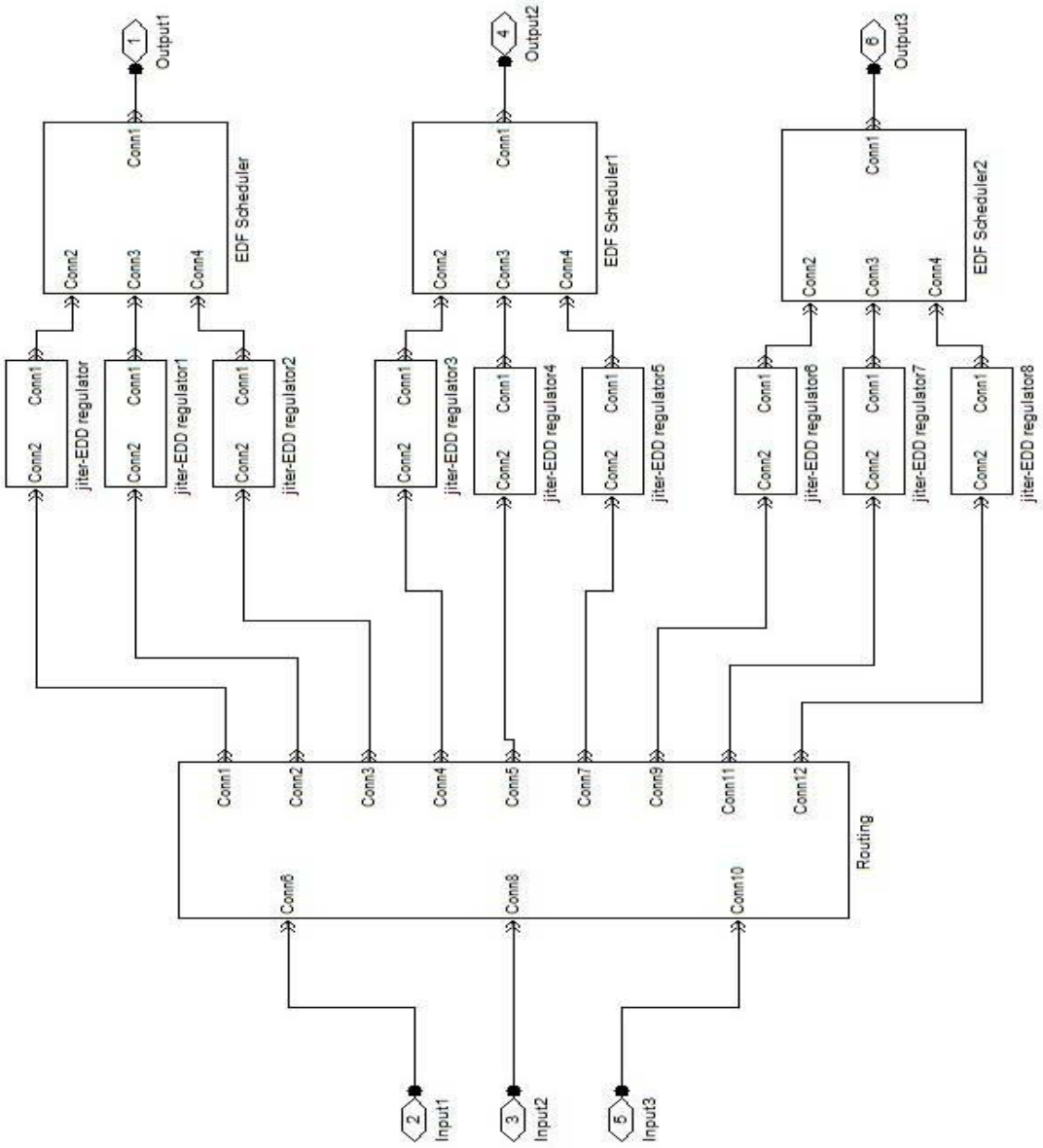

图 5-18 交换机 3 仿真结构图

Fig.5-18 The Simulation Structure of Switch 3

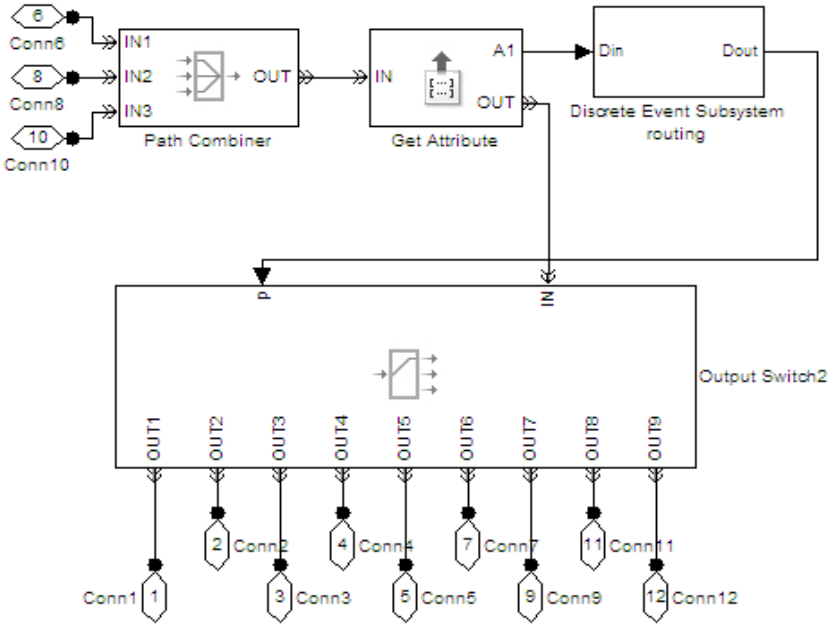

图 5-19 路由模块

Fig.5-19 Routing

(1) 路由(Routing)模块

交换机 3 共有 3 个输入端口、3 个输出端口，每个端口有 3 个 jitter-EDD 整流器，所以 Routing 模块有 9 个输出端口，将进入交换机的数据流根据虚拟链路 ID，查找配置文件，将进入交换机的数据包路由进入相应的 jitter-EDD 整流器。路由（Routing）模块的内部结构如图 5-19 所示。

代码 5-2 MATLAB Function 路由程序

Code 5-2 MATLAB Function Routing

```
function y = fcn(u)
switch u
    case 1
        y=1;
    case 4
        y=4;
    case 5
        y=3;
    case 6
        y=2;
    case 8
        y=7;
    case 9
        y=8;
    otherwise
        y=9;
end
```

其中的 Get Attribute 模块用于获取数据包的虚拟链路 ID，离散时间子系统路由模块

(Discrete Event Subsystem routing)用于根据虚拟链路 ID 对 Output Switch2 模块进行控制，控制数据包的输出端口，其触发内部的 MATLAB Function 模块，MATLAB Function 模块代码 5-2 所示。

数据包离开 Get Attribute 模块时，触发离散时间子系统路由模块(Discrete Event Subsystem routing)，调用 MATLAB Function，根据虚拟链路 ID，决定路由路径，并输出控制信号 Dout。输出交换机 2(Output Switch2)模块用于把数据包转发到对应的 jitter-EDD 整流器，为了正确的获得控制信号，输出交换机 2 的交换条件(Switching criterion)设置为 From signal port p，并设置“在交换之前，存储实体(Store entity before switching)”选项，如图 5-20 所示。

