学校代码	10699
分 类 号	TP393
密　　级	
学　　号	2009100462

西北工業大學

NORTHWESTERN POLYTECHNICAL UNIVERSITY

博士学位论文

题目　分布式 IMA 的网络性能及机载数据完整性分析与研究

作者　焦文喆

学科、专业　计算机科学与技术

指导教师　王国庆

申请学位日期　2017 年 3 月

学校代码	10699
分 类 号	TP393
密　　级	
学　　号	2009100462

题目 分布式 IMA 的网络性能及机载数据完整性分析与研究

作者 焦文喆

学科、 专业 计算机科学与技术

指 导 教 师 王国庆

申请学位日期 2017 年 3 月

西 北 工 业 大 学

博 士 学 位 论 文

（学位研究生）

题目：分布式 IMA 的网络性能及机载数据完整性分析与研究

作　　者：焦文喆

学科专业：计算机科学与技术

指导教师：王国庆

2017 年 3 月

Title: Analysis and Research on the Network Performance and Data Integrity Of Distributed IMA

By
Jiao Wen zhe

Under the Supervision of Professor
Wang Guo qing

A Dissertation Submitted to
Northwestern Polytechnical University

In partial fulfillment of the requirement
For the degree of
Doctor of Philosophy

Xi'an P. R. China
March 2017

摘 要

伴随着信息技术、微电子技术、多媒体业务的迅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和依赖性日益提高。为了满足不断扩展的飞机任务需求，分布式综合模块化航空电子系统（Distributed Integrated Modular Avionics, DIMA）中各航空电子系统子系统间的数据传输任务随之激增。然而，由于网络带宽资源有限，多任务传输产生的竞争链路共享冲突会影响数据传输的实时性和时间确定性，进而降低整个航空电子系统的稳定性和反应能力。同时，随着交互式娱乐系统在新型飞机中的广泛应用，存储在航电网络中的数据不再具有物理上的隔离安全性，而可能被非法访问、篡改和删除，从而对航空电子系统的安全性造成极大的威胁。作为新一代航空电子系统结构，DIMA 对网络数据传输性能和数据在存储器中的完整性提出了更高要求。

本文以 DIMA 系统中的网络数据为研究对象，针对传输过程中的实时性和时间确定性问题，以及存储器中数据的完整性证明问题，进行了若干关键技术的研究，进一步提高 DIMA 系统的适用性。论文的主要研究内容与创新性工作如下：

1. 提出了一种时间触发 AFDX （Time-Triggered Avionics Full Duplex Switched Ethernet, TTAFDX）网络的体系结构。通过对 AFDX 的网络结构及网络协议进行全面地分析和改进，将时间触发机制引入 AFDX，并参考 SAE AS6802 协议提出了适用于时间触发 AFDX 的时间同步技术。同时，将虚链路划分为时间触发虚链路和速率限制虚链路，按照实时性和时间确定性需求为网络传输任务选择适当的链路类型，能够有效解决 DIMA 系统中时间关键消息传输的时间不确定性问题。时间触发 AFDX 与 AFDX 网络硬件设计完全一致，能够减少开发周期与成本。

2. 设计了基于周期优先原则和基于帧长度优先原则的端系统及交换机系统时间触发虚链路调度算法。当同周期时间触发虚链路的数据帧长度存在较大差异时，周期优先的调度表设计方法会降低带宽的利用率。帧长度优先调度表设计方法能够在保证时间触发流量无发送冲突的同时，有效解决带宽利用率的问题。搭建典型网络，对 TTAFDX 及采用 FIFO 和静态优先级调度算法的 AFDX 网络分别使用网络演算方法计算网络时延并进行对比。计算结果表明：TTAFDX 中，固定时延是时间触发虚链路时延的主要构成，同时，速率限制虚链路的时延较采用静态优先级调度算法的 AFDX 中的低优先级虚链路也有所提高。

3. 为了进一步提高时间触发 AFDX 网络的性能，针对时间触发 AFDX 网络中速率限制虚链路提出了具有低复杂度、高公平性和低调度延时特性的逐次最小定额轮询（Successive Minimal-Quantum Round Robin, SMQRR）调度机制。SMQRR 调度算法通过改变对每一个数据流权值所对应的数据量的连续服务方式，在保留轮询调度算法 $O(1)$ 时间复杂度的同时，有效克服了分组长度不同带来的不公平性并避免了数据流可能长时

间无法获得服务的情况。通过理论分析和仿真验证，SMQRR 调度算法在公平性和时延上界均优于加权可变轮询调度算法（Weighted Elastic Round Robin, WERR）和差额加权轮询调度算法（Deficit Weighted Round Robin, DWRR）。

4. 提出了基于第三方的数据完整性检查模型以及适用于静态数据完整性验证的方案。对于采用 DIMA 系统的新型民用飞机，开放性与交互性使航空电子数据安全面临前所未有的威胁。基于代数签名技术提出的静态数据完整性检查方案，能够极大降低通讯开销，在数据遭到破坏的情况下，以高概率检查出数据错误。通过对数据采用前向纠错编码，提高了数据的完整性保障。安全性分析和实验结果表明所提方法是安全可证明的并且系统的性能瓶颈由磁盘速率决定。

5. 提出了基于跳表和基于梅克尔哈希树（Merkle Hash Tree, MHT）存储结构的数据完整性验证方案。在航空电子数据存储过程中采用跳表结构或 MHT 结构，能够支持数据的动态操作。应用双线性对映射签名技术，在为数据完整性检查提供了信息隐私保护的同时，系统开销较代数签名技术有所增加，但不影响系统传输和计算性能。所提出的数据完整性检查方案无需与原始数据相比较，且验证不受次数限制。通过理论分析和实验表明所提方法是安全的。

论文研究的周期优先和帧长度优先调度算法、逐次最小定额轮询（SMQRR）调度机制和存储数据的完整性验证方法等部分成果及其思路，已应用于国防基础预研、民机预研以及部分型号任务子课题中，提高了时间关键消息传输的实时性和时间确定性，保证了数据存储的可靠性与安全性，验证了论文研究成果的实用性。

关键词：分布式综合模块化航空电子系统，时间触发，AFDX，调度算法，网络演算，数据完整性验证

Abstract

With the rapid development of micro-electronic technology, information technology, multimedia service and more complicated airspace environment, aircraft is highly dependent on the avionics system. In order to satisfy the growing demand of aircraft, the data transfer between subsystems of Distributed Integrated Modular Avionics (DIMA) has soared. Since the network bandwidth is limited, sharing conflicts between competitive links, which are generated by multi-tasking, will affect temporal determinacy of data transmission and decrease the stability and reaction capability of avionics. Meanwhile, with the extensive implementation of interactive entertainment system, data stored in the avionics network is no longer physically isolated. The illegal tampering and deletion will bring a great threat to avionics system. As a new generation structure of avionics system, DIMA puts forward a higher request for the real-time, low-latency of network data transmission and security of data storage.

This thesis focuses on time uncertainty problem in data transmission and data integrity verification issues in data storing. This thesis presents several key techniques and research achievements, which will further improve the applicability of DIMA. The main research contents and innovative work of this thesis are as follows:

1. This thesis proposes a Time-triggered AFDX (TTAFDX) network architecture, which combines time-triggered mechanism with traditional AFDX by comprehensive analysis and improvement. Refers to the SAE AS6802, this thesis proposes a compatible clock synchronization technique for TTAFDX. The virtual link is classified into two categories: time-triggered and rate-constrained virtual link. To solve the temporal uncertainty problem of time-critical message transmission in DIMA, system will select appropriate virtual link type for transmission task according to the real-time and temporal determinacy requirement. The hardware design of TTAFDX is the same as AFDX, which will shorten the development cycle and reduces the cost.

2. This thesis proposes two scheduling strategies for both TTAFDX end system and switch system respectively. One of them is frame-length prior based scheduling and the other one is cycle-prior based scheduling. Cycle-prior based scheduling drops the bandwidth utilization when time-triggered virtual links' frame lengths differ a lot. Frame-length prior based scheduling is proposed to solve this problem. Using network calculus method, TTAFDX is compared with FIFO scheduling based AFDX and static priority scheduling based AFDX respectively in the aspect of real-time performance. The result shows that, the fixed delay is the main part of the time-triggered virtual link latency, and the deterministic of rate constraint virtual link is also improved.

3. To further improve the performance of TTAFDX, this thesis proposes a Successive Minimal-Quantum Round Robin (SMQRR) scheduling algorithm for rate-constrained virtual link, which is fairness, efficient, and has a low latency bound. For each data flow, the data served in each round is corresponding to its weight. By refining the data amount to be served and round robin mode, SMQRR effectively overcomes the unfairness caused by different frame lengths and avoids the situation that the data flow may not be served for a long time. Theoretical derivation and simulation analysis shows that SMQRR has better latency and fairness than Weighted Elastic Round Robin (WERR) and Deficit Weighted Round Robin (DWRR).

4. This thesis proposes a data integrity verification model and an algebraic signature based data integrity verification scheme. For new model civil aircraft using DIMA system, openness and interactivity brings unprecedented security threats to avionics data. The algebraic signature based static data integrity verification scheme can greatly reduce communication overhead, and check out the data error with high probability in the case of data destruction. Forward error correction coding is used to guarantee the integrity of avionics data. The safety analysis and experimental results reveal that the computation performance is effective, and bounded by disk I/O.

5. Based on skip-list and Merkle Hash Tree (MHT) data structure, this thesis proposes two dynamic data integrity verification schemes respectively. Skip-list and MHT structures can support fully dynamic data operations. This scheme uses Bilinear Map to support privacy-preserving data integrity verification operations. The scheme allows unlimited times verification without the need for the verifier to compare against the original data, which reduces the communication and computation complexity dramatically and preserves the privacy of the data. Extensive security analysis and simulation show that the proposed scheme is highly provably secure.

Some of research achivements and thoughts elaborated in this thesis have been implemented in some sub-topics of national basic research, civil aircraft research. The results show that temporal determinacy is improved for time-critical data transmission and data reliability and security is guaranteed for distributed data storage. The practicabilities of the research results are verified.

Keywords: Distributed integrated modular avionics; Time-triggered; Avionics full duplex switched ethernet; Scheduling algorithm; Network calculus; Data integrity verification

目 录

1 绪 论

1.1 选题背景和意义

DIMA（Distributed Integrated Modular Avionics, DIMA）[1]作为新一代航空电子系统，满足了航空电子系统“信息一体化”的发展需求，进一步推动了航空电子系统向综合化、模块化、网络化方向的发展。DIMA 系统为了满足日益增长的功能综合与数据融合需求，必须保证传感器、处理节点与功能节点间的数据通信满足实时性和时间确定性要求。根据 SAE 组织预测[2]，未来飞行器的数据传输延迟在千兆位的速率网络带宽下不应大于 $100\,\mu s$。航空网络互连技术作为航空电子系统发展的关键技术，随着航空电子全双工交换式以太网（Avionics Full Duplex Switched Ethernet, AFDX）[3]在新型客机上的应用，已由低速子系统间总线互连发展为高速交换式网络[4]。然而，由于网络带宽资源有限，多任务传输产生的竞争链路共享冲突仍会影响数据传输的实时性和时间确定性，进而降低整个航空电子系统的稳定性和反应能力。同时，随着交互式娱乐系统在新型飞机中的广泛应用，航空数据在数量、多样性和重要性等方面发生了巨大的变化，且存储在航空电子网络中的数据不再具有物理上的隔离安全性，而可能被非法访问、篡改和删除，从而对航空电子系统的安全性造成极大的威胁。

本文以 DIMA 系统为研究对象，分别对系统中网络数据的传输性能和数据存储的完整性进行专题研究，为 DIMA 系统的应用提供了有力的支持。本文的研究成果不仅具有重要的理论意义，而且在国防领域及国民经济领域具有广阔的应用前景。

1.2 航空电子系统结构发展现状

随着微电子技术和信息技术的迅速发展，综合考虑飞机性能需求以及经济可承受性，航空电子系统结构经历了多个阶段的发展和演变，已成为决定飞机性能和效能的核心要素。

1）分立式航空电子系统结构

20 世纪 40 年代至 60 年代前期，第一代分立式航空电子系统结构仅针对航空任务有效性进行研究[5]，各航空电子设备拥有独立的控制器、传感器和显示器[6]，开发完全独立，各设备间采用点对点连接且交联较少，信息交换困难，灵活性差。

2）联合式航空电子系统结构

20 世纪 60 年代中期，数字计算机作为控制中心大量应用于集中式航空电子系统中的导航与火控计算，其它模拟计算子系统通过 A/D、D/A 转换与之交互。

20 世纪 70 年代，航空电子数字信息化使得数字计算机逐步替代了集中式航空电子系统中的模拟计算机，功能各自独立的航电子系统在与中心控制计算机进行通信的同时，使用 1553B 等数据总线相互交联[7]，集中式航空电子系统结构发展为集中分布式系统结

构。

第二代联合式航空电子系统结构包含集中式与集中分布式航空电子系统结构，具有故障传播壁垒（fault propagation barrier）特性，但存在软、硬件耦合大、系统升级困难、系统结构灵活性和开放性差等特点[4]。

3）综合式航空电子系统结构

20 世纪 80 年代，美国空军“宝石柱”（Pave Pillar）[8]计划对航空电子系统结构在模块化、开放式等方面提出了更高的要求，推动其正式进入综合化。第三代综合式航空电子系统结构将各功能分区通过高速数据总线互联[5]，使用通用综合处理器或综合核心处理机完成传感器信号与数据的高度综合处理。

4）先进综合式航空电子系统结构

1990 年以来，航空电子系统的综合化、网络化、资源共享性随着 “宝石台”（Pave Pace）计划的开展较“宝石柱”计划阶段有了进一步加强，因此被称为第四代先进综合式航电系统结构[9]。它采用开放式体系结构，充分应用 COTS 产品实现软、硬件功能单元，推动雷达、通信导航识别（Communication, Navigation and Identification, CNI）等射频部件及天线孔径的模块化、标准化及可现场重构，使用统一网络连接所有功能区。

综合式和先进综合式航空电子系统结构通常统称为综合模块化航空电子（Integrated Modular Avionics, IMA）。IMA 通过实时机载网络系统，有效、充分地利用并综合各航空电子系统子系统和设备的信息，从而使系统性能达到更高的水平[4]。IMA 作为开放式的标准化体系结构，能够有效降低飞机生命周期费用，更易于系统集成、维护和技术升级[10]。由于 IMA 结构中的模块比较集中，使其存在综合机柜安置难度大、背板设计较复杂、电缆偏多[4]，故障管理能力和容错能力较差等不足。

5）分布式综合模块化航空电子系统结构

NASA 在“猎户座”（Orion）航天飞机电子系统的实践中，提出了 DIMA 的结构设计，欧盟框架 7（7th Framework Program）项目中提出的“IMA2G”的互连技术也与 DIMA 结构相似。

DIMA 在继承 IMA 优势的同时，将多个分布式模块电子分离到与信号源（作动器和传感器）相近的区域，且与前端信号源的预处理相结合，将大机柜拆分为若干小机柜，进而降低了系统复杂度，克服了散热、机柜尺寸、布线及电磁兼容等[11]问题。DIMA 通过采用高容错的实时通信网络，将分布于整个飞机范围的多个分布式模块电子相连接，不仅为各子系统提供了故障传递隔离，也提高了整个航空电子系统的资源综合与信息共享的程度[12; 13]。DIMA 的系统结构如图 1-1 所示。

航空电子系统结构已经经历了 4 代、6 个阶段的发展，图 1-2 描述了航电系统结构的演化过程。

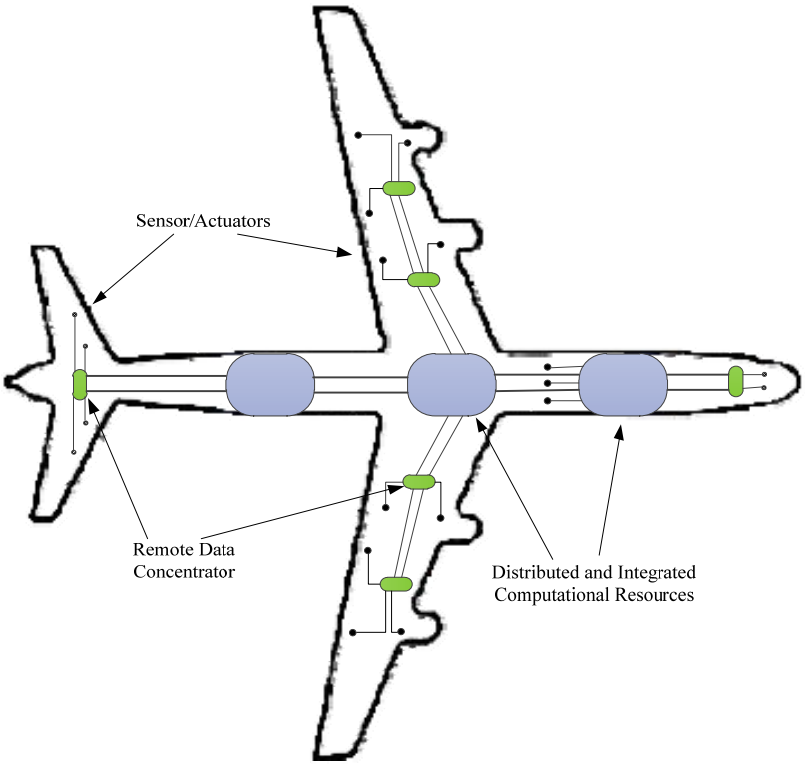

图 1-1 分布式 IMA 系统结构

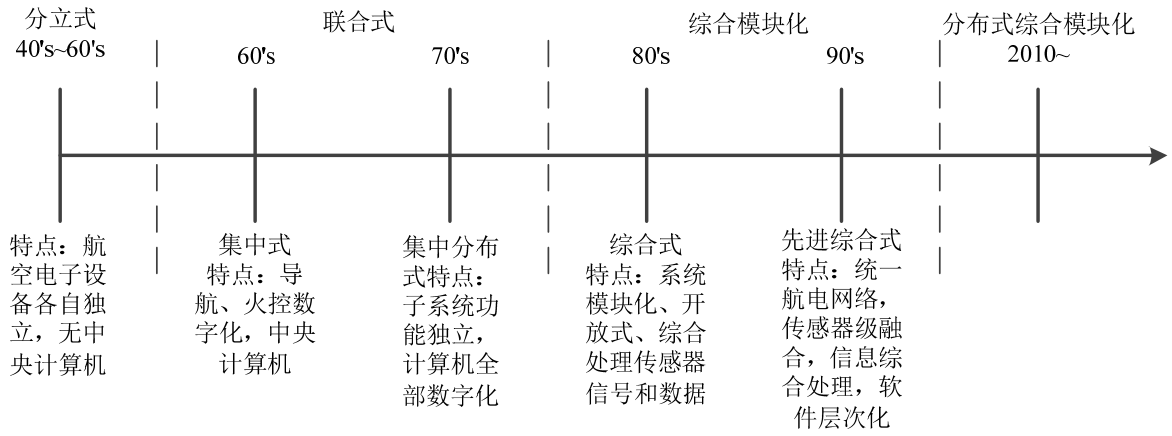

图 1-2 航空电子系统结构演化过程

1.3 航空电子网络性能研究现状

1.3.1 网络实时性分析方法研究现状

分布式综合模块化航空电子系统对软硬件资源共享、数据资源共享、各模块间相互资源调度和访问等方面提出了更加严格的要求。作为航空电子系统的“中枢神经”，航空数据网络承担着机载设备和航空电子系统子系统之间信息传输的重要任务，其实时性与时间确定性将直接影响飞机的整体性能。

目前对航空电子数据网络实时性与时间确定性的研究主要有以下三种方式[14]：

（1）网络模拟：采用网络排队论方法对网络情景进行模拟再现，观测一定条件下的网络性能。此方法采用队列和对象等结构体刻画网络实体，建模复杂，适用于简单网络的测试。

（2）模型检验：采用时间自动机理论，考察系统所有可能的状态，以得到最差情况下的端到端延迟的精确理论分析值。此方法时间复杂度高，模型构建方式复杂，适用于有穷状态系统的测试。

（3）网络演算：利用网络演算结论，计算网络性能的最坏边界。

Hussein Charara 等人[15]使用上述三种方法分别对特定 AFDX 网络配置下的端到端延时上界进行了评估，在实际应用中网络演算方法更为适用。网络演算可以进一步分为两个研究分支：

确定性网络演算[16; 17]，是由 Cruz 提出的一种基于最小加代数的确定性排队分析理论，现已广泛应用于各种网络性能分析领域，如实时系统[18-20]、AFDX[15; 21; 22]等。确定性网络演算能够在网络节点按照规定的服务方式工作的前提下，依据系统到达和服务曲线迅速地计算得到数据流在网络传输过程中的传输延迟和流量积压的上界。

随机网络演算是对确定性网络演算的随机扩展[23; 24]，它不仅充分考虑到数据流的突发性和自相似等特性，还考虑到网络信道的接入拥塞等一些实际情况。基于随机网络演算的网络性能分析，能够在向数据流提供一定程度 QoS 保证的同时，有效提高网络资源的利用率。F. Ridouard 等人[25; 26]利用随机网络演算方法分别计算了不同调度算法下的 AFDX 网络端到端延迟上界，同时证明了随机型网络边界要明显小于确定性网络边界。

除了上述方法之外，L. Malta 等人[27]通过使用数学编程的方法得到了网络的延迟边界，H. Bauer 等人[28; 29]提出了应用轨迹法计算确定的延迟边界。轨迹法是依据数据帧在网络拓扑结构上的传输轨迹来计算传输延时。

1.3.2 AFDX 实时性优化研究现状

AFDX 网络实时性优化研究主要包括网络管理和网络调度算法研究。王平等人[30]建立了一个有效的网络管理模型来提高 AFDX 网络的传输性能，并在此基础上提出了混合管理策略[31]。这些模型的实现和应用进一步提高了网络管理的智能化和自动化。A Al Sheikh 等人研究了配置虚链路参数和路由虚链路的不同方法，并提供了能够获得最小网络时延的虚链路设计方法[32]。Dongha 等人提出了一种可行的算法以此得到 AFDX 网络虚链路的具体配置，并通过验证该算法获得的虚链路配置可以有效地提高网络的实时性[33]。

AFDX 网络调度算法研究按照一定的规则从不同的等待队列中，合理选择队列并将其中的数据包发送到输出端口，主要包括终端系统虚链路调度算法的研究与交换机调度算法的研究。

（1）AFDX 终端系统虚链路调度算法研究

陈昕等人通过对 AFDX 网络数据流的分析，将数据流划分为三个优先级，改进了传统的优先级算法，提出了 AFDX 虚链路静态优先级调度算法[34]。然而，由于需要实现虚链路到优先级的映射表配置，该算法的复杂度较高。贾卫松等人[35]提出了索引表静态优先级调度方法，仿真结果表明其具有良好的实时性。Suthaputchakun 等人[36]通过在端系统分别采用轮询、FIFO、最小 BAG、最小帧长和最大队列长度的调度算法，分析比较了不同调度算法对时延和抖动的影响。Zheng 等人[37]通过在逻辑层和物理层分别采用 Frank Wolfe 和先验算法，对虚拟链路规划进行了优化。Gurjar 等人[38]提出了采用 Small Frame Earliest 调度算法有效降低了虚链路在端系统的时间抖动。

（2）AFDX 交换机调度算法研究

AFDX 交换机调度算法大多都是借鉴传统以太网上的调度算法，主要有以下 4 类:

1）先来先服务调度算法（First In First Out, FIFO）

AFDX 标准协议支持 FIFO 调度算法，数据包转发顺序仅依照其到达调度服务器的顺序决定，具有一定的公平性。Scharbarg 等人[21]采用随机网络演算的方法得到了数据流在 FIFO 算法下的网络端到端延时。计算的结果显示 FIFO 算法不区分优先级，容易引起某些紧急任务数据包的传输超时，不能很好地满足不同数据流的传输要求。

2）基于轮询的调度算法（Round Robin, RR）

传统轮询调度算法无区别地对分配到各个队列的分组进行循环调度服务。但当各个队列所传输的分组长度不同时，队列间无法公平地接受同等数量的服务，影响了时延保证。

一些优化的算法能够改进传统轮询调度算法的时延特性和其在变长分组环境下的不公平性，如差额轮询调度算法[39; 40]（Deficit Round Robin, DRR）、嵌入式差额轮询调度算法[41]（Nested Deficit Round Robin, NDRR），可变轮询调度算法[42; 43]（Elastic Round Robin, ERR），加权轮询调度算法[44]（Weighted Round Robin, WRR），差额加权轮询调度算法[45]（Deficit Weighted Round Robin, DWRR），可变差额加权轮询调度算法[46]（Variable Deficit WRR）以及紧急轮询调度算法（Urgency Based Round Robin, URR）[47]等。

轮询类调度算法不需要维持全局时间变量，也不需要任何的排序过程，这类调度算法实现较为容易，复杂度为 $O(1)$。

3）基于优先级的调度算法

最主要的基于优先级的算法是优先级队列（Priority Queue, PQ）和队列长度阈值（Queue Length Thresh, QLT）算法。Bauer 等人在交换机中应用了静态优先级调度算法（Static Priority, SP），并使用轨迹算法得到了不同数据流的网络延时边界[24]。所得延时边界虽优于 FIFO 调度算法，但仍无法完全满足航空电子系统中不同的传输要求。Flores 等人将中断重发机制引入静态优先级调度中，保证了高优先级数据的实时性[48]。Fang 等人[49]通过使用信用量的概念细化服务数量，并按照安全关键等级对虚链路进行分类，

保证了调度算法的有效性和高效性。

4）基于广义处理器共享模型的调度算法（Generalized Processor Sharing, GPS）

GPS 模型作为理想化的流媒体模型，队列之间具有相同的优先级。因为基于业务流无限可分的假设，调度器能够依照各个队列的对服务器的共享比例同时服务所有队列。在实际分组传输系统中，GPS 模型是无法实现的，继而出现了一类在分组环境中逼近 GPS 模型的分组调度算法，如：加权公平排队算法[50-52]，自时钟公平排队算法[53; 54]，开始时间公平排队调度算法[55]，最坏情况加权公平排队[56]等。基于广义处理器共享模型的调度算法使用类似 “分组有序排队” 的机制。这种机制会根据系统状态为每个到达的数据包计算各自的时间戳，并且将这个时间戳作为数据包调度优先顺序的重要指标。主要的不同在于计算时间戳的不同方法。这些基于排序优先级的调度算法在公平性、时延及时延抖动等方面具有较好的性能；但是时间戳的计算、存储及排序复杂性较高，不利于硬件实现，在实际应用中受到了极大的限制，不适用于高速网络设备中[52]。

1.3.3 时间触发通信协议发展现状

具有容错机制的时间触发技术因其所具有的实时性和时间确定性特点而成为了工业界的研究热点。TTTech computertechnik AG 公司提出的时间触发协议（Time-Triggered Protocol, TTP），是首个被 SAE 标准化的通信协议（SAE AS6003）。安全关键的容错系统、可管理的模块化设计以及严格确定的通信机制，使其在航天、航空、铁路和汽车工业中都有着广泛的应用。TTP 已成功应用于航空航天等领域[57]，如：洛克希德·马丁公司 F-16 飞机的全权限发动机数字控制系统、意大利 Aermacchi 公司的 M-346[58]以及波音 787 客机。

时间触发结构（Time-Triggered Architecture, TTA）是在 TTP 基础上为现代航空控制系统和航空电子开发的。系统中的各个节点在物理上具有分布性和分立性，能够在规定的时间内完成通信任务[59]，且符合综合模块化航空电子的综合特性，可用于飞机智能控制系统的开发。

时间触发以太网（Time-Triggered Ethernet, TTE）[60; 61]是基于 TTA 实现的，其规范 SAE AS6802 已经于 2011 年发布。TTE 通过在以太网标准之上引入时间触发通信机制，使其网络硬件设备与以太网相兼容，有效降低了系统的开发成本。TTE 采用时分多路访问技术，实现了数据传输的无冲突和低时延[62]，广泛适用于航空航天等领域。TTE 已经分别被美国国家航空航天局的“猎户座”载人航天飞行器项目和美国西科斯基飞机公司（Sikorsky Aircraft Company）的新一代模块化飞行管理系统选取作为主干网络[63]。Zhu 等人通过综合优化多数据流到达曲线的特点提高了 TTE 在最差情况下的时延，并采用网络演算方法进行了验证[64]。Zheng[65]等人提出了基于 Path-hop 的调度表设计方法，并按照数据流类型依次完成规划以适用于航空网络。Li[66]1 等人将 TTE 调度规划问题视为 NP 问题，采用 bin-packing 与遗传算法相结合的思路，降低了时间规划的时间。国内在

TTE 领域的研究和开发仍处于推广阶段。Wisniewski[67]等人采用贪婪调度算法在资源有限的硬件设备中完成了调度设计。

为了提高控制器局域网络（Controller Area Network, CAN）总线数据传输的可预测性和可靠性，提出了基于 TTA 协议的 TTCAN（Time-Triggered CAN）总线，多应用于汽车通信系统[68-72]。

1.4 分布式网络数据完整性技术研究现状

随着多媒体业务在航空电子系统中的快速发展，航空数据在数量、多样性和重要性等方面发生了巨大的变化。航空电子数据不仅包含传统的航空电子数据，同时相应地增加了乘客可访问的影音娱乐等数据。在 DIMA 系统结构下，数据的存储均在分布式网络服务器中，由于乘客拥有一定的数据访问权限，服务器中存储的数据，尤其是飞行核心数据将面临潜在的被篡改、删除等危险。

目前分布式网络中数据完整性[73; 74]的研究主要集中在可恢复证明（Proofs Of Retrievability, POR）方案和可证明数据持有（Provable Data Possession，PDP）方案。PDP 方案仅能够对存储数据的完整性进行检测，无法确保对损毁数据的恢复 [75; 76]；POR 方案保证了存储数据在一定程度损坏的情况下，利用编码技术仍然能够提供数据的可恢复性。

Ateniese 等人[77]最先提出了 PDP 方案，该方案对每一个数据块生成一个同态可验证“标签”，并与数据本身一同存放在服务器上[78]。当用户对数据的持有性进行验证时，通过随机选取一些数据块向服务器发出挑战。服务器根据挑战中所包含的数据块选取信息及所持有的“标签”数据，生成并返回数据持有的证据。由于“标签”所具有的同态性，在证据生成过程中随机选取的数据块“标签”能够聚合成一个值，有效降低了带宽的使用。由于该方案欠缺对数据更新的考虑，Ateniese 等人在后续工作中又提出了一种支持文件更新功能的 PDP 方案[79]。

2007 年，A. Juels 等人[80]提出了 POR 的概念，并提出基于“哨兵”（sentinel）的 POR 方案。Bower 等人[81]提出的 HAIL（High-Availability and Integrity Layer）方案将 POR 的概念扩展至多个存储服务器中，当采用 POR 方案检测到某一服务器中数据被破坏时，可以使用其他存储服务器中的数据副本进行恢复，Yi 等人在 PDP 方案中也引入了多副本的概念[82]。

Shah[83]首次将第三方审计者（Third Party Auditor, TPA）的概念引入数据完整性检查中，同时提出了具有隐私保护特性的检查方案。

Wang 等人[84]于 2011 首次实现了数据完整性验证的批处理，使得 TPA 能够同时响应多个验证请求。同时，该方案通过采用梅克尔散列树（Merkel Hash Tree, MHT）存储结构解决了数据更新问题，提高了实用性。

肖达等人[85]于 2009 年在国内首次提出了数据完整性检查方案设计，该方案采用校

验块循环队列的挑战更新机制，能够动态增加有效挑战次数[86]。

随着数据加密技术以及数据签名技术的发展，网络数据完整性验证方法中对数据的处理方法也出现了多样性，比如：基于身份的签名技术[87-90]、基于属性的签名技术[91]和同态线性认证技术[92-96]等。

同时，通过采用多种验证数据结构，PDP 方案中所缺少的文件更新功能也得到了有效地支持，其中比较典型的可验证数据结构有 CBT 存储结构[97]、优化的 MHT[95; 98]存储结构和动态二叉树[99]等存储结构。

1.5 存在的问题

DIMA 系统体系结构对航空电子网络传输性能及网络数据完整性提出了更高的要求，目前在相关研究中存在的问题有：

（1）已在新型民用飞机中使用的航空网络技术 AFDX 并没有完全消除传统以太网传输中的时间不确定性，各种数据流竞争交换机内部资源可能导致更大的时间延迟、抖动和不公平性，无法满足现代航空应用对实时性和时间确定性的要求。时间触发 AFDX 网络在 AFDX 网络基础上扩展了带宽分区的概念，将时间触发流所占用的带宽视为一种特殊的、具有高确定性的带宽分区。这种扩展性带宽分区方法为航空电子数据传输的高时间确定性提供了保证，但增加了系统配置的复杂度。同时，采用时间触发流与速率限制流传输的 AFDX 数据对带宽存在竞争关系，在一定程度上可能增加了速率限制流数据传输的时间不确定性和延迟抖动。

（2）时间触发 AFDX 网络中时间触发数据流依照预先规划好的调度表进行调度。如何设计调度表，使其能够具有对时间触发数据流的保护性，减少对非时间触发数据流的影响以及提高带宽利用率，都是伴随时间触发概念的引入所需要解决的问题。

（3）时间触发 AFDX 网络结构中速率限制流在端系统和交换机的数据传输具有多路复用的特点，现有的调度算法在复杂性、公平性以及时间确定性方面存在矛盾，因而，在保证时间确定性的条件下，优化调度算法复杂性，提高系统公平性是目前急需解决的问题。

（4）在 DIMA 系统体系结构下，航空电子数据均存储在分布式网络服务器中。随着多媒体业务在航空系统中的快速发展，乘客拥有了一定的数据访问权限，这对于服务器中的数据将带来潜在的被恶意篡改、删除等危险。为了确保数据在分布式网络服务器中的完整性，航空电子系统应具有定时验证机载数据完整性的功能。随着机载数据数量的不断增加，传统的对被查数据完全下载后进行检查的方法并不适用于资源紧缺的航空电子系统。在降低网络通信和处理资源开销的前提下，对分布式环境中存储的航空数据进行完整性和可靠性检查将极大地提高航空电子系统的可靠性和安全性。

1.6 论文的研究内容和组织结构

通过对国内外大量相关参考文献的收集、整理和研究，本文分别针对新一代 DIMA

系统存在的网络传输确定性以及网络存储设备中的数据完整性证明问题进行了研究。

为了提高 DIMA 系统中网络传输的实时性和时间确定性，在已有 AFDX 基础上引入时间触发机制，提出了时间触发 AFDX（Time-Triggered AFDX, TTAFDX）网络的概念[100]，通过依次对时间触发 AFDX 从网络体系结构设计、时间触发虚链路调度算法与速率限制虚链路调度算法设计与优化方面进行研究，使时间触发 AFDX 在兼容 AFDX 硬件设计的同时，能够保证时间触发数据流传输的时间确定性和实时性，并有效降低非时间触发数据流的传输时延。

在提高了航空电子网络中数据传输的可预测性、可达性和时间确定性之后，航空电子数据的存储过程将是所面临的又一重要研究内容。由于 DIMA 系统中数据存储的分布性以及网络结构的同一性，航空电子数据在存储服务器中面临前所未有的威胁，为了有效地验证航空电子网络存储服务器中数据的完整性，提出了适用于 DIMA 的数据完整性验证模型，并按照静态数据和动态数据的特点分别进行研究。论文各部分研究内容间的组织关系如图 1-3 所示。

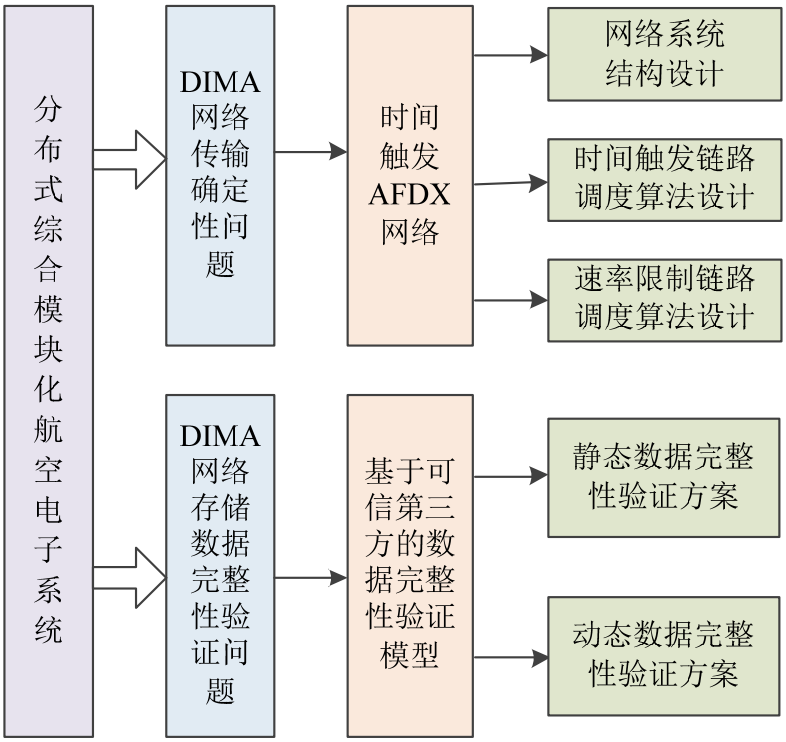

图 1-3 研究内容之间的相互关系

论文共分为七章，各章节内容安排如下：

第一章，绪论。概括介绍了 DIMA 系统的体系结构以及应用中面临的一些主要问题的研究现状，最后介绍了论文的研究内容和组织结构。

第二章，DIMA 的时间触发 AFDX 体系结构设计。提出了一种时间触发 AFDX 网络的体系结构，并对时间触发 AFDX 的网络结构及网络协议进行了全面地分析与改进，包括时钟同步技术、虚链路类型、协议栈和数据帧格式等。时间触发 AFDX 网络设备能

够有效解决分布式综合模块化航空电子系统中时间关键消息传输的时间确定性问题。另外，时间触发 AFDX 硬件设计与 AFDX 完全一致，在兼顾开发成本和端系统之间兼容性的同时，便于时间触发 AFDX 端系统的实现。

第三章，DIMA 网络的时间触发虚链路调度算法设计与实时性分析。针对时间触发虚链路的调度问题，设计了基于帧长度优先原则和基于周期优先原则的端系统时间触发虚链路调度算法。搭建典型网络，通过网络演算方法计算分别对比了 TTAFDX 与采用 FIFO 调度算法和静态优先级调度算法的 AFDX 网络的实时性能。计算结果表明：TTAFDX 中的时间触发虚链路的延迟主要由固定延迟部分组成，并且速率限制虚链路的实时性较 AFDX 中的低优先级虚链路也有所提高。证明了 TTAFDX 体系结构和调度算法在兼容 AFDX 的同时，能够改善网络的时间确定性，并适用于具有硬实时传输要求的航空电子系统。

第四章，DIMA 网络的速率限制虚链路调度算法研究与实时性分析。本章采用轮询调度的基本思想，提出了基于逐次最小定额轮询（Successive Minimal-Quantum Round Robin, SMQRR）的速率限制虚链路调度算法。SMQRR 调度算法通过改变对每一个数据流权值所对应的数据量的连续服务方式，在保留轮询调度算法 $O(1)$ 时间复杂度的同时，有效克服了分组长度不同带来的不公平性并避免了数据流可能长时间无法获得服务的情况。通过理论分析和仿真验证，SMQRR 调度算法在公平性和时延上界均优于加权可变轮询调度算法和加权差额轮询调度算法。

第五章，DIMA 网络中静态数据存储完整性验证方案。提出了适用于 DIMA 系统结构的航空数据完整性验证模型。该验证模型除用户和存储服务器之外，依靠可信第三方完成对存储服务器中数据的完整性验证。依照验证模型，采用线性签名（Linear Signature）技术对航空电子数据在存储前进行处理，使其验证标签具有同态特性，能够有效降低验证过程中的各种开销。同时，结合前向纠错码编码技术，在数据少量丢失的情况下使数据具有可复原性，以提高数据的完整性保障。

第六章，DIMA 网络中动态数据存储完整性验证方案。提出了采用梅克尔散列树（Merkel Hash Tree, MHT）或跳转列表（Skip-list）的数据存储索引结构，以支持飞机在飞行中数据的动态操作。通过与线性签名、双线性映射签名（Bilinear Map Signature）技术相结合，所提出的数据完整性验证方案在验证过程中均具有常量级的计算开销、存储开销和通信开销，适用于带宽有限的航空电子网络系统。

第七章，结论与展望。对论文主要工作和研究成果进行总结，并提出了进一步研究的方向。

2　DIMA 的时间触发 AFDX 网络体系结构设计

空客 A380 和波音 787 作为 IMA 系统应用的代表，采用 AFDX 互连技术为各航空电子系统子系统间的数据融合和功能综合提供有效支撑。AFDX 是在 IEEE802.3 以太网基础上利用 COTS 技术和开放式标准发展起来的一种全双工、高速率、双余度网络[101]。它具有共享网络资源、节省成本、便于远程监控、扩展容易与维护方便等优点。虽然 AFDX 一定程度上提高了网络数据传输的实时性和可靠性，但当多任务传输时可能产生的竞争链路共享冲突，仍会影响数据传输的实时性和通信系统的时间确定性。对于时间关键消息，传输时延及时延抖动的确定性在 AFDX 网络中仍不易得到保证[100; 102; 103]。

随着时间触发通信机制在分布式实时系统领域的提出[104; 105]，DIMA 系统中各航空电子系统子系统或功能区之间的并行处理和精确同步成为可能。本章通过将时间触发机制引入 AFDX 网络，设计完成了时间触发 AFDX 的体系结构，提高了 DIMA 系统中时间关键消息的时间确定性，并有效控制了对现有设备改造的技术与商业风险[103]。

2.1 时间触发 AFDX 网络结构

TTAFDX 有机融合和继承了 AFDX 与 TTE 的特点，网络系统主要由航空电子系统子系统、端系统（End System, ES）和 TTAFDX 内部通信链路组成，如图 2-1 所示。其中：

（1）航空电子系统子系统：与传统的航空电子系统一致，如轮胎压力监视系统、飞行控制计算机和飞行管理系统等。各航空电子系统子系统通过一个端系统与 TTAFDX 网络建立连接[106]。

（2）端系统：嵌入在航空电子计算机中，为航空电子系统子系统与 TTAFDX 通信链路的连接提供了一个应用程序接口（Application Programming Interface, API），确保 TTAFDX 网络中各航空电子系统子系统之间能够通过一条简单的信息传输接口进行可靠、安全的数据传输。一个端系统能够支持多个航空电子系统子系统的数据收发处理，不同的端系统通过多个交换机互相连接。图 2-1 中有两个 TTAFDX 端系统是为航空电子系统子系统提供通信接口，另一个则是用来为 TTAFDX 网络提供网关支持。该端系统为航空电子系统子系统与外部的 IP 网络节点之间提供了通信路径[107]。

（3）TTAFDX 内部通信链路：由一个能够将数据帧传送到正确目的地的交换以太网组成，所提供的全双工交换以太网接口能够有效地连接各航空计算机，完成端系统与交换机间的点对点通信。

TTAFDX 交换机作为 TTAFDX 内部通信链路的重要组成部分，采用基于缓冲的存储转发机制为网络上的各端系统提供交换通路，实现各数据链路的帧转发功能[108]。每个 TTAFDX 交换机最多支持全双工连接 24 个端系统，形成接入交换网络；TTAFDX 交换机之间通过高速背板连接，构成层叠式星型拓扑结构的骨干交换网络[109]，使得

TTAFDX 网络具有结构灵活、扩展性强的特点。TTAFDX 交换机继承了 AFDX 交换机的 5 个功能模块并在各模块进行了针对时间触发数据帧的适应性改动，如图 2-2 所示。

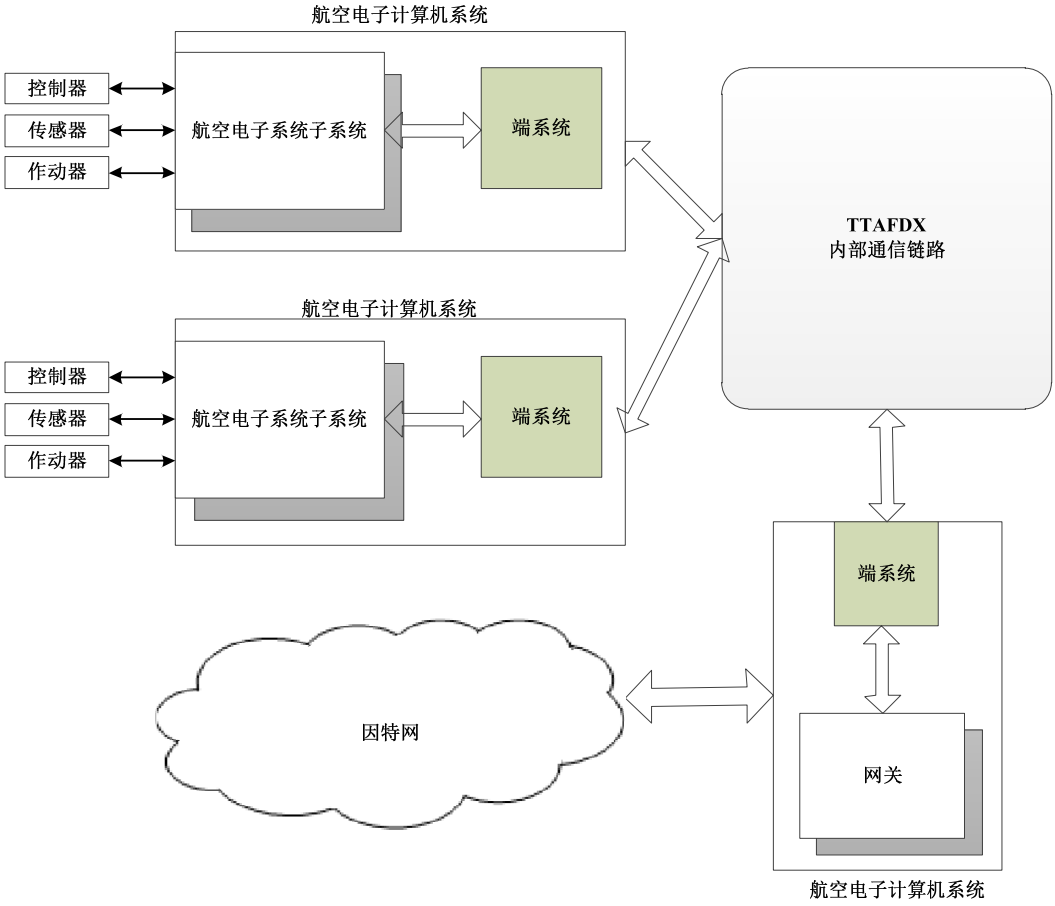

图 2-1 TTAFDX 网络结构

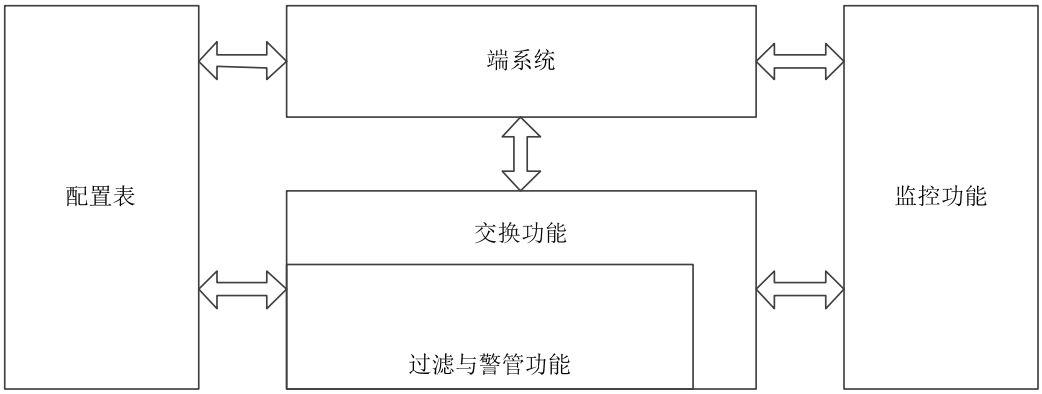

图 2-2 TTAFDX 交换机主要功能模块

（1）过滤警管模块：对通过交换机的数据帧按照传输延迟及帧内容进行检测和过滤[110]，避免无效帧的传输以节省带宽资源。

（2）交换功能模块：设计人员预先根据网络配置情况设计静态路由表，并在 TTAFDX 网络启动时载入交换机，它不会随网络结构的改变而改变。TTAFDX 为了控制寻址时间，采用静态路由表完成帧转发过程中目的端口的寻址。

（3）配置模块：交换机系统按照 TTAFDX 网络集成时定义的各类配置文件对相应模块完成配置操作，例如用于实现时间触发数据无冲突收发操作的时间调度表和静态路由表等。

