

申请上海交通大学工程硕士学位论文

航空机载 AFDX 仿真平台的设计与实现

学校代码: 10248
作者姓名: 冯涛
学 号: 1110372213
第一导师: 姚建国
第二导师: 潘凌云
学科专业: 软件工程
答辩日期: 2016 年 5 月 10 日

上海交通大学软件学院
2015 年 10 月

A Dissertation Submitted to Shanghai Jiao Tong University
for Master Degree of Engineering

**THE DESIGN AND IMPLEMENTATION OF THE AFDX
SIMULATION PLATFORM FOR AVIONICS SYSTEM**

University Code:	10248
Author:	FengTao
Student ID:	1110372213
Mentor 1:	Yao Jianguo
Mentor 2:	Pan Lingyun
Field:	Software Engineering
Date of Oral Defense:	10.May.2016

School of Software
Shanghai Jiaotong University
Oct. 2015

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：马涛

日期：2015 年 10 月 9 日

航空机载 AFDX 仿真平台的设计与实现

摘 要

近年来,人们对航空的需求越来越旺盛,对飞机的要求也越来越多。更安全、更舒适的飞机需要更多的机载系统来支撑,而机载系统的增加对负责数据传输的航空数据总线提出了更高的要求^[12]。如何更好的研究航空数据总线,对航空数据总线的发展有重要的意义。本文着重于航空机载数据总线系统的仿真,完成了航空机载数据总线系统 AFDX 的模型建立、系统实现与验证。证明了软件实现航空机载数据总线 ARINC 664 P7 仿真的可行性。

论文所开展的研究工作包括:

1. 本论文探讨当前航空数据总线的历史发展以及实际状况,分析目前市场之中较为普及的航空数据总线的特点,认为 AFDX 是目前前景最好的数据总线。目前,国内外的研究尚处于起步阶段,尽早开展 AFDX 航空数据总线的研究,对未来航空数据总线的发展有重要意义。
2. 本论文对 AFDX 的通讯行为进行了分析和研究,给出了 AFDX 网络在仿真过程中的各个要素。
3. 本论文围绕着系统的需求开展针对性的分析,实现了系统整体的架构设计。系统是在 Fedora 系统基础上,通过 C 技术实现。系统之中包括着多个模块,即:数据库、仿真平台主程序、数据发送与接收模块等所共同构成。系统还提供了记录平台运行的日志模块。
4. 本论文实现了航空机载 AFDX 仿真平台。完成了数据库模块、仿真平台主程序的设计和实现,以及数据接收模块、数据发送模块的设计与实现。平台可以在网络结构可变的情况下实现终端系统间的数据通讯。
5. 最后本论文对仿真平台的功能进行验证。验证了仿真平台的可用性,证明了基于商业软硬件实现 AFDX 仿真的可行性。

关键词 航空数据总线, 航空电子全双工通信以太网交换, 仿真, 网络通讯

THE DESIGN AND IMPLEMENTATION OF THE AFDX SIMULATION PLATFORM FOR AVIONICS SYSTEM

ABSTRACT

The demands of aviation rapidly growth these years, as well as requirements on aircraft. Safety and comfortable planes need more avionics systems to achieve which raise the standard of the avionics databus. Intensive study has a great meaning for the development of avionics databus. This paper focuses on the simulation of avionics databus, accomplishing the model establishment, system achievement and verification of avionics system AFDX simulation platform. This paper successfully proves the feasibility of software simulation on ARINC 664 P7.

The paper focuses on researches as below:

Firstly, the paper discusses the history and status in avionics databus, works on the characteristic of avionics databus in current market, and points out AFDX has the best prospect. As the study on AFDX has just begun in and abroad, an early study has a significant meaning on development of avionics databus.

Secondly, this paper studies on the characteristic of AFDX communication behavior and find out the fundamental elements on simulation.

Thirdly, the paper analyzes on demand of the system and accomplishes the designing of the main framework of system. The system designs base on Fedora operating system, architecture and implement under C language. The system is composed by modules such as database, main part of the simulation, data transmission and data reception. The system also provides log module to record the status of simulation platform.

Fourthly, the paper implements the avionics databus AFDX simulation platform. The paper design and implement the data base module, main part module, data transmission and data reception module. Simulation can be achieved under alterable network topology.

Fifthly, the paper validates the function and performance of the simulation platform. The simulation of AFDX under commercial software is possible.

Keywords avionics databus, AFDX, simulation, network communication

目 录

1	绪 论	1
1.1	研究背景与意义	1
1.2	航空数据总线的历史和发展	2
1.3	国内外航空总线模拟的发展现状	4
1.4	论文的主要工作	6
1.5	论文结构	7
2	AFDX 航空总线	8
2.1	AFDX 的相关理论	8
2.1.1	AFDX 的基本概念	8
2.1.2	AFDX 数据帧格式	9
2.1.3	虚拟链接	10
2.1.4	网络交换机	13
2.2	AFDX 网络拓扑	13
2.3	AFDX 网络协议及通讯端口	17
2.4	网络通讯过程	18
2.5	总结	20
3	航空机载 AFDX 仿真平台需求分析与架构设计	21
3.1	航空机载 AFDX 仿真平台分析流程	21
3.2	系统总体需求	23
3.3	功能模块分析	24
3.3.1	仿真平台数据库模块	24
3.3.2	仿真平台主程序模块	24
3.3.3	仿真平台信息发送模块	26
3.3.4	仿真平台信息接收模块	26
3.3.5	仿真平台日志模块	26
3.4	系统架构设计	26
3.5	本章小结	26
4	航空机载 AFDX 仿真平台模块详细设计与实现	28
4.1	数据库模块设计与实现	28

4.2	仿真平台主程序模块设计与实现	29
4.2.1	数据包处理发送	29
4.2.2	数据包序号管理	32
4.2.3	轮询调度	35
4.2.4	虚拟链接	37
4.2.5	数据包接收处理	38
4.2.6	数据包重排	40
4.2.7	数据包封装	43
4.3	信息发送模块	45
4.4	信息接收模块	46
4.5	本章小结	48
5	航空机载 AFDX 仿真平台验证	49
5.1	系统部署	49
5.2	系统功能验证	49
5.2.1	应用场景	49
5.2.2	验证方法	50
5.2.3	系统功能验证	52
5.2.4	系统效果分析	56
5.3	本章小结	56
6	总结与展望	57
6.1	总结	57
6.2	展望	57
	参考文献	59
	致 谢	61
	攻读学位期间发表的学术论文目录	62

1 绪论

如今,航空业影响着人们学习、工作和生活的方方面面。航空业的高速发展极大的改变了人类的生活方式。人们通过航空公司提供的服务实现出行、旅游,“地球村”变得更小。伴随着使用频率的增加,人们对飞机的要求也更高:安全、舒适、人性化,以及更智能的人机交互系统。

为了实现以上这些目标,航空电子系统做出了巨大的贡献。各类传感器和飞行状态监测系统实时关注飞行的状态,更科学智能的人体感知系统进一步保证飞行旅途变得舒适,人机交互系统使得旅途不再漫长,而这些系统的背后,都需要航空数据总线的支撑。

1.1 研究背景与意义

航空业发展至今,已经经历了一个世纪。随着对飞行器要求日益提升,飞机制造商已经意识到传统的数据总线,如 ARINC 429, MIL-STD-1553B 已无法满足飞行器各组件间的通讯需求^[1]。飞行器数据总线的新标准呼之欲出。2005 年,航空电子全双工通信以太网交换(Avionics Full Duplex Switched Ethernet)被提出,该标准被定义在文档 ARINC 664 P7 中。

航空电子全双工通信以太网交换(以下简称 AFDX)在不同的飞行器数据总线中充当底层通信的协议^[2]。飞行器设计工程师需要提供大量通信过程中涉及到的具体参数,如数据包分发和排序、通讯线路冗余、端到端延迟、日志文件大小等。而以上这些需求都需要有飞行器航电工程师逐一实现。

AFDX 是以传统以太网为基础的通讯网络。有别于前几代数据总线半双工传输效率低的缺点,AFDX 采用的是一种全双工的网络通讯方式。这样做的好处是提高了传输效率并解决了数据包碰撞问题,但随之而来的传输时延成为新的问题^[3]。由于多路通讯数据同时到达交换机,交换机的缓存存在溢出风险。一旦溢出,数据包将被丢弃,这对通讯实时性要求极高的航电系统而言是不能允许的。这就需要航电工程师通过合理的系统设计,避免这些问题,搭建高吞吐量、低延时、无丢包的实时网络^[4]。

在此之前,部分研究指出,为了保证 AFDX 规范持续有效的指导航电数据总线的发展,其传输能力是需要重点关注的因素。最近的研究,研究人员把目光聚焦在优化 AFDX 的系统架构、组件拓扑分布上以保证通讯质量。以上研究虽然关注点有不同,但共同点在于研究的基础相同:搭建一套 AFDX 仿真平台,并在此基础上做进一步的研究。

中国立志大力发展中国的民机事业,于 2008 年 5 月 11 日成立了中国商用飞机有限

责任公司(简称“中国商飞公司”)。中国商飞公司是经国务院批准成立,由国务院国有资产监督管理委员会、中国航空工业集团公司等多家国有大型企业共同出资组建,是一家由国家控股的有限责任公司。而作为中国人自己的大飞机,C919 已经决定采用 AFDX 作为其数据总线规范。因此,建立一套仿真平台,深入研究其通讯特性、通讯性能,掌握 AFDX 通讯的实现方式,将可以直接用于 C919 的研发和生产,现实意义十分重大。

AFDX 网络作为最新的民用航空数据总线标准刚被推出,目前采用这一标准的民用飞机并不多。航空业普遍认为,AFDX 的出现,解决了过去航空数据总线总线结构存在单点故障的问题,其冗余链路不但能有效解决单点故障这一问题,还能在数据传输的可靠性上给予保证。同时,AFDX 的传输速率有了很大的提升,达到了 100M/s,这一传输速率是目前在民机应用最广泛的航空数据总线规范 ARINC 429 的 1000 倍。综合以上两点,航空界认为 AFDX 代表了航空数据总线发展的方向,未来市场前景良好。由于 AFDX 于 2005 年推出,因此目前的研究才刚刚开始。国内 AFDX 网络的相关产品非常少,无论是研究人员还是最终用户,对 AFDX 网络的认识更多的停留在理论阶段,对 AFDX 的特性、网络通讯过程、数据传输能力、数据传输稳定性、传输响应时间等都没有深刻的认识。而这些都需要通过对现实中 AFDX 网络通讯的研究来实现。现实中的 AFDX 网络通讯可通过通讯终端利用 AFDX 网络通讯卡配合 AFDX 交换机实现数据的传输。但采购 AFDX 网络通讯卡和 AFDX 交换机的费用高昂,阻碍了业界对 AFDX 网络的研究。

AFDX 网络是基于 IEEE802.3 网络标准,并在其基础上通过协议扩展实现了确定的网络通讯行为和流量控制。而 IEEE802.3 是传统以太网的通讯标准,如果利用以太网基础,并在以太网通讯中通过软件实现 AFDX 网络的协议扩展,进而实现 AFDX 网络的通讯行为,这样既可以利用以太网通讯的设备,又可以实现 AFDX 网络的通讯,这将大大减低搭建 AFDX 网络通讯环境的成本,有效降低了研究 AFDX 网络的准入门槛,为未来更多的研究力量投入该网络的研究打下基础。

1.2 航空数据总线的历史和发展

航空电子设备第一次开始受到重视源于第二次世界大战期间,飞行员与指挥所间的通讯、战斗机之间的通讯使得通讯的需求变得尤为迫切^[5]。机载雷达的引入,其使用的磁电管、热阴极电子管及相关技术对航空电子设备的发展提供了重要的帮助。上世纪 50 年代后期诞生的晶体管取代了之前广泛采用的热阴极电子管,其相对低廉的成本使得早期的航电系统在 60 年代到 70 年代得到了巨大的发展^[6]。

从上世纪 70 年代以来,飞机系统的设计任务的方向有一次革命性的转换,设计人员的关注重点从原先的飞机外形设计逐步过渡到对飞机内部的航空电子系统的设计上^[7]。

发生这样转变的原因在于,设计人员从飞机诞生之初所累积的飞机外形设计的研究经验,已经帮助他们掌握了飞机气动性能等问题。研究人员在研究中发现,在飞机气动性能方面可挖掘的潜力已经几经枯竭。另一方面,同时代的计算机、控制以及电子等多方面技术均有着显著的进步,此类技术所带来的产品使得航空自动化整体均有着显著的发展。而此类变化使得计算机充当飞机自动化的核心具备较高的可行性。当代飞机之中存在大量的电子设备和子系统如飞行控制系统、导航系统、自动驾驶系统等,这些系统都通过计算机来完成,而此类计算机非但需要满足子系统的实际需要,同时需要开展信息交互以实现整体功能整合的目的,其就属于航空电子综合系统。其中航空数据总线所需要处理的就是飞机上各个系统之间的信息交换,它是航空电子综合系统的基石。

早期,航空电子设备是通过模拟设备和系统来实现,各类信号被以某种线性或预测方式为模拟为信号电平,如电压,电流,频率,脉冲宽度或相移等。这种类型的模拟系统的一般容易发生变异导致模拟不准确,如固有特性和制造误差,部件温度特性,设备使用年限,漂移和其它非线性因素^[8]。

在被应用于飞行器之前,数字计算的原理已经被广泛理解了。数字计算克服了模拟计算的可变性,提供高精度准确的可重复的结果且不受制造偏差和运行温度的影响。早期的数字电脑是体型庞大,仅被用于完成大型的办公应用程序,无法被用来完成与飞机设计相关的计算,直到集成电路器件实现所有功能集成在单个设备才使得这一切成为可能。

在美国,第一个使用数字技术的飞机是第一架飞机是北美的 A-5,是于 20 世纪 60 年代开始服役的美国海军舰载轰炸机。在英国,原本打算使用数字技术的第一架飞机是 TSR2,但命途多舛的 TSR2 还没有完成就被英国政府在 1965 年取消了。直到英法联合设计捷豹飞机和哈维兰彗星海上巡逻飞机在 20 世纪 60 年代的开始武器系统设计工作时才重新启用数字计算^[9]。

自 20 世纪 70 年代后期和 80 年代早期开始,数字技术在飞机控制系统和任务相关系统的设计方面被使用的越来越多。功能强大和成本低廉的微处理器和更先进的软件开发工具的应用,使得数字技术的在整个飞机设计中得到了广泛的应用。如今,飞机上所有的系统的设计,包括航空数据总线系统都有数字技术的身影。

ARINC 429 是目前在民机领域使用最广泛的数据总线规范。大量的飞机如民用运输机、支线客机、行政公务机都使用其作为自己的数据总线。自从波音 757/767 飞机和空客飞机在 20 世纪 80 年代初引进,几乎所有的飞机在生产时都选择采用这套数据总线。它是一个单源,多同步,线性拓扑数据总线。它在传输时使用双绞线屏蔽电缆。最多有 20 个接收终端可以连接到总线上。由于其传输速率低,只有 100Kbps,因此只需要将设备简单相连即可,甚至连用于校验数据的终端都没有被设计^[10]。

MIL-STD-1553B 于上世纪 70 年代末被正式推出,是目前仍在使用的航空数据总线^[11]。同 ARINC 429 不同, MIL-STD-1553B 更多的被运用于军机。它是一种按指令/响应模式工作的串行时间分割多路总线。由于它能够显著降低各种航空电子设备占用的体积以及重量,同时使得实际运用更加便捷,从而使得航空电子系统整体成本控制于合理范畴之内。上述优势使得军用领域对其青睐有加。但是, 1553B 总线的设计毕竟在四十年前被研发出来,其设计架构使其工作模式只能是集中式控制、分布式处理消息的线性拓扑结构,最大的传输速率也不过 1Mbps。在 1553B 总线中,单次的消息传输中最大的消息块长度仅为 16bpw。由于消息块长度的瓶颈,导致 1553B 的消息传输的传输速率仅为 200~300Kbps。而 1553B 支持的网络上能接入的终端数量不能超过 31 个^[12]。

ARINC 629 开发于上世纪 80 年代末,以提供民用飞机具有多源,多同步,线性拓扑网络结构。它是建立在 MIL-STD-1553 的经验上,但是不再需要集中总线控制器;民用航空界希望通过这种设计来避免单点故障的问题^[13]。ARINC 629 使用 20 比特位的曼彻斯特编码,其 2 Mbps 的传输速率是其前辈 MIL-STD-1553 的两倍。接入终端的数量上,它已允许高达 128 个设备的连入。ARINC-629 甚至支持在支持双冗余的基础上,进一步支持三重冗余和四重冗余。

ARINC 664 P7 通信网络是一个多数据总线。目前已采用这套数据总线规范的包括空客 A380、A350、A400M,波音 787 梦想飞机,COMAC ARJ21 和苏霍伊 Super-jet 100。该标准极有可能在未来成为大部分民用飞机的标准数据总线标准。

ARINC 664 P7, 也被航空电子全双工通信以太网交换 (Avionics Full Duplex Switched Ethernet), 是一个基于 10/100 Mbps 交换式以太网技术 (IEEE802.3)、媒体访问控制 (MAC) 地址、互联网协议 (IP) 和用户数据报协议 (UDP) 结合,并通过特殊的协议扩展和流量管理,实现了确定性的网络通讯行为,并符合航空电子系统使用的一套数据总线。

1.3 国内外航空总线模拟的发展现状

由于数据总线的硬件设备非常昂贵,利用这些设备进行研究的费用过于高昂。因此国内外普遍通过软件、或者软件硬件结合的方式搭建仿真平台来研究数据总线。

国内在 2000 年以前已经开始了 ARINC 429 的仿真模拟。这些仿真都基于某个给定的应用场景。这样做的好处是和应用场景结合更紧密^[14]。ARINC 429 作为已经被广泛使用的数据总线规范,技术已经成熟,且已经被证明没有进一步深入研究的意义。因此结合用户的使用场景完成相应系统的建设是其目前的主要用途。

国内对 MIL-STD-1553B 的研究较多而对目前应用不广的 ARINC 629 的研究较少,通过对 MIL-STD-1553B 的研究,不少研究人员已经开始尝试把它应用到航空领域外的

其他领域。和国外的研究者一样，国内的研究者也开始逐步把目光转移到新推出不久的 AFDX 上^[14]，因此，谁先占领这个目前最新兴的航空数据总线技术的制高点，将有可能引领未来航空数据总线的发展，进而对世界的航空工业带来不可估量的影响。

目前，针对 AFDX 网络的仿真，业界有两种做法，即硬件仿真和软件仿真。通常来说，硬件仿真的优势在于其接收、处理、转发数据时的速度比软件仿真快。而对于硬件仿真来说，需要根据使用 AFDX 网络的应用的网络通讯需求来设置 AFDX 网络的硬件架构及配置文件。完成配置后，AFDX 网络就可以开始传输数据了。

AFDX 网络的协议 ARINC 664 P7 把协议的重点关注在 AFDX 网络的通讯行为以及通讯性能上，而选择将网络系统设计的主动性交给设计人员。ARINC 664 P7 要求网络的传输延迟和抖动都应该低于其规定的上限。而对于符合达到这些要求的 AFDX 仿真，都被视为合格的 AFDX 仿真。

既然 AFDX 协议并没有严格限定仿真的方式，那么软件仿真也是一个选择。与硬件仿真相比，软件仿真并非一无是处，其优点如下：

- 软件仿真在开发完成后的维护成本比硬件的维护成本低廉。在硬件系统中查错往往需要替换掉故障组件，而与之相比，软件仿真的组件是模块化的且可再利用的，这为维护节约了时间和金钱。
- 软件仿真可移植性强。软件仿真可以在不同的平台环境下运行。软件的系统架构通常不会只针对一个平台开发，它会通过平台运行模块将软件核心部分与其运行的平台进行隔离。一旦该软件需要在不同的平台环境下运行，只要对平台运行模块进行必要的修改和开发，而保留软件的核心部分不变即可实现支持不同平台。这种方式可以通过最少的时间和最小的开发成本实现软件仿真在不同平台环境上的迁移。
- 软件仿真的实现灵活性高。当需求变更时，软件的可调整性是显而易见的。新的应用或需求可以轻松地在原有软件仿真上实现。而与之相比，硬件仿真的系统架构非常固定，新的应用或需求往往需要重新设计硬件的架构才能够实现，往往带来时间和成本的增加。
- 软件仿真的升级非常灵活。上面提到硬件仿真的系统架构非常固定，一旦产品成型，很难再对其进行改变和升级。同时，所有的仿真都是针对特定的应用的需求来设计的，一旦该应用提出了新的要求而现有硬件仿真无法满足时，那么重新设计在所难免，而现有的系统也只能移作他用甚至报废。软件仿真则可以针对应用的新需求进行扩充，灵活度更高，加之可以利用原有的组件，成本也更低。
- 虽然硬件仿真的优势是处理速度比软件仿真更快，然而，值得注意的是，硬件

的处理速度、内存大小等参数都在随着硬件发展的日新月异而飞速进步，这些都给了软件仿真充分利用硬件资源的条件。同时，航空业往往并不需要使用最新的硬件产品，如此看来，硬件仿真的速度优势就更不明显了。