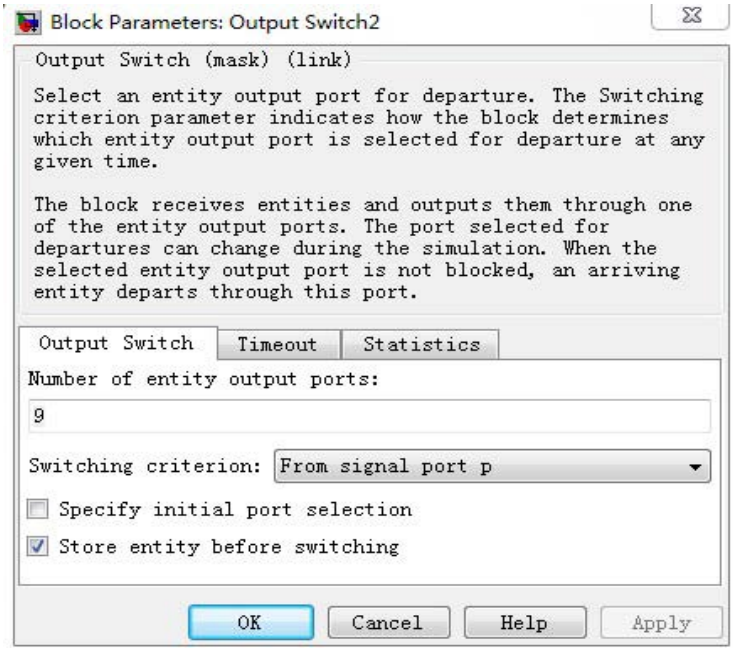

图 5-20 输出交换机 2 属性

Fig.5-20 The Parameters of Output Switch 2

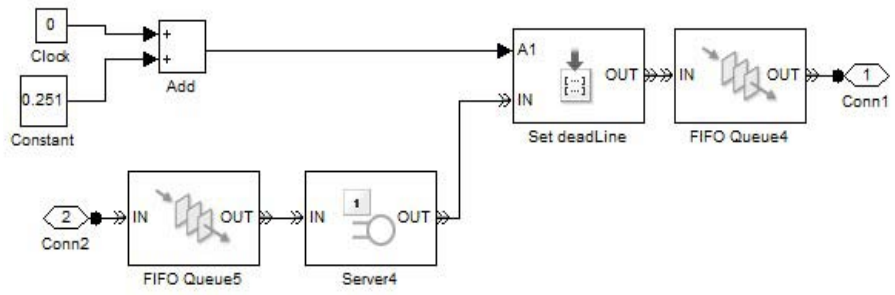

图 5-21 jitter-EDD 整流器

Fig.5-21 jitter-EDD regulator

(2) Jitter-EDD 整流器

Jitter-EDD 整流器负责将数据流进行重构，其内部结构如图 5-21 所示。

Jitter-EDD 整流器实现数据的整流，设定 Server4 的属性如图 5-22 所示。

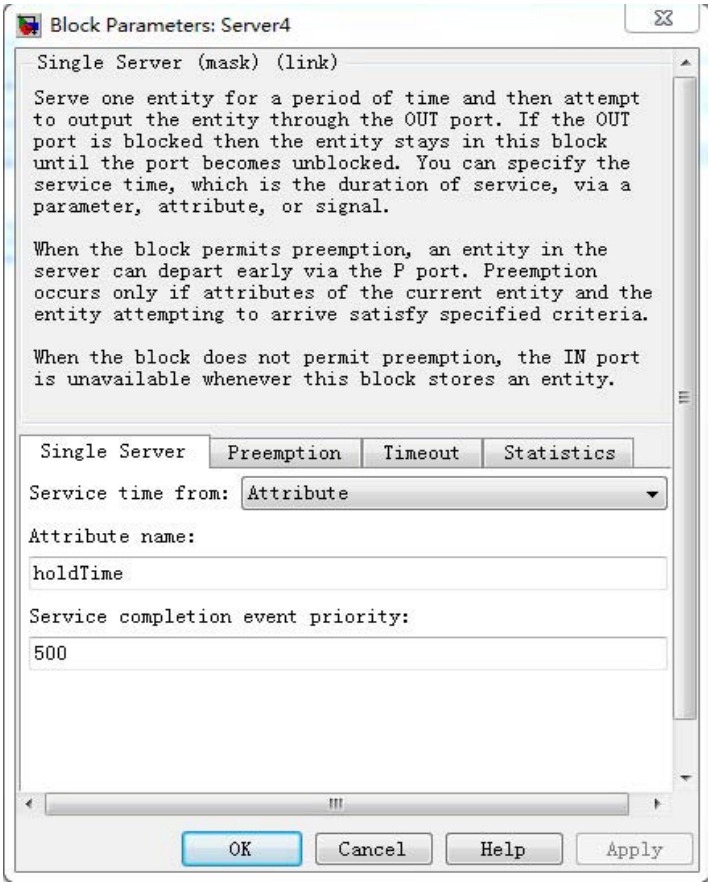

图 5-22 Server 4 属性

Fig.5-22 The Parameters of Server 4

这样，经过 Server4 后，数据流就完成了整流。然后准备开始 EDF 调度，所以在数据包中利用 Set deadline 模块，设置该虚拟链路数据包在这个交换机节点的 deadline。

(3) EDF 调度器

EDF 调度器的结构与发送端系统中的结构完全一致，请参考发送端系统中的关于 EDF 调度器的设计。

5.2.3 接收系统模型

接收端系统模型主要由 CRC 校验、jitter-EDD regulator 模块、AFDX 冗余模块、应用层(Application)模块组成，如图 5-23 所示。

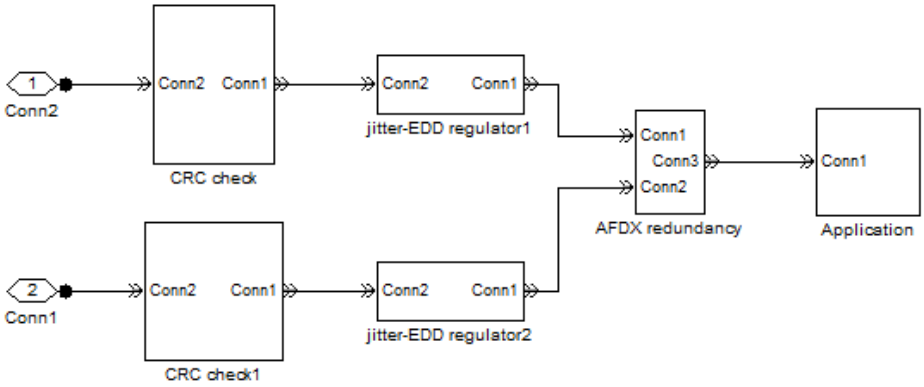

图 5-23 接收端系统 12

Fig.5-23 The Receiving End System

CRC 错误检测模块进行 CRC 和数据长度等数据包错误的检测。jitter-EDD regulator 与交换机上的 jitter-EDD 整流器有一定的区别，具体查看第 3 章接收端系统模型一节。AFDX 冗余(AFDX redundancy)模块进行数据的完成性检查和冗余管理，消除冗余的数据包，应用层模块(Application)代表航空子系统应用程序（端系统接收到的数据包最终提交给对应的应用程序，并进行处理）。

(1) CRC 错误检测模块

CRC 错误检测模块的内部结构如图 5-24 所示。

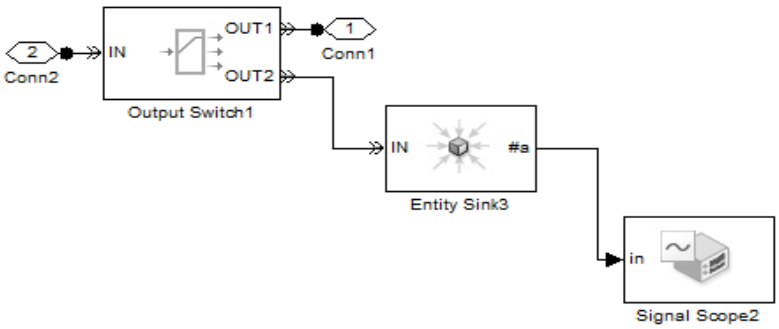

图 5-24 CRC check1 的内部结构

Fig.5-24 The Structure of CRC Check1

Output Switch1 的属性如图 5-25。

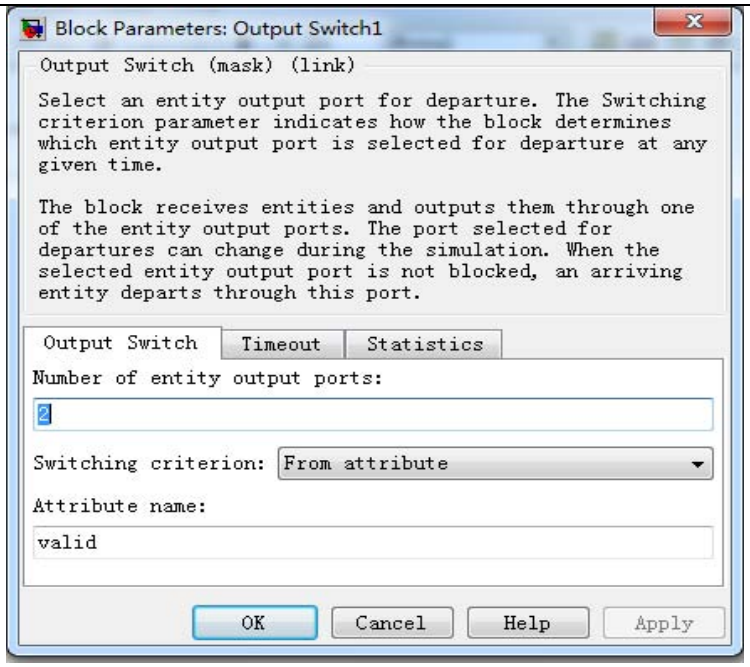

图 5-25 输出交换机 1 属性

Fig.5-25 The Parameters of Output Switch1

通过设定 CRC 错误检测模块中的输出交换机(Output Switch)的属性，就可以检测出数据包长度或 CRC 错误的数据包，并丢弃到端口 2 处。Signal Scope2 可以对错误数据包进行监测和统计。

(2) Jitter-EDD 整流器

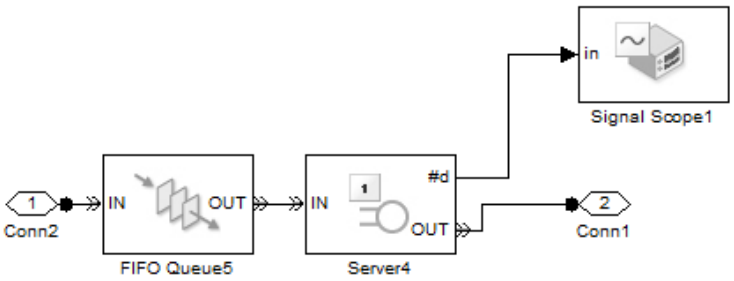

图 5-26 Jitter-EDD 整流器

Fig.5-26 Jitter-EDD Regulator

整流器的结构如图 5-26 所示，与交换机内部的 jitter-EDD 整流器比较，该模块去掉了 deadline 注入机制。这是因为接收端系统是数据目的地，所以后续不需要调度，没有队列等待延时，所以去掉 deadline 的配置。