（4）监控模块：对网络系统中的交换机工作状态进行监控。

（5）端系统模块：其功能遵循 TTAFDX 网络中端系统的规范[111]。

由于 DIMA 系统中计算资源高度共享，一台航空电子计算机可以通过使用分区机制为多个航空电子系统子系统提供计算环境，即通过限制每个分区的地址空间和 CPU 工作时间来对各航空电子系统子系统加以区分和隔离，如图 2-3 所示。分区机制能够防止一个分区内的航空电子系统子系统在运行出现错误的时候不会影响其它分区内运行的航空电子子系统的正常运行，并易于验证和确认整个系统的安全性[112]。

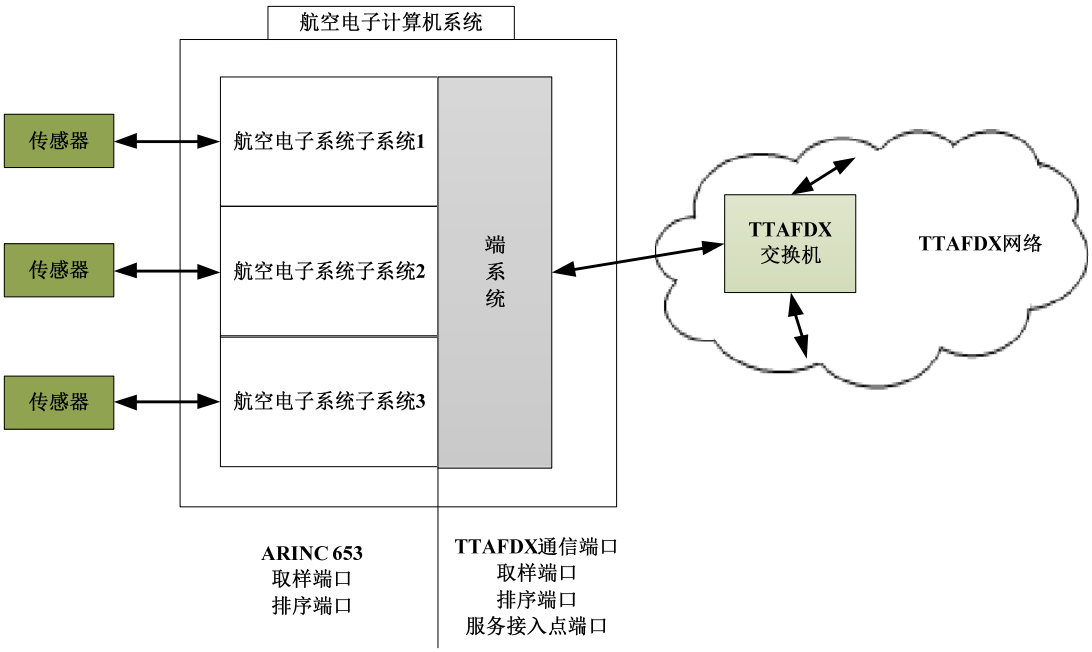

图 2-3 TTAFDX 系统的航空电子子系统和端系统

各航空电子系统子系统都是通过应用软件使用通信端口发送信息来进行彼此之间的交流，通信端口为发送和接收信息提供了一个编程机制。航空电子应用软件接口标准规范——ARINC 653 定义的取样端口（Sampling Port）和队列端口（Queuing Port）两种类型通信端口在航空电子系统子系统中扮演重要的角色，因此，TTAFDX 端系统提供了与之匹配的 API 接口来支持取样和队列端口。同时，TTAFDX 也支持 ARINC 664 中定义的服务接入点（Service Access Point, SAP）端口，用来进行 TTAFDX 系统与非 TTAFDX 系统之间的通信，如数据加载和网络管理等航空电子系统的维护应用。

2.2 时间触发 AFDX 网络技术

2.2.1 时钟同步技术

只有当时间触发 AFDX 网络中的时钟具有全局一致性的前提下，依靠时间触发的数

据传输任务才能够保证在调度时刻独享节点网络带宽，避免多任务传输时竞争链路共享冲突带来的时间延迟和抖动，确保数据传输的时间确定性[113]。网络时钟同步机制主要有 3 类：1）网络时间协议（Network Time Protocol, NTP）[114]/简单网络时间协议（Simple Network Time Protocol, SNTP）[115]；2）IEEE 1588[116]；3）基于 TDMA 的时钟同步技术。NTP/SNTP 和 IEEE 1588 进行时钟同步操作具有不确定性，即在任意时间点都可进行时钟同步操作，这不仅使时钟同步数据包面临潜在的排队延迟，从而降低时钟的精确度[117]，而且给时间触发数据帧的发送引入了时间不确定性，可能会导致数据帧在帧过滤过程中的丢失。

TTAFDX 网络采用 SAE AS6802 协议中提出的时钟同步技术，通过时钟同步协议控制帧（Protocol Control Frame, PCF）在每个调度周期的固定时刻进行时钟同步。PCF 帧长为 28 字节，其中当前时间所在的位置由前 4 个字节记录，最后 8 个字节称为透明时钟（Transparent Clock, TC）域，用于以 2^{-16} 纳秒为计时单位累计 PCF 的传输延迟和驻留时间。

在 TTAFDX 网络中，各网络节点根据时钟同步的功能，担当相应的同步控制器（Synchronization Master, SM）、同步客户端（Synchronization Client, SC）和压缩控制器（Compression Master, CM）角色[102]。TTAFDX 网络中的交换机需要支持透明时钟，用以累计数据帧在交换节点的驻留时间。

时钟同步操作使用两个基本功能函数：固化函数（Permanence Function）和压缩函数（Compression Function）。在接收端，数据帧之间的精确时间差和绝对时间顺序都由固化函数通过数据帧间的相对时刻恢复完成， SM，SC，CM 中均需要实现该函数。压缩函数将与之相连的 SM 时钟进行加权作为 CM 同步域的基准时钟[102]，仅在 CM 中实现。

2.2.2 虚链路技术

传统以太网中交换机通过以太网的目的地址(目的 MAC 或 IP 地址)选择路由帧。TTAFDX 网络中则使用目的地址后 16 位的虚链路（Virtual Link，VL）ID 来确定数据帧的路由路径。虚链路概念与 ARINC 429 总线的点对多点传输特性相类似，为一个源端系统提供了向一个或多个目的端系统的单向逻辑路径[118]。端系统可以支持多条虚链路，但这多条虚链路将共享物理带宽，并通过交换机进行交换与多播，有效地减少了连接器和电缆数量，降低了机载负重[119]。虚链路通过带宽分配间隔（Bandwidth Allocation Gap, BAG）、最大帧长（L_{max}）和时延抖动（Jitter）来实现流量控制的功能，确保了所有虚链路之间逻辑上的独立，从而避免了在同一物理链路上不同 VL 间的相互干扰。

TTAFDX 网络定义了两类虚链路：时间触发虚链路（Time-Triggered VL, TTVL）和速率限制虚链路（Rate-Constrained VL, RCVL）[120]。

采用 TTVL 承载时间关键消息，在全局时钟精确同步的条件下，严格按照规划的时

间调度表进行触发，能够确保其时间确定性和实时性，并且避免数据在链路上发生共享冲突。RCVL 优先级低于 TTVL，用以承载传统 AFDX 网络中的 VL。

2.2.3 冗余管理技术

TTAFDX 通过冗余链路来提高数据传输的可靠性，防止因交换机的意外故障或者链路断开的原因导致系统崩溃。在冗余传输模式下，源端系统将数据帧同时通过 TTAFDX 系统中两个相互独立的交换网络 A 和 B 进行传输，如图 2-4 所示。在正常情况下，目的端系统将会收到两份同样的数据帧。

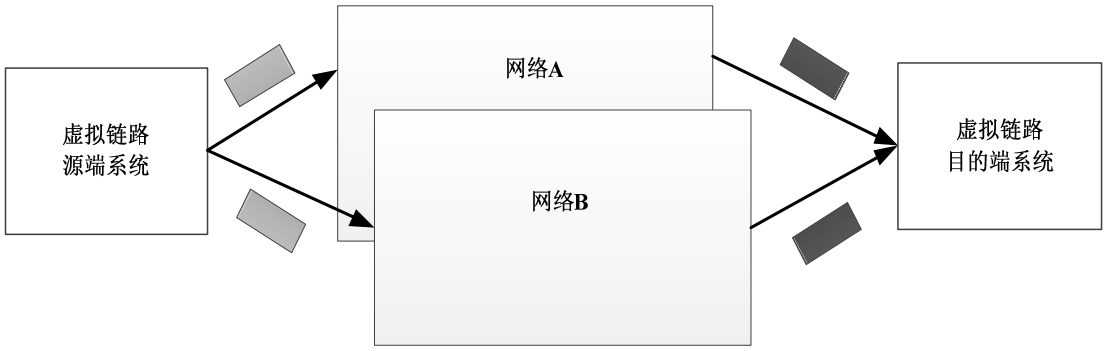

图 2-4 冗余传输模式

TTAFDX 设计了两种冗余管理机制：异步冗余管理（asynchronous Redundancy Management, aRM）和同步冗余管理（synchronous Redundancy Management, sRM）[103]。同步冗余管理与异步冗余管理的基本流程相似，如图 2-5 所示。

TTAFDX 网络中的 RC 流量数据帧沿用符合标准 ARINC 664 Part 7 的异步冗余管理机制。每条 VL 的源终端为每一帧发送的数据添加一个长度为 8 位的序列号（Sequence Number, SN），通过计数数据帧的发送顺序，以供完整性检查使用。SN 在 0-255 范围内取值，其中序列号“0”保留用于源终端初始化和系统复位。每条 VL 都有自己独立的 SN 计数器，每发送一帧，序列号加 1，当达到上限 255 之后回滚至 1 [30]。

VL 发送数据时，依据网络配置可使用网络 A 发送、网络 B 发送、或者两个网络同时发送。当使用两个网络同时发送时，所要发送的数据帧将由源端系统送到冗余管理模块进行复制，继而原帧和复制帧分别通过这两个独立的网络传输到目的终端系统[110]。

当链路层有数据传入时，首先需要检查成功到达帧的帧序列号是否符合顺序要求，即是否在[PSNN+1，PSNN+2]间隔内，其中 PSNN 为当前 VL 成功接收到的前一个帧的序列号。若完整性检查发现错误，则丢弃该无效帧，并通知网络管理模块。而没有错误发生时，则将所接收到的数据帧发送至冗余管理器进行冗余处理。目的端系统按照数据帧序列号顺序接收，并依据“先到优先使用”（First Valid Wins）原则重建一个没有副本的单一序列数据流。“先到优先使用”原则即当目的端系统一旦接收到一个有效的数据帧后，后续接收到的具有相同帧序列号的数据帧将被丢弃。根据帧序列号可以判断经冗余路径传输后同编号数据帧的到达顺序，避免重复帧在重构过程中的出现。若传输过程中

一个帧出现问题，则可用另一个帧代替[110]。

同步冗余管理以全局时钟同步为基础，对 TT 消息进行冗余管理。数据传输路径中的各个网络节点通过查询其上一节点的时刻调度表能够计算出 TTVL 帧到达该节点的时间点并确定数据帧接收时窗。控制器将接收在接收时窗内并通过完整性检查的多个有效数据帧，并在接收时窗关闭时，发送任一接收到的数据帧至应用层并删除所有存储的冗余数据帧。

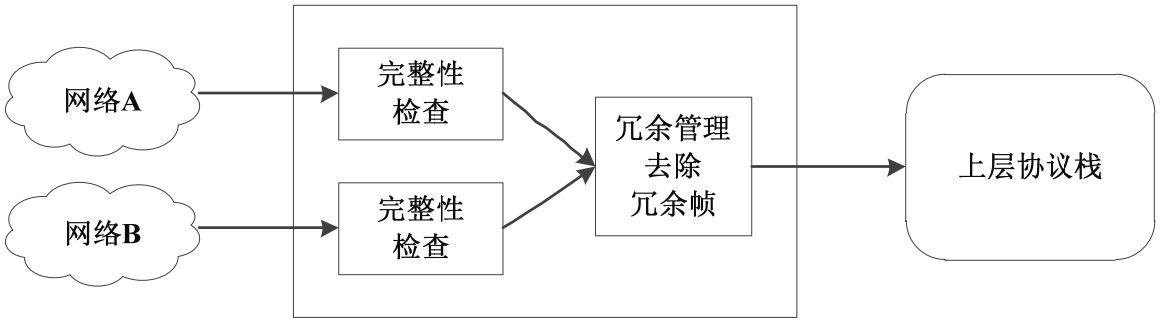

图 2-5 冗余管理机制

2.3 TTAFDX 网络协议

2.3.1 TTAFDX 网络协议栈设计

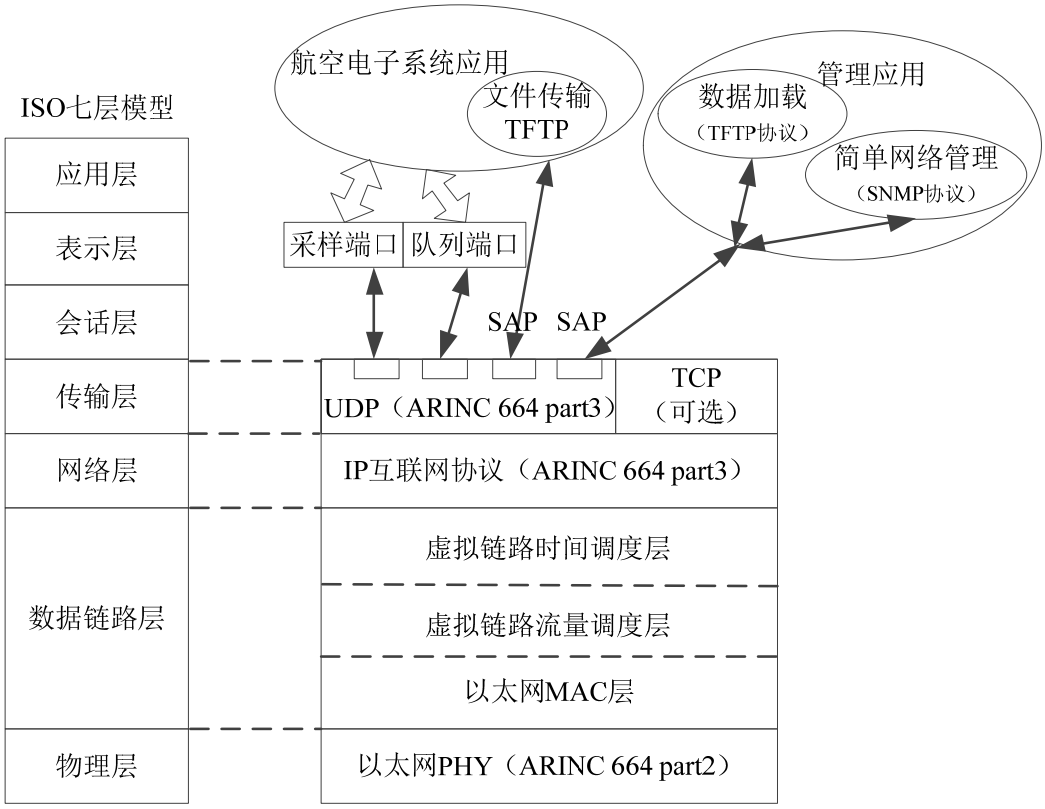

图 2-6 TTAFDX 网络协议栈

网络协议栈通过在端系统的发送和接收过程中对数据帧完成相应的控制调度，实现

网络中各端系统的通讯功能，其中包括按照规定的数据帧格式，分别在发送和接收过程中进行有效的数据帧封装和解析。TTAFDX 在传统 AFDX 基础上引入时间触发机制，并根据时间触发特性改进 VL 的调度策略。TTAFDX 网络的端系统协议与 AFDX 相同，符合 OSI 的 7 层网络模型，其中网络层、传输层和应用层与以太网相同。为了完全兼容标准 AFDX，TTAFDX 仅在数据链路层中增加了 VL 的时间触发通信机制，所增加的 VL 时间调度层不影响数据链路层以上的协议，标准 AFDX 中的 VL 以 RCVL 的形式在 TTAFDX 中传输[103; 120]，改进后的 TTAFDX 网络协议栈如图 2-6 所示。

2.3.2 TTAFDX 网络数据帧设计

TTAFDX 网络是在标准以太网基础上为了适用于时间触发机制而改造形成的，因此 TTAFDX 网络数据帧在帧结构上与以太网帧差别不大，唯一的差别是 TTAFDX 网络帧增加了一个位于以太网 FCS 字段前面的 SN 字段，用于冗余管理和完整性检查，此字段占用一个字节。TTAFDX 的数据帧结构如图 2-7 所示。

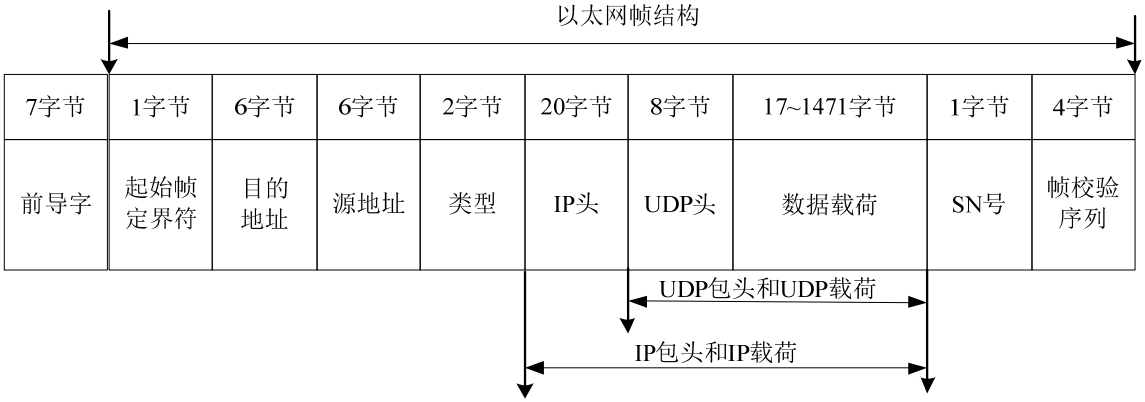

图 2-7 TTAFDX 数据帧

引入时间触发机制之后，相比于标准 AFDX 网络协议栈，TTAFDX 网络协议栈只是对链路层进行了修改，仅需要修改相应的标准 AFDX 数据帧格式的 MAC 地址部分。TTAFDX 沿用标准 AFDX 数据帧中目的 MAC 地址地 16 位 VL 标识符，并对其中的 32 位常量域进一步划分为 24 位的虚链路类型设置域和 8 位固定域，如图 2-8 所示。

在数据帧传输过程中，各网络节点通过对 TT 标识符域进行检测以判断数据帧所属虚链路的类型。若 TT 标识符域的高 4 位为“0110”，则该虚链路为时间触发虚链路，采用时间触发调度策略进行调度；若不是，则为速率限制虚链路，采用常规调度策略[103]。通过此方式，能够在不改变 AFDX 终端系统硬件结构的情况下，实现 TTAFDX 终端系统，体现了 TTAFDX 终端系统与标准 AFDX 终端系统的兼容性。

<table>
<tr><td colspan="3">48bit</td></tr>
<tr><td colspan="2">固定域 32bit</td><td rowspan="2">VL标识符
16bit</td></tr>
<tr><td>固定域 8bit</td><td>TT标识符 24bit</td></tr>
<tr><td>XXXX XX11</td><td>0110…XXXX</td><td>XXXX…XXXX</td></tr>
</table>

图 2-8 TTAFDX 帧的目的 MAC 地址

2.4 本章小结

DIMA 系统中各航空电子系统子系统和功能区间的数据融合与功能综合对航空电子网络互联技术提出了更高的要求。本章将时间触发机制引入现有航空电子网络 AFDX，提出了一种时间触发 AFDX 网络的体系结构。通过对时钟同步、虚链路、冗余管理等网络技术及网络协议进行适应性修改，使得时间触发 AFDX 在兼容标准 AFDX 的基础上，能够解决 DIMA 系统中时间关键消息传输的时间不确定性问题。

3　DIMA 网络的时间触发虚链路调度算法设计与实时性分析

在 DIMA 网络多任务传输时，TTAFDX 网络中的竞争链路会产生共享冲突，对通信系统的稳定性和数据传输的实时性产生影响[121]，TTAFDX 通过对 TTVL 采用表驱动调度方法，消除了传输过程中的竞争时延，保障了数据的时域通信行为和带宽需求。

本章设计了 TTVL 在端系统和交换机系统的调度算法，并提出了两种适用于 TTVL 的端系统时间调度表设计的方法。

采用对适航取证更具说服力的网络演算方法，分别对 TTAFDX 和使用静态优先级或先来先服务调度策略的 AFDX 进行端到端时延分析比较，充分验证了在时间触发机制下所提出的调度算法能够确保时间触发虚链路的时间确定性和实时性，同时对速率限制虚链路的实时性也有所提高。

3.1 时间触发虚链路端系统调度算法设计

TTAFDX 端系统首先通过流量整形器对数据源产生的不规律数据帧进行流量整形，整形后同一 VL 上相邻的两个数据帧将按照 BAG 时间均匀出现。TTVL 数据帧按照预先定义好的 TTVL 时刻调度表，由 VL 时刻调度器在相应时刻由时间触发机制完成发送。RCVL 数据帧则按照预先设计的调度方式，在 VL 时刻调度器的空闲时段完成发送。TTAFDX 端系统调度过程如图 3-1 所示[100]。

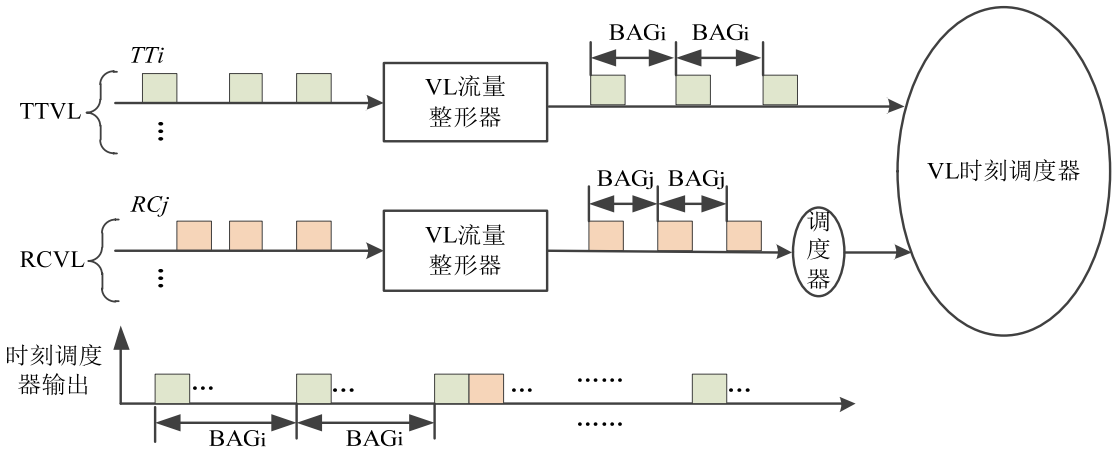

图 3-1 TTAFDX 端系统 VL 调度过程

由于 TT 数据帧由时间进行触发完成发送，TTAFDX 网络中的各个发送端都存有通过该节点发送的每个 TT 数据帧的发送时刻调度表。因此，TTAFDX 与 TTCAN 等总线在本质上都是基于静态表的调度，但通常情况下，TTAFDX 中各个调度表的内容是不同的，有别于 TTCAN 等总线结构中各个端系统具有相同的调度表[70]。

TTAFDX 端系统 VL 时刻调度表由一个矩阵周期（Matrix Cycle, MC）构成。MC 持续时间选用所有数据帧调度周期的最小公倍数或最小公倍数的整数倍[100; 122]，以保证各

个 TTVL 数据帧的基本调度周期。TTAFDX 中各 VL 在经过流量整形后，其基本调度周期，即：带宽分配间隔应符合系统所规定的取值范围：$2^k ms, k = 0,...,7$。由此可将所有带宽分配间隔的最小公倍数，即 128ms，设定为端系统 VL 时刻调度表的 MC。同时，采用 TTCAN 总线调度表设计思想，将 MC 细分为若干个基本周期（Basic Cycle, BC），BC 持续时间需要至少能够完成一个 TTVL 数据帧的传输任务，且由 TTVL 传输任务的最小周期决定。根据 TTAFDX 中所有 VL 的调度周期满足带宽分配间隔的规定，选取所有带宽间隔取值的最大公约数 1ms 赋值于 BC。

TTAFDX 只有在确保了网络中各个节点的时钟同步性前提下，离线设计的 VL 时刻调度表才能够保证 TT 数据帧在端系统不会发生物理链路的争用[100]。通过采用 SAE AS6802 协议中所提出的基于 TDMA 的同步技术，并在 VL 时刻调度表中每一个 BC 的开始时刻使用时钟同步数据帧（SYNC）进行时钟同步，以实现 TTAFDX 网络的时钟同步性。其中 SYNC 帧采用 SAE AS6802 协议中定义的 PCF 格式。

在每个 BC 中，TT 流量数据帧优先安排调度，且 SYNC 帧按照 TT 流量进行处理。BC 中安排用于发送 TT 流量数据的时间资源根据调度任务划分为多个时窗（time window），时窗大小由该时段所发送 TT 数据的帧长决定，并非固定值。为了简化 MC 内的 TT 数据帧调度时刻，对 MC 中在同一列上的时窗大小统一化，其值等于各 BC 在此时窗中传输消息的最大帧长。每个 BC 的剩余时间中安排 RC 流量数据发送任务，由于 RC 流量数据传输具有一定的不确定性，为避免对下一 BC 内 TT 帧的发送造成延迟，可将当前 RC 流量数据的最大帧长传输时间作为保护间隔，预留在用于发送 RC 流量数据的时间段最后。VL 时刻调度表如图 3-2 所示。

即使采用了本文所提出的时钟同步算法，网络中的各个节点仍旧可能存在时钟漂移的现象。通过为每个时窗预留一定的余量，能够对时钟误差进行补偿，使相邻时窗在时域内完全正交，避免由于本地时钟漂移相互重叠而引发的干扰，从而保证系统的可靠性。

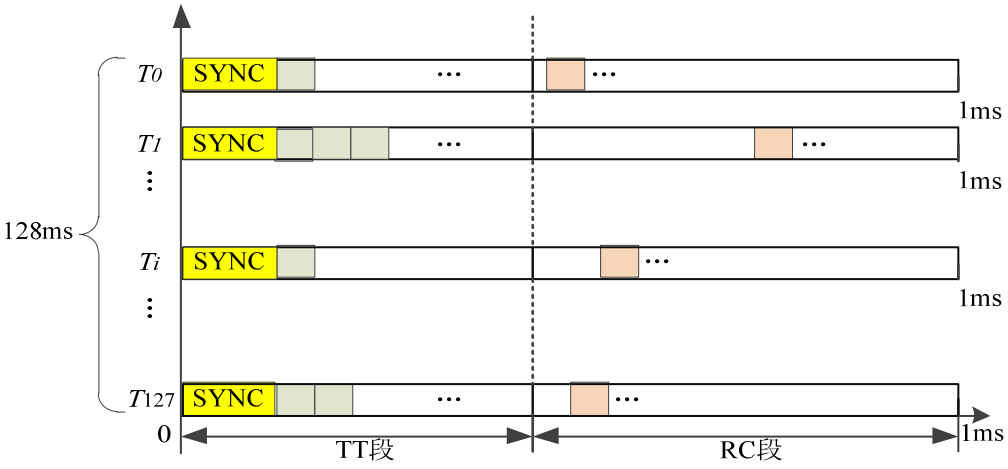

图 3-2 VL 时刻调度表示意图

3.1.1 周期优先的调度表设计

在网络带宽为 C 的 TTAFDX 中，TTAFDX 端系统对其所承载的 N 条 TTVL 数据流执行如下调度表设计方法进行发送时刻规划。

1）按照 BAG 由小到大的顺序对 N 条 TTVL 排序，当多条 TTVL 具有相同的 BAG 时，按照 Lmax 长度由大到小的顺序对具有相同 BAG 的多条 TTVL 进一步排序，记排序后的 TTVL 为 $VL_k, k \in \{1,...,N\}$，且记各 VL_k 的传输的最大帧长和 BAG 分别为 L_k, G_k, $k \in \{1,...,N\}$，转至第 2 步。

2）按照 VL_k 中标记 k 由小到大的顺序规划各条时间触发虚链路的调度时刻，记 BC 中已配置的帧长和为 L_{sum}，由于每个 BC 均始于 SYNC 帧，则初始时刻可设 $L_{sum} = L_{SYNC}$，转至第 3 步。

3）若 $k > N$，则完成了该端系统中所有 TTVL 发送时刻的规划；否则，从调度表完成 SYNC 帧发送后，即 L_{SYNC}/C 开始，在 G_k 时间范围中，使用“左紧凑”原则对 VL_k 的发送时刻进行规划，即尽量在靠近 MC 左侧的时窗列中分配可用的时窗。当所分配的时窗列为空时，则 $L_{sum} = L_{sum} + L_k$。若更新后的 L_{sum} 大于 BC 中能够发送的最大帧长和，则转至第 4 步；否则，将 VL_k 在一个 MC 内的第 m 个调度时刻（单位 ms）配置为 $t_{k,m} = (m-1) \times G_k + 8 \times (L_{sum} - L_k) \times 1000 / (C \times 10^6), m \in \{1,...,128/G_k\}$，同时将 L_k 赋予当前所在时窗列长度 L_{TW}。当所分配的时窗列不为空时，比较 L_k 与 L_{TW}。若 $L_{TW} \geq L_k$，则 L_{TW} 保持不变；否则更新 $L_{TW} = L_k$，$L_{sum} = L_{sum} + L_k - L_{TW}$。若 L_{sum} 大于 BC 中能够发送的最大帧长和，则跳到第 4 步；否则，记首个调度时窗属于第 a 个 BC，将 VL_k 的第 m 个调度时刻配置为 $t_{k,m} = (a-1) + (m-1) \times G_k + 8 \times (L_{sum} - L_k) \times 1000 / (C \times 10^6),\ m \in \{1,...,128/G_k\}$。[103]

4）带宽有限，无法完成该端系统中所有 TTVL 发送时刻的规划。

周期优先调度表设计方法采用的“左紧凑”分配原则，确保了只有当一列时窗分配完全后，才会在新的一列时窗中配置发送任务，不仅简化了 TTVL 流量调度时刻的配置过程，同时也增加了 RCVL 流量可传输的时间。

以表 3-1 中所列时间触发虚链路配置表为例，通过此方法规划所得的时间调度表如图 3-3 所示。

表 3-1　时间触发虚链路配置表

虚链路 ID	L_{max}(Bytes)	BAG(ms)
VL1	500	2
VL2	150	8
VL3	300	16
VL4	100	2
VL5	200	8
VL6	800	4

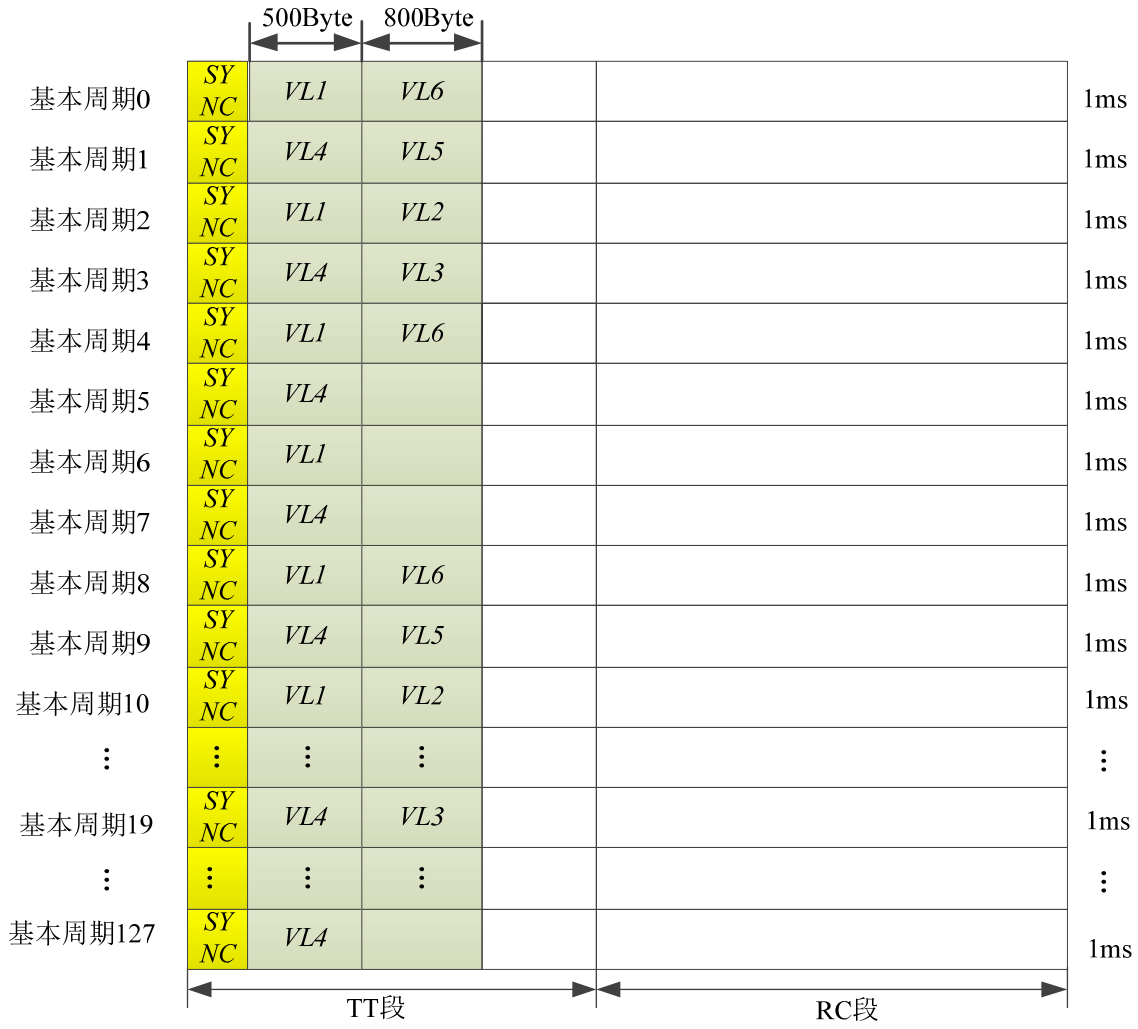

图 3-3 周期优先时间调度表

3.1.2 帧长度优先的调度表设计

采用周期优先调度表设计方法时，MC 中的同列时窗具有相同的长度，当同列时窗各 BC 传输任务帧长度相差较大时，时窗利用率较低、带宽利用不足。为了解决可能产生的 TT 时窗利用率问题，本小节提出了帧长度优先调度表设计方法。该方法能够有效提高 BC 中 TT 时窗的利用率，从而在相同 TT 传输任务的情况下缩短 TT 时间段的长度，增加 RC 数据流的传输时间，在带宽有限的条件下保证了带宽的有效利用。具体设计流程如下：

1） 按照数据帧长度从大到小的顺序对 N 条 TTVL 排序，当多条 TTVL 具有相同的数据帧长度时，按照 BAG 由小到大的顺序对具有相同数据帧长度的多条 TTVL 进一步排序，记排序后的 TTVL 为 $VL_k, k \in \{1,...,N\}$，且记各 VL_k 的传输的最大帧长、BAG 以及首帧发送所在的 BC 周期分别为 L_k, G_k, S_k, $k \in \{1,...,N\}$，转至第 2 步。

2） 按照 VL_k 中标记 k 由小到大的顺序规划各条时间触发虚链路的调度时刻，记 BC 中已配置的帧长和为 L_{sum}，由于每个 BC 均始于 SYNC 帧，则初始时刻可设 $L_{sum} = L_{SYNC}$，

且令 $S_k=0$，转至第 3 步。

3）若 $k>N$，则完成了该端系统中所有 TTVL 发送时刻的规划；否则，从调度表完成 SYNC 帧发送后，即 L_{SYNC}/C 开始，在 G_k 时间范围中，对 VL_k 的发送时刻仍然按照“左紧凑”原则分配可用的发送时窗。记 VL_k 在 MC 中的首个数据帧所分配的调度时窗属于第 a 个 BC，则当首个数据帧所分配的时窗所属列为空，即 $a=0$ 时，则 $S_k=a$，$L_{sum}=L_{sum}+L_k$。若更新后的 L_{sum} 大于 BC 中能够发送的最大帧长和，则转至第 4 步；否则，将 VL_k 在一个 MC 内的第 m 个调度时刻（单位 ms）配置为 $t_{k,m}=(m-1)\times G_k+8\times(L_{sum}-L_k)\times 1000/(C\times 10^6),\ m\in\{1,...,128/G_k\}$，同时将 L_k 赋予当前所在时窗列长度 L_{TW}。若所分配的时窗列不为空时，则需要通过 $(a-S_i)\%(\min(G_k,G_i))?=0$ 判断是否与时窗列中已完成规划的 TTVL 存在调度冲突。当 $(a-S_i)\%(\min(G_k,G_i))=0$ 时，存在调度冲突，需要在所选时窗后重新开始第 3 步；若 $(a-S_i)\%(\min(G_k,G_i))\neq 0$，则 VL_k 在一个 MC 内的第 m 个调度时刻（单位 ms）配置为 $t_{k,m}=(a-1)+(m-1)\times G_k+8\times(L_{sum}-L_k)\times 1000/(C\times 10^6),m\in\{1,...,128/G_k\}$，并将 a 赋予 S_k [100]。

4）带宽有限，无法完成该端系统中所有 TTVL 发送时刻的规划。

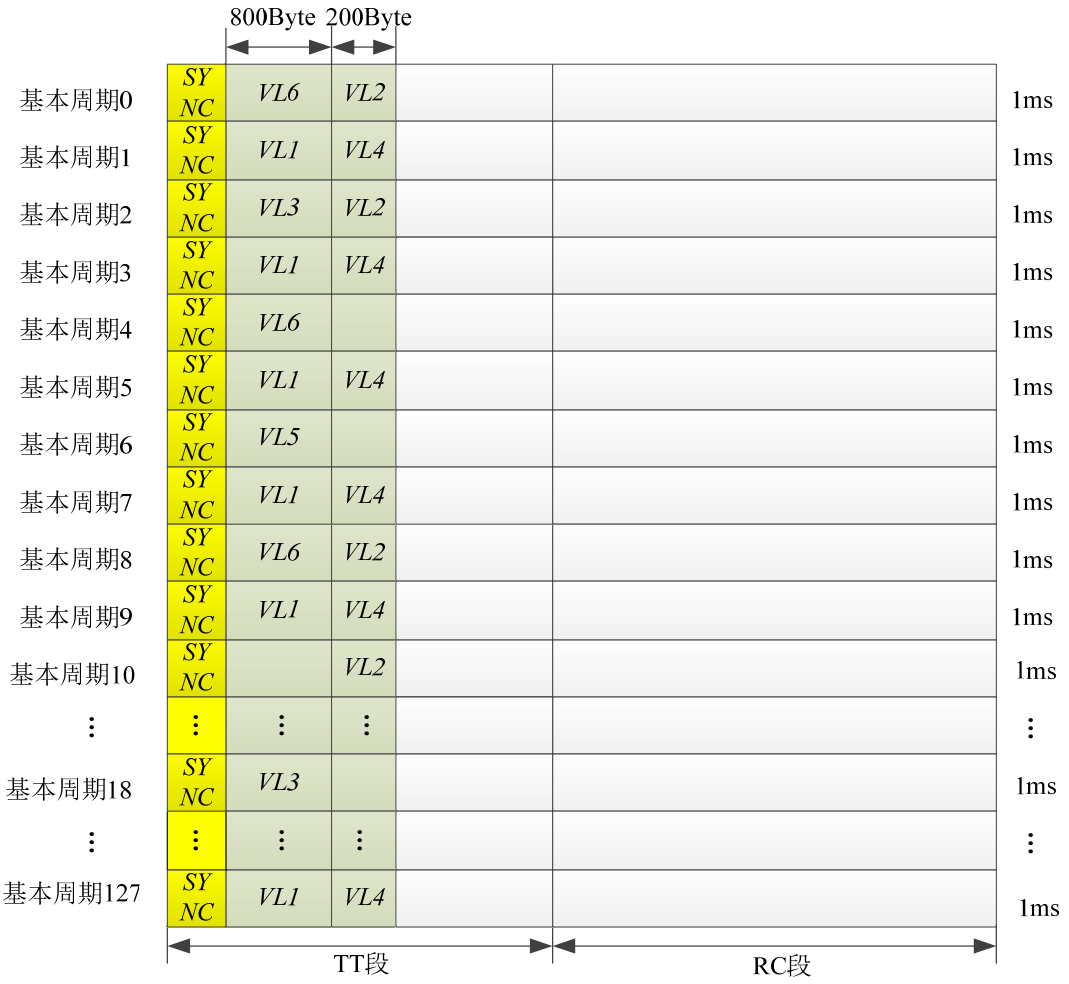

图 3-4 帧长度优先时间调度表

通过对表 3-1 所列的时间触发虚链路配置信息采用帧长度优先调度表设计方法，将得到规划后的时间调度表如图 3-4 所示。对比图 3-3 和图 3-4，能够发现帧长度优先调度表设计方法所得调度表在同样使用两列时窗的情况下，占用的 TT 时窗较周期优先设计方法减少了 300Bytes，有效地提高了 TT 时窗的利用率，并能够进一步增加 RC 流量的带宽，为 TTAFDX 网络的整体时间确定性提供更好的保障。此设计方法在选择适用时窗时需要额外的运算，增加了一定的运算复杂度，但运算只是简单地“取余”，对系统性能影响很小。

3.2 时间触发虚链路交换机调度算法设计

当虚链路数据帧到达 TTAFDX 交换机输入端口并完成完整性检查后，将根据数据帧类型分别进行处理。当数据帧属于 TTVL 时，交换机将首先检查 TT 数据帧的到达时间是否在所规定的接收时窗内。若不在，则丢弃该帧，剩余的接收到的数据帧则通过漏桶算法（Leaky Bucket）完成流量管制后送至交换机输出端口[123]。当数据帧属于 RCVL 时，同样需要对其进行流量管制，并在送至输出端口前按照调度器所预设的调度算法进行处理。TTAFDX 交换机输出端口按照所规划和配置好的时刻调度表优先转发 TTVL 数据帧，并在空闲时间段完成 RCVL 数据帧的转发。TTAFDX 交换机系统虚链路调度如图 3-5 所示。

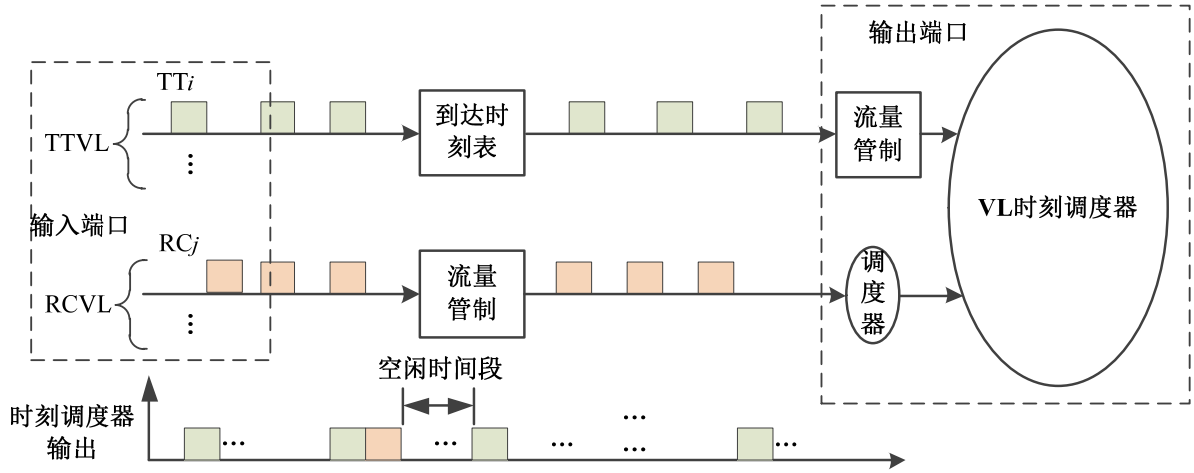

图 3-5 TTAFDX 交换机系统 VL 调度

在配置完成各端系统 TTVL 时刻调度表后，各 TTVL 数据帧将在特定的时刻到达转发路径中的各个交换机。交换机将根据所承载的不同 TTVL 转发任务及 TT 数据帧到达时间对每个输出端口配置转发时刻调度表。为了减少 TT 时窗及 TT 时窗间的空闲时间，提高带宽利用率，转发时刻调度表仍以 128ms 为周期，但不再按照基本周期进行时刻规划，具体算法如下：

1）当端系统采用周期优先调度表设计时，按照 BAG 由小到大的顺序对所转发的 N 条 TTVL 进行排序，当多条 TTVL 具有相同的 BAG 时，按照 Lmax 长度由大到小的顺

序对具有相同 BAG 的多条 TTVL 进一步排序，记排序后的 TTVL 为$VL_k, k \in \{1,...,N\}$；当端系统采用帧长度优先调度表设计时，按照数据帧长度从大到小的顺序对所转发的 N 条 TTVL 进行排序，当多条 TTVL 具有相同的数据帧长度时，进一步按照 BAG 由小到大的顺序对具有相同数据帧长度的多条 TTVL 排序，记排序后的 TTVL 为 $VL_k, k \in \{1,...,N\}$。记各 VL_k 的传输的最大帧长和 BAG 分别为 L_k, G_k, $k \in \{1,...,N\}$，转到至 2 步。

2） 按照 VL_k 中标记 k 由小到大的顺序规划各条时间触发虚链路在输出端口的转发时刻，转至第 3 步。

3）若 $k > N$，则完成了该端系统中所有 TTVL 发送时刻的规划；否则，以 $V_{k,i}\ (1 \le i \le 128/G_k)$ 依次标识 VL_k 在一个调度周期中将被转发的数据帧。按照 $V_{k,i}\ (1 \le i \le 128/G_k)$ 在源端系统中的发送顺序及 VL_k 数据帧转发路径的顺序，依次对各帧在各转发节点交换机输出端口的转发时刻进行规划。根据公式（3-1），当已知数据帧 $V_{k,i}\ (1 \le i \le 128/G_k)$ 在前一传输节点 t_1 时刻发送时，能够计算出该数据帧在此交换机的接收窗口关闭时刻 t_2。

$$t_2 = t_1 + \frac{8 \times L_k \times 1000}{C \times 10^6} + D_{switchtech} + 2 \times \Delta T \qquad (3\text{-}1)$$

公式中：

C—— 数据流在物理连输上的传输速率；

$D_{switchtech}$—— 交换机固有技术延时；

ΔT—— 系统所允许的最大时钟漂移。

从 t_2 时刻起，在已部分规划好转发时刻的调度表中查找能够完成转发任务的空闲时间段。当在 Δt 时间后找到了空闲时间段，其中 $0 \le \Delta t \le 128$，则转发时刻为 $t_3 = (t_2 + \Delta t)\%128$，更新交换机的转发时刻表；若不存在可利用的空闲时间段，则转至第 4 步。当 VL_k 在一个调度周期中的所有数据帧在转发路径中的各个交换机中都能完成转发时刻规划，则 $k++$，循环第 3 步；否则，转至第 4 步。

4）带宽有限，无法完成该交换机系统中所有 TTVL 的发送时刻规划。

3.3 TTAFDX 网络建模及时延分析

3.3.1 TTAFDX 网络模型

本文中的 TTAFDX 沿用标准 AFDX 的星型拓扑网络结构，如图 3-6 所示[103]。模型中包括 3 个交换机 SW1，SW2，SW3，8 个端系统 ES1，…，ES8 和 12 条虚链路 *VL1*，…，*VL12*，各个虚链路均为单播虚链路。其中 12 条虚链路的相关参数，包括最大帧长 *L*max，带宽分配间隔（BAG）以及虚链路类型由表 3-2 列出。