软件仿真又分为两类：一类是利用已有的仿真软件，结合仿真软件提供的功能和模块，根据 ARINC 664 P7 规范，模拟 AFDX 网络的通讯行为。如宋东^[15]等采用 Network Simulation2 Platform（以下简称 NS2）软件模拟 AFDX 网络。这款软件是一种针对网络技术的开源、免费的软件模拟平台。通过它，用户可以进行网络协议通讯行为的模拟和研究。不仅如此，NS2 还支持用户通过软件开发，模拟该软件平台本身并不包含的网络协议。正是这一特点，经过广大用户多年的使用和扩展，时至今日，NS2 所包含的功能几乎囊括了当今网络研究的方方面面，NS2 也成了业界广泛使用的一款网络模拟软件平台。选择 NS2 进行 AFDX 仿真的好处在于可以充分利用 NS2 中提供的节点、链路、代理、包等基本元素进行仿真，减少了研究人员的编程工作量。但是缺点也显而易见，该项研究工作是建立在 NS2 这款仿真平台上的，除了研究以外，不存在将仿真平台直接运用在工业上的可能性。

另一类是直接进行软件开发。选择实时系统或非实时系统为平台，根据 ARINC 664 P7 规范，模拟 AFDX 网络的通讯行为。如陈欣^[16]等利用实时系统 VxWorks 操作系统开发的 AFDX 仿真。这样的仿真工作量比利用软件模拟的工作量大，好处就在于软件开发出来的产品是实实在在的 AFDX 网络，可以运用在工业上，并且基于工业环境进行的 AFDX 研究结果显然更有说服力。

1.4 论文的主要工作

论文分析了航空数据总线的历史与现状，研究了当前市场常见航空数据总线的特点，认为 AFDX 是目前前景最好的数据总线。目前，国内外的研究尚处于起步阶段，尽早开展 AFDX 航空数据总线的研究，对未来航空数据总线的发展有重要意义。

论文对 AFDX 的规范进行了研究，并着重对规范中定义的 AFDX 网络通讯行为进行了分析和研究，给出了 AFDX 网络在仿真过程中的各个要素。首先，论文对 AFDX 网络仿真平台的需求进行分析，并在此基础上完成了平台架构设计。平台开发在 Fedora 系统上，通过 C 语言实现。系统包括数据库模块、仿真平台主程序模块、数据发送模块、数据接收模块等模块。同时，系统还提供了记录平台运行的日志模块。

本论文对实现的 AFDX 仿真平台的功能进行验证。验证了仿真平台的可用性，证明了基于商业软硬件实现 AFDX 仿真的可行性。为探索满足我国民用飞机航空数据总线的研究及应用的需求提出了新思路。

最后，论文对本次研究工作进行了总结，给出了本次研究的局限性及未来工作的方

向。

1.5 论文结构

第一章讨论了航空机载 AFDX 仿真平台的研究背景,介绍了航空数据总线的发展和历史,分析了航空数据总线仿真平台的研究现状及研究 AFDX 仿真的实际意义。本章还讨论了国内外航空数据总线模拟的进展,引出了 AFDX 仿真研究必要性和紧迫性。

第二章进行 AFDX 的分析和研究,给出了 AFDX 网络在仿真过程中的各个要素,并介绍了 AFDX 网络的通讯行为。

第三章分析了航空机载 AFDX 仿真平台的具体需要,对于当代航空制造业的实际业务特性,开展系统的架构设计工作。

第四章介绍航空机载 AFDX 仿真平台各功能模块的详细设计与实现。

第五章对航空机载 AFDX 仿真平台进行了验证测试,验证了系统的有效性和可靠性。

第六章对本次研究工作进行了总结,并指出了研究工作的局限,同时对相关的技术进行了总结与展望。

2 AFDX 航空总线

AFDX 航空数据总线是当今前景最好的民用数据总线，目前已被运用在多架最新的商用客机上。而理解 AFDX 协议规范及其网络通讯行为是 AFDX 网络仿真的基础。

2.1 AFDX 的相关理论

AFDX 网络是目前前景最好的航空数据总线规范。对于其细节的研究，尤其是重要元素及网络通讯行为，对仿真工作有重要的指导意义。

2.1.1 AFDX 的基本概念

AFDX 是一套用于飞行器计算机系统和各子系统间传输数据的数据总线。该总线起源于以太网规范 802.3，最终被定义为 ARINC 664 P7^[17]。它比目前应用最广泛的航空数据总线标准 ARINC 429 快 1000 倍。目前，ARINC 664 P7 已被应用在空中客车 A380、波音 787、和中国商飞 C919 上。

AFDX 被业界广泛关注的一个重要因素，在于摒弃了它的前辈 ARINC 429 半双工的传输模式，取而代之的是全双工传输模式。全双工传输模式带来的传输效率的提升是显而易见的，无冲突传输也是其优点之一。然而全双工的传输机制也有它难以克服的弊端，多数据流同时传输容易造成传输时延。同时，无法立刻被传输的数据存储在交换机的缓存中，这些缓存一旦被写满，数据丢失在所难免^[18]。不过，通过航电工程师调整通讯中的参数配置，AFDX 可以有效避免这一问题的发生。

AFDX 由以下三部分组成：

- (1) 终端系统：提供数据传输接口并保证功能模块和终端系统的数据交换稳定可靠，同时可以提供数据接收接口接收来自其他终端系统发送过来的信息。
- (2) 网络交换机：用于把飞行器内的各通讯终端系统接入网络，接收、处理、传输数据帧至目标地址。同时，网络交换机还能够根据预定义的规则对数据进行放行或丢弃操作。
- (3) 虚拟链接：终端系统通过虚拟链接传输数据。虚拟链接是单向的，从唯一的源终端系统出发，把数据传输到一个或多个目标终端系统。

终端系统利用虚拟链接传输数据。网络交换机被用来进行流量控制、信息过滤、数据完整性校验并把数据发送到目标终端系统。以上这些功能的配置参数都被存储在交换机的配置文件内，以保证网络和终端在数据交换过程中的带宽、时延、抖动和完整性检

查。

由于 AFDX 基于已经被证明了的以太网技术，它的带宽有了显著的提升，最高可达到 100Mbps，是 ARINC 629 的传输效率的 50 倍。它改变了以往航空数据总线的总线型拓扑结构，取而代之的是性能更可靠、传输效率更高的星型结构。通过利用带冲突检测的载波监听多路访问技术 (CDMA/CD)，AFDX 也可以轻易避免数据帧冲突的问题^[18]。

2.1.2 AFDX 数据帧格式

以太网的帧格式如图 2-1 所示：

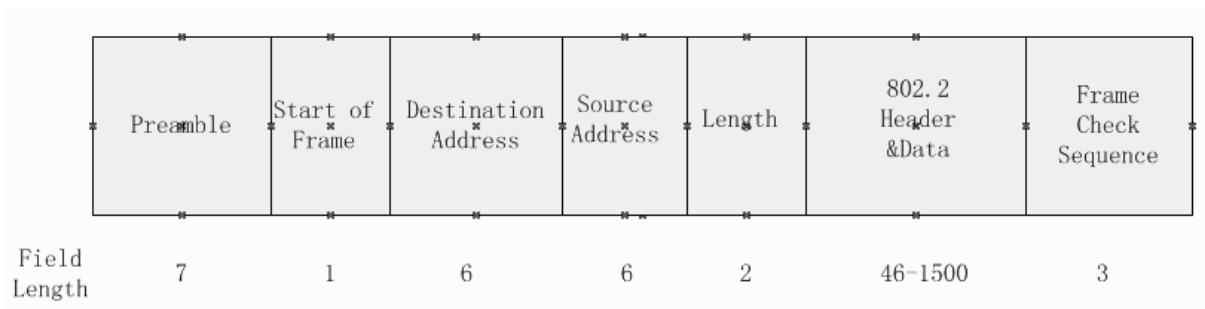


图 2-1 FAT 以太网数据帧格式示意图

Figure 2-1 FAT Ethernet Data Frame

以太网(IEEE 802.3)帧格式：

- 前导码 (Preamble)：7 字节 0x55, 1、0 间隔，主要用在信号同步工作之中；
- 帧起始定界符 (Start of Frame)：1 字节 0xD5(10101011)，主要用在代表数据帧的开始；
- 目的地址 (Destination Address)：长度为 6 字节，用于表示数据帧的目的地址；
- 源地址(Source Address)：长度为 6 字节，用于表示数据帧的源地址
- 数据类型/长度 (Length)：长度为 2 字节，0~1500 保留为长度域值，1536~65535 保留为类型域值(0x0600~0xFFFF)
- 数据 (802.2 Header & Data)：长度范围为 46~1500 字节，用于存储数据帧的数据部分
- 帧校验序列(Frame Check Sequence)：长度为 4 字节，通过 CRC 计算从目的 MAC 到达数据域之中的内容而获取对应的校检。

运用 CSMA/CD 来充当规避冲突的局域网也有别称为以太网。而具体规避通讯之中存在冲突问题的方式为：先侦听网络后发送数据、边侦听网络边发送数据、侦听到冲突后随机延迟后重发。CSMA/CD 保证如果在网络内的通讯过程中存在冲突，系统之中的

所有主机均能够监测。而其中存在的最小间隙以及帧长限制，目的也是为了规避可能存在的冲突。

传统以太网已经在大量的实际应用中证明了自己。面对一般的民用通讯场景，以太网的传输已经足够稳定，毫秒级甚至秒级的时延也不会对用户体验造成严重的影响，丢失的数据包通过重传一样可以提供服务^[19]。然而，作为一种极特殊的应用场景，飞行器内部系统间的通讯对通讯网络的要求更加苛刻，飞行过程任何一个组件发出的信号都必须被完整、及时的传递到目标系统，稍有差池将可能威胁飞行安全。在这种情况下，传统以太网的传输机制就显得不够稳定，丢失数据包意味着飞行器可能无法获得某些重要信息；同时，高达秒级的延时对于高速行进的飞行器而言是无法接受的，飞行员将无法得知此刻系统显示的飞行状态究竟是当前位置的飞行状态还是几公里以前的飞行状态，这将严重影响他们的判断，进而威胁整架飞机上机组人员和乘客的安全。因此，飞行器内部的通讯网络必须是可靠的实时通讯网络^[20]。

有别于传统以太网数据帧的格式，ARINC 664 P7 针对航电系统通讯的实际需求，对其格式做出了一定的调整，如图 2-2 所示。

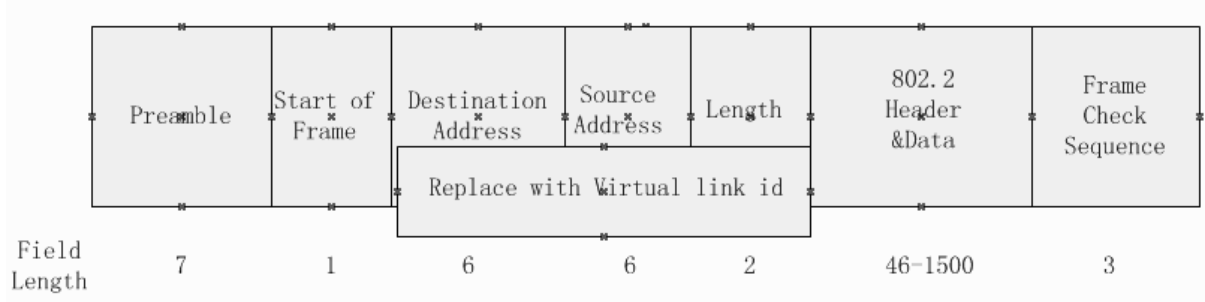


图 2-2 AFDX 数据帧格式
Figure 2-2 AFDX Data Frame

ARINC664 P7 并没有调整前导码、帧起始定界符、数据、帧校验序列这四个部分，它用虚拟链接 ID 取代了目标地址、源地址、数据类型/长度这三项信息。

2.1.3 虚拟链接

虚拟链接（如图 2-3）是一种用于在 AFDX 网络中传输数据的逻辑链接。一旦传输的带宽确定，该网络的最大时延和最大抖动都将随之确定。虚拟链接通过交换机连接网络内的输入、输出端口。需要注意的是，一个确定的虚拟链接有且只能有一个发送方，但可以有一个或以上的接收方。由于虚拟链接单向通信的特点，网络中的终端系统可以不是任何一个虚拟链接的发送方，或者不是任何一个虚拟链接的接收方^[21]。

虚拟链接可以由子虚拟连接构成。一个虚拟链接的子虚拟连接数量小于等于 4 个。

子虚拟连接的用处在于更有效的利用虚拟链接的资源^[22]。

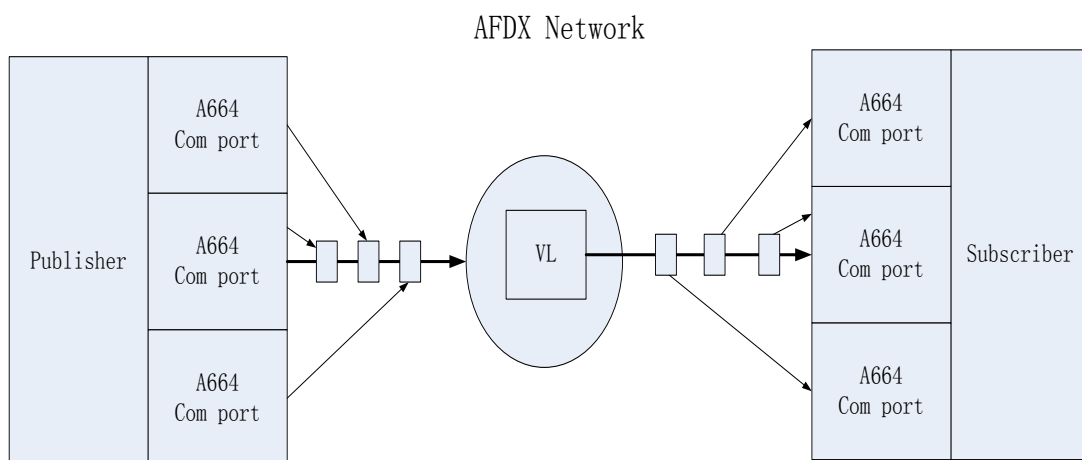


图 2-3 虚拟链接示意图^[23]

Figure 2-3 Virtual Link

在 AFDX 中，交换机需要把接收到的信息传递给一个或多个目的链接。数据帧和唯一的一个虚拟链接 ID 绑定，并且是唯一的源终端系统，并被发往一个或多个目的端口。如图 2-4 所示。

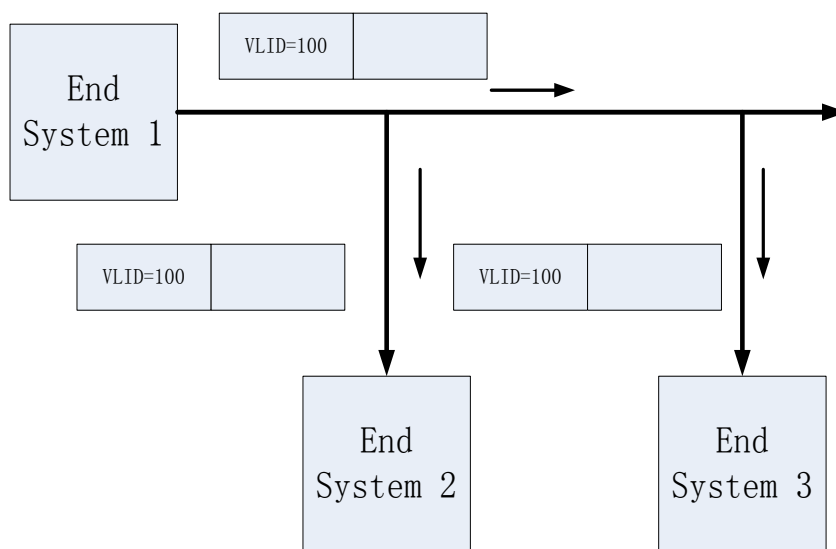


图 2-4 数据包发送示例^[23]

Figure 2-4 Data Transmission

虚拟链接有以下几个特点：

- 虚拟链接可以用来连接一个源终端系统和一个或多个目标终端系统。航电工程师可以根据航电系统的通讯需要自由调整每条虚拟链接的最大带宽。

- 虚拟链接的带宽被设定在终端系统上，一旦被设定，其他在终端系统上使用该虚拟链接的应用是无法改变它的带宽的。

虚拟链接的具体算法如下所示：

```
returned=removeQueue(&(env->vl_queue),(Queueable)&(env->vl_m),NULL);pthread_mutex_unlock(&(env->vl_mutex));  
  
if (returned)  
{  
    sem_post(&(env->vl_full));  
  
    pthread_mutex_lock(&(env->vl_mutex));  
    removeQueue(&(env->vl_queue),(Queueable)env->vl_m.m.rd.data,&env->vl_m.m.rd.length);  
    pthread_mutex_unlock(&(env->vl_mutex));  
    sem_post(&(env->vl_full));  
    memset(&local_addr, 0, sizeof(struct sockaddr_in));  
    memset(&server_addr, 0, sizeof(struct sockaddr_in));  
    local_addr.sin_family = AF_INET;  
    local_addr.sin_addr.s_addr = INADDR_ANY;  
    local_addr.sin_port = env->vl_m.source_port;  
    server_addr.sin_family = AF_INET;  
    memcpy(&server_addr.sin_addr, &env->vl_m.ip_address, sizeof(in_addr_t));  
    server_addr.sin_port = env->vl_m.dest_port;  
    bind(sockfd, (struct sockaddr *) &local_addr, sizeof(local_addr));  
    /*We prepare the message*/  
    memmove(env->vl_m.m.rd.data+ sizeof(unsigned int) + sizeof(size_t), env->vl_m.m.rd.data, env->vl_m.m.rd.length);  
    memcpy(env->vl_m.m.rd.data+sizeof(unsigned int),&env->vl_m.m.rd.length, sizeof(size_t));  
    memcpy(env->vl_m.m.rd.data, &env->vl_m.m.sequence, sizeof(unsigned int));  
    int senddatasize=0;  
    /* sendto() send env->vl_m.m.rd.data to sockfd  
    senddatasize=sendto(sockfd, env->vl_m.m.rd.data, sizeof(unsigned int) + sizeof(size_t) + env->vl_m.m.rd.length, 0,(struct sockaddr *)&server_addr,sizeof(server_addr));
```

```
if (senddatasize==-1)
{
    printf("send data size -1!\n");
};
sendcount++;
};
```

算法首先从控制虚拟链接信息存储区域的互斥锁，并将存储在虚拟链接队列中的数据复制到虚拟链接信息中，同时，虚拟链接绑定 UDP 套接字，用于将虚拟链接数据发送给目的终端系统实现通讯。

而对于每一个虚拟链接而言，其链接内传输的每一个数据帧都被单独赋予一个序列号，该序列号范围从 0 开始，至 255 结束。对于连续的数据帧，他们的序列号每次增加一。当序列号为 255 时，下一个数据帧的序列号将回到 1。这样做的好处是一旦有数据帧丢失，目的终端系统可以清楚的掌握这一情况。值得注意的是，第一个数据帧的序列号从 1 开始而非 0，序列号 0 只有在传输设备重启的时候才被发送^[24]。

2.1.4 网络交换机

传输过程中，每条虚拟链接的上限带宽是不能被突破的。这项工作由交换机进行检查，而相应的参数则被保存在交换机上。带宽的保障工作由交换机和终端系统共同完成。而交换机可以利用设置，来对于多种虚拟链接开展优先级的设置工作，其中的核心信息需要优先保证。

交换机还可以通过流量整形和流量调度来控制网络内的流量，以防止过多的数据造成网络拥塞。

为了避免流量争夺有限的带宽，交换机使用缓存来存储一定量的数据。如果某个终端系统需要接收来自多个终端系统发来的信息，交换机可以分配一定的缓存资源保证该终端系统有足够的接收能力接收并处理这些信息，防止因处理能力不足丢失数据。

2.2 AFDX 网络拓扑

以太网技术最初并没有在航空领域广泛应用。它最早被用来连接地面的各种系统和设备，如航空流量控制系统、天气预报广播系统和地面导航系统。

而 AFDX 的出现改变了这一切。正如前面提到的，已有多款民用客机正在使用或将要使用 AFDX 作为它们的数据总线规范。AFDX 也为航电系统的架构改进指明了方向，航电系统由先前单一的总线式架构分布向更复杂、更高效的星型架构发展。

如图 2-5 所示，图中有两个通讯域，分别是通讯域 A 和通讯域 B。每个通讯域都配置域内的网络交换机，负责通讯域内的终端系统间的通讯。同时，这两台网络交换机彼此相连负责连接两个通讯域。同一个通讯域的终端系统通讯时，通过域内交换机就可以完成数据传输实现通讯，不同的通讯域内的终端系统通讯时，源终端系统将数据发送至所在域的网络交换机，再由该交换机将数据传递至另一个通讯域的交换机，最后由另一个域的交换机将数据发送至对应的目标终端系统完成通讯。

