的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：　　日期：　　年　月　日

学位论文使用授权说明

本人完全了解杭州电子科技大学关于保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属杭州电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为杭州电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密论文在解密后遵守此规定）

论文作者签名：　　日期：　　年　月　日

指导教师签名：　　日期：　　年　月　日

杭州电子科技大学硕士学位论文

基于 OSPFv3 协议的 GR 机制实现

研 究 生： 平 诞

指导教师： 胡 建 萍 教 授

2012 年 12 月

Dissertation Submitted to Hangzhou Dianzi University

for the Degree of Master

Implementation of GR Mechanism

Based on OSPFv3 Protocol

Candidate: Ping Dan

Supervisor: Prof. Hu Jianping

December，2012

摘　要

随着网络技术的迅速发展，IP 网络上承载着越来越多的互联网业务，同时，IP 网络的可靠性要求也不断提高。对于运营商网络和大型企业网络，短暂的网络中断将带来巨大的损失。因此，可靠性和持续性逐渐成为衡量网络综合性能的关键技术指标之一。OSPFv3 协议是目前世界上使用最多的内部路由协议，主要提供对 IPv6 网络的支持。在全球加紧推进 IPv4 向 IPv6 过渡的背景下，OSPFv3 协议必将得到更广泛的应用。但由于 OSPFv3 协议自身运行机制的限制，可能导致协议重启时网络中出现路由振荡和数据转发流量中断。GR 机制是一种高可靠性技术，能够在路由器设备重启过程中保证数据转发流量不中断，网络中也不会产生路由振荡，从而有效地提高网络的可靠性。当前，各种主流路由协议提供对 GR 机制的支持和融合已成为一种必然趋势。本文即是对基于 OSPFv3 协议的 GR 机制进行研究和实现。

本文首先介绍了 IPv6 网络及路由器设备的发展概况，归纳和总结了 IPv6 网络路由协议及其可靠性技术的国内外研究现状和发展趋势。

其次，对 OSPFv3 协议和 GR 机制进行了深入研究，为全文奠定了理论基础。从协议的基本概念出发，文中详细阐述了 OSPFv3 协议的工作原理和运行机制，包括接口状态机、邻居状态机的运转和 LSA 报文的处理。然后，通过对运行 OSPFv3 协议的路由器设备重启过程的具体分析，提出其问题所在，从而引入 GR 机制，详细分析了 GR 机制的基本原理和工作机制，指出其优越性所在。

在分析协议的基础上，文中给出了软件系统的总体设计方案，包括与外部模块的交互关系和内部线程的功能划分，并设计了系统的公用数据结构。接着，详细论述了 GRRestarter 模块和 GRHelper 模块的设计实现，主要包括 GR 状态机的设计，GR 重启过程中 GRRestarter 和 GRHelper 对协议报文、定时器、邻居数据、LSDB 数据以及路由信息的维护和处理。

最后，文中设计了相应的测试组网，对软件系统的设计结果进行了测试和验证，具体包括对 GRRestarter 模块和 GRHelper 模块的基本功能测试，以及对整体系统的性能测试和压力测试。测试结果表明，该软件系统的实现方案是正确可行的，满足设计要求和交付标准，可运行于实际网络中。同时，本文最后也指出了该系统的设计实现中一些可以进一步改进和完善的地方，并对下一阶段的发展进行了展望。

本文设计实现的基于 OSPFv3 协议的 GR 机制能够保证路由器设备重启过程中数据转发流量不中断，从而有效地提高了网络的可靠性、持续性和稳定性，在实际网络中具有广泛应用。

关键词：OSPFv3，GR 机制，IPv6 网络，协议重启，可靠性

ABSTRACT

With the rapid development of network technology, the IP network carrying more and more Internet business, and at the same time, the requirement of IP network reliability is also constantly improving. For carrier network and large enterprise network, a brief network interruption will bring huge losses. Therefore, reliability and sustainability gradually become one of the key technical indicators to measure the overall performance of the network. The OSPFv3 protocol is the most used internal routing protocol in the world currently, and mainly to provide support for IPv6 network. Under the background of intensively promote the transition from IPv4 to IPv6 globally, OSPFv3 protocol is bound to be more widely applied. However, due to the limitation of operation mechanism of OSPFv3 protocol itself, it may lead to routing oscillation and data forwarding flow interruption in the network during the protocol restarting. GR mechanism is a kind of high reliability technology, can ensure that the data forwarding flow without interruption during the router restarting, and also will not produce the routing oscillation in the network, so as to effectively improve the reliability of the network. Currently, it has become an inevitable trend that all the mainstream routing protocols provide support and integration for GR mechanism. This article just research and implement the GR mechanism based on OSPFv3 protocol.

The paper firstly introduces the overview of the development of IPv6 network and router equipment, and summarizes the research status and development trend of IPv6 network routing protocol and its reliability technology in the world.

Secondly, deeply research the OSPFv3 protocol and GR mechanism to lay the theoretical foundation for the full text. Starting from the basic concept of the protocol, the text elaborates the working principle and operation mechanism of OSPFv3 protocol, including the operation of interface state machine and neighbor state machine, and process of LSA packets. Then, by analyzing the process of router restarting which running OSPFv3 protocol, it proposes the problem, and introduces the GR mechanism, detailed analysis the basic principle and working mechanism of GR and point out its advantage.

Based on the analysis of the protocol, it presents the overall design scheme of the software system in the text, including the interactive of external modules and the functional partition of internal threads, and designs the system common data structure. Then, it expounds the design and implementation of GRRestarter module and GRHelper module in detail, mainly including the design of GR state machine, the maintainance and processing of protocol packets, timer, neighbor data, LSDB data and routing information for GRRestarter and GRHelper during the graceful

restarting.

Finally, the paper designs the corresponding network to test and verify the design result of the software system, including the basic function test of GRRestarter module and GRHelper module, the performance test and stress test of whole system. The test result shows that the implementation of the software system is correct and feasible, meeting the design requirement and delivery standard, and can be running in the actual network. Meanwhile, it also points out some place that can be further improved and perfected in the design and implementation of the system at the end of the text, and make prospect for the development in the next stage.

The GR mechanism based on OSPFv3 protocol which is designed and realized in the article can ensure the data forwarding flow without interruption during the router restarting, effectively improve the reliability, sustainability and stability of the network, and have widely application in the actual network.

Keywords： OSPFv3, GR mechanism, IPv6 network, protocol restart, reliability

目　　录

第1章 绪 论

1.1 研究背景及意义

近年来，随着网络通信技术的不断发展，网络规模越来越大，数据传输速率越来越快，复杂度也越来越高，IP 网络已经从各种网络设备的简单互连向以各种复杂模式互连的网络群发展[1]。与此同时，IP 网络已在交通、能源、企业、学校、通信、军事等领域得到了广泛应用[2]，IP 网络上承载着大量的互联网业务，这也要求 IP 网络具有更高的可靠性和稳定性。对于运营商和大型企业来说，由于业务中断导致的损失越来越难以接受。如今，IP 网络运行的可靠性和稳定性问题也受到了网络开发商和运营商的普遍重视，将其作为衡量 IP 网络综合性能的关键技术指标之一[3]。因此，研究 IP 网络的可靠性技术具有广泛的科学价值和实际意义。

目前，国家正在大力推进三网融合，网络视频、语音传输等各种新型业务不断涌现，再加上云计算、物联网等相关技术发展的需要，对 IP 地址的需求量急剧增加[4]。然而，以前的 IPv4 协议由于地址数量匮乏、安全性不高、路由选择效率低下、服务质量较差等不足之处，在实际应用中已经不能满足 IP 网络迅速发展的需要。从而，下一代网络标准——IPv6 应运而生，IPv6 协议因其自身的诸多优点必将成为新一代互联网的 IP 层技术[5]，由 IPv4 向 IPv6 平稳过渡也越来越成为国内外各界关注的焦点。目前，各国正在积极加紧对 IPv6 网络的部署和应用[6]。另一方面，路由协议是保障和提高 IP 网络性能的重要因素之一，IPv6 网络的建设同样需要高效稳定的路由协议的支持。OSPF（Open Shortest Path First，开放最短路径优先）是目前网络中使用最多、应用最广泛的内部路由协议。OSPFv3 是 OSPF 版本 3 的简称，主要提供对 IPv6 网络的支持，是 IPv6 网络中的主流路由协议[7]。因此，在 IPv6 作为下一代网络 IP 层技术已成定局的情况下，深入研究 OSPFv3 协议及其可靠性技术具有重要的学术价值和经济意义[8]。

路由器设备是 IP 网络中的重要节点，也是 Internet 网络互连的核心设备，主要实现数据分组的选路和转发功能。路由器设备的性能直接影响网络互连的质量，其处理速度是网络通信中的主要瓶颈之一[9]。随着用户业务的数据化、IP 化，路由器设备在 IP 骨干网络中的作用和地位日益突出。与此同时，各种互联网业务的快速发展也对路由器设备的性能和功能提出了更高的要求[10]。因此，当今的主流路由器设备都由集中式向分布式转变。在分布式路由器设备中，其控制层面和数据转发层面相分离，控制层面主要负责系统任务的控制以及路由信息的管理和下发，而数据转发层面主要负责数据报文的接收和转发[11]。在这种分布式处理情况下，当路由器的控制层面软件重启/重载、主备倒换或异常重启的时候，数据转发层面必须仍然能够正确持续地转发数据报文，保证数据流量的不中断。对于一个大型网络，尤其是运营商网络和企业网络来说，避免因路由器设备重启而引起的网络路由振荡和数据转发流量中

断具有极其重要的社会意义和经济意义。

1.2 国内外研究现状和发展趋势

近年来，为了解决 IPv4 地址濒临枯竭的问题，IPv6 的发展已经受到了国内外各界的高度关注。目前，美国、日本、中国和欧盟都已经建立起了 IPv6 骨干网络，并逐渐开始推进 IPv4 向 IPv6 的过渡[12]。据了解，日本已经向企业和家庭用户提供了 IPv6 服务；美国也已经开始将电信网络迁移到 IPv6 平台，并提供了 WWW、Email 等 IPv6 服务，2014 年底前再提供基础网络与系统的 IPv6 服务[13]；欧洲的多个国家则计划先以政府进行网络转换为主，再陆续提供 IPv6 的相关服务。中国政府和运营商也都发布了 IPv4 向 IPv6 过渡的时间表，规划了 IPv6 发展的明确路线图[14]。可以说，互联网由 IPv4 向 IPv6 过渡已经开始全面进入了实施阶段。

路由协议作为 IPv6 网络的核心技术，涉及网络协议、路由协议栈软件和路由器设备等多方面的技术。路由协议实现的好坏，直接决定了 IPv6 网络运行的可靠性和稳定性，也影响着网络中上层应用的质量。目前在 IPv6 网络中广泛使用的主流路由协议主要包括 RIPng（Routing Information Protocol next generation，下一代路由信息协议）、OSPFv3（Open Shortest Path First version 3，开放最短路径优先版本 3）、IS-ISv6（Intermediate System to Intermediate System，中间系统到中间系统协议）和 BGP4+（Border Gateway Protocol 4+，边界网关协议）[15]。其中 BGP4+是自治系统间运行的路由协议，是一种外部网关协议，而 RIPng、OSPFv3 和 IS-ISv6 是自治系统内部运行的路由协议，为内部网关协议。

RIPng 协议基于距离向量算法，应用较早，是一种较为成熟的内部路由协议。RIPng 协议的最大优点是实现简单，在规模较小、拓扑结构简单的小型网络中易于配置和维护[16]，但对于规模较大的网络，RIPng 协议存在路由环路和占用过多带宽等问题，选路性能不如基于链路状态的路由协议，这在一定程度上限制了网络规模的扩张。因此，对于大型网络，IS-ISv6 协议和 OSPFv3 协议得到了广泛应用。IS-ISv6 协议和 OSPFv3 协议都是基于链路状态的路由协议，具有良好的扩张性和更多的控制功能，适合于运营商建设大规模网络[17]。两者在质量和性能上的差别并不大，但 OSPFv3 协议更适用于 IP 操作，较 IS-ISv6 协议更具有活力，应用也更为广泛。IETF（Internet Engineering Task Force，因特网工程任务组）也始终在致力于 OSPFv3 协议的修改和改进工作，其修改节奏要比 IS-ISv6 协议快得多[18]。这使得 OSPFv3 协议在 IPv6 网络中得到了迅速发展，已成为目前世界上 IPv6 网络中应用最为广泛、性能最优的内部路由协议。

为了满足 IP 网络迅速发展而不断提出的需求，当前的主流路由器设备都由集中式设备向分布式设备发展，路由器设备的形态越来越复杂，同时也越来越灵活。当网络中的一台路由器设备发生故障重启时，往往只影响该设备的一部分（如路由器的控制层面），而设备的其它部分（如路由器的转发层面）还可以正常工作。基于此，可以采用快速恢复网络中故障设备的故障模块的方法来实现整个设备的恢复，而其它部分无需进行重启恢复，仍然可以保持正常工作[19]。这样整个恢复过程对运行网络透明，可以保证数据转发业务的稳定，从而提高网

络的可靠性和稳定性。GR（Graceful Restart，平滑重启）就是这样一种高可靠性技术，能实现路由器设备重启过程中网络路由保持稳定，数据转发流量不中断的功能。目前，GR 机制的实现主要有 IETF 标准 GR 和非 IETF 标准 GR 两种方式，前者基于 IETF 标准，而后者不基于 IETF 标准。随着网络可靠性要求的不断提高，IPv6 网络中的各种主流路由协议提供对 GR 机制的支持和融合已成为一种必然趋势。

1.3 本文研究内容和组织结构

本文主要对基于 OSPFv3 协议的 GR 机制进行研究和实现。文章在对 OSPFv3 协议基本概念和运行机制进行深入研究的基础上，重点阐述了 IETF 标准 GR 机制的工作原理和实现细节，详细分析了 OSPFv3 GR 机制的运行流程和报文交互过程，同时对整体软件系统进行了设计实现，并且通过设计测试组网对设计结果进行了功能测试和性能测试，验证了设计方案的正确性和可行性。

文中各章节的内容安排如下：

第一章绪论，主要介绍了本课题的研究背景和意义，以及 IPv6 网络路由协议及其可靠性技术的国内外研究现状和发展趋势，并给出了本文的主要研究内容和组织结构。

第二章从 OSPFv3 协议基本概念的介绍入手，详细阐述了 OSPFv3 协议的工作原理和运行机制，具体分析了邻接关系的建立和 LSA 的处理。

第三章通过对运行 OSPFv3 协议的路由器设备协议重启过程的具体分析，指出其问题所在，从而引入 GR 机制，介绍了 GR 机制的基本概念和 Grace LSA 报文格式，并详细阐述了 GR 机制的工作原理和具体实现。

第四章给出了软件系统的总体设计方案，对内部线程进行了功能划分，并阐述了与外部模块的交互关系，同时还定义了公用的数据结构，为下文 GRRestarter 模块和 GRHelper 模块的具体实现奠定了基础。

第五章详细论述了 GRRestarter 模块的设计实现，主要包括 GR 状态机的设计，GR 重启过程中 GRRestarter 对协议报文、定时器、邻居数据、LSDB 数据、路由信息和异常事件的维护和处理。

第六章详细论述了 GRHelper 模块的设计实现，主要包括进入 GRHelper 模式的判定，GR 重启过程中 GRHelper 对协议报文、定时器、邻居数据、LSDB 数据、路由信息和异常事件的维护和处理。

第七章设计了相应的测试组网，对设计结果进行测试，包括对 GRRestarter 模块和 GRHelper 模块的基本功能测试、对整体系统的性能测试和压力测试，并对测试结果进行了分析，验证设计方案的正确性和可行性。

第八章对全文进行总结，提出系统设计中可以改进和完善的地方，并对下一阶段的发展进行展望。

第2章 OSPFv3 协议

2.1 OSPFv3 协议简介

OSPF 是 IETF 组织开发的一种基于链路状态的自治系统内部路由协议，OSPFv3 是 OSPF 版本 3 的简称，主要提供对 IPv6 网络的支持[20]，遵循的标准为 RFC5340（OSPF for IPv6）。

OSPF 具有适应范围广、收敛迅速、无自环、便于层级化网络设计等特点[21]，是目前广域网和大型企业网中应用最广泛的内部路由协议。随着 IPv6 网络建设的快速推进，同样需要准确高效的动态内部路由协议为 IPv6 报文的转发提供路由信息。因此，IETF 在保留 OSPF 协议优点的基础上针对 IPv6 网络修改形成了 OSPFv3 协议，主要用于在 IPv6 网络中提供路由功能，是 IPv6 网络中的主流路由协议。

OSPFv3 与 OSPF 的不同之处主要有：OSPFv3 是基于链路运行，而不是基于网段运行；OSPFv3 在同一条链路上支持多实例；OSPFv3 通过路由器 ID 来标识邻居，而不是 IP 地址[22]。

OSPFv3 协议的基本工作机制是各路由器设备之间通过周期性地交互 Hello 报文来建立并维持邻居关系，在建立邻接关系的设备之间互相扩散用于描述链路状态的 LSA（Link State Advertisement，链路状态通告），并且最终形成相同的 LSDB（Link State DataBase，链路状态数据库），在 LSDB 的基础上进行路由计算生成路由信息并下发到路由表中[23]。

2.1.1 区域

随着网络规模的不断扩大，当一个大型网络中的路由器都运行 OSPFv3 路由协议时，整个 LSDB 就会非常庞大，占用大量的存储空间，并且计算最短路径树的复杂度也随之增加，使得 CPU 的负担加重 [24]。同时，网络规模的增大也会导致拓扑结构发生变化的概率随之增大，网络中会存在大量的 OSPFv3 协议报文，降低了网络的带宽利用率。更为严重的是，每一次的拓扑变化都会引起网络中的所有路由器重新进行路由计算[25]，导致网络路由振荡。OSPFv3 协议通过将自治系统划分为不同的区域来解决上述问题，从逻辑上将网络中的路由器划分为不同的组，即区域，每个区域用区域号来标识。一台路由器可以接入多个区域，并为每个接入的区域建立单独的 LSDB，独立运行一套链路状态路由算法，这样就极大地减少了网络中的协议报文数量和路由流量。

区域 0 为 OSPFv3 的骨干区域，负责分发其它非骨干区域之间的路由信息。骨干区域必须是连续的，但是并不需要物理上的连续，骨干区域的连续性可以通过虚链路来建立并保持。所有的非骨干区域必须与骨干区域保持连通。

2.1.2 协议报文类型

OSPFv3 协议中定义了五种类型的报文[26]：

（1）Hello 报文

Hello 报文用于发现和维持 OSPFv3 邻居关系，通过定时器周期性地进行发送。Hello 报文中包含了 Hello 定时器间隔（HelloInterval）、Dead 定时器间隔（DeadInterval）、DR（Designated Router，指定路由器）、BDR（Backup Designated Router，备份指定路由器）和自己已知的邻居路由器 ID。在广播网和 NBMA(Non-Broadcast Multi-Access，非广播多点接入)网络中，Hello 报文也被用于 DR 和 BDR 的选举。

（2）DD（Database Description，数据库描述）报文

DD 报文用于向邻居路由器描述本地 LSDB 汇总情况，报文内容不需要包含完整的 LSA 信息，只需包含每条 LSA 的头部摘要信息即可。

（3）LSR（Link State Request，链路状态请求）报文

LSR 报文用于向邻居路由器请求所需要的 LSA，报文内容是所请求 LSA 的头部摘要信息。

（4）LSU（Link State Update，链路状态更新）报文

LSU 报文用于向邻居路由器发送相应的 LSA，报文内容是完整的 LSA 信息，一个 LSU 报文可同时包含多条 LSA 信息。

（5）LSAck（Link State Acknowledgement，链路状态确认）报文

LSAck 报文用于对所收到的 LSA 向邻居路由器进行确认，报文内容是所收到 LSA 的头部信息。一个 LSAck 报文可包含多个 LSA 的头部信息，同时对多个 LSA 进行确认。

上述五种类型的报文是 OSPFv3 协议运行的基础。除了 Hello 报文以外，其它四类报文都只在形成邻接关系的路由器之间进行传送，也即只会在 IP 网络中传播一跳。

2.1.3 LSA 类型

在 OSPFv3 协议中，所有的链路状态信息都体现在 LSA 中，LSA 是 OSPFv3 协议计算和维护路由信息的主要来源和基础。各类 LSA 的汇总形成了 LSDB，其中包含了自治系统内的各路由器及网络拓扑信息，是路由器建立路由表的依据[27]。OSPFv3 协议中定义了七种类型的 LSA[28]:

（1）Router-LSA

Router-LSA 由每台路由器自己产生，其中描述了路由器的所有接口、链路状态及其开销。该类 LSA 只在生成路由器所在的区域内进行传播。

（2）Network-LSA

Network-LSA 由广播网络和 NBMA 网络中的 DR 产生，其中描述了链路上所有路由器的链路状态。该类 LSA 只在 DR 所在的区域内进行传播。

（3）Inter-Area-Prefix-LSA

Inter-Area-Prefix-LSA 由 ABR[29]（Area Border Router，区域边界路由器）生成。ABR 同时属于两个或两个以上的区域，且其中一个为骨干区域。ABR 通过 Inter-Area-Prefix-LSA 将一个区域的路由信息发布到所接入的其它区域中，其中携带了相应区域的 IPv6 地址前缀路由信