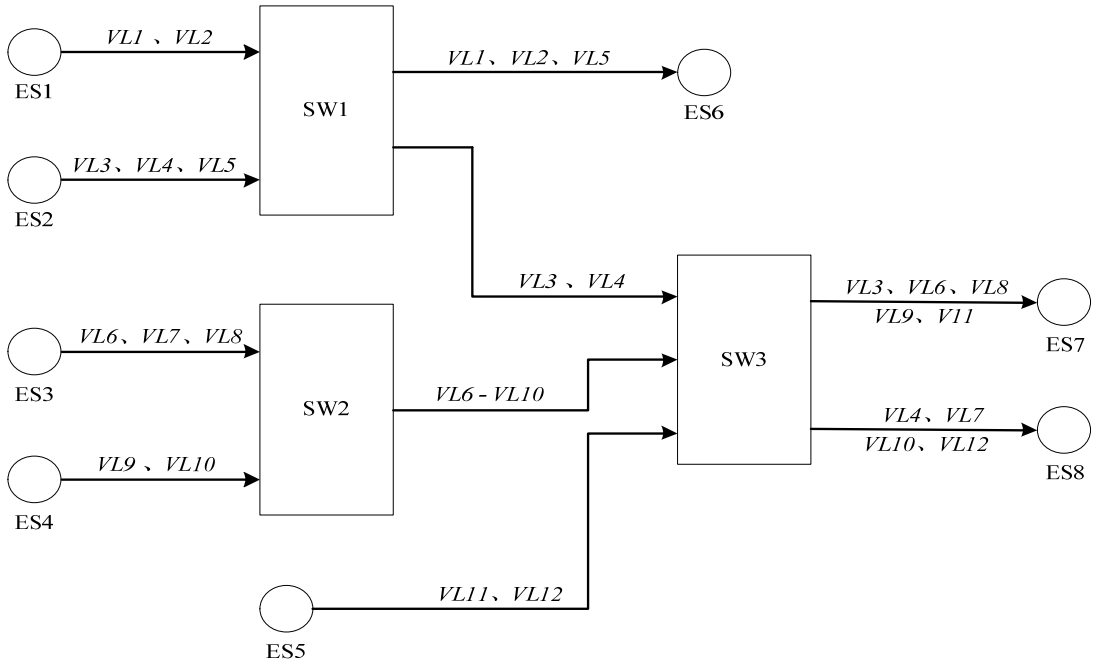

图 3-6 TTAFDX 网络模型

表 3-2 VL 配置表

虚链路 ID	*L* max (Bytes)	BAG(ms)	类型
*VL*1	512	16	时间触发
*VL*2	256	8	速率限制
*VL*3	128	32	时间触发
*VL*4	256	64	时间触发
*VL*5	1024	32	速率限制
*VL*6	512	32	时间触发
*VL*7	256	32	时间触发
*VL*8	512	64	时间触发
*VL*9	128	128	速率限制
*VL*10	128	4	速率限制
*VL*11	1024	16	时间触发
*VL*12	64	64	速率限制

TTAFDX 数据帧的网络时延，即由源端传输到目的端所耗费的时间，可定义如下：

$$D = D_{propagation} + D_{tx,\,k} + \sum_{i=0}^{j} D_{switch,\,k} + D_{rx,\,k} \tag{3-2}$$

公式（3-2）中，$\sum_{i=0}^{j} D_{switch,\,k}$ 表示虚链路 VL_k 经过 j 级交换机之后的所得交换机时延之

和。现将各时延部分的组成分别介绍如下：

1）链路传输时延 $D_{propagation}$：是数据帧在物理链路上传输所引起的时延。由于链路中数据传输的速度由介质特性决定，则链路传输时延仅由链路距离决定。假设图 3-6 所示 TTAFDX 网络结构中相邻节点间的距离为 100 米，所使用物理链路的传输速率为 $2\times10^8 m/s$，则其链路传输时延为固定值 $0.5\mu s$。

2）源端系统时延 $D_{tx,k}$：数据在发送端的时延可用公式 3-3 定义：

$$D_{tx,k} = G_k + \tau_{tech} + \tau_{frame,k} + EJ_{\max,k} \quad (3\text{-}3)$$

式中：

G_k —— 虚链路 VL_k 的带宽分配间隔；

τ_{tech} —— 源端系统的技术时延；

$\tau_{frame,k}$ —— 数据帧时延；

$EJ_{\max,k}$ —— 源端系统多路复用时延。

τ_{tech} 是发送端在处理数据时造成的操作时延，主要依赖设备软硬件的性能[124]，具有可预测性。主要包括硬件接口处理时延、网络协议包构成时延等。ARINC664 协议规定端系统的 τ_{tech} 应低于 $150\mu s$；

$\tau_{frame,k}$ 是虚链路 VL_k 将数据帧发送至物理链路层所造成的延迟，假设物理带宽为 C，数据帧的最大长度为 $S_{\max}$，则帧延迟上界为 $\tau_{frame,k} = S_{\max}/C$。

$EJ_{\max,k}$ 由复用调度策略导致的数据队列等待时延，主要由源端节点中待发送的数据量和此刻网络流量决定，是 TTAFDX 延时分析研究的重点，加入时间触发调度机制之后，这一部分的时延对于不同流量类型将发生相应改变。

3）交换机时延 $D_{rx,k}$：包含了数据帧在交换机中所经历的过滤、转发和接收三个步骤，计算方法如下：

$$D_{switch,i} = \tau_{filter} + \tau_{fw} + \tau_{rv} + SJ_{\max,i} \quad (3\text{-}4)$$

式中：

τ_{filter} —— 数据帧过滤时延；

τ_{fw} —— 数据帧转发时延；

τ_{rv} —— 数据帧接收时延；

$SJ_{\max,i}$ —— 交换机系统多路复用时延。

依据标准 AFDX 协议的规定，TTAFDX 中每个交换机端口必须能以线速对到达的任意大小的有效数据帧及时进行过滤。线速是指交换机处理到达的数据包的最大速度，由协议可知最小数据包的长度为 84 字节，当物理线路的传输速度为 100Mb/s 时，大概 $84\times8/100 = 6.72\mu s$ 就有一个数据包到达，因此，数据帧过滤 τ_{filter} 的最大延时为 $6.72\mu s$，通常设定为 $8\mu s$；

τ_{fw} 对应于数据过滤的过程，TTAFDX 规定了交换机在 1ms 内要求转发的数据帧数量，一般为交换机的端口数×125，则数据转发过程中最大的延迟可以认为是 $8\mu s$；

τ_{rv} 是在接收端口由数据帧接收过程产生的时延，其界限由各 VL 的最大帧长决定，为 $S_{\max}/C$；

$SJ_{\max,i}$ 是可变延时，主要指数据帧在缓冲队列中的排队等待延时。这部分的延时和具体的调度算法有关，是数据帧经过交换机产生的延时的决定部分。

4）目的端系统时延 $D_{rx,k}$：主要为技术延时，ARINC664 协议中规定其不得大于 $150\mu s$。

综上所述，TTAFDX 网络的端到端时延可分为两部分：固定时延和非固定时延，具体如图 3-7 所示。

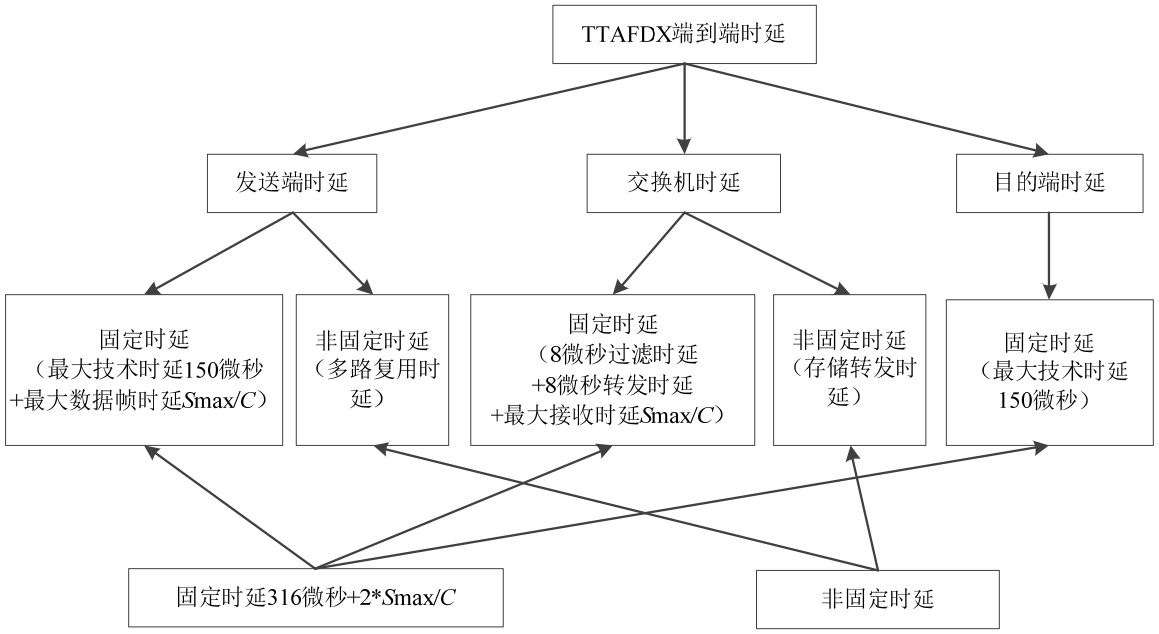

图 3-7 TTAFDX 网络端到端时延分析

如图 3-7 所示，TTAFDX 网络的端到端延时上界主要包括了 $316+2\times(S_{\max}/C)$ 微秒的固定延时部分和非固定延时部分，而非固定延时主要由多路复用排队造成。多路复用排队包括源端系统输出时虚链路的聚合，以及经过各交换节点时的多路复用排队[125]。在以下实时性能分析中，为了更加直观地体现调度算法对时延的影响，发送端系统中的技术时延将不参与计算。

3.3.2 确定性网络演算理论基础

上世纪九十年代初，R. L. Cruz 在研究网络延时时提出了网络元素的概念，同时提出了对输入流进行整流的思路，并定义了诸如整形器、服务曲线和到达曲线等概念。伴随着最小加代数（Min-Plus Algebra）理论的引入，网络演算理论逐渐发展和成熟起来。目前已成功应用于工业以太网的缓冲区调度和时间延迟分析应用中，特别是很好地解释

了“Pay Burst Only Once”现象，从而得到了广泛应用。

最小加代数是以离散时间为模型所建立起来的一个数学体系，其主要函数及运算性质如下：

定义 3.1 $\forall s, t \in (R \cup \infty), s \le t$，有 $f(s) \le f(t)$，则称 f 为广义增函数。

定义 3.2 函数 f 在最小加代数 $(R \cup \{\infty\}, \wedge, +)$ 中的积分运算定义为：$\inf_{s \in R,\ 0 \le s \le t} \{f(s)\}$。

定义 3.3 若 $f, g \in F$，其中 F 表示 $t<0, f(t)=0$ 的广义增函数。函数 f, g 的卷积运算为：

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\} \tag{3-5}$$

定义 3.4 若 $f, g \in F$，则函数 f, g 的反卷积运算为：$(f \oslash g)(t) = \sup_{0 \le u} \{f(t+u) - g(u)\}$。

定义 3.5 若 $f, g \in F$，则函数 f, g 之间的最大垂直距离 $v(f, g)$ 和最大水平距离 $h(f, g)$ 分别定义如下：

$$v(f, g) = \sup_{t \ge 0} \{f(t) - g(t)\} \tag{3-6}$$

$$h(f, g) = \sup_{t \ge 0} \{\inf\{d \ge 0 \ and \ f(t) \le g(t+d)\}\} \tag{3-7}$$

网络演算应用上述最小加代数理论对网络进行分析，其中的基本概念及重要结论如下：

定义 3.6 累计函数 $R(t)$ 定义为一个数据流在时间间隔 $[0, t]$ 中所发送的数据位的总和，同时满足 $t \le 0, R(t) = 0$。

定义 3.7 在时刻 t，一个无损耗系统 S 中的积压数据量定义为 $R(t) - R^*(t)$，即输入输出数据量的差。其中 $R(t)$ 表示无损耗系统 S 的输入端累计函数，$R^*(t)$ 表示无损耗系统 S 的输出端累计函数。

定义 3.8 在时刻 t，无损系统 S 中数据流的延时为 $d(t) = \inf\{\tau \ge 0 : R(t) \le R^*(t+\tau)\}$。

定义 3.9 对于一个广义增函数 $\alpha(t), t \ge 0$，如果数据流的累计函数 $R(t)$ 对时刻 s, t 满足：$R(t) - R(s) \le \alpha(t-s)$，则称 $\alpha(t)$ 是 $R(t)$ 的到达曲线。

定义 3.10 当输入累计函数为 $R(t)$ 的一个数据流经过无损耗系统 S 时，称系统 S 提供了一个服务曲线 $\beta(t)$，当且仅当 $\beta(t)$ 具有广义增性质，且满足 $\beta(0) = 0$ 时，$R^* \ge R \otimes \beta$。

定理 3.1 假定一个到达曲线为 $\alpha(t)$ 的数据流经过系统 S 时，系统 S 提供的服务曲线为 $\beta(t)$，则对于任意一个时间 t，系统 S 中的积压数据量满足：

$$R(t) - R^*(t) \le \sup_{s \ge 0} \{\alpha(s) - \beta(s)\} \tag{3-8}$$

定理 3.2 假定一个到达曲线为 $\alpha(t)$ 的数据流经过系统 S 时，系统 S 提供的服务曲线为 $\beta(t)$，则对于任意一个时间 t，延时 $d(t)$ 满足：

$$d(t) \le h(\alpha, \beta) \tag{3-9}$$

定理 3.3 假定一个到达曲线为$\alpha(t)$的数据流经过系统 S 时，系统 S 提供的服务曲线为$\beta(t)$，则对于任意一个时间t，输出端的数据流满足约束条件：

$$\alpha^{*}=\alpha \oslash \beta \tag{3-10}$$

定理 3.4 假定一个到达曲线为$\alpha(t)$的数据流依次流经系统 S_1 和 S_2 时，系统 S_1 和 S_2 提供的服务曲线分别为$\beta_1(t)$和$\beta_2(t)$，则串联系统 S_1 和 S_2 后系统提供的服务曲线为[126]：

$$\beta(t)=\beta_1(t)\otimes\beta_2(t) \tag{3-11}$$

应用以上的网络演算理论，在已知数据流的到达曲线和网络元素服务曲线的情况下，可以方便计算数据流通过此网络元素过程中的延迟、积压等性能参数上确界。当到达曲线为$\alpha(t)=rt+b$，网络服务节点的服务曲线为$\beta(t)=C[t-d]^{+}$时，最大延迟为$\mathrm{Delay}=d+b/C$，最大积压为$\mathrm{Backlog}=r*d+b$，如图 3-8 所示：。

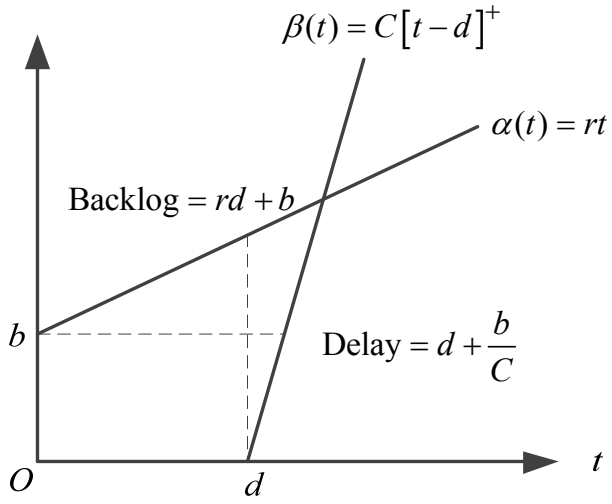

图 3-8 经过网络元素的延迟和积压

3.4 AFDX 虚链路实时性分析

ARINC664 协议中默认使用先来先服务（First In First Out，FIFO）调度机制，无需进行重新分组。FIFO 调度算法是一种非抢夺式的调度算法，任意数据流的数据帧只有在已经到达缓冲区的数据帧转发完毕后，才能接受调度服务，无法对突发紧急数据流提供立刻服务，从而造成严重后果。FIFO 机制保证了所有链路消息的公平性，但同时会引起在交换机中竞争链路产生拥塞，导致不确定的时延。ARINC664 协议也支持静态优先级调度算法（Static Priority，SP），以降低紧急数据的最大延迟抖动，提高 AFDX 数据传输的实时性。本小节通过网络演算方法，分别对采用 FIFO 和 SP 调度算法的传统 AFDX 网络时间确定性和实时性进行了分析。

1）FIFO 调度算法实时性分析

当 n 条虚链路共享同一交换机端口时，记各$VLi(1\le i\le n)$的带宽分配间隔和最大数

据帧长度分别为 BAG_i 和 $L_{\max}^i$，且由于共享物理带宽 C 而满足 $\sum_{i=1}^{n}\frac{L_{\max}^i}{BAG_i}\le C$，如图 3-9 所示。此时，各虚链路的到达曲线可表示为：

$$\alpha_i(t)=L_{\max}^i+\frac{L_{\max}^i}{BAG_i}t \tag{3-12}$$

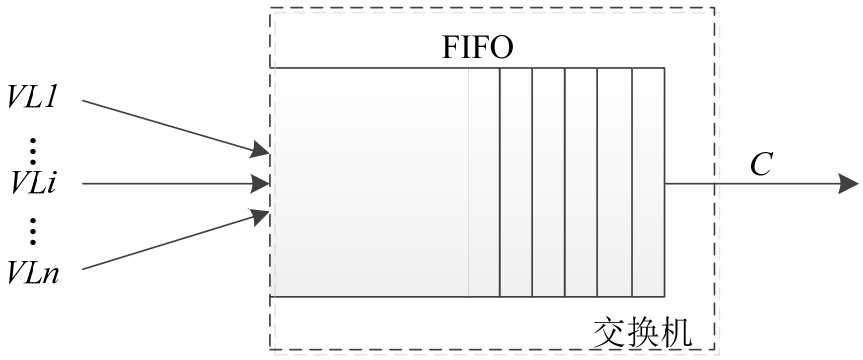

图 3-9 带缓冲区的 FIFO 示意图

由于到达曲线具有累加性，可得输出端口需要处理的所有虚链路聚合到达曲线为:

$$\alpha(t)=\sum_{i=1}^{n}\alpha_i(t)=\sum_{i=1}^{n}L_{\max}^i+t\sum_{i}^{n}\frac{L_{\max}^i}{BAG_i} \tag{3-13}$$

由于输出端口物理带宽为 C，且各虚链路数据帧按照到达顺序依次存储在 FIFO 缓冲区中，则其提供给所有虚链路的总服务曲线为[103]：

$$\beta(t)=C[t-0]^+ \tag{3-14}$$

由 FIFO 最小服务曲线推论[127]，可得交换机为第 i 个虚链路提供的服务曲线为：

$$\beta_i(t)=[\beta(t)-(\alpha(t-\theta)-\alpha_i(t-\theta))]^+ \tag{3-15}$$

公式（3-15）右边是聚合虚链路中除第 i 条虚链路以外的其它数据流在 0 时刻的积压数据[103]，由 $\beta(\theta)=\alpha(0)-L_{\max}^i$ 可得 $\theta=\frac{\sum_{j=1}^{n}L_{\max}^j-L_{\max}^i}{C}$。由聚合到达曲线定义可得：

$$\alpha(t)-\alpha_i(t)=\sum_{j=1}^{n}\alpha_j(t)-\alpha_i(t)=(\sum_{j=1}^{n}\frac{L_{\max}^j}{BAG_i}-\frac{L_{\max}^i}{BAG_i})t+(\sum_{j=1}^{n}L_{\max}^j-L_{\max}^i) \tag{3-16}$$

将公式（3-16）带入公式（3-15）可得：

$$\begin{aligned}\beta_i(t)&=\left[Ct-\left(\sum_{i=1}^{n}\frac{L_{\max}^{i}}{BAG_i}-\frac{L_{\max}^{i}}{BAG_i}\right)\left(t-\frac{\sum_{i=1}^{n}L_{\max}^{i}-L_{\max}^{i}}{C}\right)-\left(\sum_{i=1}^{n}L_{\max}^{i}-L_{\max}^{i}\right)\right]^{+}\\&=\left[C-\left(\sum_{i=1}^{n}\frac{L_{\max}^{i}}{BAG_i}-\frac{L_{\max}^{i}}{BAG_i}\right)\right]\left(t-\frac{\sum_{i=1}^{n}L_{\max}^{i}-L_{\max}^{i}}{C}\right)\end{aligned}\tag{3-17}$$

由公式（3-17）可得交换机提供给第 *i* 条虚链路的服务速率和服务时延分别为：

$$R_i(t)=C-\left(\sum_{i=1}^{n}\frac{L_{\max}^{i}}{BAG_i}-\frac{L_{\max}^{i}}{BAG_i}\right)\tag{3-18}$$

$$d_i=\frac{\sum_{i=1}^{n}L_{\max}^{i}-L_{\max}^{i}}{C}\tag{3-19}$$

由定理 3.3 可求得第 *i* 条虚链路数据流在此交换机的离开曲线为：

$$\alpha_i^{*}=\alpha_i(t+d_i)\tag{3-20}$$

在级联网络系统中，所得离开曲线即为下一级交换机的到达曲线。

由图 3-6 所示的 AFDX 网络模型图可知，*VL1*、*VL2* 和 *VL5* 都是通过交换机 SW1 到达目标端系统 ES6，则其在 SW1 输出端口的聚合到达曲线 $\alpha_{1,2,5}(t)$ 为：

$$\alpha_{1,2,5}(t)=\sum_{i=1,2,5}\alpha_i(t)=\sum_{i=1,2,5}L_{\max}^{i}+t\sum_{t=1,2,5}\frac{L_{\max}^{i}}{BAG_i}\tag{3-21}$$

VL3 和 *VL4* 通过交换机 SW1 转发至交换机 SW3，则其在 SW1 输出端口的聚合到达曲线 $\alpha_{3,4}(t)$ 为：

$$\alpha_{3,4}(t)=\sum_{i=3,4}\alpha_i(t)=\sum_{i=3,4}L_{\max}^{i}+t\sum_{t=3,4}\frac{L_{\max}^{i}}{BAG_i}\tag{3-22}$$

VL6-VL10 均通过交换机 SW2 转发至交换机 SW3，则其在 SW2 输出端口的聚合到达曲线 $\alpha_{6-10}(t)$ 为：

$$\alpha_{6-10}(t)=\sum_{i=6}^{10}\alpha_i(t)=\sum_{i=6}^{10}L_{\max}^{i}+t\sum_{i=6}^{10}\frac{L_{\max}^{i}}{BAG_i}\tag{3-23}$$

SW1 为 *VL1*、*VL2* 和 *VL5* 提供向 ES6 转发服务的聚合流服务曲线为 $\beta_{SW1-ES6}(t)=Ct$。

由公式（3-17）可得 *VL1*、*VL2* 和 *VL5* 所获得的服务曲线分别为：

$$\beta_{SW1,i}(t)=\left[Ct-\left(\sum_{i=1,2,5}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\left(t-\frac{\sum_{i=1,2,5}L^i_{\max}-L^i_{\max}}{C}\right)-\left(\sum_{i=1,2,5}L^i_{\max}-L^i_{\max}\right)\right]^+$$
$$=\left[C-\left(\sum_{i=1,2,5}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{\sum_{i=1,2,5}L^i_{\max}-L^i_{\max}}{C}\right) \quad (3\text{-}24)$$

SW1 为 *VL3* 和 *VL4* 提供向 SW3 转发服务的聚合流服务曲线为 $\beta_{SW1-SW3}(t)=Ct$，可得 *VL3* 和 *VL4* 所获得的服务曲线为：

$$\beta_{SW1,i}(t)=\left[Ct-\left(\sum_{i=3,4}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\left(t-\frac{\sum_{i=3,4}L^i_{\max}-L^i_{\max}}{C}\right)-\left(\sum_{i=3,4}L^i_{\max}-L^i_{\max}\right)\right]^+$$
$$=\left[C-\left(\sum_{i=3,4}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{\sum_{i=3,4}L^i_{\max}-L^i_{\max}}{C}\right) \quad (3\text{-}25)$$

依据公式（3-20）分别计算其对于 SW3 的到达曲线：

$$\alpha_i^*=\alpha_i(t+\frac{\sum_{i=3,4}L^i_{\max}-L^i_{\max}}{C}) \quad (3\text{-}26)$$

同理可得 SW2 为 *VL6-VL10* 提供的服务曲线为：

$$\beta_{SW2,i}(t)=\left[Ct-\left(\sum_{i=6}^{10}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\left(t-\frac{\sum_{i=6}^{10}L^i_{\max}-L^i_{\max}}{C}\right)-\left(\sum_{i=6}^{10}L^i_{\max}-L^i_{\max}\right)\right]^+$$
$$=\left[C-\left(\sum_{i=6}^{10}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{\sum_{i=6}^{10}L^i_{\max}-L^i_{\max}}{C}\right) \quad (3\text{-}27)$$

同理，*VL6-VL10* 离开 SW2 后对于 SW3 的到达曲线为：

$$\alpha_i^*=\alpha_i(t+\frac{\sum_{i=6}^{10}L^i_{\max}-L^i_{\max}}{C}) \quad (3\text{-}28)$$

在 SW3 中，*VL3*、*VL6*、*VL8*、*VL9* 和 *VL11* 转发至 ES7，则其在 SW3 输出端口的聚合到达曲线 $\alpha_{SW3-ES7}(t)$ 为：

$$\alpha_{SW3-ES7}(t)=\sum_{i=3,6,8,9}\alpha_i^*(t)+\alpha_{11}(t)=t\sum_{i=3,6,8,9,11}\frac{L^i_{\max}}{BAG_i}+L^* \quad (3\text{-}29)$$

$$L^{*}=\sum_{i=3,6,8,9,11}L^{i}_{\max}+\frac{L^{4}_{\max}}{C}\frac{L^{3}_{\max}}{BAG_3}+\frac{\sum_{i=7,8,9,10}L^{i}_{\max}}{C}\frac{L^{6}_{\max}}{BAG_6}+\frac{\sum_{i=6,7,9,10}L^{i}_{\max}}{C}\frac{L^{8}_{\max}}{BAG_8}+\frac{\sum_{i=6,7,8,10}L^{i}_{\max}}{C}\frac{L^{9}_{\max}}{BAG_9}。$$

而 *VL4*、*VL7*、*VL10* 和 *VL12* 转发至 ES8，其在 SW3 输出端口的聚合到达曲线 $\alpha_{SW3-ES8}(t)$ 为：

$$\alpha_{SW3-ES8}(t)=\sum_{i=4,7,10}\alpha^{*}_{i}(t)+\alpha_{12}(t)=t\sum_{i=4,7,10,12}\frac{L^{i}_{\max}}{BAG_i}+L^{**} \tag{3-30}$$

其中，$L^{**}=\sum_{i=4,7,10,12}L^{i}_{\max}+\frac{L^{3}_{\max}}{C}\frac{L^{4}_{\max}}{BAG_4}+\frac{\sum_{i=6,8,9,10}L^{i}_{\max}}{C}\frac{L^{7}_{\max}}{BAG_7}+\frac{\sum_{i=6,7,8,9}L^{i}_{\max}}{C}\frac{L^{10}_{\max}}{BAG_{10}}$。

结合公式（3-17）和以上四个公式，可得 *VL3*、*VL6*、*VL8*、*VL9* 和 *VL11* 所获得的服务曲线分别为：

$$\begin{aligned}\beta_{SW3-ES7,i}(t)&=\left[Ct-\left(\sum_{i=3,6,8,9,11}\frac{L^{i}_{\max}}{BAG_i}-\frac{L^{i}_{\max}}{BAG_i}\right)\left(t-\frac{L^{*}-{L^{i}_{\max}}^{*}}{C}\right)-\left(L^{*}-{L^{i}_{\max}}^{*}\right)\right]^{+}\\&=\left[C-\left(\sum_{i=3,6,8,9,11}\frac{L^{i}_{\max}}{BAG_i}-\frac{L^{i}_{\max}}{BAG_i}\right)\right]\left(t-\frac{L^{*}-{L^{i}_{\max}}^{*}}{C}\right)\end{aligned} \tag{3-31}$$

其中，${L^{11}_{\max}}^{*}=L^{11}_{\max}$，

$${L^{3}_{\max}}^{*}=L^{3}_{\max}+\frac{L^{4}_{\max}}{C}\times\frac{L^{3}_{\max}}{BAG_3} \tag{3-32}$$

$${L^{6}_{\max}}^{*}=L^{6}_{\max}+\frac{\sum_{i=7,8,9,10}L^{i}_{\max}}{C}\times\frac{L^{6}_{\max}}{BAG_6} \tag{3-33}$$

$${L^{8}_{\max}}^{*}=L^{8}_{\max}+\frac{\sum_{i=6,7,9,10}L^{i}_{\max}}{C}\times\frac{L^{8}_{\max}}{BAG_8} \tag{3-34}$$

$${L^{9}_{\max}}^{*}=L^{9}_{\max}+\frac{\sum_{i=6,7,8,10}L^{i}_{\max}}{C}\times\frac{L^{9}_{\max}}{BAG_9} \tag{3-35}$$

同理，可得 *VL4*、*VL7*、*VL10* 和 *VL12* 的服务曲线为：

$$\begin{aligned}\beta_{SW3-ES8,i}(t)&=\left[Ct-\left(\sum_{i=4,7,10,12}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\left(t-\frac{L^{**}-L^{i}_{\max}{}^{*}}{C}\right)-\left(L^{**}-L^{i}_{\max}{}^{*}\right)\right]^{+}\\&=\left[C-\left(\sum_{i=4,7,10,12}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{L^{**}-L^{i}_{\max}{}^{*}}{C}\right)\end{aligned}\tag{3-36}$$

其中，$L^{12}_{\max}{}^{*}=L^{12}_{\max}$，

$$L^{4}_{\max}{}^{*}=L^{4}_{\max}+\frac{L^{3}_{\max}}{C}\times\frac{L^{4}_{\max}}{BAG_4}\tag{3-37}$$

$$L^{7}_{\max}{}^{*}=L^{7}_{\max}+\frac{\sum_{i=6,8,9,10}L^i_{\max}}{C}\times\frac{L^{7}_{\max}}{BAG_7}\tag{3-38}$$

$$L^{10}_{\max}{}^{*}=L^{10}_{\max}+\frac{\sum_{i=6,7,8,9}L^i_{\max}}{C}\times\frac{L^{10}_{\max}}{BAG_{10}}\tag{3-39}$$

由于物理链路的延迟固定，可设其服务曲线为$\beta_\tau(t)$，其表示形式类似一个脉冲函数。

$$\beta_\tau(t)=\delta_\tau=\begin{cases}0,\ 0\le t\le\tau\\+\infty,\ t>\tau\end{cases}\tag{3-40}$$

交换机上过滤和调度等操作的技术时延固定为$\eta=16\mu s$，可以设其服务曲线为$\beta_\eta(t)$。

$$\beta_\eta(t)=\delta_\eta=\begin{cases}0,\ 0\le t\le\eta\\+\infty,\ t>\eta\end{cases}\tag{3-41}$$

交换机上每条 VL 的数据帧接收时延为固定的$L^i_{\max}/C$，可以设其服务曲线为$\beta_{rv}(t)$。

$$\beta_{rv}(t)=\delta_{rv}=\begin{cases}0,\ 0\le t\le L^i_{\max}/C\\+\infty,\ t>L^i_{\max}/C\end{cases}\tag{3-42}$$

假设所有 12 条虚链路均为传统 AFDX 虚链路，且$\beta_\tau(t)=0.5\mu s$，则根据节点串联服务曲线定理 3.2 计算 3.4.1 节所示网络模型中各条虚链路的时延如下：

虚链路 *VL1*、*VL2* 和 *VL5* 的串联服务曲线为：

$$\begin{aligned}\beta^{\Sigma}_i(t)&=\delta_\tau\otimes\beta_{SW1,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\left[C-\left(\sum_{i=1,2,5}\frac{L^i_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\times\left[t-\left(\frac{\sum_{i=1,2,5}L^i_{\max}-L^i_{\max}}{C}+2\times\tau+\eta+L^i_{\max}/C\right)\right]^{+}\end{aligned}\tag{3-43}$$

虚链路 *VL3* 的串联服务曲线为：

$$\beta_3^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW1,3}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES7,3}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\min\{\beta_{SW1,3}(t),\beta_{SW3-ES7,3}(t)\}\times\left[t-\left(\frac{L_{\max}^4+L^*-L_{\max}^{3\ *}}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^3}{C}\right)\right)\right]^+ \quad (3\text{-}44)$$

虚链路 *VL4* 的串联服务曲线为：

$$\beta_4^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW1,4}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES8,4}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\min\{\beta_{SW1,4}(t),\beta_{SW3-ES8,4}(t)\}\times\left[t-\left(\frac{L_{\max}^3+L^{**}-L_{\max}^{4\ *}}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^4}{C}\right)\right)\right]^+ \quad (3\text{-}45)$$

虚链路 *VL6*，*VL8* 和 *VL9* 的串联服务曲线为：

$$\beta_i^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW2,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES7,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\min\{\beta_{SW2,i}(t),\beta_{SW3-ES7,i}(t)\}$$
$$\times\left[t-\left(\frac{\sum_{i=6}^{10}L_{\max}^i-L_{\max}^i+L^*-L_{\max}^{i\ *}}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^i}{C}\right)\right)\right]^+ \quad (3\text{-}46)$$

虚链路 *VL7* 和 *VL10* 的串联服务曲线为：

$$\beta_i^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW2,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES8,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\min\{\beta_{SW2,i}(t),\beta_{SW3-ES8,i}(t)\}$$
$$\times\left[t-\left(\frac{\sum_{i=6}^{10}L_{\max}^i-L_{\max}^i+L^{**}-L_{\max}^{i\ *}}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^i}{C}\right)\right)\right]^+ \quad (3\text{-}47)$$

虚链路 *VL11* 的串联服务曲线为：

$$\beta_{11}^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW3-ES7,11}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\left[C-\left(\sum_{i=3,6,8,9,11}\frac{L_{\max}^i}{BAG_i}-\frac{L_{\max}^{11}}{BAG_i}\right)\right]\times\left[t-\left(\frac{L^*-L_{\max}^{11\ *}}{C}+2\times\tau+\eta+L_{\max}^{11}/C\right)\right]^+ \quad (3\text{-}48)$$

虚链路 *VL12* 的串联服务曲线为：

$$\beta_{12}^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW3-ES8,12}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\left[C-\left(\sum_{i=4,7,10,12}\frac{L_{\max}^i}{BAG_i}-\frac{L_{\max}^i}{BAG_i}\right)\right]\times\left[t-\left(\frac{L^{**}-L_{\max}^{12\ *}}{C}+2\times\tau+\eta+L_{\max}^{12}/C\right)\right]^+ \quad (3\text{-}49)$$

由以上推导可知，对于由 n 级交换机组成的虚链路 VLi，其服务曲线形如

$\beta_i^{\Sigma}(t)=R_i^{\Sigma}\times\left[t-T_i^{\Sigma}\right]^{+}$，在不考虑端系统多路复用的情况下，端到端的延迟上界 D_i 为：

$$D_i=T_i^{\Sigma}+\frac{L_{\max}^{i}}{R_i^{\Sigma}}\times n+\frac{L_{\max}^{i}}{C} \tag{3-50}$$

根据虚链路配置表 3-2，在无时间触发虚链路并采用 FIFO 调度算法时，各条虚链路的延迟上界计算结果如表 3-3。

表 3-3　FIFO 调度端到端时延上界统计表

TTVL 帧	时延上界（μs）	TTVL 帧	时延上界（μs）
VL1	242.49	*VL7*	274.62
VL2	201.43	*VL8*	464.26
VL3	289.81	*VL9*	371.73
VL4	182.47	*VL10*	243.54
VL5	324.63	*VL11*	365.52
VL6	464.1	VL12	83.94

2）非抢占静态优先级调度算法实时性分析

非抢占 SP 调度算法按照系统需求将数据分级，高优先级享有优先调度权，仅当其队列为空时下一优先级队列才能够获取服务。非抢占 SP 调度策略有效地提高了紧急数据传输的时间确定性，但是逻辑设计相对复杂，如图 3-10 所示。

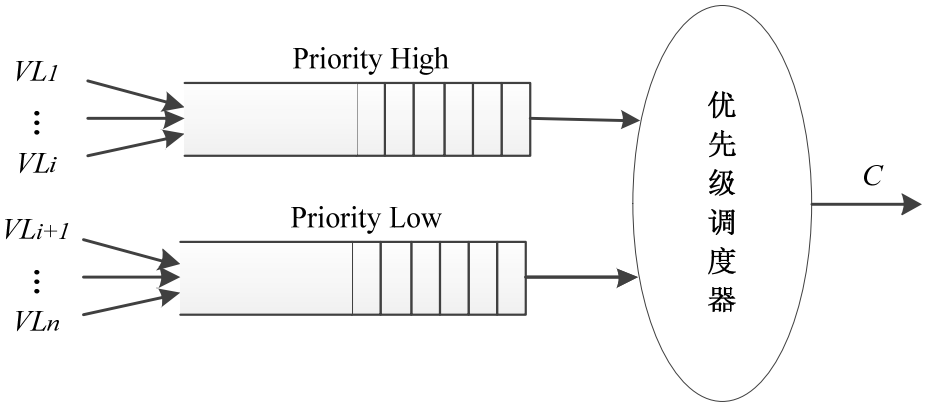

图 3-10 非抢占 SP 调度算法示意图

AFDX 系统支持对虚链路分为两个优先级，将图 3-6 中的 TTVL 设为高优先级，则 *VL1*、*VL3*、*VL4*、*VL6*、*VL7*、*VL8* 和 *VL11* 都具有高优先级，其余 VL 为低优先级。各 VL 的优先级使用 $p(i)$ 进行说明，当 VL*i* 为高优先级时，$p(i)=H$；当 VL*i* 为低优先级时，$p(i)=L$。[100]

对于共享交换机同一输出端口的多个 VL，定义 $G^H=\bigcup_{\{p(i)=H\}} VLi$ 和 $G^L=\bigcup_{\{p(i)=L\}} VLi$ 分别代表高优先级和低优先级虚链路的聚合流，$l_{\max}^{L}=\max\{L_{\max}^{i}:p(i)=L\}$ 为低优先级 VL

的最长数据帧长度。假设 VL 的到达曲线为 $\alpha(t)=rt+b$，则 $A^L(t)=\sum_{\{p(i)=L\}}\alpha_i(t)$ 和 $A^H(t)=\sum_{\{p(i)=H\}}\alpha_i(t)$ 分别为 G^L 和 G^H 的到达曲线。

当 G^L 和 G^H 采用非抢占式 SP 调度策略接受交换机服务时，由于输出端口物理带宽为 C，则其提供给所有虚链路的总服务曲线为：$\beta(t)=C[t-0]^+$，而提供给聚合 VL 流 G^H 的服务曲线为：

$$\beta_{G^H}(t)=\left[\beta(t)-l^L_{\max}\right]^+=C\left[t-\frac{l^L_{\max}}{C}\right]^+ \tag{3-51}$$

记 VL 聚合流 G^H 的服务速率和服务时延参数分别为

$$R_{G^H}=C,\quad T_{G^H}=l^L_{\max}/C \tag{3-52}$$

而对于 VL 聚合流 G^L 的服务曲线为：

$$\beta_{G^L}(t)=\left[\beta(t)-A^H(t)\right]^+ \tag{3-53}$$

其中：

$$A^H(t)=\sum_{\{p(i)=H\}}\alpha_i(t)=\sum_{\{p(i)=H\}}r_i\times t+\sum_{\{p(i)=H\}}b_i \tag{3-54}$$

将公式（3-54）代入公式（3-53），得

$$\begin{aligned}\beta_{G^L}(t)&=\left[Ct-\sum_{\{p(i)=H\}}r_i\times t-\sum_{\{p(i)=H\}}b_i\right]^+\\&=\left[C-\sum_{\{p(i)=H\}}r_i\right]\left[t-\frac{\sum_{\{p(i)=H\}}b_i}{C-\sum_{\{p(i)=H\}}r_i}\right]^+\end{aligned} \tag{3-55}$$

记 VL 聚合流 G^L 的服务速率和服务时延参数分别为

$$R_{G^L}=C-\sum_{\{p(i)=H\}}r_i \tag{3-56}$$

$$T_{G^L}=\sum_{\{p(i)=H\}}b_i\Big/R_{G_L} \tag{3-57}$$

对于聚合流 G^L 的组成虚链路 VLi，其在 G^L 中仍以 FIFO 调度策略接受交换机的服务，按照本小节 FIFO 调度策略部分的相关推导，可得：

$$\begin{aligned}&\beta_{VL_i}(t)\\&=\left[R_{G^L}(t-T_{G^L})-\left[\sum_{\{p(j)=L\}}r_j-r_i\right]\times\left[t-T_{G^L}-\frac{\sum_{\{p(j)=L\}}b_j-b_i}{R_{G^L}}\right]-\left[\sum_{\{p(j)=L\}}b_j-b_i\right]\right]^+\\&=\left[R_{G^L}-\sum_{\{p(j)=L\}}r_j+r_i\right]\times\left[t-\frac{R_{G^L}\times T_{G^L}+\sum_{\{p(j)=L\}}b_j-b_i}{R_{G^L}}\right]^+\end{aligned}\tag{3-58}$$

则 VL*i* 的服务速率和服务时延参数分别为：

$$R_{VL_i}=R_{G^L}-\sum_{\{p(j)=L\}}r_j+r_i\tag{3-59}$$

$$T_{VL_i}=T_{G^L}+\left(\sum_{\{p(j)=L\}}b_j-b_i\right)\Big/R_{G^L}\tag{3-60}$$

类似的可得聚合流 G^H 中的 VL*i* 服务速率和服务时延参数分别为：

$$R_{VL_i}=R_{G^H}-\sum_{\{p(j)=H\}}r_j+r_i=C-\sum_{\{p(j)=H\}}r_j+r_i\tag{3-61}$$

$$T_{VL_i}=T_{G^H}+\left(\sum_{\{p(j)=H\}}b_j-b_i\right)\Big/R_{G^H}\tag{3-62}$$

SW1 为 *VL1*、*VL2* 和 *VL5* 提供向 ES6 转发服务的的聚合流服务曲线为 $\beta_{SW1-ES6}(t)=Ct$。其中 *VL1* 具有高优先级，则其获得 SW1 提供的服务曲线为：

$$\beta_{SW1,1}(t)=\left[\beta(t)-l^L_{\max}\right]^+=C\left[t-\frac{L^5_{\max}}{C}\right]^+\tag{3-63}$$

由上式可得虚链路 *VL1* 的串联服务曲线为：

$$\begin{aligned}\beta_1^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW1,1}(t)\otimes\delta_\eta\otimes\delta_{r\nu}\otimes\delta_\tau\\&=C\times\left[t-\left(\frac{L^5_{\max}}{C}+2\times\tau+\eta+L^1_{\max}/C\right)\right]^+\end{aligned}\tag{3-64}$$

SW1 向 *VL2* 和 *VL5* 低优先级聚合流提供的服务曲线为：

$$\beta_{G^L}(t)=\left[Ct-\frac{L^1_{\max}}{BAG_1}t-L^1_{\max}\right]^+=\left[C-\frac{L^1_{\max}}{BAG_1}\right]\left[t-\frac{L^1_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}\right]^+\tag{3-65}$$

由公式（3-65）可得 *VL2* 和 *VL5* 所获得的服务曲线分别为：

$$\beta_{SW1,i}(t)=\left[C-\frac{L^1_{\max}}{BAG_1}-\left(\sum_{j=2,5}\frac{L^j_{\max}}{BAG_j}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{L^1_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}-\frac{\sum_{j=2,5}L^j_{\max}-L^i_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}\right)^+ \tag{3-66}$$

由上式可得虚链路 *VL2* 和 *VL5* 的串联服务曲线为：

$$\begin{aligned}\beta_i^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW1,i}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\left[C-\frac{L^1_{\max}}{BAG_1}-\left(\sum_{j=2,5}\frac{L^j_{\max}}{BAG_i}-\frac{L^i_{\max}}{BAG_i}\right)\right]\\&\times\left[t-\frac{L^1_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}-\left(\frac{\sum_{j=2,5}L^j_{\max}-L^i_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}+2\times\tau+\eta+L^i_{\max}/C\right)\right]^+\end{aligned} \tag{3-67}$$

由于 SW1 仅为 *VL3* 和 *VL4* 提供向 SW3 转发服务的聚合流服务曲线为 $\beta_{SW1-SW3}(t)=Ct$，可得 *VL3* 和 *VL4* 所获得的服务曲线分别与公式（3-25）相同，且 *VL3* 和 *VL4* 对于 SW3 的到达曲线与公式（3-26）相同。

SW2 为具有高优先级的 *VL6*、*VL7* 和 *VL8* 聚合流提供的服务曲线为：

$$\beta_{SW2,G^H}(t)=\left[\beta(t)-l^L_{\max}\right]^+=C\left[t-\frac{L^9_{\max}}{C}\right]^+ \tag{3-68}$$

由公式（3-68）可得 *VL6*、*VL7* 和 *VL8* 所获得的服务曲线分别为：

$$\beta_{SW2,i}(t)=\left[C-\left(\sum_{j=6}^{8}\frac{L^j_{\max}}{BAG_j}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{L^9_{\max}}{C}-\frac{\sum_{j=6}^{8}L^j_{\max}-L^i_{\max}}{C}\right)^+ \tag{3-69}$$

VL6、*VL7* 和 *VL8* 离开 SW2 后，对于 SW3 的到达曲线为：

$$\alpha_i'=\alpha_i\left(t+\frac{L^9_{\max}}{C}+\frac{\sum_{j=6}^{8}L^j_{\max}-L^i_{\max}}{C}\right) \tag{3-70}$$

SW2 为具有低优先级的 *VL9* 和 *VL10* 聚合流提供的服务曲线为：

$$\begin{aligned}\beta_{SW2,G^L}(t)&=\left[Ct-\sum_{j=6}^{8}\frac{L^j_{\max}}{BAG_j}\times t-\sum_{j=6}^{8}L^j_{\max}\right]^+\\&=\left[C-\sum_{j=6}^{8}\frac{L^j_{\max}}{BAG_j}\right]\left[t-\frac{\sum_{j=6}^{8}L^j_{\max}}{C-\sum_{j=6}^{8}\frac{L^j_{\max}}{BAG_j}}\right]^+\end{aligned} \tag{3-71}$$

由公式（3-71）可得 *VL9* 和 *VL10* 所获得的服务曲线为：

$$\beta_{SW2,i}(t)=\left[C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}-\left(\sum_{j=9,10}\frac{L_{\max}^{j}}{BAG_j}-\frac{L_{\max}^{i}}{BAG_i}\right)\right]\left(t-\frac{\sum_{j=6}^{8}L_{\max}^{j}+\sum_{j=9,10}L_{\max}^{j}-L_{\max}^{i}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}\right)^{+} \tag{3-72}$$

由公式（3-72）可得 *VL9* 和 *VL10* 离开 SW2 后对于 SW3 的到达曲线为：

$$\alpha_i'=\alpha_i\left(t+\frac{\sum_{j=6}^{8}L_{\max}^{j}+\sum_{j=9,10}L_{\max}^{j}-L_{\max}^{i}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}\right) \tag{3-73}$$

经 SW3 转发至 ES7 的具有高优先级的 VL 为：*VL3*、*VL6*、*VL8* 和 *VL11*，其聚合流到达曲线 $\alpha_{SW3-ES7,G^H}(t)$ 为：

$$\alpha_{SW3-ES7,G^H}(t)=\sum_{i=3,6,8}\alpha_i'(t)+\alpha_{11}(t)=t\times\sum_{i=3,6,8,11}\frac{L_{\max}^{i}}{BAG_i}+L' \tag{3-74}$$

其中：$L'=\sum_{i=3,6,8,11}L_{\max}^{i}+\frac{L_{\max}^{4}}{C}\frac{L_{\max}^{3}}{BAG_3}+\frac{\sum_{i=7,8,9}L_{\max}^{i}}{C}\frac{L_{\max}^{6}}{BAG_6}+\frac{\sum_{i=6,7,9}L_{\max}^{i}}{C}\frac{L_{\max}^{8}}{BAG_8}$。

而 *VL4*、*VL7*、*VL10* 和 *VL12* 转发至 ES8，其在 SW3 输出端口的高优先级和低优先级聚合流到达曲线 $\alpha_{SW3-ES8,G^H}(t)$ 和 $\alpha_{SW3-ES8,G^L}(t)$ 分别为：

$$\alpha_{SW3-ES8,G^H}(t)=\sum_{i=4,7}\alpha'(t)=t\sum_{i=4,7}\frac{L_{\max}^{i}}{BAG_i}+L_H'' \tag{3-75}$$

其中：$L_H''=\sum_{i=4,7}L_{\max}^{i}+\frac{L_{\max}^{3}}{C}\frac{L_{\max}^{4}}{BAG_4}+\frac{\sum_{i=6,8,9}L_{\max}^{i}}{C}\frac{L_{\max}^{7}}{BAG_7}$。

$$\alpha_{SW3-ES8,G^L}(t)=\alpha_{10}'(t)+\alpha_{12}(t)=t\sum_{i=10,12}\frac{L_{\max}^{i}}{BAG_i}+L_L'' \tag{3-76}$$

其中：$L_L''=\sum_{i=10,12}L_{\max}^{i}+\frac{\sum_{i=6,7,8,9}L_{\max}^{i}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}\frac{L_{\max}^{10}}{BAG_{10}}$。

可得 *VL3*、*VL6*、*VL8* 和 *VL11* 在 SW3 中获得的聚合流服务曲线为：

$$\beta_{SW3-ES7,G^H}(t)=\left[\beta(t)-l_{\max}^{L}\right]^{+}=C\left[t-\frac{L_{\max}^{9}}{C}\right]^{+} \tag{3-77}$$

由公式（3-77）可得 *VL3*，*VL6*，*VL8* 和 *VL11* 所获得的服务曲线分别为：

$$\beta_{SW3-ES7,i}(t)=\left[C-\left(\sum_{j=3,6,8,11}\frac{L_{\max}^{j}}{BAG_i}-\frac{L_{\max}^{i}}{BAG_i}\right)\right]\left(t-\frac{L_{\max}^{9}}{C}-\frac{L'-L_{\max}^{i}{}'}{C}\right)^{+} \tag{3-78}$$