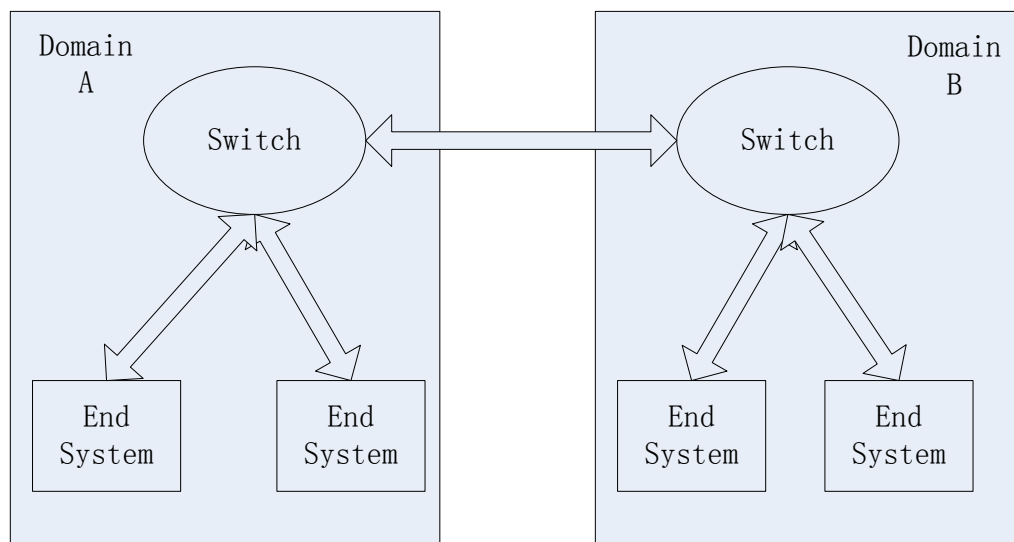


图 2-5 AFDX 星型拓扑^[23]

Figure 2-5 AFDX Star Topology

新的拓扑结构为 AFDX 网络更高的可靠性带来了可能。AFDX 网络可以通过冗余的通讯架构来实现高可靠性，如交换机集群、双逻辑链路等。如图 2-6 所示，该通讯架构就使用了双逻辑链路来保证通讯可靠性。循环冗余校验码也被用来保证数据的完整性。

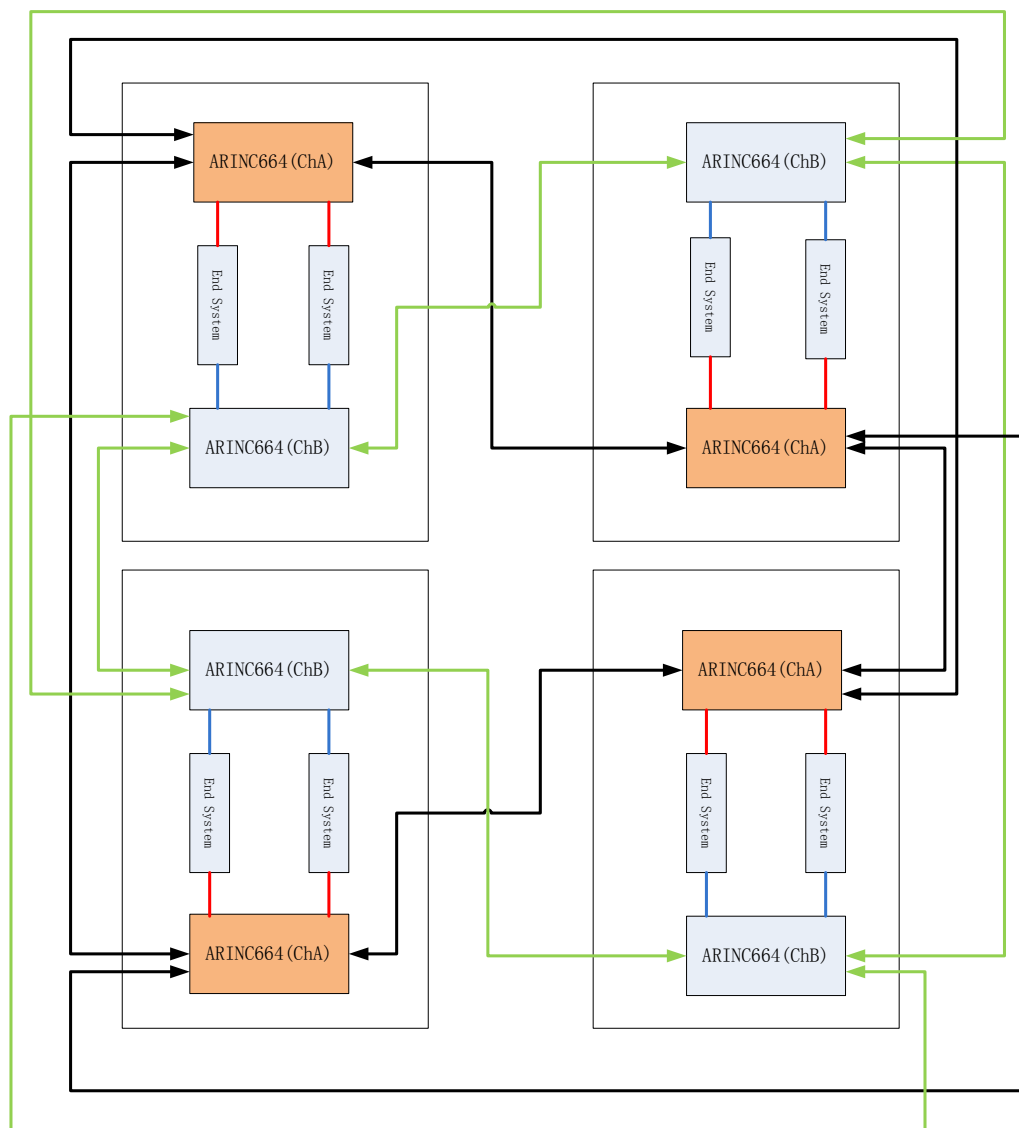


图 2-6 AFDX 冗余链路示意图^[17]

Figure 2-6 AFDX Redundancy Link

在冗余的通讯链路中，两条数据链路被用来同时传输数据^[25]。如图 2-7 所示。BAG (Band Allocated Gap)是 AFDX 网络内数据帧传播的最小时间单位，即在没有网络抖动的情况下，同一个虚拟链接内的相邻两个数据帧的第一个字节传播的时间间隔。而对于每一个确定的虚拟链接，终端系统应该在自身的配置表内保存一个确定的 BAG 值。BAG 的单位为毫秒，值从 1 到 128。通过对终端系统的设置，BAG 的值应同时满足是 2 的指数，即 $BAG=2^k$ (k 为从 0 到 7 的整数)^[17]。BAG 的实现方法如下所示。

```
memset(&t.tv_sec, 0, sizeof(time_t));

/* To convert nanoseconds into milliseconds*/

t.tv_nsec = vls[id].bandwidthAllocationGap * 1000000;
```

```

aux = &t;
intime = false;
do{
    if(nanosleep(aux , &rem)==-1)
    {
        aux = &rem;
    }
else
    {
        intime = true;
    }
}while(!intime);

```

在图 2-7 中，第一个 BAG 内，网络 A 和网络 B 同时传输数据，网络 A 传输的数据标记为 A1，网络 B 传输的数据为 B1，由于 A1 早于 B1 到达目标地址并完成数据完整性检查，因此最终得到的数据是 A1。同理，第二个 BAG 期间得到的数据是 A2。需要注意的是，在第四个 BAG 期间，A 网络的数据丢失，而 B 网络的数据顺利到达目的地，因此得到的数据是 B4，从中可以看出冗余设计在 AFDX 网络的使用中是非常必要的。

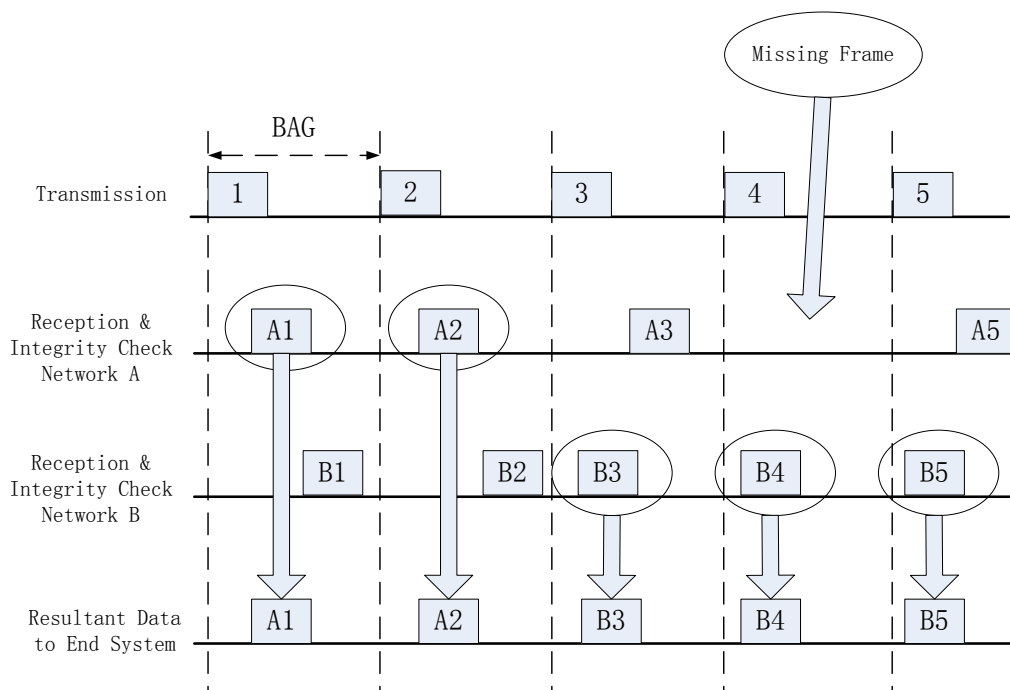


图 2-7 AFDX 冗余链路的数据传输^[17]

Figure 2-7 AFDX Redundancy Link Data Transmission

2.3 AFDX 网络协议及通讯端口

AFDX 网络的协议示意如图 2-8 所示。

在 AFDX 网络中，航电终端系统利用用户数据报协议进行数据交换。用户数据包协议（UDP, User datagram protocol）是建立在传输层的协议。在数据交换过程中，目标终端系统的 MAC 地址被用来路由网络中的数据帧。同时，一些被用于维护的协议如简单文件传输协议（TFTP, Trivial File Transfer Protocol）、简单网络管理协议（SNMP, Simple network management protocol），也是基于用户数据包协议的。

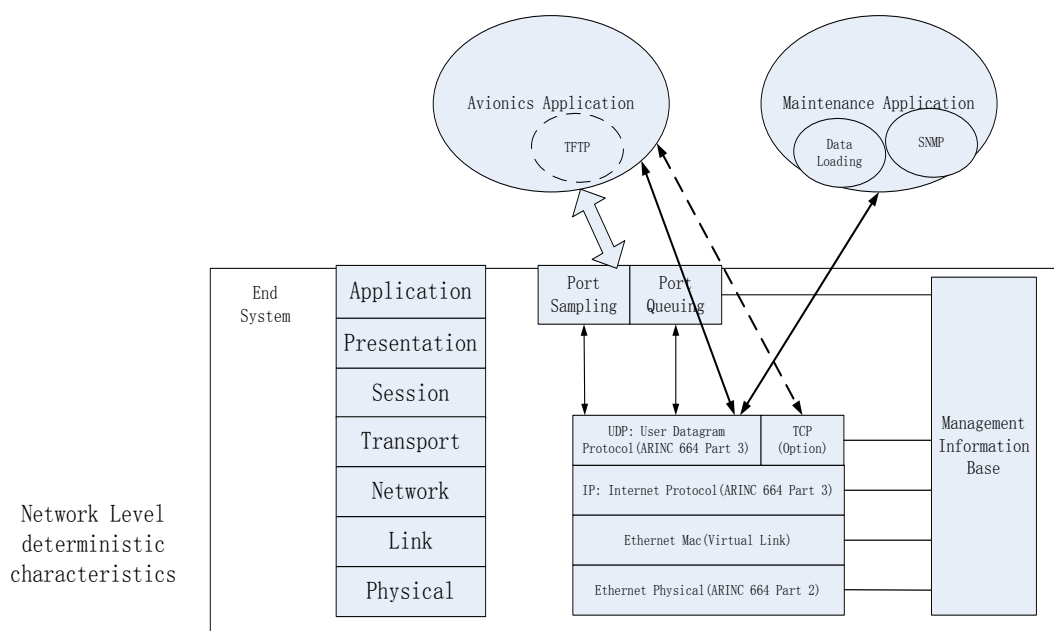


图 2-8 AFDX 协议示意图^[17]

Figure 2-8 AFDX Services Protocol

在图 2-8 中，还有两个通信端口 Port Sampling、和 Port Queuing。这两个通信端口就是 AFDX 网络中的采样型端口和队列型端口。航电系统利用 AFDX 网络通讯端口进行通信。以上两种通信端口由 ARINC 653^[18]定义。

这两种通信端口的主要区别在于接收信息的方式。采样型端口（图 2-9）把接收到的信息存储在缓存中，一旦新信息到来，该缓存将被覆盖，在此之前，即使该信息已经被其他应用读取过，仍然被保存在缓存中，因此该信息可以被反复读取。出于这个原因，该信息应当有一个可更新的标志，用来判断航电分系统接收到的信息是否是同一个。

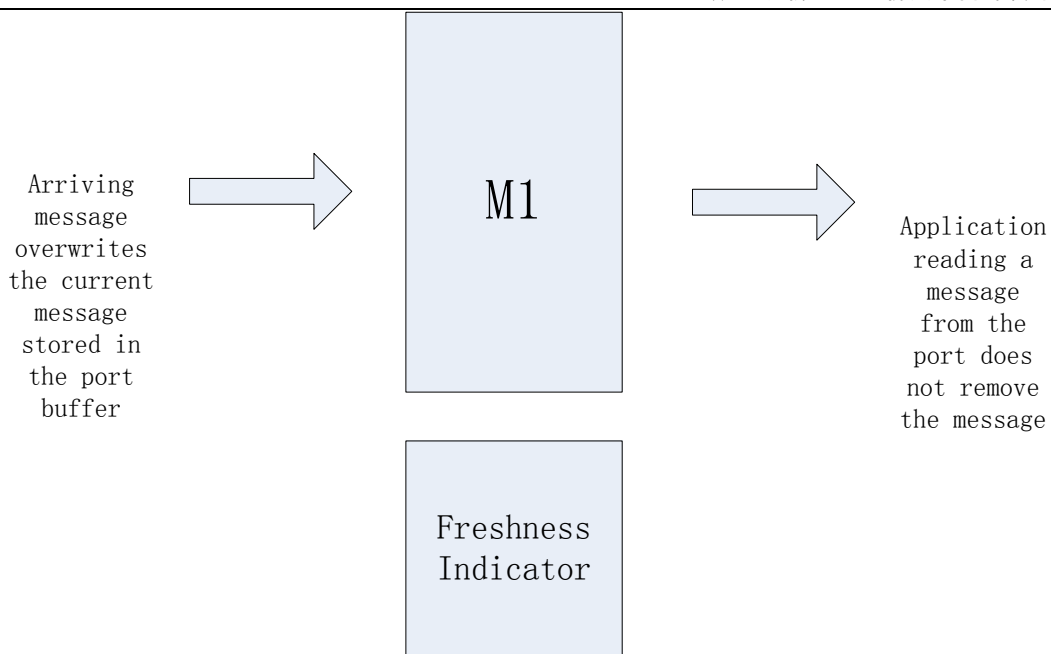


图 2-9 采样型端口通讯示意图^[18]
Figure 2-9 Data Exchange of Sampling Port

另一边，队列型端口（图 2-10）需要足够大的缓存来存储信息，新信息被排入队列，读取后就被移除。该操作严格按照 FIFO 原理进行。

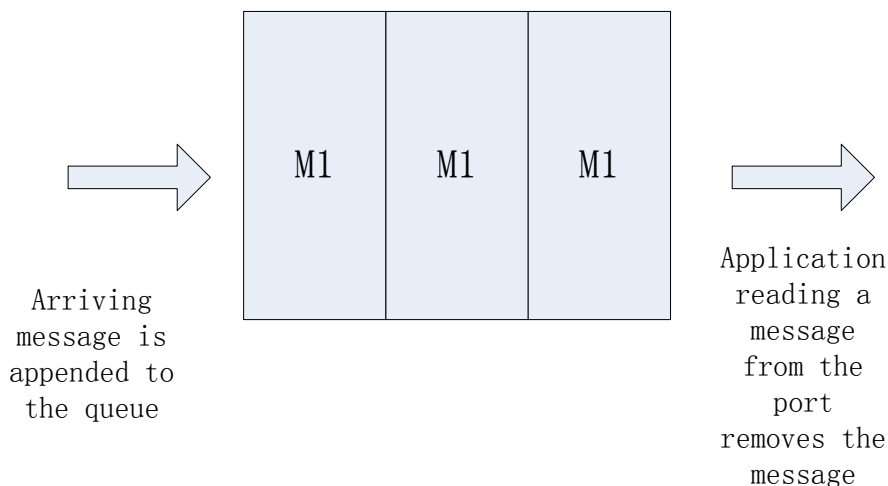


图 2-10 队列型端口通讯示意图^[18]
Figure 2-10 Data Exchange of Queuing Port

2.4 网络通讯过程

AFDX 利用 802.3 以太网技术，实现了前辈总线规范没有实现的全双工通讯模式。其网络传输介质可以采用具有两对铜线的线缆。这两对铜线具有不同的功能，一对铜线用于发送数据，另一对用于接收数据。

交换机有两种缓存，一种是发送缓存，用于存储将要被发送的数据；一种是接收缓存，用于存储接收到而尚未及时处理的数据。所有接收到的信息都带有各自虚拟链接赋予的序号，该序号被用于保证接收数据的顺序，由交换机的处理器进行检查。当信息的序号正确时，该信息将被发往对应的发送缓存等待被发出（如图 2-11 所示）。数据通过源终端系统被从对应的虚拟链接中发送出来，经由交换机的传输，被目的终端系统接收并处理。

因此，AFDX 仿真平台需要模拟从源终端系统至目标终端系统的全过程，包括数据包的生成、发送、接收、处理。而本次的仿真平台模拟的终端系统大于两台，因此通过网络交换机对其进行连接。交换机部分的功能直接由设备本身提供，不在仿真范围之内。

图 2-12 则总结了 AFDX 网络在通讯过程，其保障通讯质量的各种手段如下：

- 流量整形：通过对终端系统配置表的设置，保证其只传输在固定带宽下的特定的虚拟链接。
- 流量控制：网络交换机预先对要传输的虚拟链接的带宽和延迟进行设置，这些设置被保存在交换机的配置文件内，任何不符合预先这些设置的数据都将被丢弃。
- 数据接收：目标终端系统通过端到端的循环冗余码校验来验证接收到的数据，第一个被接收到的合格数据将被采用。
- 网络交换机：用来防止无效数据、错误的虚拟链接。
- 通讯过程：通过交换机缓存和给数据包编号来消除数据包相互间挤占资源。通过虚拟链接来实现通讯的最小可用带宽和最大传输延时。保证数据按照其被发送的顺序进行接收，同时保证数据不丢失。

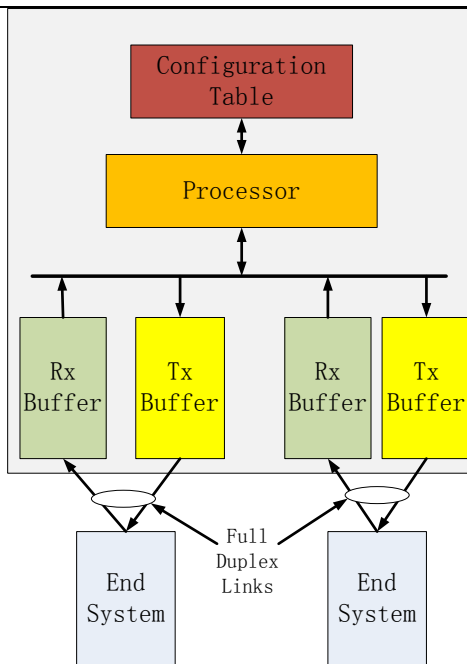


图 2-9 AFDX 网络交换机内部通讯示意图^[17]

Figure 2-11 Transmission In AFDX Switch

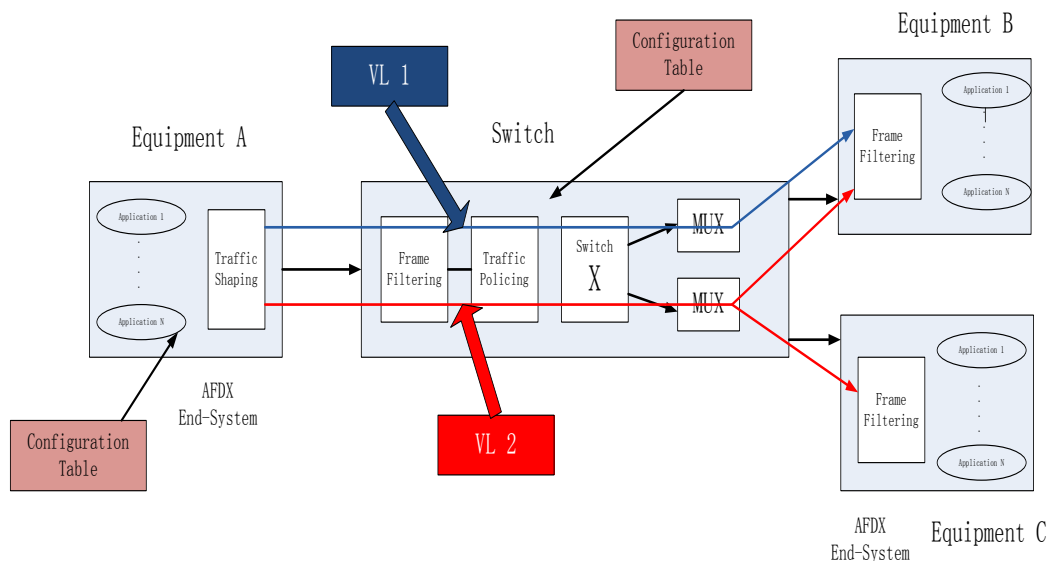


图 2-10 AFDX 通讯示意图^[23]