息。该类 LSA 只在与该 LSA 相关的区域内传播。

（4）Inter-Area-Router-LSA

Inter-Area-Router-LSA 由 ABR 产生，其中描述了一条到达本自治系统内 ASBR 的路由信息。该类 LSA 只在 ABR 所在区域内传播。

（5）AS-external-LSA

AS-external-LSA 由自治系统内的 ASBR（Autonomous System Border Router，自治系统边界路由器）产生，其中描述了到达其他自治系统的外部路由信息。ASBR 与其它自治系统相连并交换路由信息，将引入的自治系统外部路由信息通过 AS-external-LSA 在本自治系统内部进行重新发布，该类 LSA 在除 Stub[30]区域和 NSSA 区域[31]以外的整个自治系统内传播。

（6）Link-LSA

Link-LSA 由路由器为所接入的每条链路产生，其中描述了接入接口的 IPv6 地址前缀以及 Link-Local 地址。该类 LSA 只在本地链路范围内传播。

（7）Intra-Area-Prefix-LSA

Intra-Area-Prefix-LSA 由每台路由器自己产生，其中描述了路由器上的接口 IPv6 地址前缀信息。该类 LSA 只在生成路由器所在的区域内传播。

2.1.4 邻居与邻接

运行 OSPFv3 协议的路由器启动后，便会向外发送 Hello 报文来通告自身和发现邻居，同时也会收到网络中其它路由器发送的 Hello 报文。收到 Hello 报文后，会对报文中所携带的参数进行检查，如果双方一致就会形成邻居关系。

形成邻居关系的双方并不一定都会进一步形成邻接关系，邻接关系的形成与否取决于所接入的网络类型。在 PTP（Point-to-Point，点到点）和 PTMP（Point-to-MultiPoint，点到多点）网络中邻居双方始终形成邻接关系，而在广播网和 NBMA 网络中，路由器只与 DR 和 BDR 建立邻接关系。只有当邻居双方成功交换 DD 报文和 LSA 并达到 LSDB 的同步之后，才真正形成邻接关系，在形成邻接关系的路由器之间必须始终保持 LSDB 的同步。

2.1.5 DR 与 BDR

在广播网和 NBMA 网络中，为了减少网络中的报文数量和路由流量，OSPFv3 协议中定义了 DR 和 BDR。网络中的路由器都将路由信息发送给 DR，由 DR 再将其以组播方式发送给链路上的其它路由器。BDR 是对 DR 的备份，并在 DR 失效时立即成为 DR，既不是 DR 又不是 BDR 的路由器为 DRother。DRother 只与 DR 和 BDR 建立邻接关系并交换路由信息，DRother 之间不会建立邻接关系，也不交换任何路由信息，从而减少了网络中邻接关系的数量，同时也减少了网络中的报文数量和路由流量，节约了带宽资源。

广播网和 NBMA 网络中的 DR 和 BDR 是由同一链路上的所有路由器根据路由优先级和路由器 ID 通过 Hello 报文选举出来的。每个路由器接口都可以配置相应的路由优先级，并被包含在 Hello 报文中，只有接口路由优先级大于 0 的路由器才有资格参与 DR/BDR 选举。每

台路由器都将自己选出的 DR 和 BDR 写入 Hello 报文中发送到相应链路上。当同一链路上的两台路由器同时宣布自己是 DR 或 BDR 时，路由优先级高者胜出；如果路由优先级相同，则路由器 ID 大者胜出[32]。

2.1.6 路由计算

OSPFv3 协议中的路由计算是基于 LSA 进行的。协议触发路由计算时，从各类 LSA 中提取出相应的拓扑、前缀和下一跳信息，将 LSDB 转换成一张带权有向图，使用 SPF（Shortest Path First，最短路径优先）算法[33]计算出一棵以自己为根的最短路径树，在此基础上计算出相应的路由信息，并下发路由表来刷新相应的路由表项。OSPFv3 网络中的路由计算大致可以分为如下几类：区域内拓扑计算、区域内路由计算、ASBR 路由计算、区域间路由计算和外部路由计算。当经过路由计算存在多条等价路径时，数据流量将在这些路径上进行分摊。

2.2 形成邻接

OSPFv3 协议中邻接关系的建立通过接口状态机和邻居状态机的运转来实现，接口状态机和邻居状态机也是整个协议运行的基础和核心所在[34]。

2.2.1 接口状态机

接口用以连接路由器和网络，OSPFv3 协议中的接口状态机如图 2.1 所示，由相应的接口事件触发接口状态机的运转，进而推动接口状态的变迁。

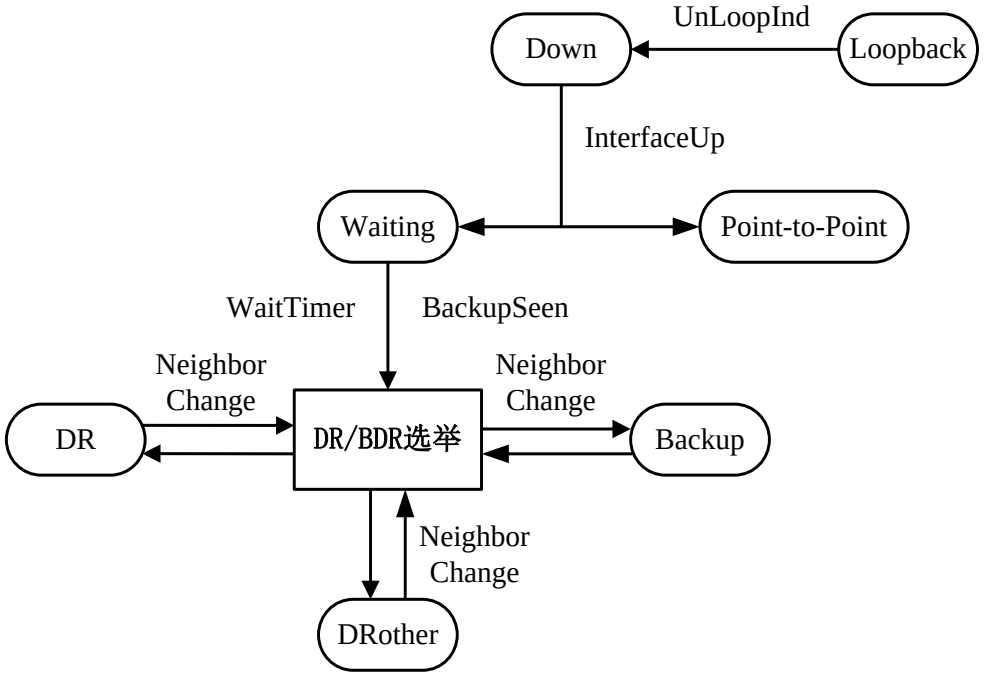

图 2.1　OSPFv3 协议接口状态机

如图中所示，OSPFv3 协议中定义了 7 种接口状态：

Down：接口的初始状态。此时，接口上没有 OSPFv3 协议报文的收发，接口的所有参数都为初始值。

Loopback：表明接口处于环回状态，可能以硬件或软件方式实现。处于 Loopback 状态的

接口不能用于正常的数据传输，但是接口地址仍可以通过 Router-LSA 通告出去。

Point-to-Point：当连接到 PTP 网络、PTMP 网络或虚链路的接口收到 InterfaceUp 接口事件时，进入 Point-to-Point 接口状态。

Waiting：当连接到广播网或 NBMA 网络的接口接收到 InterfaceUp 接口事件时，并且具有 DR/BDR 选举资格（即接口优先级不为 0），则进入 Waiting 接口状态。接口进入 Waiting 状态后，路由器会启动 Waiting 定时器，并且尝试通过监听接收到的 Hello 报文来判定链路上的 DR 和 BDR。在 Waiting 定时器超时前，该路由器不能被选举为 DR 或 BDR，以此来避免链路上 DR 和 BDR 的频繁改变。

DR：该接口状态表明路由器是所在链路上的 DR。此时，该路由器与链路上的其它所有路由器建立邻接关系。

Backup：该接口状态表明路由器是所在链路上的 BDR。此时，该路由器与链路上的其它所有路由器建立邻接关系，并且在当前的 DR 失效时成为 DR。

DRother：该接口状态表明路由器在所在链路上既不是 DR 也不是 BDR。此时，该路由器与 DR 和 BDR 建立邻接关系，而 DRother 之间并不建立邻接关系。

接口状态的变迁由接口事件触发，相应的，OSPFv3 协议中定义了 7 种接口事件：

InterfaceUp：由下层协议触发，表明当前接口使能到相应的 OSPFv3 进程中，并且是可操作的。触发该接口事件后，启动接口下的 Hello 定时器向外发送 Hello 报文来发现网络中的邻居。

LoopInd：由网管或下层协议触发，表明接口将进入 Loopback 状态，不能用于正常的数据传输。任意接口状态下收到 LoopInd 接口事件都将进入 Loopback 接口状态。

UnloopInd：由网管或下层协议触发，表明接口退出 Loopback 状态。

WaitTimer：Waiting 定时器超时，表明对链路上 DR 和 BDR 信息的监听过程结束，可以开始进行链路上 DR 和 BDR 的选举。

BackupSeen：该事件表明路由器已经探测到所在链路上是否存在 BDR。有两种方式可以得知：（1）收到邻居的 Hello 报文中将其自身标识为 BDR；（2）收到邻居的 Hello 报文中将其自身标识为 DR，并且没有 BDR。

Neighbor Change：该事件表明接口上的邻居状态发生了变化，需要重新进行链路上 DR 和 BDR 的选举。

InterfaceDown：由下层协议触发，表明接口断开，不可操作。任意接口状态下收到 InterfaceDown 事件都将进入 Down 状态。

2.2.2 邻居状态机

每台运行 OSPFv3 协议的路由器都会与网络中的其它路由器建立邻居关系，OSPFv3 协议中的邻居状态机如图 2.2 所示，由相应的邻居事件触发邻居状态机的运转，进而推动邻居状态的变迁[35]。

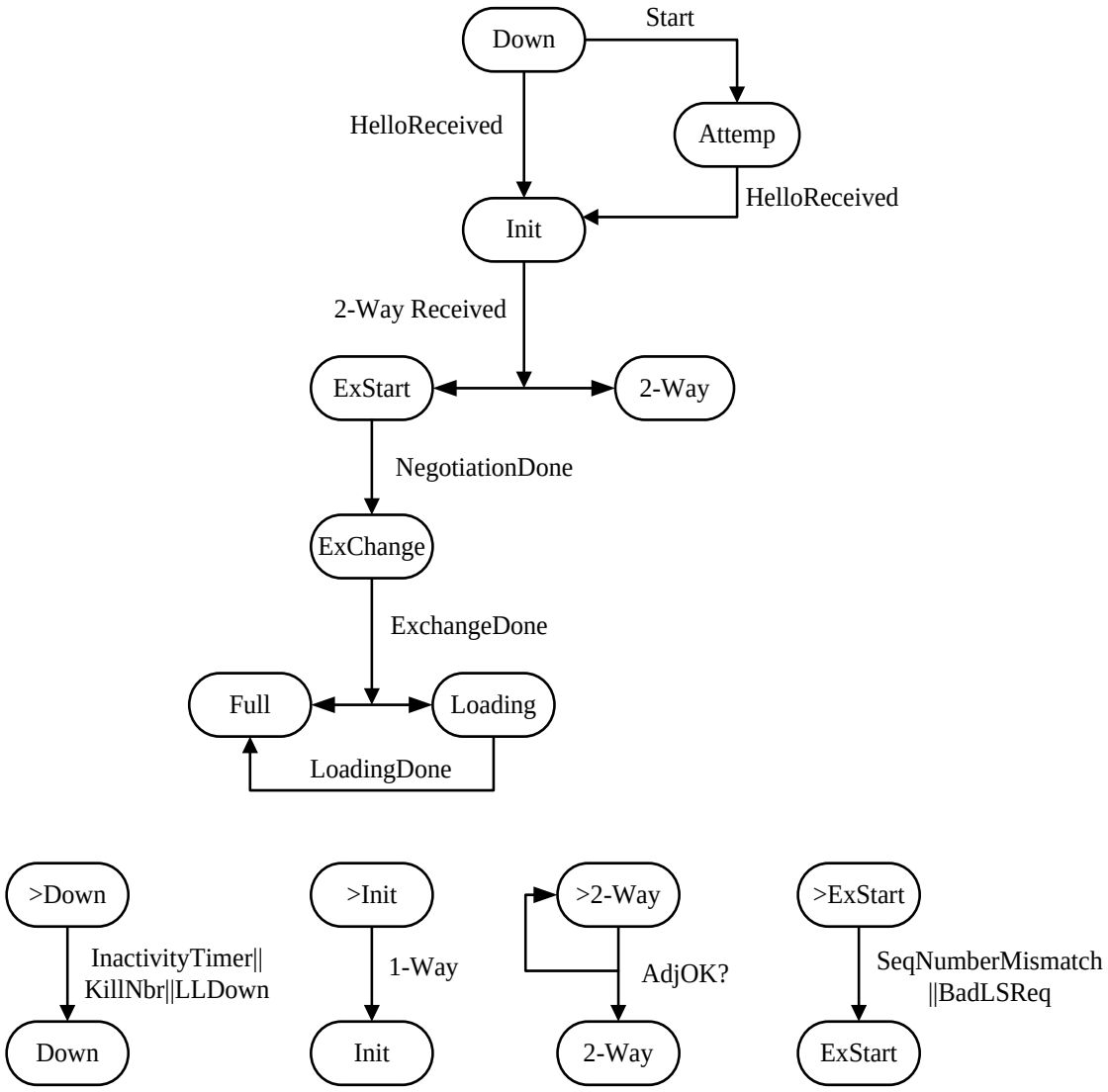

图 2.2　OSPFv3 协议邻居状态机

OSPFv3 协议中定义了 8 个邻居状态：

Down：邻居的初始化状态，表明当前没有从邻居收到过 Hello 报文，邻居关系处于断开状态。

Attemp：虽然当前没有从邻居收到过 Hello 报文，但仍需要尝试进行邻居关系的建立，以 HelloInterval 时间间隔向邻居周期性地发送 Hello 报文。该邻居状态只适用于 NBMA 网络类型。

Init：该状态表明当前与邻居仅为单向通讯关系，即收到了邻居发送的 Hello 报文，但该 Hello 报文中并没有宣告本地路由器，该类 Hello 报文称为 1-Way Hello 报文。

2-Way：表明邻居双方建立了双向通讯关系，即收到的邻居发送的 Hello 报文中宣告了本地路由器，该类 Hello 报文称为 2-Way Hello 报文。2-Way 状态是建立邻接前的最高状态，DR/BDR 就是从达到 2-Way 或更高邻居状态的路由器中选举出来的。建立双向邻居关系的路由器之间需要周期性地交互 2-Way Hello 报文来维持邻居状态。

ExStart：该状态是形成双向邻居关系的路由器之间开始建立邻接关系的第一步。该状态

下，邻居双方会进行主从关系和初始 DD 报文序列号的协商。主从关系的确定取决于邻居双方的路由器 ID，路由器 ID 较大的一方为主机，而另一方为从机，并且由主机来确定 DD 报文的初始序列号

Exchange：该状态下，邻居双方通过向对端发送 DD 报文来描述本地的 LSDB 汇总情况。DD 报文交互过程中，始终由主机来控制报文序列号，而从机始终以主机的序列号作为自己发送的 DD 报文序列号来进行应答。达到 Exchange 或更高邻居状态的路由器参与 LSA 洪泛过程。

Loading：在邻居双方交互 DD 报文后，本地路由器就会得知邻居路由器中有哪些 LSA 是本地 LSDB 中没有的。因此在该状态下，路由器通过向邻居发送 LSR 报文来请求所需的 LSA，同时收到邻居发送的 LSR 报文后通过 LSU 报文向邻居发送其所需要的 LSA。

Full：完全邻接状态，表明邻居双方达到了 LSDB 的同步，建立了完全邻接关系。

邻居状态的改变受一系列事件的影响，OSPFv3 协议中定义了 13 个邻居事件。

HelloReceived：表明收到了邻居发送的 Hello 报文，但 Hello 报文中并没有宣告本地路由器，即收到了邻居发送的 1-Way Hello 报文，触发进入 Init 邻居状态。

Start：该事件只适用于 NBMA 网络。触发该事件后路由器将以 HelloInterval 时间间隔向邻居发送 Hello 报文来尝试建立邻居关系。

2-Way Received：表明收到了邻居发送的 Hello 报文，并且 Hello 报文中宣告了本地路由器，即收到了邻居发送的 2-Way Hello 报文，此时表明邻居双方建立了双向通信关系，触发进入 2-Way 邻居状态。

NegotiationDone：表明邻居双方协商完毕，确定了主从关系和 DD 报文初始序列号，触发进入 Exchange 邻居状态，开始进行 DD 报文交互。

ExchangeDone：表明邻居双方完成了 DD 报文交互，获取了对方的 LSDB 汇总情况，触发进入 Loading 邻居状态，开始进行 LSDB 数据的同步。

LoadingDone：表明已经从邻居取得了所有需要的 LSA，即和邻居完成了 LSDB 的同步，触发进入 Full 邻居状态。

InactivityTimer：邻居的非激活定时器超时，表明最近 DeadInterval 时间间隔内没有收到过邻居发送的 Hello 报文，从而导致邻居状态进入 Down 状态。

KillNbr：表明邻居关系断开，无法与邻居继续进行通信，邻居状态进入 Down 状态。

LLDown：由下层协议触发，表明底层链路断开，邻居不可达，邻居状态进入 Down 状态。

1-Way：表明收到的邻居路由器发送的 Hello 报文中不再宣告本地路由器，即 1-Way Hello 报文，此时与该邻居不再是双向通讯关系。

AdjOK？：该事件触发进行是否需要和邻居建立或维持邻接关系的判定，判定结果可能导致一些邻接关系的建立和拆除。

SeqNumberMismatch：该事件表明 DD 报文交互过程中发生了错误。该事件可能有下列情形触发：（1）接收到含有意外序列号的 DD 报文；（2）接收到意外设定 I-bit 位的 DD 报文；

（3）接收到与上一个 DD 报文选项域不同的 DD 报文。

BadLSReq：收到的 LSR 报文中包含的 LSA 在本地 LSDB 中无法找到，即邻居请求了本地 LSDB 中并不存在的 LSA，表明在数据库交互过程中出现了错误。

2.3 LSA 处理

2.3.1 LSA 头部

OSPFv3 协议中的各类 LSA 都具有相同的 LSA 头部，长度固定为 20 字节[36]，如图 2.3 所示。

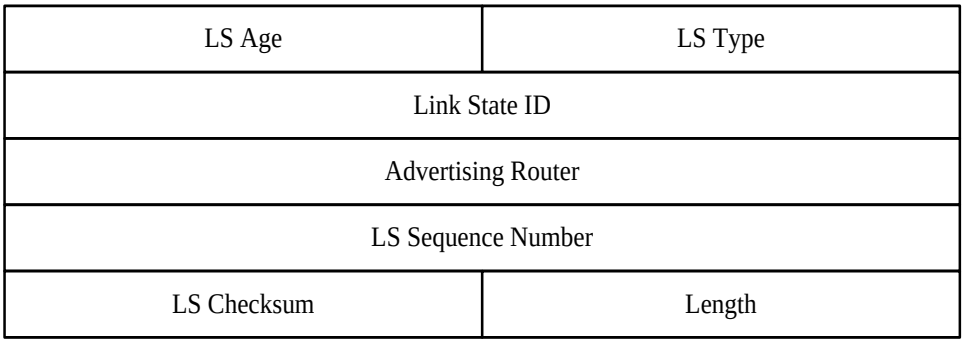

图 2.3　LSA 头部

其中各字段的具体含义如下：

LS Age：该字段反映了 LSA 自生成起所经过的时间，以秒为单位。新生成的 LSA 其 LS Age 为 0，LSA 保存在 LSDB 中其 LS Age 不断增加，发送 LSA 时也需要在 LS Age 上叠加相应接口的传输延迟时间。LS Age 达到 3600s 的 LSA 不再用于路由计算，可能需要对该 LSA 进行重新产生或删除处理。

LS Type：表明该 LSA 所属类型及其相关属性，LS Type 格式如图 2.4 所示。

图 2.4　LS Type 格式

其中高 3 比特位指定了该 LSA 的一些控制属性。U-bit 标识了对未知类型 LSA 的处理方式，若为 0 则将该 LSA 作为具有本地链路范围洪泛[37]的 LSA 来处理，若为 1 则以已知类型 LSA 的处理方式来存储和洪泛该 LSA。S2、S1 标识了该 LSA 的洪泛范围，如表 2.1 所示。