其中：$L_{\max}^{11}{}'=L_{\max}^{11}$，$L_{\max}^{3}{}'=L_{\max}^{3}{}^{*}$，

$$L_{\max}^{6}{}'=L_{\max}^{6}+\frac{\sum_{i=7,8,9}L_{\max}^{i}}{C}\times\frac{L_{\max}^{6}}{BAG_6} \tag{3-79}$$

$$L_{\max}^{8}{}'=L_{\max}^{8}+\frac{\sum_{i=6,7,9}L_{\max}^{i}}{C}\times\frac{L_{\max}^{8}}{BAG_8} \tag{3-80}$$

由 SW3 向 ES7 转发的低优先级 VL 仅有 *VL9*，其所获得的服务曲线为：

$$\beta_{SW3-ES7,9}(t)=\left[C-\sum_{j=3,6,8,11}\frac{L_{\max}^{j}}{BAG_j}\right]\left(t-\frac{\sum_{j=3,6,8,11}L_{\max}^{j}{}'}{C-\sum_{j=3,6,8,11}\frac{L_{\max}^{j}}{BAG_j}}\right)^{+} \tag{3-81}$$

VL4 和 *VL7* 在 SW3 中获得的聚合流服务曲线为：

$$\beta_{SW3-ES8,G^H}(t)=\left[\beta(t)-l_{\max}^{L}\right]^{+}=C\left[t-\frac{L_{\max}^{10}}{C}\right]^{+} \tag{3-82}$$

由公式（3-82）可得 *VL4* 和 *VL7* 所获得的服务曲线分别为：

$$\beta_{SW3-ES8,i}(t)=\left[C-\left(\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_i}-\frac{L_{\max}^{i}}{BAG_i}\right)\right]\left(t-\frac{L_{\max}^{10}}{C}-\frac{L_H''-L_{\max}^{i}{}'}{C}\right)^{+} \tag{3-83}$$

其中：$L_{\max}^{4}{}'=L_{\max}^{4}{}^{*}$，

$$L_{\max}^{7}{}'=L_{\max}^{7}+\frac{\sum_{i=6,8,9}L_{\max}^{i}}{C}\times\frac{L_{\max}^{7}}{BAG_7} \tag{3-84}$$

由 SW3 向 ES8 转发的低优先级 *VL10* 和 *VL12* 所获得的服务曲线分别为：

$$\beta_{SW3-ES8,i}(t)=\left[C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}-\left(\sum_{j=10,12}\frac{L_{\max}^{j}}{BAG_j}-\frac{L_{\max}^{i}}{BAG_i}\right)\right]\left(t-\frac{L_H''+L_L''-L_{\max}^{i}{}'}{C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}}\right)^{+} \tag{3-85}$$

其中：$L_{\max}^{12}{}'=L_{\max}^{12}$，

$$L_{\max}^{10\ '}=L_{\max}^{10}+\frac{\sum_{i=6}^{9}L_{\max}^{i}}{C-\sum_{j=6,7,8}\frac{L_{\max}^{j}}{BAG_{j}}}\times\frac{L_{\max}^{10}}{BAG_{10}} \tag{3-86}$$

虚链路 *VL3* 的串联服务曲线为：

$$\begin{aligned}\beta_{3}^{\Sigma}(t)&=\delta_{\tau}\otimes\beta_{SW1,3}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\otimes\beta_{SW3-ES7,3}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\\&=\min\{\beta_{SW1,3}(t),\beta_{SW3-ES7,3}(t)\}\\&\times\left[t-\left(\frac{L_{\max}^{4}+L_{\max}^{9}+L'-L_{\max}^{3}\ '}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^{3}}{C}\right)\right)\right]^{+}\end{aligned} \tag{3-87}$$

虚链路 *VL4* 的串联服务曲线为：

$$\begin{aligned}\beta_{4}^{\Sigma}(t)&=\delta_{\tau}\otimes\beta_{SW1,4}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\otimes\beta_{SW3-ES8,4}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\\&=\min\{\beta_{SW1,4}(t),\beta_{SW3-ES8,4}(t)\}\\&\times\left[t-\left(\frac{L_{\max}^{3}+L_{\max}^{10}+L_{H}''-L_{\max}^{4}\ '}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^{4}}{C}\right)\right)\right]^{+}\end{aligned} \tag{3-88}$$

虚链路 *VL6* 和 *VL8* 的串联服务曲线为：

$$\begin{aligned}\beta_{6}^{\Sigma}(t)&=\delta_{\tau}\otimes\beta_{SW2,6}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\otimes\beta_{SW3-ES7,6}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\\&=\min\{\beta_{SW2,6}(t),\beta_{SW3-ES7,6}(t)\}\\&\times\left[t-\left(\frac{L_{\max}^{9}+\sum_{j=6}^{8}L_{\max}^{j}-L_{\max}^{6}+L_{\max}^{9}+L'-L_{\max}^{6}\ '}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^{6}}{C}\right)\right)\right]^{+}\end{aligned} \tag{3-89}$$

虚链路 *VL7* 的串联服务曲线为：

$$\begin{aligned}\beta_{7}^{\Sigma}(t)&=\delta_{\tau}\otimes\beta_{SW2,7}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\otimes\beta_{SW3-ES8,7}(t)\otimes\delta_{\eta}\otimes\delta_{rv}\otimes\delta_{\tau}\\&=\min\{\beta_{SW2,7}(t),\beta_{SW3-ES8,7}(t)\}\\&\times\left[t-\left(\frac{L_{\max}^{9}+\sum_{j=6}^{8}L_{\max}^{j}-L_{\max}^{7}+L_{\max}^{10}+L_{H}''-L_{\max}^{7}\ '}{C}+3\tau+2\times\left(\eta+\frac{L_{\max}^{7}}{C}\right)\right)\right]^{+}\end{aligned} \tag{3-90}$$

虚链路 *VL9* 的串联服务曲线为：

$$\begin{aligned}\beta_9^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW2,9}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES7,9}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\min\{\beta_{SW2,9}(t),\beta_{SW3-ES7,9}(t)\}\\&\times\left[t-\left(\frac{\sum_{j=6}^{8}L_{\max}^{j}+\sum_{j=9,10}L_{\max}^{j}-L_{\max}^{9}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}+\frac{\sum_{j=3,6,8,11}L_{\max}^{j}{}'}{C-\sum_{j=3,6,8,11}\frac{L_{\max}^{j}}{BAG_j}}+3\tau+2\left(\eta+\frac{L_{\max}^{9}}{C}\right)\right)\right]^{+}\end{aligned}\tag{3-91}$$

虚链路 *VL10* 的串联服务曲线为：

$$\begin{aligned}\beta_{10}^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW2,10}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES8,10}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\min\{\beta_{SW2,10}(t),\beta_{SW3-ES8,10}(t)\}\\&\times\left[t-\left(\frac{\sum_{j=6}^{8}L_{\max}^{j}+\sum_{j=9,10}L_{\max}^{j}-L_{\max}^{10}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}+\frac{L_H''+L_L''-L_{\max}^{10}{}'}{C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}}+3\tau+2\left(\eta+\frac{L_{\max}^{10}}{C}\right)\right)\right]^{+}\end{aligned}\tag{3-92}$$

虚链路 *VL11* 的串联服务曲线为：

$$\begin{aligned}\beta_{11}^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW3-ES7,11}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\left[C-\sum_{i=3,6,8,}\frac{L_{\max}^{i}}{BAG_i}\right]\times\left[t-\left(\frac{L_{\max}^{9}+L'-L_{\max}^{11}{}'}{C}+2\tau+\eta+L_{\max}^{11}/C\right)\right]^{+}\end{aligned}\tag{3-93}$$

虚链路 *VL12* 的串联服务曲线为：

$$\begin{aligned}\beta_{12}^{\Sigma}(t)&=\delta_\tau\otimes\beta_{SW3-ES8,12}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\\&=\left[C-\sum_{j=4,7,10}\frac{L_{\max}^{j}}{BAG_j}\right]\times\left[t-\left(\frac{L_H''+L_L''-L_{\max}^{12}{}'}{C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}}+2\tau+\eta+L_{\max}^{12}/C\right)\right]^{+}\end{aligned}\tag{3-94}$$

根据虚链路配置表 3-2，已有 AFDX 采用 SP 调度算法时，各条虚链路的延迟上界计算结果如表 3-4。

表 3-4　AFDX 采用 SP 调度策略 VL 端到端时延上界统计表

TTVL 帧	时延上界（μs）	TTVL 帧	时延上界（μs）
*VL*1	221.8	*VL*7	258.86
*VL*2	201.74	*VL*8	453.99
*VL*3	246.04	*VL*9	373.3
*VL*4	176.94	*VL*10	243.87
*VL*5	324.78	*VL*11	348.49
*VL*6	453.89	*VL*12	119.88

3.5 TTAFDX 时间触发虚链路实时性分析

在 3.3.1 节设计的网络模型中，共有 12 条虚链路，其中 *VL1*、*VL3*、*VL4*、*VL6*、*VL7*、*VL8* 和 *VL*11 是承载时间关键消息的时间触发虚链路。*VL*1 由端系统 ES1 发送给交换机 SW1，因为端系统 ES1 只有一条时间触发虚链路需要发送，所以 *VL1* 在每个基本周期的 SYNC 帧后进行发送。

时间触发虚链路 *VL3* 和 *VL4* 由端系统 ES2 发送至交换机 SW1，按照周期优先调度表设计方法对对 *VL3* 和 *VL4* 调度时刻进行规划表，结果如图 3-11 所示。

在规划时刻调度表时，首先按照 BAG 的大小对 TTVL 进行排序，BAG 小的 TTVL 优先进行规划。对于 *VL3* 和 *VL4* 这两条 TTVL 来说，*VL3* 的 BAG 小于 *VL4* 的 BAG，优先对 *VL3* 进行调度规划。图 3-11 的端系统调度时刻表中，基本周期为 1ms，任务周期为 128ms，*VL3* 在一个任务周期中可以被调度 4 次，*VL4* 可以被调度 2 次。

在端系统 ES2，*VL3* 和 *VL4* 第一次被调度的时刻，即 $V_{3,1}$ 和 $V_{4,1}$ 的发送时刻为：

$$t_{3,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^{6}}=\frac{28\times 8\times 1000}{100\times 10^{6}}=0.00224ms$$

$$t_{4,1}=1+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^{6}}=1+\frac{28\times 8\times 1000}{100\times 10^{6}}=1.00224ms$$

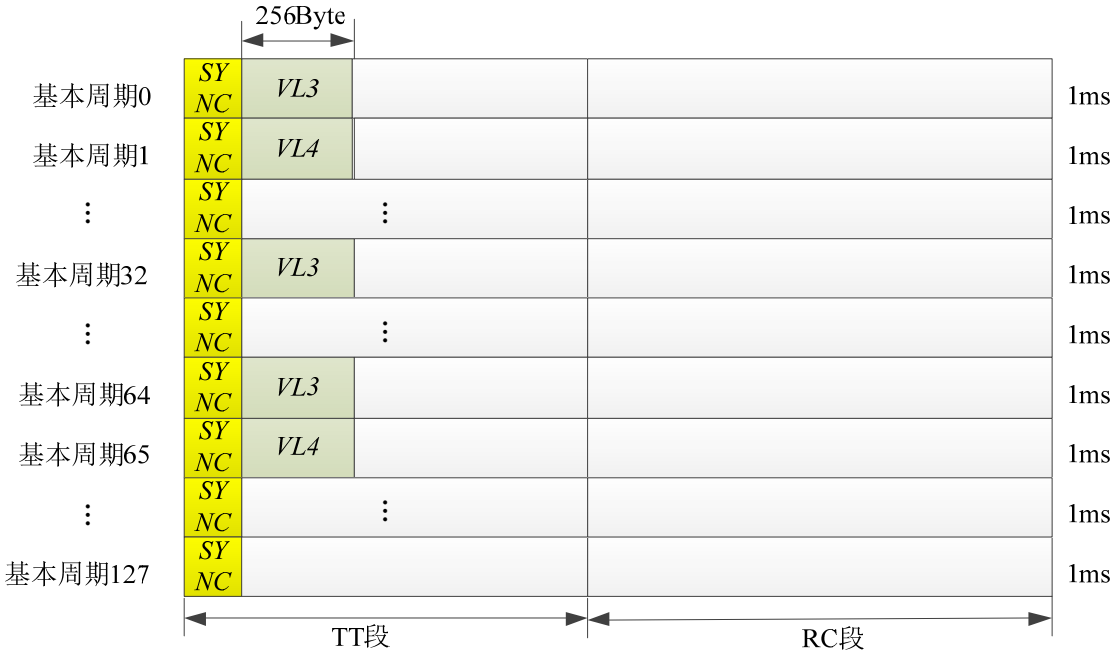

图 3-11 端系统 ES2 调度时刻表

其它数据帧被调度的时刻根据 3.1.1 小节中描述的周期优先调度表设计方法步骤 3 中所述的不同情况选用不同的计算公式，所得统计结果如表 3-5 所示。

表 3-5　时间触发虚链路在端系统 ES2 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{3,1}$	0.00224	$V_{3,4}$	96.00224
$V_{3,2}$	32.00224	$V_{4,1}$	1.00224
$V_{3,3}$	64.00224	$V_{4,2}$	65.00224

VL1、*VL3* 和 *VL4* 三条 TTVL 到达交换机 SW1 之后，对不同端口分别进行时刻调度表规划。通过查看 ES1 和 ES2 的时刻调度表能够知道 *VL1*、*VL3* 和 *VL4* 各个帧的发送时间，从而确定接收时窗的关闭时刻，为了讨论简单，以下章节均不考虑时钟漂移。由于仅有 *VL1* 一条 TTVL 通过 SW1 发送至 ES6，不用对此端口所转发 VL 进行排序处理。当接收到 *VL1* 数据帧后，按照交换机调度表设计方法可得 SW1—ES6 调度时刻表，如图 3-12 所示。

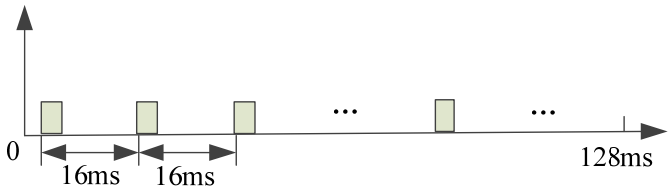

图 3-12 交换机 SW1 对虚链路 *VL1* 的调度时刻示意图

在交换机 SW1 中，*VL1* 第一次被调度的时刻，即 $V_{1,1}$ 的转发时刻为：

$$\begin{aligned} t_{1,1} &= \tau_{frame,k} + \tau_{rv} + \tau_{filter} + \tau_{fw} + D_{propagation} \\ &= \frac{L_{SYNC} \times 8 \times 1000}{100 \times 10^6} + 2 \times \frac{L_{VL_1} \times 8 \times 1000}{100 \times 10^6} + \tau_{filter} + \tau_{fw} + D_{propagation} \\ &= \frac{(28 + 2 \times 512) \times 8 \times 1000}{100 \times 10^6} + 0.016 + 0.0005 = 0.10066ms \end{aligned}$$

依次对一个 MC 中 *VL1* 所发送的 8 个数据帧进行类似计算，所得统计结果如表 3-6 所示。

表 3-6　时间触发虚链路 *VL1* 在交换机 SW1 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{1,1}$	0.10066	$V_{1,5}$	64. 10066
$V_{1,2}$	16. 10066	$V_{1,6}$	80. 10066
$V_{1,3}$	32. 10066	$V_{1,7}$	96. 10066
$V_{1,4}$	48. 10066	$V_{1,8}$	112. 10066

VL3 和 *VL4* 两条 TTVL 到达交换机 SW1 之后将转发至交换机 SW3，由于共用一个端口，首先需要进行排序。对于 *VL3* 和 *VL4* 这两条虚链路来说，*VL3* 的 BAG（32ms）

小于 *VL4* 的 BAG（64ms），优先对 *VL3* 周期内各数据帧进行调度规划，通过交换机调度表设计方法可得 SW1—SW3 的调度时刻，如图 3-13 所示。

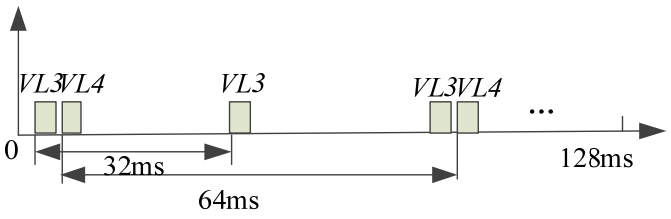

图 3-13 交换机 SW1 对虚链路 *VL3* 和 *VL4* 的调度时表示意图

在交换机 SW1 第一次调度 *VL3*、*VL4* 的时刻，即 $V_{3,1}$ 和 $V_{4,1}$ 的转发时刻分别为：

$$\begin{aligned} t_{3,1} &= \tau_{frame,k} + \tau_{rv} + \tau_{filter} + \tau_{fw} + D_{propagation} \\ &= \frac{L_{SYNC} \times 8 \times 1000}{100 \times 10^6} + 2 \times \frac{L_{VL_3} \times 8 \times 1000}{100 \times 10^6} + \tau_{filter} + \tau_{fw} + D_{propagation} \\ &= \frac{(28 + 2 \times 128) \times 8 \times 1000}{100 \times 10^6} + 0.016 + 0.0005 = 0.03922ms \end{aligned}$$

$$\begin{aligned} t_{4,1} &= 1 + \frac{L_{SYNC} \times 8 \times 1000}{100 \times 10^6} + 2 \times \frac{L_{VL_4} \times 8 \times 1000}{100 \times 10^6} + \tau_{filter} + \tau_{fw} + D_{propagation} \\ &= 1 + \frac{(28 + 2 \times 256) \times 8 \times 1000}{100 \times 10^6} + 0.016 + 0.0005 = 1.0597ms \end{aligned}$$

依次对 *VL3* 一个矩阵周期中的 4 个数据帧及 *VL4* 一个矩阵周期中的 2 个数据帧进行类似计算，所得统计结果如表 3-7 所示。

表 3-7　时间触发虚链路 *VL3* 和 *VL4* 在交换机 SW1 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{3,1}$	0.03922	$V_{4,1}$	1.0597
$V_{3,2}$	32. 03922	$V_{4,2}$	65. 0597
$V_{3,3}$	64. 03922		
$V_{3,4}$	96. 03922		

时间触发虚链路 *VL6*、*VL7* 和 *VL8* 由端系统 ES3 发送至交换机 SW2，所以在端系统 ES2 需要对 *VL6*、*VL7* 和 *VL8* 设计调度时刻表，按照周期优先调度表设计方法可得 ES2 的调度时刻表，如图 3-14 所示.

在规划端系统 ES3 调度时刻表时，首先按照 BAG 的大小对 TTVL 进行排序，BAG 小的 TTVL 优先进行规划。对于 *VL6*、*VL7* 和 *VL8* 这三条虚链路来说，*VL6* 和 *VL7* 的 BAG 小于 *VL8* 的 BAG，且 *VL6* 的帧长大于 *VL7* 的帧长，则依照 *VL6*、*VL7* 和 *VL8* 的顺序进行调度规划。在图 3-14 的端系统调度时刻表中，*VL6*、*VL7* 在一个任务周期中可以被调度 4 次，*VL8* 可以被调度 2 次。

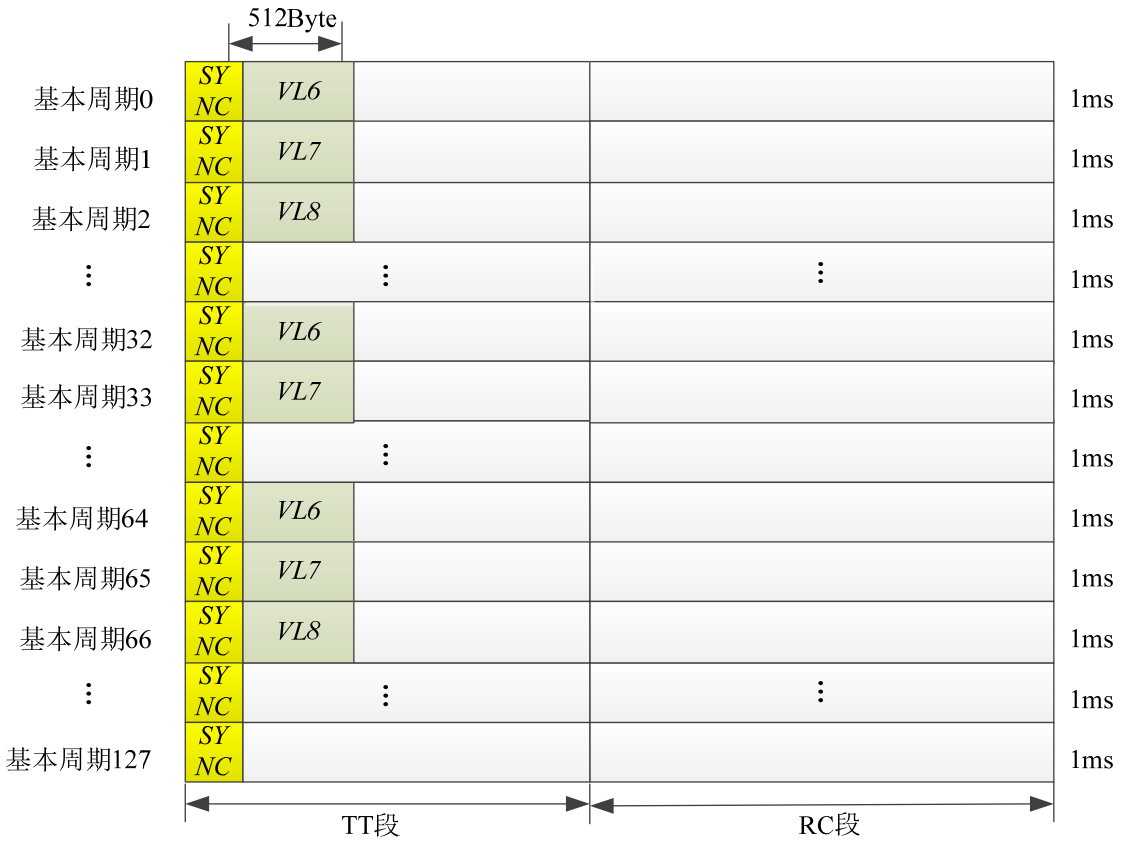

图 3-14 端系统 ES3 的调度时刻表

在端系统 ES3，*VL6*、*VL7* 和 *VL8* 首次被调度的时刻，即$V_{6,1}$、$V_{7,1}$和$V_{8,1}$的发送时刻为：

$$t_{6,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^{6}}=\frac{28\times 8\times 1000}{100\times 10^{6}}=0.00224ms$$

$$t_{7,1}=1+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^{6}}=1+\frac{28\times 8\times 1000}{100\times 10^{6}}=1.00224ms$$

$$t_{8,1}=2+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^{6}}=2+\frac{28\times 8\times 1000}{100\times 10^{6}}=2.00224ms$$

表 3-8 时间触发虚链路在端系统 ES2 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{6,1}$	0.00224	$V_{7,2}$	33.00224
$V_{6,2}$	32.00224	$V_{7,3}$	65.00224
$V_{6,3}$	64.00224	$V_{7,4}$	97.00224
$V_{6,4}$	96.00224	$V_{8,1}$	2.00224
$V_{7,1}$	1.00224	$V_{8,2}$	66.00224

其它数据帧被调度的时刻根据 3.1.1 小节中描述的周期优先调度表设计方法步骤 3 中所述的不同情况选用不同的计算公式，所得统计结果如表 3-8 所示。

*VL*11 由端系统 ES5 发送给交换机 SW3，因为端系统 ES5 只有一条时间触发虚链路需要发送，所以 *VL*11 在每个基本周期的 SYNC 帧后进行发送。

VL6、*VL7* 和 *VL8* 三条 TTVL 到达交换机 SW2 之后将转发至交换机 SW3，由于共用一个端口，首先需要进行排序。由于 *VL6* 和 *VL7* 的 BAG 小于 *VL8* 的 BAG，且 *VL6* 的帧长大于 *VL7* 的帧长，则依照 *VL6*、*VL7* 和 *VL8* 的顺序进行调度规划，由交换机调度表设计方法可得 SW2—SW3 的调度时刻，如图 3-15 所示。

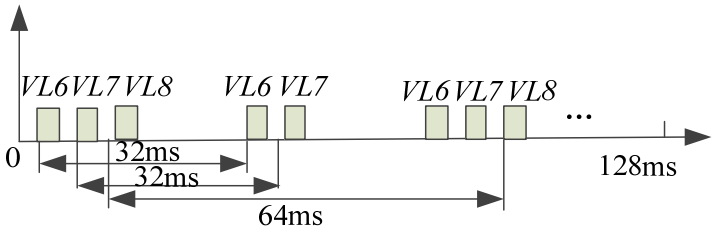

图 3-15 交换机 SW2 对虚链路 *VL6*、*VL7* 和 *VL8* 的时刻调度示意图

在交换机 SW2 中，*VL6*、*VL7* 和 *VL8* 第一次被调度的时刻，即$V_{6,1}$、$V_{7,1}$和$V_{8,1}$的转发时刻为：

$$t_{6,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+2\times\frac{L_{VL_6}\times 8\times 1000}{100\times 10^6}+\tau_{filter}+\tau_{fw}+D_{propagation}$$
$$=\frac{(28+2\times 512)\times 8\times 1000}{100\times 10^6}+0.016+0.0005=0.10066ms$$

$$t_{7,1}=1+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+2\times\frac{L_{VL_7}\times 8\times 1000}{100\times 10^6}+\tau_{filter}+\tau_{fw}+D_{propagation}$$
$$=1+\frac{(28+2\times 256)\times 8\times 1000}{100\times 10^6}+0.016+0.0005=1.0597ms$$

$$t_{8,1}=2+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+2\times\frac{L_{VL_8}\times 8\times 1000}{100\times 10^6}+\tau_{filter}+\tau_{fw}+D_{propagation}$$
$$=2+\frac{(28+2\times 512)\times 8\times 1000}{100\times 10^6}+0.016+0.0005=2.10066ms$$

依次对 *VL6* 和 *VL7* 一个矩阵周期中的 4 个数据帧及 *VL8* 一个矩阵周期中的 2 个数据帧进行类似计算，所得统计结果如表 3-9 所示。

表 3-9　时间触发虚链路 *VL6*、*VL7* 和 *VL8* 在交换机 SW1 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{6,1}$	0.10066	$V_{7,2}$	33. 0597
$V_{6,2}$	32. 10066	$V_{7,3}$	65. 0597
$V_{6,3}$	64. 10066	$V_{7,4}$	97. 0597
$V_{6,4}$	96. 10066	$V_{8,1}$	2. 10066
$V_{7,1}$	1.0597	$V_{8,2}$	66. 10066

VL3、*VL4*、*VL6*、*VL7*、*VL8* 和 *VL11* 六条 TTVL 到达交换机 SW3 之后，对不同端口分别进行调度时刻表规划。通过查看 SW1、SW2 和 ES5 的调度时刻表能够知道各 TTVL 在一个矩阵周期中所有帧的发送时间，从而确定接收时窗的关闭时刻。

VL3、*VL6*、*VL8* 和 *VL11* 四条 TTVL 到达交换机 SW3 之后将转发至端系统 ES7，由于共用一个端口，首先需要进行排序。*VL11* 的 BAG 最小，*VL3* 和 *VL6* 的 BAG 次之，*VL8* 的 BAG 最大，且 *VL6* 的帧长大于 *VL3* 的帧长，则依照 *VL11*、*VL6*、*VL3* 和 *VL8* 的顺序进行调度表规划，通过使用交换机调度表设计方法可得 SW3—ES7 时刻调度，如图 3-16 所示。

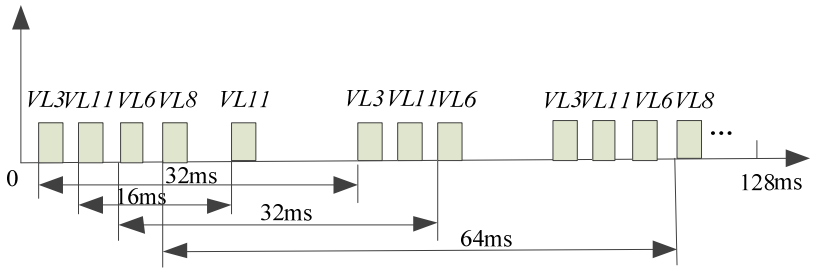

图 3-16 交换机 SW3 对虚链路 *VL3*、*VL6* 和 *VL8* 的时刻调度示意图

在交换机 SW3 中，*VL3*、*VL6*、*VL8* 和 *VL11* 第一次被调度的时刻，即 $V_{3,1}$、$V_{6,1}$、$V_{8,1}$ 和 $V_{11,1}$ 的转发时刻为：

$$t_{11,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+2\times\frac{L_{VL_{11}}\times 8\times 1000}{100\times 10^6}+\left(\tau_{filter}+\tau_{fw}+D_{propagation}\right)$$
$$=\frac{(28+2\times 1024)\times 8\times 1000}{100\times 10^6}+(0.016+0.0005)=0.18258ms$$

$$t_{3,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+4\times\frac{L_{VL_3}\times 8\times 1000}{100\times 10^6}+2\times\left(\tau_{filter}+\tau_{fw}+D_{propagation}\right)$$
$$=\frac{(28+4\times 128)\times 8\times 1000}{100\times 10^6}+2\times(0.016+0.0005)=0.0762ms$$

$$t_{6,1}=\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+4\times\frac{L_{VL_6}\times 8\times 1000}{100\times 10^6}+2\times\left(\tau_{filter}+\tau_{fw}+D_{propagation}\right)$$
$$=\frac{(28+4\times 512)\times 8\times 1000}{100\times 10^6}+2\times(0.016+0.0005)=0.19908ms$$

$$t_{8,1}=2+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+4\times\frac{L_{VL_8}\times 8\times 1000}{100\times 10^6}+2\times\left(\tau_{filter}+\tau_{fw}+D_{propagation}\right)$$
$$=2+\frac{(28+4\times 512)\times 8\times 1000}{100\times 10^6}+2\times(0.016+0.0005)=2.19908ms$$

依次对 *VL3* 和 *VL6* 一个矩阵周期中的 4 个数据帧，*VL8* 一个矩阵周期中的 2 个数据帧以及 *VL11* 一个矩阵周期中的 8 个数据帧进行类似计算，所得统计结果如表 3-10 所示。

VL4 和 *VL7* 到达交换机 SW3 之后将转发至端系统 ES8，首先对其进行排序。由于 *VL7* 的 BAG 小于 *VL4* 的 BAG，优先对 *VL7* 周期内各数据帧进行调度规划，通过使用交换机调度表设计方法可得 SW3—ES8 调度时刻，如图 3-17 所示。

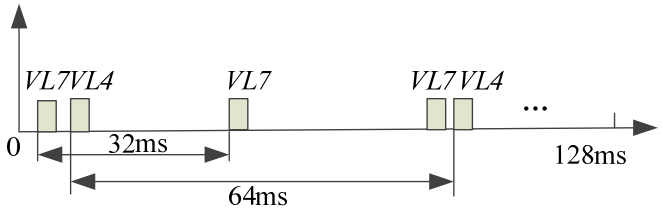

图 3-17 交换机 SW3 对虚链路 *VL4* 和 *VL7* 的时刻调度示意图

表 3-10　时间触发虚链路 *VL3*、*VL6* 和 *VL8* 在交换机 SW3 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{3,1}$	0.0762	$V_{8,2}$	66. 19908
$V_{3,2}$	32. 0762	$V_{11,1}$	0.18258
$V_{3,3}$	64. 0762	$V_{11,1}$	16.18258
$V_{3,4}$	96. 0762	$V_{11,1}$	32.18258
$V_{6,1}$	0.19908	$V_{11,1}$	48.18258
$V_{6,2}$	32. 19908	$V_{11,1}$	64.18258
$V_{6,3}$	64. 19908	$V_{11,1}$	80.18258
$V_{6,4}$	96. 19908	$V_{11,1}$	96.18258
$V_{8,1}$	2. 19908	$V_{11,1}$	112.18258

VL4 和 *VL7* 到达交换机 SW3 之后将转发至端系统 ES8，首先对其进行排序。由于 *VL7* 的 BAG 小于 *VL4* 的 BAG，优先对 *VL7* 周期内各数据帧进行调度规划，通过使用交换机调度表设计方法可得 SW3—ES8 调度时刻，如图 3-17 所示。

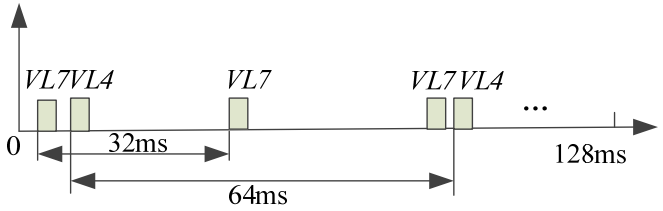

图 3-17 交换机 SW3 对虚链路 *VL4* 和 *VL7* 的时刻调度示意图

在交换机 SW3 中， *VL7* 第一次被调度的时刻，即 $V_{7,1}$ 的转发时刻为：

$$t_{7,1}=1+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+4\times\frac{L_{VL_7}\times 8\times 1000}{100\times 10^6}+2\times\left(+\tau_{filter}+\tau_{fw}+D_{propagation}\right)$$
$$=1+\frac{(28+4\times 256)\times 8\times 1000}{100\times 10^6}+2\times(0.016+0.0005)=1.11716ms$$

而 *VL4* 的时间接收窗与 *VL7* 重合，需要等待 *VL7* 完成数据帧传输后才能发送消息，所以 *VL7* 第一次被调度的时刻，即 $V_{7,1}$ 的转发时刻为：

$$t_{4,1}=1+\frac{L_{SYNC}\times 8\times 1000}{100\times 10^6}+4\times\frac{L_{VL_4}\times 8\times 1000}{100\times 10^6}+2\times\left(+\tau_{filter}+\tau_{fw}+D_{propagation}\right)+\frac{L_{VL_7}\times 8\times 1000}{100\times 10^6}$$
$$=1+\frac{(28+4\times 256)\times 8\times 1000}{100\times 10^6}+2\times(0.016+0.0005)+\frac{256\times 8\times 1000}{100\times 10^6}=1.13764ms$$

依次对 *VL7* 一个矩阵周期中的 4 个数据帧及 *VL4* 一个矩阵周期中的 2 个数据帧进行类似计算，所得统计结果如表 3-11 所示。

表 3-11 时间触发虚链路 *VL4* 和 *VL7* 在交换机 SW3 中的调度时刻统计表

TTVL 帧	调度时刻（ms）	TTVL 帧	调度时刻（ms）
$V_{7,1}$	1.11716	$V_{7,4}$	97.11716
$V_{7,2}$	33.11716	$V_{4,1}$	1.13764
$V_{7,3}$	65.11716	$V_{4,2}$	65.13764

由于 *VL4* 和 *VL7* 在交换机 SW3 中发生冲突，导致 *VL4* 的调度时刻产生了 0.02048ms 的附加延迟。

各 *VL* 数据帧在由发送端系统至目的端的时延计算过程中，需将在目的端系统的上一级调度时刻基础上增加一次数据帧时延 $\tau_{frame,k}$ 和一段链路传输时延 $D_{propagation}$，由本节中所得各个节点的调度时刻表能够计算出 TTAFDX 模型中各条 TTVL 的端到端时延，如表 3-12 所示。

由于 TTVL 数据流在配置的时域内独享整个带宽，且在传输路径中完全按照调度时刻表进行发送，无需进行排队等待，所以 TTVL 的端到端时延仅包含固定时延部分，也不存在时延抖动的现象[103]。表 3-12 中各 VL 的时延能够有效地证明以上结果。

表 3-12　时间触发虚链路端到端时延统计表

TTVL 帧	端到端时延（μs）	TTVL 帧	端到端时延（μs）
*VL*1	139.88	*VL*7	135.9
*VL*3	84.7	*VL*8	238.3
*VL*4	156.38	*VL*11	262.76
*VL*6	238.3		

3.6 TTAFDX 速率限制虚链路实时性分析

与普通 AFDX 所采用的 SP 调度算法相比，TTAFDX 网络中 TT 数据流具有高优先级，而 RC 数据流具有低优先级。当混合数据流到达交换机的某个输出端口时，交换机按照配置时间对 TT 数据流进行转发，TT 数据流在配置时域内独占整个带宽，不受 RC 数据流的影响。当 TT 数据流完成传输后，调度器会对所在端口等待发送的 RC 数据流按照数据帧进入缓存的顺序对 RC 数据帧依次进行操作。TTAFDX 交换机调度模型如图 3-18 所示。

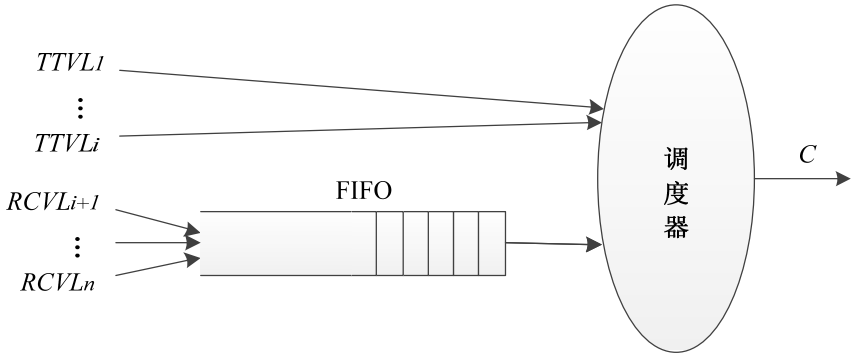

图 3-18 时间触发 AFDX 交换机调度模型

对于共享 TTAFDX 交换机同一输出端口的多个 VL，定义 $G^{TT}=\bigcup TTVLi$ 和 $G^{RC}=\bigcup RCVLi$ 分别代表 TTVL 和 RCVL 的聚合流。

由于 TT 数据流不会受到 RC 数据流的阻塞，因此输出端口提供给 TTVL 的服务曲线即交换机输出端口能够提供的总服务曲线，当其物理带宽为 C 时，其为：$\beta_{G^{TT}}(t)=\beta(t)=C[t-0]^{+}$，即 TTVL 聚合流 G^{TT} 的服务时延和服务速率参数分别为：

$$T_{G^{TT}}=0,\ R_{G^{TT}}=C \tag{3-95}$$

而对于 RCVL 聚合流 G^{RC} 的服务曲线为：

$$\beta_{G^{RC}}(t)=\left[\beta(t)-A^{TT}(t)\right]^{+} \tag{3-96}$$

假设 VL 的到达曲线为 $\alpha(t)=rt+b$，则 G^{TT} 的到达曲线 $A^{TT}(t)$ 为：

$$A^{TT}(t)=\sum_{\{i\in G^{TT}\}}\alpha_i(t)=\sum_{\{i\in G^{TT}\}}r_i\times t+\sum_{\{i\in G^{TT}\}}b_i \tag{3-97}$$

将公式（3-97）代入公式（3-96），得:

$$\beta_{G^{RC}}(t)=\left[Ct-\sum_{\{i\in G^{TT}\}}r_i\times t-\sum_{\{i\in G^{TT}\}}b_i\right]^+=\left[C-\sum_{\{i\in G^{TT}\}}r_i\right]\left[t-\frac{\sum_{\{i\in G^{TT}\}}b_i}{C-\sum_{\{i\in G^{TT}\}}r_i}\right]^+ \tag{3-98}$$

即 RCVL 聚合流 G^{RC} 的服务速率和服务时延参数分别为：

$$R_{G^{RC}}=C-\sum_{\{i\in G^{TT}\}}r_i \tag{3-99}$$

$$T_{G^{RC}}=\sum_{\{i\in G^{TT}\}}b_i\Big/R_{G^{RC}} \tag{3-100}$$

TTAFDX 网络对于聚合流 G^{RC} 仍沿用 AFDX 网络的 FIFO 调度策略接受交换机的服务，通过第 3.4 节相关推导可求得各 RCVL 的服务曲线为：

$$\begin{aligned}&\beta_{VL_i}(t)\\&=\left[R_{G^{RC}}(t-T_{G^{RC}})-\left[\sum_{\{j\in G^{RC}\}}r_j-r_i\right]\times\left[t-T_{G^{RC}}-\frac{\sum_{\{p(j)=L\}}b_j-b_i}{R_{G^{RC}}}\right]-\left[\sum_{\{j\in G^{RC}\}}b_j-b_i\right]\right]^+\\&=\left[R_{G^{RC}}-\sum_{\{j\in G^{RC}\}}r_j+r_i\right]\times\left[t-\frac{R_{G^{RC}}\times T_{G^{RC}}+\sum_{\{j\in G^{RC}\}}b_j-b_i}{R_{G^{RC}}}\right]^+\end{aligned} \tag{3-101}$$

则 VL*i* 的服务速率和服务时延参数分别为

$$R_{VL_i}=R_{G^{RC}}-\sum_{\{j\in G^{RC}\}}r_j+r \tag{3-102}$$

$$T_{VL_i}=T_{G^{RC}}+\left(\sum_{\{j\in G^{RC}\}}b_j-b_i\right)\Big/R_{G^{RC}} \tag{3-103}$$

由图 3-6 所示的 AFDX 网络模型图可知，*VL1*、*VL3*、*VL4*、*VL6*、*VL7*、*VL8* 和 *VL11* 都是 TTVL，其时延分析在 3.5 小节中已做了详细分析，现对各 RCVL 的时延进行计算。

SW1 为 *VL1*、*VL2* 和 *VL5* 提供向 ES6 转发服务的的聚合流服务曲线为 $\beta_{SW1-ES6}(t)=Ct$。则 SW1 向 *VL2* 和 *VL5* 聚合流提供的服务曲线为：

$$\beta_{G^{RC}}(t)=\left[Ct-\frac{L^1_{\max}}{BAG_1}t-L^1_{\max}\right]^+=\left[C-\frac{L^1_{\max}}{BAG_1}\right]\left[t-\frac{L^1_{\max}}{C-\frac{L^1_{\max}}{BAG_1}}\right]^+ \tag{3-104}$$

由公式（3-104）可得 *VL2* 和 *VL5* 所获得的服务曲线与公式（3-66）相同，且 *VL2* 和 *VL5* 的串联服务曲线与公式（3-67）一致。

同理可得 SW2 为 *VL9* 和 *VL10* 聚合流提供的服务曲线与公式（3-71），*VL9* 和 *VL10* 所获得的服务曲线与公式（3-72）相同，其离开 SW2 后对于 SW3 的到达曲线则与公式（3-73）一致。

经 SW3 转发至 ES7 的 TTVL 聚合流到达曲线 $\alpha_{SW3-ES7,G^{TT}}(t)$ 为：

$$\alpha_{SW3-ES7,G^{TT}}(t)=\sum_{i=3,6,8,11}\alpha_i(t)=t\times\sum_{i=3,6,8,11}\frac{L^i_{\max}}{BAG_i}+\sum_{i=3,6,8,11}L^i_{\max} \tag{3-105}$$

而 *VL4*、*VL7*、*VL10* 和 *VL12* 转发至 ES8，其在 SW3 输出端口的时间触发和速率限制聚合流到达曲线 $\alpha_{SW3-ES8,G^{TT}}(t)$ 和 $\alpha_{SW3-ES8,G^{RC}}(t)$ 分别为：

$$\alpha_{SW3-ES8,G^{TT}}(t)=\sum_{i=4,7}\alpha(t)=t\sum_{i=4,7}\frac{L^i_{\max}}{BAG_i}+\sum_{i=4,7}L^i_{\max} \tag{3-106}$$

$$\alpha_{SW3-ES8,G^{RC}}(t)=\alpha_{10}'(t)+\alpha_{12}(t)=t\sum_{i=10,12}\frac{L^i_{\max}}{BAG_i}+L_{RC}' \tag{3-107}$$

其中：$L_{RC}'=\sum_{i=10,12}L^i_{\max}+\dfrac{\sum_{i=6,7,8,9}L^i_{\max}}{C-\sum_{j=6}^{8}\dfrac{L^j_{\max}}{BAG_j}}\dfrac{L^{10}_{\max}}{BAG_{10}}$。

由 SW3 向 ES7 转发的低优先级 VL 仅有 *VL9*，其所获得的服务曲线为：

$$\beta_{SW3-ES7,9}(t)=\left[C-\sum_{i=3,6,8,11}\frac{L^i_{\max}}{BAG_i}\right]\left(t-\frac{\sum_{i=3,6,8,11}L^i_{\max}}{C-\sum_{i=3,6,8,11}\dfrac{L^i_{\max}}{BAG_i}}\right)^{+} \tag{3-108}$$

对于由 SW3 向 ES8 转发的速率限制虚链路 *VL10* 和 *VL12* 所获得的服务曲线分别为：

$$\beta_{SW3-ES8,i}(t)=\left[C-\sum_{j=4,7}\frac{L^j_{\max}}{BAG_j}-\left(\sum_{j=10,12}\frac{L^j_{\max}}{BAG_j}-\frac{L^i_{\max}}{BAG_i}\right)\right]\left(t-\frac{\sum_{j=4,7}L^j_{\max}+L_{RC}'-L^i_{\max}'}{C-\sum_{j=4,7}\dfrac{L^j_{\max}}{BAG_j}}\right)^{+} \tag{3-109}$$

其中，$L^{12}_{\max}{}'=L^{12}_{\max}$，

$$L^{10}_{\max}{}'=L^{10}_{\max}+\frac{\sum_{i=6}^{9}L^i_{\max}}{C-\sum_{j=6,7,8}\dfrac{L^j_{\max}}{BAG_j}}\times\frac{L^{10}_{\max}}{BAG_{10}} \tag{3-110}$$

根据以上推导，*VL2*、*VL5* 和 *VL9* 的串联服务曲线与 3.4 小节中的非抢占 SP 调度算法相同。虚链路 *VL10* 的串联服务曲线为：

$$\beta_i^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW2,10}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau\otimes\beta_{SW3-ES8,10}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\min\{\beta_{SW2,10}(t),\beta_{SW3-ES8,10}(t)\}$$
$$\times\left[t-\left(\frac{\sum_{j=6}^{8}L_{\max}^{j}+\sum_{j=9,10}L_{\max}^{j}-L_{\max}^{10}}{C-\sum_{j=6}^{8}\frac{L_{\max}^{j}}{BAG_j}}+\frac{\sum_{j=4,7}L_{\max}^{j}+L_{RC}{}'-L_{\max}^{10}{}'}{C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}}+3\tau+2\left(\eta+\frac{L_{\max}^{i}}{C}\right)\right)\right]^{+}\quad(3\text{-}111)$$

虚链路 *VL12* 的串联服务曲线为：

$$\beta_{12}^{\Sigma}(t)=\delta_\tau\otimes\beta_{SW3-ES8,12}(t)\otimes\delta_\eta\otimes\delta_{rv}\otimes\delta_\tau$$
$$=\left[C-\sum_{j=4,7,10}\frac{L_{\max}^{j}}{BAG_j}\right]\times\left[t-\left(\frac{\sum_{j=4,7}L_{\max}^{j}+L_{RC}{}'-L_{\max}^{12}{}'}{C-\sum_{j=4,7}\frac{L_{\max}^{j}}{BAG_j}}+2\tau+\eta+L_{\max}^{12}/C\right)\right]^{+}\quad(3\text{-}112)$$

根据表 3-2 中所设定的各虚链路配置参数，经过计算，得到各 RCVL 的时延上界，如表 3-13。

表 3-13　TTAFDX 中速率限制虚链路端到端时延上界统计表

TTVL 帧	时延上界（μs）	TTVL 帧	时延上界（μs）
*VL*2	201.74	*VL*10	243.8
*VL*5	324.78	*VL*12	119.48
*VL*9	373.3		

3.7 本章小结

本章根据 TTAFDX 网络的特性和需求，设计完成了 TTVL 调度表结构，并提出了周期优先和帧长度优先两种调度表设计方法。构建 TTAFDX 网络拓扑模型，对数据帧在网络传输过程中的端到端时延进行了详细分析和研究，并确定了 TTAFDX 网络端到端时延主要有固定时延和非固定时延组成。以采用周期优先调度表设计方法为例，通过对时间触发虚链路在各网络节点的发送时间进行计算，得出其非固定时延部分在一般均为 0，确保了时间确定性和实时性。采用网络演算方法，分别对普通 AFDX 所支持的“先到先服务”和 2 级“静态优先级”调度算法进行了时延分析。通过与 TTAFDX 中的时间触发虚链路和速率限制虚链路时延进行比较，能够发现时间触发虚链路能够支持硬实时特性的数据传输，同时速率限制虚链路的时延也能够有所降低。

4　DIMA 网络的速率限制虚链路调度算法研究与实时性分析

时间触发 AFDX 在保证时间触发虚链路数据传输时延的前提下，同样需要对速率限制虚链路的调度策略进行优化。AFDX 标准协议对速率限制虚链路采用先来先服务的调度策略，按照数据流到达交换机的顺序来转发相应的数据包，在保证公平性的同时，容易引起数据包的传输超时，无法满足不同数据流的端到端时延要求。