Figure 2-12 Communication of AFDX

2.5 总结

本章介绍了 AFDX 网络的基本概念，包括其三个基本组成部分：终端系统、网络交换机、虚拟链接。对 AFDX 网络的数据帧格式、虚拟链接的详细定义，AFDX 网络架构和冗余设计及其重要性，列举了 AFDX 网络的基本协议和通讯端口，介绍了 AFDX 网络的通讯过程及保障措施。

3 航空机载 AFDX 仿真平台需求分析与架构设计

本章将分析航空机载 AFDX 仿真平台的需求，归纳总结出该仿真平台的总需求。同时根据总需求的结果分析设计出仿真平台的架构，并在此基础上细化各功能模块的需求。

3.1 航空机载 AFDX 仿真平台分析流程

航空机载 AFDX 仿真平台是为实现对 AFDX 网络通讯行为的模拟，通过硬件平台的搭建，进而实现 AFDX 网络的通讯模拟。仿真平台的硬件架构示意简图如图 3-1 所示。

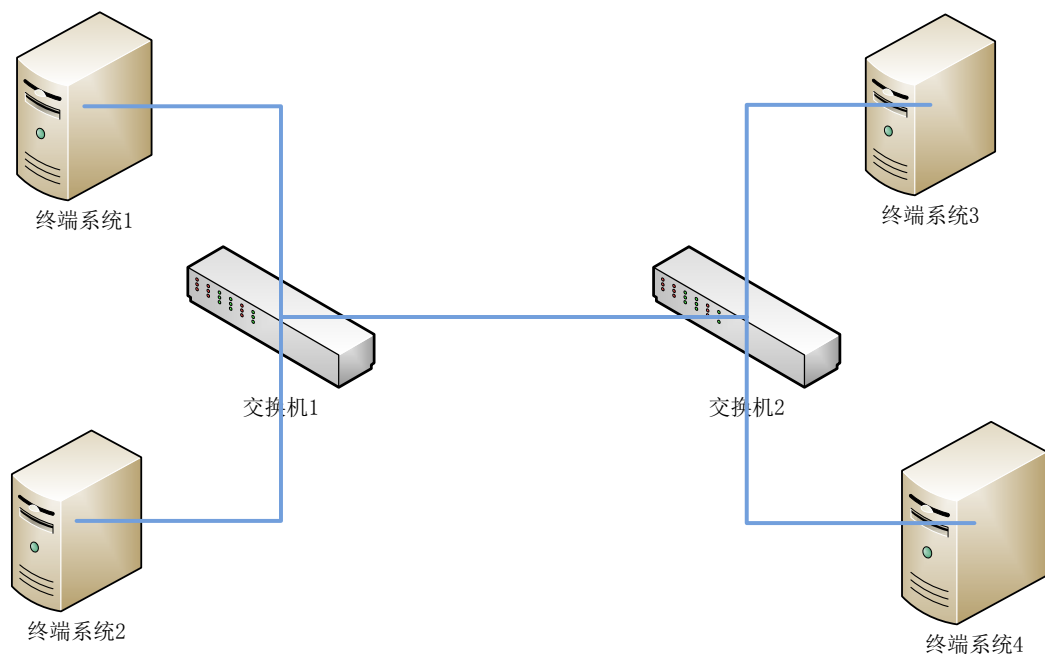


图 3-1 AFDX 硬件架构示意简图
Figure 3-1 Chart of AFDX Hardware Architecture

如图 3-1 所示，图中有 4 个终端系统，分别为终端系统 1 至 4。交换机 1 和交换机 2 用于连接这 4 个终端系统。在本架构图中，当终端系统 1 模拟信号发送的航电系统时，其他的终端系统模拟接收信号的航电系统。根据实际需要，终端系统 2 至 4 也可以依次模拟信号发送的航电系统，同样的，其他的终端系统被用来模拟接收信号的航电系统。

- (1) 配置加载：如前文所述，终端系统需要获取必要的配置信息，以保证仿真平台能够正常加载虚拟链接参数的配置。终端系统还需要加载发送端配置参数、接收方配置参数。
- (2) 仿真平台主程序：该部分主要仿真 AFDX 网络的通讯行为，包括数据包接收后

的包序号发放、轮询调度、分配虚拟链接、发送数据包等。

- (3) 数据包生成及发送: 在正确加载完配置后, 源终端系统需要生成并封装数据包, 按照配置表中的配置, 将数据包发送至对应的虚拟链接。并由虚拟链接发送至对应的目标终端系统。
- (4) 数据包接收: 目标终端系统开启数据包监听, 接收来自源终端系统发送的数据包并予以反馈。通讯结束。

仿真平台发送数据和接收数据的流程如图 3-2、3-3 所示。

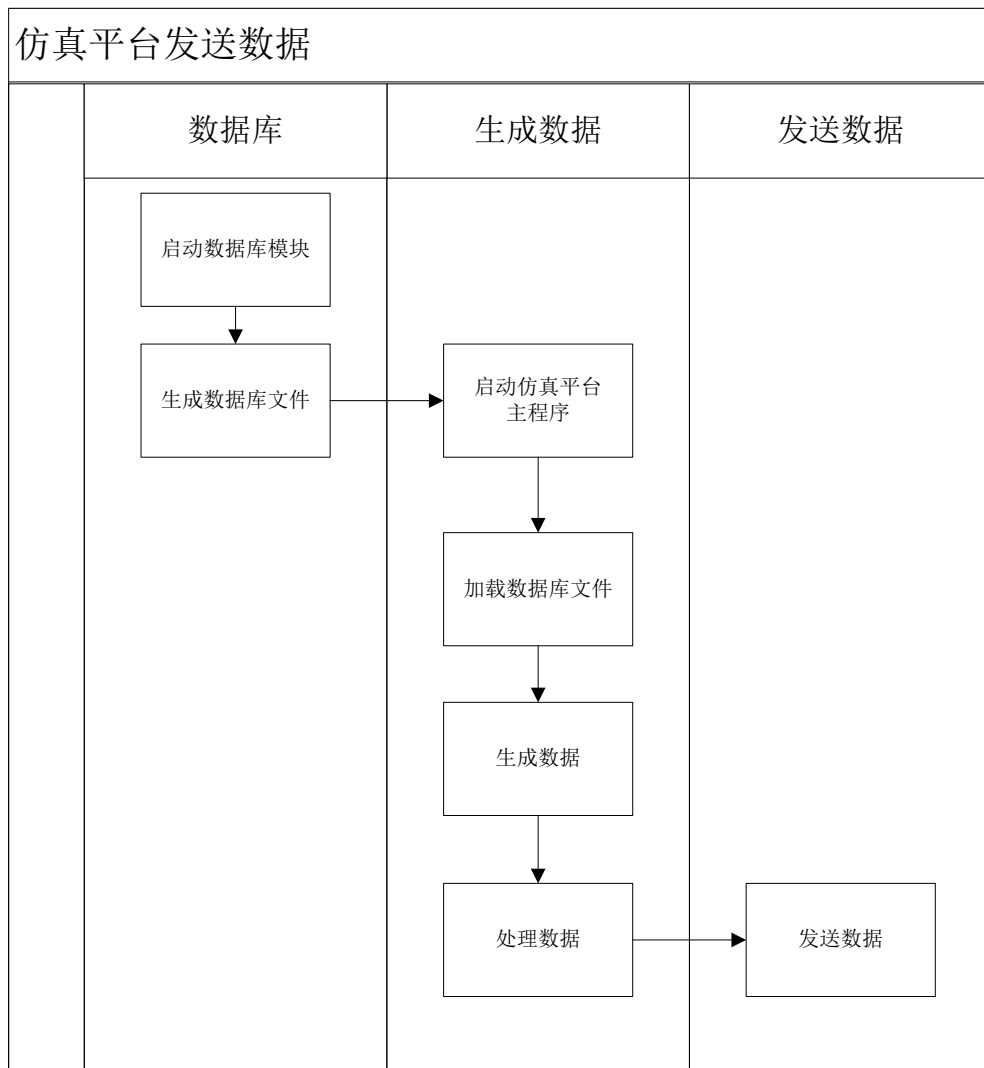


图 3-2 仿真平台数据发送流程
Figure 3-2 Simulation platform Data Transmission Process

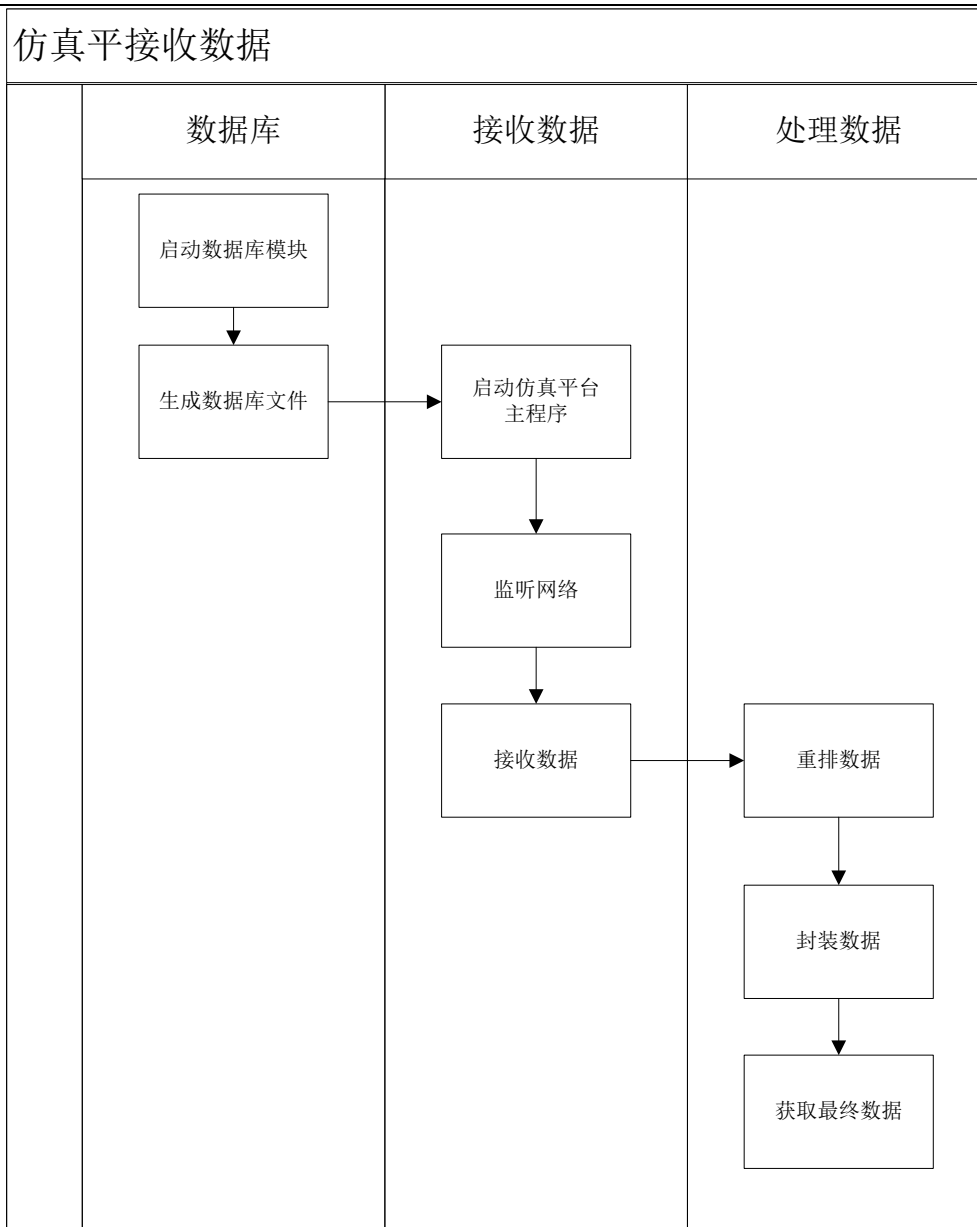


图 3-3 仿真平台数据接收流程

Figure 3-3 Simulation platform Data Reception Process

3.2 系统总体需求

AFDX 仿真平台需求主要是以下三点：

- 以常见的商用设备搭建仿真平台所需的硬件系统，软件选择也仅限于非商业软件。该平台的仿真基于 Fedora 系统。
- 该仿真平台能够仿真 AFDX 网络的通讯行为，即从数据包的生成到数据包的接收的全过程。
- 该仿真平台可以被部署在由两台 HP Procurve 4000M 交换机及多台计算机构成

的网络拓扑中。该拓扑将被用于仿真平台调试及通讯过程模拟。同时，该仿真平台应具有可移植性，能够被用于未来航电系统的研究。

结合仿真平台的需要，仿真平台的应具备如下功能：

- 仿真软件平台能够在 Fedora 操作系统环境下运行。
- 可以通过配置文件实现仿真平台的配置设置。如各仿真设备的通信端口号、虚拟链接大小等。
- 实现包括 IP 地址管理、传输信息管理、虚拟链路轮询、数据分包、数据包序号管理和队列管理在内的功能。
- 能够实现仿真平台数据文件加载管理，可以加载 IP 地址配置、端口信息配置等。
- 能够实现仿真过程可跟踪，可以记录包括错误、问题、警告及正常仿真过程中的全部信息。

3.3 功能模块分析

本节将在仿真平台总需求的基础上详细分解总需求，得出各模块的需求，并以此作为各模块功能的设计依据。

3.3.1 仿真平台数据库模块

仿真平台的数据库模块主要体现在平台加载的配置信息上，例如源终端系统配置参数、虚拟链接配置参数、目的终端系统配置参数等方面。

在数据库的维护方式上，有手动维护和自动维护两种方式。手动维护即手动修改并保存数据库文件。自动方式即启动程序自动生成数据库文件。手动维护的坏处在于这些数据库文件可能因人为或其他原因被篡改或丢失。一旦数据库加载出现错误，将影响整个仿真平台的正常运行。自动方式则避免了这一问题，数据库文件由仿真平台启动数据库模块自动生成。这些生成数据库文件的代码被预先写入程序，仿真平台运行前，启动数据库模块生成配置文件。仿真平台将生成的数据库的信息直接显示在屏幕上，方便用户检查生成的配置。

3.3.2 仿真平台主程序模块

仿真平台主程序模块的主要功能为处理并发送的数据发送功能模块、接收并处理数据至目标终端系统的数据接收功能模块。

发送功能模块是仿真平台主程序模块一个非常重要的功能模块，主要包含以下两个功能：模块的初始化和数据包发送功能。

模块的初始化功能包括以下几点：

- 利用加载配置文件功能从配置文件中加载配置参数。
- 通过读取数据库中的数据获取网络通讯的数据。
- 初始化信息队列、信号量和互斥锁。
- 为环境变量分配缓存。
- 运行和管理模块内的子模块。

发送功能模块主要管理下面三个子模块：

- 序号分配子模块：根据限定的参数封装数据包，并在数据包内加入序号，并把数据包排入对应的子虚拟链接缓存中。
- 轮询调度子模块：利用轮询算法，把数据包从子虚拟链接缓存发送到对应的虚拟链接缓存。
- 虚拟链接子模块：根据预先设定的虚拟链接参数，将数据包发送至目的终端系统。

接收功能模块通过发送功能模块启动，提供 AFDX 网络的数据包接收功能。该功能模块通过网络套接字接口和网络内其他终端系统通讯。该模块把接收到的来自其他终端系统的数据包存储在对应的缓存中，通过处理后数据包发送至信息接收模块。

接收功能模块是仿真平台主程序模块另一个重要的功能模块，主要包括以下两个功能：模块初始化功能和数据包接收功能。

模块的初始化功能包括以下几点：

- 利用加载配置文件功能从配置文件中加载配置参数。
- 通过读取数据库中的数据获取网络通讯的数据。
- 初始化信息队列、信号量和互斥锁。
- 为环境变量分配缓存。
- 运行和管理模块内的子模块。

接收功能模块主要管理以下两个子模块：

- 数据重排子模块：分析接收到的数据包并把数据包发送至对应的数据封装子模块。
- 数据封装子模块：将收集到的数据包合并成完整的信息，并把信息发送至对应的信息队列端口。

3.3.3 仿真平台信息发送模块

信息发送模块仿真发送数据的终端系统，将数据包发送至主程序模块。模块在发送数据的过程中显示发送数据包的大小、源地址、发送的源端口，同时在程序页面中显示每个数据包是否发送成功。

3.3.4 仿真平台信息接收模块

信息接收模块仿真实接收数据的终端系统，接收来自主程序模块的数据包。模块在接收数据的过程中显示发送数据包的源地址、接收的端口，同时在程序页面中显示每个数据包是否接收成功。

3.3.5 仿真平台日志模块

日志模块用于记录仿真平台工作信息，数据包的接收、处理、发送等。每个日志文件都存储在各终端系统上。

3.4 系统架构设计

AFDX 仿真平台的系统架构如图 3-4 所示。

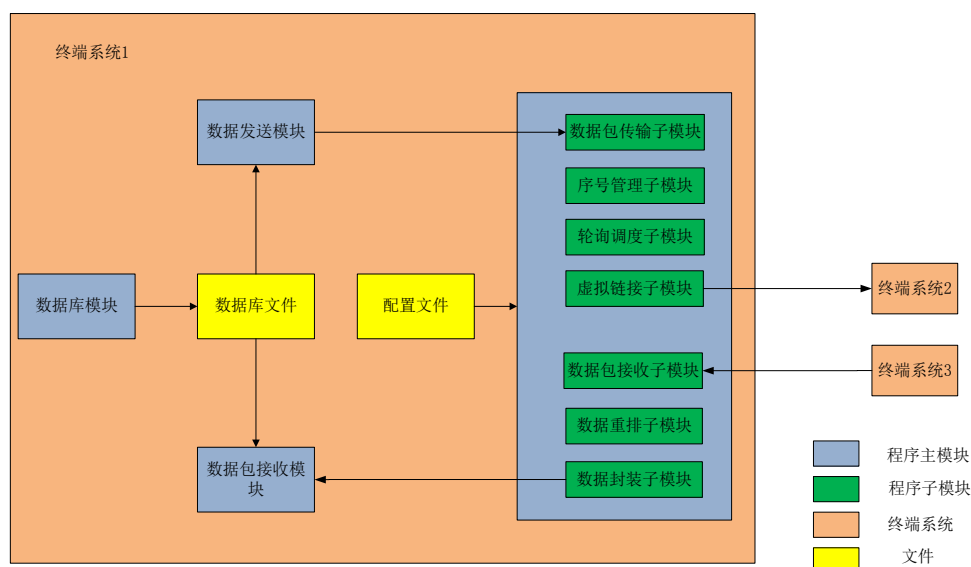


图 3-4 系统架构

Figure 3-4 System Architecture

3.5 本章小结

本章对航空机载 AFDX 仿真平台的总需求进行了归纳总结，并在总需求的基础上进

行了 AFDX 仿真平台的架构设计。本章首先收集了航空机载 AFDX 仿真平台的总需求，并根据总需求进行了仿真平台的架构设计。进而在仿真平台架构的基础上对功能模块进行了细化，完成了平台各功能模块的分析。

4 航空机载 AFDX 仿真平台模块详细设计与实现

本章完成了航空机载 AFDX 仿真平台各功能模块的设计与实现。本章将根据航空机载 AFDX 仿真平台的各功能模块的业务需求,实现 AFDX 仿真平台。仿真 AFDX 网络的通讯行为,平台为基于 Fedora 系统下的 C 语言的网络编程。

4.1 数据库模块设计与实现

前文提到,航空机载 AFDX 仿真平台主程序模块启动后,需要加载相应的数据库文件,以读取源终端系统、目的终端系统、虚拟链接等配置参数的信息。维护这些数据库的工作是繁琐的,且手动操作数据库存在误操作的可能性。一旦参数加载失败,会导致主程序模块无法正常工作,进而影响整个仿真平台的仿真行为。

出于以上考虑,仿真平台数据库模块集成了数据库生成功能。程序预先将需要生成的数据库文件定义在数据库模块中。启动数据库模块的同时,数据库文件也同步生成,并将生成的信息反馈在屏幕上,便于用户检查。