(3) AFDX 冗余模块

AFDX 冗余模块负责去掉多余的数据包，将两条冗余的数据流进行整合成一条有效的数据流，该模块包括两部分，完整性检查和冗余管理。

完整性检查模块结构如图 5-27 所示。完整性检查模块检查收到的数据包的序列号是否满足要求，由离散事件子系统序列号检查模块(Discrete Event Subsystem sequence Check)实现，其内部具体实现代码为代码 5-3 所示。

代码 5-3 完整性检查代码

Code 5-3 Integrity Check

```
function y = fcn(u)
persistent a;
if isempty(a)
    a=0;
end
c=a+1;
if(c>255)
    c=c-256;
end
d=a+2;
if d>255
    d=d-256;
end
if u==c||u==d
    a=u; y=2;
else
    y=1;
end
```

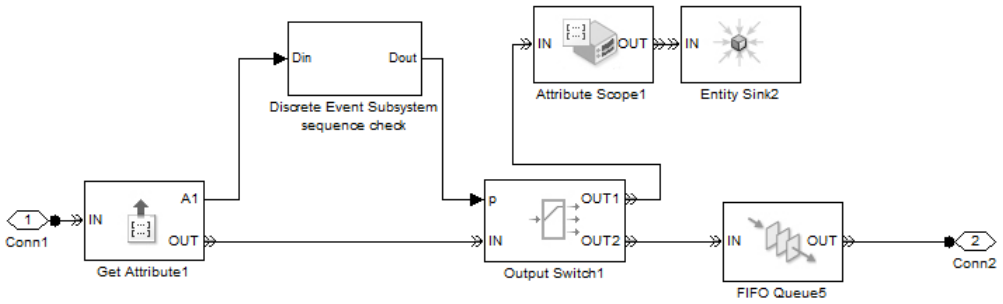

图 5-27 完整性检查

Fig.5-27 Integrity Check

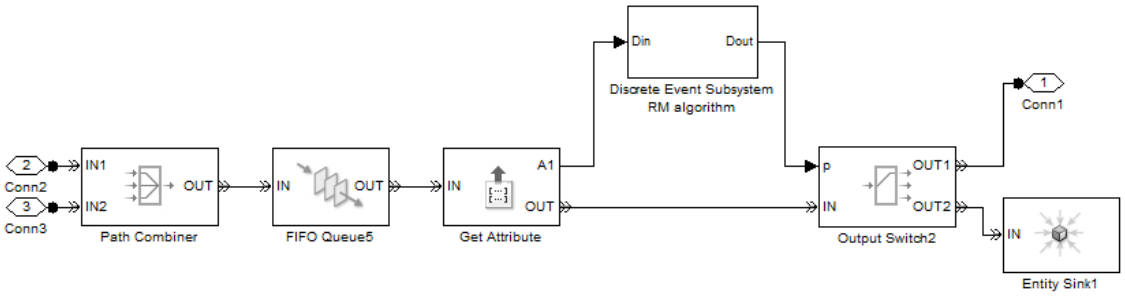

图 5-28 冗余管理

Fig.5-28 Redundancy Management

冗余管理模块负责消除多余的数据包，模块结构如 5-28 所示。首先，通过 Path

Combiner 模块将网络 A 和网络 B 的数据流进行合并成一条数据流，其次，通过 Get Attribute 模块获取数据包的序列号，触发离散事件子系统冗余管理算法模块(Discrete Event subsystem RM algorithm)进行序列号的判断，进而控制输出交换机模块(Output Switch2)的输出选择，将无效的数据包通过端口 2 输出，有效的数据包通过端口 1 发送到航空子系统应用程序。

5.2.4 错误注入模块

为了对网络的安全性进行验证，我们使用错误注入的方法，合理注入各种错误，进而分析网络的安全性。AFDX 中注入的错误包括序列号错误、数据包错误、数据包丢失错误等。

错误注入模块包括两个模块，CRC 和包错误注入模块、网络 babbling 错误注入模块。CRC 和包错误注入模块结构如图 5-29 所示，网络 babbling 错误注入模块结构如图 5-30 所示。

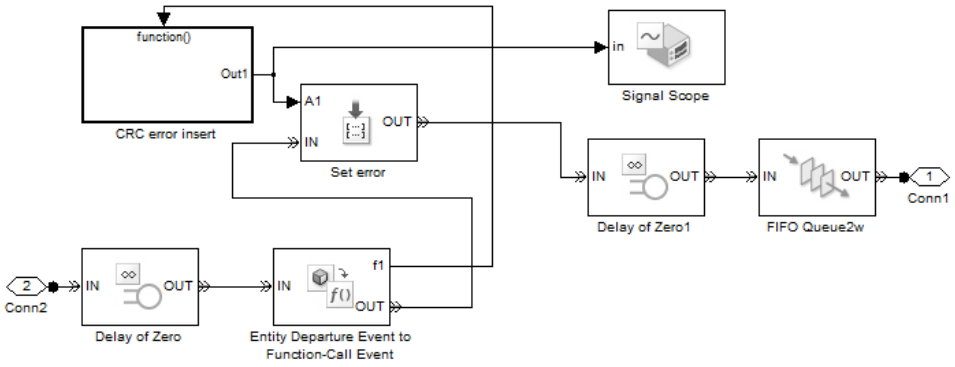

图 5-29 包错误注入模块

Fig.5-29 Packet Error Injection

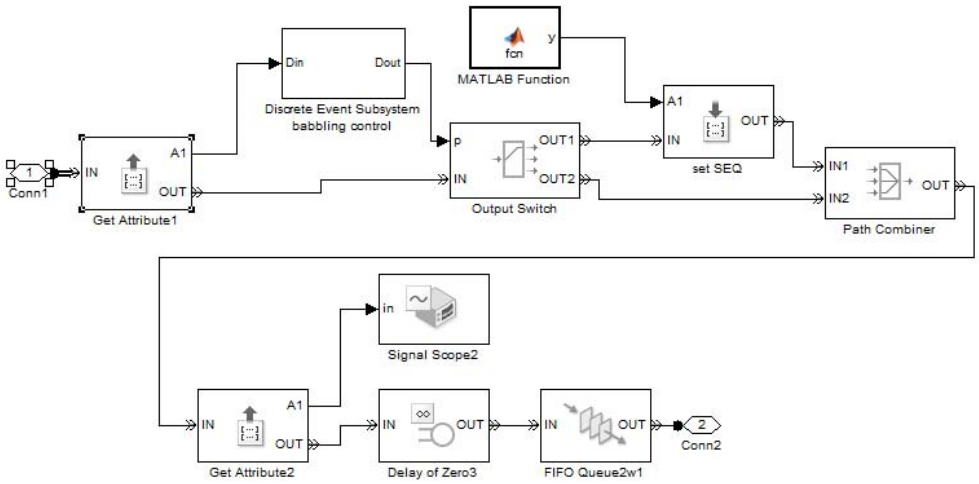

图 5-30 网络混乱错误注入

Fig.5-30 Network Babbling Injection

网络 babbling 错误注入模块的实现原理为：离散事件子系统 babbling 控制模块(Discrete Event Subsystem babblingcontrol)控制何时产生随机的网络 babbling 和产生网路 babbling 的随机的时间长度，此时控制输出交换机的出口为 1，MATLAB function 模块产生网络 babbling 时的数据包随机的序列号，通过 set SEQ 模块更改数据包的序列号。离散事件子系统 babbling 控制模块(Discrete Event Subsystem babblingcontrol)中的模块代码为代码 5-4 所示。

代码 5-4 网络混乱注入代码

Code5-4 Network Babbling Injection Code

```
function y = fcn()
persistent a,i,j;
if isempty(a)
    a= floor(rand*1300);
end
if isempty(i)
    i=0;
end
if isempty(j)
    j=10;
end
i=i+1;
if (i==a)
    if(j>0)
        y=1; i=i-1; j=j-1;
    else   y=2;
    end
 else   y=2;
end
```

5.3 本章小结

本章对仿真工具 SimEvents 中的一些基本概念与 SimEvents 的主要模块进行了介绍。SimEvents 基本概念包括实体、事件、实体端口、路径、数据和信号等。SimEvents 主要功能模块有产生器、属性、服务器、路由模块、实体管理、信号管理、子系统等。本章还介绍了 AFDX 网络仿真中接收端系统、发送端系统、交换机和错误注入的架构原理，并详细介绍了数据包的发送,整流、数据包序列号添加、EDF 调度器、jitter-EDD 机制、路由控制、CRC 和包错误注入、网络 Babbling、CRC 校验、完整性检查、冗余管理等模块的具体实现。

6 实验与仿真结果

6.1 仿真场景

我们建立了一个由 9 个交换机、13 个端系统和 9 条虚拟链路组成的平台，如图 6-1 所示。虚拟链路 1、4、5 在两个相互冗余的网络 A 和网络 B 上发送数据包，其他虚拟链路在一个网络上发送数据帧。表格 6-1 为各个虚拟链路的配置表。

我们使用 4.2 提出的基于权重的 deadline 分配分配方法，对 deadline 进行分配，得到各个交换机和端系统上的局部 deadline，如表 6-2 所示。