由于基于时间戳（Time-stamp）的分组调度算法在确保良好时延性、公平性的同时，算法复杂度较高，不适用于时间触发 AFDX 高速分组网络。本章采用轮询调度的基本思想，为速率限制虚链路设计了逐次最小定额轮询（Successive Minimal-Quantum Round Robin, SMQRR）调度算法。SMQRR 调度算法通过改变对每一个数据流权值所对应的数据量的连续服务方式[128]，在保留轮询调度算法 $O(1)$ 时间复杂度的同时，有效克服了分组长度不同带来的不公平性并避免了数据流可能长时间无法获得服务的情况。通过理论分析和仿真验证，SMQRR 调度算法在公平性和时延上界上均优于加权可变轮询调度算法（Weighted Elastic Round Robin，WERR）和加权差额轮询调度算法（Deficit Weighted Round Robin, DWRR）。

4.1 基于 SMQRR 的速率限制虚链路调度算法

SMQRR 调度算法包括初始化、数据帧入队排队和数据帧出队调度三个过程，如图 4-1 所示。

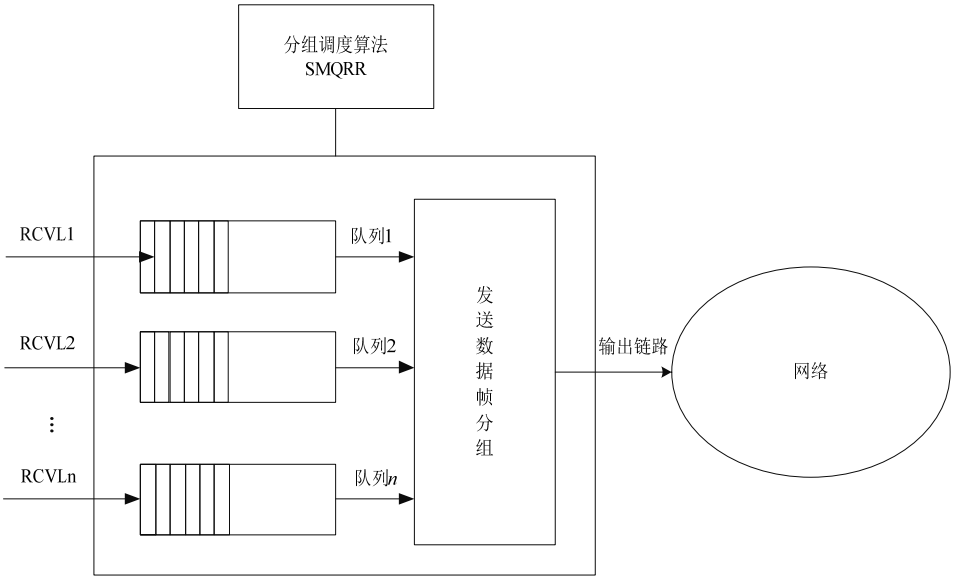

图 4-1 SMQRR 算法分组调度示意图

调度模块当 TTAFDX 交换机输出端口有数据帧调度需求时进行初始化过程。每个速率限制虚链路数据流对应一个队列，数据帧入队排队过程是指当一个新的数据帧到达时，加入到与它所属速率限制虚链路数据流对应的队列中排队并等候发送[129]。数据帧

出队调度是 SMQRR 算法的核心内容，描述了如何从不同队列中选择数据帧，并将所选取的数据帧通过输出链路发送到 TTAFDX 网络中的过程。分组调度模块通过使用 SMQRR 调度算法，能够提高输出链路带宽资源分配的公平性，减少拥塞，降低数据帧丢失率。

定义 4.1 活动（active）数据流是指该数据流在一定时间间隔内至少有一个分组在等待服务或正在接受服务，其活动期（active period）为该数据流持续活动的最大时间间隔[130]。

定义 4.2 当一个数据流 i 在一个最大的时间间隔 (τ_1, τ_2) 内保持以预留服务速率接受服务，则称此时间间隔为数据流 i 的一个忙期（busy period）。

SMQRR 利用速率限制虚链路号唯一标识数据流。对于任一传输速率为 r 的输出链路，假设由 n 条 RCVL 所共享，记 RCVL_i 的预留服务速率为 ρ_i, $\rho_i = L_{\max}/BAG, (i=1,...,n)$，则有 $\sum_{i=1}^{n}\rho_i \le r$。SMQRR 调度器通过为每条数据流分配一个权值（Weight），以保证其所接收的服务与其预留服务速率成比例，以满足数据流优先级、QoS 要求等特性。分配给数据流 i 的权值记为 ω_i，其值由公式（4-1）定义为：

$$\omega_i = \frac{\rho_i}{\rho_{\min}} \tag{4-1}$$

其中 $\rho_{\min}$ 表示最小预留服务速率。显然对任意的 $1 \le i \le n$，都有 $\omega_i \ge 1$。

SMQRR 调度器为每条活动的数据流 i 分配一个定额值（Quantum）Q_i、一个剩余定额值（UnservedQuantum）UQ_i 和一个差额值（Deficit Count）DC_i。其中 Q_i 表示在一轮外循环服务机会中数据流 i 最多可发送的比特位数量，该定额值与其预留服务速率成比例。具有最小预留服务速率的数据流所分配的定额值为 $Q_{\min}$，则有 $Q_i = \omega_i \times Q_{\min}$。$UQ_i$ 记录在当前外循环中分配给数据流 i 的定额值 Q_i 中未完成服务的数量。在一些内循环服务机会中，由于定额值的限制，数据流 i 可能无法完成一个数据帧的发送，使用 DC_i 记录当前数据流可用服务量和实际服务量的差值，并将其累加到下一轮内循环服务机会中，当一轮外循环中最后一轮内循环结束后，DC_i 将累加至下一轮外循环中。因此，在特定服务机会中获得较少服务的数据流将在下一轮服务中得到补偿，获得较多的服务量。

4.1.2 SMQRR 算法描述

SMQRR 调度算法维持两个队列：一个数据流“活动列表（ActiveList）”和一个数据流“剩余列表（SurplusList）”。ActiveList 用于存放所有活动数据流，当一个非活动数据流变为活动时，将该数据流加入到 ActiveList 的尾部[129]，而当一个活动数据流变为非活动时，该数据流将从 ActiveList 中移除[130]。

当一个数据流的剩余定额值 UQ 大于 0 时，其所分配的服务定额值没有用完。将系统中所有剩余定额值 UQ 大于 0 的数据流存放到一个链表中，称该链表为数据流“剩余列表（SurplusList）”。当一个数据流的剩余定额值 UQ 等于 0 时，其数据流所分配的服

务定额值已用完，则将数据流从“剩余列表”中删除。如果一个数据流属于“剩余列表”，那它一定属于“活动列表”，但反之不然。

SMQRR 算法将 DWRR 调度算法中的一次轮询细分为两层循环，每一轮外循环（Outer Round）包含若干个内循环（Inner Round），并在每一轮内循环中执行类似 DRR 算法的调度。一轮外循环是指“活动列表”中包含的所有数据流被分组调度模块访问服务的过程。一轮内循环则是指“剩余列表”中所包含的数据流被分组调度模块访问服务的过程。在一轮外循环的执行过程中，重新成为活动的或新到达的数据流将加入到“活动列表”的尾部，并将从下一轮外循环开始接受服务。例如在第一轮外循环开始时刻 T1，“活动列表”中包含两个数据流 1 和 2，则分组调度模块在第一轮外循环中将仅对数据流 1 和数据流 2 进行访问。在第一轮外循环结束时刻 T2（T1<T2）前，假设数据流 3 变为活动的并加入到了“活动列表”，由于其在第一轮外循环开始时刻 T1 并不存在于“活动列表”，分组调度模块将不会访问它。当第二轮外循环在 T2 时刻开始后，“活动列表”中的数据流将依次被分组调度模块访问，此轮循环将包括数据流 3。另外，所说的外循环轮次是相对于某个数据流来讲的，如从 T2 开始的外循环，对于数据流 1 和 2 来说是第 2 个轮次，但对于数据流 3 来说，则是第 1 个轮次。

通过引入“循环访问计数器（RoundRobinVisitCount）”记录每一轮内循环中所需要访问数据流的数目，即内循环开始时 SurplusList 中数据流的数量。每当 SMQRR 调度器结束一个数据流的访问，RoundRobinVisitCount 就减少 1，当 RoundRobinVisitCount 最终等于 0 时，则该内循环结束。

SMQRR 分组调度算法的实现主要包括三个过程，如图 4-2 所示：

1）初始化（Initialize）过程：初始化数据流“活动列表”，“剩余列表” 和“循环访问计数器”，并为系统中每个数据流分配一个初始权值。

2）分组入队（Enqueue）排队过程：将新到达的分组加入其所属数据流对应的队列，若该数据流不在“活动列表”中，则在数据流“活动列表”的尾部加入该分组所属的数据流。

3）分组出队（Dequeue）调度过程：依据 SMQRR 调度算法，从数据流“活动列表”中选取此轮循环将服务的数据量，按照列表中的顺序，在各数据流中选取相应数量的分组，并通过输出链路发送。

由于 SMQRR 调度算法将各活动数据流所分配的权值细分为多轮内循环进行发送，每一个活动数据流在每轮内循环中都将获得最小剩余定额值的服务机会，所以在保证数据流发送数量的同时，缩短了接受服务的间隙，降低了等待时延。SMQRR 算法的分组调度流程如图 4-3 所示。

```
Initialize : (Invoked when the scheduler is initialized)
    RoundRobinVisitCount=0
    ActiveList=NULL  SurplusList=NULL
    for (i = 1; i ≤ n; i = i + 1) {
        DC_i = 0
        Assigning a Quantum Q_i to each flow i }
Enqueue : (Invoked when a packet arrives)
    i=QueueInWhichPacketArrives
    if (ExistInActiveList(i)==FALSE) then
        AddQueueToActiveList(i)
        Increment SizeOfActiveList
        DC_i = 0; UQ_i=Q_i
    end if
Dequeue :
    while (TRUE) do {
        if (ActiveListIsEmpty==FALSE) then
            SurplusList=ActiveList
        else
            Waiting for a packet arrival
        end if
        while (SurplusListIsEmpty == FALSE) do{
            RoundRobinVisitCount=SizeOfSurplusList
            UQ_min =MinimumOfUnservedQuantum
            while (RoundRobinVisitCount>0) do{
                i=HeadOfSurplusFlowList
                RemoveHeadOfSurplusFlowList()
                DC_i = DC_i + UQ_min ; UQ_i=UQ_i − UQ_min
                do{
                    TransmitPacketFromQueue(i)
                    DC_i = DC_i − LengthInBitsOfTransmittedPacket;}
                while ((QueueIsEmpty==FALSE)
                    & & (LengthInBitsOfPacketAtHead(Queue_i) ≤ DC_i)
                if (QueueIsEmpty==TRUE) then
                    DC_i = 0
                    RemoveQueueFromActiveList(i)
                else
                    if ((UQ_i+DC_i <LengthInBitsOfPacketAtHead(Queue_i)) then
                        DC_i = DC_i + UQ_i ; UQ_i=Q_i
                    else
                        AddQueueToSurplusList(i)
                    end if
                end if
                Decrement RoundRobinVisitCount
            end while
        end while
    end while
```

图 4-2 SMQRR 分组调度算法实现

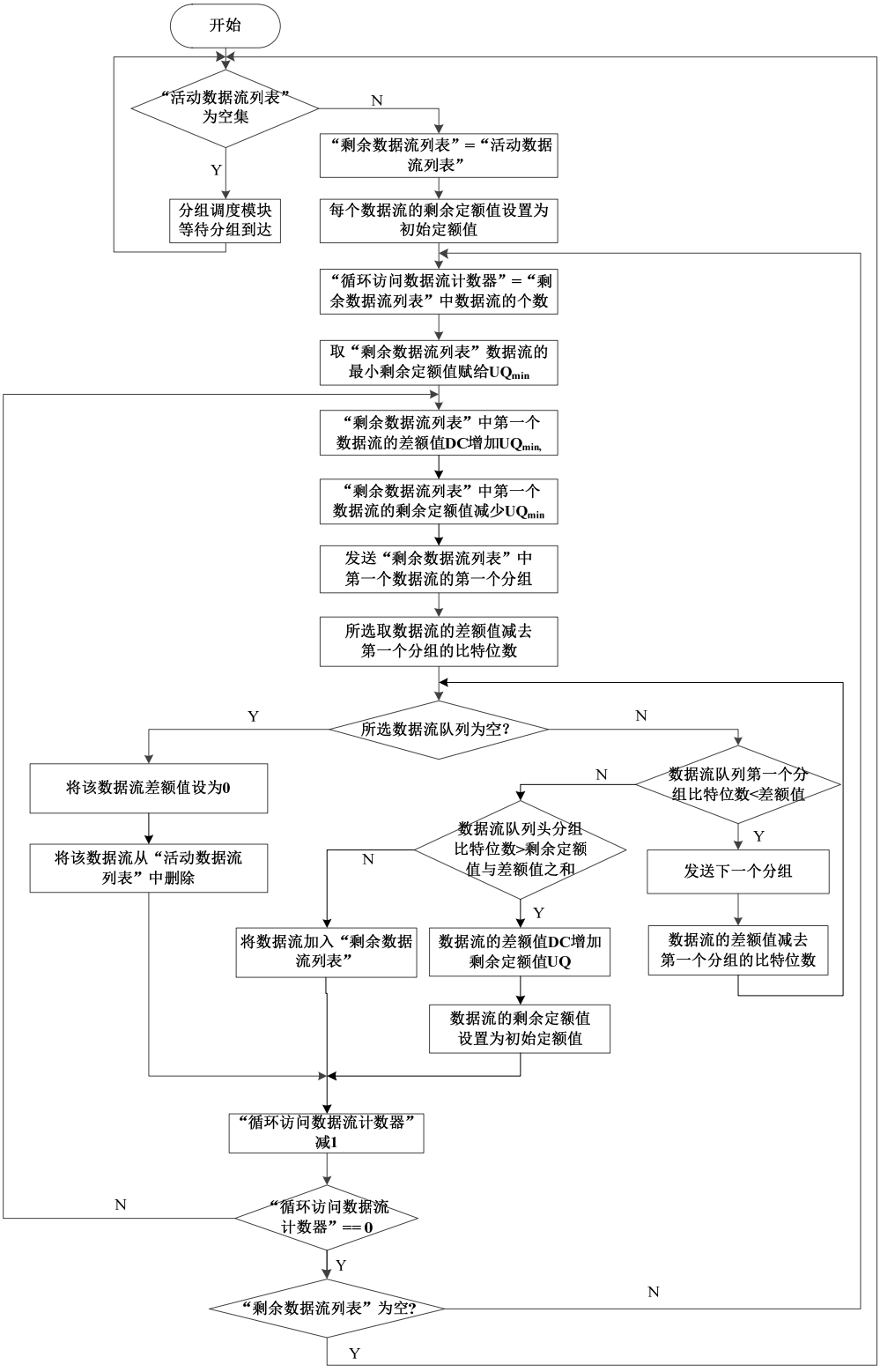

图 4-3 SMQRR 分组调度流程图

4.1.3 举 例

设有四条速率限制虚链路 RCVL1，RCVL2，RCVL3 和 RCVL4，它们的权值分别为 $\omega_{RCVL1}=4$，$\omega_{RCVL2}=2$，$\omega_{RCVL1}=3.5$ 和 $\omega_{RCVL1}=1$，其中最大数据帧长度为 20。通过使

用示意图对比说明 SMQRR 和 DWRR 算法的调度过程，如图 4-4、图 4-5 所示，其中一轮循环中每条数据流允许发送的数据量在图中均由矩形的长度表示。

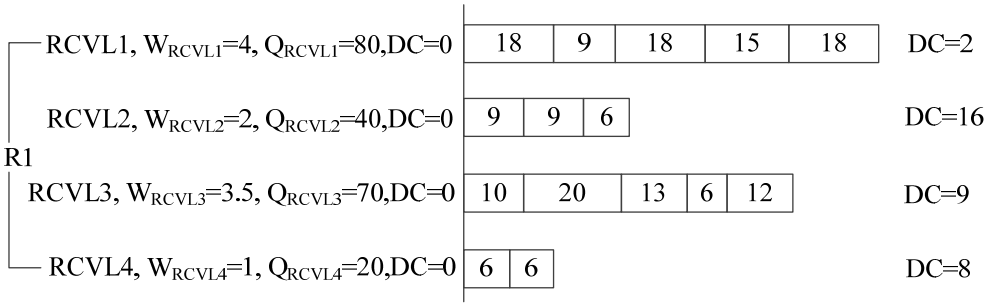

图 4-4 DWRR 分组调度示意图

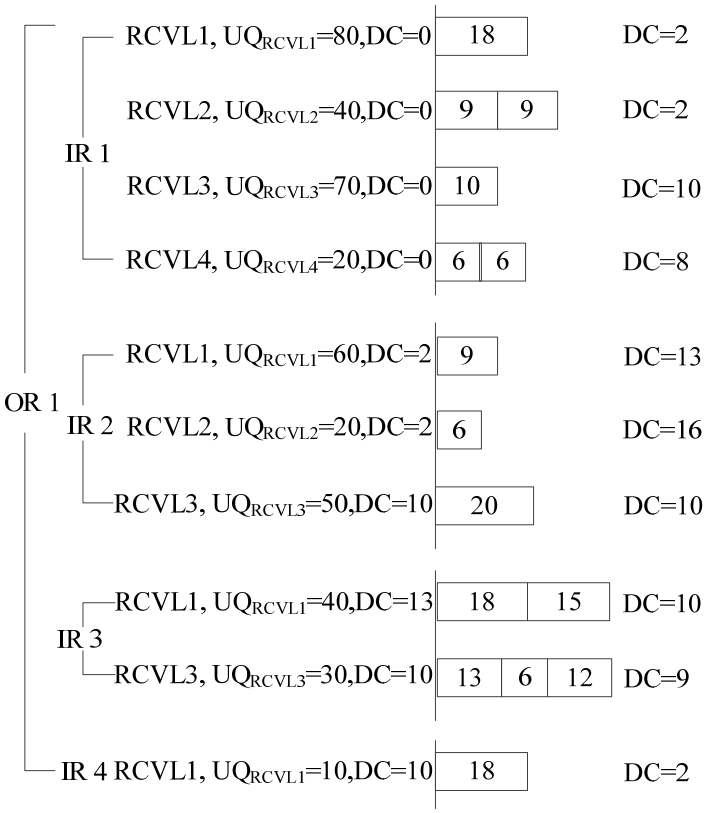

图 4-5 SMQRR 分组调度示意图

在图 4-5 所示的一轮外循环调度初始时刻，四条速率限制虚链路均为活动的，则有 $\text{ActiveList=SurplusList=}\{\text{RCVL1, RCVL2, RCVL3, RCVL4}\}$。在第 1 轮内循环中，由于最小剩余定额值 $UQ_{\min}=UQ_4$，所以在此轮内循环中，四个数据流发送的数据量均为 UQ_4。RCVL4 的剩余定额值在第 1 轮内循环结束时变为 0，从数据流“剩余列表”中删除，其余数据流的剩余定额值分别为 $UQ_{RCVL1}-UQ_{RCVL4}$，$UQ_{RCVL2}-UQ_{RCVL4}$，$UQ_{RCVL3}-UQ_{RCVL4}$。

在第 2 轮内循环开始时，比较数据流“剩余列表” $S2=\{\text{RCVL1, RCVL2, RCVL3}\}$ 中各数据流的最小剩余定额值，可得 $UQ_{\min}=UQ_2-UQ_4$。因此，在此轮内循环中“剩余列表”内的三个数据流发送的数据量均为 UQ_2-UQ_4。RCVL2 的剩余定额值在第 2 轮内循

环结束时变为 0，从数据流“剩余列表”中删除，其余数据流的剩余定额值分别为 $UQ_{RCVL1}-UQ_{RCVL2}$，$UQ_{RCVL3}-UQ_{RCVL2}$。

在第 3 轮内循环开始时，比较数据流“剩余列表” $S3=\{RCVL1, RCVL3\}$ 中各数据流的最小剩余定额值，可得 $UQ_{\min}=UQ_3-UQ_2$。因此，在此轮内循环中“剩余列表”内的两个数据流发送的数据量均为 UQ_3-UQ_2。RCVL3 的剩余定额值在第 3 轮内循环结束时变为 0，从“剩余数据流列表”中删除，RCVL4 的剩余定额值为 $UQ_{RCVL1}-UQ_{RCVL3}$。

在第 4 轮内循环开始时，数据流“剩余列表”仅剩下 $S4=\{RCVL1\}$ 一个数据流，因此，在此轮内循环中 RCVL1 将发送 $UQ_{RCVL1}-UQ_{RCVL3}$ 的数据量。第 4 轮内循环结束时，RCVL1 的剩余定额值在第 4 轮内循环结束时变为 0，从“剩余数据流列表”中删除。“剩余数据流列表”此时变为空集，说明所有轮次内循环结束，也意味着由以上四轮内循环构成的外循环也已经结束。

由图 4-5 所示，一轮外循环包含四轮内循环的执行，并且，当有两个或多个数据流所分配定额值相等时，则内循环的轮数就一定小于数据流“活动列表”中数据流的个数。当所有数据流的定额值相等时，一轮外循环中仅包含一轮内循环，即外循环本身。

4.2 SMQRR 时延分析

本节将根据 Stiliadis 和 Verma 定义的 LR 服务器（Latency-Rate，LR Servers）[131] 理论对 SMQRR 算法的时延特性进行分析。

分析过程中使用的符号定义如下：

$\mathrm{OR}(k)$：第 k 轮外循环；

$\mathrm{IR}(k,v)$：第 k 轮外循环中的第 v 轮内循环；

$\mathrm{ODC}_i(k)$：数据流 i 在完成第 k 轮外循环后的差额值；

$\mathrm{IDC}_i(k,v)$：数据流 i 在完成第 k 轮外循环中第 v 轮内循环后的差额值；

$\mathrm{Sent}_i(t_1, t_2)$：在时间间隔 (t_1, t_2) 内数据流 i 获得的总服务量；

$\mathrm{Sent}_i(k)$：数据流 i 在第 k 轮外循环获得的服务量；

$\mathrm{Sent}_i(k,v)$：数据流 i 在第 k 轮外循环中第 v 轮内循环中获得的服务量；

$S_i(\alpha_i,t)$：数据流 i 在时间间隔 (α_i, t) 内获得的服务量，其中数据流 i 在时间 α_i 开始一个忙期，且在 $(\alpha_i, t), t>\alpha_i$ 内处于连续忙期；

$\tau_i^{(k,v)}$：数据流 i 开始获得第 k 轮外循环中第 v 轮内循环 (k,v) 服务的时间；

m：调度过程中已发送的最大数据帧长度；

M：未来可能到达的最大数据帧长度；

W：由 SMQRR 调度器提供服务的所有活动数据流权值总和。

由 SMQRR 的调度算法描述中可知，在第 k 轮外循环结束时，数据流 i 满足：

$$Sent_i(k)=ODC_i(k-1)+Q_i-ODC_i(k) \tag{4-2}$$

因为 $Q_i=\omega_i\times Q_{\min}$，所以上列公式可写为：

$$Sent_i(k)=\omega_i Q_{\min}+ODC_i(k-1)-ODC_i(k) \tag{4-3}$$

在不失一般性的情况下，假设 n 条数据流的权值互不相等，对 n 个不同权值由小到大进行排列，则 ω_{v-1} 和 ω_v 分别表示排在第 $v-1$ 位和第 v 位的权值。在 SMQRR 的内循环 (k,v) 结束时，数据流 i 满足：

$$\begin{aligned} Sent_i(k,v)&=(Q_v-Q_{v-1})+IDC_i(k,v-1)-IDC_i(k,v)\\ &=(\omega_v-\omega_{v-1})Q_{\min}+IDC_i(k,v-1)-IDC_i(k,v)\end{aligned} \tag{4-4}$$

定义 4.3 一个数据流 i 的时延定义为一个最小的非负常数 Θ_i，对数据流 i 的所有可能的忙期均满足[130]：

$$S_i(\alpha_i,\ t)\geq\max\{0,\rho_i(t-\alpha_i-\Theta_i)\} \tag{4-5}$$

对于一个调度器如果存在非负常数 Θ_i 使其满足公式（4-5），那么这个调度器就属于 LR 服务器。

引理 4.1 设调度器 S 上的数据流 i 在时刻 τ_i 成为活动的，且在时间间隔 $(\tau_i,\ t),\ t>\tau_i$ 内连续活动，若存在最小非负数 $\Theta_i^{'}$ 满足：

$$Sent_i(\tau_i,\ t)\geq\max\{0,\rho_i(t-\tau_i-\Theta_i^{'})\} \tag{4-6}$$

，则由公式（4-5）所定义的时延不会超过 $\Theta_i^{'}$。

证明：由定义 4.1 和 4.2 可知忙期一定是活动期，所以数据流 i 在活动期 $(\tau_i,\ t)$ 中的任意一个忙期 $(\alpha_k,\ \beta_k)$ 如果都满足公式(4-5)即可得证。在 $\alpha_k=\tau_i,\ t\in(\alpha_k,\beta_k)$ 的情况下，有 $S_i(\alpha_k,\ t)=\mathrm{Sent}_i(\alpha_k,t)\geq\max\{0,\rho_i(t-\alpha_k-\Theta_i^{'})\}$；而在 $\alpha_k>\tau_i,\ t\in(\alpha_k,\beta_k)$ 的情况下，有 $S_i(\alpha_k,\ t)=\mathrm{Sent}_i(\tau_k,t)-\mathrm{Sent}_i(\tau_i,\alpha_k)$。由于数据流 i 在时间间隔 $(\tau_i,\ \alpha_k)$ 内不是忙期，所以 $Sent_i(\tau_i,\alpha_k)\leq\rho_i(\alpha_k-\tau_i)$，于是 $S_i(\alpha_k,\ t)>\max\{0,\rho_i(t-\tau_i-\Theta_i^{'})\}-\rho_i(\alpha_k-\tau_i)$ $=\max\{0,\rho_i(t-\alpha_k-\Theta_i^{'})\}$。由此可知，调度器 S 属于 LR 服务器且其时延小于或等于 $\Theta_i^{'}$。

定义 4.4 集合 T 是调度器结束一个数据流服务并开始另一个数据流服务的时间集合。集合 T_i 是数据流 i 开始接受调度器服务的所有时间集合。集合 T 包含了所有活动数据流 i 的集合 T_i。

引理 4.2 数据流 i 在活动期 $(\tau_i,\ t)$ 经历的时延达到其上界 $\Theta_i^{'}$，仅当数据流 i 变成活动的时刻 τ_i 是某个数据流服务的开始，且活动期结束时间 t 属于集合 T_i。

假设数据流 i 在时刻 τ_i 变成活动的，依据引理 4.2，确定 SMQRR 的时延上界只需要在所有的时间间隔中选择一个合适的时间间隔 $(\tau_i,\tau_i^{(k,v)})$，且这个时间间隔的跨度尽可能是最大的。

数据流 i 开始活动的时刻 τ_i 既可以是一轮外循环的开始，也可以不是。假设在 τ_i 时刻第 k_0 轮外循环正好开始或正在执行，第 (k_0+h) 轮外循环的开始时刻为 t_h，当 τ_i 与第 k_0 轮的开始时间 t_0 不一致时，如 $\tau_i>t_0$，时间间隔 (t_0,τ_i) 就不在所考察的时间间隔中；而当 τ_i 与 t_0 一致时，时间间隔 $(\tau_i,\tau_i^{(k,v)})$ 具有最大的时间跨度。当数据流 i 在每一轮内循环中作为最后接受服务的数据流，并在每一个外轮循环中的第 i 轮内循环完成接受这个外

循环为其分配的服务总量时，时间间隔的跨度最大，时延也最大。设(k,v)等于(k_0+s,i)，则考察时间间隔如图 4-6 所示。

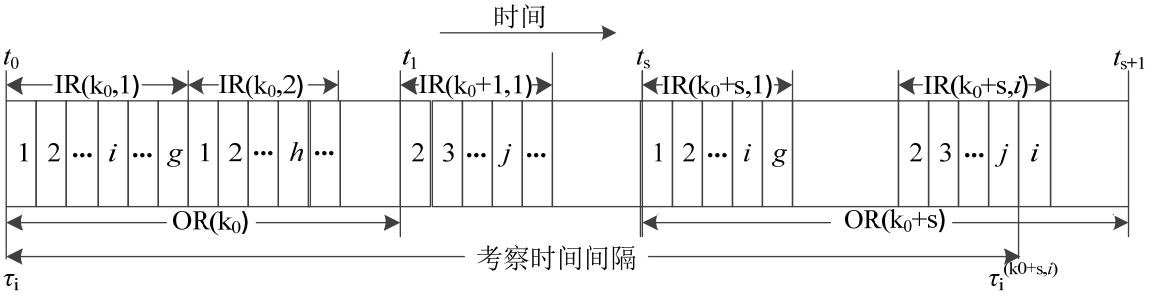

图 4-6 SMQRR 考察时间间隔示意图

将考察时间间隔$(\tau_i,\tau_i^{(k_0+s,i)})$分为两个子时间间隔分别进行分析：

1）(τ_i,t_s)：在这一子时间段，所有的 n 条数据流均经历了 s 个完整的外循环服务。设第(k_0+h)轮外循环的执行时间为(t_h,t_{h+1})，根据公式（4-2），n 条数据流在第(k_0+h)轮外循环内接受服务的时间为：

$$t_{h+1}-t_h=\frac{W}{r}Q_{\min}+\frac{1}{r}\sum_{j=1}^{n}\left\{ODC_j\left(k_0+h-1\right)-ODC_j\left(k_0+h\right)\right\} \tag{4-7}$$

从第k_0轮外循环开始，对 s 轮外循环求和有：

$$t_s-\tau_i=\frac{W}{r}sQ_{\min}+\frac{1}{r}\sum_{j=1}^{n}\left\{ODC_j\left(k_0-1\right)-ODC_j\left(k_0+s-1\right)\right\} \tag{4-8}$$

2）$(t_s,\tau_i^{(k_0+s,i)})$：这段时间包含第(k_0+s)轮外循环中数据流 i 在开始接受第 i 轮内循环服务之前的那段时间，只有部分数据流完成了第(k_0+s)轮外循环的服务。因此，在这个时间段内，将对三类数据流分别进行分析：即 G_{FIN} 中的数据流、数据流 i 以及剩余数据流。

对于数据流集合 G_{FIN} 来说，所有的数据流均完成了一轮完整的外循环，其所接受的服务总量可表示为：

$$\sum_{j\in G_{FIN}}Sent_j(t_s,\tau_i^{(k_0+s,i)})=\sum_{j\in G_{FIN}}\left\{\omega_jQ_{\min}+ODC_j\left(k_0+s-1\right)-ODC_j\left(k_0+s\right)\right\} \tag{4-9}$$

设第(k_0+s,m)轮内循环的执行时间为$(t^{(k_0+s,m-1)},t^{(k_0+s,m)})$，其中 $m<i$，根据公式(4-4)，数据流 i 在第(k_0+s,m)轮内循环内接受的服务量为：

$$\begin{aligned}Sent_i(k_0+s,m)&=(Q_m-Q_{m-1})+IDC_i(k_0+s,m-1)-IDC_i(k_0+s,m)\\&=(\omega_m-\omega_{m-1})Q_{\min}+IDC_i(k_0+s,m-1)-IDC_i(k_0+s,m)\end{aligned} \tag{4-10}$$

当$m=1$时，公式（4-10）中$\mathrm{IDC}_i(k_0+s,m-1)$即为第$(k_0+s-1)$轮外循环后数据流 i 所得的差额值。

依照对考察时间的设定，数据流 i 在时间间隔$(t_s,\tau_i^{(k_0+s,i)})$中仅接受前$i-1$个内循环的服务，其所接受的服务总量表示为：

$$Sent_i(t_s,\tau_i^{(k_0+s,i)})=\omega_{i-1}Q_{\min}+ODC_i\left(k_0+s-1\right)-IDC_i\left(k_0+s,i-1\right) \tag{4-11}$$

对于剩余数据流，其权值均大于ω_i，在时间间隔$(t_s,\tau_i^{(k_0+s,i)})$中，数据流接受的是前i轮内循环的服务，所接受的服务总量可表示为：

$$\sum_{j\notin G_{FIN},j\neq i} Sent_j(t_s,\tau_i^{(k_0+s,i)}) = \sum_{j\notin G_{FIN},j\neq i}\left\{\omega_i Q_{\min} + ODC_j\left(k_0+s-1\right) - IDC_j\left(k_0+s,i\right)\right\} \quad (4\text{-}12)$$

综合上述三类数据流在时间间隔$(\tau_i,\tau_i^{(k_0+s,i)})$中接受的服务量，可以得出：

$$\begin{aligned}\tau_i^{(k_0+s,i)} - t_s = {} & \frac{W}{r}sQ_{\min} + \frac{1}{r}\sum_{j=1}^{n}\left\{ODC_j\left(k_0-1\right) - ODC_j\left(k_0+s-1\right)\right\} \\ & + \frac{1}{r}\sum_{j\in G_{FIN}}\left\{\omega_j Q_{\min} + ODC_j\left(k_0+s-1\right) - ODC_j\left(k_0+s\right)\right\} \\ & + \frac{1}{r}\left\{\sum_{j\notin G_{FIN},j\neq i}\left\{\omega_i Q_{\min} + ODC_j\left(k_0+s-1\right) - IDC_j\left(k_0+s,i\right)\right\}\right\} \\ & + \frac{1}{r}\left\{\omega_{i-1}Q_{\min} + ODC_i\left(k_0+s-1\right) - IDC_i\left(k_0+s,i-1\right)\right\}\end{aligned} \quad (4\text{-}13)$$

引理4.3 在SMQRR算法中，对于$\forall i\in n$，当外循环$\mathrm{OR}(k)$结束时，有$0\le \mathrm{ODC}_i(k)<m$；当内循环$\mathrm{IR}(k,v)$结束时，有$0\le \mathrm{IDC}_i(k,v)<m$。

证明：ODC_i值仅在一轮外循环结束后才进行更新，并在初始时$\mathrm{ODC}_i(k)\ge 0$。SMQRR 算法中结束一轮外循环的条件为$\mathrm{UQ}_i+\mathrm{IDC}_i$小于将要发送的数据包，此时有$ODC_i = IDC_i + UQ_i < m$，可得$0\le \mathrm{ODC}_i(k)<m$。

IDC_i值仅在一轮内循环结束后才进行更新且等于刚结束的内循环的差额值。在SMQRR 算法中结束一轮内循环的条件为IDC_i小于要发送的数据包，因此可得出：$0\le IDC_i(k,v)<m$。

由于数据流i在外循环k_0才开始活动，即$ODC_i\left(k_0-1\right)=0$。对公式（4-13）进行合并相关项，并依据引理4.3可推出：

$$\begin{aligned}\tau_i^{(k_0+s,i)} - t_s \le {} & \frac{W}{r}sQ_{\min} + \frac{1}{r}(n-1)(m-1) \\ & + \frac{1}{r}\left(\sum_{j\in G_{FIN}}\omega_j Q_{\min} + \omega_{i-1}Q_{\min} + \sum_{j\notin G_{FIN},j\neq i}\omega_i Q_{\min}\right) \\ & - \frac{1}{r}\left\{IDC_i\left(k_0+s,i-1\right)\right\}\end{aligned} \quad (4\text{-}14)$$

由公式（4-14）可导出s的表达式：

$$\begin{aligned}s \ge {} & \frac{r}{WQ_{\min}}\left(\tau_i^{(k_0+s,i)} - t_s\right) - \frac{1}{WQ_{\min}}(n-1)(m-1) \\ & - \frac{1}{W}\left(\sum_{j\in G_{FIN}}\omega_j + \omega_{i-1} + \sum_{j\notin G_{FIN},j\neq i}\omega_i\right) + \frac{1}{WQ_{\min}}\left\{IDC_i\left(k_0+s,i-1\right)\right\}\end{aligned} \quad (4\text{-}15)$$

在考察时间间隔$(\tau_i,\tau_i^{(k_0+s,i)})$中，数据流$i$接受的服务量可表示为：

$$Sent_i(\tau_i,\tau_i^{(k_0+s,i)}) = Sent_i(\tau_i,t_s) + Sent_i(t_s,\tau_i^{(k_0+s,i)}) \quad (4\text{-}16)$$

其中$\mathrm{Sent}_i(t_s,\tau_i^{(k_0+s,i)})$等于$\mathrm{Sent}_i(s,i-1)$，$\mathrm{Sent}_i(\tau_i,t_s)$等于 s 轮外循环中数据流 i 传输的数据量，得出：

$$\begin{aligned}&Sent_i(\tau_i,\tau_i^{(k_0+s,i)})=\\&\omega_i(sQ_{\min})-ODC_i\left(k_0+s-1\right)+\omega_{i-1}Q_{\min}+ODC_i\left(k_0+s-1\right)-IDC_i\left(k_0+s,i-1\right)\\&\qquad=\omega_i(sQ_{\min})+\omega_{i-1}Q_{\min}-IDC_i\left(k_0+s,i-1\right)\end{aligned}\tag{4-17}$$

将公式（4-15）的 s 代入公式（4-17），得出：

$$\begin{aligned}Sent_i(\tau_i,\tau_i^{(k_0+s,i)})\ge&\frac{\omega_i r}{W}\left(\tau_i^{(k_0+s,i)}-\tau_i\right)-\frac{\omega_i}{W}(n-1)(m-1)\\&-\frac{\omega_i Q_{\min}}{W}\left(\sum_{j\in G_{FIN}}\omega_j+\omega_{i-1}+\sum_{j\notin G_{FIN},j\neq i}\omega_i\right)\\&+\frac{\omega_i-W}{W}\left\{IDC_i\left(k_0+s,i-1\right)\right\}+\omega_{i-1}Q_{\min}\end{aligned}\tag{4-18}$$

由于所有数据流的预留速度之和不得大于输出链路的最大传输速率，因此：

$$\rho_i\le\frac{\omega_i r}{W}\tag{4-19}$$

将公式（4-19）代入公式（4-18）得出：

$$\begin{aligned}&Sent_i(\tau_i,\tau_i^{(k_0+s,i)})\\&\ge\rho_i\left\{\left(\tau_i^{(k_0+s,i)}-\tau_i\right)-\left\{\begin{array}{l}\dfrac{(n-1)(m-1)}{r}+\dfrac{Q_{\min}}{r}\left(\displaystyle\sum_{j\in G_{FIN}}\omega_j+\sum_{j\notin G_{FIN},j\neq i}\omega_i\right)\\+\left(\dfrac{1}{\rho_i}-\dfrac{1}{r}\right)\left\{IDC_i\left(k_0+s,i-1\right)\right\}\\+\left(\dfrac{1}{r}-\dfrac{1}{\rho_i}\right)\omega_{i-1}Q_{\min}\end{array}\right\}\right\}\end{aligned}\tag{4-20}$$

在公式（4-20）中，当$i=1$时，说明ω_i是所有活动数据流权值中最小的，则$\omega_{i-1}=0$。如果$i>1$，则有$\omega_{i-1}\ge1$。引入参数Ψ，其取值情况如下：当ω_i为 n 个活动数据流权值中的最小值时，取$\Psi=0$；其它情况，取$\Psi=1$。可得出:

$$\begin{aligned}&Sent_i(\tau_i,\tau_i^{(k_0+s,i)})\\&\ge\rho_i\left\{\left(\tau_i^{(k_0+s,i)}-\tau_i\right)-\left\{\begin{array}{l}\dfrac{(n-1)(m-1)}{r}+\dfrac{Q_{\min}}{r}\left(\displaystyle\sum_{j\in G_{FIN}}\omega_j+\sum_{j\notin G_{FIN},j\neq i}\omega_i\right)\\+\left(\dfrac{1}{\rho_i}-\dfrac{1}{r}\right)\left\{IDC_i\left(k_0+s,i-1\right)\right\}\\+\Psi\left(\dfrac{1}{r}-\dfrac{1}{\rho_i}\right)\omega_{i-1}Q_{\min}\end{array}\right\}\right\}\end{aligned}\tag{4-21}$$

当取 $\mathrm{IDC}_i\left(k_0+s,i-1\right)$ 的值为 $m-1$ 时，时间延迟可达到上界，且公式（4-21）可进一步简化得出：

$$\begin{aligned} & Sent_i(\tau_i,\tau_i^{(k_0+s,i)}) \\ & \geq \rho_i\left\{\left(\tau_i^{(k_0+s,i)}-\tau_i\right)-\left\{\begin{array}{l}\dfrac{(n-1)(m-1)}{r}+\dfrac{Q_{\min}}{r}\left(\sum\limits_{j\in G_{FIN}}\omega_j+\sum\limits_{j\notin G_{FIN},j\neq i}\omega_i\right) \\ +\left(\dfrac{1}{\rho_i}-\dfrac{1}{r}\right)(m-1) \\ +\Psi\left(\dfrac{1}{r}-\dfrac{1}{\rho_i}\right)\omega_{i-1}Q_{\min}\end{array}\right\}\right\} \end{aligned} \tag{4-22}$$

即：

$$\Theta_i^{'}\leq\left\{\begin{array}{l}\dfrac{(n-1)(m-1)}{r}+\dfrac{Q_{\min}}{r}\left(\sum\limits_{j\in G_{FIN}}\omega_j+\sum\limits_{j\notin G_{FIN},j\neq i}\omega_i\right) \\ +\left(\dfrac{1}{\rho_i}-\dfrac{1}{r}\right)(m-1) \\ +\Psi\left(\dfrac{1}{r}-\dfrac{1}{\rho_i}\right)\omega_{i-1}Q_{\min}\end{array}\right\} \tag{4-23}$$

由引理 4.1 可得运行 SMQRR 调度算法的调度器属于 LR 服务器，且数据流 *i* 的时延 Θ_i 上界如公式（4-23）所示，其中，当 ω_i 为 *n* 个权值中的最小值时 $\Psi=0$；其他情况 $\Psi=1$。

4.3 时延上界对比分析

Salil S. Kanhere 推导出了 DWRR 调度算法的时延上界 Θ_i^{DWRR} [132]：

$$\Theta_i^{DWRR}=\frac{(W-w_i)M+(m-1)(n-1)}{r}+(m-1)\left(\frac{1}{\rho_i}-\frac{1}{r}\right) \tag{4-24}$$

由于 $m-1<M=Q_{\min}$，比较 SMQRR 和 DWRR 调度算法的延迟，可得到：

$$\begin{aligned} \Theta_i^{DWRR}-\Theta_i &\geq \frac{Q_{\min}}{r}\left(W-\omega_i-\sum_{j\in G_{FIN}}\omega_j-\sum_{j\notin G_{FIN},j\neq i}\omega_i\right)+\Psi\left(\frac{1}{\rho_i}-\frac{1}{r}\right)\omega_{i-1}Q_{\min} \\ &=\frac{Q_{\min}}{r}\left(\sum_{j\notin G_{FIN},j\neq i}(\omega_j-\omega_i)\right)+\Psi\left(\frac{1}{\rho_i}-\frac{1}{r}\right)\omega_{i-1}Q_{\min} \end{aligned} \tag{4-25}$$

分别对两个边界条件下的时延比较情况进行分析。

条件 1：对任意 $j\in n, j\neq i, \rho_i \ll \rho_j$。

在这种情况下，$\omega_v=\omega_i=1$ 且 $\omega_i \ll \omega_j$，参数 $\Psi=0$，G_{FIN} 为空集，且根据引理 4.3，可得：

$$\Theta_i^{DWRR}-\Theta_i\geq\frac{(m-1)}{r}\left(\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-\omega_i\right)\right)=\frac{(m-1)}{r}\left(\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-1\right)\right)\gg 0 \quad (4\text{-}26)$$

条件 2：对任意 $j\in n, j\neq i, \rho_i\gg\rho_j$。

在这种情况下，$\omega_i\gg\omega_j$，参数 $\Psi=1$，G_{FIN} 包含了除数据流 i 之外的 n-1 个数据流，由公式（4-25）可得出：

$$\begin{aligned}\Theta_i^{DWRR}-\Theta_i&\geq\frac{(m-1)}{r}\left(W-\omega_i-\sum_{j\in G_{FIN}}\omega_j-\sum_{j\notin G_{FIN},j\neq i}\omega_i\right)+\left(\frac{1}{\rho_i}-\frac{1}{r}\right)\omega_{i-1}Q_{\min}\\&=\frac{Q_{\min}}{r}\left(\sum_{\varnothing}\left(\omega_j-\omega_v\right)\right)+\left(\frac{1}{\rho_i}-\frac{1}{r}\right)\omega_{i-1}Q_{\min}\\&=\left(\frac{1}{\rho_i}-\frac{1}{r}\right)\omega_{i-1}Q_{\min}\gg 0\end{aligned} \quad (4\text{-}27)$$

综合以上在边界条件下对 SMQRR 和 DWRR 调度算法的时延分析，可知 SMQRR 调度算法的延迟边界优于 DWRR 调度算法。

Salil S. Kanhere 等人对 DWRR 调度算法进一步优化，并提出了 WERR 调度算法[42]，其时延上界 Θ_i^{WERR} 如公式（4-28）：

$$\Theta_i^{WDRR}=\frac{(n-1)(m-1)+\left(W-\omega_i\right)m}{r} \quad (4\text{-}28)$$

同理，比较 SMQRR 和 WERR 调度算法的延迟可得到：

$$\Theta_i^{WERR}-\Theta_i\geq\left\{\begin{array}{l}\dfrac{(m-1)}{r}\left(\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-\omega_i\right)\right)+\dfrac{1}{r}\left(\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-\omega_i\right)\right)\\-\left(\dfrac{1}{\rho_i}-\dfrac{1}{r}\right)(m-1)\left\{1-\Psi\omega_{i-1}\right\}\end{array}\right\} \quad (4\text{-}29)$$

当 $\omega_i\neq 1$ 时，参数 $\Psi=1$ 且 $\omega_{i-1}>1$，可得公式（4-29）大于 0，即 SMQRR 调度算法的延迟边界优于 WERR 调度算法。

当 $\omega_i=1$ 时，参数 $\Psi=0$，G_{FIN} 为空集，且根据引理 4.3，可得出：

$$\Theta_i^{WERR}-\Theta_i\geq\left\{\frac{(m-1)}{r}\sum_{j\notin G_{FIN},j\neq i}\omega_j+\frac{1}{r}\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-1\right)-\frac{1}{\rho_i}(m-1)\right\} \quad (4\text{-}30)$$

仅当 $\sum_{j\notin G_{FIN},j\neq i}\left(\omega_j-1\right)\geq m-1$ 时，SMQRR 调度算法的延迟边界优于 WERR 调度算法。

4.4 SMQRR 公平性

对于公平性的分析，采用了 Golestani[54]提出的一种基于相对公平性度量的公平性评价方法。通过对任意时间间隔 (t_1, t_2) 内，具有连续分组排队的任意数据流 i 和 j 计算公平性系数 $FM(t_1, t_2)=\left|Sent_i(t_1,t_2)/\rho_i-Sent_j(t_1,t_2)/\rho_j\right|$ 以此衡量调度算法的公平性，并记 $FM(t_1, t_2)$ 的最大值为 FM。当 FM 的值越小，则该调度算法的公平性越好。

当服务器 S 采用 SMQRR 调度算法时，设任意两个数据流 i 和 j 在时间间隔 (t_1, t_2) 内具有连续分组排队。由于数据流 i 和 j 之间最大相差一个内循环的服务量，不失一般性，设在时间间隔 (t_1, t_2) 内两个数据流均接受了 k 轮外循环的服务，且数据流 i 和 j 分别接受了 v 轮和 $v-1$ 轮内循环的服务，则有：

$$Sent_i(t_1,t_2)=k(\omega_i Q_{\min})+\omega_v Q_{\min}+\mathrm{ODC}_i(k)-\mathrm{IDC}_i(k+1,v) \tag{4-31}$$

$$Sent_j(t_1,t_2)=k(\omega_j Q_{\min})+\omega_{v-1} Q_{\min}+\mathrm{ODC}_j(k)-\mathrm{IDC}_j(k+1,v-1) \tag{4-32}$$

由公式（4-31）和（4-32）可得：

$$\left|\frac{Sent_i(t_1,t_2)}{\rho_i}-\frac{Sent_j(t_1,t_2)}{\rho_j}\right|=\left|\begin{aligned}&\frac{k(\omega_i Q_{\min})}{\rho_i}-\frac{k(\omega_j Q_{\min})}{\rho_j}+\frac{\omega_v Q_{\min}}{\rho_i}-\frac{\omega_{v-1} Q_{\min}}{\rho_j}\\&+\frac{ODC_i(k)}{\rho_i}-\frac{ODC_j(k)}{\rho_j}\\&+\frac{IDC_j(k+1,v-1)}{\rho_j}-\frac{IDC_i(k+1,v)}{\rho_i}\end{aligned}\right| \tag{4-33}$$

由公式（4-1）可得 $\dfrac{\omega_i}{\rho_i}=\dfrac{\omega_j}{\rho_j}$，所以

$$\left|\frac{Sent_i(t_1,t_2)}{\rho_i}-\frac{Sent_j(t_1,t_2)}{\rho_j}\right|=\left|\begin{aligned}&\frac{\omega_v Q_{\min}}{\rho_i}-\frac{\omega_{v-1} Q_{\min}}{\rho_j}+\frac{ODC_i(k)}{\rho_i}-\frac{ODC_j(k)}{\rho_j}\\&+\frac{IDC_j(k+1,v-1)}{\rho_j}-\frac{IDC_i(k+1,v)}{\rho_i}\end{aligned}\right| \tag{4-34}$$