为了实现数据库模块,由程序 main5.c,dbm.c,dbm.h 三个文件实现。其中 main5 通过调用 dbm.c 和 dbm.h 实现数据库文件的创建。

系统设计了数据库 API,完成数据库的建立、读取等操作。这些 API 的设计如表 4-1。

表 4-1 数据库操作 API
Table 4-1 API of Database function

编号	API	描述
1	writeSourceDB	读取源终端系统的参数并将其存入源终端系统数组备用
2	readSourceDB	读取源终端系统数组
3	writeDestDB	读取目的终端系统的参数并将其存入目的终端系统数组备用
4	readDestDB	读取目的终端系统数组
5	writeVLDB	读取虚拟链接的信息并将其存入虚拟链接数组
6	readVLDB	读取虚拟链接数组
7	createTxerSourceDB	读取源终端系统数组内的数据,建立源终端系统数据库文件
8	createTxerVLDB	读取虚拟链接数组内的数据,建立源终端系统的虚拟链接的数据库文件
9	createTxerDestDB	读取目的终端系统数组内的数据,建立源终端系统的目的地数据库文件
10	createRxerDestDB	读取目的终端系统数组内的数据,建立目的终端系统的目的地数据库文件

4.2 仿真平台主程序模块设计与实现

主程序模块是航空机载 AFDX 仿真平台的重要模块，提供 AFDX 仿真平台的主要功能，对 AFDX 网络通讯行为的仿真。

主程序模块按照功能划分可以分为数据包处理发送、序号管理、轮询调度、虚拟链接、数据包接收处理、数据包重排、数据包封装等子模块。

- 数据包处理发送：加载配置文件、读取数据库文件、为环境变量分配内存空间、接收来自数据包发送模块传递的数据包。
- 序号管理：将接收到的数据按照预先设定的网络允许传输上限进行拆分，并为每个拆分后的数据包分配序号。
- 轮询调度：启用轮询调度、将数据包分配对应的虚拟链接。
- 虚拟链接：虚拟链接按照数据包内的目的地址发送数据。
- 数据包接收处理：加载配置文件、读取数据库文件、为环境变量分配内存空间、收集数据包。
- 数据包重排：重排数据包，将数据包按顺序发送至对应的数据封装程序。
- 数据包封装：封装数据包以得到完整的信息，将信息发送至数据接收模块。

4.2.1 数据包处理发送

数据包处理发送子模块涉及的操作有：加载配置文件、读取数据库文件、初始化并加载数据包接收处理子模块、为数据包处理发送子模块初始化参数、初始化并加载轮询调度、虚拟链接子模块和序号管理子模块、接收信息发送模块发送的信息并将其传递至序号管理子模块。

收集到的信息包括源终端系统的地址、发送端口、目的终端地址、接收端口、虚拟链接 ID 等。

数据包处理发送子模块的功能是初始化数据包处理发送子模块、数据包序号管理子模块、轮询调度子模块、虚拟链接子模块及相关参数，接收信息发送模块通过消息队列发送过来的数据，并为该数据寻找空闲的数据包序号管理子模块。

首先，数据包处理发送子模块的主函数 `runTxer` 调用 `initializeTxer` 函数为进程初始化运行环境。`InitializeTxer` 函数主要完成了以下操作：

1、使用 API `sigprocmask` 屏蔽“SIGTERM”、“SIGINT”信号，防止初始化运行环境过程中被意外中断。

2、调用 loadConfigurationTxer 函数加载源终端系统配置文件。

3、调用 loadDBTxer 函数。首先，loadDBTxer 函数为源终端系统关键字、源终端系统端口、目的终端系统关键字、目的终端系统端口、虚拟链接关键字、虚拟链接初始化存储空间。其次，在完成存储空间初始化后，loadDBTxer 函数分别读取源终端系统源数据库文件、源终端系统目的数据库文件、虚拟链接数据库文件中的参数，完成数据库文件加载。最后，loadDBTxer 函数调用 atexit 函数，当主程序退出时，atexit 调用 freeDB 函数释放分配的存储空间。

4、调用 setEnvironmentRxer 函数。首先，setEnvironmentRxer 函数为数据包接收处理子模块监听的端口、dispatcher 接收数据、dispatcher 互斥锁、dispatcher 信号量、数据包重排子模块数、数据包重排子模块 ID、rxer 互斥锁、数据包封装子模块 ID 等分配存储空间。其次，setEnvironmentRxer 函数初始化数据包重排子模块信号量、dispatcher 接收数据、rxer 接收数据等。

5、调用 launchRxerTxer 函数。首先，launchRxerTxer 函数新建一个进程号。其次，launchRxerTxer 函数调用 runRxer 函数，完成数据包接收处理子模块、数据包重排子模块、数据包封装子模块的启动，有关 runRxer 函数的具体功能将在介绍数据包接收处理子模块时详细介绍。再次，将新建的进程号赋值给 Rxer 的进程号完成进程的挂起。最后，launchRxerTxer 函数调用 atexit 函数，当主程序退出时，atexit 函数调用 waitForRxer 函数完成数据包接收处理子模块的退出。

6、调用 launchRRSandVLSTxer 函数。首先，launchRRSandVLSTxer 函数新建 roundRobinScheduler 进程和 virtualLinkScheduler 进程。其次，launchRRSandVLSTxer 函数调用 atexit 函数，当主程序退出时，atexit 函数调用 waitForRRSandVLS 完成退出 roundRobinScheduler 进程和 virtualLinkScheduler 进程的退出。

7、调用 setSequencersEnv 函数。首先，setSequencersEnv 函数为数据包序号管理子模块号 sequencer_set 和 sequencer 信号量 sequencer_sem 分配存储空间。其次，setSequencersEnv 函数为 sequencer 接收的数据 petition_tray 和发送至子虚拟链接的数据 fmessage 分配存储空间，初始化 sequencer_sem、fmessage 等。再次，setSequencersEnv 函数初始化数据包处理发送子模块数据存储空间 in_tray 及其互斥锁 in_tray_mutex、sequencer 空闲线程标志 free_sequencer 及其互斥锁 free_sequencer_mutex 并初始化数据包处理发送子模块接收数据的存储空间 txer_petition。最后，setSequencersEnv 函数调用 atexit 函数，当主程序退出时，atexit 函数调用 freeSequencersEnv 释放所有的变量。

8、调用 launchSequencers 函数。函数新建数据包序号管理子模块，并调用 atexit 函数。当主程序退出时，atexit 函数调用 waitForSequences 函数退出 sequencers 线程。

9、调用 openMQTxer 函数。openMQTxer 函数新建消息队列 txer_mq 及其信号量，

该消息队列用于接收来自信息发送模块发送的信息。同样，openMQTxer 函数调用了 atexit 函数用于主程序退出时关闭消息队列及其信号量。

10、步骤 2-9 为数据包处理发送子模块及其相关进程的初始化，在完成上述操作后，initializeTxer 函数重新启用了“SIFTERM”和“SIGINT”信号，保证程序在接下来程序运行的过程中可以正常退出。

其次，runTxer 函数进入 while 循环，在循环中，函数接收消息队列 txer_mq 的数据并将其存储在 txer_petition 中。数据接收完毕后，被保存在 in_tray 中。此时，runTxer 函数轮询数据包序号管理子模块，一旦发现空闲的线程，in_tray 中的数据将交由该空闲线程进行处理。

平台设计了以下 API 来实现数据包处理发送子模块的功能，模块在运行时，根据设计的需要加入对以下 API 的调用，API 的设计如表 4-2。

表 4-2 数据包处理发送子模块 API

Table 4-2 API Of Data management and Transmission Sub-Module

编号	API	描述
1	InitializeTxer	启动数据包处理发送子模块的初始化工作
2	LoadconfigurationTxer	读取并加载程序需要使用的配置
3	loadDBTxer	读取并加载由数据库模块生成的源系统终端、目的系统终端、虚拟链接等的数据库文件
4	setEnviromentRxer	为数据包接收处理子模块需要使用的参数分配内存空间并进行初始化
5	launchRxerTxer	加载数据包接收处理子模块
6	setEnvironmentTxer	为数据包处理发送、轮询调度、虚拟链接三个子模块需要使用的参数分配内存空间并进行初始化
7	launchRRSandVLSTxer	加载轮询调度和虚拟链接子模块
8	setSequencersEnv	为序号管理子模块需要使用的参数分配内存空间并进行初始化
9	launchSequencers	加载序号管理子模块
10	openMQTxer	建立信息发送模块和数据包处理发送模块间的信息队列、信号灯量，用于两个模块间的数据传输

表 4-2 的 API 为 C 实现的本地 API，仿真平台调用该 API 完成接收数据后，应通过序号管理子模块对数据包进行序号管理。

数据包处理发送子模块的流程图如图 4-1 所示。

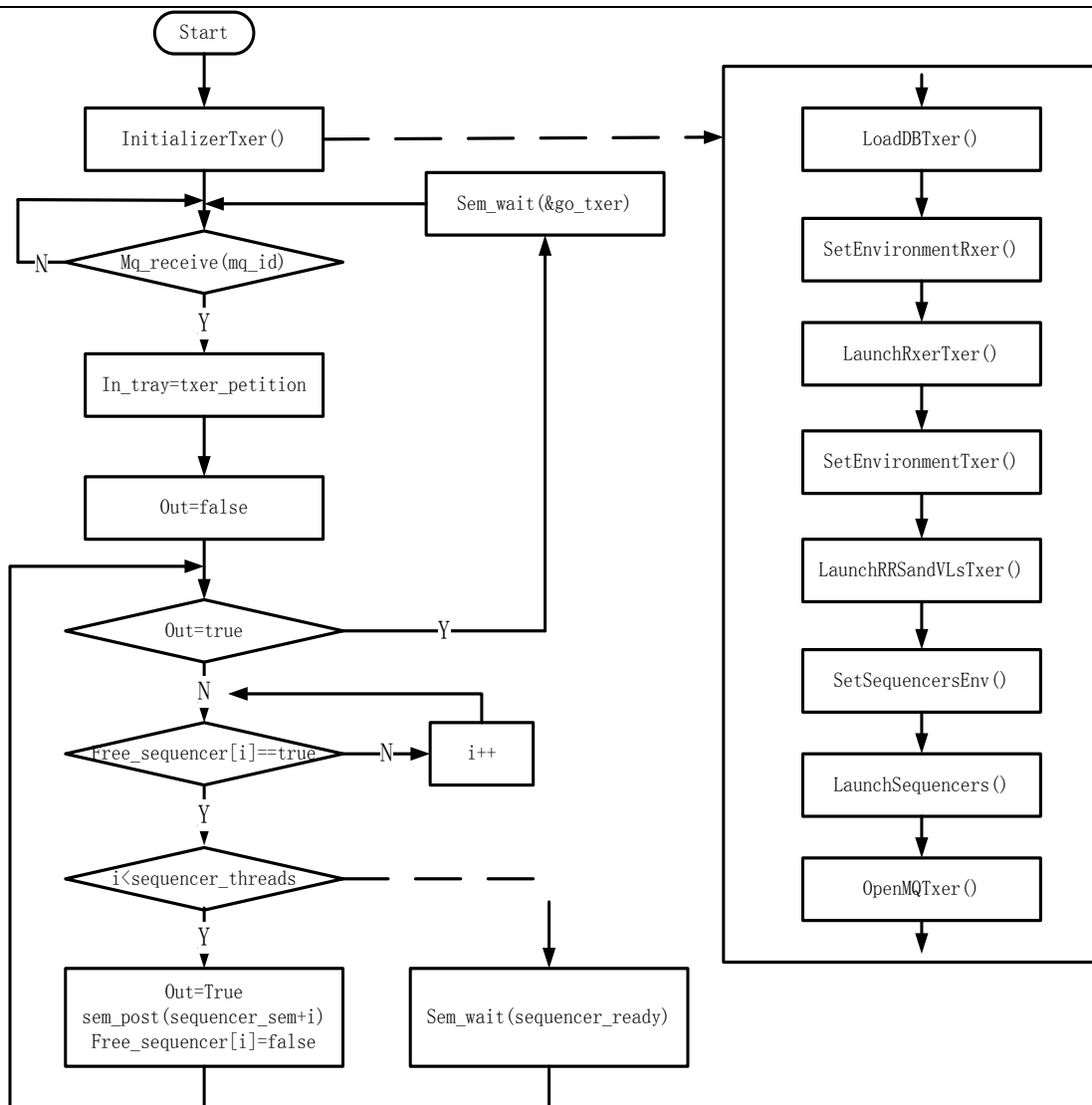


图 4-1 数据包处理发送模块流程图

Figure 4-1 Flow Chart of Data management and Transmission sub-module

4.2.2 数据包序号管理

数据包序号管理子模块接收来自数据包处理发送模块的数据包，将这些数据包按照预先设定的 L_{max} （虚拟链接中被允许传递的数据包的最大值）来拆分数据包，并给每个拆分后的数据包打上一个序号，序号范围从 0-255。为了让程序能够区分被拆分的数据包是否为最后一个数据包，该子模块中定义了特殊的序号 0。一旦判断出该拆分后的数据包是原数据的最后一个拆分数据包，则该拆分数据包的序号为 0。拆分的数据包在获取自己的序号后，将被发送至其所属的子虚拟链接。

首先，数据包序号管理子模块主函数 Sequencer 初始化布尔型变量 out_seq 为 false，该变量用于指示接收的数据 in_tray 是否是最后一个数据。Out_seq 初始化完成后，数据包序号管理子模块进入 while 循环。该循环首先判断 in_tray 的数据是否为最后一个数据，

如果 in_tray 是最后一个数据, 则 out_seq 为 true, 表明已经接收完了最后一个数据, while 循环将在完成这次循环后结束运行。

反之则继续运行该循环, 数据包序号管理子模块将 in_tray 中的数据复制到 petition_tray, 完成数据包序号管理子模块的数据保存。接下来, 数据包序号管理子模块调用 lMax 函数查找接收到数据的源端口 petition_tray.port 相对应的虚拟链接的最大允许传输数据帧大小 lmax。查找完毕后, petition_tray 中的数据将被按照 lmax 的大小分割成 fmessage 进行传输。首先, 数据包序号管理子模块将 petition_tray.port 赋值给 fmessage.source_port, fmessage.source_port 将被用来标示数据帧的源端口用于接下来的传输。其次, 数据包序号管理子模块将数据包 petition_tray 按照 lmax 分割为数据帧 fmessage, 并给每个 fmessage 一个序列号 Sequence, 从 1 开始, 至 255 结束, 并用 0 标示 petition_tray 的最后一个 fmessage。接下来, 数据包序号管理子模块调用 subVirtualLink 函数查找 petition_tray.port 对应的子虚拟连接的指针, 并用 addQueue 函数将 fmessage 数据复制到该子虚拟连接的队列中。最后, 数据包序号管理子模块计算 petition_tray 中的剩余数据大小 remain_bytes, 如果 remain_bytes 不为 0, 表示 petition_tray 中数据尚未全部被分割为数据帧 fmessage, 则用未被分割为数据帧的数据部分覆盖原来的数据包, 供下次分割使用, 此时数据包序号管理子模块重复该循环, 直到 remain_bytes 等于 0 为止。

数据包序号管理模块用到的 API 函数如表 4-3 所示:

表 4-3 数据包序号管理使用的 API
Table 4-3 API of Data Sequence Management Sub-module

编号	API	描述
1	sequencer	序号管理的主程序
2	isEndPetition	用来判断该请求是否是最后一个请求
3	lMax	用来读取最大允许传输的数据包的大小
4	subVirtualLink	找出端口对应的子虚拟链接指向的队列, 并返回其指针
5	addQueue	将数据包添加到相应的队列

数据包序号管理子模块的流程图如图 4-2 所示。

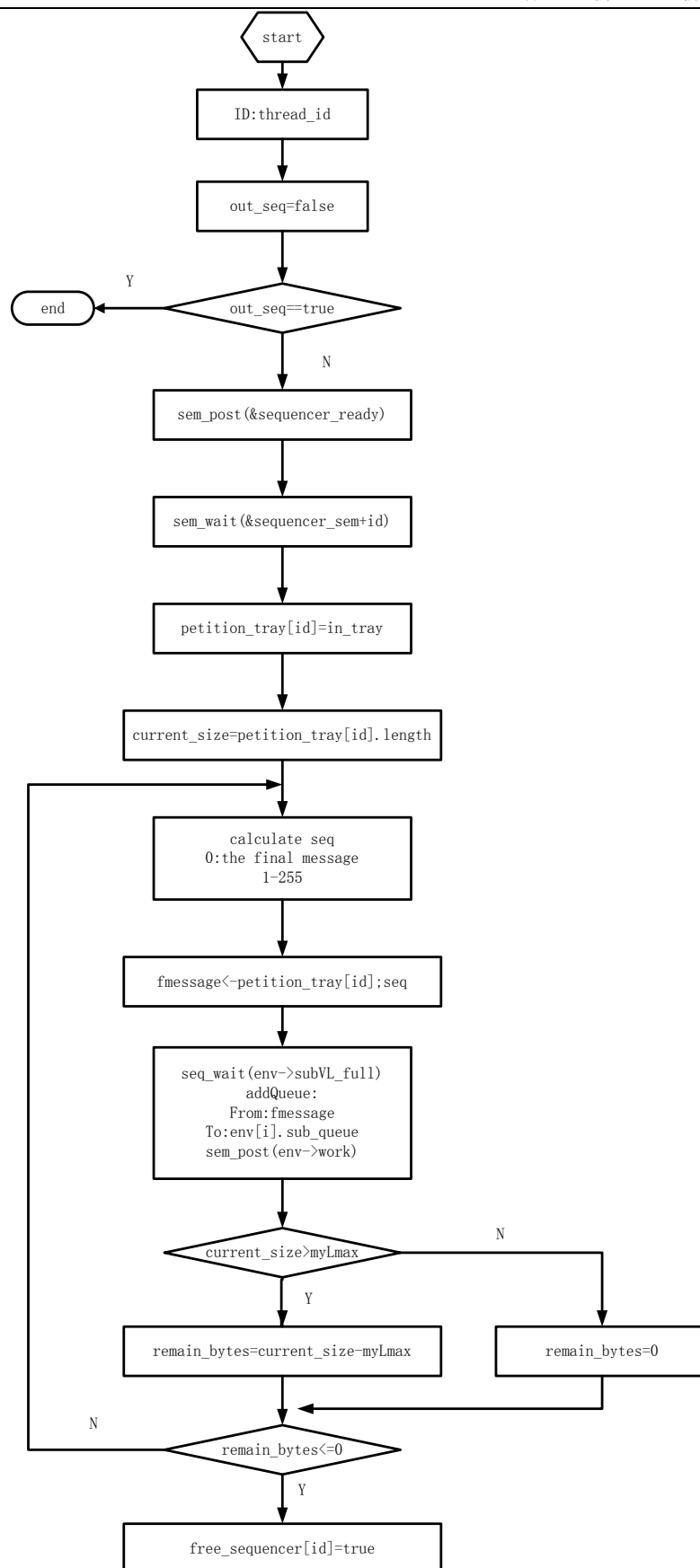


图 4-2 数据包序号管理子模块流程图

Figure 4-2 Flow Chart of Data Sequence Management Sub-module

4.2.3 轮询调度

轮询调度子模块通过轮询调度算法, 将被拆分后的数据包按照正确的顺序, 利用轮询调度算法, 由子虚拟连接发送至虚拟链接。

首先, 轮询调度子模块的主函数 `roundRobinScheduler` 初始化结构体 `env`, 用来保存和虚拟链接线程相关的环境变量。其次, 轮询调度子模块进入 `do` 循环, 信号量 `env-work` 减一, 该信号量用来指示数据队列 `subvl_queues`, 该数据队列用来保存将从子虚拟连接发送至虚拟链接的数据。再次, 轮询调度子模块判断 `subvl_queues` 是否为空, 如果为空, 表明该子虚拟连接已经没有数据, 完成了工作, 这时变量 `count` 增加一; 如果不为空, `subvl_queues` 中的数据将被复制到 `subvl_m` 中。从此, 轮询调度子模块判断 `subvl_m` 是否为最后一个数据, 如果是, 则将 `subvl_m` 的数据发送到对应的虚拟链接中, 变量 `finishing` 为 `true`; 如果 `subvl_m` 不是最后一个数据, 则将来自同一源端口的数据根据目的地址数量进行复制并发送至对应的虚拟链接。当所有的子虚拟连接中最后一个数据都发送完毕并且空闲下来的时候, 表示传输完毕, 循环结束。

轮询调度模块使用的 API 函数如表 4-4 所示。

表 4-4 轮询调度使用的 API

Table 4-4 API of Round Robin Scheduler

编号	API	描述
1	<code>roundRobinScheduler</code>	轮询调度的主程序
2	<code>isEmptyQueue</code>	用来判断该队列是否为空
3	<code>removeQueue</code>	将位于队列头部的数据取出
4	<code>isLastMessage</code>	用于判断接收的是否为原数据包拆分后的最后一个数据包
5	<code>addMessageToVLQueue</code>	将数据包发送至对应的虚拟链接
6	<code>processMessage</code>	将从同一个源终端系统相同端口的数据包进行复制, 发送给不同的目的终端系统。主要用于没有组播功能的设备。