表 2.1　LSA 洪泛范围控制

S2	S1	洪泛范围
0	0	本地链路范围洪泛
0	1	区域范围洪泛
1	0	自治系统范围洪泛
1	1	保留

LSA Function Code 为 LSA 的类型编码，用于标识和区分各 LSA 的所属类型。

Link State ID：链路状态标识[38]。OSPFv3 协议中，Network-LSA 的链路状态标识为 DR

在对应链路上的接口 ID，Link-LSA 的链路状态标识为路由器在对应链路上的接口 ID，而其他类型 LSA 的链路状态标识不具有任何意义。

Advertising Router：生成该 LSA 的路由器 ID。

LS Sequence Number：LSA 所对应的序列号，用于判定 LSA 的新旧。

LS Checksum：该字段为除了 LS Age 域外 LSA 各域的校验和。

Length：该字段为除去 LS Age 域外，LSA 各域的长度。

2.3.2 LSA 生成、刷新和老化

运行 OSPFv3 协议的每台路由器都会生成相应的 LSA。接入网络的路由器开始工作后都会生成 Router-LSA、Link-LSA 和 Intra-Area-Prefix-LSA；DR 会生成 Network-LSA 和 Intra-Area-Prefix-LSA；ABR 会生成 Inter-Area-Prefix-LSA 和 Inter-Area-Router-LSA；ASBR 会生成 AS-external-LSA。当路由器生成 LSA 的新实例时，会增加 LSA 的序列号，将 LS Age 设为 0，计算 LSA 的校验和，然后将 LSA 加入 LSDB 中并通过相应的接口洪泛给邻居路由器。

当路由器自己生成的 LSA 的 LS Age 达到 LSRefreshTime（1800s）时，路由器需要重新生成该 LSA 的新实例来刷新该 LSA，即使该 LSA 中的具体内容未发生变化也需要进行刷新处理。这种 LSA 的定时刷新机制增强了链路状态算法的健壮性。

当 LSA 从路由器接口洪泛出去时，LS Age 需要叠加上相应接口的传输延迟时间。当 LSA 保存在 LSDB 中时，LSA 的 LS Age 也不断增加，即进行老化处理。LSA 的 LS Age 绝不会超过 3600s，达到 3600s 的 LSA 将不再用于路由计算，需要从网络中废止，这可以通过将达到 3600s 的 LSA 作为新生成的 LSA 洪泛出相应接口来实现。在邻居双方进行 DD 报文交互时，LSDB 中达到 3600s 的 LSA 不会被添加到 LSDB 汇总列表中，而是添加到邻居的链路状态重传列表中。

当路由器宣告的某条路径不再可达，需要通知其他邻居删除该条路由时，或者自己生成的 LSA 在网络中回绕时，需要提前老化当前的 LSA。提前老化即不增加 LSA 的序列号而将 LS Age 设为 3600s, 并重新进行洪泛来将 LSA 从网络中废止的机制。路由器只能提前老化自己生成的 LSA，而不能提前老化其它路由器生成的 LSA。

2.3.3 LSA 洪泛

OSPFv3 协议中提供了 LSA 的洪泛机制,通过 LSU 报文携带一个或多个 LSA，并将 LSA 洪泛到距离其起源更远的一跳。为了确保洪泛机制的可靠性，洪泛过程中使用 LSA 的重传机制[39]，即发送 LSA 时将 LSA 添加到邻居的链路状态重传列表中，若超过一定时间仍未收到邻居的 LSAck 应答报文则对 LSA 进行重传，收到邻居的 LSAck 应答报文后再从邻居的链路状态重传列表中摘除相应的 LSA。

当收到邻居发送的 LSU 报文时开始洪泛过程处理[40]。首先，需要对所接收到的 LSU 报文进行一致性检查，包括 LSU 报文是否与特定的邻居和区域相关联；未达到 Exchange 状态的邻居不参与洪泛过程，直接丢弃不作处理。其次，还需要对 LSU 报文中携带的 LSA 进行

LSA 类型、LSA 校验和等检查。

若所有检查都通过，则从本地 LSDB 中查找该 LSA 的实例，若找到则比较本地 LSDB 中的 LSA 与所收到的 LSA 的新旧关系，具体判定方法如下：首先比较两者的报文序列号，具有较大序列号的 LSA 较新；若两者序列号相同，则比较两者的校验和，具有较大校验和的 LSA 较新；若两者的校验和也相同，则检查两者的 LS Age，LS Age 达到 3600s 的 LSA 较新；否则，若两者的 LS Age 差异大于 MaxAgeDiff（900s），则 LS Age 较小的 LSA 较新；否则，两个 LSA 为同一实例。根据判定结果，处理如下：

（1）收到的 LSA 较新

当本地 LSDB 中不存在所收到 LSA 的实例或者收到的 LSA 比 LSDB 中的 LSA 新时，则将本地 LSDB 中的 LSA 实例从所有邻居的链路状态重传列表上摘除，将所收到的较新的 LSA 添加到 LSDB 中，并且根据最新的 LSA 重新进行路由计算刷新路由表项，同时向邻居发送 LSAck 报文进行确认。此外，还需通过路由器的相应接口将最新的 LSA 洪泛给网络中的其它邻居路由器。

（2）收到的 LSA 与本地 LSDB 中的 LSA 为同一实例

当本地 LSDB 中的 LSA 与所收到的 LSA 为同一实例，并且该 LSA 处于该邻居的链路状态重传列表中，则表明本地路由器正在等待邻居路由器对该 LSA 的确认。此时，将该 LSA 视作隐含确认，从邻居的链路状态重传列表中摘除该 LSA，并向邻居发送 LSAck 报文来进行确认。

（3）收到的 LSA 较旧

当本地 LSDB 中的 LSA 比所接收到的 LSA 新时，则需要通过 LSU 报文将本地 LSDB 中较新的 LSA 发送给邻居，并且对于所接收到的 LSA 不做确认。该 LSA 将被直接发送给邻居，而不是添加到邻居的链路状态重传列表中。

洪泛过程中发送携带最新 LSA 的 LSU 报文时，需要进行合格接口的选取，具体的选取标准如下：若 LSA 的类型为已知类型，则合格接口的选取取决于该 LSA 的洪泛范围。对于自治系统范围洪泛的 LSA，合格接口包括除虚链路以外的所有接口。此外，AS-external-LSA 不会通过接入 Stub 区域的接口进行发送。对于区域范围洪泛的 LSA，合格接口包括接入该区域的所有接口。对于本地链路范围洪泛的 LSA，合格接口只有接入该链路的接口一个。若 LSA 的类型为未知类型并且其 U-bit 位为 0，则该未知类型 LSA 为本地链路范围洪泛，合格接口只有接收该 LSA 的接口一个。若 LSA 的类型为未知类型并且其 U-bit 位为 1，则将该未知类型 LSA 当做已知类型 LSA 来进行存储和洪泛，合格接口的选取同 LSA 类型为已知类型的情况。

OSPFv3 协议中共有三种洪泛范围：本地链路范围洪泛、区域范围洪泛和自治系统范围洪泛。具体的洪泛范围由 LSA 头部中的 S1、S2 比特位指定，不同的 LSA 具有不同的洪泛范围：Link-LSA 在本地链路范围进行洪泛；Router-LSA、Network-LSA、Inter-Area-Prefix-LSA、Inter-Area-Router-LSA、Intra-Area-Prefix-LSA 在区域范围进行洪泛；AS-external-LSA 在自治

系统范围进行洪泛。

2.4 本章小结

本章介绍了 OSPFv3 协议的基本概念和相关术语。邻接关系的建立和 LSDB 数据的维护是 OSPFv3 协议运行的基础和核心所在，文中结合接口状态机和邻居状态机对邻接关系的建立过程进行了详细分析，同时重点阐述了 LSA 的生成、刷新、老化和洪泛机制。本章剖析了 OSPFv3 协议中的重点和难点，为全文奠定了理论基础。

第3章 GR 机制

3.1 问题的提出

当今的主流路由器为了满足IP网络快速发展的需要，都由集中式设备向分布式设备转变。在分布式路由器设备中，其控制层面和数据转发层面相分离。控制层面主要负责系统控制和任务管理，通常由主控制板和备控制板组成，备控制板作为主控制板的备份并在主控制板发生故障失效时转变为主控制板。数据转发层面则主要负责数据流量的接收和转发任务，通常由接口板组成。

通常情况下，运行 OSPFv3 协议的路由器控制层面进行协议重启或发生主备倒换时，都会重新进行邻居关系的建立和 LSDB 数据的同步。如图 3.1 中所示，RT2 与 RT1、RT3 和 RT4 建立了全邻接关系，网络处于稳定状态。此时，RT2 控制层面进行协议重启或主备倒换操作，当 RT2 的备控制板转变为主控制板开始运行后，会通过所有接口发送 1-Way Hello 报文来发现周边路由器并尝试建立邻居关系。而周边路由器 RT1、RT3 和 RT4 由于之前已经与 RT2 建立了全邻接关系，为了避免由于缺少 LSDB 的同步而造成路由环路或黑洞路由，周边路由器 RT1、RT3 和 RT4 在收到 RT2 发送的 1-Way Hello 报文后认为 RT2 在报文转发路径上已不可用，会将 RT2 从邻居列表中删除，断开与 RT2 的邻居关系，重新生成相应的 LSA 来更新链路状态信息并通知网络中的其它路由器，同时进行路由计算刷新路由表项，将 RT2 从相应的报文转发路径上移除。当 RT2 控制层面协议重启或主备倒换结束，与 RT1、RT3 和 RT4 再次建立全邻接关系并同步 LSDB 数据后，周边路由器 RT1、RT3 和 RT4 也会重新生成相应的 LSA 并进行路由计算刷新路由表项，重新将 RT2 添加到相应的报文转发路径上。这样就会造成在 RT2 控制层面协议重启或主备倒换过程中，网络中出现路由振荡和数据转发流量中断，从而降低了网络的可靠性和稳定性。对于一个大型网络，尤其是运营商网络和企业网络，这种路由振荡和转发业务中断是不可容忍的。

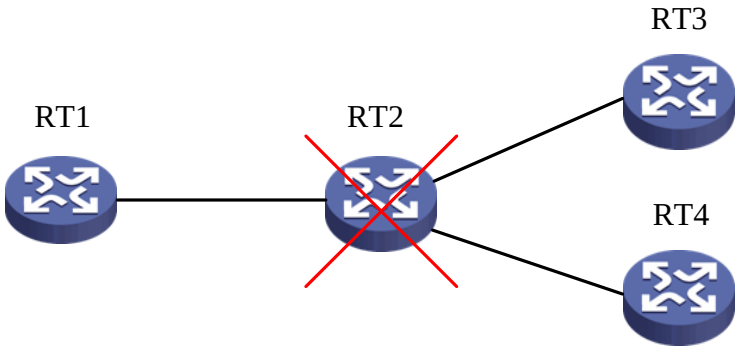

图 3.1　OSPFv3 协议重启

然而，如果在路由器控制层面协议重启或主备倒换过程中，网络的拓扑结构仍然保持稳定不变，重启路由器仍然能保证它的转发表正常地进行报文转发，则该重启路由器在相应的

报文转发路径上仍然是安全可用的，周边路由器可继续维持与该重启路由器的邻居关系，无需删除相应的邻居信息、重新产生相应的 LSA 并重新进行路由计算。

3.2 GR 机制简介

GR 机制是一种高可靠性技术，能在路由器设备控制层面进行协议重启或主备倒换过程中保证转发层面的数据流量不中断，从而实现平滑重启功能。GR 机制的核心在于：当路由器设备控制层面进行协议重启或主备倒换操作时，能够通知其周边路由器设备，使周边路由器设备到该重启路由器设备的邻居关系和路由信息在一定时间内保持稳定。在协议重启或主备倒换结束后，周边路由器设备协助重启路由器设备进行拓扑、路由和会话等数据信息的恢复和同步，在尽量短的时间内恢复到重启前的状态。这样，在路由器设备控制层面协议重启或主备倒换过程中不会产生路由振荡，报文转发路径也没有任何变化，不会出现数据转发业务的中断，整个系统可以实现不间断运行，从而提高了网络的可靠性和稳定性[41]。

3.2.1 GR 过程中的角色

运行 OSPFv3 协议的路由器在 GR 重启过程中可分为两种角色[42]：

GRRestarter：即控制层面发生协议重启或主备倒换并且具有 GR 能力的路由器设备，也称为 GR 重启路由器。

GRHelper：即和发生协议重启或主备倒换的路由器具有邻居关系，并协助其完成 GR 重启流程的路由器设备，也称为 GR 帮助路由器。一台路由器设备可以同时成为多台重启路由器设备的 GRHelper 角色。

3.2.2 GR 机制分类

目前，GR 机制的实现方式主要有两种：IETF 标准 GR 机制和非 IETF 标准 GR 机制。IETF 标准 GR 机制基于 IETF 标准设计实现，在 GR 重启过程中，GRRestarter 设备通过向 GRHelper 设备发送 Grace-LSA 报文来控制 GR 交互流程。非 IETF 标准 GR 机制不是基于 IETF 标准设计实现的，在 GR 重启过程中，GRRestarter 设备与 GRHelper 设备之间通过相互发送携带 LLS（Link-Local Signaling，本地链路信令）[43]和 OOB（Out-of-band LSDB Resynchronization，带外的链路状态数据库重新同步）[44]扩展信息的 OSPFv3 协议报文来控制 GR 交互流程。IETF 标准 GR 机制和非 IETF 标准 GR 机制的实现是互斥的，在路由器设备上同时只能配置其中的一种 GR 能力。本课题主要对 IETF 标准 GR 机制进行研究和实现。

根据进入 GR 重启流程方式的不同，GR 机制又可分为计划性 GR 机制和非计划性 GR 机制。计划性 GR 机制即事先知道要进行 GR 重启，主动进入 GR 重启流程的一种实现方式，如软件版本升级等场景。对于计划性 GR 机制，可以在进入 GR 重启流程前做一些准备工作。非计划性 GR 机制即事先不知道何时会进行 GR 重启，被动进入 GR 重启流程的一种实现方式，如进程异常或单板异常导致协议重启等场景。对于计划性 GR 机制，成为 GRRestarter 的路由器设备控制层面不需要具有主备环境即可完成 GR 重启流程。而对于非计划性 GR 机制，成为 GRRestarter 的路由器设备控制层面必须具有主备环境，否则无法完成 GR 重启流程。本

课题主要对非计划性 GR 机制进行研究和实现。

3.3 Grace LSA

Grace-LSA 是 OSPFv3 协议中一种特殊类型的 LSA，由路由器设备在进行 GR 重启时生成，用于通告周边的邻居路由器设备并与其进行 GR 重启协商。Grace-LSA 中携带了相应的 GR 重启信息[45]，具体报文格式如图 3.2 所示。

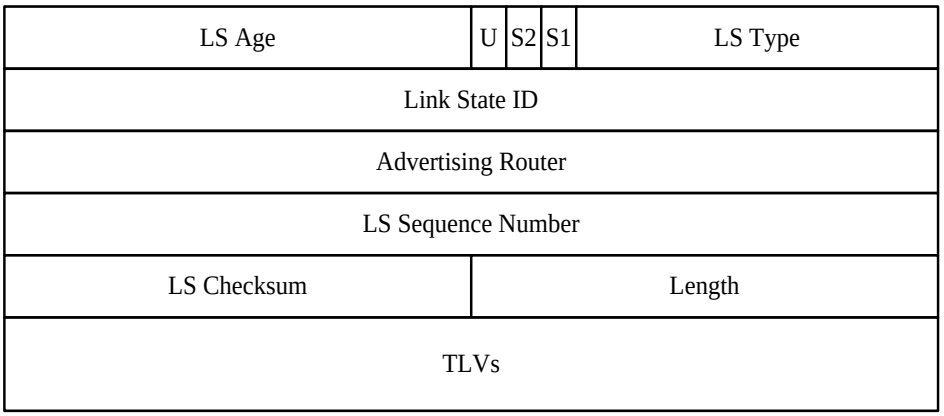

图 3.2　Grace-LSA 格式

Grace-LSA 和 OSPFv3 协议中其它七种类型的 LSA 具有相同的长度为 20 字节的 LSA 头部格式[46]。其中，LS Age 的当前值表明了进行协议重启或主备倒换操作的路由器设备进入 GR 重启流程后所经过的时间。OSPFv3 协议中规定 Grace-LSA 只在本地链路范围进行洪泛，因此 S1、S2 比特位固定为 0。其余字段的设定和含义均与普通类型的 LSA 相同，此处不再赘述。TLV 部分为 Grace-LSA 的数据部分，具体格式如图 3.3 所示。

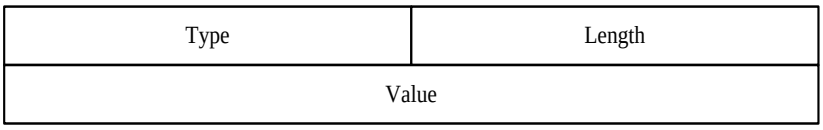

图 3.3　Grace-LSA 中的 TLV 格式

OSPFv3 协议中，每个 Grace-LSA 都固定携带有两个 TLV 三元组：

（1）Type = 1，Length = 4 的 TLV，此时 Value 为 GR 周期时间长度。GR 周期时间长度由进行协议重启或主备倒换操作的路由器设备在进入 GR 重启流程时设定，用于通告周边的邻居路由器设备在该时间段内继续维持到该重启路由器的邻居关系和路由信息的稳定，并在重启路由器协议重启或主备倒换结束后协助其进行拓扑、邻居和路由信息的同步。

（2）Type = 2，Length = 1 的 TLV，此时 Value 为路由器设备进入 GR 重启流程的原因，由进行协议重启或主备倒换操作的路由器设备在进入 GR 重启流程时设定。Value 取值为 0 表示未知原因，为 1 表示软件重启，为 2 表示软件重载/升级，为 3 表示主备倒换。

3.4 GR 机制工作原理

支持 GR 重启能力的路由器设备在控制层面协议重启或主备倒换过程中，为了实现转发层面数据流量的不中断，必须满足如下条件：

（1）GR 重启过程中，重启路由器的转发表项保持稳定不变。

（2）协议重启结束后，重启路由器能够重新获取网络中的有效邻居信息，重新建立所有邻居关系。

（3）邻居关系重建后，重启路由器能够重新获取网络中的 LSDB 数据，即能够获取完整的网络拓扑信息。

当使能了 GR 能力的路由器设备控制层面进行协议重启或主备倒换操作时，会首先向周边的邻居路由器设备发送 Grace-LSA 报文来通告邻居路由器本端设备进入 GR 重启流程，该重启路由器自身则成为 GRRestarter。周边的邻居路由器在收到重启路由器发送的 Grace-LSA 报文后，若支持 GRHelper 能力则成为 GRHelper，进入 GR 重启流程，协助重启路由器完成 GR 重启。在 GR 重启过程中，邻居路由器设备会在 Hello 报文和 LSA 中继续通告与重启路由器设备的邻居关系，保证网络中的路由稳定和数据转发流量不中断。当重启路由器设备控制层面协议重启或主备倒换结束，与周边的路由器重新建立双向邻居关系后，周边的路由器设备协助其进行 LSDB 数据的恢复和同步。LSDB 数据同步完成后，重启路由器和周边路由器的邻居关系重新达到全邻接状态，GR 重启流程结束，重启路由器退出 GRRestarter 模式。此时，重启路由器会向周边的邻居路由器发送 LS Age 为 3600s 的 Grace-LSA 报文，通知邻居路由器本端设备 GR 重启流程结束。周边的邻居路由器设备在收到该 Grace-LSA 报文后，退出 GRHelper 模式，结束 GR 重启流程。在退出 GRRestarter 模式后，重启路由器根据最新的 LSDB 信息重新进行路由计算更新路由表和转发表，删除失效的路由信息，从而完成协议收敛。

3.5 本章小结

本章通过对运行 OSPFv3 协议的路由器设备协议重启或主备倒换过程的具体分析，指出其问题所在，从而引入 GR 机制，介绍了 GR 机制的基本概念，包括 GR 过程中的角色划分和 GR 机制的实现分类。此外，文中还对 GR 重启过程中使用的 Grace-LSA 的报文格式和具体内容进行了详细介绍。最后，重点阐述了 GR 机制的基本工作原理，以及 GR 重启过程中路由器设备之间的交互过程。