对内循环轮次 v 分下列情况进行讨论：

当 $v=1$ 时，

$$\begin{aligned}\left|\frac{Sent_i(t_1,t_2)}{\rho_i}-\frac{Sent_j(t_1,t_2)}{\rho_j}\right|&=\left|\begin{aligned}&\frac{\omega_v Q_{\min}}{\rho_i}+\frac{ODC_i(k)}{\rho_i}-\frac{ODC_j(k)}{\rho_j}\\&+\frac{ODC_j(k)}{\rho_j}-\frac{IDC_i(k+1,v)}{\rho_i}\end{aligned}\right|\\&\le\frac{\omega_{\min} Q_{\min}}{\rho_i}+\frac{ODC_i(k)-IDC_i(k+1,v)}{\rho_i}\\&\le\frac{Q_{\min}}{\rho_{\min}}+\frac{m-1}{\rho_{\min}}\le\frac{2Q_{\min}}{\rho_{\min}}\end{aligned} \tag{4-35}$$

当 $v>1$ 时，如果 $\dfrac{\omega_v Q_{\min}}{\rho_i}\ge\dfrac{\omega_{v-1} Q_{\min}}{\rho_j}$，则：

$$\left|\frac{Sent_i(t_1,t_2)}{\rho_i}-\frac{Sent_j(t_1,t_2)}{\rho_j}\right|\le\left|\begin{array}{l}\dfrac{\omega_v Q_{\min}}{\rho_v}-\dfrac{\omega_{v-1}Q_{\min}}{\rho_j}+\dfrac{ODC_i(k)-IDC_i(k+1,v)}{\rho_i}\\ -\dfrac{ODC_j(k)-IDC_j(k+1,v-1)}{\rho_j}\end{array}\right| \tag{4-36}$$
$$\le\left|\frac{Q_{\min}}{\rho_{\min}}+\frac{Q_{\min}}{\rho_i}-\frac{-Q_{\min}}{\rho_j}\right|\le\frac{3Q_{\min}}{\rho_{\min}}$$

如果$\frac{\omega_v Q_{\min}}{\rho_i}\le\frac{\omega_{v-1}Q_{\min}}{\rho_j}$，则：

$$\left|\frac{Sent_i(t_1,t_2)}{\rho_i}-\frac{Sent_j(t_1,t_2)}{\rho_j}\right|\le\frac{\omega_{v-1}Q_{\min}}{\rho_{v-1}}-\frac{\omega_v Q_{\min}}{\rho_i}+\left|\begin{array}{l}\dfrac{ODC_i(k)-IDC_i(k+1,v)}{\rho_i}\\ -\dfrac{ODC_j(k)-IDC_j(k+1,v-1)}{\rho_j}\end{array}\right| \tag{4-37}$$
$$\le\frac{Q_{\min}}{\rho_{\min}}+\left|\frac{Q_{\min}}{\rho_i}-\frac{-Q_{\min}}{\rho_j}\right|\le\frac{3Q_{\min}}{\rho_{\min}}$$

根据公式（4-35）、（4-36）和（4-37），可得出 SMQRR 调度算法的$FM\le 3Q_{\min}/\rho_{\min}$，即 SMQRR 具有一定的公平性。

4.5 SMQRR 实现复杂度

除了时延特性、公平性，调度算法在实际应用中的效率也受实现复杂度的影响。航空电子系统对于网络传输速度的要求迅速提高，为了适用于时间触发 AFDX 网络，SMQRR 调度算法的实现复杂度需要进行衡量。

定理 4.1 运行 SMQRR 调度算法的调度器的实现复杂度为$O(1)$。

证明：当一个新的数据分组到达时，SMQRR 调度算法实现的入队排队过程包含以下 2 个操作：1）将该数据分组加入所属数据流对应的队列，操作时间复杂度为$O(1)$；2）若该数据分组所属数据流尚未存在于数据流“活动列表”中，在数据流“活动列表”的尾部增加该数据流项。本方案通过对每一数据流项设置标识位而非采用遍历的方式进行查询，操作时间复杂度也为$O(1)$。

SMQRR 调度算法对于一个数据分组出队的调度过程包含以下操作：从数据流“剩余列表”的首部移出下一个要访问的数据流，当该数据流完成本轮内循环数据分组发送后，若数据流非空，则将其加入到数据流“剩余列表”的尾部，反之，则将其从数据流“剩余列表”中删除。以上对于链表的操作时间复杂度均为$O(1)$。除此之外，还包括数据流活动状态、接受服务状态及数据流相关变量（活动数据流计数器，剩余定额值）的更新等操作，且完成时间均为常数。因此，SMQRR 调度一个数据分组出队的时间复杂度为$O(1)$。

根据定理 4.1 得证，运行 SMQRR 调度算法的调度器的实现复杂度不依赖于活动数据流的数量。

4.6 仿真平台搭建及实验结果

4.6.1 仿真平台简介

NS2[133]， 即 Network Simulator Version 2，是由 UC Berkely 大学开发的一个采用面向对象技术的、开源的、基于离散事件驱动的网络环境模拟器。

NS2 体系结构按照功能划分为编译层和解释层，实现了数据操作和仿真控制的分离，保证了程序执行与开发的效率，如图 4-7 所示。编译层使用 C++开发实现对数据包的相关处理，包括事件调度器和基本的网络业务模型、传输协议、路由协议、队列模型等。解释层是指 NS2 在前端为用户提供的一个 OTcl 解释器，以方便使用者能够方便灵活地对仿真过程中的参数进行配置。Tclcl 模块起到了编译层与解释层间桥梁的作用，将编译类与解释层中的对象和变量相连接。

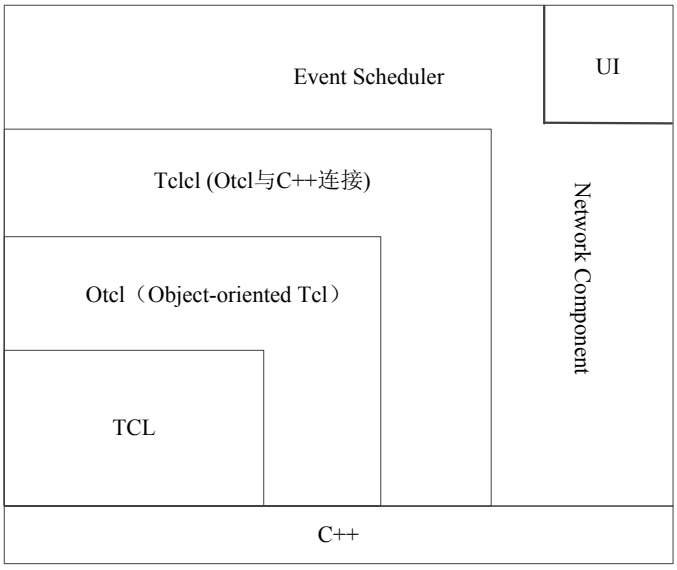

图 4-7 NS2 体系结构图

4.6.2 仿真网络搭建

为实现对 DWRR、WERR 及本章提出的 SMQRR 调度算法性能的比较，在 Fedora 操作系统中，搭建了 NS2 仿真平台。本文采用的 NS2 软件版本为 2.35。

由于 NS2 中并没有实现 SMQRR、DWRR 以及 WERR 调度算法，在仿真比较过程中首先需要对 NS2 的队列模型类库进行扩展。使用 C++分别编写 SMQRR.cc、DWRR.cc、WERR.cc 程序和对应的 SMQRR.h、DWRR.h、WERR.h 文件以实现相应的队列调度算法，并通过编译及修改 Tclcl 源码向 OTcl 脚本提供可使用的对象接口。按照仿真设计，编写 OTcl 脚本构建网络拓扑结构，选择业务模型、配置网络参数并设定仿真时间等参数。最终通过运行 OTcl 脚本完成仿真实验。由于 AFDX 采用类似 UDP 的传输协议，在

经过流量整形后具有 CBR 的流量特性，本文在仿真过程中对各链路均配置为 UDP 传输协议和 CBR 业务模型，并分别采用 SMQRR、DWRR 和 WERR 调度算法进行仿真比较。模拟过程如图 4-8 所示。

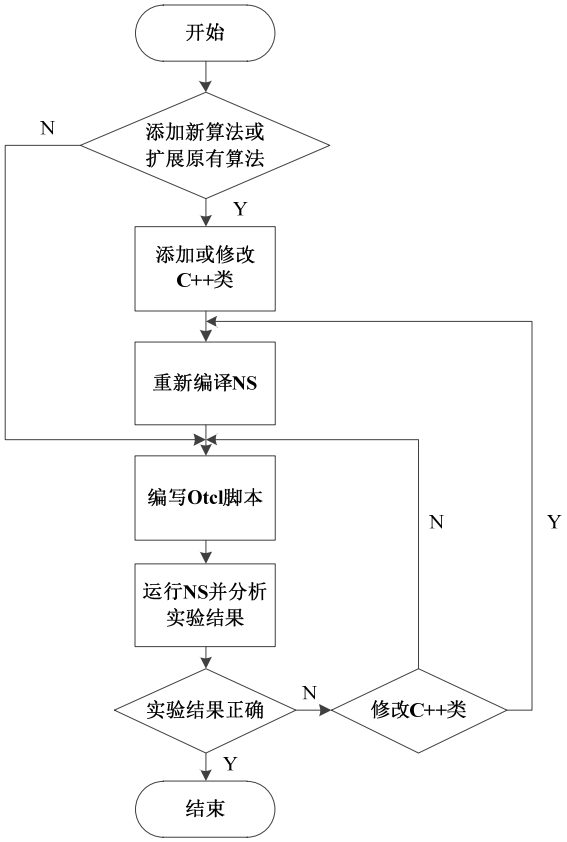

图 4-8 NS2 仿真过程示意图

4.6.3 仿真实验设计

本节使用 OTcl 语言定义了由 1 台交换机组成网络核心并互联 5 个端系统的仿真网络拓扑结构，如图 4-9 所示。其中，源端系统 0、1、2 和 3 均经过交换机 4 存储转发，最后到达指定的目的端系统 5。每个端系统所承载的数据流均为速率限制虚链路数据流，其虚链路配置信息如表 4-1 所示。交换机分别依次采用 DWRR、WERR 和 SMQRR 调度算法并在相同系统参数下进行仿真实验。若当数据包发送速率大于交换机处理速率时，交换机默认对数据包采取丢弃处理。

表 4-1　虚链路配置表

虚链路 ID	*L*max(Bytes)	BAG(ms)	类型
VL1	50	0.4	速率限制
VL2	100	0.4	速率限制
VL3	100	0.6	速率限制
VL4	200	0.4	速率限制

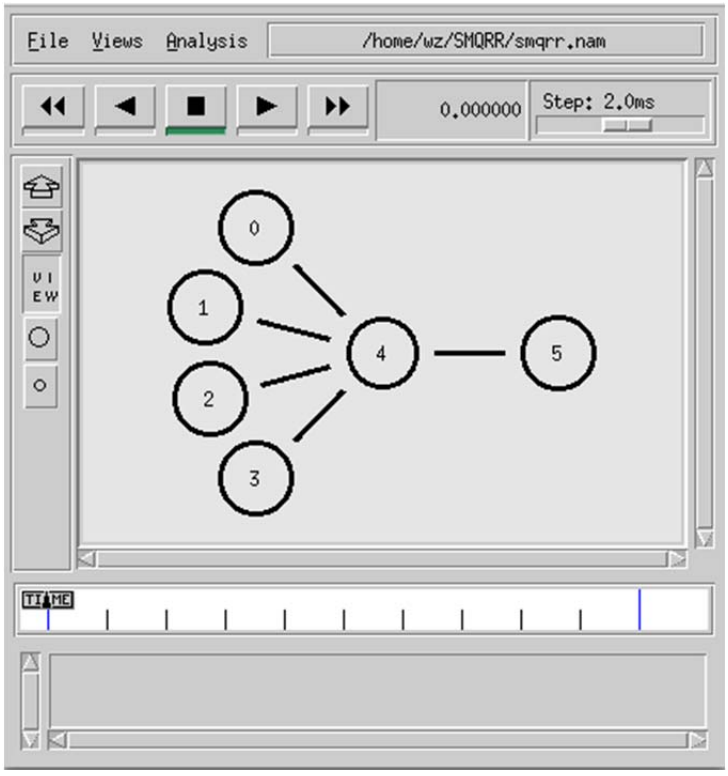

图 4-9 网络拓扑结构

4.6.4 仿真实验结果分析

通过在所编写的 OTcl 仿真脚本程序中加入代码，将仿真过程中每个数据包在任何时刻的状态记录到指定文件中，以实现对网络流量的有效跟踪，从而能够对所构建的网络模拟环境进行全面分析。NS2 为每次仿真过程提供了两个记录文件，分别为.nam 文件和.tr 文件。

仿真过程结束后，通过使用 NS2 提供的 Nam 观察器能够将记录在.nam 文件中的流量数据信息用视觉化的方式动态模拟整个仿真过程，使用者则可以在 Nam 窗口中方便地查看数据包如何从源端向目的端进行传送。图 4-10 为交换机采用 SMQRR 调度算法时的 Nam 仿真结果示意图。

另外，通过在 OTcl 脚本程序中指定 Trace 跟踪对象，能够根据用户的需要记录仿真过程中的任何一个细节或特定类型的事件，并将所得信息记录在.tr 文件中。trace 文件作为唯一记录仿真过程的文件，可以通过使用 awk 语言程序计算衡量算法性能所需要的网络参数，如：端到端的延迟时间、延迟时间变化、丢包率和吞吐量等。同时，NS2 还向用户提供了 XGraph、Gnuplot 等绘图工具以便于更加直观地显示所提取和对比所提取的网络参数。

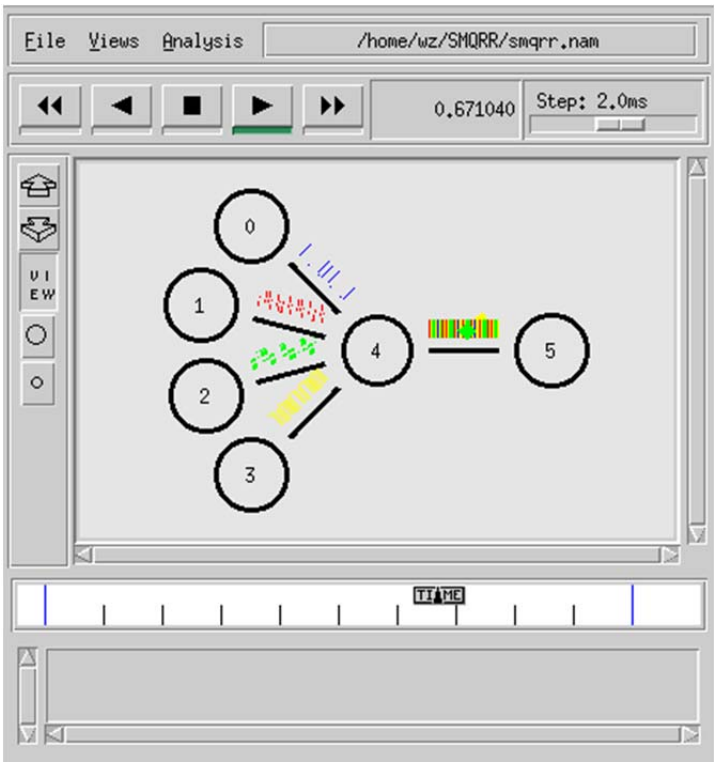

图 4-10 SMQRR 调度算法 Nam 仿真结果

当交换机分别采用 SMQRR、DWRR 和 WERR 调度算法完成仿真实验后，通过编写 awk 程序对端系统 1 和端系统 2 所产生的 trace 记录文件使用进行处理和计算，以比较各调度算法的端到端时延特性。图 4-11 和图 4-12 显示了通过 Gnuplot 绘图工具画出的分别在端系统 1 和端系统 2 产生的比较结果。

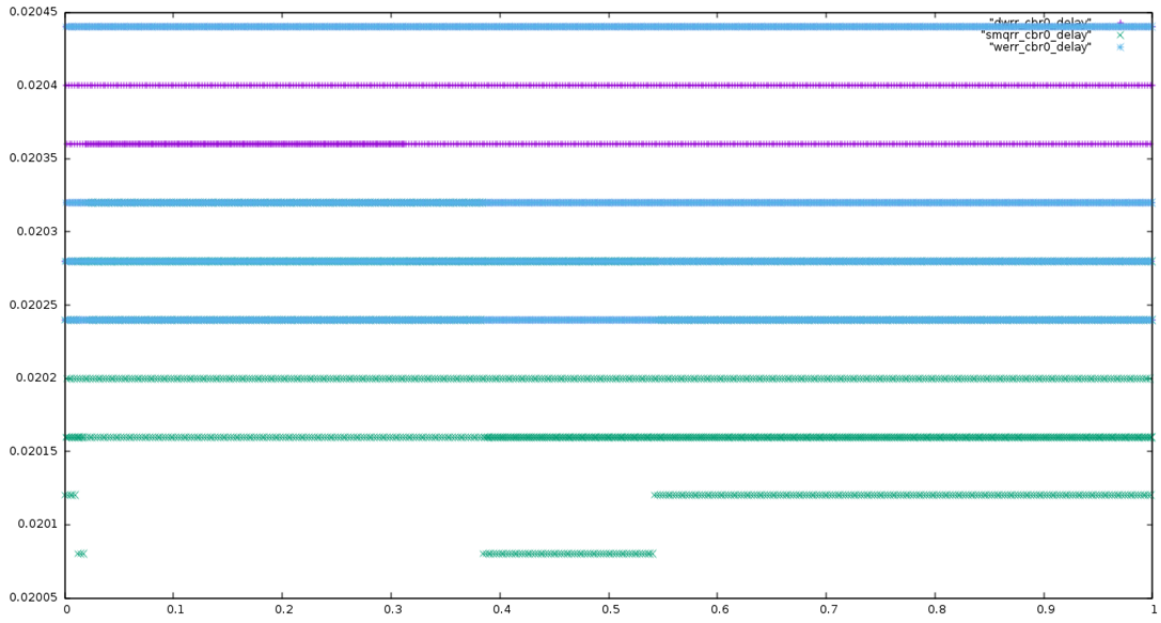

图 4-11 不同调度算法的虚链路 1 端到端延时比较

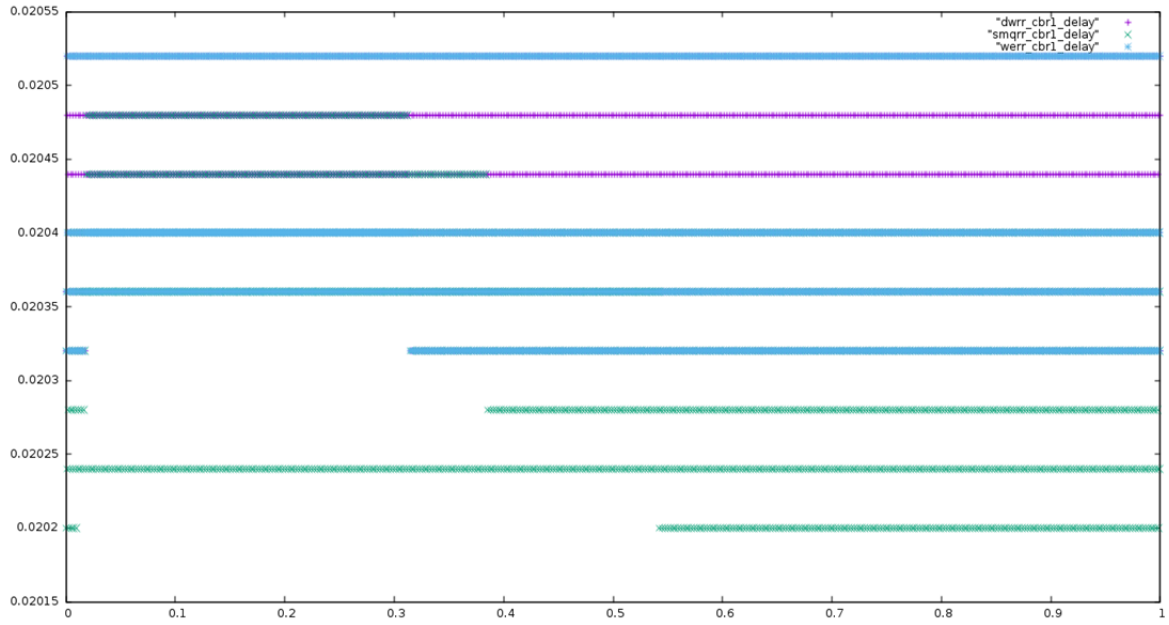

图 4-12 不同调度算法的虚链路 2 端到端延时比较

由图 4-12 和图 4-13 可得，SMQRR 调度算法较 DWRR 和 WERR 调度算法，能够有效地降低虚链路数据流的时延。进一步使用 awk 语言编写代码，分别对 trace 文件中记录的各数据流所产生的仿真过程信息进行处理和计算，可得到各数据流传输的平均时延和最大时延，如表 4-2 所示。

表 4-2　各端系统在不同调度算法情况下的平均时延和最大时延统计

	DWRR	WERR	SMQRR
VL1ave	0.020344	0.020320	0.020213
VL1max	0.020440	0.020440	0.020320
VL2ave	0.020428	0.020402	0.020328
VL2max	0.020520	0.020520	0.020480
VL3ave	0.018035	0.016906	0.018053
VL3max	0.020400	0.020480	0.020400
VL4ave	0.018178	01018774	0.018181
VL4max	0.020560	0.020640	0.020587

DIMA 系统中网络带宽应大于各虚链路传输需求的总和，在这种情况下，上述实验中的 SMQRR、DWRR 和 WERR 调度算法均没有出现数据包丢失。同时，由于三种调度算法均属于轮询类调度，具有时间复杂度为 $O(1)$的特点，系统开销基本没有区别。

4.7 本章小节

本章为了进一步提高时间触发 AFDX 网络的性能，提出了适用于时间触发 AFDX 网络中速率限制虚链路的，具有低复杂度、高公平性和低调度时延特性的逐次最小定额

轮询调度算法。SMQRR 在 DRR 调度算法基础上，对每一个数据流所分配的服务数据量服务方式进行细化，通过采用两层循环调度的模式，有效克服了分组长度不同带来的不公平性并避免了数据流可能长时间无法获得服务的情况。通过理论分析和仿真验证，SMQRR 调度算法在公平性和时延上界均优于加权可变轮询调度算法和加权差额轮询调度算法。

5 DIMA 网络中静态数据存储完整性验证方案设计

随着航空电子系统的快速发展，航空电子数据在数量、多样性和重要性等方面发生了巨大的变化。航空电子数据不仅包含传统的航空电子数据，同时相应地增加了乘客可访问的影音娱乐、飞行信息等数据。通过前面三章所提出的时间触发 AFDX 网络及分别适用于时间触发虚链路和速率限制虚链路的调度算法，能够有效提高 DIMA 系统中多类型数据的传输性能，满足各航空电子系统对数据传输的时间确定性和时延需求。在数据完成传输到达并成功存储在网络服务器后，由于 DIMA 系统体系结构中服务器的分布式特性、以及航空电子网络的同一化发展，存储在航空电子网络中的数据不再具有物理上的隔离安全性，而可能面临被恶意用户非法访问、篡改和删除的威胁。

为了确保航空电子系统存储服务器中数据的完整性，以避免关键数据被删除或篡改所带来的严重后果，航空电子系统应具有定时验证航空数据完整性及相应的数据修复功能。随着航空数据数量的不断增加，传统的对被查数据完全下载后进行验证的方法并不适用于资源紧缺的航电系统。

本章在 DIMA 系统上构建了基于可信第三方的数据完整性验证模型，并设计实现了针对静态数据特点的基于代数签名的数据完整性验证算法。通过对各类航空电子数据在存储前完成相应的预处理，能够在降低网络通信和处理资源开销的前提下对数据的完整性进行验证，并在数据少量损毁的条件下实现数据的复原，以提高数据的可靠性。

5.1 基于第三方的数据完整性验证模型

本文结合 PDP 模型和数据持有性检查（Data Possession Auditor, DPA）模型，设计出一种 DIMA 系统结构下基于第三方的数据完整性验证模型。该模型借用 DPA 模型框架中第三方检查者（Third Party Auditor, TPA）思想，采用“挑战—响应”的验证方式，并结合 PDP 模型的同态验证标签机制以及概率抽样检查原理，在减轻数据持有者负担、降低验证所需开销的同时，进一步提高模型的安全性，使其适用于 DIMA 系统。基于可信第三方的数据完整性验证模型如图 5-1 所示：

根据图 5-1 可知，基于可信第三方的数据完整性检查模型由四个实体构成，即存储服务器、数据持有者，数据使用者和可信第三方检查者（Third Party Checker, TPC）。数据持有者即数据生产者，泛指利用各种途径获取的机内及机外信息，主要包括各类传感器数据、飞行计划、数据链信息等。对于 DIMA 系统，由于“信息一体化”、“网络一体化”的发展要求，数据持有者需要将数据按照使用情况存储到由综合化分布式模块电子构成的分布式存储服务器中，以便于数据融合与功能综合的实现。数据使用者既可以是综合化分布式模块电子的计算单元，也可以是飞行器驾驶者或是乘坐者。在通常情况下，数据使用者由于权限的不同，有区别地访问和使用存储在服务器端的数据。TPC 是模型

中的核心组成模块，主要负责对数据持有者生产的数据进行可靠的完整性检查任务，通常由独立于存储服务器和数据使用者的服务器组成，需要具有较大的存储和计算资源。

本文对所提模型及后续算法的研究和分析基于如下假设：

（1）TPC 绝对可信并且安全。TPC 具有物理和逻辑隔离性，在系统运行过程中无法对其操作逻辑进行查看和修改，完全不受他方干扰。

（2）由于 DIMA 系统中数据不再具有物理上的隔离安全性，数据使用者中的飞机乘客可能会对航空电子数据进行恶意的破坏，影响飞机的安全性和可靠性，是系统中的唯一威胁来源。

（3）整个验证过程通信安全。在数据完整性验证过程中，存储服务器、TPC 和数据持有者持有的密钥并不存在因意外攻击而泄露的可能。

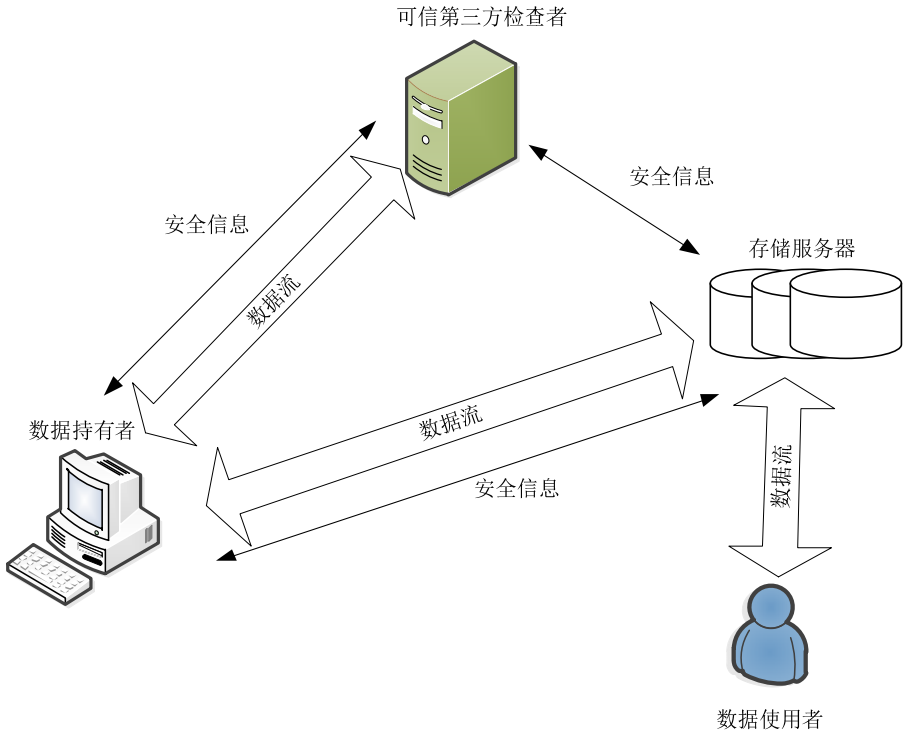

图 5-1 DIMA 数据完整性检查模型

5.2 基于代数签名的数据完整性验证方案

5.2.1 相关概念及定义

1）代数签名（Algebraic Signatures）

代数签名是一个具有同态性和代数性的哈希函数，其基本特性是数据块代数签名的和与相应数据块和的代数签名一致[86]。设 α 是长度为 L 伽罗华域(Galois Field, GF)，即有限域 $GF(2^L)$ 中的一个元组，则在 GF 域中所有非零元素都是的 α 阶，比如 $\beta=\alpha^i, 0\le i\le 2^L-1$，且有 $i=\log_\alpha(\beta)$ 和 $\beta=anti\log_\alpha(i)$。GF 域中的加、减法同字符串的按位异或（XOR）运算一致，而乘法相对复杂。GF 域中定义特殊元素“0”为全由 0 组

成的比特串。

当按照所选用的 GF 域元素长度 L 划分数据文件 F 时，假设划分数量为 n，记各数据子块分别为 $s_1, s_2, ..., s_n$，则数据文件 F 的代数签名定义如公式（5-1）：

$$AS_\alpha(F)=\sum_{i=1}^{n}s_i\times\alpha^{i-1} \tag{5-1}$$

其代数签名性质如下：

性质 1 文件 F 的所有数据块的总和的代数签名与各数据块代数签名的总和相等，如公式（5-2）所示

$$AS_\alpha(s_1)+AS_\alpha(s_2)+...+AS_\alpha(s_n)=AS_\alpha(s_1+s_2+...+s_n) \tag{5-2}$$

性质 2 两个文件 F, G 的总和的代数签名与每个文件签名的总和相等，如公式(5-3）所示：

$$AS_\alpha(F)+AS_\alpha(G)=AS_\alpha(F+G) \tag{5-3}$$

2）前向纠错（Forward Error-Correcting，FEC）码

(n, k) 前向纠错码能够为 k 个字符编码构成 $n-k$ 个校验字符，以容忍 $n-k$ 数量的数据丢失或损坏。本方案选用 Reed-Solomon（RS）纠删码对文件进行编码，并采用柯西矩阵构建 RS 编码操作时需要的一个 $n\times k$ 的分布矩阵 G。以（6，4）RS 码为例，消息 M 包含 4 个数据块 b1，b2，b3，b4，任一数据块都属于有限域 GF(2^L)。分布矩阵 G 由 4×4 单位矩阵构成前 4 行；并采用柯西矩阵（Cauchy matrix）构成剩余 2 行，如公式（5-4）所示：

$$G=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\a_{11}&a_{12}&a_{13}&a_{14}\\a_{21}&a_{22}&a_{23}&a_{24}\end{bmatrix},\ where\ a_{ij}=\frac{1}{i\oplus(d+j)} \tag{5-4}$$

编码后可得出 M^*，计算如公式（5-5），其中 r_1、r_2 为校验数据块，分别为：

$$M^*=G\times M^T=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\a_{11}&a_{12}&a_{13}&a_{14}\\a_{21}&a_{22}&a_{23}&a_{24}\end{bmatrix}\times\begin{bmatrix}b_1\\b_2\\b_3\\b_4\end{bmatrix}=\begin{bmatrix}b_1\\b_2\\b_3\\b_4\\r_1\\r_2\end{bmatrix} \tag{5-5}$$

其中，$r_1=a_{11}*b_1+a_{12}*b_2+a_{13}*b_3+a_{14}*b_4$，$r_2=a_{21}*b_1+a_{22}*b_2+a_{23}*b_3+a_{24}*b_4$。

编码 M^* 中的任意 4 块数据就能够消息 M 进行恢复，设 b_2、b_3 损坏，则恢复过程如公式（5-6）所示。

$$M^T = G^{'-1} \times \begin{bmatrix} b_1 \\ b_4 \\ r_1 \\ r_2 \end{bmatrix}, where\ G^{'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \quad (5\text{-}6)$$

方案中以约束组（constraint group）为单位进行编码，每一约束组包含 k 个数据块，由于约束组间相互独立，各个组的编码可以并行。

3）系统码及置换冗余（Permute-Redundancy， πR）算法

本方案采用系统码（systematic code）及 πR 算法以获取文件顺序性并保持系统的可靠性。系统码将未修改的输入符号嵌入在输出中。冗余置换克服了置换全部文件编码开销巨大的缺点，并且使数据完整性证明具有相同数量级的可靠性[11]。经冗余置换与系统码计算文件后的输出结构如图 5-2 所示。图 5-2 中不同的约束组使用不同的颜色进行标记。

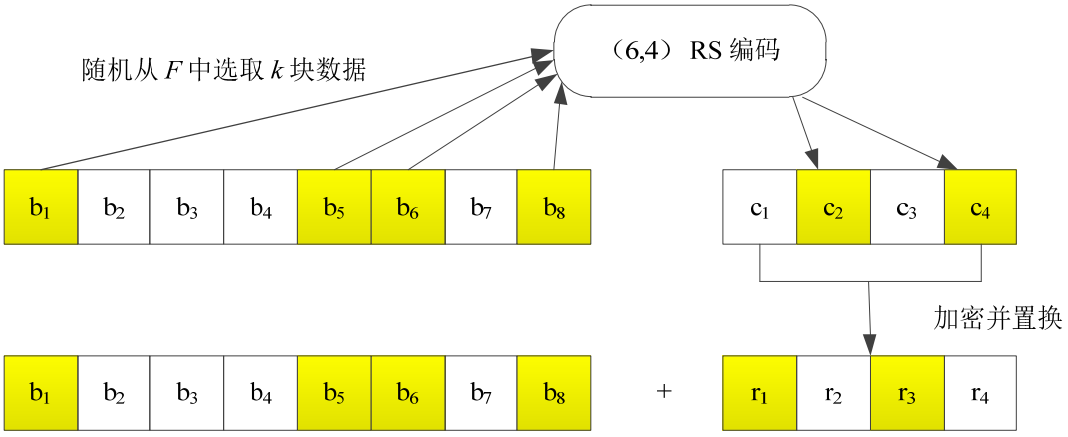

图 5-2 （6,4）RS 码及 πR 置换冗余算法示例

4）相关定义

在所提出的数据完整性验证方案中，文件存储的基本单位是 L 比特的数据块。文件 F 可看作由 n 块数据所组成， $F = \{m_1, ..., m_n\}$ 。

DIMA 架构中数据存储的完整性验证方案包含 5 个多项式时间算法，具体如下：

$\text{ParaGen}(1^K) \rightarrow (k, z, u, v, \gamma 1, \gamma 2, \alpha)$：该算法在文件存储到存储服务器前由用户执行，生成秘钥 (k, z, u, v)，随机数 $(\gamma 1, \gamma 2)$ 并赋值代数签名参数 α 。

$\text{TagBlock}(\alpha, F) \rightarrow S$：该算法同样在将文件存储到存储服务器前由用户执行，以代数签名参数 α 和文件 F 作为为每个文件数据块生成代数签名，所生成的签名数据附在 F 之后形成新文件 F*作为输出。

$\text{GenChal}(c, \text{S}) \rightarrow \text{chal}$：该算法以随机数 c 和代数签名集 S 作为输入，由第三方检查者执行生成一个挑战和一个相应的校验标签 T。

$\text{GenProof}(F^*, \text{chal}) \rightarrow \text{V}$：该算法由存储服务器执行，在接收到第三方检查者发出

的一个挑战 chal 时，根据其所存储文件 *F**和挑战 chal 做出应答，即生成数据完整性证明 V。

$\mathrm{CheckProof}(T, \mathrm{V}) \rightarrow \{\text{"success"}, \text{"failure"}\}$：该算法由第三方检查者在接收到数据完整性证明 V 时执行。根据在 TagBlocks(•)过程中生成的校验标签 T 对 V 是否是挑战 chal 的一个有效应答，根据验证结果输出成功、失败。

上述算法中使用了如下五个密码学原语：

$\mathcal{F}_{key}(\bullet)$−伪随机函数：$\{0,1\}^{\kappa} \times key \rightarrow \{0,1\}^{\kappa}$；

$\sigma_{key}(\bullet)$−伪随机置换：$\{0,1\}^{\log_2(n+n\bullet d/k)} \times key \rightarrow \{0,1\}^{\log_2(n+n\bullet d/k)}$；

$\pi_{key}(\bullet)$−伪随机置换：$\{0,1\}^{\log_2(n)} \times key \rightarrow \{0,1\}^{\log_2(n)}$；

$\psi_{key}(\bullet)$−伪随机置换：$\{0,1\}^{\log_2(n\bullet d/k)} \times key \rightarrow \{0,1\}^{\log_2(n\bullet d/k)}$；

k，v，u，z 分别是 PRF $\mathcal{F}_{(\cdot)}(\bullet)$, PRP $\sigma_{(\cdot)}(\bullet)$, PRP $\pi_{(\cdot)}(\bullet)$ 和 PRP $\psi_{(\cdot)}(\bullet)$ 的秘钥。

5.2.2 方案设计

DIMA 系统结构下的数据完整性验证方案由初始阶段（Setup phase）和挑战阶段（Challenge phase）组成。

1）初始化阶段

用户调用 $\mathrm{ParaGen}(\bullet)$ 为 PRF $\mathcal{F}_{(\cdot)}(\bullet)$, PRP $\sigma_{(\cdot)}(\bullet)$, PRP $\pi_{(\cdot)}(\bullet)$ 和 PRP $\psi_{(\cdot)}(\bullet)$ 生成秘钥，为 $\gamma 1$，$\gamma 2$ 在有限域中随机取值，并确定代数签名的参数 α。通过执行 $\mathrm{TagBlock}(\bullet)$，对文件 F 的各数据块 $m_i, 1 \le i \le n$ 按照数据块约束组进行 RS 编码产生校验数据集合 $C = (c_1, ..., c_{n\bullet d/k})$，并对所得校验集合 C 应用置换算法得到 $R = (r_1, ..., r_{n\bullet d/k})$；将 R 附于文件 F 后得到 $F^* = F \| R$；为 F^* 中每个数据块生成代数签名，得到代数签名集 $S = \{s_i\}, 1 \le i \le n + n \cdot d / k$；用户发送 F^* 给分布式 IMA 存储服务器，将 S 和随机数 $\gamma 1$ 发送至 TPC，并删除本地存储的 F^* 和 S。

2）挑战阶段

第三方检查者执行 $\mathrm{GenChal}(\bullet)$ 获取 c 个不同数据块的完整性证明来验证存储服务器所存储数据的完整性。第三方检查者为 c 取值并选取 PRP $\sigma_{(\cdot)}(\bullet)$ 的秘钥 v，其中 $1 \le c \le n + n \cdot d / k$；通过调用 c 轮 $\sigma_v(\bullet)$ 并求和所选定数据块的代数签名，第三方检查者可得校验标签 T；将 $\mathrm{chal}\{(v, c)\}$ 发送至 DIMA 存储服务器。

存储服务器根据所接收到的挑战 $\mathrm{chal}\{(v, c)\}$ 和存储文件 *F**，运行 $\mathrm{GenProof}(\bullet)$ 算法。在此算法中，存储服务器首先通过 chal 确定所要求的 c 个数据块的位置，并计算这些数据块的和；将计算结果作为应答 V 发送至第三方检查者。

第三方检查者运行 CheckProof(•)算法，计算所收到证明 V 的代数签名，并与在 GenChal(•)中得到的 T 进行比较。由于代数签名的特殊性质使得签名操作与求和操作可调换顺序，在存储服务器可靠的情况下，此次比较应该相等。方案的具体细节如图 5-3 中所示[86]。

Setup Phase:

ParaGen(1^K):

$k \xleftarrow{R} \{0,1\}^{\kappa}$, $v \xleftarrow{R} \{0,1\}^{\kappa}$, $u \xleftarrow{R} \{0,1\}^{\kappa}$, $z \xleftarrow{R} \{0,1\}^{\kappa}$;

$\gamma_1 \xleftarrow{R} \{0,1\}^{\kappa}$, $\gamma_2 \xleftarrow{R} \{0,1\}^{\kappa}$, $\alpha \xleftarrow{R} \{0,1\}^{\kappa}$;

TagBlock(α, F):

for $(i = 1;\ i \le n;\ i = i + 1)$ {

$p_i = m_{\pi_u(i)}$; }

$P = \{p_1, ..., p_n\}$;

for $(i = 1;\ i \le n/k - 1;\ i = i + 1)$ {

$(c_{i \cdot d+1}, ..., c_{(i+1) \cdot d}) = RS(p_{i \cdot k+1}, ..., p_{(i+1) \cdot k})$; }

$C = \{c_1, ..., c_{n \cdot d/k}\}$;

for $(i = 1;\ i \le n \cdot d/k;\ i = i + 1)$ {

$r_i = c_{\psi_z(i)}$; }

$R = \{r_1,\ ...,\ r_{n \times d/k}\}$;

$F^* = F \,||\, R = \{m_1, ..., m_{n+n \cdot d/k}\}$;

for $(i = 1;\ i \le n + n \cdot d/k;\ i = i + 1)$ {

$s_i = AS_{\alpha}(m_i)$; }

$S = \{s_1, ..., s_{n+n \cdot d/k}\}$;

Output F^* to CSS; Output (S, k, γ_1) to TPC;

Challenge Phase:

GenChal(c, S):

$c \xleftarrow{R} \{1,\ ...,\ n + n \times d/k\}$;

$v = \mathcal{F}_k(\gamma_1)$;

$T = 0$;

for $(j = 1;\ j \le c;\ j = i + 1)$ {

$i_j = \sigma_v(\gamma_2 + j)$;

$T = T + s_{i_j}$; }

Output chal$\{(v, c)\}$ to CSS;

GenProof(F^*, chal):

$V = 0$;

for $(j = 1;\ j \le c;\ j = i + 1)$ {

$i_j = \sigma_v(\gamma_2 + j)$;

$V = V + F^*[i_j]$

Output V to TPC;

CheckProof(T, V):

$AS_{\alpha}(V)$;

if $AS_{\alpha}(V) == T$; output success; else output failure.