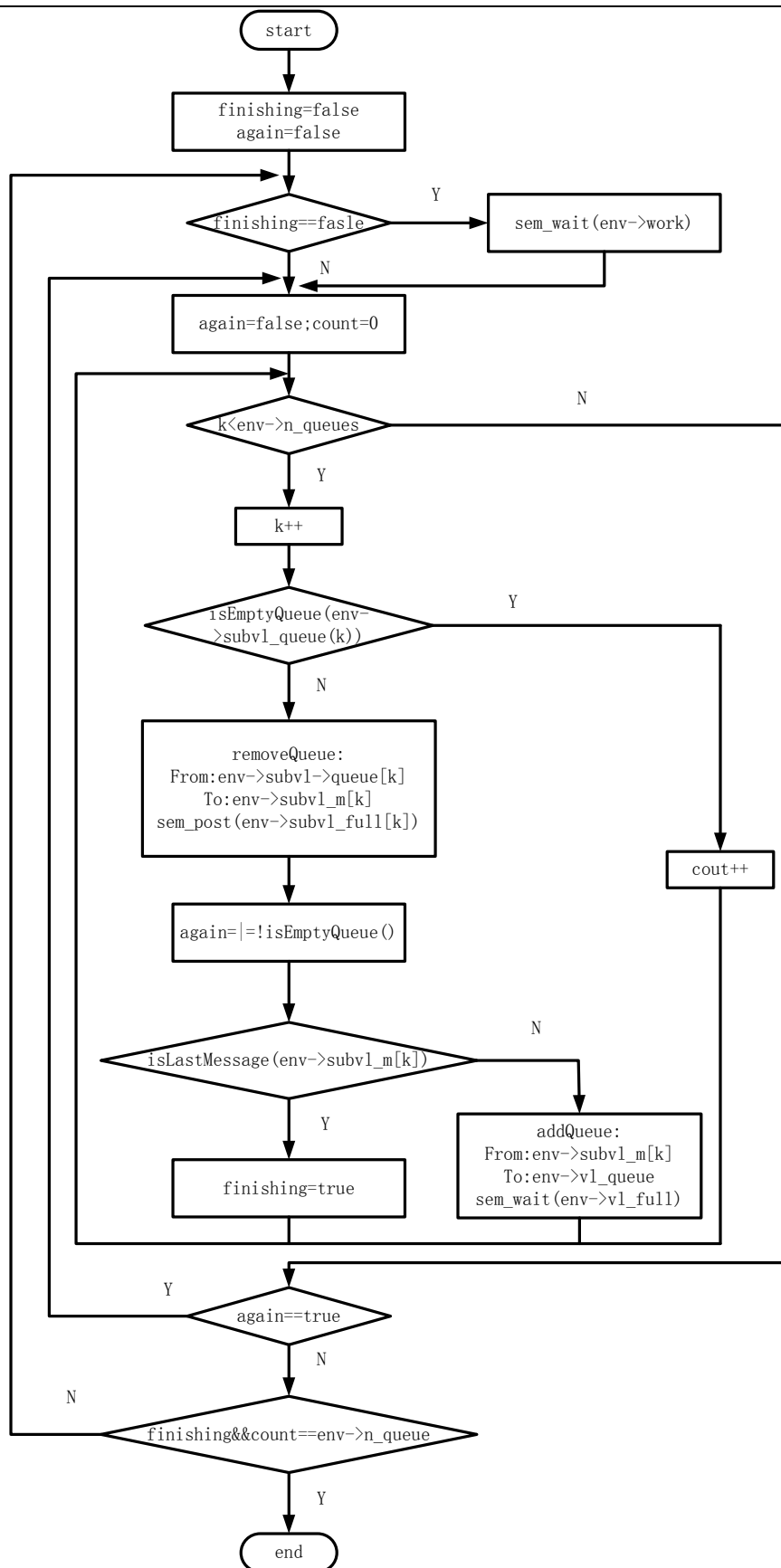


图 4-3 轮询调度子模块流程图

Figure 4-3 Flow Chart of Round-Robin Scheduler Sub-module

4.2.4 虚拟链接

虚拟链接子模块根据预先定义的 BAG 参数进行数据包的发送，它将数据包发送至网络内，由网络设备进行数据包的传递。数据包最终将被发送至目标终端系统的对应端口。

首先，虚拟链接子模块初始化结构体 env，用来保存和虚拟链接线程相关的环境变量。其次，虚拟链接子模块读取对应虚拟链接的带宽传输时间间隔 vls.bandwidthAllocationGap，虚拟链接子模块将按照该时间间隔发送数据帧，同时，虚拟链接子模块创建 UDP 套接字以发送数据帧到目的终端系统。完成上述操作后，虚拟链接子模块将进入 do 循环，直到参数 out 为 true。

在 do 循环中，首先，虚拟链接子模块将虚拟链接中的队列 vl_queue 中的第一个数据复制到数据帧 vl_m 中，如果 vl_m 不是最后一个数据，虚拟链接子模块继续将 vl_queue 中的数据复制到 vl_m 中，并将 vl_m 的数据封装并发送给目的终端系统并执行下一次循环。如果 vl_m 是最后一个数据，则 out 为 true，do 循环结束。

虚拟链接子模块的流程图如图 4-4 所示。

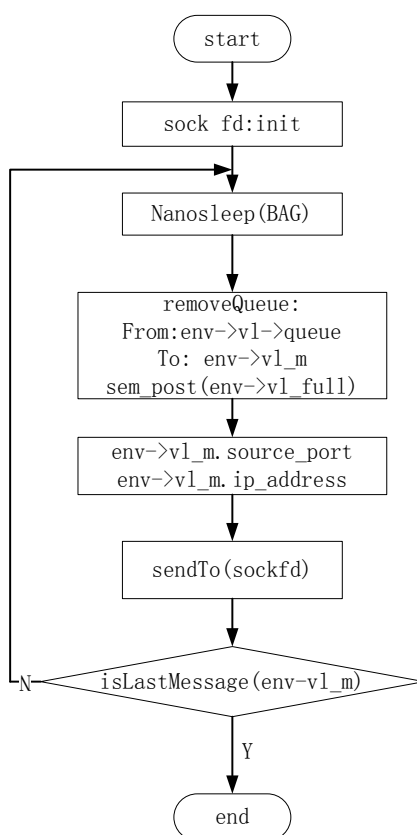


图 4-4 虚拟链接子模块的流程图

Figure 4-4 Flow Chart of Virtual-Link Sub-module

4.2.5 数据包接收处理

数据包接收处理子模块由数据包处理发送子模块通过 `fork()` 调用，实现目的终端系统从网络中收取由源终端系统发送的数据包的功能。

数据包接收处理子模块包含的操作有：读取数据库文件、为数据包接收处理子模块初始化参数并分配内存、初始化并加载数据包重排子模块、创建并加载数据包封装子模块、接收并存储数据包。

首先，数据包接收处理子模块的主函数 `runRxer` 函数主要完成了以下操作：

1、使用 API `sigprocmask` 屏蔽“SIGTERM”、“SIGINT”信号，防止初始化运行环境过程中被意外中断。

2、调用 `loadDBRxer` 函数加载目的终端系统配置文件。首先，`loadDBRxer` 函数为参数目的终端系统关键字、目的终端系统端口初始化存储空间。其次，`loadDBRxer` 函数调用 `readRxerDestDB` 函数从配置文件中读取终端系统关键字、目的终端系统端口的信息并将数值传递到先前初始化的存储空间中。

3、调用 `setEnvironmentRxer` 函数。该函数的作用在数据包处理发送子模块中有详细描述，这里不再赘述。

4、调用 `setDispatcherEnv` 函数。首先，该函数为 `assembler` 队列 `assembler_queue`、`assembler` 进程互斥锁 `assembler_queue_mutex`、`assembler` 进程资源分配信号量 `assembler_full`、`assembler` 进程工作状态监控信号量 `assembler_work` 分配存储空间并初始化这些变量。

5、调用 `launchDispatcher` 函数，加载数据包重排子模块。

6、调用 `openAssemblerMQ` 函数，为消息队列 `AFDXport_id` 和消息队列信号量 `AFDXport_sem` 分配存储空间并初始化这些变量。

7、调用 `launchAssembler` 函数，加载数据包封装子模块。

8、调用 `server` 函数。首先，该函数新建并绑定 UDP 套接字 `in_port_set`。其次，`server` 函数利用 UDP 套接字 `in_port_set` 接收来自源终端系统的数据帧，并将这些数据保存在 `rxer_message` 中。再次，`server` 函数分别读取数据帧中的序列号 `sequence`、数据帧数据 `data`、数据帧目的端口 `dest_port`、数据帧源端口 `source_port`、数据帧源地址 `ip_address`。最后，`server` 函数查找空闲的数据包重排子模块并将数据交给该线程处理。

平台设计了以下 API 来实现数据包接收处理子模块的功能，子模块在运行时，根据设计的需要加入对以下 API 的调用，API 的设计如表 4-5。

表 4-5 数据包接收处理子模块 API

Table 4-5 API of Data Management and Reception Sub-module

编号	API	描述
1	runRxer	数据包接收处理子模块的主程序
2	LoadDBRxer	读取并加载由数据库模块生成的目的系统终端的数据库文件
3	setEnviromentRxer	为数据包接收处理子模块需要使用的参数分配内存空间并进行初始化
4	setDispatcherEnv	为数据包重排子模块需要使用的参数分配内存空间并进行初始化
5	LaunchDispatcher	启动数据包重拍子模块
6	openAssemblerMQ	初始化并加载数据包对应端口的信息队列
7	LaunchAssemblers	启动数据包封装子模块
8	SigtermHandlerRxer	关闭数据包接收处理、数据包重排、数据包封装子模块并释放其所占用的空间
9	Server	初始化网络套接字，进行数据接收和存储

表 4-5 的 API 为 C 实现的本地 API，仿真平台调用该子模块接收数据后，将数据包传递至数据包重排子模块。

数据包接收处理子模块流程图如图 4-5 所示。

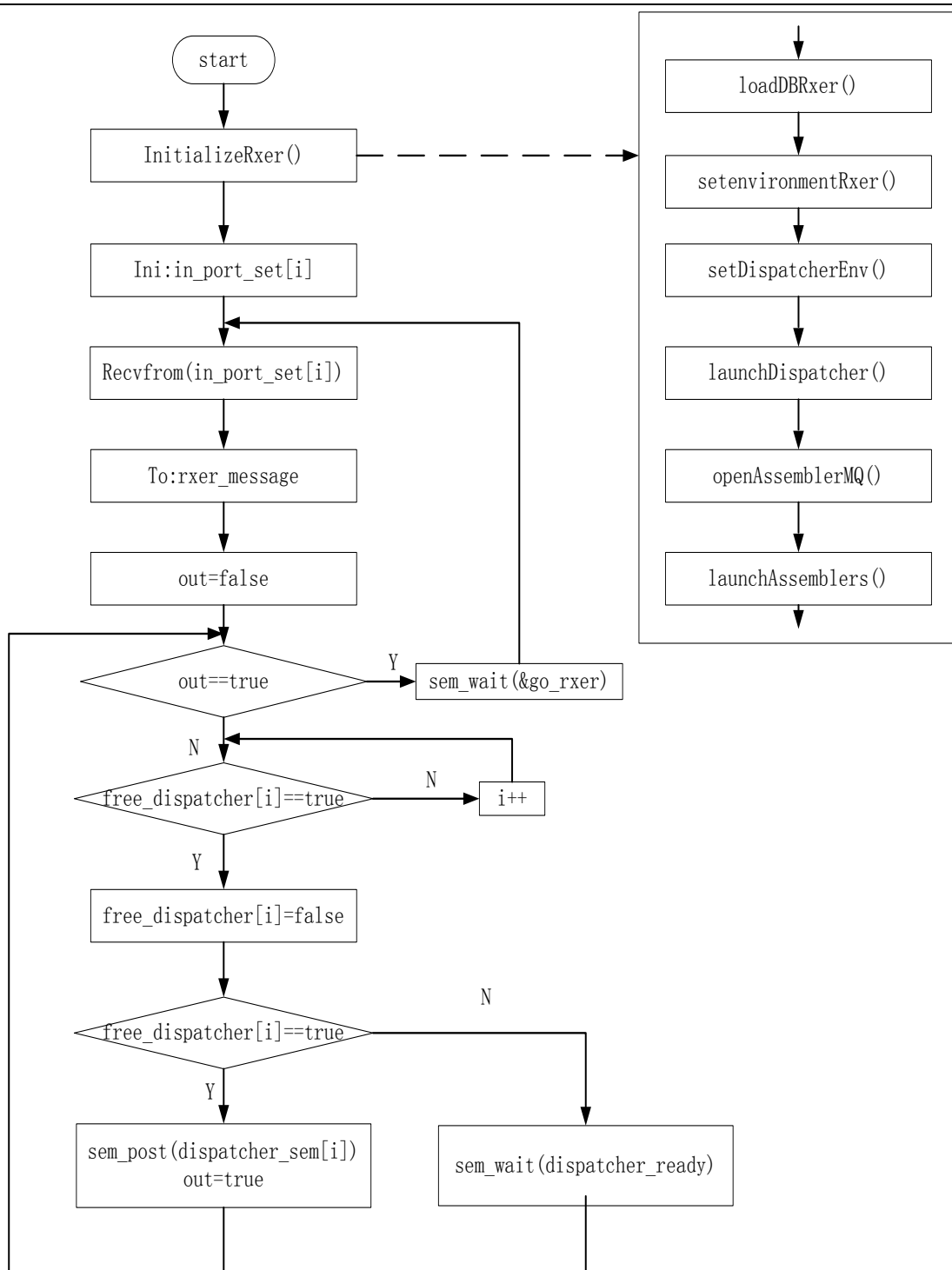


图 4-5 数据包接收处理子模块流程图

Figure 4-5 Flow Chart of Data Management and Reception Sub-module

4.2.6 数据包重排

数据包重排子模块接收来自数据包接收处理子模块发送的数据包，读取数据包内包含的如目的端口、数据、数据包长度、数据包序号等信息，并将数据包内的数据发送给

数据包封装子模块。

首先,数据包重排子模块的主函数 dispatcher 函数进入 while 循环,判断 rxer_message 中的数据是否为最后一个数据,如果是,则变量 out_dis 为 true,本次循环结束后,整个循环将终止。如果 rxer_message 不是最后一个数据,则数据包重排子模块读取 rxer_message 中的目的端口、数据帧数据、数据帧长度、数据帧序号,并将这些数据复制到数据帧 dispatcher_message 中。其次,数据包重排子模块将这些数据帧中发送给数据包封装子模块的队列 assembler_queue 中。

数据包重排子模块的流程图如图 4-6 所示。

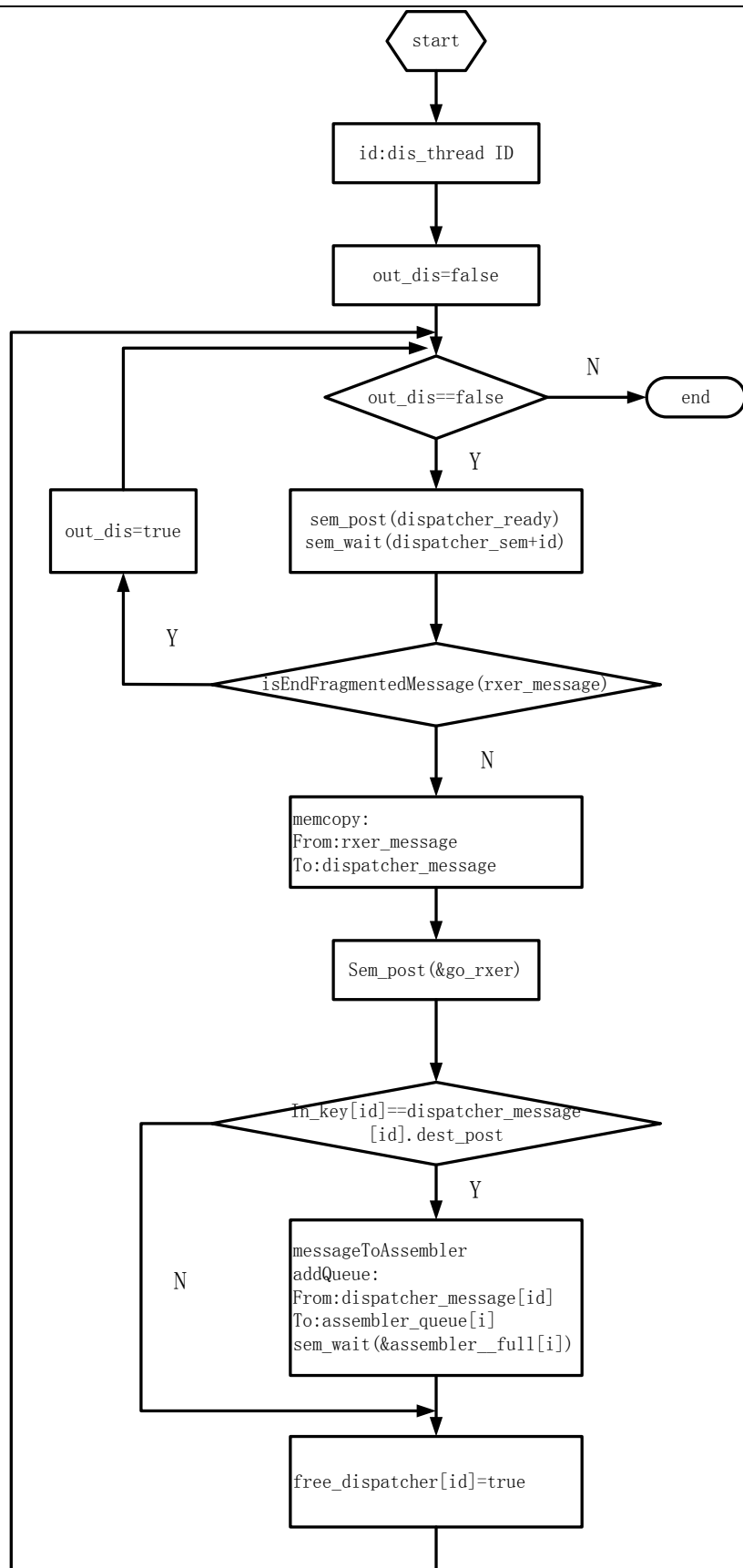


图 4-6 数据包重排子模块流程图

Figure 4-6 Flow Chart of Dispatcher of Sub-module

4.2.7 数据包封装

数据包封装子模块将从数据包重排子模块接收到的数据包存储在 `buffer` 中, 当所有数据包接收完毕后, 数据包被从 `buffer` 中取出, 发送至数据包对应的信息队列。

首先, 数据包封装子模块的主函数 `assembler` 函数进入 `do` 循环, 将需要处理的数据从 `assembler_queue` 中复制到数据帧 `fagm` 中。其次, `assembler` 函数判断 `fagm` 是否为最后一个数据帧, 如果是, 则变量 `out` 为 `true`, `do` 循环将在本次运行完毕后结束。如果不是, 则 `assembler_queue` 中的数据将被复制到数据存储空间 `buffer` 中实现数据包的重组。再次, `assembler` 函数将判断接收到的数据帧 `fagm` 是否为该数据包的最后一个数据帧, 如果不是, 则继续接收该数据包的其他数据帧; 如果是, 则将收到的数据包通过 `mq_send` 发送给信息接收模块。