第4章 总体设计

4.1 软件开发环境

本课题的软件系统是基于某通信公司研制的软件平台进行设计实现的，该软件平台的具体架构如图 4.1 所示。

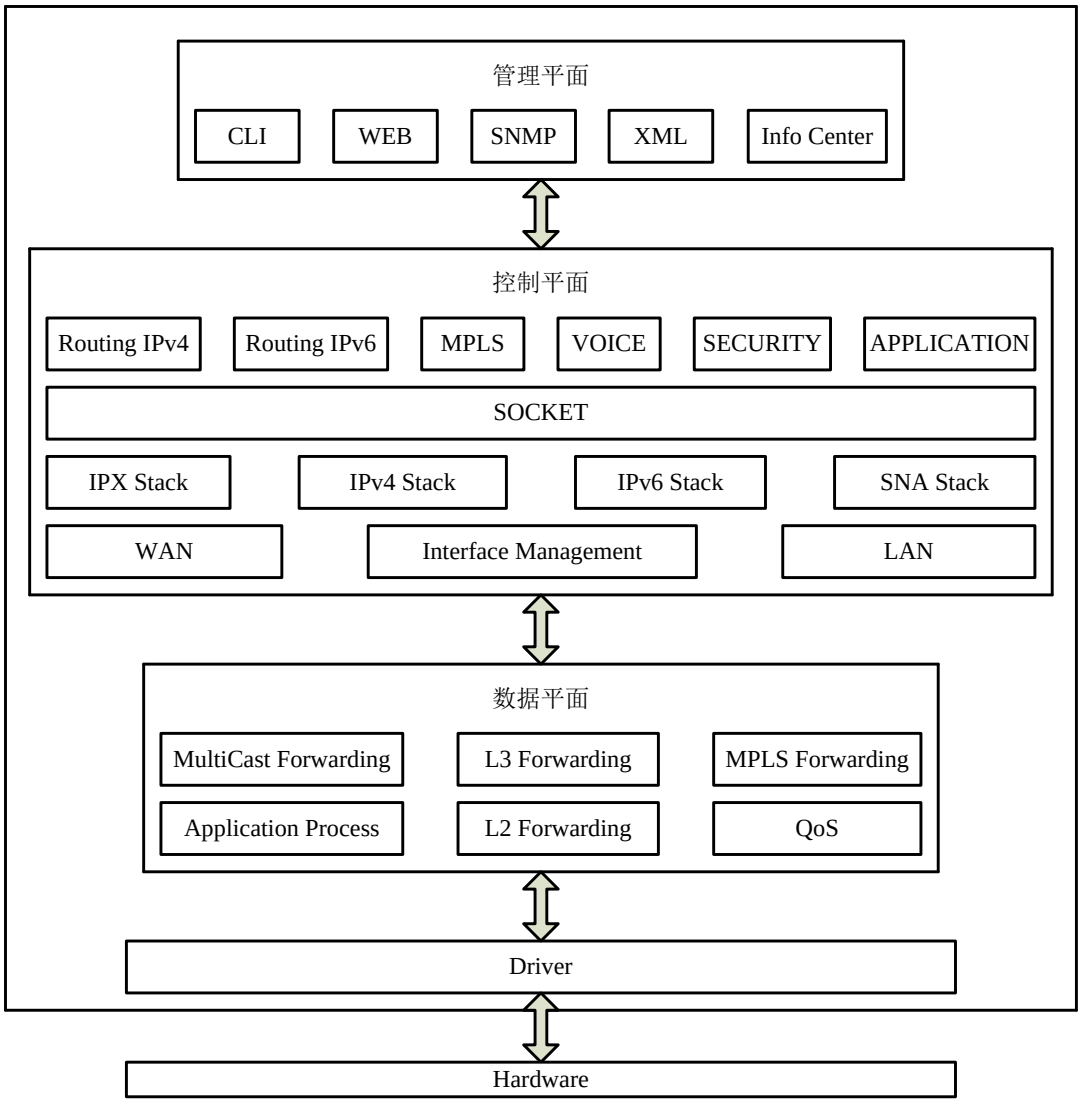

图 4.1　软件平台体系架构

该软件平台的体系架构以 IPv4/IPv6 协议栈为基础，其中集成了多种数据通信网络的技术，主要包括 IPv4/IPv6 路由技术、MPLS 技术、网络语音技术、网络安全技术、二层转发技术、三层转发技术、MPLS 转发技术和 QoS 技术等特性，是目前业内最为成熟的网络操作系统之一。该软件平台从上到下依次分为 4 个平面：管理平面、控制平面、数据平面和支撑平面。其中，管理平面主要对外提供整个体系架构的管理接口，如命令行、WEB 管理和 SNMP 管理等，实现对整个软件平台的管理；控制平面通过运行路由、MPLS、网络安全等各种路由、

信令和控制协议来控制整个软件平台的运转；数据平面主要负责数据报文的接收和转发功能；支撑平面在操作系统的基础上为整个平台提供业务运行的软件基础。本课题中研究实现的系统模块位于该软件平台控制平面中的 IPv6 路由技术部分。

4.2 总体方案设计

4.2.1 可选方案

本课题中的软件系统基于开源的 Linux 操作系统进行开发，系统整体设计遵循成熟的 Unix 系统架构，支持抢占式调度。目前，软件系统的总体设计方案主要有如下两种：

方案一：单进程[47]实现，整体系统以一个独立进程的方式存在，内部不划分线程，统一进行协议报文、接口数据、邻居数据、LSDB 数据和路由数据等的维护和处理。

方案二：多线程[48]实现，在系统进程下继续划分出多个线程，协议报文、接口数据、邻居数据、LSDB 数据和路由数据等由不同的线程分别进行维护和处理。

4.2.2 方案选择与论证

对于方案一，将整体系统作为一个单一进程来进行处理，实现上较为简单，不会造成数据访问的冲突和时序问题，但是在系统处理上效率较低，不利于进行并行程序设计。对于方案二，采用多线程程序设计，不同的数据由不同的线程分别进行处理，可以利用多核 CPU 的并行处理能力来提高数据处理效率和系统整体性能，但可能造成多个线程之间数据访问的冲突和时序问题[49]。

另一方面，从协议本身出发，本课题设计实现的软件系统处理流程较为复杂，数据处理较为繁琐，涉及系统事件、协议报文、接口状态机、邻居状态机、LSDB 数据、路由计算和路由引入等多方面的处理和维护，同时系统内部各个模块之间需要进行相互操作和访问，交互性较强，如图 4.2 所示。

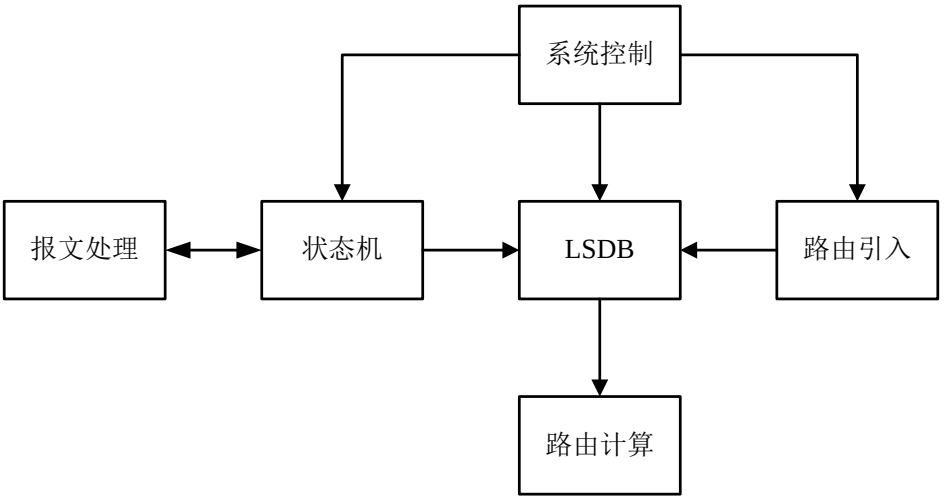

图 4.2　内部模块交互关系

（1）协议报文的收发属于 I/O 密集型操作，需要及时进行处理。如果将 Hello 报文和其它数据报文等同看待，在存在大量邻居的情况下，会由于大量数据报文的积压以及系统忙于其它处理而造成 Hello 报文不能得到及时响应，从而导致网络中的邻居振荡。因此，需要将

Hello 报文和其它数据报文分开，为 Hello 报文分配单独的报文缓冲区进行优先处理，刷新邻居超时定时器以维持邻居关系的稳定，然后再上送状态机进行处理。

（2）路由计算属于 CPU 消耗型操作，是整个 OSPFv3 协议的关键，实现较为复杂。在实际网络中，路由计算所消耗的时间通常占路由收敛总时间的一半。因此，需要将 LSDB 的维护和路由计算进行分离，将路由计算作为一个单独的线程来处理。

（3）LSDB 是 OSPFv3 协议实现的核心，也是数据量最大的一部分。同时，路由计算、系统事件和状态机模块都会访问 LSDB 进行数据读取和更新，对 LSDB 的操作遍布在各种流程当中，若不将 LSDB 的维护分离出来，很容易造成数据访问错误。另一方面，接口状态机和邻居状态机的处理对 LSDB 数据的依赖性强，存在大量相互引用的情况，数据保护复杂度较高、风险较大。因此，需要将接口状态机、邻居状态机和 LSDB 的维护作为一个单独的线程来进行处理。

（4）路由引入的触发场景较多，如用户配置触发、接口变化触发和 ASBR 角色变化触发等，并且路由引入的触发机制和引入路由处理较为繁琐，因此，有必要将路由引入作为一个单独的线程来进行处理。

（5）针对多个线程之间数据访问的冲突和时序问题，可以利用 Linux 操作系统中提供的线程同步机制[50]来解决。对于多个线程共同访问的内存数据，用锁来进行保护，通过加锁和解锁机制，保证了各线程间数据访问的正确性和有序性。

综上所述，本课题具体实现时选择方案二，采用多线程程序设计，在系统进程下继续划分出多个线程来进行并行处理。

4.2.3 线程划分

整个软件系统的维护和处理主要包括如下几部分：系统事件、协议报文、状态机、LSDB 数据、路由计算和路由引入。其中，系统事件处理主要包括对配置命令、接口事件、地址事件的响应和处理；协议报文处理主要包括对 Hello、DD、LSR、LSU 和 LSAck 五类协议报文的接收、解析和发送处理；状态机处理包括对接口状态机和邻居状态机的维护和处理；LSDB 数据处理包括对各类 LSA 报文的生成、刷新、老化和洪泛处理；路由计算处理包括对拓扑信息、地址前缀信息和下一跳信息的提取，以及路由信息生成和路由下发的处理；路由引入处理包括对外部路由信息的引入和重发布的处理。

因此，根据具体功能的不同，设计实现时在系统进程下划分出五个线程：

（1）系统控制线程：负责整体系统的对外交互和任务控制，同时通过调度其它线程来实现对整体系统的控制和维护。

（2）报文预处理线程：负责对 Hello 报文进行预处理、邻居超时定时器的维护以及 Hello 报文的周期性发送。

（3）LSDB 和状态机线程：负责数据报文的处理、接口信息和邻居信息的维护以及 LSDB 数据的维护。根据相应的接口事件和邻居事件来推动接口状态机和邻居状态机的变迁，进而

推动整个 OSPFv3 协议的运转。同时，进行 LSA 数据的处理，为路由计算提供相应的拓扑信息、地址前缀信息和下一跳信息。

（4）路由计算线程：负责路由计算和路由下发处理，即根据 LSDB 中的各类 LSA 生成相应的路由信息并下发路由表以指导报文转发，具体包括拓扑信息提取、拓扑计算、地址前缀信息提取、地址前缀信息计算和下一跳信息的维护等。

（5）路由引入线程：负责路由引入和激活路由变化处理，即将其他协议的路由信息引入到 OSPFv3 协议内部进行重新发布。

4.3 外部模块交互

本系统基于的软件平台采用模块化设计[51]原则，模块内部的具体实现对其它模块不可见，各模块之间通过相应接口进行联系，从而降低了各模块之间的依赖性。本课题所研究实现的模块为 OSPFv3 模块，与外部其它模块之间的交互关系如图 4.3 所示，主要包括 IPv6 协议栈、配置管理、信息中心、路由管理和 RIB 表，而接口管理、地址管理和 HA 等模块并不直接与 OSPFv3 模块进行交互，而是通过路由管理模块来进行间接交互。

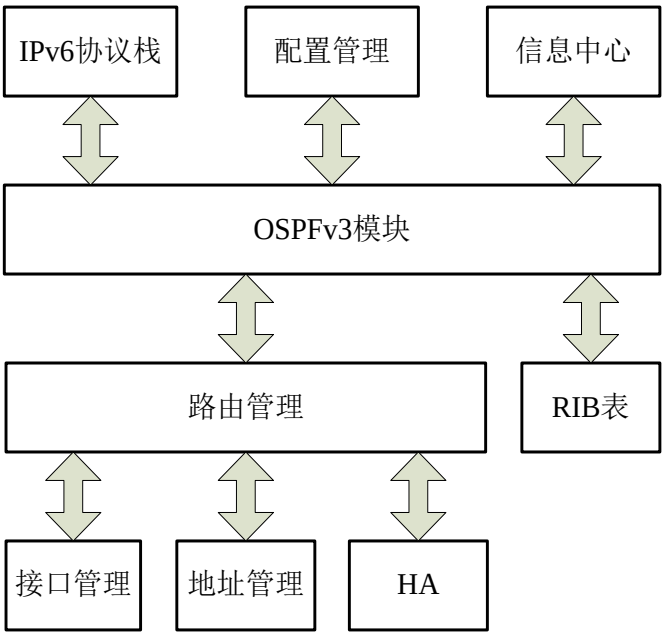

图 4.3　外部模块交互关系

（1）IPv6 协议栈

由于 OSPFv3 协议是基于 IPv6 网络来进行设计的，所有的协议报文都基于 IPv6 协议，因此 OSPFv3 协议模块通过与 IPv6 协议栈的交互来实现对 IPv6 协议报文的接收和发送。

（2）配置管理

命令行模块主要完成人机交互功能，提供相应配置信息和管理信息的输入输出通道。OSPFv3 协议命令行模块运行于命令行守护进程上下文，作为独立的插件链接到配置管理模块中，主要负责 OSPFv3 协议命令的注册与解析，并通过进程间通信机制将相应的命令参数发送给 OSPFv3 协议模块。

（3）信息中心

信息中心模块为其它模块提供了统一的信息输出接口。目前，OSPFv3 协议的输出信息主要有显示命令输出信息、调试信息、系统日志信息和系统告警信息。

（4）路由管理

路由管理是 OSPFv3 模块的支撑模块，为 OSPFv3 协议提供运行平台并屏蔽外部事件输入，是 OSPFv3 协议正常运行的基础。接口管理提供接口相关信息，如接口类型、接口状态和接口属性变化等。地址管理提供 IPv6 地址信息。HA 提供主备状态控制及主备控制板之间数据备份的通道。OSPFv3 协议模块不直接和接口管理、地址管理和 HA 等模块进行联系，而是由路由管理模块进行封装屏蔽，接口事件、地址事件和 HA 事件等也通过路由管理模块分发给 OSPFv3 协议模块。

（5）RIB 表

RIB 表是各类路由信息的汇总。OSPFv3 协议进行路由计算后通知路由管理模块对 RIB 表中的路由信息进行增加、删除或更新操作。在进行路由引入时，OSPFv3 协议模块也需要访问 RIB 表引入外部路由信息。

4.4 公用数据结构设计

整体系统中的公用数据结构按照“进程→区域→接口→邻居”的层次关系进行设计：进程下包含多个区域，每个区域下又包含多个接口，每个接口都属于特定的进程和区域，同时接口下又记录了相应的邻居信息。

4.4.1 进程数据结构

OSPFv3 协议的进程数据结构中包含了进程 ID、路由器 ID、路由器类型和进程下区域等基本信息，也包含所有的 AS-external-LSA 和 OSPFv3 协议路由表。进程数据结构下维护的一些主要字段如下：

```
typedef struct tagPROC
{
    ……
    unsigned short    usProcessID;          /*  进程 ID */
    unsigned int      uiRouterID;           /*  路由器 ID */
    unsigned char     ucRtrType;            /*  路由器类型  */
    unsigned char     ucProcStatus;         /*  进程状态  */
    SORTLIST_S        sllArea;              /*  进程下区域链表  */
    LSDB_S            *pstLSDB;             /*  进程下 LSDB，存储  AS-external-LSA */
    HTB_S             htIntraRtrTbl;        /*  区域内部路由表  */
    HTB_S             htASBRRtrTbl;         /* ASBR 路由表  */
    RADIX6_S          stIntraASRdx;         /*  自治系统内部路由表  */
```

```
    RADIX6_S      stInterASRdx;          /* 自治系统间路由表 */
    ……
} PROC_S;
```

4.4.2 区域数据结构

OSPFv3 协议的区域数据结构中描述了区域的基本信息，如区域 ID、区域类型、接入该区域的接口等。同时，区域范围洪泛的 LSA 也存储在区域数据结构下，包括 Router-LSA、Network-LSA、Inter-Area-Prefix-LSA、Inter-Area-Router-LSA 和 Intra-Area-Prefix-LSA。区域数据结构下维护的一些主要字段如下：

```
typedef struct tagAREA
{
    ……
    unsigned int      uiAreaID;          /* 区域 ID */
    unsigned char     ucAreaType;        /* 区域类型 */
    unsigned char     ucAreaStatus;      /* 区域状态 */
    SORTLIST_S        sllIf;             /* 接入该区域的接口链表 */
    LSDB_S            *pstLSDB;          /* 区域下 LSDB，存储区域范围洪泛 LSA */
    unsigned char     ucAreaTransCap;    /* 区域传输能力，标志是否为传输区域 */
    ……
} AREA_S;
```

4.4.3 接口数据结构

OSPFv3 协议的接口数据结构中描述了接口相关的基本信息，如接口网络类型、接口状态、接口相关的定时器间隔、接口下 DR/BDR 信息等。其中一些接口参数必须在接入同一个网络的所有路由器之间保持一致，否则无法建立邻居关系，如接口 Hello 定时器间隔和 Dead 定时器间隔等。此外，本地链路范围洪泛的 Link-LSA 也存储在接口数据结构下。接口数据结构下维护的一些主要字段如下：

```
typedef struct tagIF
{
    ……
    unsigned char     ucIfNetworkType;   /* 接口网络类型 */
    unsigned char     ucIfState;         /* 接口状态 */
    unsigned char     ucPriority;        /* 接口路由优先级 */
    unsigned short    usHelloInterval;   /* Hello 定时器间隔 */
    unsigned short    usDeadInterval;    /* Dead 定时器间隔 */
    unsigned short    usRxmtInterval;    /* 重传定时器间隔 */
```

```
    unsigned short    usInfTransdelay;     /* 接口传输延迟时间 */
    unsigned int      uiCost;              /* 接口链路代价 */
    unsigned int      uiDR;                /* 接口的 DR 信息 */
    unsigned int      uiBDR;               /* 接口的 BDR 信息 */
    ADDR_V6_S         stLinkLocalAddr;     /* 接口 LinkLocal 地址 */
    SORTLIST_S        stIPv6AddrList;      /* 接口 IPv6 全球单播地址 */
    LSDB_S            *pstLSDB;            /* 接口下 LSDB，存储本地链路范围洪泛 LSA */
    SORTLIST          sllNbr;              /* 接口下的邻居链表 */
    ……
} IF_S;
```

4.4.4 邻居数据结构

OSPFv3 协议的邻居数据结构中包含了网络中的两台路由器之间形成邻居关系所需要的相关信息，如邻居路由器 ID、邻居状态、邻居地址和邻居选项等基本信息。同时也包括了建立邻接关系所需要的数据信息，如 DD 报文序列号、数据库汇总列表、链路状态重传列表和链路状态请求列表等。邻居数据结构中维护的一些主要字段如下：

```
typedef struct tagNBR
{
    ……
    unsigned int      uiNbrID;             /* 邻居路由器 ID */
    unsigned char     ucNbrState;          /* 邻居状态 */
    unsigned char     ucNbrPriority;       /* 邻居路由优先级 */
    IP6ADDR_S         stNbrAddr;           /* 邻居的接口地址 */
    unsigned int      uiNbrOption;         /* 邻居选项 */
    unsigned long     ulInactivityTimer;   /* 邻居的非激活定时器 */
    unsigned char     ucNbrFlag;           /* 主从角色 */
    unsigned int      uiDDSeqNum;          /* DD 报文序列号 */
    unsigned int      uiNbrDr;             /* 邻居的 DR 信息 */
    unsigned int      uiNbrBdr;            /* 邻居的 BDR 信息 */
    SORTLIST_S        sllDBSumList;        /* 数据库汇总列表 */
    SORTLIST_S        sllLSRxmtList;       /* 链路状态重传列表 */
    SORTLIST_S        sllLSRequestList;    /* 链路状态请求列表 */
    ……
} NBR_S;
```

4.5 本章小结

本章首先对软件系统的开发环境进行了介绍，并基于该软件平台对整体系统进行总体架构设计。文中给出了两种可行的设计方案，并结合软件系统需求和 OSPFv3 协议自身运行特点对设计方案进行了论证，选择采用多线程程序设计方案来进行设计实现，同时对内部线程进行了功能划分，阐述了 OSPFv3 模块与外部其他模块的交互关系。此外，文中还对整体系统的公用数据结构进行了设计，为下文各模块的设计实现奠定了基础。

第5章 GRRestarter 模块设计

本课题设计实现的 GR 机制为非计划性 GR 机制，因此能够成为 GRRestarter 的路由器设备需要具有主备环境，即路由器设备的控制层面由主控制板和备控制板组成，备控制板平时作为主控制板的备份并在当前主控制板重启或发生故障失效时转变为新的主控制板。

5.1 需求点分析

在对 GRRestarter 模块进行软件设计时，需要支持如下需求点：

（1）只有满足特定条件时，路由器才能进入 GR 重启流程。

（2）进行 GR 重启时，路由器能够正常地发送 Grace LSA 来通知周边的路由器设备，并且保证 Grace LSA 先于所有 Hello 报文进行发送。

（3）在 GR 重启过程中，路由器能够重新获取之前的邻居信息和 LSDB 数据信息，进行数据恢复。

（4）在 GR 重启过程中，路由器能够维持路由信息和转发表的稳定，保证数据转发流量不中断。

（5）正常退出 GR 重启流程后，路由器能够完成对 LSDB 数据和路由表项的更新。

（6）GR 重启过程中出现异常情况时，路由器能够完成对邻居信息、LSDB 数据和路由信息的后续处理。

5.2 GR 状态机设计

GRRestarter 在 GR 重启流程中的处理需要分阶段依次进行，同时需要各个线程之间相互调度进行协调处理，因此在 GRRestarter 模块中需要设计一个高效可靠的状态机，来控制 GRRestarter 在 GR 重启过程中的报文交互、协议数据同步、路由信息维护以及 LSDB 数据维护等各方面处理流程的有序进行。GR 状态机作为 GRRestarter 模块运转的核心，是 GRRestarter 模块设计实现的关键所在。根据运行 OSPFv3 协议的路由器设备重启过程中的处理特点，GR 状态机的设计如图 5.1 所示。