由于在接收端系统上没有局部的 deadine 分配，所以虚拟链路在接收端系统 6、7、8、9、10、11、12、13 上分配的 deadline 为 0，表 6-2 中省略了这些数据。

根据 4.2 节的可调度性分析方法，我们得知系统是可调度的，因此端到端延时是可以保证。

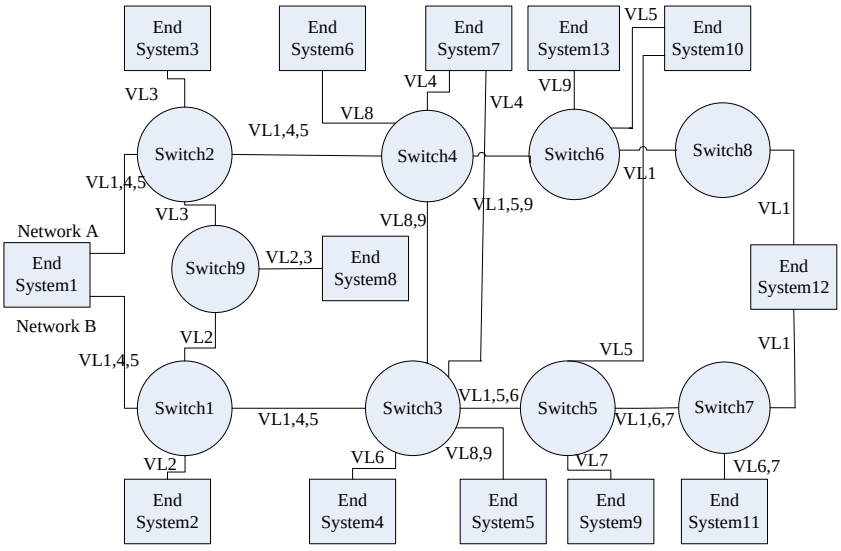

图 6-1 仿真平台拓扑

Fig.6-1 The Network Topology of Simulation Platform

表 6-1 虚拟链路配置表

Fig.6-1 The Configuration of The Virtual Links

Virtual Link	BAG (ms)	end-to-end delay(us)	frame size(byte)	Source	Destination
1	1	500	125	ES1	ES12
2	2	150	250	ES2	ES8
3	1	450	313	ES3	ES8
4	1	300	125	ES1	ES7
5	2	400	188	ES1	ES10
6	2	800	250	ES4	ES11
7	4	1000	500	ES9	ES11
8	8	1000	475	ES5	ES6
9	4	1200	500	ES5	ES13

表 6-2 Deadline 分配表

Table6-2 Deadline Distribution Table

deadline(us)	VL1	VL2	VL3	VL4	VL5	VL6	VL7	VL8	VL9
Switch1	106	50		127	122				
Switch2	134		150	127	122				
Switch3	106			46	123	251		412	290
Switch4	134			46	123			176	454
Switch5	115				33	274	50		
Switch6	49				33				166
Switch7	39					183	333		
Switch8	49								
Switch9		50	150						
End System1	134			127	122				
End System2		50							
End System3			150						
End System4						92			
End System5								412	290
End System9							167		

6.2 仿真结果及其分析

Jitter-EDD 机制减少了端到端延时的抖动，但是可能增加端到端的延时。下图中，图 6-2 到 6-5 是网络中没有加入 jitter-EDD 延迟抖动控制的网络的延迟的结果。图 6-2 代表虚拟链路 1 网络 A 上的延迟结果，图 6-3 代表虚拟链路 1 网络 B 上的延迟结果，图 6-4 代表虚拟链路 4 网络 B 的延迟。

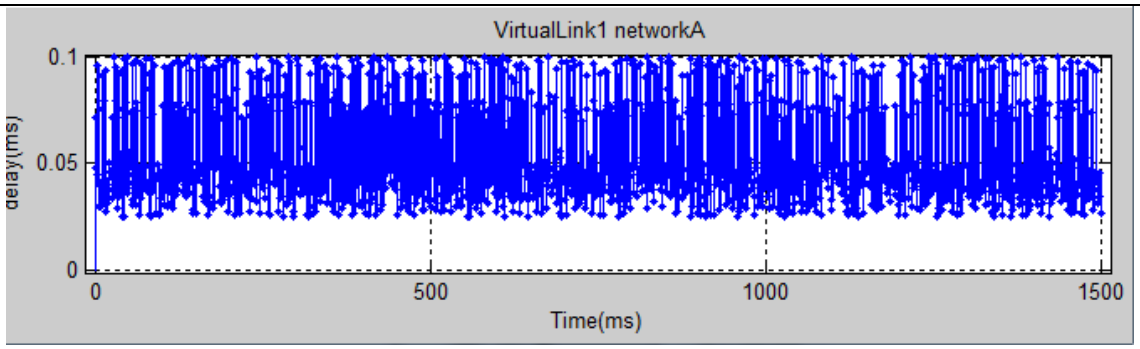

图 6-2 虚拟链路 1 网络 A 延迟

Fig.6-2 The Delay on Network A of Virtual Link 1

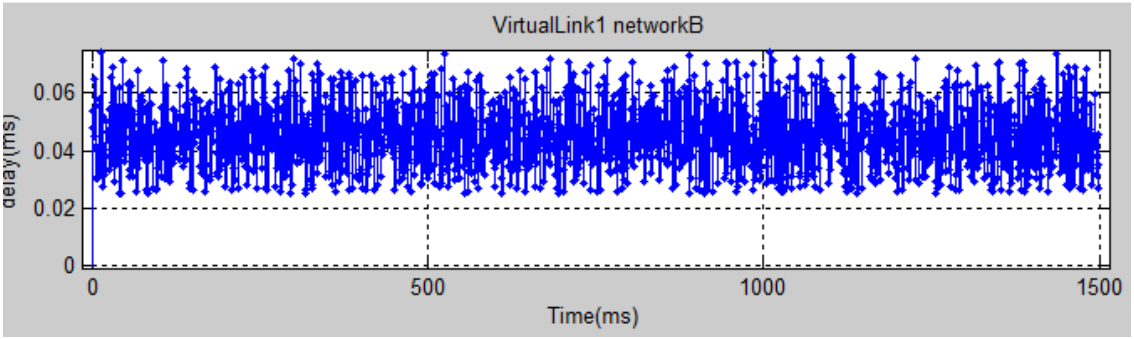

图 6-3 虚拟链路 1 网络 B 延迟

Fig.6-3 The Delay on Network B of Virtual Link 1

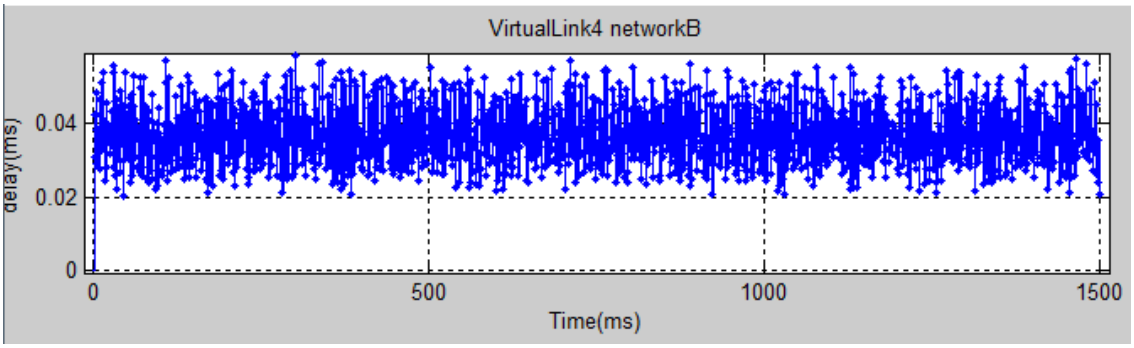

图 6-4 虚拟链路 4 网络 B 延迟

Fig.6-4 The Delay on Network B of Virtual Link 4

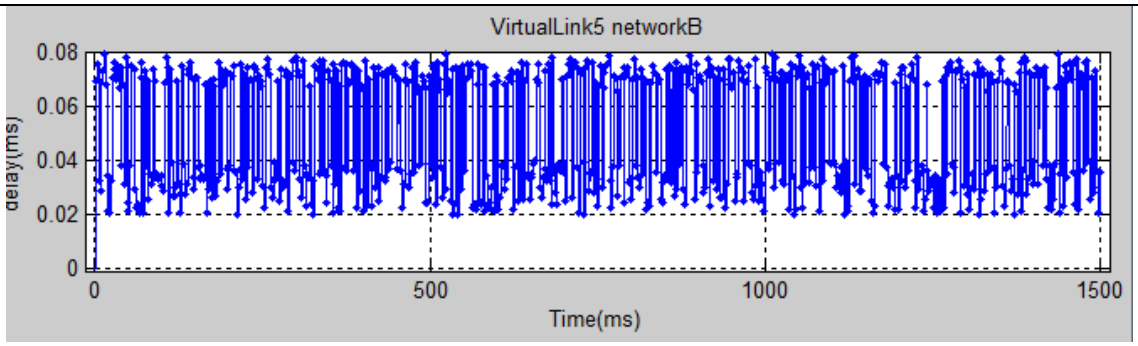

图 6-5 虚拟链路 5 网络 A 延迟

Fig.6-5 The Delay on Network A of Virtual Link 5

从图 6-2 和图 6-3 可以看到，由于虚拟链路 1 网络 A 和网络 B 经过的拓扑和网络节点的不同，导致虚拟链路 1 网络 A 和网络 B 上的延迟不一致，并且延迟抖动较大(虚拟链路 1 网络 A 的延迟分布在 0.03ms 到 0.1ms 之间，而虚拟链路 1 网络 B 的延迟分布在 0.02ms 到 0.08ms 之间)。当两个网络延迟间隔 SkewMax 较大时，会造成有效数据包丢失的问题。从图 6-2 到图 6-5 可以看到，不同虚拟链路的端到端的延迟是不同的(虚拟链路 4 网络 B 的延迟为 0.02ms 到 0.06ms，虚拟链路 5 网络 A 的延迟为 0.02ms 到 0.08ms 之间)，数据包端到端的延迟剧烈抖动。