图 5-3 基于代数签名的数据完整性验证方法算法描述

5.2.3 安全性分析

代数签名在使用中可取足够长的签名以使代数签名以极低的概率碰撞。一个 256 比

特的签名仅有 2^{-256} 的可能性发生碰撞。这对于一个未持有任何秘密信息的攻击者来说，构造签名碰撞是极难完成的。挑战中选取的数据块数量可根据用户的要求进行改变，这将防止存储服务器对固定的 c 预先计算并存储所有可能的校验标签值。

另外，攻击者对文件的成功攻击需要在未被检查机制发现的前提下，实现对任一约束组中超过纠删码恢复能力的数量的数据块实现破坏，即至少需要 d+1 的数据块被损毁；其攻击成功率如公式（5-7）：

$$P(\text{attack}) = P(\text{damage}) \times (1 - P(\text{detect})) \tag{5-7}$$

Curtmola 等人[134]证明了依照 RS 编码参数 k 和 d 的比例分别对文件数据块和校验数据块进行攻击可有效提高成功率并降低被检查到的概率。针对此种检查策略，可相应地对文件 F 和校验数据 R 两部分分别进行采样，由于这两部分的概率并非独立，可得公式（5-8）。公式（5-9）详细计算了各部分检查的成功概率，公式中的 x 表示损毁数据块总数，x_F 表示文件数据损毁数量，c_F、c_R 分别表示对文件数据块及校验数据块检查的数量。

$$P(\text{detect}) = P(\text{detect}_F) + P(\text{detect}_R) - P(\text{detect}_F) \times P(\text{detect}_R) \tag{5-8}$$

$$P(\text{detect}_F) \geq 1 - (1 - \frac{x_F}{F})^{c_F};\ P(\text{detect}_R) \geq 1 - (1 - \frac{x - x_F}{F \bullet d / k})^{c_R} \tag{5-9}$$

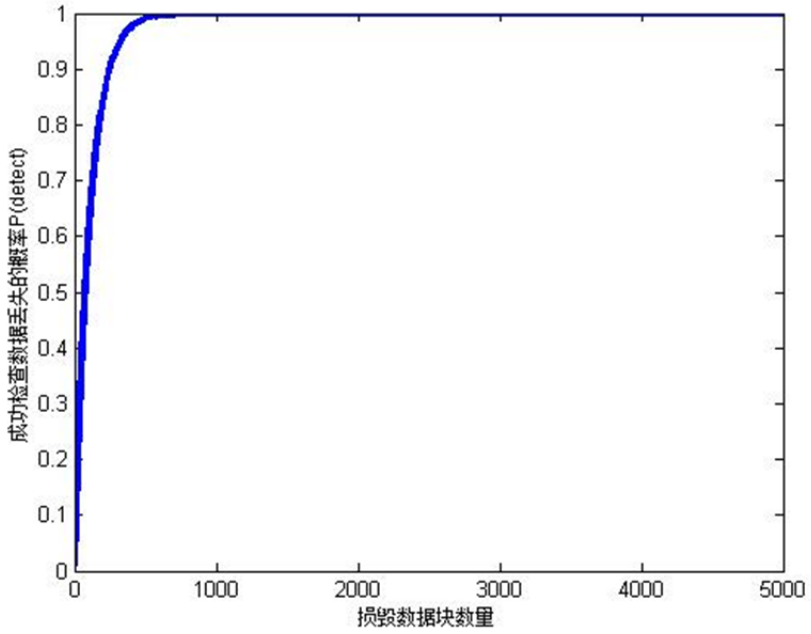

图 5-4　检查数据丢失的成功概率与损毁数据块数量关系图

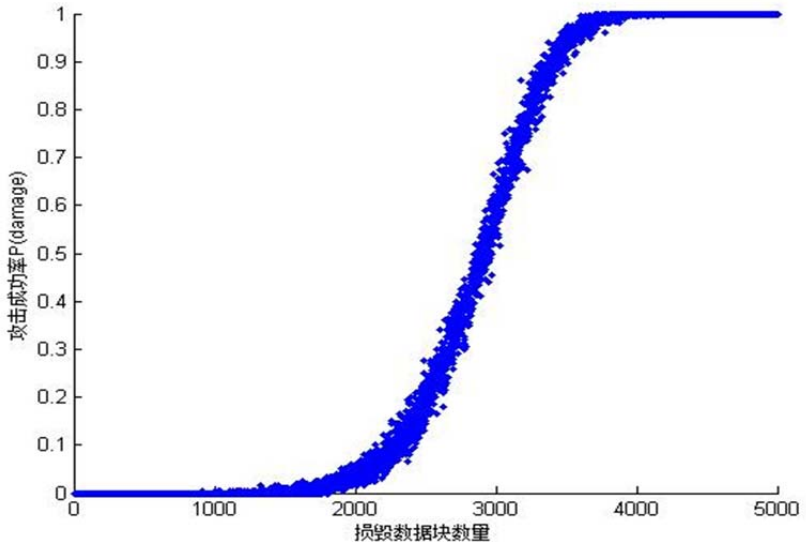

图 5-5　攻击成功概率 Monte-Carlo 结果

在实验中，选用（108,100,8）RS 编码对 100,000 块输入数据进行编码，产生 8000 冗余校验块。固定检查数据块数量为 1000，小于总体数据的 1%，不会增加系统开销。对数据块损毁数量从 1 到 108，000 对 P(detect) 进行计算并采用 Monte-Carlo 算法对 P(damage) 进行仿真，结果如图 5-4，图 5-5 所示。由以上两图可以看出，对于少量的删除， P(damage) 几乎为 0。而对于大量的删除， P(detect) 基本为 1。而在 500 到 1000 这个数量区间内，两项概率的乘积，即总体攻击成功率在任意情况下均低于 10^{-8}。

5.3 性能分析与实验

在配置为 Intel Core2 Duo CPU P8600 2.4 GHz、4GB RAM，西部数据 7200 转、250GB ATA 接口硬盘、Windows 7 操作系统的机器上，为 8 KByte 的数据块计算长度 256bit 的代数签名，平均速率达到 273 MB/s。

5.3.1 计算开销

在初始化阶段，用户需要为文件数据编码及代数签名产生所需的公共参数和私有参数。由用户计算出来的每个数据块（共 $n+n\times d/k$ ）所对应的代数签名将被重复使用。在 $\mathrm{ParaGen}(1^K)$ 中，第三方检查者调用一次 PRF、c 次 PRP 操作和 c 次代数签名求和，且仅需向存储服务器发送参数 v 和随机数 c。在 $\mathrm{GenProof}(\bullet)$ 中，存储服务器需要执行 c 次 PRP 操作与求和操作，并将所得结果传送给第三方检查者。在 CheckProof(•)中，第三方检查者需要完成一次比较。本方案中仅使用了 PRF、PRP、对称加密、XOR 和代数签名运算，根据实验结果可验证以上运算的计算效率较公钥加密算法都较高。

5.3.2 存储开销

在上述实验中，选用文件 F 拥有 n=100,000 个数据块，每个数据块为 8KByte，代数签名长为 256bit。在初始化阶段，用户将编码后的文件发送给存储服务器并将数据块签名集发送给第三方检查者。第三方检查者的附加存储空间为 3MByte。用户存储 4 个 256

bit 私有 PRP 秘钥和 2 个 256 bit 随机数。在挑战阶段，用户及存储服务器使用 AES 算法为 σ(•)以随机选择数据块索引[86]。

5.3.3 通信开销

选用 HMAC 算法为$\mathcal{F}(\bullet)$以产生随机的 PRP 秘钥。在此过程中，第三方检查者向存储服务器发送 288bit 的消息，包括 256bit 的秘钥和 32bit 的 c。存储服务器的挑战答复约为 8KByte。

5.3.4 抽样检查分析

所提出基于现场抽查（spot checking）的数据完整性验证算法不仅满足开销轻量级的需求，而且在有一部分（比如超过 1%）数据损坏的情况下能够有效地以高概率检查出错误。置信度被定义为成功检查出数据文件损毁的概率。假设存储服务器中一个由 n 个子块数据组成的文件损毁率为 s。TPC 在整个文件中随机选取 c 块不同数据用以现场抽查，定义所抽查数据块与文件损毁数据块相匹配的数量为离散变量 x，则在一次选取 c 块数据的挑战中，TPC 发现存储服务器存在违规操作的概率 P_x 可按照公式（5-10）计算。

如果所选取的数据块数量远小于文件 F 的数据块数量，即 $c \ll n$，则由 $(n-n\times s-c+1)/(n-c+1)\approx 1-s$，可得出：$P_x \approx 1-(1-s)^c$。由于置信度仅由损毁率 s 和抽查数量 c 决定，所提出的算法能够有效地完成大文件的完整性检查。

$$\begin{aligned} P_x &= P\{x \geq 1\} = 1 - P\{x = 0\} \\ &= \frac{n-n\times s}{n} \times \frac{n-n\times s-1}{n-1} \times \ldots \times \frac{n-n\times s-c+1}{n-c+1} \\ &= 1 - \prod_{i=0}^{c-1} \frac{n-n\times s-i}{n-i} \end{aligned} \tag{5-10}$$

$$\because 1-s \geq \frac{n-n\times s-i}{n-i} \geq \frac{n-n\times s-i-1}{n-i-1} \tag{5-11}$$

$$\therefore 1-(1-s)^c \leq P \leq 1-\left(\frac{n-n\times s-c+1}{n-c+1}\right)^c \tag{5-12}$$

在不同损毁率 s 下，完整性验证可信度 P_x 与抽样块数 c 之间的关系如图 5-6 所示。由该图可知，验证可信度随抽样块数的增加而增大。例如，当存储服务器中的文件数据损坏率达 1%时，若置信度要求达到 95%，则抽样检测块数需要 300 块；若置信度要求 99%，则需要随机选取数据文件 F 中的 460 块数据即可。同时，对于低损毁率的情况下，可以通过进行多次检测以提高可信度。例如，当 $s=0.1\%$ 时，在抽样检查 1000 块数据的条件下，可信度仅能达到 60%。当进行 5 此独立检查后，则发现损毁数据块的概率可达到：$1-(1-0.6)^5 \approx 98.9\%$。

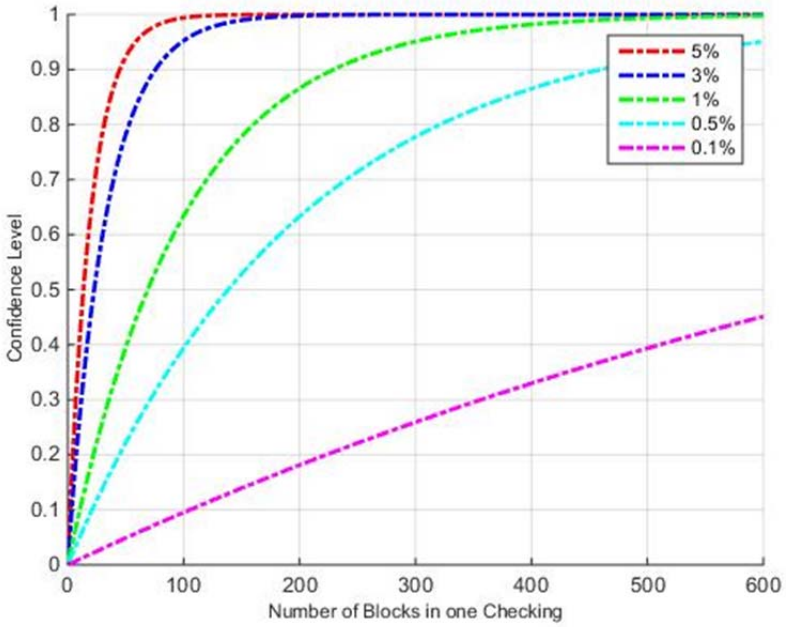

图 5-6 抽样检查块与置信度关系

5.4 本章小节

根据 DIMA 系统数据存储的特点，本节提出了一种基于可信第三方的数据完整性验证模型。依照模型所采用的“挑战—响应”验证模式，提出了基于代数签名的数据完整性验证方案。此方案较现有技术有以下优点：首先，采用非公钥密码技术提高了执行效率；其次，第三方检查者在挑战过程中并不需要与原始数据进行比对，仅对由存储服务器返回的数据进行验证即可；第三，挑战与回复的消息分别仅有 200bit 和 8 KByte，降低了带宽开销[86]。因此，高效性与可靠性使得该方法适用于计算资源、带宽资源均有限的 DIMA 系统。第四，由于基于代数签名的数据完整性验证方案在每次挑战时，文件块索引值都是随机选取，并不固定，因而对验证的次数没有限制。较 POR 模型中挑战次数被初始阶段所产生的哨兵数量所限制有明显优势。最后，由于在预处理阶段对文件采用纠错码进行编码变换，该方案能够支持数据的恢复操作，使该方案具有容错和纠错功能，提高了数据的可靠性。实验结果表明，磁盘的吞吐速率是影响完整性验证算法的主要原因。

概率性安全是本方案的缺点。当存储服务器中仅有少量文件遭到破坏或删除时，第三方检查者通过查询少量的数据块仅能够得到概率性的安全性分析，而只有通过对全部数据的验证才能获得绝对的安全保证。同时，本方案无法支持任何插入、删除、修改等更新操作，此问题将在下一章中进行详细讨论和解决。

6　DIMA 网络中动态数据存储完整性验证方案设计

航空电子数据不仅包含地图信息、影音娱乐数据等静态数据，同时具有大量需要动态操作的数据，如传感器数据、导航信息等。为了支持航空电子数据的插入、修改和删除操作，依照 5.1 小节所提出的基于可信第三方的数据完整性验证模型，分别提出了基于跳表和基于 Merkle 哈希树的动态数据完整性验证方案。

6.1 基于跳表的动态数据完整性验证方案

为了有效地支持航空电子数据的动态操作，本节在 5.3 小节中的基于代数签名的数据完整性验证方案基础上，结合跳表存储结构，提出了基于跳表的动态数据完整性验证方案，对数据的插入、修改和删除操作做了详细的验证设计。

6.1.1 相关概念及定义

1）跳表（Skip-list）

跳表是一个类似于二叉树的结构，具有快速查找第 *i* 个节点的性质。跳表中的每个节点存储一个表示从该节点能访问到的底层节点的数目的数值，每个节点有 1 个或 2 个子节点。构建跳表时，首先对每一文件块构建一个底层节点，每两个底层节点构建一个父节点，若有单个底层节点剩余则留至下一层处理，循环执行直至顶层节点。跳表的访问方法类似于二叉树的访问方法。从根节点开始，依次判断当前节点（r）向下的子节点（$\mathrm{dwn(r)}$）和向右的子节点（$\mathrm{rgt(r)}$）的访问范围，若目的节点落在$\mathrm{dwn(r)}$的访问范围内，则$\mathrm{r=dwn(r)}$，否则$\mathrm{r=rgt(r)}$。

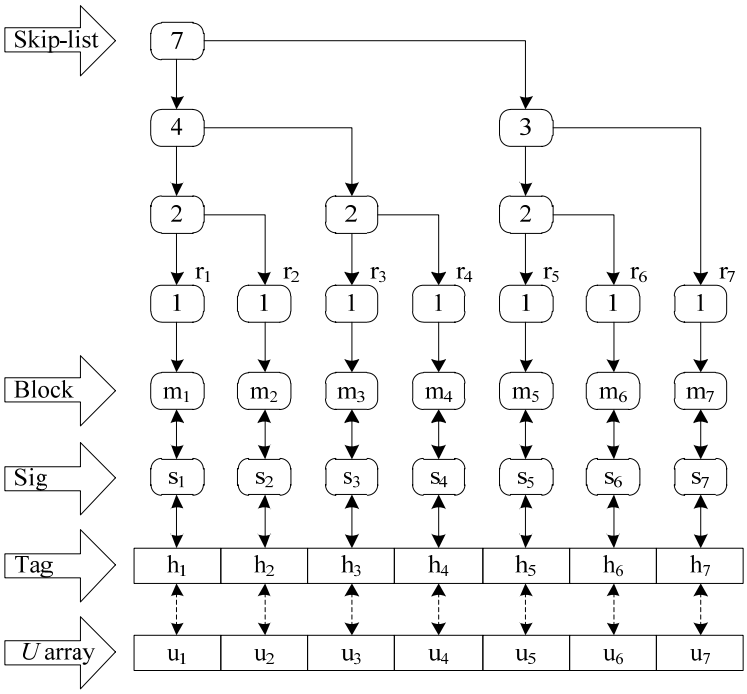

图 6-1　文件的 skip-list 结构图

此外，用户还为每个数据块 m_i 创建一个 u_i 值构成 U 值数组，并计算数据块 m_i 的代数签名 s_i 及标签值 t_i，其中 u_i 值代表相应数据块更新的次数。跳表、数据块、标签值、签名值和 U 值间的指针均在构建跳表结构时由用户创建。文件的跳表结构如图 6-1 所示。

2）相关定义

算法中使用了三个密码学原语：

$\mathcal{F}_{key}(\bullet)$ −伪随机函数：$\{0,1\}^{\kappa} \times key \to \{0,1\}^{\kappa}$；

$H(\bullet)$ − 密码哈希函数：$\{0,1\}^{*} \to G$;

$\pi_{key}(\bullet)$ −伪随机置换：$\{0,1\}^{\log_2(n)} \times key \to \{0,1\}^{\log_2(n)}$；

k，v 分别是 PRF $\mathcal{F}_{(\cdot)}(\bullet)$，和 PRP $\pi_{(\cdot)}(\bullet)$ 的秘钥。

6.1.2 方案设计

基于跳表的动态数据完整性验证算法同样包含 5 个多项式时间算法（ParaGen，TagBlock，GenChal，GenProof，CheckProof），并由初始化阶段（Setup phase）、挑战阶段（Challenge phase）和更新阶段（Update phase）组成。其中，各多项式时间算法的功能与 6.1.3 小节所述相似，具体实现细节见图 6-8。

1）初始化阶段

用户调用 $\mathrm{ParaGen}(\bullet)$ 为 PRF $\mathcal{F}_{(\cdot)}(\bullet)$ 生成密钥 k，并确定代数签名参数 α。通过调用 $\mathrm{TagBlock}(\bullet)$ 为文件 F 的每一数据块计算代数签名 $s_i, (1 \le i \le n)$ 和标签（Tag）值 $t_i, (1 \le i \le n)$，签名集与标签集分别表示为：$S = \{s_1, ..., s_n\}$ 和 $T = \{t_1, ..., t_n\}$。用户初始化 U 数组并构建数据文件 F 的跳表。文件 F 以及所得跳表、签名集和标签集构成了处理后的文件 F*。用户上传 F*至存储服务器，并发送 U 数组、密钥 k 及代数签名参数 α 至第三方检查者。

2）挑战阶段

第三方检查者通过验证存储服务器中 c 个不同数据块的完整性证明以验证服务器所存储数据文件 F 的完整性。第三方检查者为 c 取随机值，其中 $1 \le c \le n$；生成 PRP $\sigma_{(\cdot)}(\bullet)$ 的秘钥 v，并将 $\mathrm{chal}\{(v,c)\}$ 发送至 DIMA 存储服务器。通过调用 c 轮 $\sigma_v(\bullet)$，第三方检查者与存储服务器均能够确定这轮挑战中所选数据块的位置。

存储服务器根据所接收到的挑战 $\mathrm{chal}\{(v,c)\}$ 和存储文件 F*，运行 $\mathrm{GenProof}(\bullet)$ 算法。在此算法中，存储服务器首先通过 chal 确定所要求的 c 个数据块的位置，对 c 个数据块计算完整性证明 P；并将 P 发送至第三方检查者。

第三方检查者在接收到证明 P 后，运行 $\mathrm{CheckProof}(\bullet)$ 算法以验证 P 的正确性。方案的具体细节如图 6-2 中所示。

Setup Phase:

ParaGen(1^K):

$\alpha \xleftarrow{R} \{0,1\}^{\kappa}$;

TagBlock(α, m_i, u_i, i) $\rightarrow (s_i, t_i)$;

for (i=1; $i \le n+n \cdot d/k$; $i=i+1$) {

$s_i = AS_{\alpha}(m_i)$;

$t_i = H(s_i \| \mathcal{F}_k(u_i) \| i)$;

$u_i = 0$; }

$S = \{s_1, ..., s_n\}$;

$T = \{t_1, ..., t_n\}$;

Output F^* to CSS; output U array and (k, α) to TPC;

Challenge Phase:

$c \xleftarrow{R} \{1, ..., n\}$; $v \xleftarrow{R} \{0,1\}^{\kappa}$;

for ($j = 1$; $j \le c$; $j = j + 1$) {

$i_j = \pi_v(j)$; } //compute the indices of the blocks;

Output chal$\{(v, c)\}$ to CSS;

GenProof(F^*, chal)

for ($j = 1$; $j \le c$; $j = j + 1$) {

$i_j = \pi_v(j)$; } //compute the indices of the blocks;

/* For $i_j \in \{i_1, ..., i_c\}$ search the tag value array to find t_{i_j},

then find its corresponding signature s_{i_j} and block m_{i_j}

through the pointers */

$M = m_{i_1} + \cdots + m_{i_c}$;

$t = t_{i_1} + \cdots + t_{i_c}$;

Output $P = \{M, t, \{s_{i_1}, \cdots, s_{i_c}\}\}$ to TPC;

CheckProof(k, chal, P, U) $\rightarrow$ {"success","failure"} :

$S = s_{i_1} + \cdots + s_{i_c}$; //compute by TPC

/* Search the U array with index $\{i_1, ..., i_c\}$ for values $\{u_{i_1}, ..., u_{i_c}\}$ */

H = $H(s_{i_1} \| \mathcal{F}_k(u_{i_1}) \| i_1) \cdot \ldots \cdot H(s_{i_c} \| \mathcal{F}_k(u_{i_c}) \| i_c)$;

if ($AS_{\alpha}(M) == S$ && H $== t$) output success; else output failure.

图 6-2 基于跳表的动态数据完整性验证算法描述

3）更新阶段

基于跳表的动态数据完整性验证算法能够有效地处理数据动态操作，包括：插入操作、修改操作以及删除操作，以下将分别对这三种操作进行详细说明：

插入操作：在第 j 块数据后插入新数据块 m_i^* 时，用户需要先将 m_i^* 发送给第三方检查者。第三方检查者在接收到数据块后，计算其对应的所有相关信息并发送至存储服务器。最后，TPC 向存储服务器发起一次挑战以验证插入操作是否成功。图 6-3 和图 6-4

分别详细描述了插入操作的具体细节并展示了一个插入操作的具体例子。

Insertion :

1. User sends blocks m_i^* to TPC.
2. TPC: Adds a u_{n+1}^* at the end of U array, initializes to 0;
   computes $s_i^* = AS_\alpha(m_i^*)$, $t_{n+1}^* = H(s_i^* \| \mathcal{F}_k(u_{n+1}^*) \| n+1)$;
   sends $(j, m_i^*, s_i^*, t_{n+1}^*)$ to Server.
3. CSS: Insert a r_i after the j^{th} bottom-level node r_j in the skip list;
   Insert the m_i^*, s_i^* at the appropriate position, add t_{n+1}^* at the end of tag value array;
   maintains the pointers among them, update the skip-list.
4. TPC: Sends $j+1$ to CSS;
5. CSS: Searches the skip-list to find the $(j+1)^{\text{th}}$ block m_{j+1}, its signature s_{j+1} and
   corresponding tag value $t_{(j+1)'}$; computes $t = H(s_{j+1} \| h_{(j+1)'})$;
   sends (m_{j+1}, s_{j+1}, t) to TPC;
6. TPC: **If** $AS_\alpha(m_{j+1}) == s_{j+1}$ && $H(s_i^* \| h_{n+1}^*) == t$, the Insertion is successful.

图 6-3 插入操作算法描述

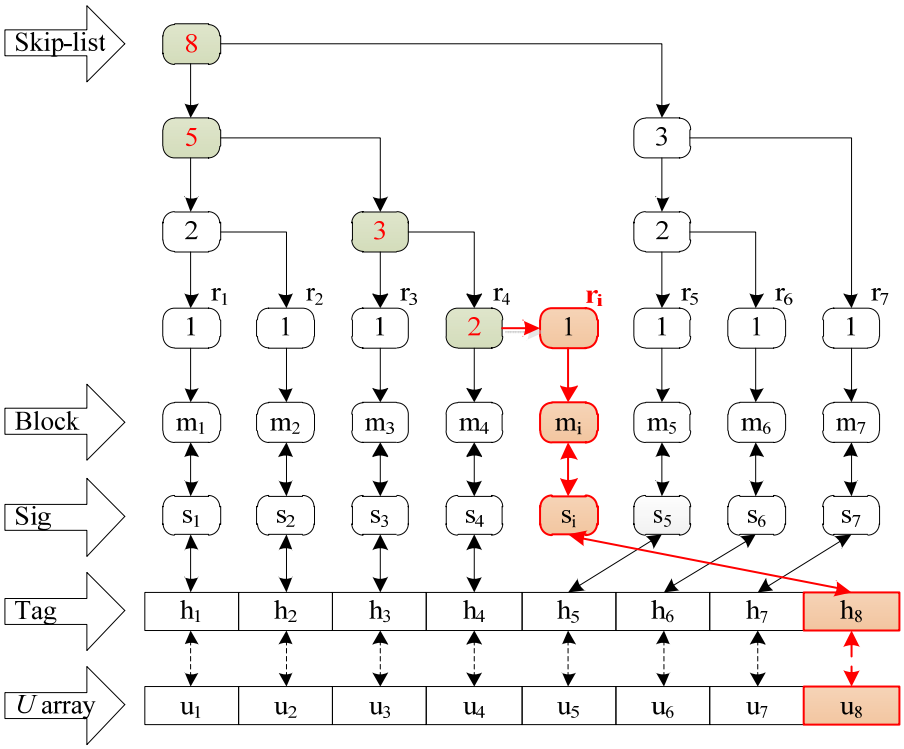

图 6-4 插入操作示例

修改操作：当需要修改第 i 个数据块 m_i 为 t_{n+1}^* 时，用户需将数据块索引号 i 发送至 TPC，TPC 继而将此索引转送至存储服务器。存储服务器在跳表中查找第 i 块数据及其相应的信息，并将所得相关数据返回给 TPC。若所接收到的数据块及其相关信息能够通过 TPC 的完整性验证，则 TPC 发送第 i 个数据块 m_i 若至用户。用户更新数据块后将更

新后的数据块发送至 TPC。TPC 在接收到更新数据块后，计算其相应的信息并发送至存储服务器。存储服务器更新所有第 i 块数据的相关信息以完成修改操作。修改操作的具体细节如图 6-5 所述。在完成一次修改操作后，TPC 向存储服务器发送一个挑战以验证修改操作是否成功。此挑战过程与插入操作的步骤 4 至步骤 6 相同。

Modification :

1. User: Sends i to TPC;
   TPC: Forwards this index to CSS.
2. CSS: Searches the skip-list to find the block m_i, the corresponding signature s_i, tag value $t_{i'}$ and its index i'; Sends $(m_i, s_i, t_{i'}, i')$ to TPC;
   computes $s_i^* = AS_\alpha(m_i^*),\ t_{n+1}^* = H(s_i^* \| \mathcal{F}_k(u_{n+1}^*) \| n+1)$;
3. TPC: Computes $AS_\alpha(m_i) ?= s_i,\ t_{i'} ?= H(s_i \| \mathcal{F}_k(u_{i'}) \| i')$ to verify the integrity of the block.
   If equalities hold, TPC returns m_i to user.
4. User: Updates $m_i \to m_i^*$ and sends to TPC.
5. TPC: Updates $u_{i'}$ in the U array $u_{i'}^* = u_{i'} + 1$, updates $m_i \to m_i^*$,
   computes $s_i^* = AS_\alpha(m_i^*),\ t_i^* = H(s_i \| \mathcal{F}_k(u_{i'}^*) \| i)$; sends (m_i^*, s_i^*, t_i^*) to CSS.
6. CSS: Updates $m_i \to m_i^*,\ s_i \to s_i^*,\ t_{i'} \to t_{i'}^*$.

图 6-5 修改操作算法描述

删除操作：当删除第 i 个数据块 s_i 时，用户需将数据块索引号 i 发送至 TPC，TPC 继而将此索引转送至存储服务器。存储服务器在跳表中查找并删除第 i 块数据及其相应的信息，同时，存储服务器将更新标签集和 U 值数组。图 6-6 和图 6-7 分别描述了删除操作的具体细节并展示了一个删除操作的具体例子。在完成一次删除操作后，TPC 向存储服务器发送一个挑战以验证删除操作是否成功。此挑战过程与插入操作的步骤 4 至步骤 6 相同。

Deletion :

1. User: Sends the index i to TPC;
   TPC: Forwards this index to CSS.
2. CSS: Searches the skip-list to find the block m_i, the corresponding signature s_i, tag value $t_{i'}$ and its index i', deletes m_i and s_i;
   find the last value of the tag array t_n and its corresponding signature $s_{n'}$;
   updates the skip-list, sends $(i', t_n, s_{n'})$ to TPC;
3. TPC: Computes $AS_\alpha(m_i) ?= s_i,\ t_{i'} ?= H(s_i \| \mathcal{F}_k(u_{i'}) \| i')$ to verify the integrity of the block.
   If equalities hold, TPC returns m_i to user.
4. CSS: Updates $t_{i'} \to t_{i'}^*$, delete t_n.

图 6-6 删除操作算法描述

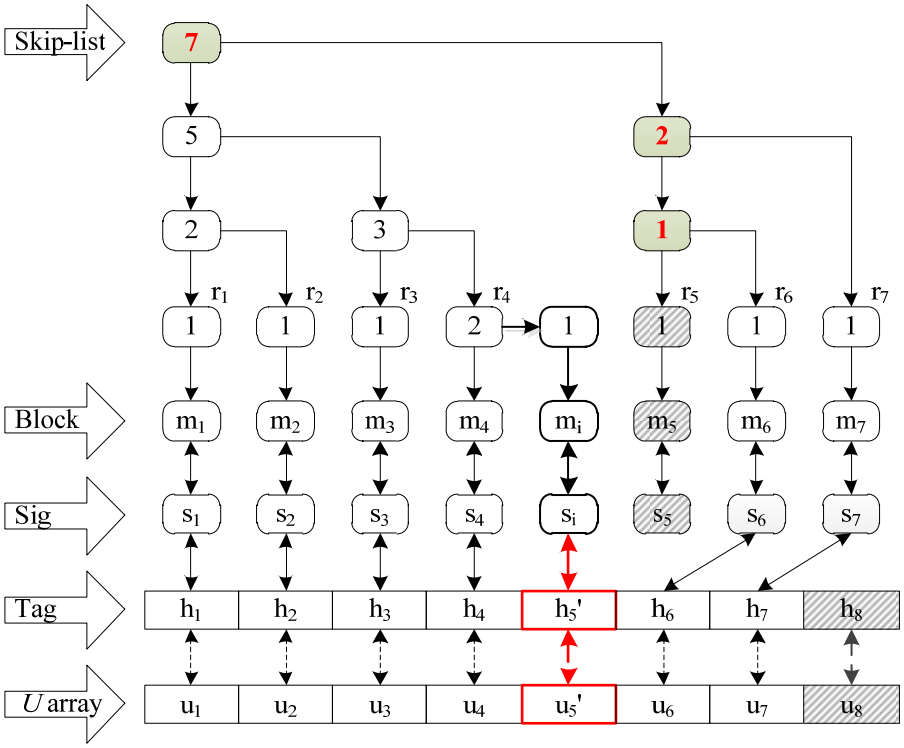

图 6-7 删除操作示例图

6.1.3 安全性分析

本小节所提出的基于跳表的动态数据完整性验证算法在基于代数签名的静态数据完整性验证方案基础上，增加了对标签的验证。标签的产生使用了哈希加密，具有密码安全性，提高了仅基于代数签名的数据完整性验证方案的安全性。

6.1.4 适用性分析

基于跳表的动态数据完整性验证方案中，TPC 在每次挑战时都是随机计算选取数据块索引值，并不固定且不具有确定性。因而该方案并不受验证次数的限制。

公开验证是指是否支持除数据持有者之外的第三方对数据的完整性进行挑战与验证。因为方案所采用的代数签名技术并非依据公钥加密算法，所以无法支持公开验证。

方案中并未对数据进行容错纠错设计，无法使数据在遭受恶意删改后完成恢复。但可采用 5.2 小节中所提出的纠删码技术在数据存储预处理阶段进行编码操作，以弥补恢复性的缺失。

6. 2 基于跳表的数据完整性验证方案性能分析与实验

6.2.1 计算开销

在初始化阶段，仅用户需要对每一数据块计算其签名和标签值，而在随后的挑战和更新阶段，用户并不需要参与。

在更新阶段：TPC 计算更新信息并验证更新结果，这些操作的计算复杂度均为常量；

存储服务器由于需要在跳表中查找所更新的数据块，其计算复杂度为$\log n$。

在挑战阶段：TPC 生成两个随机数并使用 U 值数组进行验证。存储服务器搜寻标签集以查找挑战所需的数据块及其签名和标签值，并计算完整性证明 P。以上所述各步骤的计算复杂度均为常量。

6.2.2 存储开销

在数据完整性验证过程中，TPC 需要存储密钥和 U 值数组。存储服务器在数据文件之外，需要保存签名集、标签集以及跳表结构。假设 L 长度为 1024 位，一个 4G 字节的文件 F 能够划分为 1,000,000 个 4K 字节大小的数据块。每一个数据块具有长为 128 字节的签名和 20 字节的标签。跳表结构中的节点数量大约为 2,000,000 个，即文件数据块数量的 2 倍。每一跳表节点为 3 字节，因此，存储服务器所需的额外存储空间为 154M 字节，即原文件大小的 3.8%。设定 U 值数组每项的大小为 2 字节（可支持 2^{16} = 65536 次数据更新），TPC 为存储 1,000,000 块数据所需的额外空间为 2M 字节，即原文件大小的 0.05% 。

6.2.3 通信开销

通信复杂度主要包含 TPC 与存储服务器的相互通信，以下分析均沿用存储开销分析中对各参数的假设。

在更新阶段：以 0 修改操作为例，存储服务器首先发送大小为 4,151K 字节的 $(m_i, s_i, t_{i'}, i')$ 至 TPC 以证明原始数据块 m_i 的完整持有。TPC 向存储服务器上传大小为 4,148K 字节的 (m_i^*, s_i^*, t_i^*)。

在挑战阶段：消息 $\mathrm{chal}\{(v,c)\}$ 的长度为常量 19 字节，而所接收的反馈消息 $P=\{M,t,\{s_{i_1},\cdots,s_{i_c}\}\}$ 的大小则由变量 c 决定。当 $c=460$ 时，此完整性检查算法可达到 99%的检测率，此时 P 为 62.9K 字节。

6.2.4 运行性能分析

本文实现了基于 MHT 的动态数据完整性验证方案的核心算法并进行了性能测试与对比。在硬件及操作系统条件不变的情况下，模型中的算法使用 OpenSSL 1.0.1u 中的 Crypto 库进行实现。算法中具体函数与参数大小安排如下：密钥采用 RSA 来产生，伪随机函数 f 用 HMAC 来实现，伪随机排列 r（用来随机选择块的索引）使用 AES 来实现，h 由 SHA-1 来实现，模数 N 的值定为 1024bit，文件块大小定为 4KB。AES ECB 和 HMAC（SHA-1）的速率分别为 97 MB/s 和 178 MB/s。

6.3 基于 MHT 的动态数据完整性验证方案

为了有效地提高数据存储的安全性及数据存储结构的有效性，本节提出了基于双线性对映射签名和 Merkle 哈希树的动态数据完整性验证算法。

6.3.1 相关概念及定义

1）双线性对映射（Bilinear Map）

双线性对映射$e:G\times G\to G_T$是一个具有以下性质的映射：（1）可计算性：存在一个有效的算法可以计算出映射e；（2）双线性：对于所有的$h_1,h_2\in G$和所有的$a,b\in\mathbb{Z}_p$，$e(h_1^a,h_2^b)=e(h_1,h_2)^{ab}$均成立；（3）非退化性：$e(g,g)\neq 1$。

其中，G是一个生成元为g的Gap Diffie-Hellman (GDH) 群，G_T为乘法循环群，其阶数为素数p。

2）Merkle 哈希树（Merkle Hash Tree，MHT）

Merkle哈希树是由Merkle提出的一种建立认证字典的基础数据结构[135]，它是在二叉树的结构上运用哈希函数来构造认证字典。在构建 Merkle 哈希树时，首先对每一文件数据块构建一个叶子节点，采用抗碰撞的哈希函数$h(\bullet)$计算各叶子节点所对应文件块数据的哈希值，也存储在相应的叶子节点中。对于非叶子节点，使用同样的哈希函数对其左右子节点进行计算，所得结果即为其存储数据。

本算法采用 Merkle 哈希树同时对文件数据块的数值及位置进行认证。Merkle 哈希树中的叶子节点以从左至右的顺序依次存放文件数据块的内容。第i个叶子节点同时存储一个指向数据块m_i的指针。此外，用户还为每个数据块m_i创建一个u_i值构成U值数组，并计算数据块m_i的代数签名s_i及标签值t_i，其中u_i值代表相应数据块更新的次数。MHT、数据块、标签值、签名值及U值间的指针均在构建MHT结构时由用户创建。

当在MHT上对数据块进行完整性验证时，不可信服务器需向验证者提供完整的验证路径信息（Authentication Path Informaiton，API），即从包含数据块的叶子节点到根节点的路径上各节点的同胞节点的哈希值，此外，路径上的每一节点必须依序给出并且标识出是左节点还是右节点。验证者通过所获得的API计算出MHT根节点的哈希值，并与已存储的可信根节点哈希值h_R进行比较，若相同，则证明不可信服务器所给出的答案是真实有效的，反之则认为是错误的。以m_6为例，不可信服务器向验证者提供的 API为$\Omega_6=<h(m_5),\ h(F),\ h(A)>$，即 MHT 中从存储$m_6$的叶子节点到根节点$R$路径上各同胞节点的哈希值，基于所得$\Omega_6$，验证者能够计算出$h(m_6)$、$h(E)$、$h(B)$和$h(R)$，最后将$h(R)$与$h_R$进行比较。

图6-8展示了该算法所采用的MHT结构，并描绘了数据块m_6的验证过程。

2）定义

文件F可看作由n个大小固定的数据块所组成，$F=\{m_1,...,m_n\}$，其中$m_i\in\mathbb{Z}_p$，p为大素数。

此算法中使用了五个密码学原语，如下所示：

$H(\bullet)$ – 密码哈希函数：$\{0,1\}^*\to G$；

$h(\bullet)$ – 密码哈希函数；

$\pi_{key}(\bullet)$ −伪随机置换：$\{0,1\}^{\log_2(n)} \times key \to \{0,1\}^{\log_2(n)}$；

$\mathcal{F}_{key}(\bullet)$ −伪随机函数：$\{0,1\}^{\kappa} \times key \to \{0,1\}^{\kappa}$；

$f_{key}(\bullet)$ − PRF：$\{0,1\}^{\log_2(n)} \times key \to \{0,1\}^{\log_2(n)}$；

k、k_1、k_2 分别是 PRF $\mathcal{F}_{(\cdot)}(\bullet)$、 PRP $\pi_{(\cdot)}(\bullet)$ 和 PRF $f_{(\cdot)}(\bullet)$ 的秘钥。

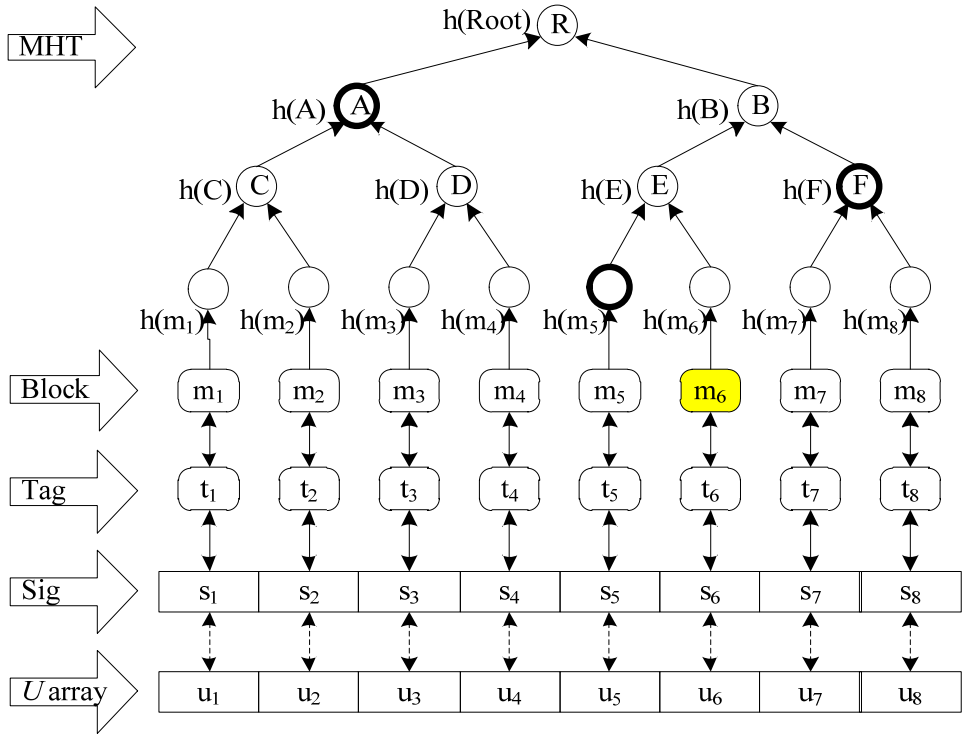

图 6-8　MHT、数据块、标签、签名数组及 U 值数组相关性示意图

6.3.2 方案设计

基于 MHT 的动态数据完整性验证算法同样包含 5 个多项式时间算法（ParaGen，TagBlock，GenChal，GenProof，CheckProof），并由初始化阶段（Setup phase）、挑战阶段（Challenge phase）和更新阶段（Update phase）组成。其中，各多项式时间算法的功能与 5.3.1 小节所述相似，具体实现细节见图 6-11。

1）初始化阶段

用户调用 $\mathrm{ParaGen}(\bullet)$ 生成公开参数 $pk=(v,g,u,e(u,v))$ 及私密参数 $sk=(\alpha,k)$。通过调用 $\mathrm{TagBlock}(\bullet)$，用户完成对文件 F 的每一数据块 m_i, $(1\le i\le n)$ 计算签名（Signature）s_i 和标签（Tag）值 t_i，并生成 U 值数组且对其中的各项赋初始值“0”。签名数组与标签数组分别记为：$S=\{s_1,...,s_n\}$ 和 $u_i = 0$；｝。用户基于文件 F 构建其 MHT 并签名 MHT 的根节点，可得 $sig_{sk}(h(R))$。用户向存储服务器上传 $(F, pk,$ MHT, $T, S, sig_{sk}(h(R)))$，并发送 U 数组、密钥至第三方检查者。最后，用户将删除本地所存储的 (F, T, S) 以完成初始化阶段的操作。

Setup Phase:

ParaGen$(1^K) \rightarrow (pk, sk)$:

$k \xleftarrow{R} \{0,1\}^{\kappa}$, $\alpha \xleftarrow{R} \mathbb{Z}_p$, $u \xleftarrow{R} G$, $v \xleftarrow{R} g^{\alpha}$; // g is a generator of G;

$pk = (v, g, u, e(u,v))$, $sk = (\alpha, k)$

TagBlock$(g, \alpha, m_i, u_i, i) \rightarrow (s_i, t_i)$:

for $(i = 1;\ i \le n;\ i = i+1)$ {

$u_i = 0;$

$t_i = (h(m_i) \cdot u^{m_i})^{\alpha} \in G;$

$s_i = H(h(m_i) \| \mathcal{F}(u_i) \| i);$ }

$U = \{u_1, ..., u_n\}$; $T = \{t_1, ..., t_n\}$; $S = \{s_1, ..., s_n\}$;

Construct MHT for F; $sig_{sk}(h(R)) \leftarrow (h(R))^{\alpha}$;

Output (F, pk, MHT, T, S, $sig_{sk}(h(R))$) to CSS; Output (U, k) to TPC;

Challenge Phase:

$c \xleftarrow{R} [1, n]$; $k_1 \xleftarrow{R} \{0,1\}^K$, $k_2 \xleftarrow{R} \{0,1\}^K$;

GenChal$(c, k_1, k_2) \rightarrow$ chal

for $(j = 1;\ j \le c;\ j = j+1)$ {

$i_j = \pi_{k_1}(j);$

$a_j = f_{k_2}(j);$

Output chal$\{(k_1, k_2, c)\}$ to CSS;

GenProof$(F,\ T,\ S,$ chal$) \rightarrow P$:

$\gamma \xleftarrow{R} \mathbb{Z}_p$; $\Re = e(u,v)^{\gamma} \in G_T$;

for $(j = 1;\ j \le c;\ j = j+1)$ {

$i_j = \pi_{k_1}(j);$

$a_j = f_{k_2}(j);$ }

Search the signature array to find s_{i_j} and find its corresponding tag $t_{i_j'}$ and block $m_{i_j'}$;

$s = s_{i_1} \cdot ... \cdot s_{i_c}$; $\tau = t_{i_1'}^{a_1} \cdot ... \cdot t_{i_c'}^{a_c}$;

$\mu' = a_1 m_{i_1'} + ... + a_c m_{i_c'}$; $\mu = \gamma + \mu' \bmod p$;

Randomly choose one selected block's $\Omega_i, i_1' \le i \le i_c'$;

Output $P = \{\mu, \tau, \Re, s, h(m_i)_{i_1' \le i \le i_c'}, \Omega_i\}$ to TPC;

CheckProof$(g,\ \alpha,$ chal$,\ P,\ U) \rightarrow$ {"success","failure"}:

Generate root R using $\{h(m_i), \Omega_i\}$;

if $e(sig_{sk}(h(R)), g) \ne e(h_R, g^{\alpha})$; output failure, otherwise search the U array with index $\{i_1, ..., i_c\}$ for values $\{u_{i_1}, ..., u_{i_c}\}$;

$s^* = H(h(m_{i_1'}) \| \mathcal{F}(u_{i_1}) \| i_1) \cdot ... \cdot H(h(m_{i_c'}) \| \mathcal{F}(u_{i_c}) \| i_c)$;

if $\Re \cdot e(\tau, g) == e(\prod_{i=s_1}^{s_c} h(m_i)^{a_i} \cdot u^{\mu}, v)$ && $s^* = s$, output success; else output failure.

图 6-9 基于 MHT 的动态数据完整性验证算法描述

2）挑战阶段

第三方检查者通过验证存储服务器中 c 个不同数据块的完整性证明以验证服务器所存储数据文件 F 的完整性，其中 $S=\{s_1,\ldots,s_n\}$。第三方检查者分别为 $\pi_{(\cdot)}(\cdot)$ 和 $f_{(\cdot)}(\cdot)$ 产生密钥 k_1 和 k_2。第三方检查者运行 GenChal 并将所得 $\text{chal}\{(k_1,k_2,c)\}$ 发送至 DIMA 存储服务器。在 c 轮 $\pi_{(\cdot)}(\cdot)$ 和 $f_{(\cdot)}(\cdot)$ 调用过程中，第三方检查者与存储服务器均能够确定这轮挑战中所选数据块的位置及相应系数。

存储服务器根据所接收到的挑战 $\text{chal}\{(k_1,k_2,c)\}$ 和存储文件 F^*，运行 $\text{GenProof}(\bullet)$ 算法。在此算法中，存储服务器首先通过 chal 确定所要求的 c 个数据块的位置及相应系数，对 c 个数据块计算完整性证明 P；并将 P 发送至第三方检查者。

第三方检查者在接收到证明 P 后，运行 $\text{CheckProof}(\bullet)$ 算法以验证 P 的正确性。验证等式的的正确性证明如公式（6-12），方案的具体细节如图 6-9 中所示。

3）更新阶段

基于 MHT 的动态数据完整性验证算法也能够有效地处理数据动态操作，包括：插入操作、修改操作以及删除操作，以下将分别对这三种操作进行详细说明：

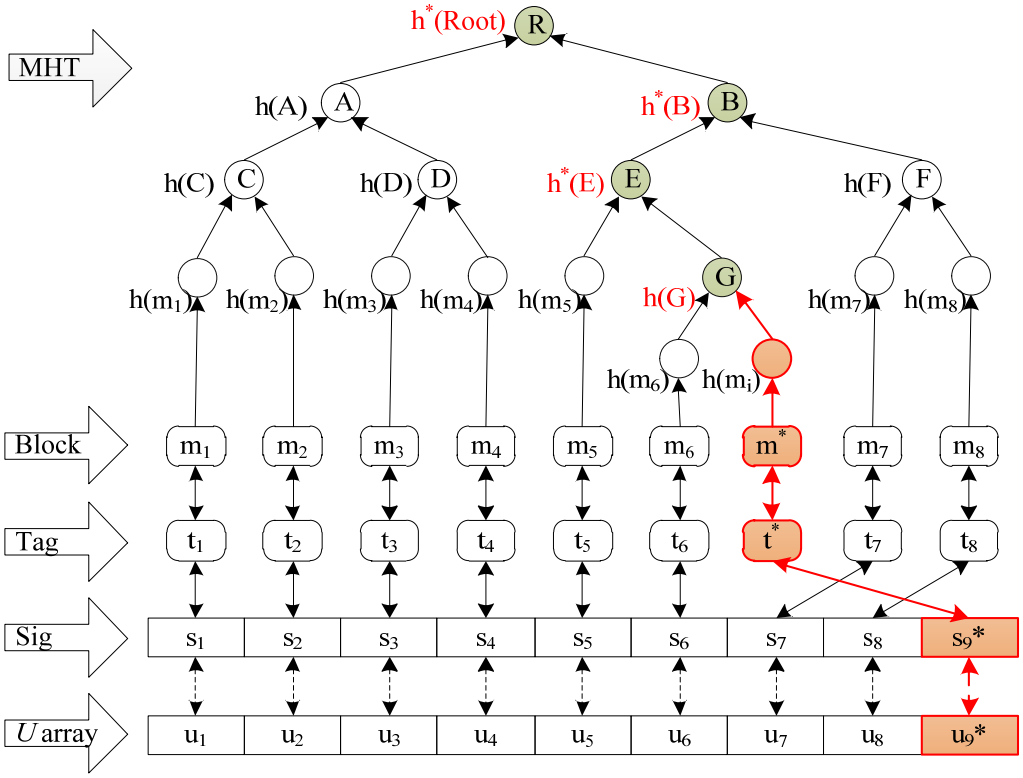

图 6-10　插入操作示例图

插入操作：在第 j 块数据后插入新数据块 t_{n+1}^* 时，用户向 U 值数组结尾处增加一项 u_{n+1}^* 并初始其为 0（假设插入操作前 MHT 中含有 n 块数据）。用户计算新数据块 t_{n+1}^* 相应的签名 $s_{n+1}^* = H(h(m^*) \| \mathcal{F}(u_{n+1}^*) \| n+1)$ 和标签 $t^* = (h(m^*) \cdot u^{m^*})^{\alpha}$，并将 (j, m^*, t^*, s_{n+1}^*) 发送至存储服务器。按照所收到的要求，存储服务器为更新后的 MHT 生成新的根节点 R'，并向

用户返回一个证明 $P_{update}=(\Omega_j,h(m_j),R')$。在接收到存储服务器发回的证明后，用户首先使用 $(h(m_j),\Omega_j)$ 生成根值 R 并检查 $e(sig_{sk}(h(R)),g)?=e(h_R,g^{\alpha})$ 以验证 API 信息。若等式不成立，则返回“fail”；否则，用户使用 $(h(m_j),h(m^*),\Omega_j)$ 计算新的根值并与 R' 相比较，以检查存储服务器是否按照要求完成了插入操作。如果等式成立，用户对新的根元数据 $sig_{sk}(h(R'))\leftarrow(h(R'))^{\alpha}$ 进行签名，并将所得结果 $sig_{sk}(h(R'))$ 发送至存储服务器；否则输出“fail”。最后，用户对 m_i^* 执行数据完整性验证，若输出为“success”，用户将删除本地存储的 P_{update} 以及 m_i^*，并向 TPC 发送消息 $(n+1,0)$。TPC 将在 U 值数组结尾处增加一项 u_{n+1}^* 并赋初值“0”。

图 6-10 描述了在叶子节点 $h(m_6)$ 中后插入 $h(m^*)$ 的具体例子。在此例中，仅有节点 $h(m^*)$ 和内部节点 $h(G)$ 被加入初始 MHT，其中 $h(G)=h(h(m_6)\|h(m^*))$。

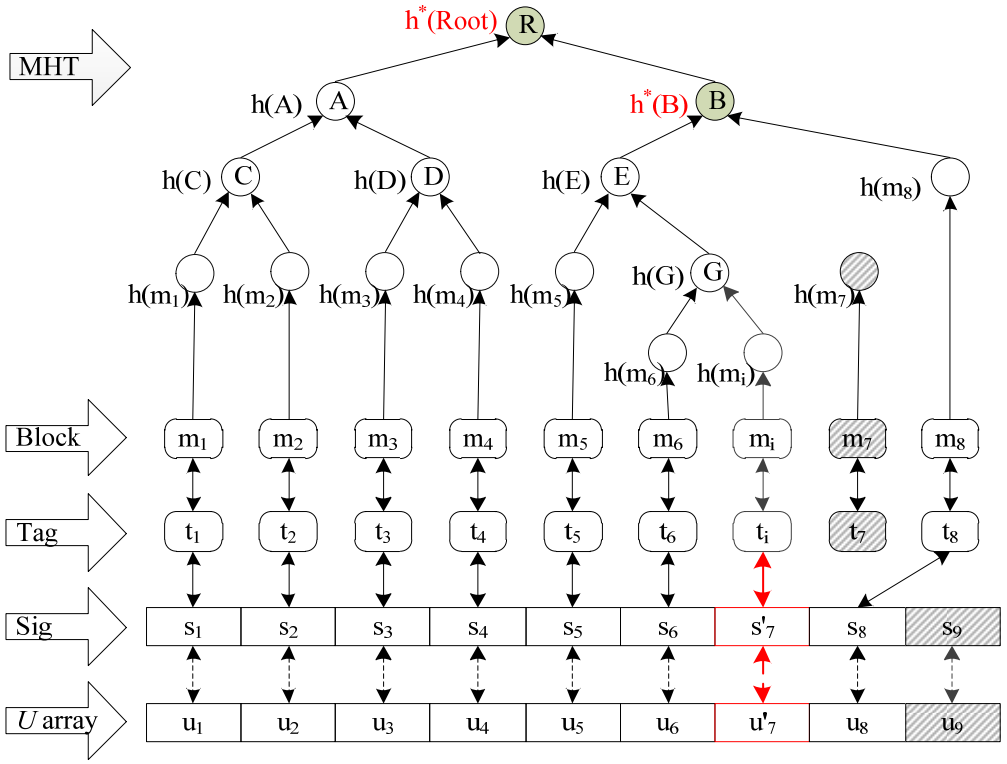

图 6-11 删除操作示例图

删除操作：当删除第 i 个数据块 m_i 时，用户首先将数据块索引号 i 发送至 DIMA。存储服务器在 MHT 中查找存储第 i 个叶子节点 $h(m_i)$ 及其 API 信息 Ω_i。根据指针信息，存储服务器同时查找数据块 m_i、标签 t_i 和相应的签名 $s_{i'}$ 及其索引号 i'，并在完成查找后，删除所找到的 m_i、$h(m_i)$ 和 t_i。完成删除后，存储服务器为更新后的 MHT 生成新的根节点 R'，并向用户返回一个更新证明 $P_{update}=(i',h(m_i),\Omega_i,R')$。在接收到存储服务器发回的证明后，用户首先使用 $(h(m_i),\Omega_i)$ 生成根值 R 并检查 $e(sig_{sk}(h(R)),g)?=e(h_R,g^{\alpha})$ 以验证 API 信息。若等式不成立，则返回“fail”；否则，用户使用 (Ω_i) 计算新的根值并与 R' 相比较。若结果一致，用户对新的根值进行签名 $sig_{sk}(h(R'))\leftarrow(h(R'))^{\alpha}$，并将所得结果