我们在 GR 重启流程中共定义了 9 个状态，分阶段依次进行 GR 重启流程处理。

Normal：OSPFv3 进程平时正常运转所处的状态。

Prepare：OSPFv3 进程进入 GR 重启流程前的准备阶段，主要对进入 GR 重启流程的各项条件进行检测以及完成相应的准备工作。

GRDoing：该状态表明 OSPFv3 进程正处于 GR 重启过程中，主要包括邻居信息和 LSDB 数据的恢复和同步。

Generate：LSA 数据重新产生状态。在完成邻居信息的恢复和 LSDB 数据的同步之后，进行 GR 重启的 OSPFv3 进程需要根据最新的链路状态重新生成各类 LSA。

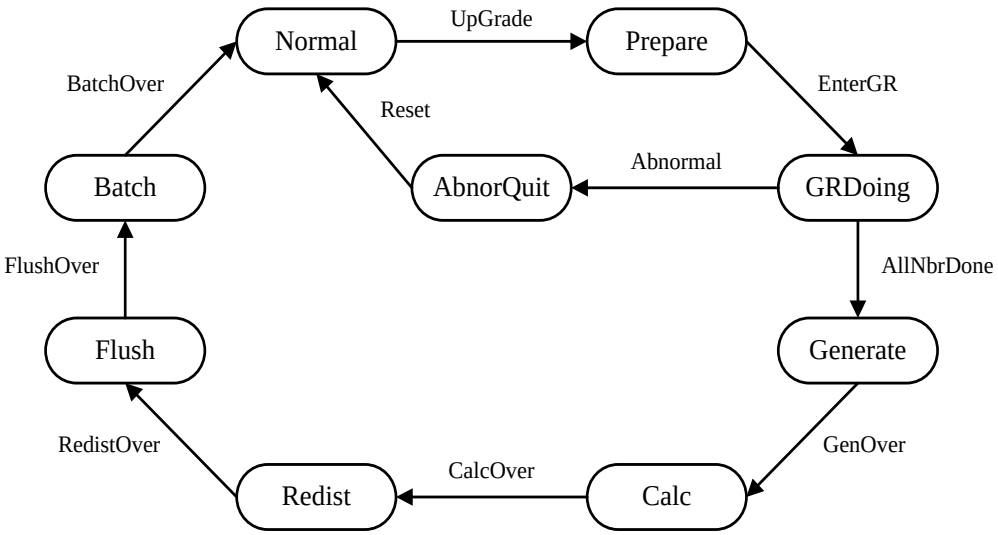

图 5.1　GR 状态机

Calc：路由计算状态。OSPFv3 进程根据新生成的各类 LSA 重新进行路由计算来刷新路由表项。

Redist：路由引入状态。OSPFv3 进程对外部路由信息重新进行引入和发布处理。

Flush：老化状态。OSPFv3 进程对 GR 重启过程中收到的自己产生的并打上老化标记的各类 LSA 进行老化处理。

Batch：批备状态。GR 重启结束后，新主进程需要向备进程触发一次数据批量备份。

AbnorQuit：异常退出状态。进入该状态表明 GR 重启过程中出现了异常情况，导致 OSPFv3 进程强制退出 GR 重启流程。

由于 GR 重启过程中以进程为单位进行处理，因此在 OSPFv3 进程数据结构下需维护相应字段来记录进程的 GR 信息。

```
typedef struct tagPROC
{
    ……
    unsigned char    ucGRRestarter;      /* 标志是否使能 GRRestarter 能力 */
    unsigned short   usGRInterval;       /* 记录 GRRestarter 端的 GR 周期时间 */
    unsigned char    ucGRHelper;         /* 标志是否使能 GRHelper 能力 */
    GR_STATE_E       enGrState;          /* 记录进程当前所处的 GR 状态 */
    ……
} PROC_S;
```

其中，enGrState 的取值范围定义如下：

```
typedef enum tagGR_STATE
{
    GR_STATE_NORMAL_E = 0，
    GR_STATE_PREPARE_E，
```

```
    GR_STATE_GRDOING_E，
    GR_STATE_GENERATE_E，
    GR_STATE_CALC_E，
    GR_STATE_REDIST_E，
    GR_STATE_FLUSH_E，
    GR_STATE_BATCH_E，
    GR_STATE_ABRQUIT_E，
} GR_STATE_E；
```

相应的，我们还定义了 10 种事件来触发 GR 状态机的运转。

UpGrade：升级事件，表明收到了 HA 模块的升级消息，备进程将升级成为主进程。

EnterGR：进入 GR 重启流程事件，表明当前环境满足 GR 重启条件，OSPFv3 进程即将进入 GR 重启流程。

AllNbrDone：邻居关系重建完成事件，表明进行 GR 重启的路由器已经和周边路由器重建了邻居关系，并完成了 LSDB 数据的同步。

GenOver：LSA 重新产生结束事件，表明 OSPFv3 进程在 GR 重启后根据最新的链路状态重新产生了相应的 LSA。

CalcOver：路由计算结束事件，表明 OSPFv3 进程根据新生成的各类 LSA 重新进行了路由计算，刷新了路由表项。

RedistOver：路由引入结束事件，表明 OSPFv3 进程完成了对外部路由信息的重新引入和发布处理。

FlushOver：LSA 老化结束事件，表明 OSPFv3 进程完成了对打上相应标记的 LSA 的老化处理。

BatchOver：批量备份结束事件，表明 OSPFv3 新主进程对备进程进行的数据批量备份结束。

Abnormal：异常事件，表明 GR 重启过程中发生了异常情况，如网络拓扑变化、接口状态和邻居状态变化等。

Reset：进程重启事件。异常退出 GR 重启流程后，需对进程进行 Reset 处理。

GR 状态机中各事件的触发标志定义如下：

```
typedef enum tagGR_EVENT
{
    GR_ EVENT _UPGRADE_E = 0，
    GR_ EVENT _ENTERGR_E，
    GR_ EVENT _ALLNBRDONE_E，
    GR_ EVENT _GENOVER_E，
    GR_ EVENT _CALCOVER_E，
```

GR_ EVENT _REDISTOVER_E，

GR_ EVENT _FLUSHOVER_E，

GR_ EVENT _BATCHOVER_E，

GR_ EVENT _ABNORMAL_E，

GR_ EVENT _RESET_E，

} GR_ EVENT _E；

GR 状态机的处理即根据相应的状态机触发事件推动 GR 状态的变迁，进而推动状态机的运转来完成 GR 重启流程处理。GR 状态机在系统控制线程进行运转，由系统控制线程负责 GR 状态的设置和变迁。收到状态机触发事件时，系统控制线程通过调度其它各线程来完成对相应事件的处理。

5.3 进入 GR

5.3.1 进入 GR 的场景

目前，本课题的设计实现中主要存在如下几种进入 GR 重启流程的场景：

场景一：人为地进行 OSPFv3 进程重启，如用户通过输入 Reset OSPFv3 Process 命令来进行 OSPFv3 进程重启。

场景二：人为地进行控制层面的主备倒换，包括进程级主备倒换和板级主备倒换。

场景三：主控制板异常重启，包括进程异常重启和单板异常重启。

尽管上述三种场景的触发原因不同，但最终都会导致 OSPFv3 备进程升级为主进程，如图 5.2 所示。

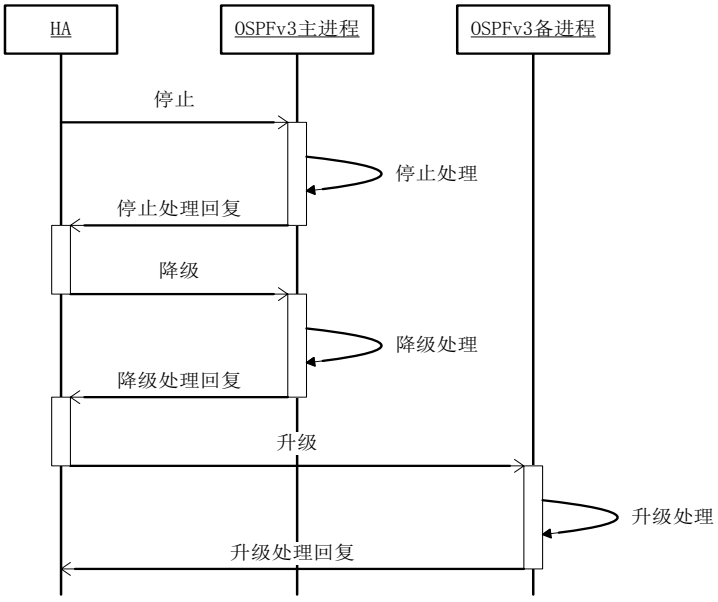

图 5.2　OSPFv3 进程升降级处理

当进行 OSPFv3 进程重启、主备倒换操作或主控制板异常重启时，都会触发 OSPFv3 进程的升降级操作。原 OSPFv3 主进程收到 HA 模块发送的停止消息后，调度各线程停止当前的

协议运转机制，处理完成后回复 HA 模块停止结束消息。随后，HA 模块又会向 OSPFv3 主进程发送降级消息，OSPFv3 主进程调度各线程进行运行数据清除，处理完成后向 HA 模块回复降级结束消息，原 OSPFv3 主进程降级为备进程。与此同时，原 OSPFv3 备进程会收到 HA 模块发送的升级消息进行升级处理，主要包括启动相应的定时器和恢复协议运行机制，处理完成后向 HA 模块回复升级结束消息，原 OSPFv3 备进程升级为主进程。

5.3.2 进入 GR 前的处理

当收到 UpGrade 升级事件后，触发 GR 状态机的运转，进程下的 GR 状态从 Normal 状态变迁到 Prepare 状态。进入 Prepare 状态后，系统控制线程首先会调用 IsEnterGR 接口进行相关条件的判定。进入 GR 重启流程需要满足一定条件，主要包括进程处于激活状态、进程下使能了 GRRestarter 能力、存在激活的非 Loopback 接口、接口下存在 Full 状态的邻居以及具有主备环境，如图 5.3 所示。

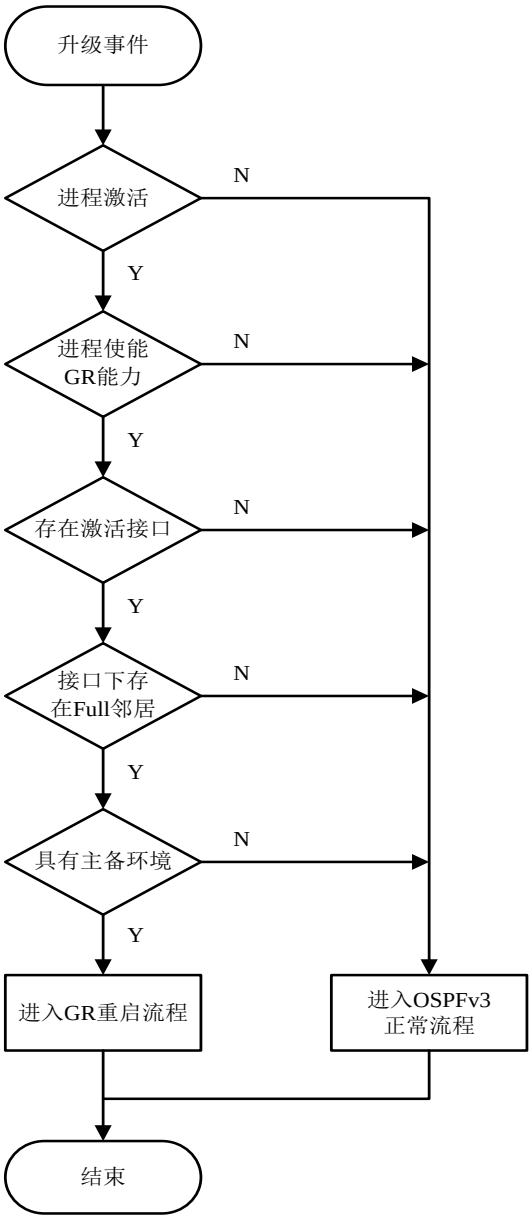

图 5.3　进入 GR 条件判定

若上述各项条件都满足，则成为 GRRestarter，进入 GR 重启流程处理；否则，进入 OSPFv3 正常流程进行邻居关系的重建和 LSDB 数据的同步。

IsEnterGR 函数接口实现如下：

```
void  IsEnterGR（）
{
    if（（进程处于 Active 状态）&&（进程下使能了 GRRestarter 能力）&&（存在激活
        的非 Loopback 接口）&&（存在 Full 邻居）&&（具有主备环境））
    {
        进入 GR 重启流程处理；
    }
    else
    {
        进入 OSPFv3 正常流程处理；
    }
}
```

此外，在 Prepare 状态下还需设置相应的 GR 重启原因，并且保存到非易失性的储存介质中。目前，GR 重启原因可能为软件重启、软件升级/重载、主备倒换和异常重启中的一种。

5.4 GR 过程处理

若满足 GR 重启的各项条件，则触发 EnterGR 状态机事件，进入 GR 重启流程处理[52]，并将进程下的 GR 状态设置为 GRDoing 状态。

5.4.1 定时器维护

系统控制线程收到 EnterGR 触发事件后，向 LSDB 和状态机线程发送控制消息，通知其进入 GR 重启流程处理。LSDB 和状态机线程收到该控制消息后，启动相应的定时器来控制 GR 重启流程的处理，如图 5.4 所示。本课题中，GRRestarter 端在 GR 重启过程中需要维护的定时器主要包括全局抑制 Hello 报文定时器、Grace LSA 发送定时器、GR 周期定时器和 WaitNbr 定时器。

（1）全局抑制 Hello 报文定时器

非计划性 GR 机制实现的一个关键之处在于需要保证 Grace LSA 的发送先于 1-Way Hello 报文的发送。OSPFv3 原备进程升级为主进程后，会向外发送 1-Way Hello 报文来重新发现和建立邻居关系，而周边路由器收到该 1-Way Hello 报文后，会断开与该重启路由器的邻居关系，并将其从相应的数据转发链路上删除，这样就会导致数据转发流量的中断。因此，需要启动全局抑制 Hello 报文定时器，来抑制 1-Way Hello 报文的发送，以腾出时间进行 Grace LSA 的发送。等全局抑制 Hello 报文定时器超时后，再启动接口下的 Hello 报文定时器来周期性地向外发送 Hello 报文以发现和建立邻居关系。此时，周边路由器收到 1-Way Hello 报文后，由于

已经通过 Grace LSA 进行了 GR 重启协商，不会断开与重启路由器的邻居关系。这样就通过推迟 1-Way Hello 报文的发送时间，来保证 Grace LSA 的发送先于 1-Way Hello 报文，从而保证了数据转发流量的不中断。本课题中，全局抑制 Hello 报文定时器的时间间隔为 5s。

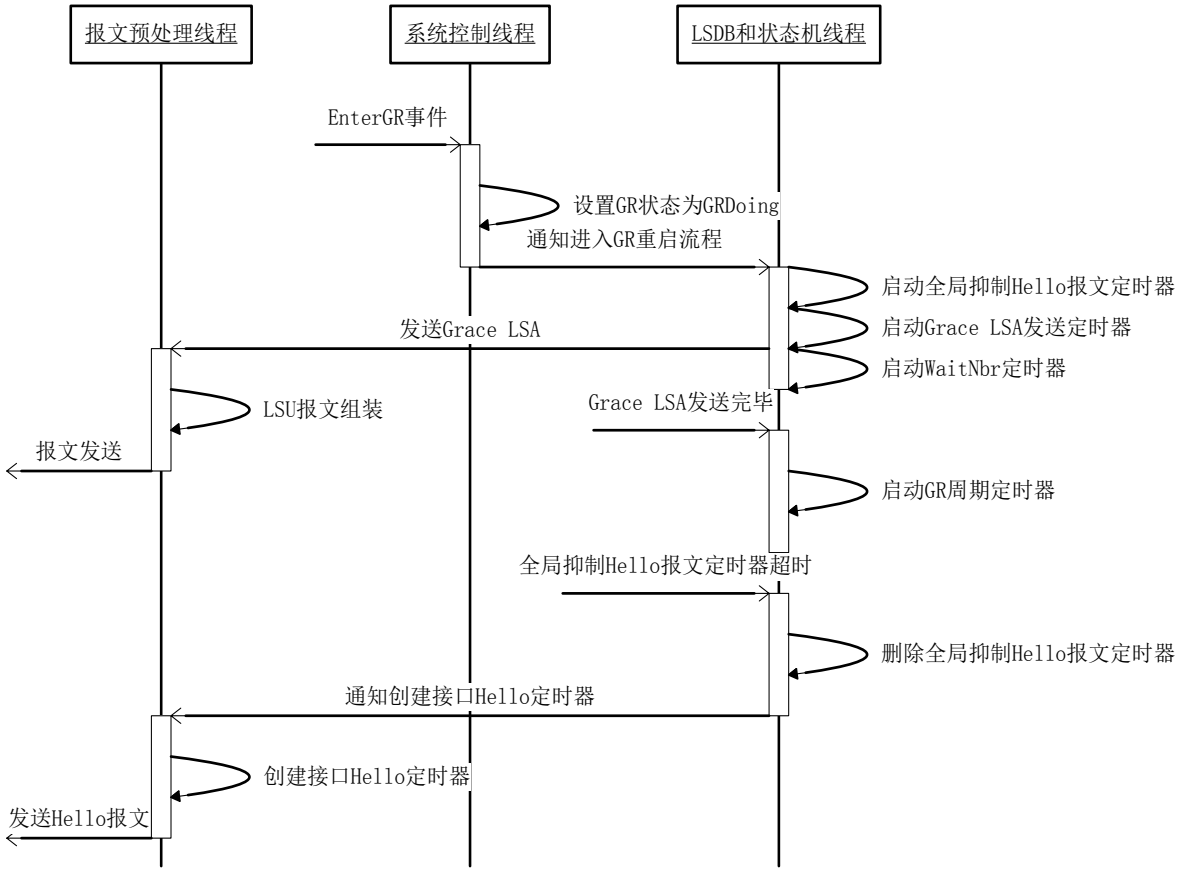

图 5.4　GR 过程中的定时器维护

（2）Grace LSA 发送定时器

GRRestarter 通过发送 Grace LSA 来与周边路由器进行 GR 重启协商，其中携带了 GR 重启的相关信息，包括 GR 周期时间间隔和 GR 重启原因。Grace LSA 通过 LSU 报文携带并通过所有接口进行发送。

为了保证 Grace LSA 能被正确地发送到周边路由器，具体实现时通过定时器来对 Grace LSA 进行多次发送。Grace LSA 发送定时器的时间间隔设置为 1s，定时器超时后进行发送，默认发送 5 次。Grace LSA 发送定时器超时回调函数如下所示。

```
void GraceLSASendTimerCallback（）
{
    SendGraceLSA（）;                /* 定时器超时，进行 Grace LSA 发送 */
    if（uiSendCnt > 0）              /* 默认发送 5 次，没发送完继续发送 */
    {
        uiSendCnt --;
```

```
    }
    else
    {
        StopGraceLSASendTimer（）;     /* 5 次发送完，停止 Grace LSA 发送定时器 */
        StartGRIntervalTimer（）;      /* 启动 GR 周期定时器 */
    }
}
```

（3）GR 周期定时器

在 Grace LSA 发送完成后，进程下启动 GR 周期定时器。GR 重启过程中邻居信息和 LSDB 数据的恢复和同步需要在 GR 周期时间间隔内完成，否则将导致 GR 周期定时器超时，强制退出 GR 重启流程。本课题中 GR 周期定时器的时间间隔默认为 120s，也可由用户进行配置，其值在进程数据结构下进行维护。

（4）WaitNbr 定时器

WaitNbr 定时器用于在一定时间内尽可能地发现邻居。在不超过 GR 周期的前提下，WaitNbr 定时器的时间间隔通常设置为 DeadInterval（默认为 40s），即认为在该时间段内至少能收到邻居的 Hello 报文而发现该邻居。另一方面，WaitNbr 定时器也能保证尽可能早地退出 GR 重启流程，避免没有邻居时长时间等待 GR 周期定时器的超时才退出 GR 重启流程。WaitNbr 定时器超时后检查对应进程下的 GRHelper 计数，若该计数不为 0，表明还未完成所有邻居关系的重建，则继续进行 GR 重启流程处理；若该计数为 0，并且存在 Full 状态的邻居，则可认为邻居关系全部重建完成，正常退出 GR 重启流程；若该计数为 0，但不存在 Full 状态的邻居，则异常退出 GR 重启流程。WaitNbr 定时器超时回调函数如下所示。

```
void WaitNbrTimerCallback（）
{
    if（GRHelper 计数 > 0）        /* GRHelper 计数不为 0，则返回不作处理，
    {                                 继续进行邻居关系重建处理 */
        return;
     }
    else                           /* GRHelper 计数为 0，表明完成邻居关系重建 */
    {
        if（0 ！= NbrFullCnt）
        {
            通知系统控制线程完成邻居关系的重建;
        }
        else
        {
```

```
                异常退出 GR 重启流程;
            }
        }
    }
```

5.4.2 邻居信息的收集

OSPFv3 进程在 GR 重启过程中需要收集重启前的邻居信息，统计重启前与自身具有邻居关系的路由器数量，也即 GRHelper 的个数。同时，也通过该计数来判定是否已经完成所有邻居关系的重建。本课题中，邻居信息收集的方案主要有如下三种。