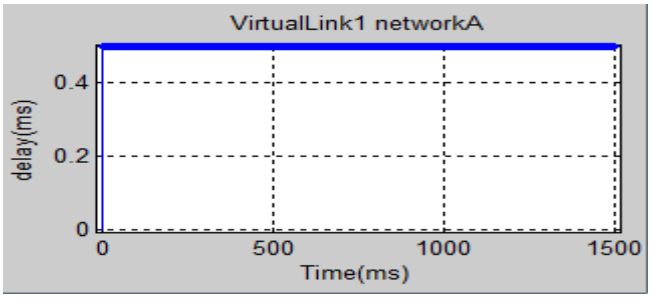

图 6-6 有 jitter-EDD 时虚拟链路 1 网络 A 延迟

Fig.6-6 The Delay on Network A of Virtual Link 1 with jitter-EDD

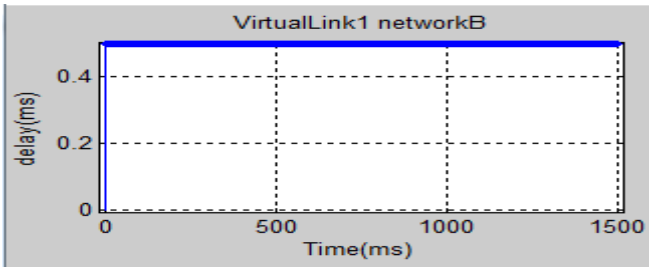

图 6-7 有 jitter-EDD 时虚拟链路 1 网络 B 延迟

Fig.6-7 The Delay on Network B of Virtual Link 1 with jitter-EDD

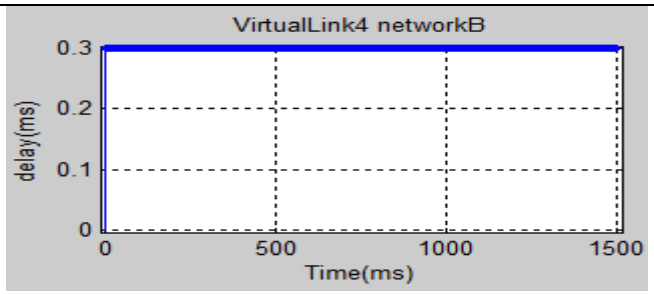

图 6-8 有 jitter-EDD 时虚拟链路 4 网络 B 延迟

Fig.6-8 The Delay on Network B of Virtual Link 4 with jitter-EDD

图 6-6 到图 6-8 为加入 jitter-EDD 抖动控制机制后，端到端延迟的结果图。图 6-6 为虚拟链路 1 网络 A 上的端到端的延时，为 0.5ms。图 6-7 为虚拟链路 1 网络 B 上的端到端的延时，为 0.5ms。图 6-8 为虚拟链路 4 网络 B 的上端到端的延时，为 0.3ms。可以看到，加入 jitter-EDD 机制后，可以对端到端的延时和抖动进行控制，使同一条虚拟链路上网络 A 和网络 B 的延迟达到一致，如图 6-6 和图 6-7 所示。将图 6-6 到图 6-8 与图 6-2 到图 6-5 比较，可以看到，没有使用 jitter-EDD 机制的虚拟链路 1 和虚拟链路 4 上的延迟要比使用了 jitter-EDD 机制的虚拟链路 1 和虚拟链路 4 的延迟小，但是使用了 jitter-EDD 机制的虚拟链路 1 和虚拟链路 4 上的端到端延迟抖动要比没有使用 jitter-EDD 机制的虚拟链路 1 和虚拟链路 4 的端到端的延迟抖动小。这对于解决 AFDX 有效数据包丢失的问题非常的有效。

Jitter-EDD 机制可以对数据流进行整流，优化数据流，如图 6-9 所示。

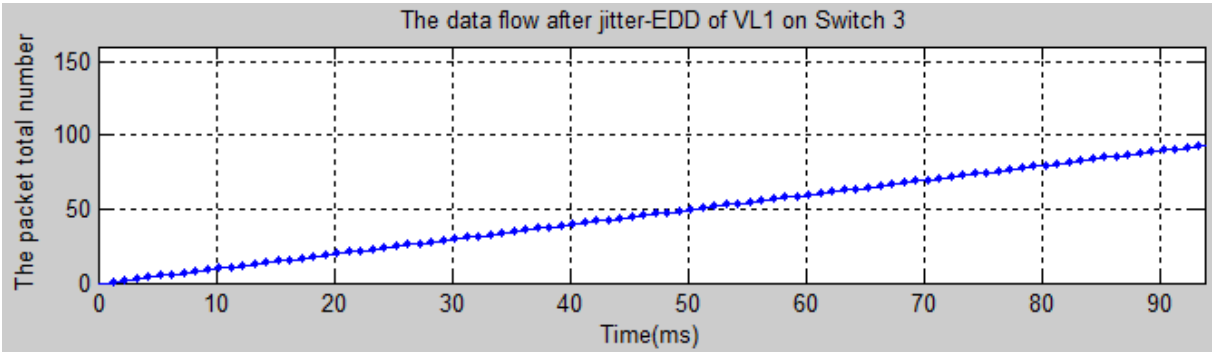

图 6-9 经 jitter-EDD 整流的数据流

Fig.6-9 The data flow after jitter-EDD regulator

图 6-9 显示的是虚拟链路 1 经过交换机 3 上虚拟链路 1 对应的 jitter-EDD 整流器后，数据流的波形图，横坐标为时间，纵坐标为数据包到达的总数，可见数据包随时间线性增长，数据流规整的到达。和普通的网络数据流相比较，经过 jitter-EDD 整流器整流的数据流明显的减少了数据突发，获得了良好的网络控制效果，能够显著减少网络所需 buffer 的大小，避免由于数据突发而导致的数据包的丢失。

下面介绍加入 jitter-EDD 机制后， 对于冗余管理的问题的影响。对于虚拟链路 1，网络 A 和网络 B 中注入 CRC 校验错误和包长错误到网络中，得到结果如图 6-10 到图 6-14 所示。图 6-10 到图 6-11 中，横坐标代表时间，纵坐标为 1 时，代表数据包有效，为 2 时，代表数据包无效。