$sig_{sk}(h(R'))$发送至存储服务器；否则输出“fail”。存储服务器查找签名数组中的最后一个值s_n及其相应的标签$t_{n'}$，并向用户发送消息$(s_n, t_{n'})$。用户在 U 值数组中查找u_n并检查$s_n ?= H(h(m_{n'}) \| \mathcal{F}(u_n) \| n)$。若等式成立，用户在 U 值数组中查找$u_{i'}$，若$u_{i'} > u_n$，更新$u_{i'}^* = u_{i'} + 1$；否则更新$u_{i'}^* = u_n + 1$。应用更新后的$u_{i'}^*$，用户计算$s_{i'}^* = H(h(m_{n'}) \| \mathcal{F}(u_{i'}^*) \| i')$，在将$s_n$的指针赋值给$s_{i'}^*$后，向将$s_{i'}^*$发送至存储服务器。在接收到$s_{i'}^*$后，存储服务器更新$s_{i'} \to s_{i'}^*$并删除$s_n$；继而在签名数组中查找第 i 个签名$s_{i'}$及其相应的数据块$m_{i''}$、标签$t_{i''}$，返回消息$(m_{i''}, t_{i''}, s_{i'}, i')$至用户。最后，用户对$e(t_{i''}, g) ?= e(h(m_{i''}) \cdot u^{m_{i''}}, v)$和$s_{i'} ?= H(h(m_{i''}) \| \mathcal{F}(u_{i'}) \| i')$进行检查以验证数据块的完整性。若以上两等式均成立，用户输出“success”并删除本地存储的P_{update}，同时向 TPC 发送消息$(i', u_{i'}^*)$。TPC 将对 U 值数组中的$u_{i'}$进行更新$u_{i'} \to u_{i'}^*$以完成修改操作。

图 6-11 描述了一个具体的删除操作。

修改操作：当需要修改第 i 个数据块m_i为t_{n+1}^*时，用户需将数据块索引号 i 发送至存储服务器。存储服务器在 MHT 中查找存储第 i 个叶子节点$h(m_i)$及其 API 信息Ω_i。根据所存储的指针信息，存储服务器同时查找数据块m_i、标签t_i相应的签名$s_{i'}$及其索引号i'，并在完成查找后，将所得相关数据$(m_i, t_i, s_{i'}, i', h(m_i), \Omega_i)$返回给用户。用户使用$(h(m_i), \Omega_i)$生成根节点$R$，并检查$e(sig_{sk}(h(R)), g) ?= e(h_R, g^{\alpha})$，以验证 API 信息。若此等式成立，用户继而检查$e(t_i, g) ?= e(h(m_i) \cdot u^{m_i}, v)$，$s_{i'} ?= H(h(m_i) \| \mathcal{F}(u_{i'}) \| i')$，以验证数据块$m_i$的完整性。如果以上两等式均成立，用户将修改$u_{i'}^* = u_{i'} + 1$，$m_i \to m_i^*$并分别生成$s_{i'}^* = H(h(m_i^*) \| \mathcal{F}(u_{i'}^*) \| i')$和$t_i^* = (h(m_i^*) \cdot u^{m_i^*})^{\alpha}$。更新后的$(m_i^*, t_i^*, s_{i'}^*)$将发送至存储服务器。以上过程中的任一等式不成立，则用户输出修改失败。依据所接收到的数据，存储服务器将完成相应的更新：$m_i \to m_i^*$，$t_i \to t_i^*$，$s_{i'} \to s_{i'}^*$，以及 MHT 叶子节点$h(m_i) \to h(m_i^*)$，并重新生成 MHT 的根节点R'，作为修改操作的证明返回给用户，可得$P_{update} = (R')$。用户使用$(h(m_i^*), \Omega_i)$计算新的根值并与R'相比较，若相等，对新的根节点完成签名计算$sig_{sk}(h(R')) \leftarrow (h(R'))^{\alpha}$，并将所得的$sig_{sk}(h(R'))$发送至存储服务器；若不相等，则修改操作失败。最后，用户对m_i^*执行数据完整性验证，若输出为“success”，用户将删除本地存储的P_{update}，$(m_i, t_i, s_{i'}, i', h(m_i), \Omega_i)$以及$m_i^*$，并将$(i', u_{i'}^*)$发给 TPC。TPC 将对 U 值数组中的$u_{i'}$进行更新$u_{i'}^* = u_{i'} + 1$，以完成修改操作。

6.3.3 完备性分析

定理 6.1（离散对数问题）给定$g, g^x \in G$，对于未知的$x \in \mathbb{Z}_p^*$，不存在以不可忽视的概率计算 x 的概率多项式时间算法。

定理 6.2（计算 Diffie-Hellman 问题）给定$g, g^x, g^y \in G$，对于未知的$x, y \in \mathbb{Z}_p^*$，不存在以不可忽视的概率计算g^{xy}的概率多项式时间算法。

定理 6.3 不存在以不可忽视的概率伪造一个双线性对签名的概率多项式时间算法[136]。

定理 6.4 给定一个密码哈希函数 $H(\cdot)$，不存在以不可忽视的概率查找两个字符串 $m_1 \neq m_2$ 且 $H(m_1) \neq H(m_2)$ 的概率多项式时间算法。

根据以上定理能够容易地证明，在存储服务器没有完整存储数据的情况下无法向 TPC 生成有效的持有性证明。同时，也保证了 TPC 无法从回馈信息中获取任何数据块内容的信息。

6.3.4 正确性分析

第三方检查者在接收到证明 P 后，运行 $\mathrm{CheckProof}(\bullet)$ 算法以验证 P 的正确性。验证等式的正确性证明如公式（6-12），方案的具体细节如图 6-15 中所示。

$$\begin{aligned}\Re \cdot e(\tau, g) &= e(u,v)^{\gamma} \cdot e((\prod_{i=s_1}^{s_c} h(m_i) \cdot u^{m_i})^{\alpha \cdot a_i}, g) \\ &= e(u^{\gamma}, v) \cdot e(\prod_{i=s_1}^{s_c} h(m_i)^{a_i} \cdot u^{m_i \cdot a_i}, g^{\alpha}) \\ &= e(u^{\gamma}, v) \cdot e(\prod_{i=s_1}^{s_c} h(m_i)^{a_i} \cdot u^{\mu'}, v) \\ &= e(\prod_{i=s_1}^{s_c} h(m_i)^{a_i} \cdot u^{\mu'+\gamma}, v) \\ &= e(\prod_{i=s_1}^{s_c} h(m_i)^{a_i} \cdot u^{\mu}, v)\end{aligned} \qquad (6\text{-}12)$$

6.3.5 适用性分析

基于 Merkle 哈希树的动态数据完整性验证方案中，TPC 在每次挑战时都是随机计算选取数据块索引值，并不固定且不具有确定性，因而该方案并不受验证次数的限制。

由于该方案中采用的双线性签名是依据公钥加密基础算法，而在公钥加密系统中,只要知道公开密钥,就可以进行相关的验证工作。因而,所提出的验证方案支持公开验证。

该方案中并未对数据进行容错纠错设计，无法使数据在遭受恶意删改后完成恢复。但可采用 5.2 小节所提出的纠删码技术在数据存储预处理阶段进行编码操作，以弥补恢复性的缺失。

6.4 基于 MHT 的动态数据完整性验证方案性能分析与实验

为了有效地提高数据存储的安全性及数据存储结构的有效性，本节提出了基于双线性对映射签名和 Merkle 哈希树的动态数据完整性验证算法。

6.4.1 计算开销

在初始化阶段，仅用户需要对每一数据块计算其签名和标签值，并对文件数据块构建 MHT。TPC 与存储服务器在此阶段没有计算开销。

在挑战阶段：存储服务器与 TPC 均需要通过运行 c 轮 PRP $\pi_{(\cdot)}(\cdot)$ 和 PRF $f_{(\cdot)}(\cdot)$ 以确定所选择的数据块及其系数。存储服务器生成的数据持有性证明包含一个随机系数 $\Re = e(u,v)^{\gamma} \in G_T$、一个所选取数据块的盲线性组合（blinded linear combination）

$\mu = \sum_{i=1}^{c} a_i m_i + \gamma \in \mathbb{Z}_p$、以及一个聚合验证 $\tau = \prod_{i=1}^{c} t_i^{a_i} \in G$。以上几项的计算开销分别为：$Exp_{G_T}(|p|)$、$(c+1)\cdot Add_{\mathbb{Z}_p}$、$c\cdot Mult_{\mathbb{Z}_p}$、$c\cdot MultExp_G$。

在接收到检查回复 $\mu,\ \tau,\ \Re,\ s,\ h(m_i)_{i_1^{'}\le i\le i_c^{'}},\ \Omega_i$ 后，首先需要进行 MHT 根节点有效性验证，其计算开销为 $\log(n)\cdot hash_G$ 和 $2\cdot Pair_{G,G}$。对于数据块完整性验证的计算开销则为 $2\cdot Pair_{G,G} + c\cdot MultExp_G(|a_i|) + Exp_G(|p|) + Mult_G + Mult_{G_T} + c\cdot Hash + c\cdot Mult$。

在更新阶段：用户需要计算所更新数据的相关信息并验证更新结果。验证过程中的计算开销包括根节点验证和更新数据块验证两部分。其中根节点验证同挑战阶段，而更新数据块验证需要进行 $2\cdot Pair_{G,G} + c\cdot MultExp_G(|a_i|) + Exp_G(|p|) + Mult_G + Mult_{G_T} + Hash$ 的计算。存储服务器仅需要对更新后的 MHT 计算根值，将花费 $\log(n)\cdot hash_G$。

6.4.2 存储开销

在数据完整性验证过程中，TPC 需要存储密钥和 U 值数组。存储服务器在数据文件之外，需要保存签名集、标签集以及跳表结构。假设 L 长度为 1024 位，一个 4G 字节的文件 F 能够划分为 1,000,000 个 4K 字节大小的数据块。每一个数据块具有长为 128 字节的签名和 20 字节的标签。跳表结构中的节点数量大约为 2,000,000 个，即文件数据块数量的 2 倍。每一跳表节点为 3 字节，因此存储服务器所需的额外存储空间为 154M 字节，即原文件大小的 3.8%。设定 U 值数组每项的大小为 2 字节（可支持 2^{16} = 65536 次数据更新），TPC 为存储 1,000,000 块数据所需的额外空间为 2M 字节，即原文件大小的 0.05% 。

6.4.3 通信开销

在更新阶段：以修改操作为例，存储服务器首先向 TPC 发送 $(m_i, t_i, s_{i'}, i', h(m_i), \Omega_i)$ 以证明对原有数据块的完整持有。若持有性证明有效，TPC 上传 $(m_i^*, t_i^*, s_{i'}^*)$ 至存储服务器。在完成修改后，存储服务器向 TPC 发送 R' 作为应答。若 R' 正确，TPC 向存储服务器回送 $sig_{sk}(h(R'))$ 以用于将来的数据检查。通信开销在更新阶段为常量。

在挑战阶段：挑战消息 $\text{chal}\{(k_1, k_2, c)\}$ 为常量，而证明 $P = (\mu, \tau, \Re, s, h(m_i)_{i_1^{'}\le i\le i_c^{'}}, \Omega_i)$ 的大小取决于抽查数据块数量 c。

由 K. D. Bowers 等人[81]提出的方案存在一个严重的安全问题，即被检查方能够使用任意的数据块组合计算数据持有性证明。继而，为了证明数据是随机选取的，被检查方必须向检查者提供附加的验证信息。对于 MHT，附加验证信息将会给 TPC 和 DIMA 均造成 $c\cdot\log(n)\cdot Hash_G$ 的额外存储和计算开销。本节所提出的算法使用签名数组以保障标签集和数据集的完整性。存储服务器能够使用 U 值数组保证持有性证明是由选定的数据块产生的。通过以上方式，对于位置信息验证的存储和计算开销将减少为 $c\cdot Hash_G$。假

设每一数据块的u_i为 2Byte（可支持 65536 次更新操作），则 U 值数组仅为原始数据大小的 1%。

6.5 本章小节

本节在 5.1 小节提出的基于第三方的数据完整性检查模型基础上，分别采用跳表和梅克尔散列树数据存储结构进行数据存储，结合代数签名技术和双线性映射签名技术，提出了两种适用于 DIMA 系统中动态数据的完整性验证方案。所提出的两种方案在验证过程均不需要与原始数据进行比较，存储、计算与传输开销均为常量级，节省了大量的带宽资源和计算资源。采用跳表存储结构并结合代数签名的方案，由于采用了非公钥密码技术效率较高；而采用梅克尔散列树存储结构并结合双线性对映射的方案具有安全可证明性和隐私保护性。

7 总结和展望

7.1 工作总结

DIMA 系统作为新一代飞机的航空电子系统的发展方向，在设计应用过程中存在着各方面的问题和挑战。本文以 DIMA 系统为研究对象，主要研究了系统网络传输中的实时性和时间确定性，网络存储中数据的完整性验证等关键问题，所做研究工作总结如下：

（1）适用于 DIMA 的时间触发 AFDX 网络系统结构设计

本文分析了标准 AFDX 网络协议，对 AFDX 网络的拓扑结构以及端系统、交换机进行了研究，重点研究了 AFDX 中的虚链路，通过研究 AFDX 网络协议，得出将时间触发机制引入 AFDX 网络具有可行性的结论。将时间触发机制引入 AFDX 网络中，设计了时间触发 AFDX 网络的协议，并对 AFDX 网络数据帧中的部分内容进行了修改，使其可以兼容时间触发机制，加入时间触发机制之后，AFDX 网络中的虚链路被分为时间触发虚链路和流量约束虚链路。

（2）设计了时间触发 AFDX 网络中时间触发虚链路的调度算法

时间触发 AFDX 网络中时间触发虚链路依照静态配置好的时间调度表对数据流进行服务调度。本文分别提出了基于周期优先原则和基于帧长度优先原则的调度表设计方法，并说明了在采用这两种调度表情况下，端系统及交换机系统中的时间触发虚链路调度算法。基于周期优先原则的调度表设计方法相对简单，而帧长度优先调度表设计方法能够在保证时间触发流量无发送冲突的同时，有效解决由于各数据帧长度存在较大差异时带来的带宽利用率问题。搭建典型网络，通过网络演算方法将 TTAFDX 分别与 FIFO 调度算法和静态优先级调度算法的 AFDX 网络的实时性能进行对比。计算结果表明：TTAFDX 中的时间触发虚链路的延迟主要由固定延迟部分组成，而速率限制虚链路的实时性较 AFDX 中的低优先级虚链路也有所提高。证明了：提出的 TTAFDX 体系结构和调度算法在兼容 AFDX 的同时，能够改善网络的时间确定性。

（3）设计了时间触发 AFDX 网络中速率限制虚链路的调度算法

AFDX 网络协议中仅规定了先来先服务（FIFO）调度算法，由于 FIFO 调度不区分数据传输任务的优先级，无法满足时间关键任务的需求。虽然时间触发 AFDX 网络对速率限制虚链路的实时性能也有所提高，为了进一步提高时间触发 AFDX 网络的性能，针对时间触发 AFDX 网络中速率限制链路提出了具有低复杂度、高公平性和低调度延时特性的逐次最小定额轮询（Successive Minimal-Quantum Round Robin, SMQRR）调度机制。SMQRR 调度算法通过改变对每一个数据流权值所对应的数据量的连续服务方式，在保留轮询调度算法 $O(1)$ 时间复杂度的同时，有效克服了分组长度不同引起的不公平性并避

免了数据流可能长时间无法获得服务的情况。通过理论分析和 NS2 仿真验证：SMQRR 调度算法在公平性和时延上界均优于现有轮询调度算法。

（4）DIMA 系统中静态数据完整性验证方案

针对 DIMA 系统中网络数据存储所面临的安全问题，提出了基于第三方的数据完整性检查模型，该模型以可信第三方为核心，采用了“挑战—响应”检查模式。在此模型基础上，设计了基于代数签名技术的静态数据完整性检查方案，能够极大降低通讯开销，在数据遭到破坏的情况下，以高概率检查出数据错误。通过对数据采用前向纠错编码，提高了数据的完整性保障。安全性分析和实验结果表明：所提方法是安全可证明的。

（5）DIMA 系统中动态数据完整性验证方案

提出了基于跳表和基于梅克尔散列树存储结构的数据完整性验证方案。在航空电子数据存储过程中采用跳表结构或梅克尔散列树结构，能够支持数据的动态操作。应用双线性对映射签名技术，在为数据完整性检查提供了数据隐私保护的同时，系统开销较代数签名技术较大，但不影响系统传输和计算性能。所提出的数据完整性检查方案无需与原始数据相比较，且验证不受次数限制。安全性分析和实验结果表明：所提方法是安全可证明的。

（6）研究成果应用及验证

通过将时间触发 AFDX 的周期优先和帧长度优先调度算法、逐次最小定额轮询（SMQRR）调度机制和存储数据的完整性验证方法等部分论文研究成果及其思路应用于国防基础预研、民机预研以及部分型号任务子课题中，提高了航空电子系统中数据传输的实时性和时间确定性，保证了数据存储的可靠性和安全性，验证了论文研究成果的实用性。

7.2 进一步研究方向

本文的出发点是提高 DIMA 系统中网络数据的传输及存储可靠性，以适应航空电子系统的需求。结合本文的研究情况，后续研究可从以下方面着手：

（1）降级通信

本文在分析时间触发 AFDX 网络的过程中，是在时钟同步的大前提下，在实际的应用过程中，可能会出现时钟漂移的情况，这样就有可能造成大量的冲突而导致数据帧的丢失，在这种情况下，可以使发生冲突的时间触发虚链路降级为流量约束虚链路，后续可以研究这方面的调度情况。

（2）时钟同步

本文是在时钟同步的前提下进行时间触发调度算法的设计的，没有对时钟同步的相关问题进行研究。时钟同步是时间触发机制中的一个重要概念，如何减少同步信息给系统带来的额外负载，可以作为后续的一个研究点来进行研究。

（3）网络演算技术

确定性网络演算技术仅能获得网络时延的最坏边界，在深入地分析时间触发 AFDX 网络架构、协议模型和流量情况的基础上，优化网络演算算法或采用随机网络演算能够更加精确地计算和评估网络性能。

（4）认证字典技术

动态数据完整性验证方法都基于不同的认证字典结构，构建更加高效的认证字典结构，能够提高数据查找和操作的效率，降低开销，减少耗时，后续研究可以针对认证字典技术进行。

（5）身份认证技术

DIMA 系统中网络数据的完整性验证只能被动地检查出存储数据是否被恶意删改，如何有效地主动地监管用户的操作，妥善地限制用户权限与完整性检查互相支持，提高系统网络数据生存周期中的可靠性和安全性。

参考文献

[1] Wolfig R, Jakovljevic M. Distributed IMA and DO-297: Architectural, communication and certification attributes[C]. Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th, 2008: 1. E. 4-1-1. E. 4-10.

[2] 张英静, 熊华钢, 刘志丹等. 可用于航空电子系统的时间触发以太网[J]. 电光与控制, 2015, (05): 49-53.

[3] Aircraft Data Network Part 7: Avionics Full Duplex Switched Ethernet (AFDX) Network[J]. ARINC Specification 664p7, 2005.

[4] 周强, 熊华钢. 新一代民机航空电子互连技术发展[J]. 电光与控制, 2009, (04): 1-6.

[5] 赵宁社, 翟正军, 王国庆. 新一代航空电子综合化及预测与健康管理技术[J]. 测控技术, 2011, 30(1): 1-5,9.

[6] 郑娟. 航空电子系统综合技术的发展与模块化趋势[J]. 信息通信, 2014, (04): 285.

[7] 张凤鸣, 褚文奎, 樊晓光等. 综合模块化航空电子体系结构研究[J]. 电光与控制, 2009, (09): 47-51+59.

[8] 罗海明, 谢剑斌, 陆志肖. 机电系统综合化控制和管理[J]. 直升机技术, 2010, (1): 62-67.

[9] 朱晓飞, 黄永葵. 综合模块化航空电子系统标准分析及发展展望[J]. 航空电子技术, 2010, 41(4): 17-22.

[10] 王鹏, 刘锐, 刘万和等. 综合模块化航空电子系统可靠性评估方法研究[J]. 电光与控制, 2015, (10): 56-61.

[11] 许晋瑞, 李峭, 赵露茜等. DIMA 系统实时通信流量的时延分析方法[J]. 计算机工程与设计, 2015, (04): 879-885.

[12] 徐黎, 庞瑞帆, 张怡等. 攻击直升飞机通信导航识别系统的分布式综合模块化航空电子设备综合技术研究[J]. 上海交通大学学报, 2012, 46(5): 756-761.

[13] 王国庆, 谷青范, 王淼等. 新一代综合化航空电子系统构架技术研究[J]. 航空学报, 2014, (06): 1473-1486.

[14] 李浩峰. AFDX 网络测试研究现状分析[J]. 一重技术, 2009, (2): 64-66.

[15] Charara H, Scharbarg J, Ermont J, et al. Methods for bounding end-to-end delays on an AFDX network[C]. Real-Time Systems, 2006. 18th Euromicro Conference on, 2006: 10 pp.-202.

[16] Cruz R L. A calculus for network delay. I. Network elements in isolation[J].

Information Theory, IEEE Transactions on, 1991, 37(1): 114-131.

[17] Cruz R L. A calculus of delay Part II: Network analysis[J]. IEEE Trans. Inform. Theory, 1991, 37(1): 132-141.

[18] Thiele L, Chakraborty S, Naedele M. Real-time calculus for scheduling hard real-time systems[C]. Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on, 2000: 101-104.

[19] Wang G, Liu J. Network Calculus using in real-time Industrial Ethernet[C]. Information Science and Engineering, 2008. ISISE'08. International Symposium on, 2008: 77-79.

[20] 张奇智, 张彬, 张卫东. 基于网络演算计算交换式工业以太网中的最大时延[J]. 控制与决策, 2005, (01): 117-120.

[21] 杨云, 熊华钢. 计算 AFDX 延迟的网络演算方法[J]. 电光与控制, 2008, (09): 57-60.

[22] 赵永库, 王红春, 唐来胜. AFDX 网络端到端时延分析方法[J]. 电光与控制, 2013, 20(4): 81-83.

[23] Jiang Y. A basic stochastic network calculus[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(4): 123-134.

[24] Jiang Y, Liu Y. Stochastic network calculus[M]. 1. Springer, 2008.

[25] Ridouard F, Scharbarg J, Fraboul C. Stochastic network calculus for end-to-end delays distribution evaluation on an avionics switched Ethernet[C]. Industrial Informatics, 2007 5th IEEE International Conference on, 2007: 559-564.

[26] Scharbarg J-L, Ridouard F, Fraboul C. A probabilistic analysis of end-to-end delays on an AFDX avionic network[J]. Industrial Informatics, IEEE Transactions on, 2009, 5(1): 38-49.

[27] Malta L, Da Silva Oliveira R. A Model to Calculate Exact End-to-End Delay of Sporadic Flows on AFDX Network Using Mathematical Programming[C]. Computing System Engineering (SBESC), 2012 Brazilian Symposium on, 2012: 87-92.

[28] Bauer H, Scharbarg J, Fraboul C. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach[J]. Industrial Informatics, IEEE Transactions on, 2010, 6(4): 521-533.

[29] Bauer H, Scharbarg J-L, Fraboul C. Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources[J]. Real-time systems, 2012, 48(1): 101-133.

[30] 王平, 卢选民, 陈文刚. 基于主动策略的 AFDX 智能网络管理模型研究[J]. 计

算机测量与控制, 2010, 18(9): 2178-2180.

[31] 陈文刚, 卢选民, 单长等. 基于混合策略的 AFDX 分布式网络管理模型研究与实现[J]. 计算机测量与控制, 2011, 19(9): 2266-2268.

[32] Al Sheikh A, Brun O, Chéramy M, et al. Optimal design of virtual links in AFDX networks[J]. Real-Time Systems, 2013, 49(3): 308-336.

[33] An D, Jeon H W, Kim K H, et al. A feasible configuration of AFDX networks for real-time flows in avionics systems[J], 2013.

[34] 陈昕, 周拥军, 蒋文保等. AFDX 协议性能分析及调度算法研究[J]. 电子学报, 2009, 37(5): 1000-1005.

[35] 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.

[36] Suthaputchakun C, Lee K M B, Sun Z. Impact of End System scheduling policies on AFDX performance in avionic on-board data network[C]. 2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA), 2015: 1-6.

[37] Zheng X, Huang N, Zhang Y, et al. Performability optimization design of virtual links in AFDX networks[C]. 2016 Annual Reliability and Maintainability Symposium (RAMS), 2016: 1-6.

[38] Gurjar S, Lakshmi B. Optimal scheduling policy for jitter control in AFDX End-System[C]. Recent Advances and Innovations in Engineering (ICRAIE), 2014, 2014: 1-4.

[39] Kos A, Miletic J, Tomazic S. Improved Latency Bound of Deficit Round Robin Scheduler[J]. Proceedings of YU INFO, 2007.

[40] Zhang X, Bhuyan L N. Deficit round-robin scheduling for input-queued switches[J]. Selected Areas in Communications, IEEE Journal on, 2003, 21(4): 584-594.

[41] Kanhere S S, Sethu H. Fair, efficient and low-latency packet scheduling using nested deficit round robin[C]. High Performance Switching and Routing, 2001 IEEE Workshop on, 2001: 6-10.

[42] Kanhere S S, Sethu H. Low-latency guaranteed-rate scheduling using elastic round robin[J]. Computer Communications, 2002, 25(14): 1315-1322.

[43] Kanhere S S, Sethu H, Parekh A B. Fair and efficient packet scheduling using elastic round robin[J]. Parallel and Distributed Systems, IEEE Transactions on, 2002, 13(3): 324-336.

[44] 董民, 沈庆国. 轮循类分组调度算法的性能研究[J]. 系统仿真学报, 2010, (11):

2593-2596.

[45] Semeria C. Supporting differentiated service classes: queue scheduling disciplines[J]. Juniper networks, 2001: 11-14.

[46] 江文静, 蔡祥宝. DiffServ 队列调度算法研究[J]. 计算机技术与发展, 2015, (4): 85-88.

[47] 刘文波, 郭云飞, 马海龙. 一种基于紧急程度的自时钟开始时间公平排队分组调度算法[J]. 电子与信息学报, 2010, 32: 1452-1456.

[48] Flores R T, Boyer M. Performance analysis of the Disrupted Static Priority scheduling for AFDX[C]. Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on, 2014: 94-103.

[49] Fang Y, Hua Y, Liu X. Efficient end-to-end communication services for mixed criticality avionics systems[C]. Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of, 2014: 188-197.

[50] Fischer M J, Masi D M B, Shortle J F. Simulating the performance of a class-based weighted fair queueing system[C]. Proceedings of the 40th Conference on Winter Simulation, 2008: 2901-2908.

[51] Georges J-P, Divoux T, Rondeau E. Strict Priority versus Weighted Fair Queueing in Switched Ethernet networks for time critical applications[C]. Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, 2005: 141-141.

[52] 李秉权, 张松, 王兆伟等. WFQ 与 WRR 调度算法的性能分析与改进[J]. 北京理工大学学报, 2015, (03): 316-320.

[53] Chiussi F M, Francini A. Minimum-delay self-clocked fair queueing algorithm for packet-switched networks[C]. INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 1998: 1112-1121.

[54] Golestani S J. A self-clocked fair queueing scheme for broadband applications[C]. INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE, 1994: 636-646.

[55] Goyal P, Vin H M, Cheng H. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks[J]. Networking, IEEE/ACM Transactions on, 1997, 5(5): 690-704.

[56] Bennett J C, Zhang H. WF^2Q: Worst-case fair weighted fair queueing[C]. INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, 1996: 120-128.

[57] 赵罡, 何锋, 徐亚军等. 时间触发总线流量调度机制及其实时性分析[J]. 计算

机工程, 2015, (10): 59-65.

[58] Cummings R, Richter K, Ernst R, et al. Exploring Use of Ethernet for In-Vehicle Control Applications: AFDX, TTEthernet, EtherCAT, and AVB[J]. SAE International Journal of Passenger Cars-Electronic and Electrical Systems, 2012, 5(1): 72-88.

[59] Li Q R, Wang H C, Han W. Real-Time Fault-Tolerant Ethernet Technology-TTEthernet and its Feature[J]. Applied Mechanics and Materials, 2013, 321: 2711-2714.

[60] Kopetz H. The rationale for time-triggered ethernet[C]. Real-Time Systems Symposium, 2008, 2008: 3-11.

[61] Kopetz H, Ademaj A, Grillinger P, et al. The time-triggered ethernet (TTE) design[C]. Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on, 2005: 22-33.

[62] 刘帅, 张喜民, 郭鹏. TTE 通信技术在混合安全关键系统的应用[J]. 航空计算技术, 2013, 43(2): 120-122,127.

[63] 朱闻渊, 尹家伟, 蒋祺明. 新型航空电子系统总线互连技术发展综述[J]. 计算机工程, 2011, (S1): 398-402.

[64] Zhao L, Xiong H, Zheng Z, et al. Improving Worst-Case Latency Analysis for Rate-Constrained Traffic in the Time-Triggered Ethernet Network[J], 2014.

[65] Zheng Z, He F, Xiong Y. The research of scheduling algorithm for time-triggered ethernet based on path-hop[C]. 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016: 1-6.

[66] Bingqian L, Yong W. Hybrid-GA based static schedule generation for time-triggered ethernet[C]. 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), 2016: 423-427.

[67] Wisniewski L, Schumacher M, Jasperneite J, et al. Increasing flexibility of Time Triggered Ethernet based systems by optimal greedy scheduling approach[C]. 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), 2015: 1-6.

[68] Li X. Research on the Real-Time Property of TTCAN Protocol in Vehicle Communication Network[J]. Applied Mechanics and Materials, 2014, 487: 674-677.

[69] 冯晓东, 张争明, 张刚. TTCAN 网络调度平台的设计优化[J]. 微电子学与计算机, 2010, 27(3): 67-70.

[70] 孟祥, 曹万科, 林程等. 独立驱动电动汽车 TTCAN 调度策略与特性研究[J]. 北京理工大学学报, 2011, 31(6): 662-665.

[71] 陶震宇, 夏继强, 满庆丰. TTCAN 非周期信息概率延时分析及改进[J]. 仪表技

术与传感器, 2014, (12): 137-141.

[72] 舒领, 康乐峰, 韦勇宇等. 一种柔性容错 TTCAN 协议调度算法的设计与应用[J]. 火力与指挥控制, 2014, (z1): 120-123,126.

[73] Nayak S K, Tripathy S. Privacy Preserving Provable Data Possession for Cloud Based Electronic Health Record System[C]. 2016 IEEE Trustcom/BigDataSE/ISPA, 2016: 860-867.

[74] Yi M, Wang L, Wei J. Distributed data possession provable in cloud[J]. Distributed and Parallel Databases, 2016: 1-21.

[75] 陈兰香. 一种基于同态Hash的数据持有性证明方法[J]. 电子与信息学报, 2011, (09): 2199-2204.

[76] 张立红, 陈晶, 杜瑞颖等. 云存储中支持安全去重的数据完整性验证方法[J]. 计算机工程, 2017, (01): 32-36+42.

[77] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores[C]. Proceedings of the 14th ACM conference on Computer and communications security, 2007: 598-609.

[78] 陈兰香, 许力. 云存储服务中可证明数据持有及恢复技术研究[J]. 计算机研究与发展, 2012, (S1): 19-25.

[79] Ateniese G, Di Pietro R, Mancini L V, et al. Scalable and efficient provable data possession[C]. Proceedings of the 4th international conference on Security and privacy in communication netowrks, 2008: 1-10.

[80] Juels A, Kaliski Jr B S. PORs: Proofs of retrievability for large files[C]. Proceedings of the 14th ACM conference on Computer and communications security, 2007: 584-597.

[81] Bowers K D, Juels A, Oprea A. HAIL: a high-availability and integrity layer for cloud storage[C]. Proceedings of the 16th ACM conference on Computer and communications security, 2009: 187-198.

[82] Yi M, Wei J, Song L. Efficient integrity verification of replicated data in cloud computing system[J]. Computers & Security, 2017, 65: 202-212.

[83] Shah M A, Swaminathan R, Baker M. Privacy-Preserving Audit and Extraction of Digital Contents[J]. IACR Cryptology ePrint Archive, 2008, 2008: 186.

[84] Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing[J]. Parallel and Distributed Systems, IEEE Transactions on, 2011, 22(5): 847-859.

[85] 肖达, 舒继武, 陈康等. 一个网络归档存储中实用的数据持有性检查方案[J]. 计算机研究与发展, 2009, 46(10): 1660-1668.

[86] 焦文喆, 王国庆, 翟正军等. 云存储下基于代数签名的数据持有性检查方法[J]. 东北师大学报(自然科学), 2013, 45(4): 55-61.

[87] Wu S, Zhang Y. Efficient verification of data possession in cloud computing[C]. Cloud Computing and Intelligence Systems (CCIS), 2016 4th International Conference on, 2016: 424-428.

[88] Zhou F, Peng S, Xu J, et al. Identity-Based Batch Provable Data Possession[C]. International Conference on Provable Security, 2016: 112-129.

[89] Kiruthika E, Ravichandran S. Unique Identity Based Provable Data Possession at Unstructured Stores in Multi-Cloud Storage Using IBE[J]. International Journal, 2016, 4(6).

[90] Wang H, He D, Yu J, et al. Incentive and Unconditionally Anonymous Identity-Based Public Provable Data Possession[J]. IEEE Transactions on Services Computing, 2016.

[91] Zha Y, Luo S, Bian J, et al. A novel provable data possession scheme based on geographic location attribute[J]. China Communications, 2016, 13(9): 139-150.

[92] Han S, Liu S, Zhang F, et al. Homomorphic Linear Authentication Schemes from (ε)-Authentication Codes[C]. Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, 2016: 487-498.

[93] 徐洋, 朱丹, 张焕国等. 云环境下基于代数签名持有性审计的大数据安全存储方案[J]. 计算机科学, 2016, (10): 172-176.

[94] 王惠清, 洪志全. 一种基于代数签名的远程数据完整性验证方法[J]. 计算机应用与软件, 2016, (02): 302-306.

[95] Yu Y, Zhang Y, Ni J, et al. Remote data possession checking with enhanced security for cloud storage[J]. Future Generation Computer Systems, 2015, 52: 77-85.

[96] Chen L, Zhou S, Huang X, et al. Data dynamics for remote data possession checking in cloud storage[J]. Computers & Electrical Engineering, 2013, 39(7): 2413-2424.

[97] Xu H, Jiang J, Xu C. An Authentication Data Structure of Provable Data Possession with Dynamic Data Operation in Cloud Computing[C]. Security, Privacy and Anonymity in Computation, Communication and Storage: SpaCCS 2016 International Workshops, TrustData, TSP, NOPE, DependSys, BigDataSPT, and WCSSC, Zhangjiajie, China, November 16-18, 2016, Proceedings 9, 2016: 371-381.

[98] Yu Y, Ni J, Au M H, et al. Improved security of a dynamic remote data possession checking protocol for cloud storage[J]. Expert Systems with Applications, 2014, 41(17): 7789-7796.

[99] Li C, Wang H. Efficient Dynamic Provable Data Possession from Dynamic Binary

Tree[C]. International Conference on Provable Security, 2016: 101-111.

[100] 焦文喆, 翟正军, 王国庆. 时间触发 AFDX 调度设计及实时性分析[J]. 计算机工程, 2016, 42(7): 42-48.

[101] 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, (3): 35-40.

[102] 刘晚春, 李峭, 何锋等. 时间触发以太网同步及调度机制的研究[J]. 航空计算技术, 2011, 41(4): 122-127.

[103] 焦文喆, 翟正军, 王国庆. 时间触发 AFDX 调度策略设计与实时性分析[J]. 山东农业大学学报自然科学版, 2016, (1): 111-117.

[104] 易娟, 熊华钢, 何锋等. TTE 网络流量转换策略及其延时性能保障调度算法研究[J]. 航空学报, 2014, 35(4): 1071-1078.

[105] Xiaomin L, Chen C, Xiaohu Z, et al. Network performance analysis of Time-Triggered Ethernet based on network calculus for DIMA[C]. 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), 2015: 10A3-1-10A3-7.

[106] 罗杰, 霍曼. AFDX通信链路技术及其在航空电子系统的应用[C]. 全国第十届信号与信息处理四届 DSP 应用技术联合学术会议论文集, 2006: 16-21.

[107] 支超有, 李振水, 李育. 大飞机先进机载数据总线 AFDX[C]. 大型飞机关键技术高层论坛暨中国航空学会 2007 年年会论文集, 2007: 265-272.

[108] 吴建鲁, 杨福彪, 刘煜等. AFDX 技术特点及在舰载武器系统中的应用分析[J]. 指挥控制与仿真, 2010, 32(2): 112-115.

[109] 赵永库, 唐来胜. AFDX 网络应用关键技术分析与研究[J]. 测控技术, 2013, 32(4): 86-89,94.

[110] 杜宏伟, 马捷中. 航空电子全双工交换式以太网及其关键技术研究[J]. 测控技术, 2008, 27(12): 65-67.

[111] 许燕婷. AFDX 端系统协议栈虚拟链路层分析及仿真研究[D]. 上海交通大学, 2011.

[112] 杜旭阳. 面向 IMA 的实时操作系统的安全机制[J]. 计算机安全, 2013, (02): 31-34.

[113] Steinbach T, Korf F, Schmidt T C. Real-time Ethernet for automotive applications: A solution for future in-car networks[C]. Consumer Electronics-Berlin (ICCE-Berlin), 2011 IEEE International Conference on, 2011: 216-220.

[114] Mills D. Network Time Protocol (Version 3) specification, implementation and analysis[J], 1992.

[115] Mills D L. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and

OSI[J], 2006.

[116] Lee K, Eidson J C, Weibel H, et al. IEEE 1588-standard for a precision clock synchronization protocol for networked measurement and control systems[C]. Conference on IEEE, 2005: 2.

[117] Hillebrand J, Rahmani M, Bogenberger R, et al. Coexistence of time-triggered and event-triggered traffic in switched full-duplex ethernet networks[C]. Industrial Embedded Systems, 2007. SIES'07. International Symposium on, 2007: 217-224.

[118] 赵永库, 唐来胜. AFDX 网络延迟计算方法[J]. 测控技术, 2013, 32(5): 85-88,94.

[119] 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(续)[J]. 航空标准化与质量, 2008, (2): 20-23.

[120] 刘成, 王彤, 李铮, et al. 时间触发 AFDX 网络的设计和实时性分析[J]. 北京航空航天大学学报, 2013, 39(6): 728-733.

[121] 陈文刚, 卢选民, 单长, et al. 基于 AFDX 自适应优先调度算法的实时性分析[J]. 测控技术, 2011, 30(10): 73-76.

[122] 冯晓东, 果艳红. TTCAN 协议静态调度算法研究与仿真[J]. 计算机仿真, 2008, 25(6): 108-112.

[123] 王红春, 张涛, 牛文生. 面向 AFDX 网络的交换机流量管制算法[J]. 西北工业大学学报, 2012, 30: 301-304.

[124] 王平, 卢选民, 陈文刚. 基于时延分析的 AFDX 智能网络管理模型全局调度算法研究[J]. 计算机测量与控制, 2010, 18(10): 2424-2426,2429.

[125] 谌文涛, 何锋, 周一伟, et al. AFDX 网络虚拟链路时延抖动测试研究[J]. 电光与控制, 2013, 20(11): 93-96.

[126] 刘静. 应用混合队列调度策略的 AFDX 实时性优化研究[J]. 计算机工程, 2015, 41(11): 135-141.

[127] Boudec J-Y L, Thiran P. Network calculus: a theory of deterministic queuing systems for the internet[M]. Springer-Verlag, 2001: 272.

[128] 杨帆, 刘增基, 邱智亮, et al. 一种具有低时延的分组公平循环调度[J]. 电子与信息学报, 2007, (04): 785-788.

[129] 刘桂开, 高蕾. 基于弹性定额值的分组轮询调度算法[J]. 计算机科学, 2013, 40(8): 72-78.

[130] 刘桂开. 用逐次最小权值轮询算法实现公平和低时延分组调度[J]. 系统科学与数学, 2014, (09): 1080-1099.

[131] Stiliadis D, Varma A. Latency-rate servers: a general model for analysis of traffic

scheduling algorithms[C]. INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, 1996: 111-119 vol.1.

[132] Kanhere S S, Sethu H. On the latency bound of deficit round robin[C]. Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on, 2002: 548-553.

[133] 于斌. NS2 与网络模拟[M]. 人民邮电出版社, 2007.

[134] Curtmola R, Khan O, Burns R. Robust remote data checking[C]. Proceedings of the 4th ACM international workshop on Storage security and survivability, 2008: 63-68.

[135] Merkle R C. A certified digital signature[C]. Advances in Cryptology—CRYPTO'89 Proceedings, 1990: 218-238.

[136] Yang J, Wang H, Wang J, et al. Provable data possession of resource-constrained mobile devices in cloud computing[J]. Journal of networks, 2011, 6(7): 1033-1040.

致　谢

值此论文完成之际，谨向所有给予我指导、关心和帮助的各位老师、同学和亲人表示衷心的感谢！

本论文的研究思路来源于导师王国庆教授，衷心感谢导师王国庆教授！王老师以宽广的学术视野带领我进入了分布式航空电子系统网络技术这一充满生机的研究领域，站在全局的高度给我指明了研究方向，并以其深厚而渊博的理论知识始终指导着我的科研和学习，其丰富的实践经验和科学研究方法使我受益匪浅。王老师严谨的治学态度、踏实谨慎的工作作风为我树立了学习的榜样，特别是他对航空事业的献身精神，更令我终生难忘。

本论文是在翟正军教授的悉心指导下完成的。首先，向导师翟正军教授表示衷心的感谢。在课题研究和论文撰写中，翟老师给予了无私的指导和帮助，从论文的选题到各个研究阶段始终给予了关怀、鼓励和悉心指导，为本文的完成倾注了大量的心血，对本论文进行了多次修改和认真评阅，提出了很多宝贵的意见。同时，对我的严格要求使得我在工程和学术上进步很大。特别是他严谨的治学态度、精湛的专业知识、在科学研究上的进取和献身精神，永远是我学习的楷模。

六年多的博士学习中，两位老师从思想、学习到生活上都给予了我无微不至的关怀，我深感师恩难忘，在此谨向王老师和翟老师致以崇高的敬意！

在多年的求学历程中，还得到了郭阳明老师、陆艳洪老师、羊天德老师和张隽老师的无私帮助和支持，在此向他们表示最真挚的感谢！

还要感谢的是教研室中的赵宁社博士、王恒博士、牛伟博士、张丽花博士以及师弟宋宵罡等，与他们进行的学术讨论和交流对我的学业和论文的进展有着很大的帮助。

永远感激我的父母和姐姐，深深地感谢他们对我学习上的全力支持以及生活上细致入微的关怀和照顾。

感激我的岳父、岳母，他们用真挚的爱默默地鼓励我、支持我。

感谢我的妻子叶晓雪女士，在我求学的道路上，与我风雨同舟、不离不弃，即使相隔万里依旧时刻关心、包容和支持我，感谢她对我生活上的照顾和对家庭的付出，感谢她为我所做的一切。

最后，感谢本文的评阅老师和专家对论文的评审，你们的宝贵意见使本论文更加完善。同时，也感谢答辩委员会的各位老师出席我的论文答辩。

攻读博士学位期间发表的学术论文和参加科研情况

学术论文发表情况：

[1] **Wenzhe Jiao**, Guoqing Wang, Zhengjun Zhai, et al. Dynamic Data Possession Checking for Secure Cloud Storage Service[J]. Journal of Networks, 2013, 8(12): 2713-2720. (EI Compendex: 20135117100728) .

[2] **Wenzhe Jiao**, Guoqing Wang, Xiaoxue Ye, et al. Efficient Dynamic Data Possession Checking in Cloud Computing[C]. 2013 International Conference on Applied Science, Engineering and Technology, September 2013, Qingdao, China. Advanced Materials Research, Vol. 709, 603-610, ISSN: 1662-8985. (EI Compendex: 20132916521229).

[3] **Wenzhe Jiao**, Xiaoxue Ye. A Cloud Architecture for Service Security Management[C]. 2013 International Symposium on Linear Drives for Industry Applications, July 2013, Hangzhou, China. Applied Mechanics and Materials, Vol. 416-417, 1413-1417, ISSN: 1662-7482. (EI Compendex: 20134416920897).

[4] **焦文喆**, 翟正军, 王国庆. 时间触发 AFDX 调度设计与实时性分析[J]. 计算机工程, 2016,42(7): 42-48.

[5] **焦文喆**, 翟正军, 王国庆. 时间触发 AFDX 调度策略设计与实时性分析[J]. 山东农业大学学报, 2016,47(1): 111-117.

[6] **焦文喆**, 王国庆, 翟正军等. 云存储下基于代数签名的数据持有性检查方法[J].东北师大学报（自然科学版）, 2013,45(4): 55-61.

[7] **Wenzhe Jiao**, Guoqing Wang, Zhengjun Zhai. Fair and Efficient Packet Scheduling Using SMQRR[J]. Journal of Communications, 改后录用.

参加科研情况：

[1] 项目名称：“十二五”国防基础预研项目“机载高速 FC-AE 总线 xxx 技术研究”
参加时间：2011 年 5 月-2013 年 8 月
主要工作：研究 FC-AE 总线协议，搭建 FC-AE 接口验证平台，设计通信指令，设计 FC-AE 故障注入与故障诊断系统的总体框架，完成了任务处理模块与用户操作模块的代码编写。项目已验收并通过国防成果鉴定。

[2] 项目名称：“十一五”国防基础预研项目“网络化远程测试与故障 xxx 技术研究”
参加时间：2009 年 4 月-2010 年 12 月
主要工作：研究故障诊断规则，设计故障诊断数据库并实现相应的 SQL 数据

库，设计开发了远程故障诊断系统的系统自检模块，解决了分布式系统的时钟同步问题和网络数据安全传输问题，搭建软件框架，编写故障诊断模块代码。项目获得国防科技进步二等奖。

[3] 项目名称：“十二五”民机预研专项“民用飞机实时故障诊断和维护技术”

参加时间：2011 年 4 月 2014 年 12 月

主要工作：完成故障模拟注入软件、远程数据转换软件及远程故障诊断软件开发。故障模拟注入软件模拟产生民用飞机故障信号，远程数据转换软件通过 ARINC429 总线和 RS422 总线接收故障模拟注入软件模拟的故障信号或者民机实际产生的故障信号，并进行实时图形化显示，然后通过 UDP 网络包将故障信号发送至远程故障诊断软件，由远程故障诊断软件对故障数据进行分析诊断。对故障模拟注入软件与远程数据转换软件进行设计并实现，利用多线程技术解决了故障信号实时显示与实时发送传输的难点。项目已通过专家验收。

[4] 项目名称：某国家重点型号无人机配套子课题“机载音视频记录器综合检测仪”

参加时间：2012 年 11 月-2014 年 10 月

主要工作：技术负责人，负责系统开发的硬件选型、软件架构设计以及软件测试，组织相关人员完成整个项目的实施、现场调试和交付，突破了巨量数据的并行快速下载、多盘数据快速索引、图像拼接、多类型视频图像识别等关键技术，已经在某型号无人机的机载设备研发和交付中发挥重要作用。

[5] 项目名称：某国家重点型号无人机配套子课题“高空红外扫描仪（扫描相机）综合仿真与检测系统”

参加时间：2009 年 9 月-2010 年 7 月

主要工作：技术负责人，负责系统软件方案设计、组织实施、硬件驱动、软件开发及最终交付，设计开发了高速图像数据采集卡和多极旋转变压器角度数据采集卡的逻辑程序及驱动程序，突破了基于 FPGA 的高速扫描图像并行采集、大幅面图像的特征识别与拼接、图像校准等关键技术，已应用于某型高空无人机的设备研制和交付中。

[6] 项目名称：某运输机“显示与控制系统仿真测试设备”

参加时间：2013 年 8 月-2015 年 8 月

主要工作：研究 ARINC429 总线和 AFDX 总线协议，对网络数据传输过程中的数据传输时延、可靠性等特性进行测试。实物样机内部通过 AFDX、ARINC429 总线连接，测试设备通过 ARINC429 总线电缆和交叉网线和实物样机连接，测试数据通过人机界面提供给测试人员进行归类分析，为进一步改进系统提供准确的验证和测试数据。

西北工业大学

学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西北工业大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校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。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西北工业大学。

保密论文待解密后适用本声明。

学位论文作者签名：隆文喆　　指导教师签名：王国庆

2017 年 6月 1 日　　2017 年 6 月 1 日

西北工业大学

学位论文原创性声明

秉承学校严谨的学风和优良的科学道德，本人郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容和致谢的地方外，本论文不包含任何其他个人或集体已经公开发表或撰写过的研究成果，不包含本人或其他已申请学位或其他用途使用过的成果。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表明。

本人学位论文与资料若有不实，愿意承担一切相关的法律责任。

学位论文作者签名：隆文喆

2017 年 6月 1 日