数据包封装子模块的流程图如图 4-7 所示。

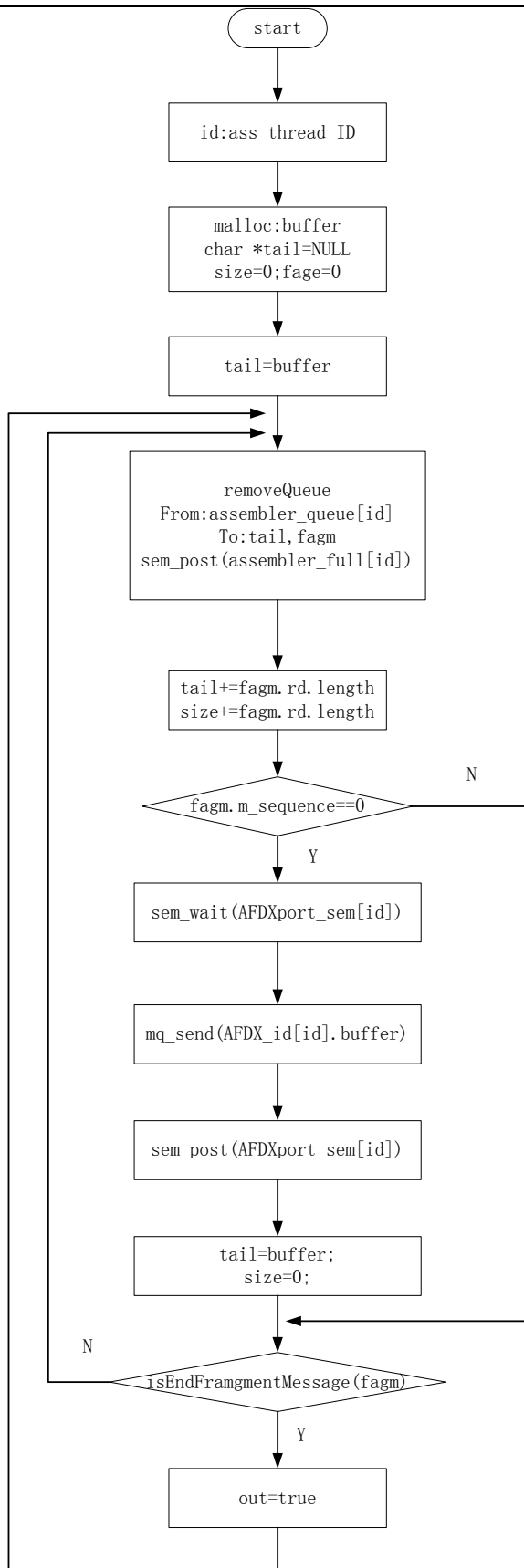


图 4-7 数据包封装子模块流程图

Figure 4-7 Flow Chart of Assembler Sub-module

4.3 信息发送模块

信息发送模块生成数据包，并将数据包发送至仿真平台主程序，由数据包处理发送子模块接收并处理。

数据发送模块流程图如图 4-8 所示。

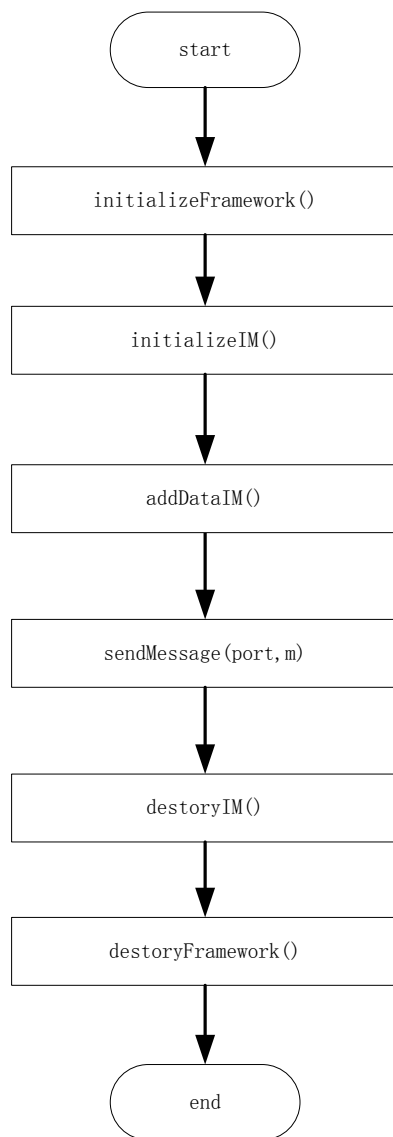


图 4-8 信息发送模块流程图

Figure 4-8 Flow Chart of Data Transmission Module

信息发送模块的功能是接收来自 Avionics Software Application 的数据 Rawdata，并将其反序列化为能够在 AFDX 内部传递的数据格式 Implicit Message，然后将 Implicit Message 传递给数据包处理发送子模块。

首先，信息发送模块利用 argv[] 变量，获取通讯端口号和 ES 号，该通讯端口号和

ES 号用来指定信息发送模块调用的数据库文件。当上述参数获取成功后，信息发送模块调用 initializeFramework 函数为 Af 初始化环境，包括初始化信号量、加载配置、加载数据库文件、打开消息队列等。其次，当环境初始化成功后，信息发送模块新建并绑定 UDP 套接字。并通过该套接字接收来自 Avionics Software Application 的数据 Rawdata 并这些数据保存在 buffer 里。

完成上述操作后，信息发送模块将进入 while 循环。首先，信息发送模块初始化 Implicit Message。其次，信息发送模块利用先前新建的 UDP 套接字接收来自 Avionics Software Application 的数据 Rawdata 并这些数据保存在 buffer 里。第三，利用 innerDeserialize 函数反序列化后的 Rawdata 并将其保存在 Implicit Message 中。第四，通过 sendMessage 函数将 Implicit Message 发送给数据包处理发送子模块。最后，信息发送模块调用 DestroyIM 函数释放 Implicit Message 并再次进入循环。通过使用该循环，信息发送模块实现了从 Avionics Software Application 接收处理数据并将数据发送给数据包处理发送子模块。

当信息发送模块接收到 SIGINT 信号时，信息发送模块的运行将中断。但在中断该模块前，应先中断 AFDX 主程序。

平台设计了以下 API 来实现数据包接收处理子模块的功能，子模块在运行时，根据设计的需要加入对以下 API 的调用，API 的设计如表 4-6。

表 4-6 信息发送模块 API

Table 4-6 API of Data Transmission Module

编号	API	描述
1	InitializeFramework	初始化数据发送模块需要使用的服务
2	initializeIM	初始化发送的数据的存储空间
3	addDataIM	将数据添加到数据的存储空间
4	sendMessage	将数据发送至仿真平台主程序
5	destoryIM	在程序退出时释放数据的存储空间
6	destoryFramework	释放数据发送模块使用的服务

4.4 信息接收模块

信息接收模块接收来自仿真平台主程序发来的数据包。

数据接收模块流程图如图 4-9 所示。

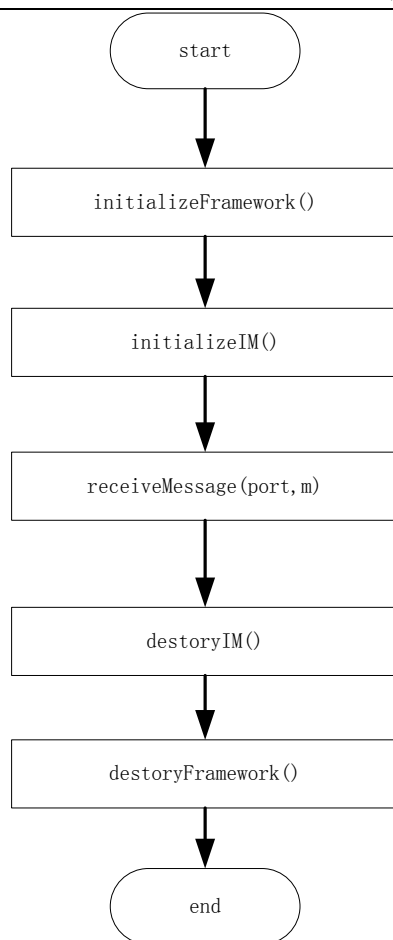


图 4-9 信息接收模块流程图

Figure 4-9 Flow Chart of Data Reception Module

信息接收模块的功能是接收来自 AFDX 程序中数据包封装子模块的数据 Rawdata, 并将这些数据发送给 Avionics Software Application。

首先, 信息接收模块利用 argv[] 变量, 获取通讯端口号和 ES 号, 该通讯端口号和 ES 号用来指定信息接收模块调用的数据库文件。其次, 信息接收模块调用 initializeFramework 函数为 Raf 初始化环境, 包括初始化信号量、加载配置、加载数据库文件、打开消息队列等。完成上述操作后, 信息接收模块新建 UDP 套接字。通过该套接字发送来自 AFDX 程序的数据 Rawdata。

完成上述操作后, 信息接收模块将进入 while 循环。首先, 模块初始化 Implicit Message。其次, 模块调用 receiveMessage 函数接收来自 AFDX 程序发送过来的数据 Rawdata 并将其序列化后存储在 Implicit Message 中获取数据最后, 模块调用 DestrotIM 函数释放 Implicit Message 并再次进入循环。通过使用该循环, 信息接收模块实现了从 AFDX 程序接收处理数据。

当信息接收模块接收到 SIGINT 信号时, 信息接收模块的运行将中断。但在中断该

模块前，应先中断 AFDX 主程序。

表 4-7 信息接收模块 API

Table 4-6 API of Data Reception Module

编号	API	描述
1	InitializeFramework	初始化数据发送模块需要使用的服务
2	initializeIM	初始化发送的数据的存储空间
3	receiveMessage	接收来自仿真平台主程序的数据包
4	destoryIM	在程序退出时释放数据的存储空间
5	destoryFramework	释放数据放松模块使用的服务

4.5 本章小结

本章对系统进行了总体分析，完成了平台的框架设计，并在此框架设计的基础上对航空机载 AFDX 仿真平台的各主要功能模块进行了详细设计和实现，包括数据库模块、数据发送模块、数据接收模块、数据包处理发送、序号管理、轮询调度、虚拟链接、数据包接收处理、数据包重排、数据包封装等子模块。

5 航空机载 AFDX 仿真平台验证

本章完成航空机载 AFDX 仿真平台的验证，包括从数据的生成、传输、序号管理、通过轮询管理虚拟链接、接收信息、重排并封装信息、将生成信息发送至信息接收模块。

5.1 系统部署

为了更好的研究航空机载 AFDX 仿真平台，本文对航空机载 AFDX 仿真平台进行了实际部署。航空机载 AFDX 仿真平台的部署如图 5-1 所示。

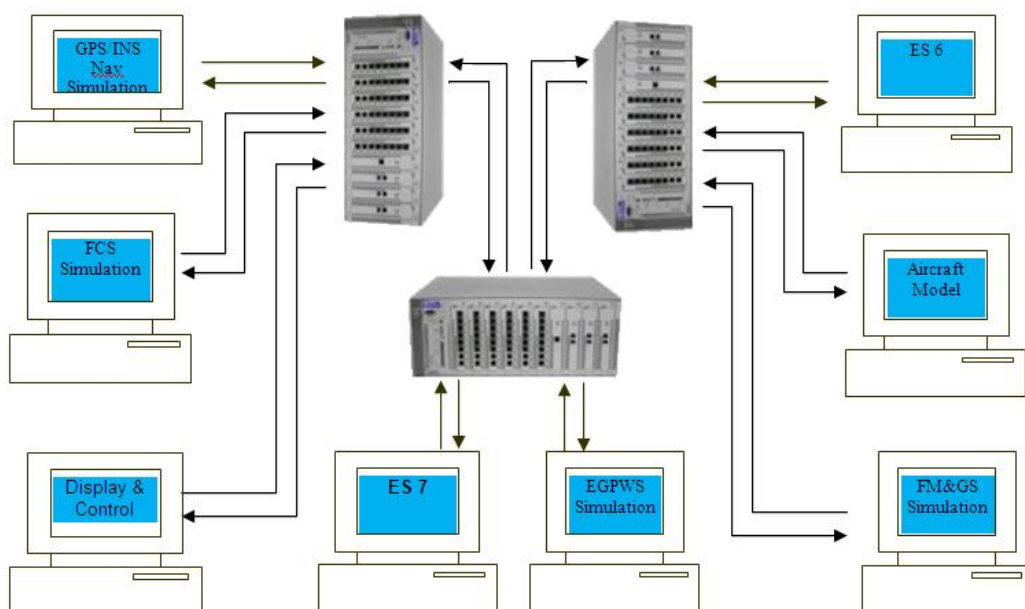


图 5-1 航空机载 AFDX 仿真平台部署

Figure 5-1 Avionics AFDX Simulation Platform Deployment

导航功能模块、显示控制模块、飞行功能模块、增强地面接近预警模块等模块由计算机进行模拟，这些计算机由三台交换机连接至实验室内网。

5.2 系统功能验证

本节将假设一个仿真场景，验证仿真平台是否可以成功运行。

5.2.1 应用场景

导航功能模块生成导航数据，并将其传送至飞行功能模块、显示控制模块和终端系统 6。这四个终端由四台计算机进行仿真，仿真信息如下：

表 5-1 系统对比验证

Table 5-1 comparative analysis example data

终端名称	IP 地址	通讯端口
导航功能模块	192.168.1.1	49011
飞行功能模块	192.168.1.5	49012
显示控制模块	192.168.1.8	49013
终端系统 6	192.168.1.14	49014

5.2.2 验证方法

首先，启动 192.168.1.5 的数据库模块、仿真程序主模块、数据接收模块，如图 5-2 所示。

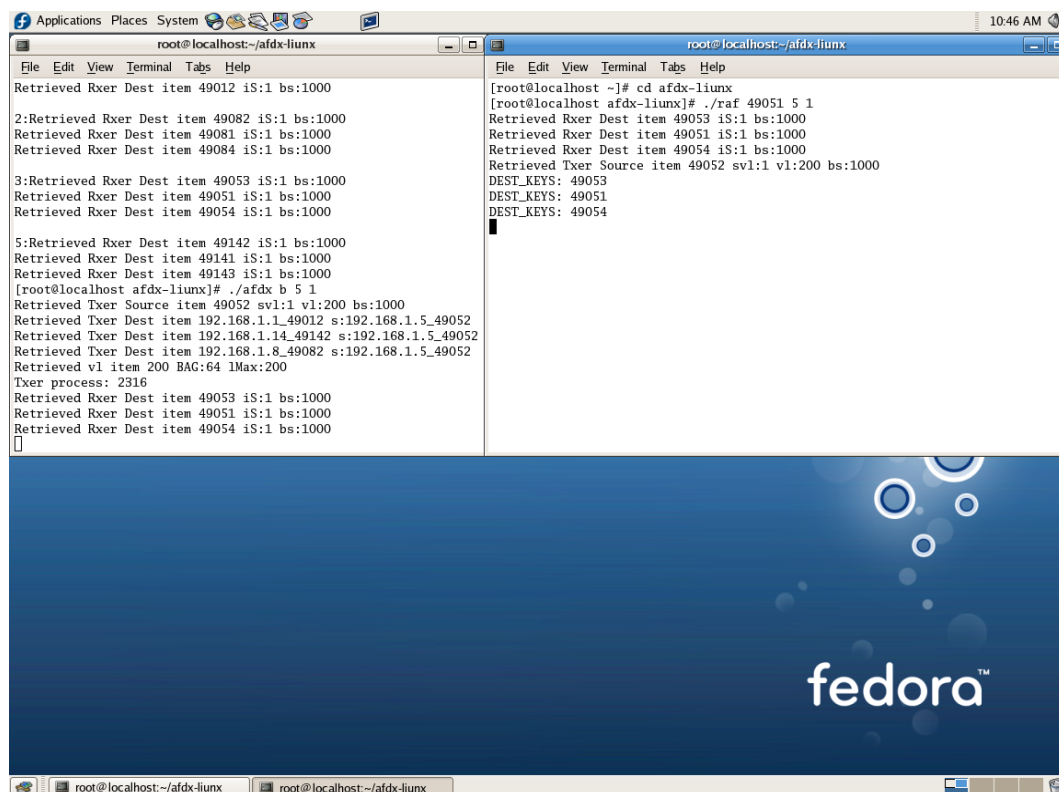


图 5-2 192.168.1.5 启动

Figure 5-2 Start-up of 192.168.1.5

启动 192.168.1.8 的数据库模块、仿真程序主模块、数据接收模块，如图 5-3 所示。

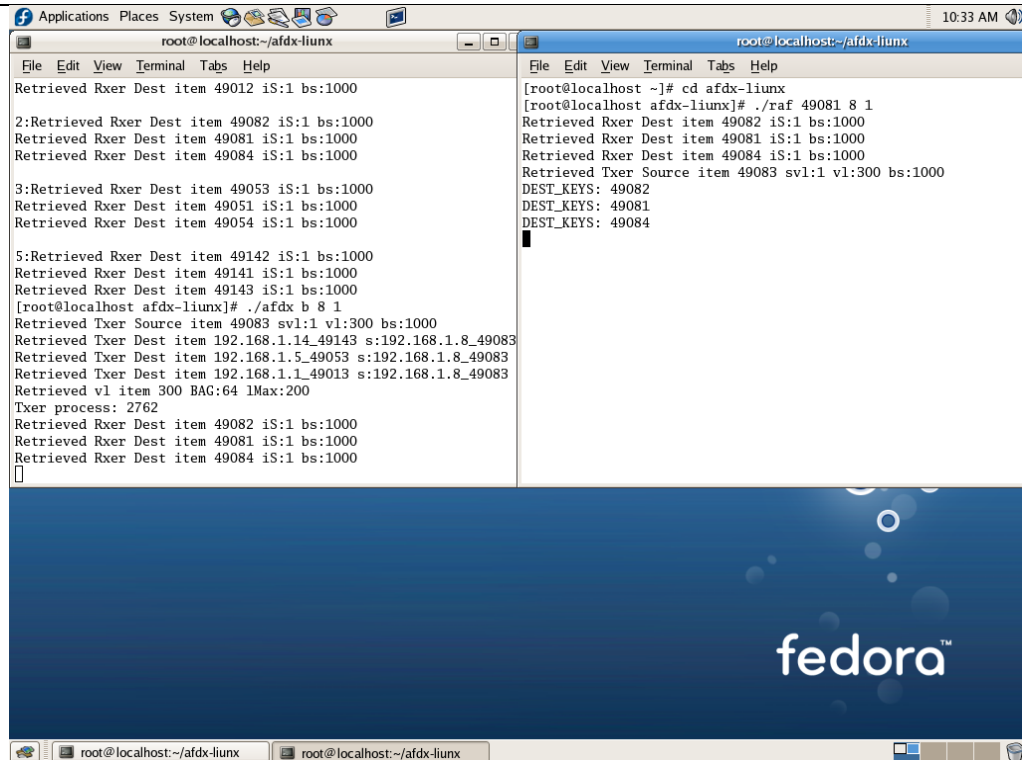


图 5-3 192.168.1.8 启动

Figure 5-3 Start-up of 192.168.1.8

启动 192.168.1.14 的数据库模块、仿真程序主模块、数据接收模块，如图 5-4 所示。

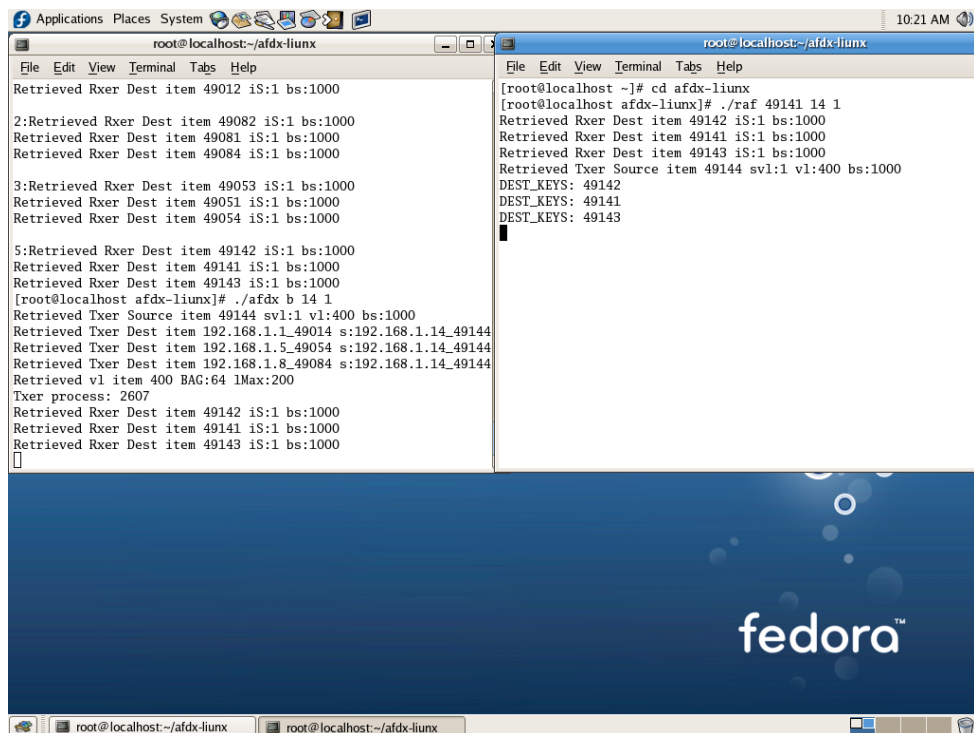
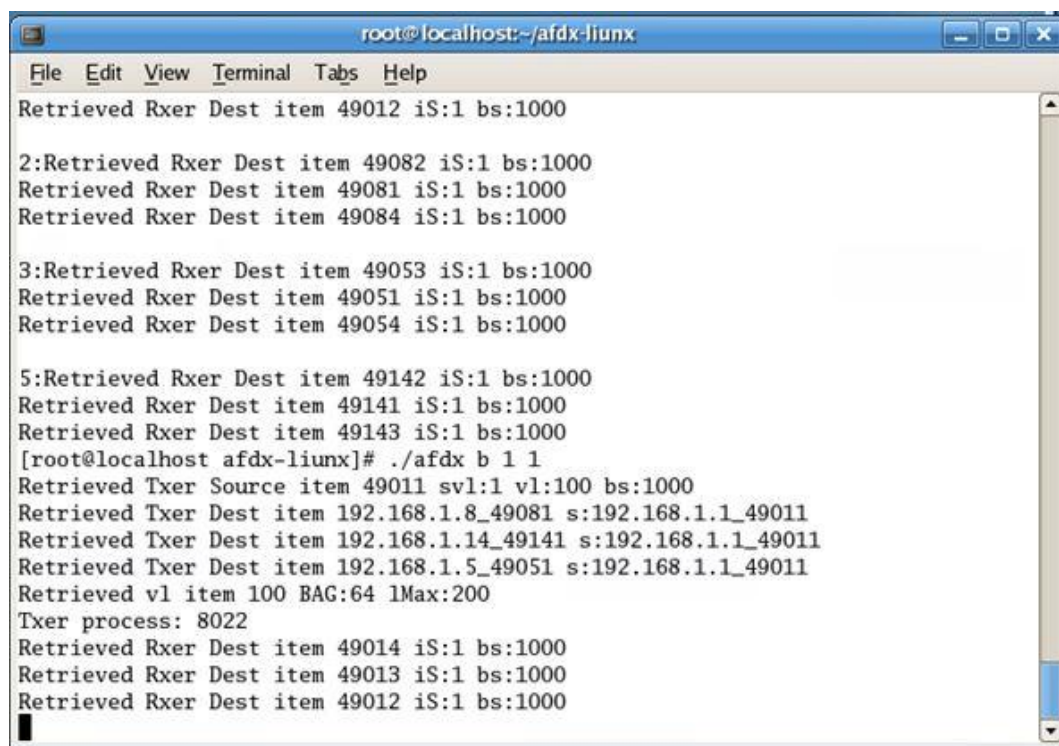