方案一：通过邻居发送的 Hello 报文进行 GRHelper 个数的统计。GR 重启过程中，每收到一个不同邻居发送的 Hello 报文，就将 GRHelper 计数加 1。邻居关系重建后，就将 GRHelper 计数减 1，当 GRHelper 计数减到 0 时，即表明所有邻居关系都已重建完毕。

方案二：通过已接收到的 Hello 报文检查邻居是否已经发现全。GR 重启过程中，对不同邻居发送的 Hello 报文中所通告的邻居字段进行检查，判定是否收集了全部的邻居信息。

方案三：OSPFv3 主进程平时将邻居数量备份到备进程，进行 GR 重启时根据备份信息来进行邻居关系的重建。

对于方案一，实现上较为简单，通过维护相应的计数来统计邻居数量。同时，结合 WaitNbr 定时器和该邻居计数来判定邻居关系是否重建完成较为可靠，可以避免长时间等待 GR 的结束，缩短了 GR 周期。对于方案二，由于重启后无法知道具体的邻居个数，需要按接口进行细化检查，实现上较为复杂。同时，若收到的不同邻居发送的 Hello 报文中所通告的邻居信息不一致，无法判定哪个更可靠。对于方案三，重启前后邻居数量可能发生变化，备份信息不一定可靠，并且当存在大量邻居时进行数据备份的开销较大。

综上所述，本课题在具体实现时采用方案一来进行邻居信息的收集。

此外，GR 重启过程中，GRRestarter 从收到的第一个 Hello 报文中提取链路上的 DR 和 BDR 信息，而不是自己运行 DR/BDR 选举算法来进行 DR 和 BDR 的选举，以此保证了链路上 DR 和 BDR 信息的稳定，避免网络中产生振荡。若 GRRestarter 通过邻居发送的 Hello 报文发现自己在重启前是对应链路上的 DR 或 BDR，则会继续维持自己的 DR 或 BDR 身份，在向邻居发送的 Hello 报文中宣告自己为 DR 或 BDR。

5.4.3 LSDB 数据维护

当 GRRestarter 和周边路由器重建邻居关系到 2-Way 状态后，需要进行 LSDB 数据的同步，进一步形成邻接关系。由于 GRRestarter 在重启前与邻居已经形成了 LSDB 数据的同步，所以重启后只需从邻居获取完整的 LSDB 数据即可恢复出本地的 LSDB 数据库信息。GRRestarter 通过 LSR 报文向邻居请求完整的 LSDB 数据，邻居则通过 LSU 报文将各类 LSA 发送给 GRRestarter。GRRestarter 收到 LSU 报文后，即可根据各类 LSA 重构本地 LSDB 数据库信息。

为了保证网络拓扑和链路状态的稳定，GR 重启过程中对本地 LSDB 数据的维护需要作如下特殊处理：

（1）不产生各类 LSA，对邻居发送的 LSA 仅作简单的接收和存储操作，网络中的其它路由器均使用 GRRestarter 重启前产生的各类 LSA 来进行路由计算。

（2）在从邻居同步 LSA 的过程中，若 GRRestarter 收到了重启前自己产生的 LSA，不进行修改或刷新处理，而是当作普通的 LSA 进行接收，同时在这些 LSA 上打上老化标记并添加到 LSDB 中，等退出 GR 重启流程后统一进行老化和刷新处理。

（3）GRRestarter 忽略收到的 Grace LSA 报文，不作处理直接丢弃。

5.4.4 路由信息维护

GR 重启过程中保证数据转发流量不中断的一个关键之处在于重启路由器需要保证路由转发表的稳定。因此，GR 重启过程中需作如下特殊处理：

（1）原 OSPFv3 主进程在降级过程中清除运行数据时保留 Intra AS 和 Inter AS 路由信息。具体实现时可通过判定进程的 GR 状态信息，若进程处于 GRDoing 状态，则不删除 Intra AS 和 Inter AS 路由数据，同时也不会更新相应的路由表项。

（2）GRRestarter 从邻居同步各类 LSA 时，仅作接收和存储操作，并不触发路由计算，也不下发路由信息来刷新路由表项。

（3）当执行外部路由引入和路由聚合等操作时，判定若当前进程处于 GR 过程中，则不作响应，保持路由表项稳定。

5.5 退出 GR

5.5.1 正常退出 GR

当 GRRestarter 在 GR 周期时间间隔内完成了对所有邻居关系的重建和 LSDB 数据的同步，就会触发 AllNbrDone 状态机事件，正常退出 GR 重启流程。

目前实现中，AllNbrDone 状态机事件的触发时机共有两个，如图 5.5 和图 5.6 所示。

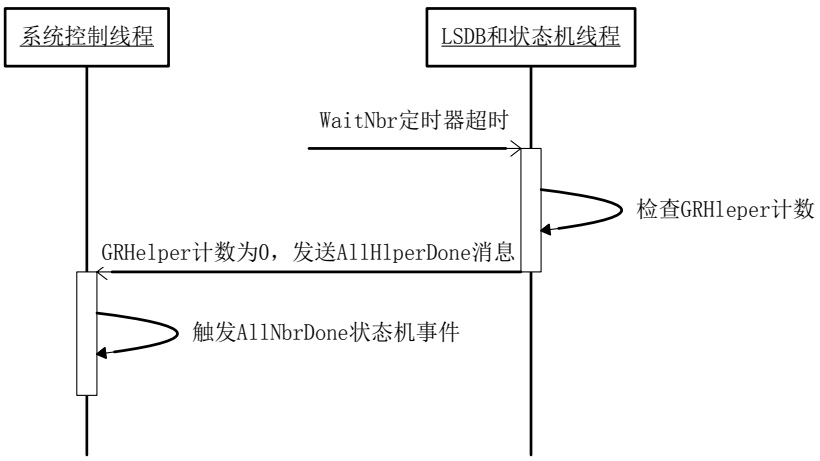

图 5.5　AllNbrDone 事件触发时机 1

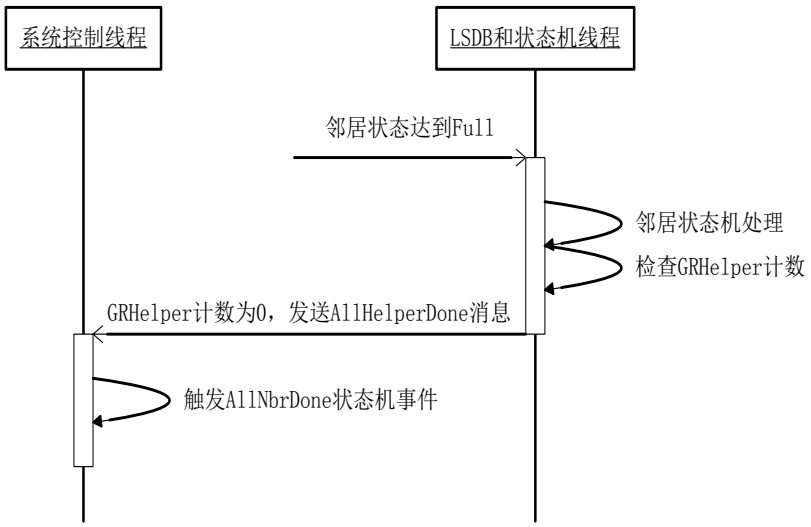

图 5.6　AllNbrDone 事件触发时机 2

（1）WaitNbr 定时器超时后，LSDB 和状态机线程检查 GRHelper 计数，若该计数为 0 并且 Full 邻居个数不为 0，则 LSDB 和状态机线程向系统控制线程发送 AllHelperDone 控制消息，触发 AllNbrDone 状态机事件。

（2）在 WaitNbr 定时器时间内未能完成邻居关系重建的情况下，当 GRRestarter 与周边路由器达到 Full 邻居状态后，在邻居状态机的处理中检查 GRHelper 计数，若该计数为 0，则 LSDB 和状态机线程向系统控制线程发送 AllHelperDone 消息，触发 AllNbrDone 状态机事件。

系统控制线程收到 LSDB 和状态机线程发送的 AllHelperDone 消息后，触发 AllNbrDone 状态机事件，推动 GR 状态机的运转，将进程的 GR 状态从 GRDoing 状态变迁到 Generate 状态，进入正常退出 GR 重启流程处理，如图 5.7 所示。

5.5.1.1 重产生阶段

LSDB 和状态机线程收到系统控制线程发送的进入 Generate 状态的控制消息后，进行 LSA 重产生阶段的处理。

首先，对 Grace LSA 进行老化处理，即产生并发送 LS Age 为 3600s 的 Grace LSA，以此来通告邻居设备自身完成了邻居关系的重建和 LSDB 数据的同步，正常退出 GR 重启流程。该 Grace LSA 通过 LSU 报文携带并通过所有接口进行发送。

其次，在退出 GR 重启流程后，GRRestarter 需要根据最新的链路状态重新产生 Router-LSA 和 Intra-Area-Prefix-LSA。若 GRRestarter 是对应链路上的 DR，则还需要对 Network-LSA 进行重产生处理。各类 LSA 的重产生处理即递增相应的 LS 序列号并生成新的 LSA 来替换 LSDB 中旧的 LSA，同时通过 LSU 报文将新生成的 LSA 洪泛给邻居来更新其 LSDB。

完成对各类 LSA 的重产生处理后，LSDB 和状态机线程向系统控制线程发送重产生阶段结束消息。

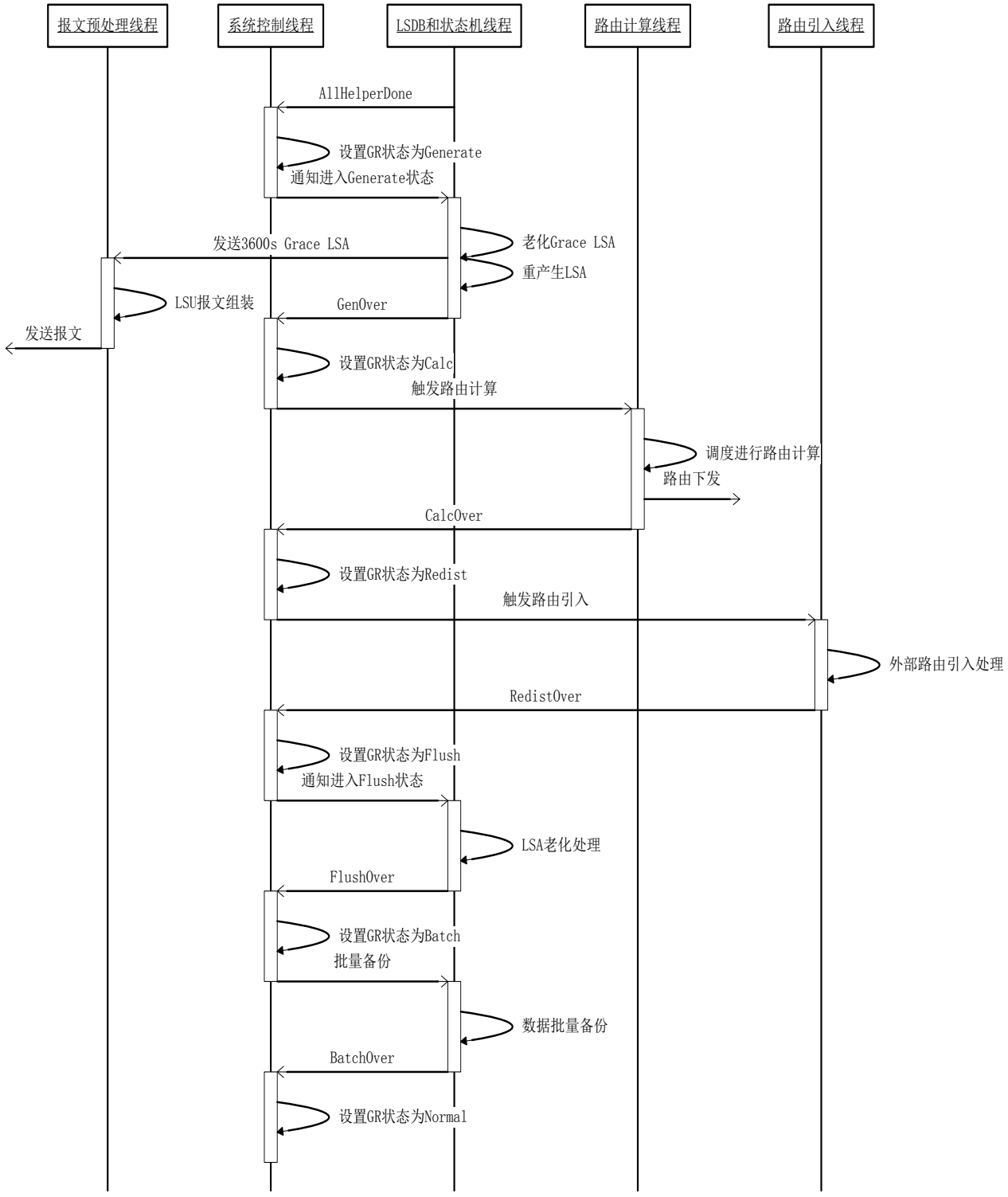

图 5.7　正常退出 GR 重启流程处理

5.5.1.2 路由计算阶段

系统控制线程收到 LSDB 和状态机线程发送的控制消息后，触发 GenOver 状态机事件，推动 GR 状态机的运转，进程下的 GR 状态变迁为 Calc 状态。同时，向路由计算线程发送控制消息来调度进行路由计算。

当前实现中，我们将 OSPFv3 协议路由的计算过程划分为 5 个阶段依次进行：根据 Router-LSA 和 Network-LSA 进行区域内拓扑计算、根据 Intra-Area-Prefix-LSA 进行区域内 IntraRtr 路由计算、根据 Inter-Area-Router-LSA 进行 ASBR 路由计算、根据 Inter-Area-Prefix-LSA 进行区域间路由计算、根据 AS-External-LSA 进行外部路由计算，如图 5.8 所示。

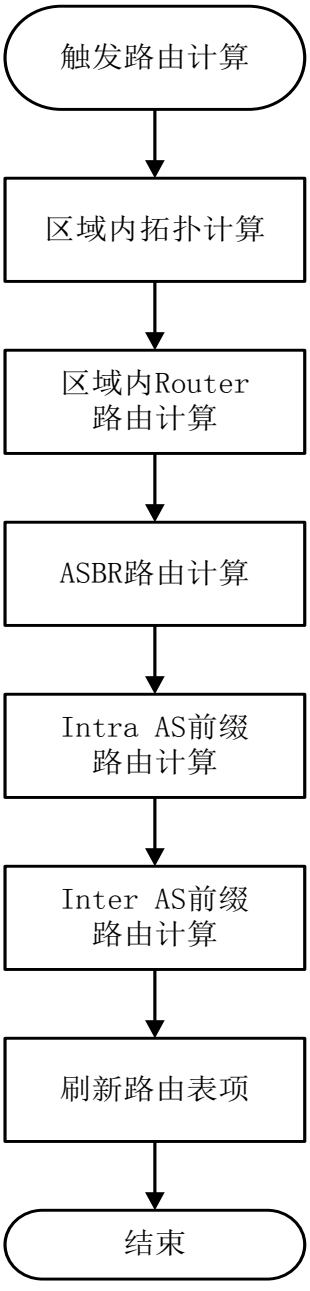

图 5.8　路由计算过程

路由计算线程收到调度消息后，调用 RT_CalcProc 接口根据新生成的各类 LSA 来进行路由计算。

```
void RT_CalcProc（PROC_S    *pstProc）
{
        ……
```

```
    ucCalcPhase = Calc_GetPhase（pstProc）;
    switch（ucCalcPhase）
    {
        case   CALC_TOPO：
        {
            进行区域内拓扑计算；
            ucCalcPhase = CALC_INTRARTR；
        }
        case   CALC_INTRARTR：
        {
            进行区域内 IntraRtr 路由计算；
            ucCalcPhase = CALC_ASBRRT；
        }
        case   CALC_ ASBRRT：
        {
            进行 ASBR 路由计算；
            ucCalcPhase = CALC_INTRAAS；
        }
        case   CALC_ INTRAAS：
        {
            进行区域内和区域间前缀路由计算；
            ucCalcPhase = CALC_INTERAS；
        }
        case   CALC_ INTERAS：
        {
            进行外部路由计算；
            ucCalcPhase = CALC_COMPLETE；
        }
        case   CALC_ COMPLETE：
        {
            回复系统控制线程路由计算结束消息；
        }
    }
    ……
}
```

经过计算生成最新的路由信息后，需要将其下发到路由表中，刷新相应的路由表项。同时，在完成全部路由信息的计算后，路由计算线程向系统控制线程发送路由计算完成的控制消息。

5.5.1.3 路由引入阶段

系统控制线程收到路由计算线程发送的控制消息后，触发 CalcOver 状态机事件，将进程的 GR 状态设置为 Redist 状态，并调度路由引入线程进行外部路由信息的引入处理。路由管理模块在 OSPFv3 协议 GR 重启结束并下发最新路由信息刷新路由表项后，会触发引入的外部路由信息的重新上报，如图 5.9 所示。

路由管理首先向 OSPFv3 协议模块发送引入路由平滑开始消息。OSPFv3 协议模块收到后进行平滑开始处理，主要包括增加引入的外部路由信息的版本号。随后，路由管理向 OSPFv3 协议模块按前缀上报引入的外部路由信息，OSPFv3 协议模块对上报路由进行引入处理，主要包括策略过滤，并将通过策略的外部路由信息用 AS-External-LSA 在自治系统内进行洪泛，通告给其它路由器。在完成对所有外部路由信息的上报后，路由管理向 OSPFv3 协议模块发送平滑结束消息。OSPFv3 协议模块收到后进行平滑结束处理，主要对版本号较旧的外部路由信息进行老化处理。

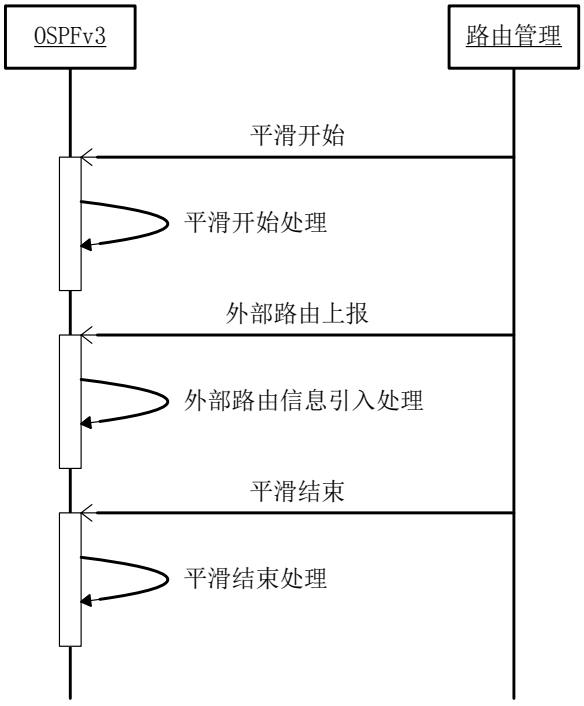

图 5.9　外部路由引入处理

完成路由引入阶段的处理后，路由引入线程向系统控制线程回复相应的控制消息。

5.5.1.4 LSA 老化阶段

系统控制线程收到路由引入结束的控制消息后，触发 RedistOver 状态机事件，设置进程的 GR 状态为 Flush 状态，并向 LSDB 和状态机线程发送控制消息，通知其进行 LSA 老化阶

段的处理。

LSA 老化阶段主要对 GR 重启过程中，GRRestarter 从邻居收到的重启前自己产生的 LSA 进行老化处理，这些需要进行老化处理的 LSA 都打上了相应的老化标记。因此，具体处理时只需遍历进程、区域和接口下的 LSDB，将打上老化标记的 LSA 的 LS Age 设置为 3600s 并通过 LSU 报文在相应接口上进行洪泛，通知邻居删除相应的 LSA。

完成 LSA 老化阶段的处理后，LSDB 和状态机线程向系统控制线程回复处理结束控制消息。

5.5.1.5 批量备份阶段

系统控制线程收到老化结束消息后，触发 FlushOver 状态机事件，GR 状态进入 Batch 状态，同时向 LSDB 和状态机线程发送批量备份消息。LSDB 和状态机线程收到后，收集相应的批量备份数据，主要为进程、区域和接口下存储的各类 LSA 的 LSID 数据，然后调用 HA 模块的批量备份接口，将备份数据从 OSPFv3 主进程备份到备进程。批量备份数据的发送使用 HA 模块的批量备份数据通道。

备份数据发送完成后，LSDB 和状态机线程向系统控制线程回复批量备份结束消息，系统控制线程收到后将进程的 GR 状态设置为 Normal 状态。至此，OSPFv3 进程完成对 GR 重启流程的处理，进入 OSPFv3 正常流程。

5.5.2 异常退出 GR

5.5.2.1 异常退出场景

当 OSPFv3 进程处于 GRDoing 状态，即进行邻居关系的重建和 LSDB 数据的同步过程中，若出现异常情况，就会触发 Abrnormal 状态机事件，异常退出 GR 重启流程。目前实现中，Abrnormal 事件的触发场景主要有如下几种：