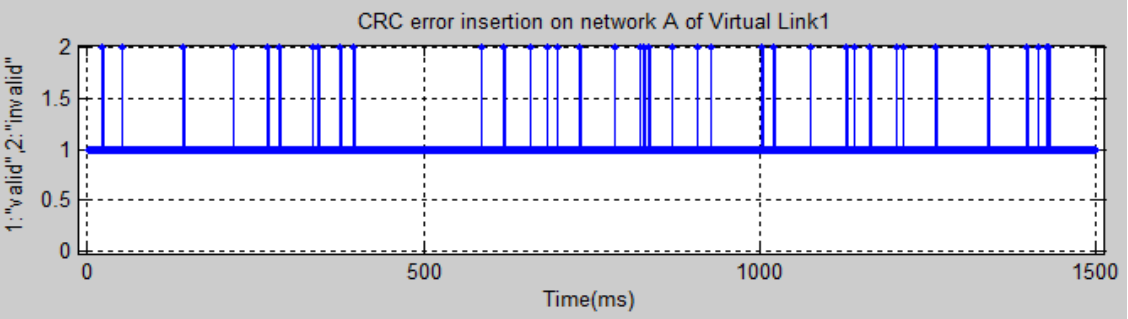

图 6-10 虚拟链路 1 网络 A 错误的注入

Fig.6-10 Error Injection on Network A of Virtual Link 1

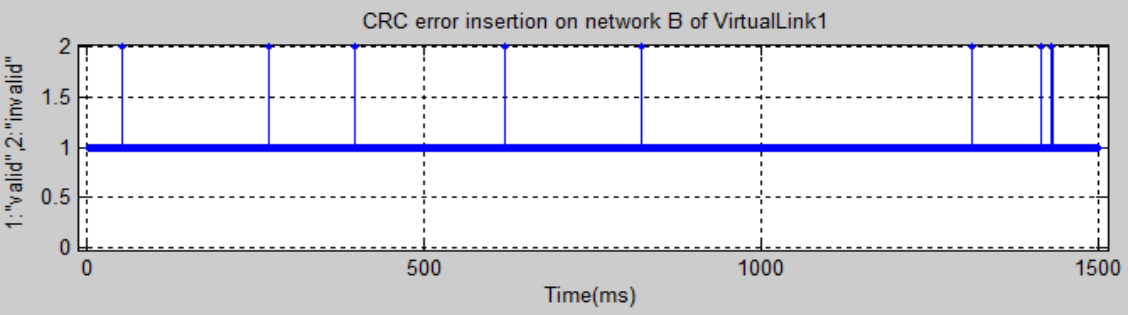

图 6-11 虚拟链路 1 网络 B 错误的注入

Fig.6-11 Error Injection on Network B of Virtual Link 1

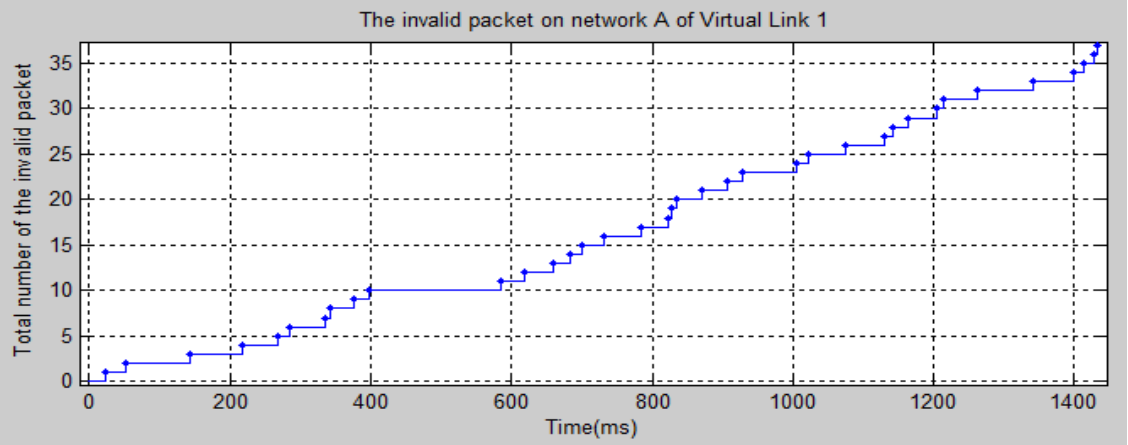

图 6-12 网络 A 上丢失的数据包

Fig.6-12 The Lost Packets on Network A

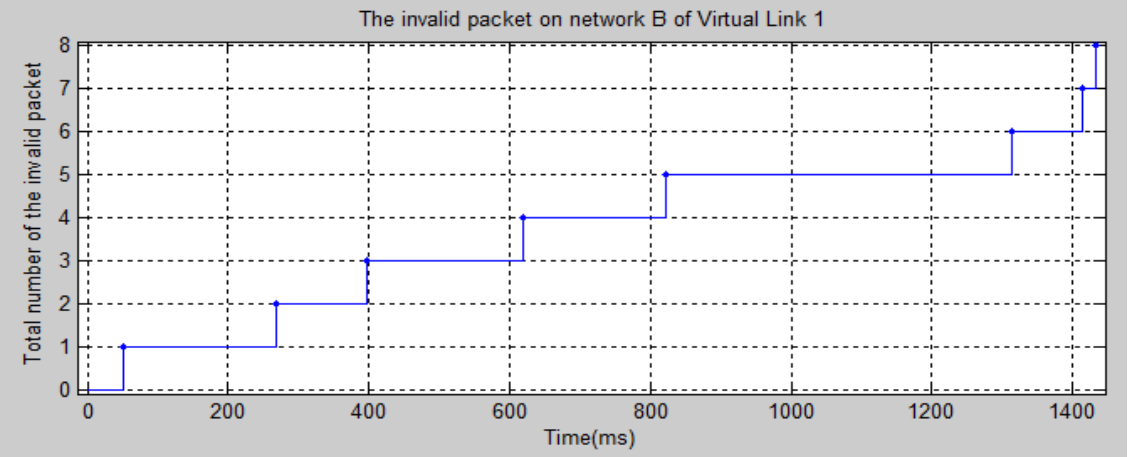

图 6-13 网络 B 上丢失的数据包

Fig.6-13 The Lost Packets on Network B

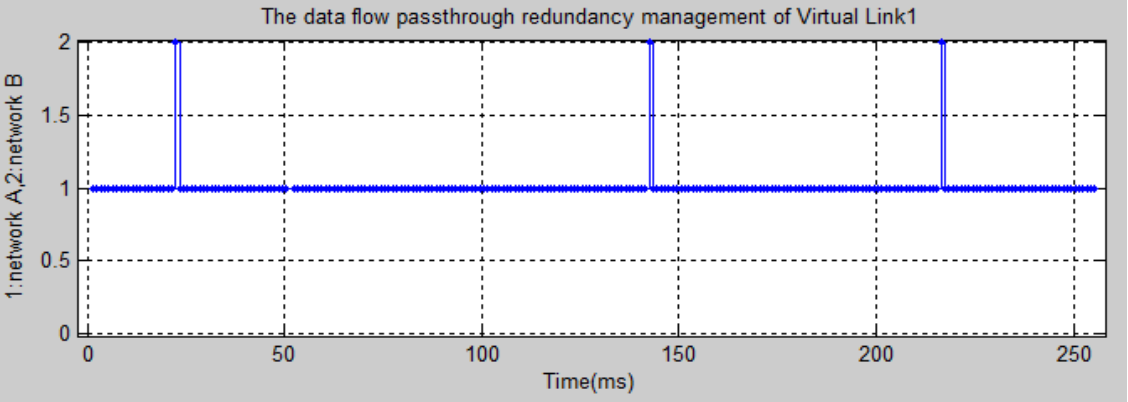

图 6-14 经过冗余管理后的数据流

Fig.6-14 The Data Flow after Redundancy Management

图 6-10 代表虚拟链路 1 网络 A 错误数据包的分布，经过 CRC Check 模块之后，丢弃的数据包图为 6-12。图 6-11 代表虚拟链路 1 网络 B 错误数据包的分布，经过 CRC Check 模块之后，对应的数据包丢弃的图为 6-13。图 6-10 中，数据包的错误率比图 6-11 中数据包的错误率高，所以对应的无效数据包丢弃图 6-12 与图 6-11 中的数据包多(图 6-12 显示网络 A 上数据丢失总数为 38，丢包率为 2%，图 6-13 显示网络 A 上数据丢失总数为 8，丢包率为 0.5%)。经过冗余管理后，两个网络中，只要相应序列号的数据包有一个有效时，就可以接收到有效数据包，如图 6-14 所示，数据包只丢失了 3 个。