图 5-4 192.168.1.14 启动

Figure 5-4 Start-up of 192.168.1.14

启动 192.168.1.1 的数据库模块、仿真程序主模块，如图 5-5 所示



```
root@localhost:~/afdx-liunx
File Edit View Terminal Tabs Help
Retrieved Rxer Dest item 49012 iS:1 bs:1000

2:Retrieved Rxer Dest item 49082 iS:1 bs:1000
Retrieved Rxer Dest item 49081 iS:1 bs:1000
Retrieved Rxer Dest item 49084 iS:1 bs:1000

3:Retrieved Rxer Dest item 49053 iS:1 bs:1000
Retrieved Rxer Dest item 49051 iS:1 bs:1000
Retrieved Rxer Dest item 49054 iS:1 bs:1000

5:Retrieved Rxer Dest item 49142 iS:1 bs:1000
Retrieved Rxer Dest item 49141 iS:1 bs:1000
Retrieved Rxer Dest item 49143 iS:1 bs:1000
[root@localhost afdx-liunx]# ./afdx b 1 1
Retrieved Txer Source item 49011 svl:1 vl:100 bs:1000
Retrieved Txer Dest item 192.168.1.8_49081 s:192.168.1.1_49011
Retrieved Txer Dest item 192.168.1.14_49141 s:192.168.1.1_49011
Retrieved Txer Dest item 192.168.1.5_49051 s:192.168.1.1_49011
Retrieved vl item 100 BAG:64 lMax:200
Txer process: 8022
Retrieved Rxer Dest item 49014 iS:1 bs:1000
Retrieved Rxer Dest item 49013 iS:1 bs:1000
Retrieved Rxer Dest item 49012 iS:1 bs:1000
```

图 5-5 192.168.1.1 启动

Figure 5-5 Start-up of 192.168.1.1

至此，四台仿真终端已经准备就绪。

5.2.3 系统功能验证

四台终端准备就绪后，启动 192.168.1.1 的数据发送模块，如图 5-6 所示。

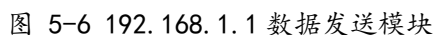


Figure 5-6 192.168.1.1 Data Transmission Module

信息发送模块的运转情况如图 5-7 所示。

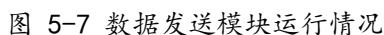


Figure 5-7 Operational Aspect of Data Transmission Module

从图 5-6 和图 5-7 可以看到, 192.168.1.1 的仿真平台数据发送模块运行正常, 持续发出数据包。

192.168.1.5 的信息接收情况如图 5-8 所示。

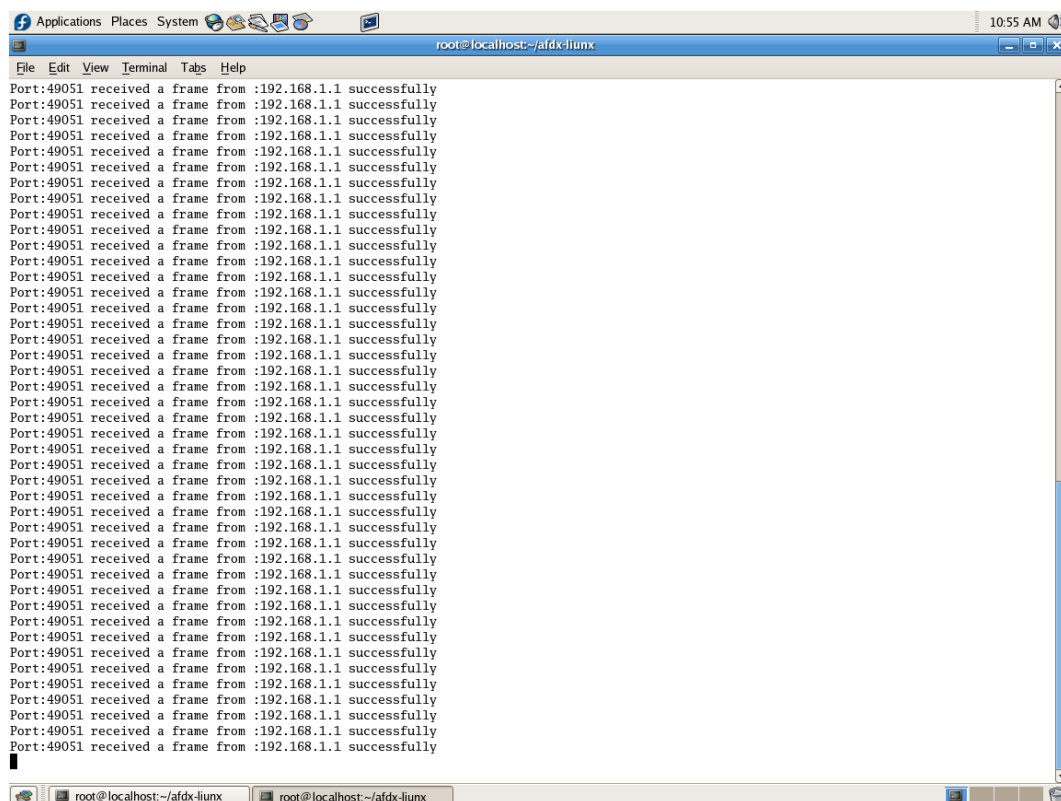


图 5-8 192.168.1.5 的数据接收情况

Figure 5-8 Data Reception Status of 192.168.1.5

192.168.1.8 的信息接收情况如图 5-9 所示。

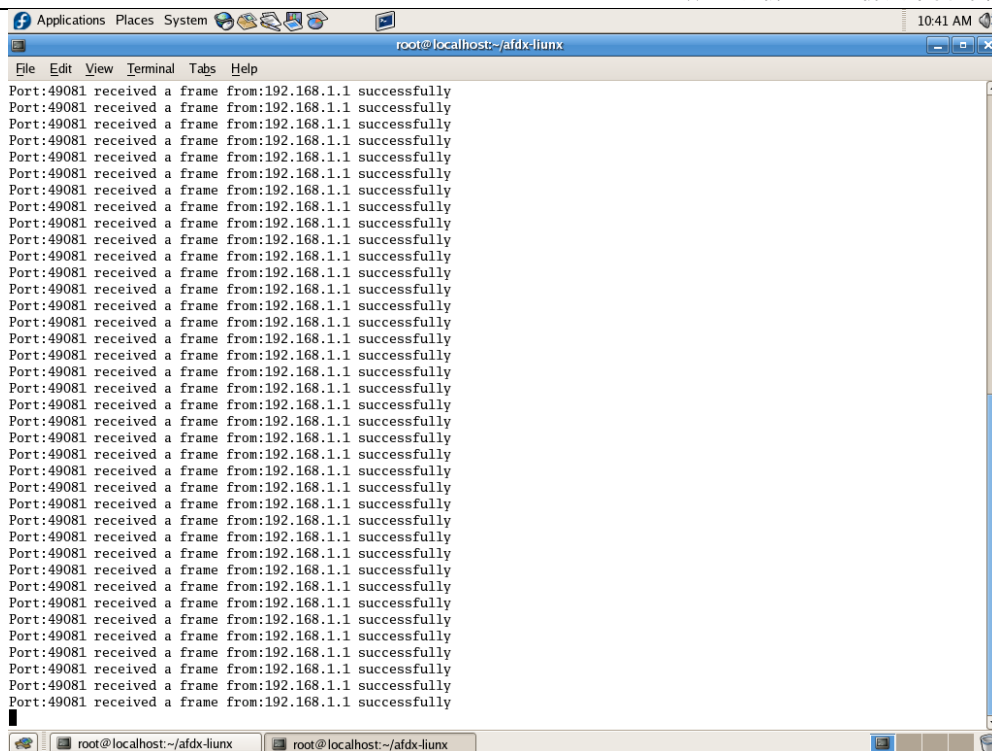


图 5-9 192.168.1.8 的数据接收情况

Figure 5-9 Data Reception Status of 192.168.1.8

192.168.1.14 的信息接收情况如图 5-10 所示。

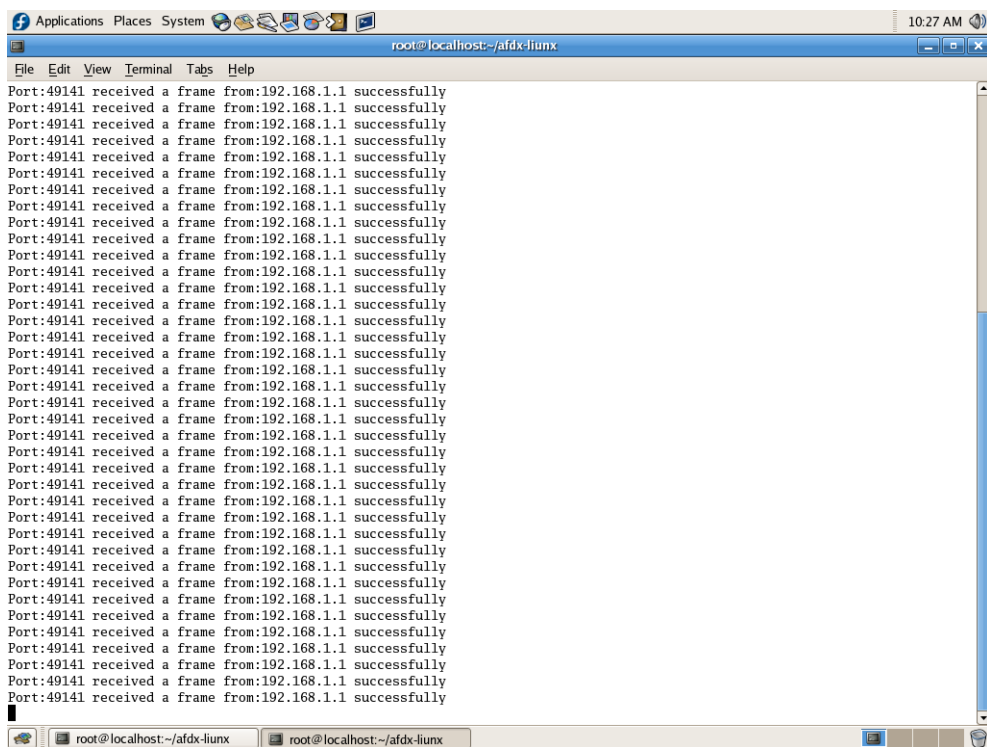


图 5-10 192.168.1.14 的数据接收情况

Figure 5-10 Data Reception Status of 192.168.1.14

从图 5-8、5-9、5-10 可以看到, 192.168.1.5、192.168.1.8、192.168.1.14 都能够正常获取来自 192.168.1.1 的信息。

5.2.4 系统效果分析

从前面的系统功能演示我们可以看到: 首先, 作为接收 AFDX 数据包的目的终端系统, 192.168.1.5、192.168.1.8 和 192.168.1.14 启动了各自的数据库模块, 顺利完成了数据库文件的生成工作。其次, 这三台设备上的仿真程序主模块也顺利启动, 完成了相关子模块和环境变量的初始化工作。至此, 这三台目的终端系统已经做好了接收 AFDX 数据包前的准备工作, 数据库模块和仿真程序主模块的功能性也得到了验证。

与此同时, 作为发送 AFDX 数据包的源终端系统, 192.168.1.1 也顺利启动了数据库模块生成数据库文件。其次, 192.168.1.1 启动了机载 AFDX 仿真平台的主模块以完成相关子模块及环境变量的初始化工作。以上两步顺利完成后, 该终端系统完成了 AFDX 数据包发送前的准备工作, 同时验证了数据库模块和仿真程序主模块的功能性。

至此, 通过启动目的终端系统的数据接收模块, 三台目的终端系统开始监听相应的 UDP 端口。而启动源终端系统的数据发送模块, 192.168.1.1 开始了 AFDX 数据包的发送, 而与之相对应的, 三台目的终端系统也接收到了来自源终端系统的 AFDX 数据包。由此, 验证了目的终端系统的数据接收功能以及源终端系统的数据发送功能。

通过软件硬件结合的方法, 验证了该航空机载 AFDX 仿真平台能够通过软件仿真结合硬件设备, 实现从源终端系统到目的终端系统的 AFDX 数据通讯的全过程仿真。

5.3 本章小结

本章节以系统功能的角度, 来对于 AFDX 仿真平台的实际可用性开展测试。最终测试的结果显示, 本设计成功的仿真了 AFDX 网络通讯的全过程。对今后同类软件工具的研发起到一定的借鉴作用。

6 总结与展望

本章将对本文的所有工作进行总结,并展望航空机载 AFDX 仿真平台下一步研究内容。

6.1 总结

本论文分析航空数据总线的历史成长以及实际状况,探讨当前航空数据总线的特点,发现了航空数据总线模拟的需求和趋势。航空业需要能够满足目前乃至未来一段时间内的数据航空总线规范。为此,本文把研究重点放在目前已被采用的最有前景的数据总线规范 ARINC 664 P7,摒弃了目前在市场上仍在广泛使用的 ARINC 429。

本文对 ARINC 664 P7 进行了深入研究,掌握了 AFDX 网络的通讯行为。通过对 AFDX 网络的研究,为仿真平台的需求分析打下了理论基础,同时为仿真平台的功能模块划分提供了重要的指导。

本文对机载航空 AFDX 仿真平台开展需求分析,设计整体仿真平台的系统架构。论文提出了 AFDX 仿真平台的整体需求,认为架构中需要涵盖数据库管理、仿真平台主程序和数据发送、数据接收等基本模块。本文对于整体需求的多个功能模块开展更为深入的需求分析,细化仿真平台主程序的需求,将仿真平台细分为包括序号管理、轮询调度、数据包重排、数据包封装等子模块。

本论文实现了机载航空 AFDX 仿真平台。实现了数据库文件的创建、数据发送、数据处理、数据接收等模块。

本文还对系统的功能进行验证。验证了机载航空 AFDX 仿真平台在 AFDX 仿真中的可用性。机载航空 AFDX 仿真平台的研究与探讨,有利于尽快形成新一代航空数据总线仿真环境的搭建。航电系统工程师也可以利用 AFDX 仿真平台模拟 AFDX 的网络通讯行为,并根据研究的实际需要搭建基于 ARINC 664 P7 的航电系统。对他们理解 AFDX 的网络通讯行为、网路性能、网络特性以及航电设备的测试有重要意义。同时,本次仿真平台搭建过程中所使用的软硬件都是目前市场上的商用产品,大部分软件都是免费的,极大降低了仿真过程中的费用,使在普通实验室内搭建基于 AFDX 的仿真平台进行研究成为可能。

6.2 展望

本文讨论了机载航空 AFDX 仿真平台的需求、建立了机载航空 AFDX 仿真平台、

验证了仿真平台的可用性，证明通过软硬件结合进行仿真的方法是发展机载航空 AFDX 仿真平台的途径之一。本文的主要工作在于在计算机及网络软硬件平台搭建一台 AFDX 仿真平台，实现 AFDX 仿真模拟。但值得注意的是，这款仿真平台还存在着不足，如没有实现冗余线路下的仿真、没有引入网络数据校验、用户界面不够友好。但作者相信，随着 AFDX 仿真的进一步深入研究，以上问题都会得到解决。

参考文献

- [1] 钟杰, 何民, 王怀胜, 郑力, AFDX 构架及协议分析[J], 电讯技术, 2010, 01, 65-71.
- [2] 校莉, AFDX 在航空通信系统中的应用[J], 电讯技术, 2010, 07, 40-43.
- [3] 彭寒, 李小飞, AFDX 航空总线仿真测试技术研究[J], 西安航空技术高等专科学校学报, 2010, 05, 10-13.
- [4] 贾卫松, 翟正军, 牛仕奇, AFDX 端系统设计中的发送调度方法研究与实现[J], 计算机测量与控制, 2010, 11, 2612-2615.
- [5] 杨金孝, 王鑫, 刘慧卓, AFDX 以太网冗余管理的算法设计[J], 电子设计工程, 2013, 11, 21-23.
- [6] 邱征, 王红春, 陈长胜, 余亚刚, AFDX 网络演算时延分析方法研究[J], 科技视界, 2013, 16, 36+82.
- [7] 叶佳宇, 陈晓刚, 张新家, 基于 AFDX 的航空电子通信网络的设计[J], 测控技术, 2008, 06, 56-58+60.
- [8] 徐科华, AFDX 总线网络数据传输分析[J], 民用飞机设计与研究, 2009, 03, 35-40.
- [9] 祝永卫, 刘俊千, AFDX 总线实验平台的设计与实现[J], 中国现代教育装备, 2009, 17, 54-56.
- [10] 牛仕奇, 严胜刚, 任向隆, AFDX 终端系统实现方案研究[J], 计算机测量与控制, 2009, 12, 2507-2509+2520.
- [11] 宋东, 曾星星, 丁丽娜, 胡琼, AFDX 网络系统建模与仿真实现[J], 测控技术, 2012, 02, 76-80.
- [12] 施太平, 娄莉, 田泽, AFDX 协议及关键技术的实现[J], 测控技术, 2012, 10:81-84.
- [13] 何向栋, 陈长胜, 张志平, 徐文杰, 基于 FIFO 的 AFDX 交换机帧捕获单元设计与实现[J], 计算机测量与控制, 2013, 12, 3362-3364.
- [14] 夏大鹏, 田泽, 基于 AFDX 终端系统测试的研究[J], 计算机技术与发展, 2011, 08, 192-195+199.
- [15] Dong, Song, et al, The design and implementation of the AFDX network simulation system. [C], Multimedia Technology (ICMT), 2010 International Conference on. IEEE, 2010, 1-4
- [16] Chen, Xin, Xudong Xiang, and Jianxiong Wan, A software implementation of afdx end system. [C], New Trends in Information and Service Science, 2009. NISS' 09, International Conference on. IEEE, 2009, 558-563.
- [17] ARINC, 664 Aircraft data network part7: avionics full duplex switched Ethernet (AFDX) [S], Aeronautical Radio Inc., Annapolis, MD, 2005.

- [18] ARINC, 653 Avionics Application Software Standard Interface[S], Part 1 - Required Service. Aeronautical Radio Inc., Annapolis, MD, 2005.
- [19] 李俊鹏, 王勇, 白焱, 刘安, AFDX 虚连接自适应调度策略设计与实现[J], 计算机测量与控制, 2012, 07:1986-1988.
- [20] 李大川, 程农, 李清, 宋靖雁, 基于 AFDX 网络的飞行管理仿真系统[J], 北京航空航天大学学报, 2011, 07:861-867+876.
- [21] 武华, 马捷中, 翟正军, AFDX 端系统通信端口的设计与实现[J], 测控技术, 2009, 03:56-59.
- [22] 黄梦玲, 翟正军, 基于 ARINC429 与 AFDX 的测试仿真系统设计与实现[J], 计算机测量与控制, 2013, 08:2090-2092+2108.
- [23] Moir, I., Seabridge, A., & Jukes, M. Civil avionics systems[M], John Wiley & Sons. 2013.
- [24] ARINC, 651-1 Design Guidance for integrated modular avionics[S], Aeronautical Radio Inc., Annapolis, MD, 1997.
- [25] 陈昕, 周拥军, 万剑雄, AFDX 端系统关键技术的研究与实现[J], 计算机工程, 2009, 05:1-3.

致 谢

在此论文完成之时，向所有曾经在我就读上海交通大学软件学院工程硕士学位期间提供过协助以及扶持的人们表达诚挚的谢意。

首先，需要感谢导师姚建国副教授，正是在姚老师的不懈指导下，我的论文撰写才能够顺利进行。在论文初期的开题、撰写，中期的检查、修改，直到最终的论文定稿，姚老师都提供了大量的帮助。姚老师在治学方面严谨的态度，对理论知识深厚的造诣都使我感触颇深，受益匪浅。遇高人岂可交臂而失之，我有理由相信，这段学习经历将对我今后的工作产生深远的影响。

其次，我要感谢中国商飞信息化中心的同事，尤其是网络与统一通讯组的同事，是他们分担了大部分本应由我承担的工作任务，他们任劳任怨的帮助我，为我的在交大的学习创造了宽松的条件，使我能够从容的完成学习工作。我还要感谢信息化中心的领导们，其为众多专研学业的同学们提供了宝贵的机遇。

再次，衷心的感谢一同学习的学友们，尤其是丁炜、张潇鹏和张林。这一路上有他们的并肩陪伴，给了我前行的信心。还要感谢家人提供的扶持，家人的支持正是我坚持学业的最大动力。若没有她们提供的辛勤付出，我的学业将难以顺利的进行。

最后，对于在繁忙的事务之中抽出时间审阅本文的老师们送上诚挚的敬佩之情。

攻读学位期间发表的学术论文目录

- [1] 第一作者, 航空机载 AFDX 仿真平台的设计与实现, 上海交通大学软件学院网站公示, 2015 年 12 月