场景一：GRRestarter 在 GR 周期时间内没有完成邻居关系的重建和 LSDB 数据的同步处理，导致 GR 周期定时器超时；

场景二：GRRestarter 在 GR 重启过程中收到了邻居发送的 1-Way Hello 报文；

场景三：对应链路上的 DR/BDR 发生变化，如收到同一邻居发送的前后两个 Hello 报文中 DR/BDR 字段不一致，或者收到的不同邻居发送的 Hello 报文中 DR/BDR 字段不一致。

场景四：GRRestarter 在 GR 重启过程中接口状态变为 Down 状态，说明链路拓扑结构发生了变化。

5.5.2.2 异常退出处理

当出现上述异常情况时，触发 Abnormal 状态机事件，系统控制线程推动 GR 状态机进入 AbnorQuit 状态，同时向 LSDB 和状态机线程发送控制消息，通知进入异常退出 GR 流程处理，如图 5.10 所示。LSDB 和状态机线程收到控制消息后，停止进程下 GR 相关的各类定时器，包括 GR 周期定时器、WaitNbr 定时器和 Grace LSA 发送定时器。同时，通过所有接口向外发送 LS Age 为 3600s 的 Grace LSA，通知邻居设备自身在 GR 重启过程中出现了异常情况，

退出 GR 重启流程。

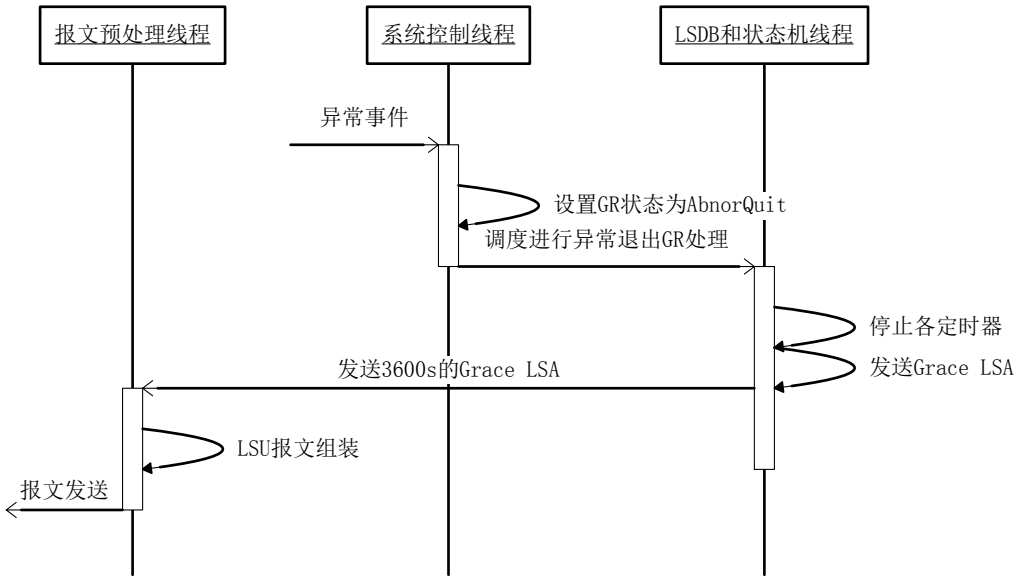

图 5.10　异常退出 GR 流程处理

异常退出 GR 重启流程后，系统控制线程需要借助 Reset 流程对邻居数据、LSDB 数据和路由数据进行后续处理，如图 5.11 所示。

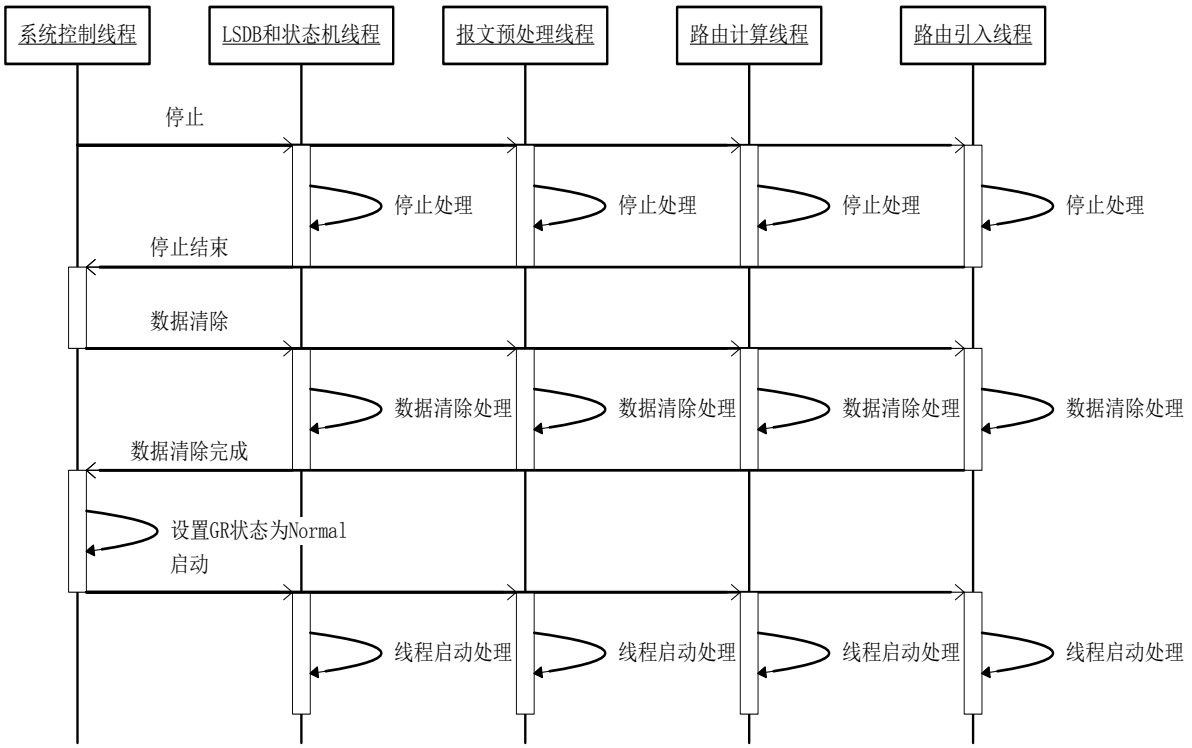

图 5.11　Reset 流程处理

首先，系统控制线程向其余各线程发送停止消息，通知各线程停止协议机制的运转和处理，主要包括停止各线程下的定时器，停止协议状态机的运转等。在收到各线程的停止结束

回复消息后，系统控制线程再向各线程发送数据清除消息，通知各线程进行运行数据的清除，主要包括邻居数据、LSDB 数据和路由数据等的清除处理。在收到各线程的数据清除完成回复消息后，系统控制线程将进程的 GR 状态设置为 Normal 状态，同时向各线程发送启动消息，触发各线程重新启动相应的定时器，恢复协议运行机制，重新进行邻居关系的建立和 LSDB 数据的同步。

至此，OSPFv3 进程完成异常退出 GR 重启流程的处理，进入 OSPFv3 正常流程。

5.6 本章小结

本章在进行软件需求分析的基础上，对 GRRestarter 模块进行了设计实现。文中首先对 GR 状态机进行了设计，介绍了 GR 状态和相应的状态机触发事件，GR 状态机是 GRRestarter 模块的核心所在。随后，文中结合 GR 状态机，详细阐述了 GRRestarter 在 GR 重启过程中的设计实现，具体包括 GR 重启流程的触发场景及进入条件，GRRestarter 在 GR 重启过程中对定时器、邻居信息、LSDB 数据和路由信息的维护和处理，以及正常退出 GR 重启流程的处理和异常退出 GR 重启流程的处理，GR 状态机的运转贯穿整个处理流程。此外，文中还对部分数据结构的设计和函数接口的实现进行了介绍。

在 GRRestarter 模块的设计过程中需要注意以下几点：

（1）对于多个线程共同访问的内存数据，要用锁机制来进行保护，避免出现数据访问的冲突和时序问题。线程在进行数据访问前需先进行加锁操作，获取相应的锁后才能对内存数据进行读写，数据访问结束后再进行解锁操作，以此来保证对内存数据的安全访问和各个流程的有序进行。

（2）正常退出 GR 重启流程后，需对 LSA 的重产生和路由计算进行阶段划分。设计初期，未对两者进行具体划分，LSDB 和状态机线程产生 LSA 后立即调度路由计算线程进行路由计算，可能出现部分路由信息未被更新的情况。后续设计进行了改进，划分出 Generate 状态和 Calc 状态分别进行 LSA 的重产生和路由计算处理，在所有 LSA 数据重新产生结束后统一触发进行路由计算，从而保证了路由信息的正常更新。

第6章 GRHelper 模块设计

GRHelper 不需要具有主备环境，集中式设备和分布式设备都可以成为 GRHelper，协助 GRRestarter 完成 GR 重启流程处理。

6.1 需求点分析

在对 GRHelper 模块进行软件设计时，需要支持如下需求点：

（1）只有满足特定条件，路由器才能成为 GRHelper，进入 GR 重启流程。

（2）在 GR 过程中，路由器能够维持与重启路由器邻居关系的稳定。

（3）在 GR 过程中，路由器能够维持路由信息和转发表的稳定。

（4）收到请求后，路由器能够将完整的 LSDB 数据同步给重启路由器，协助其进行数据恢复。

（5）正常退出 GR 流程后，路由器能够对 LSDB 数据和路由表项进行更新。

（6）GR 过程中出现异常情况时，路由器能够完成对邻居信息、LSDB 数据和路由信息的后续处理。

6.2 进入 GR

当前实现中，成为GRHelper进入GR流程的时机只有一个，即收到邻居发送的Grace LSA。收到 Grace LSA 时，系统控制线程调用 GR_GraceLSAProc 接口进行处理。

```
void GR_GraceLSAProc（）
{
    BOOL_T    bIsProc = BOOL_FALSE；

    bIsProc = GR_IsProcGraceLSA（）；
    if（BOOL_TRUE == bIsProc）
    {
        GR_EnterGRCheck（）；
    }
}
```

首先，对当前 OSPFv3 进程所处的状态进行检查，若当前进程自身处于 GR 重启过程中，则忽略收到的 Grace LSA，不进行处理。否则，对进入 GRHelper 模式的条件进行进一步的判定，通过调用 GR_EnterGRCheck 接口进行实现，具体流程如图 6.1 所示。

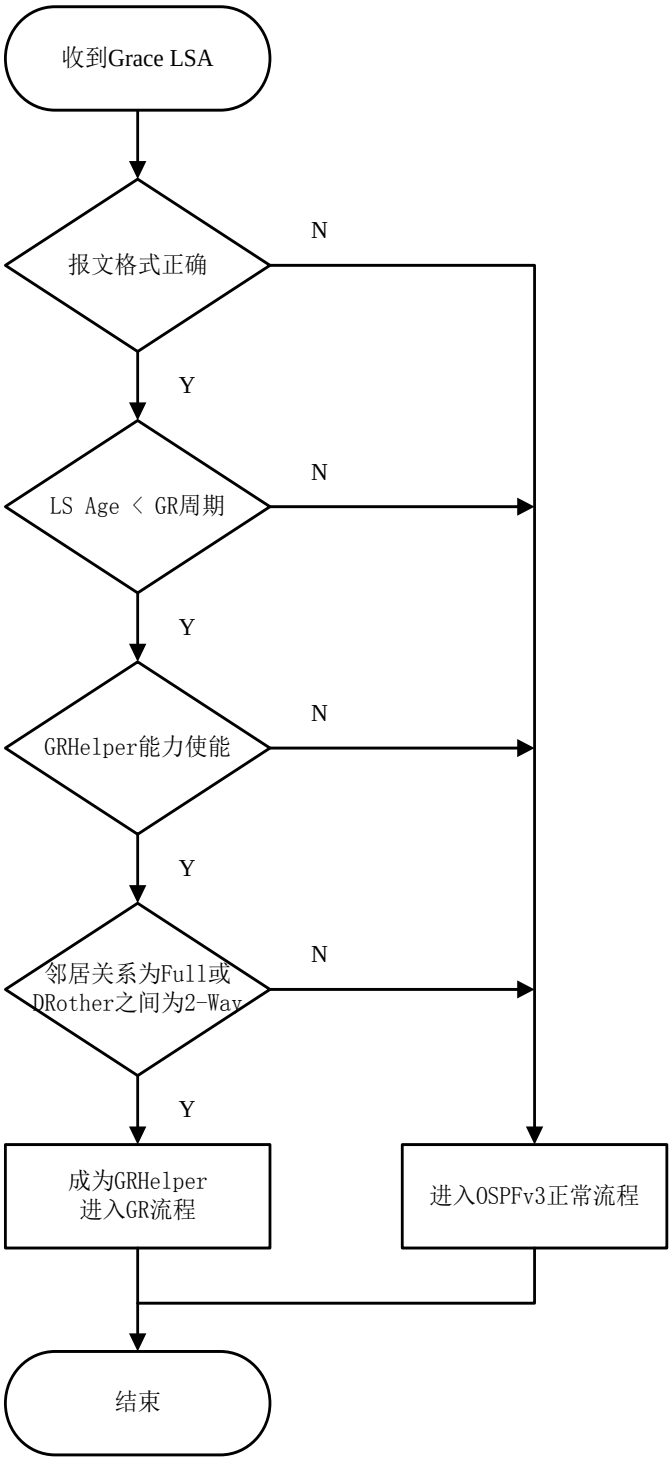

图 6.1　进入 GRHelper 模式判定

对应函数接口实现如下：

```
void GR_EnterGRCheck（）
{
    ……
    GR_GetGraceLSAData（）；    /* 提取 GR 相关字段 */
    if（报文格式错误）
```

```
{
    return;
}
if（LSAge > GR 周期时间间隔）
{
    发送 1-Way Hello 报文;
    return;
}
if（进程下 GRHelper 能力未使能）
{
    发送 1-Way Hello 报文;
    return;
}
GR_NbrGet（）;    /* 获取与 Grace LSA 发送端的邻居关系*/
if（邻居关系不满足条件）
{
    发送 1-Way Hello 报文;
    return;
}
……
}
```

如上所示，首先从收到的 Grace LSA 中提取出 GR 相关信息，包括 GR 周期时间和 GR 重启原因，检查报文格式是否正确，是否存在异常报文。其次，检查 LS Age 是否大于 GR 周期时间。然后，对本地策略进行检测，即进程下是否使能了 GRHelper 能力。最后，获取与该 Grace LSA 发送端的邻居关系，检查邻居状态是否为 Full 状态，或者 DRother 之间的邻居状态是否为 2-Way 状态。

若各项条件都满足，则本端设备成为 GRHelper，进入 GR 重启流程。否则，向对端设备发送 1-Way Hello 报文，进入 OSPFv3 正常流程进行邻居关系的重建。

6.3 GR 过程处理

进入 GR 重启流程后，系统控制线程会向 LSDB 和状态机线程发送相应的控制消息，通知其进入 GRHelper 模式。LSDB 和状态机线程收到后，进入 GR 重启流程处理，启动 GR 周期定时器，设置 GRHelper 模式标志位，并增加进程下的 GRRestarter 计数，如图 6.2 所示。

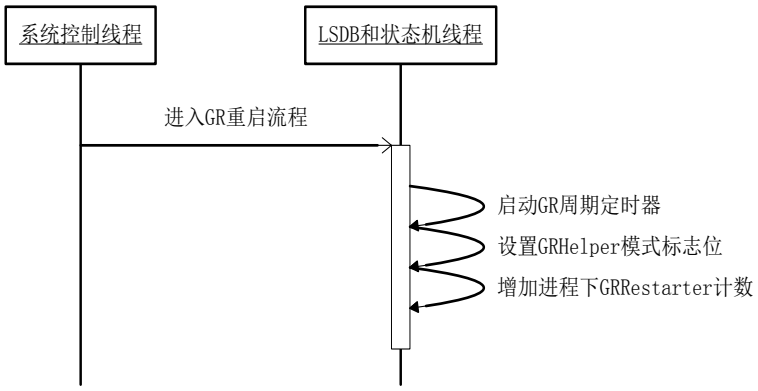

图 6.2　进入 GRHelper 处理

在 GR 周期时间内，GRHelper 需要协助 GRRestarter 完成 GR 重启流程处理，包括邻居信息、LSDB 数据和路由信息的维护和处理。

6.3.1 定时器维护

在 GR 重启过程中，GRHelper 端同样需要维护相应的 GR 周期定时器，GRRestarter 需要在一定时间内完成邻居关系的重建和 LSDB 数据的同步，否则将导致 GRHelper 端的 GR 周期定时器超时，异常退出 GR 重启流程。由于一台路由器可以同时成为多台重启路由器的 GRHelper，因此与 GRRestarter 模块的设计不同，GRHelper 模块中将 GR 相关数据信息在邻居数据结构中进行维护，包括为每个 GRRestarter 邻居单独维护的 GR 周期定时器和 GRHelper 模式标志位。

```
typedef struct tagNBR
{
    ……
    unsigned char     ucGRFlag;
    unsigned long     ulGracePeriodTimerId;
    unsigned short    usGracePeriod;
    ……
} NBR_S;
```

其中，ucGRFlag 为 GRHelper 模式标志位，用于标志当前是否作为该邻居的 GRHelper。ulGracePeriodTimerId 为 GR 周期定时器 ID，usGracePeriod 记录了 GR 周期时间间隔。GRHelper 端的 GR 周期定时器时间间隔从收到的 Grace LSA 中提取，由其中携带的 GR 周期时间减去 LS Age 后得出。

LSDB 和状态机线程通过调用 GR_EnterHelper 接口来对 GR 周期定时器进行维护。

```
void GR_EnterHelper（NBR_S *pstNbr）
{
    ……
```

```
    if（BIT_TEST（pstNbr->ucGRFlag，NBR_IETFGR））
    {
        GR_HandleGRPeriodTimer（pstNbr，TIMER_RESET_E）；
    }
    else
    {
        GR_HandleGRPeriodTimer（pstNbr，TIMER_SET_E）；
        BIT_SET（pstNbr->ucGRFlag，NBR_IETFGR）；
    }
    ……
}
```

如前文所述，为了确保 Grace LSA 能被正确接收，GRRestarter 会对 Grace LSA 进行多次发送。相应地，GRHelper 可能会连续接收到同一个邻居发送的多个 Grace LSA。当首次收到邻居发送的 Grace LSA 时，在该邻居数据结构下打上 NBR_IETFGR 标志，并启动 GR 周期定时器。当收到同一邻居发送的重复 Grace LSA 时，重置相应的 GR 周期定时器即可。可通过邻居数据结构下的 NBR_IETFGR 标志位来判定是否为首次收到该邻居发送的 Grace LSA。

6.3.2 邻居信息维护

GRHelper 在 GR 周期时间间隔内需要维持与 GRRestarter 邻居关系的稳定，因此在 GR 过程中需要对协议报文和邻居状态机作特殊处理，如图 6.3 所示。

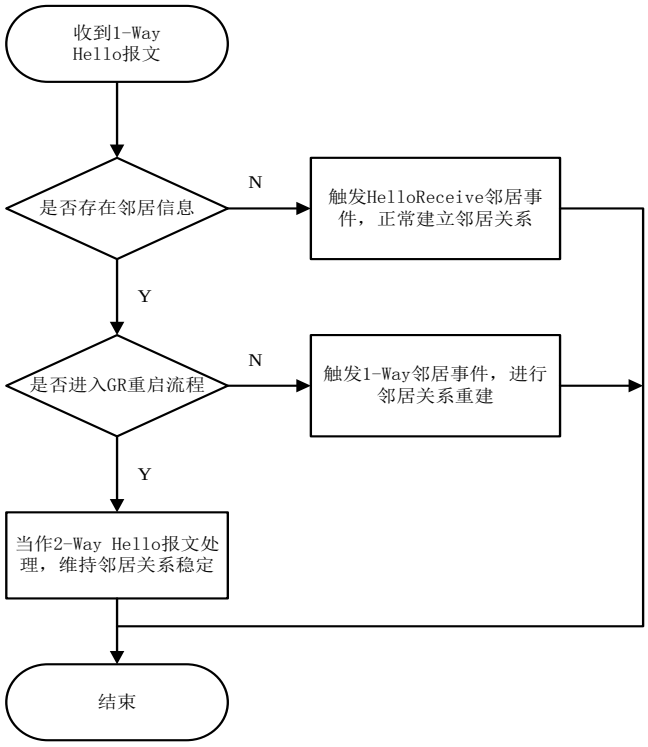

图 6.3　GR 过程中 GRHelper 对邻居信息的维护

当收到邻居发送的 1-Way Hello 报文时，GRHelper 首先在接收接口下查找是否存在该 Hello 报文发送端的邻居信息。若不存在相关邻居信息，则触发 HelloReceived 邻居事件，通过邻居状态机进行正常的邻居关系建立。否则，若接口下已存在相应的邻居信息，则说明与对端设备之前已经建立了邻居关系，需进一步判定该邻居是否处于 GR 重启过程中，通过检查邻居结构下的 ucFlag 标志位来实现。若 ucFlag 未打上 NBR_IETFGR 标记，则说明当前未处于 GR 重启流程中，则触发 1-Way 邻居事件，相应的邻居状态从 Full 状态降为 Init 状态，进行邻居关系的重建，同时也会将该邻居设备从相应的报文转发路径上删除。否则，若 ucFlag 打上了 NBR_IETFGR 标记，则说明本端设备正作为该邻居的 GRHelper，协助其进行 GR 重启，此时不触发 1-Way 邻居事件，而是将此 1-Way Hello 报文当作用于维持邻居关系的 2-Way Hello 报文来处理。这样就能维持与 GRRestarter 邻居关系的稳定，邻居状态不会下降，也不会将邻居设备从相应的报文转发路径上删除，从而保证了数据转发流量不中断。同时，GRHelper 在回复的 Hello 报文中也继续对 GRRestarter 进行通告。主要代码实现如下：

```
void OneWayHelloProc（）
{
    ……
    pstNbr = NBR_Get（）；
    if（NULL == pstNbr）
    {
        NBR_NSMSch(pstNbr, NSM_EVENT_HELLORECEIVE_E);
    }
    else
    {
        if(BIT_TEST（pstNbr->ucGRFlag，NBR_IETFGR）)
        {
            NBR_NSMSch(pstNbr, NSM_EVENT_2WAYRECEIVE_E);
        }
        else
        {
            NBR_NSMSch(pstNbr, NSM_EVENT_1WAY_E);
        }
    }
    ……
}
```