当虚拟链路 1 网络 A 注入 network babbling 错误，而网络 B 注入 CRC 和包错误时，所得结果如图 6-15 到图 6-19 所示。

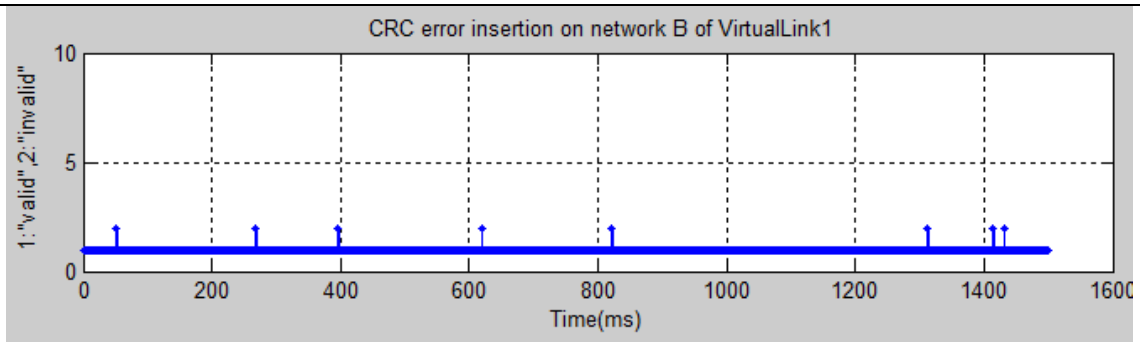

图 6-15 网络 B 注入 CRC 和包错误

Fig.6-15 CRC and Packet Error Injection on Network B

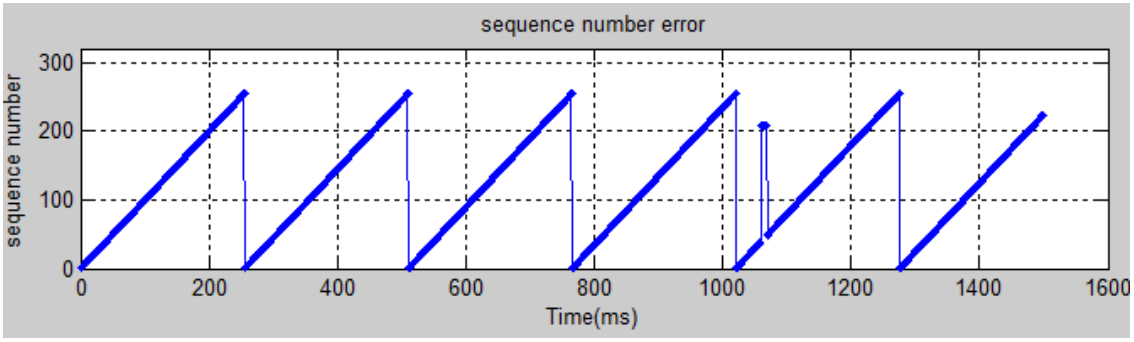

图 6-16 网络 A 网络 Babbling 错误注入

Fig.6-16 Network Babbling Error Injection on Network A

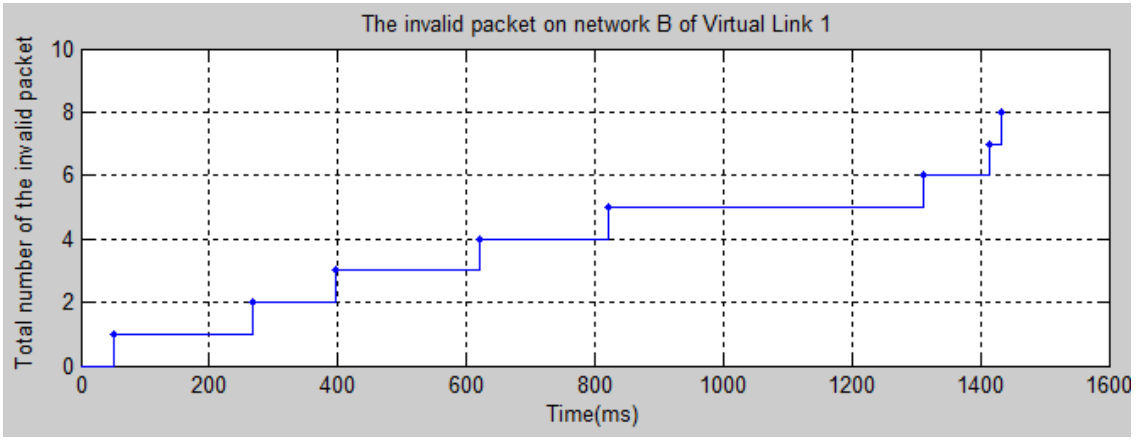

图 6-17 网络 B 上 CRC 校验后，丢弃的无效的数据包

Fig.6-17 The Lost Packet after CRC check on Network B

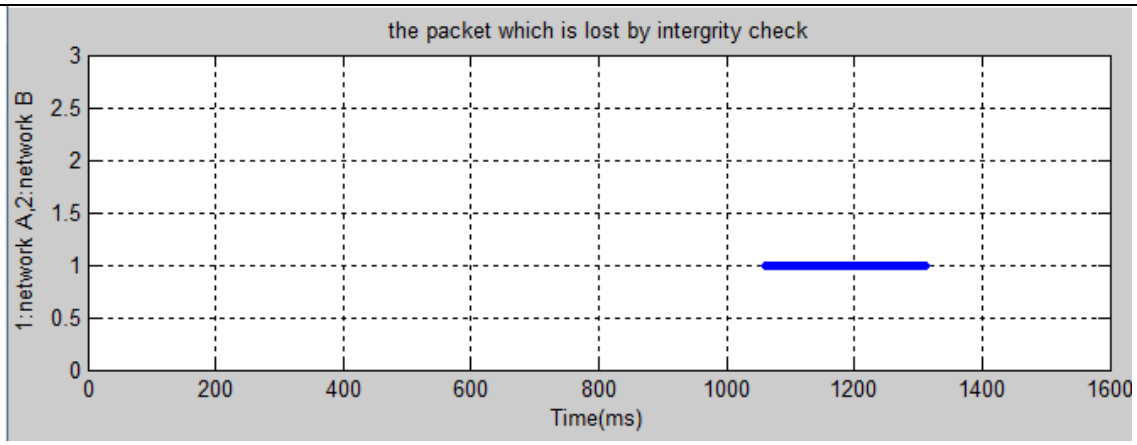

图 6-18 网络 A 上完整性检查丢弃的无效数据包

Fig.6-18 The Lost Packet after Integrity Check on Network A

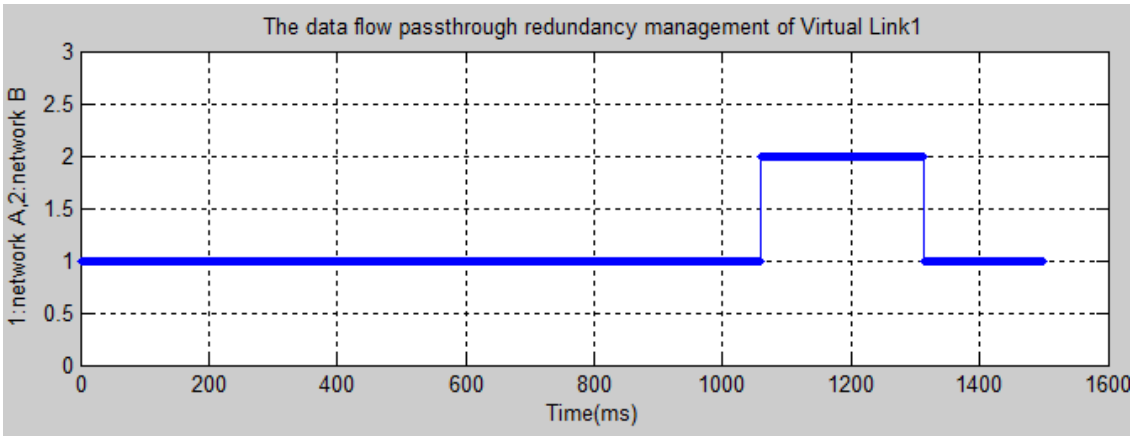

图 6-19 经过冗余管理后的数据流

Fig.6-19 The Data Flow after Redundancy Management

图 6-15 和图 6-16 显示虚拟链路 1 网络 A 和网络 B 分别注入 CRC 和包错误与网络 Babbling 错误时，数据包的图示。可以看到，网络 B 中，在 1000ms 到 1200ms 中，发生了网络 Babbling 错误。经过完整性检查后，网络可以检测出无效的数据包，如图 6-18 所示，图 6-19 为经过冗余管理系统后，航空子系统收到的有效的数据包。可见，经过冗余管理后，数据丢包率显著的降低，能够满足航空系统对于数据安全的苛刻的要求。

6.3 本章小结

本章主要利用第 5 章搭建的发送端系统，接收端系统和交换机，根据 A380 的内部拓扑结构搭建了一个 9 个交换机、13 个端系统和 9 条虚拟链路组成的平台，并使用 4.2 提出的基于权重的 deadline 分配方法进行 deadline 分配，并进行可调性分析。最后，利用搭建的仿真平台，得到端到端系统的延时抖动结果，并利用错误注入模块，注入各种错误，对新的冗余管理算法进行仿真分析研究。

7 结论

本文主要对于新一代航空电子全双工以太网（AFDX）进行了分析，对于 AFDX 网络的冗余容错进行了研究，提出了在 AFDX 中引入 jitter-EDD 机制来解决有效数据包丢失的问题。

首先，本文对于 AFDX 网络模型进行抽象，对于 jitter-EDD 机制在 AFDX 中实施建立了网络模型，包括发送端系统、交换机和接收端系统。发送端系统将航空子系统的数据经过发送端系统整流器整流后，交给 EDF 调度器进行调度。交换机负责根据静态路由，将进入端口的数据经过路由模块后，发送对应虚拟链路的 jitter-EDD 整流器后，经过对应的端口的 EDF 调度器发送到下一跳网络。接收端系统负责将接收到的数据进行完整性检查后，经过冗余管理将两条相互冗余的数据流整合为一条有效的数据流后，发送到相应应用层系统。

其次，现有的 AFDX 航空数据总线系统在可靠性和安全性方面，尚不能满足对于关键性飞行控制系统的苛刻要求。究其原因，当多个终端连接到一个交换机(Switch)上，并且在同一个终端上的多个子系统(subsystem)需要同时发送消息时，存在严重的资源竞争问题，往往导致系统性能的不确定性甚至系统崩溃。因此，研究网络节点上任务的可调度性就显得很必要。本为给出了 AFDX 网络中，同一个端口上所有数据流的可调度的充分必要条件，保证数据传输满足延时需求。

再次，我们对原始的 jitter-EDD 没有给出 deadline 分配有效的方案，本文提出了一种基于权重的 deadline 方案，有效的解决了 deadline 分配的问题。另外为了进一步减小端到端的延时，我们对原始的 jitter-EDD 机制进行了改进，在接收端系统中加入 jitter-EDD 整流器。jitter-EDD 机制可能会增加系统端到端延迟，但是可以有效的减少端到端延迟抖动。

最后，在研究端到端延时抖动的基础上，设计新的冗余管理算法，对网络数据包及其冗余数据包进行安全可靠的冗余管理，确保网络可靠的工作。我们使用 SimEvents 作为仿真工具，根据 A380 内部网络拓扑结构，搭建仿真平台。使用错误注入的方法来验证新设计的 AFDX 网络冗余管理机制对于各种网络错误的使用效果。

本文的研究成果可以用于网络整流、端到端延时抖动控制、网络容错等方面。为了简化模型，本文只对于实时数据流进行了研究，没有考虑非实时数据流的影响，需要进一步的研究。另外，jitter-EDD 中，时间戳的精度对于系统有非常重要的影响，在下一

步的工作中，应该加入时间戳的精度对于系统的影响的研究。由于时间关系和作者理论水平所限，不足和疏漏之处在所难免，敬请读者给予批评与指正。

参考文献

[1] 石改辉，张原，下一代航空数据网络体系结构研究，电子工程，2006(4)，p29-33.

[2] 袁梅，曲方伟，民用机载航空总线发展概述，中国航空学会 2007 年学术年会机载航电专题，2007.

[3] Charara H. , Fraboul, C. , Modeling and simulation of avionics full duplex switched Ethernet, Proceedings of the Advanced Industrial Conference on Telecommunications / Service Assurance with Partial and Intermittent Resources Conference, IEEE, 2005.

[4] 熊华钢，李峭，黄永葵，航空电子全双工交换式以太网标准研究，航空标准化与质量，2008(1)，p25-33.

[5] ARINC, Arinc project paper 664: Aircraft data network, part7-avionics full duplex switched ethernet(afdx) network, ARINC, 2005.

[6] ARINC, Homepage, 2005, http://www.arinc.com.

[7] Condor Engineering, AFDX Protocol Tutorial, Condor Engineering Inc, 2004.

[8] Remi Cabaret, Francoise David-Tupinier, Remi Andreoletti, Method of data Integrity control in an AFDX network, US, 7817565B2, 2010.10.

[9] GE Fanuc Embedded Systems, AFDX/ARINC 664 Protocol Tutorial, GeFanuc, 2005.

[10] 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.

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

[12] Janä Tubrichvon, Hanxleden Reinhard, Formal Specification and Analysis of AFDX Redundancy Management, SpringerLink Date, 2007.

[13] Madhukar Anand, Dajani-Brown, Steve Vestal, Formal Modeling and Analysis of AFDX Frame Management Design, The 9th IEEE International symposium on Object-oriented Real-time Distributed Computing, 2006，p393-399.

[14] 陈昕，杨杰，周拥军，航空全双工交换以太网冗余管理机制研究，计算机工程与应用[J]，2009，45(2)，p102-105.

[15] 陈昕，周拥军，路娟，AFDX 冗余管理机制的仿真，计算机工程，2008(23)，Vol.34，p92-94.

[16] 李哲，田泽，张荣华，AFDX 网络中 SkewMax 的研究，计算机技术与发展，2009(6)，

Vol.20，p249-253.

[17] Domenico Ferrari, Verma Dinesh, A Scheme for Real-Time Channel Establishment in Wide-Area Networks, IEEE Journal on Selected Areas in Communications, 1990, P368-379.

[18] Ferrari Domenico, Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks, NOSSDAV, 1991, p72-83.

[19] Fengxiang Zhang, Burns Alan，Schedulability analysis for Real-Time Systems with EDF Scheduling, IEEE TRANSACTION ON COMPUTERS, 2009.

[20] Kevin Jeffay U, Martel DonalF, Stanat,Charles, ON Non-Preemptive Scheduling of Periodic and Sporadic Tasks, RTSS, IEEE Computer Society Press, 1991, p129-139.

[21] Laurent GeorgeRivierre, Marco SpuriNicolas, Preemptive and Non-Preemptive Real-Time Uni-Processor Scheduling, INRIA, 1996.

[22] QinZheng, Kang G.Shin, On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks, IEEE TRANSACTIONS ON COMMUNICATION, 1994, p1096-1105.

[23] Jean-Philippe Georges, Eric Rondeau, Thierry Divoux, Evaluation of switched Ethernet in an industrial context by using the Network Calculus, 4th IEEE international workshop on factory communication systems, 2002.

[24] Frédéric Ridouard, Jean-Luc Scharbarg, Christian Fraboul，Stochastic upper bounds for heterogeneous flows using a Static Priority Queuing on an AFDX network, Dans Journées FAC'2008, SVF/FéRIA, 2008.

[25] Ahlem Mifdaoui, Fabrice Frances, Christian Fraboul, Real-time guarantees on Full-Duplex Switched Ethernet for military applications, Proceedings of the 3rd European Congress Embedded Real Time Software, 2006.

[26] Ahlem Mifdaoui, Fabrice Frances, Christian Fraboul, Full-Duplex Switched Ethernet for Next Generation "1553B"-based Applications, The 13th IEEE Real-Time and Embedded Technology and Applications Symposium(RTAS07), 2007.

[27] Ahlem Mifdaoui, Fabrice Frances, Christian Fraboul, Real-time characteristics of Switched Ethernet for "1553B"-Embedded Applications: Simulation and Analysis, SIES, 2007, p33-40.

[28] Rene L. Cruz, A calculus for network delay, Part I: Network elements in isolation, IEEE Transactions on information theory, 1991(1), vol. 37, p114-131.

[29] Marc Boyer, Christian Fraboul, Tightening end to end delay upper bound for AFDX

network calculus with rate latency FCFS servers using network calculus, Proceedings of the 7th IEEE International Workshop on Factory Communication Systems Communication in Automation (WFCS 2008), 2008.

[30] 王广海，贾秋玲，航空 ARINC429 总线通信的余度设计，计算机测量与控制，2008(5)，p695-697.

[31] 朱大奇，故障分析法基本原理，航空电子设备故障诊断技术研究，南京，2002.

[32] 朱大奇，基于信息融合的故障诊断技术，航空电子设备故障诊断技术研究，南京，2002.

[33] 朱大奇，刘永安，故障诊断的信息融合方法，控制与决策，2007.12，p1321-1326.

[34] Dinesh C. Verma，Hui Zhang, and Domenico Ferrari，Delay Jitter Control for Real-Time Communication in a Packet-Swtiching Network, Communications Software, 1991.

[35] Mark J. Karol, Michael G. Hluchyj, Samuel P. Morgan, Input Versus Output Queueing on a Space-Division Packet Switch, IEEE Transactions on Communications. Vol. 35, 1987.

[36] MATLAB, SimEvents Getting Start, 2001, www.mathworks.com/products/simevents/.

致 谢

经过两年多的忙碌，本论文终于接近尾声，回首往昔岁月，内心感慨万千。

首先感谢我的导师胡飞教授，他是我们 AFDX 项目的领衔带头人。胡飞老师，治学严谨、宽厚仁慈、工作认真负责，为我树立了学习工作的榜样。胡飞老师带领我们每周进行一次英文文献的阅读，使我获得了阅读英文文献的方法，通过对世界著名会议期刊论文的讨论，使我了解到当前最新最前沿的研究热点，对科学的研究方法有了一定的认识，初步掌握了一定的做研究的方法。胡老师严于律己，宽于待人，每天早早便来到学校，晚上 11 点才回家，只要有时间便来实验室指导我们做实验，坚持在一线工作，亲历其为。胡老师严谨的科学精神，认真负责的工作态度，永远值得我学习。

同时，要感谢我毕业论文的主要的指导教师李健老师。李健老师亲切和蔼，眼光独特，对于做研究有丰富的经验。在论文选题、理论推导和做实验的过程中，我得到的李健老师大量的帮助。还记得当初，我懵懵懂懂，对于自己要研究什么都不清楚的时刻，是李老师指导我们阅读网络微积分，给我们推荐大量调度理论和实时网络方面的文章，一步步引导我们提出了自己的想法。在理论推导阶段，我们每周最少开 2 次会，来对我们的工作进展汇报和对将来的工作进行讨论。李老师，以敏锐的眼光捕捉到我们工作中的不足，并提出许多具有建设性意见的想法，为我们的进一步的工作指明了方向。

周海瑞博士，工作认真负责，为人和蔼可亲，在网络研究方面具有很深的功底。最初在提出网络模型时，周海瑞博士给予了极大的帮助，提出了许多建设性的意见。在后期的工作中，也得到了周海瑞博士的帮助。周海瑞博士，还给我修改小论文，对我小论文的文章的结构调整给出了很好的建议。

姚建国老师，认真负责，认真的阅读了我的小论文，并与我对于 AFDX 的网络冗余管理进行了讨论，提出了一些有用的建议。姚老师为人非常的细心，对于我的小论文进行认真仔细的修改，哪怕一个标点，一个单词都不放过，使我的小论文的质量得到了可靠的保证。

还有我要特别的感谢实验室的同学们，他们是韩国栋、胡光宇、吴宇抗、王辉、王臣虎、周树勋、石枢、安杰、王威振、鲁晓园、胡靖飞、李晓波等人。我们互相讨论 Matlab、Simevents、Latex 的使用，共同阅读论文并讨论，共同通宵加班敲代码，还有一起去胡老师家吃烧烤，一起去打羽毛球……谢谢你们给我提供帮助，也感谢你们带给我快乐。

感谢学院的各位老师，给我们带来的精彩的课程。沈备军的软件工程，邹恒明的算

法，梁阿磊的计算机体系结构、戚正伟的操作系统等课程为我们打下了坚实的基础。谢谢各位老师对于我们的教导和关怀，你们辛苦了。

最后，我要感谢我的爸爸妈妈。是爸爸妈妈辛苦把我养大，并教育我做人做事。是你们默默的支持和鼓励是我有足够的勇气，克服困难，勇往直前，我祝愿你们永远健康快乐、幸福美满。

在此论文即将完成之际，我的内心无法平静，从科研选题开始到论文顺利完成，许许多多多的人给予我巨大的帮助，我再次的向那些帮助过我的老师和同学表示衷心的感谢和诚挚的谢意。

攻读学位期间发表的学术论文目录

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

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：

日期：2012 年 1 月 5 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：

日期：2012年1月5日

指导教师签名：

日期：2012年1月4日