6.3.3 LSDB 数据维护

在收到 GRRestarter 的请求后，GRHelper 需要协助其进行 LSDB 数据的同步和恢复，如图 6.4 所示。首先，GRRestarter 通过向 GRHelper 发送 DD 报文来进行主从关系的协商。由于 GR 过程中，所有的 LSDB 数据均处于 GRHelper 中，因此固定 GRHelper 为主，GRRestarter 为从，而不依赖于相应的路由器 ID。GRHelper 向 GRRestarter 发送宣称自己为主的 DD 报文，同时在其中携带 LSDB 汇总信息。获取了 LSDB 汇总情况后，GRRestarter 通过向 GRHelper 发送 LSR 报文来请求相应的 LSA 数据。GRHelper 在收到请求后，将完整的 LSDB 通过 LSU 报文发送给 GRRestarter，协助其进行 LSDB 数据的恢复和同步。GRRestarter 从 LSU 报文中提取出相应的 LSA，并添加到本地 LSDB 中，即可完成对本地 LSDB 数据库的重建，邻居双方重新达到 LSDB 同步状态。

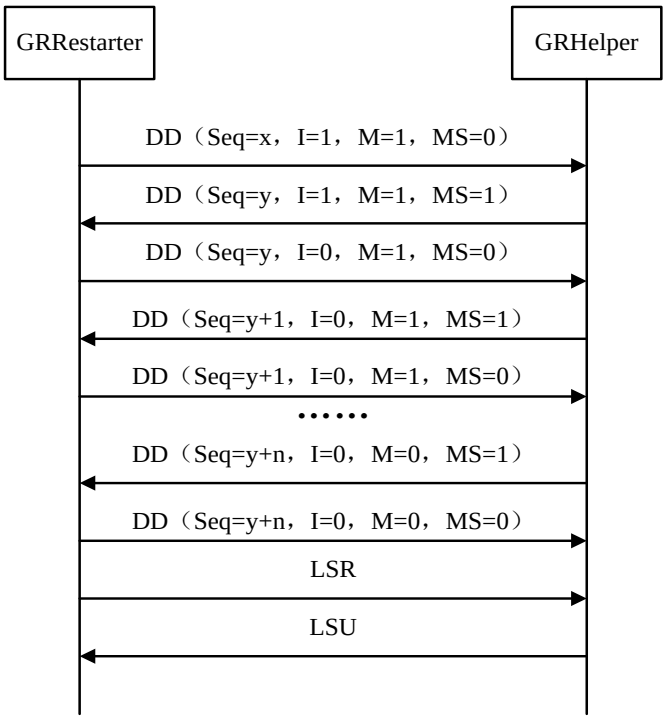

图 6.4　LSDB 数据同步

此外，为了保证网络中链路状态信息的稳定，对于因邻居状态变化引起的 LSA 更新，GRHelper 在 GR 过程中将不再进行处理，等退出 GR 流程后统一进行 LSA 的刷新处理。

6.3.4 路由信息维护

由于 GRRestarter 在 GR 重启过程中不能产生各类 LSA，因此 GRHelper 在 GR 过程中仍然根据 GRRestarter 重启前产生的各类 LSA 进行路由计算。同时，GRHelper 在 GR 过程中不会对与 GRRestarter 相关的各类 LSA 进行重产生处理，也不会触发进行路由计算来刷新路由表项，GRRestarter 设备在相应的报文转发路径上依然有效。这样就保证了 GR 重启过程中，网络中不会产生路由振荡，数据转发路径保持不变，不会产生数据流量的中断。

6.4 退出 GR

6.4.1 正常退出 GR

当收到 GRRestarter 发送的 LS Age 为 3600s 的 Grace LSA 时，GRHelper 调用 GR_MaxAgeGraceLSAProc 接口进行退出 GR 流程处理。

```
void GR_MaxAgeGraceLSAProc（）
{
    ……
    pstNbr = NBR_Get（）；
    if（（NSM_STATE_FULL_E == NSM_CURSTATE（pstNbr））||
        （（ISM_STATE_DROTHER_E == ISM_CURSTATE（pstIf））&&
          （NSM_STATE_2WAY_E == NSM_CURSTATE（pstNbr））））
    {
        GR_LeaveHelper（）；
    }
    else
    {
        GR_AbnorLeaveHelper（）；
    }
    ……
}
```

首先在接收接口下查找并获取发送端的邻居信息，并检查相应的邻居状态。若邻居状态为 Full 或者 DRother 之间的邻居状态为 2-Way，则说明 GRRestarter 完成了邻居关系的重建和 LSDB 数据的同步，GRHelper 的协助工作结束，进行正常退出 GR 重启流程处理。否则，若相应的邻居状态不满足上述条件，则说明 GRRestarter 在 GR 重启过程中出现了异常情况，强制退出了 GR 重启流程， 此时 GRHelper 进行异常退出 GR 重启流程处理。

如图 6.5 所示，正常退出 GR 重启流程时，GRHelper 对邻居结构下的 ucGRFlag 标志位进行复位，删除邻居对应的 GR 周期定时器，退出 GRHelper 模式。同时，还需将进程下的 GRRestarter 计数递减。若 GRRestarter 计数为 0，即完成对所有 GRRestarter 的协助工作，则触发重新产生 Router-LSA 和 Intra-Area-Prefix-LSA。若 GRHelper 为对应链路上的 DR，则还需对 Network-LSA 进行重新产生处理。同时，各类 LSA 的重新产生还会触发路由计算线程重新进行路由计算，并将计算得出的最新路由信息下发路由表来刷新路由表项[53]。

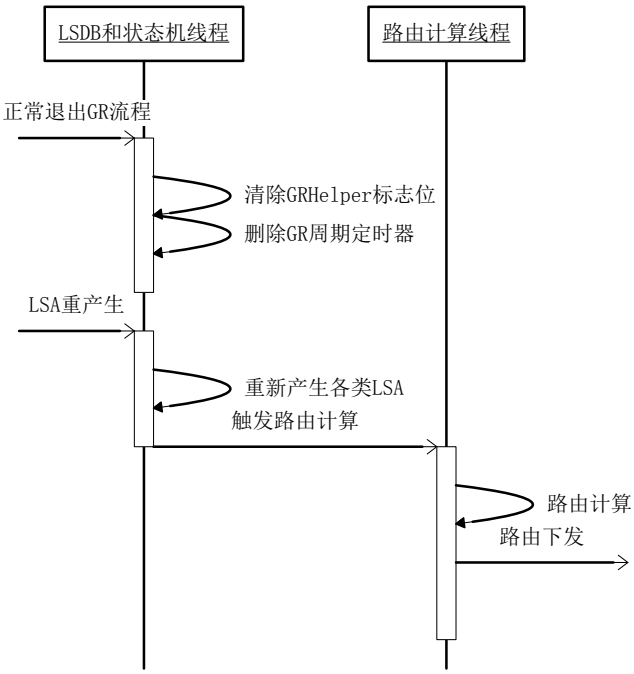

图 6.5　GRHelper 正常退出 GR 流程处理

6.4.2 异常退出 GR

在 GRHelper 协助 GRRestarter 进行 GR 重启的过程中，若出现异常事件，则会导致 GRHelper 强制退出 GR 重启流程。当前实现中，主要有如下场景会触发 GRHelper 异常退出 GR 重启流程。

场景一：GRHelper 端 GR 周期定时器超时，说明 GRRestarter 未能在 GR 周期时间间隔内完成邻居关系的重建和 LSDB 数据的同步。

场景二：触发了 1-Way 邻居状态机事件，这可能由邻居建立过程出现错误引起。

场景三：触发了 KillNbr 邻居状态机事件，这可能由网络拓扑发生变化引起。

场景四：进程下配置了 LSA 严格检查，GR 过程中触发了 LSA 的变化，这可能由 LSA 的产生、刷新和老化等操作引起。

场景五：收到 GRRestarter 发送的 LS Age 为 3600s 的 Grace LSA，但相应的邻居关系并未重建完成，这可能由 GRRestarter 端出现异常强制退出 GR 重启流程引起。

当出现上述异常情况时，GRHelper 将触发进行异常退出 GR 重启流程处理，具体处理过程如下：LSDB 和状态机线程遍历进程下的所有邻居，若处于 GR 重启过程中，则触发退出 GRHelper 模式处理，强制退出 GR 重启流程。对于每个 GRRestarter 邻居，复位邻居结构下的 GRHelper 模式标志位，删除对应的 GR 周期定时器，同时对进程下的 GRRestarter 计数进行递减。对于还未完成邻居关系建立的 GRRestarter 邻居，向其发送 1-Way Hello 报文，触发邻居设备的 1-Way 事件。在所有邻居均退出 GRHelper 模式后，对 Router-LSA 和 Intra-Area-Prefix-LSA 进行重新产生处理。若 GRHelper 是对应链路上的 DR，则还需重新产

生 Network-LSA。同时，根据新生成的各类 LSA 调度路由计算线程重新进行路由计算，刷新路由表项。

6.5 本章小结

本章在进行软件需求分析的基础上，对 GRHelper 模块进行了设计实现。文中详细阐述了 GRHelper 模块在 GR 重启过程中的具体设计流程，包括进入 GR 流程的场景及进入条件的判定，GRHelper 在 GR 重启过程中对定时器、邻居信息、LSDB 数据和路由信息的维护和处理，以及正常退出 GR 流程处理和异常退出 GR 流程处理。同时，文中也给出了部分数据结构的设计和函数接口的实现。

在 GRHelper 模块的设计过程中需要注意以下几点：

（1）GR 周期定时器时间间隔需要在 Grace LSA 中携带的 GR 周期时间基础上减去该 Grace LSA 的 LS Age 数值，即要将 Grace LSA 的传输延迟时间考虑在内，保证定时器时间的精确。

（2）收到 Grace LSA 时，需检查相应的邻居信息，之前具有邻居关系才能进入 GR 流程，成为 GRHelper。

（3）收到重复的 Grace LSA 时，需要重置对应的 GR 周期定时器，以保证邻居两端定时器时间的一致。

（4）收到 LS Age 为 3600s 的 Grace LSA 时，需根据对应的邻居信息分别进行处理。若已完成邻居关系的重建，则进入正常退出 GR 流程处理；否则，进入异常退出 GR 流程处理。

第7章 测试与验证

本章针对前文所述的系统设计方案，设计相应的测试组网，对整体系统进行测试，验证设计方案的正确性和可行性，具体包括功能测试和性能测试。在功能测试中，设计相应的测试用例对 GRRestarter 模块和 GRHelper 模块分别进行测试，主要关注基本功能实现的正确性和完整性。在性能测试中，主要关注大规格路由情况下系统的稳定性和持续性。

7.1 测试环境构建

7.1.1 测试组网设计

根据系统测试方案的需要，设计如图 7.1 所示的测试组网来对整体系统进行测试和验证。

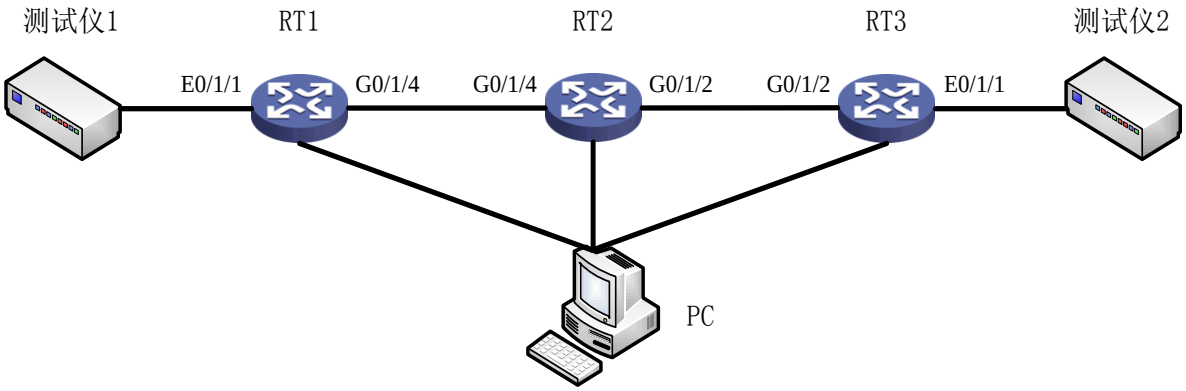

图 7.1　系统测试组网图

系统测试组网主要由测试仪、路由器和 PC 组成。其中测试仪设备选用 Spirent 公司的 TestCenter 测试仪，主要用于产生 IP 网络数据流量。路由器设备选用某通信公司的路由器，为系统运行提供软硬件环境，也是进行系统测试的对象。所选用的路由器为分布式设备，并且控制层面具有主备环境。测试仪通过相应接口与路由器进行连接，测试仪 1 通过 E0/1/1 接口与 RT1 相连接，测试仪 2 通过 E0/1/1 接口与 RT3 相连接。三台路由器设备之间也通过相应接口进行连接，RT1 和 RT2 通过 G0/1/4 接口相连接，RT2 和 RT3 通过 G0/1/2 接口相连接。PC 机通过路由器设备的网络管理接口使用 Telnet 方式远程登录到 RT1、RT2 和 RT3，主要对路由器设备进行配置管理，并获取相应的运行数据和状态信息。

7.1.2 配置命令

用户通过配置命令与路由器设备进行交互，包括配置系统的相关特性、管理系统的正常运行以及获取系统的运行数据和状态信息等。命令配置视图主要包括系统视图、进程视图、区域视图和接口视图。以下列举系统测试中涉及的主要配置命令。

（1）OSPFv3 [process-id]

系统视图配置命令，用于创建 OSPFv3 协议进程。配置时可携带进程号，配置后进入

OSPFv3 进程视图。

（2）OSPFv3 [process-id] Area [area-id]

接口视图配置命令，用于将接口使能到 OSPFv3 进程和区域中。配置时需要指定相应的进程号和区域号。

（3）graceful-restart enable

OSPFv3 进程视图配置命令，用于使能 OSPFv3 进程的 GRRestarter 能力。

（4）graceful-restart helper enable

OSPFv3 进程视图配置命令，用于使能 OSPFv3 进程的 GRHelper 能力。

（5）graceful-restart interval <40 — 1800>

OSPFv3 进程视图配置命令，用于配置 GRRestarter 端的 GR 周期定时器时间间隔，取值范围为 40——1800s，默认为 120s。

7.2 功能测试

本节主要对软件系统进行功能测试，包括正常流程测试和异常流程测试，使用图 7.1 中的 RT1、RT2 和 RT3 三台路由器设备。

7.2.1 OSPFv3 邻居建立

首先在 RT1、RT2 和 RT3 之间建立 OSPFv3 邻居关系[54]，RT1 和 RT2 通过 G0/1/4 接口在区域 0 建立邻居，RT2 和 RT3 通过 G0/1/2 接口在区域 1 建立邻居，配置过程如下：

（1）RT1 配置

1) 配置 OSPFv3 进程 1，路由器 ID 为 1.1.1.1

```
<RT1> system-view
[RT1] OSPFv3 1
[RT1-OSPFv3-1] router-id 1.1.1.1
[RT1-OSPFv3-1] quit
```

2) 接口 G0/1/4 上配置 IPv6 地址，并使能到 OSPFv3 进程 1、区域 0 中

```
[RT1] interface GigabitEthernet 0/1/4
[RT1-GigabitEthernet 0/1/4] ipv6 address 1000:4::1 64
[RT1-GigabitEthernet 0/1/4] ipv6 address FE80:100::4:1 128 link-local
[RT1-GigabitEthernet 0/1/4] ospfv3 1 area 0
[RT1-GigabitEthernet 0/1/4] quit
```

（2）RT2 配置

1) 配置 OSPFv3 进程 1，路由器 ID 配置为 2.2.2.2

```
<RT2> system-view
[RT2] OSPFv3 1
[RT2-OSPFv3-1] router-id 2.2.2.2
```

[RT2-OSPFv3-1] quit

2) 接口 G0/1/4 上配置 IPv6 地址，并使能到 OSPFv3 进程 1、区域 0 中。接口 G0/1/2 上配置 IPv6 地址，并使能到 OSPFv3 进程 1、区域 1 中。

[RT2] interface GigabitEthernet 0/1/4

[RT2-GigabitEthernet 0/1/4] ipv6 address 2000:4::2 64

[RT2-GigabitEthernet 0/1/4] ipv6 address FE80:200::4:2 128 link-local

[RT2-GigabitEthernet 0/1/4] ospfv3 1 area 0

[RT2-GigabitEthernet 0/1/4] quit

[RT2] interface GigabitEthernet 0/1/2

[RT2-GigabitEthernet 0/1/2] ipv6 address 2000:2::2 64

[RT2-GigabitEthernet 0/1/2] ipv6 address FE80:200::2:2 128 link-local

[RT2-GigabitEthernet 0/1/2] ospfv3 1 area 1

[RT2-GigabitEthernet 0/1/2] quit

（3）RT3 配置

1) 配置 OSPFv3 进程 1，路由器 ID 配置为 3.3.3.3

<RT3> system-view

[RT3] OSPFv3 1

[RT3-OSPFv3-1] router-id 3.3.3.3

[RT3-OSPFv3-1] quit

2) 接口 G0/1/2 上配置 IPv6 地址，并使能到 OSPFv3 进程 1、区域 1 中

[RT3] interface GigabitEthernet 0/1/2

[RT3-GigabitEthernet 0/1/2] ipv6 address 3000:2::3 64

[RT3-GigabitEthernet 0/1/2] ipv6 address FE80:300::2:3 128 link-local

[RT3-GigabitEthernet 0/1/2] ospfv3 1 area 1

[RT3-GigabitEthernet 0/1/2] quit

（4）查看邻居信息

1) RT1 邻居信息如图 7.2 所示

```
[RT1] display ospfv3 peer
                OSPFv3 Process 1 with Router ID 1.1.1.1
Area 0.0.0.0
--------------------------------------------------------------------------------
Router-ID         Pri         State          Interface          Inst ID
 2.2.2.2           1         Full/DR          G0/1/4              0
```

图 7.2　RT1 邻居信息

2) RT2 邻居信息如图 7.3 所示

```
[RT2] display ospfv3 peer
                OSPFv3 Process 1 with Router ID 2.2.2.2
Area 0.0.0.0
--------------------------------------------------------------------------------
Router-ID         Pri         State         Interface         Inst ID
 1.1.1.1          1           Full/BDR      G0/1/4            0
Area 0.0.0.1
--------------------------------------------------------------------------------
Router-ID         Pri         State         Interface         Inst ID
 3.3.3.3          1           Full/DR       G0/1/2            0
```

图 7.3　RT2 邻居信息

3) RT3 邻居信息如图 7.4 所示

```
[RT3] display ospfv3 peer
                OSPFv3 Process 1 with Router ID 3.3.3.3
Area 0.0.0.1
--------------------------------------------------------------------------------
Router-ID         Pri         State         Interface         Inst ID
 2.2.2.2          1           Full/BDR      G0/1/2            0
```

图 7.4　RT3 邻居信息

可以看到，RT2 和 RT1 在区域 0 建立了 Full 邻居关系，和 RT3 在区域 1 建立了 Full 邻居关系。

7.2.2 正常流程测试

对正常 GR 流程的测试主要关注 GRRestarter 和 GRHelper 在 GR 重启过程中对协议报文、定时器、邻居信息、LSDB 数据的维护处理以及 GR 状态机的运转情况。

7.2.2.1 测试输入

（1）打开调试信息开关

RT1、RT2 和 RT3 上均打开 GR 调试信息开关和邻居调试信息开关，这样就可以通过输出的调试信息查看到系统对实时事件的处理情况和邻居状态变迁情况。

<RT1> debugging ospfv3 event graceful-restart

<RT1> debugging ospfv3 event neighbor

RT2、RT3 上配置过程同 RT1，不再赘述。

（2）使能 GR 能力

对 RT2 的 OSPFv3 进程使能 GRRestarter 能力，RT1 和 RT3 的 OSPFv3 进程使能 GRHelper 能力。

[RT1-OSPFv3-1] graceful-restart helper enable

[RT2-OSPFv3-1] graceful-restart enable

[RT3-OSPFv3-1] graceful-restart helper enable

（3）触发进入 GR 重启流程

RT2 上执行进程亲和力重优化操作，进行进程级主备倒换，触发 RT2 进入 GRRestarter 模式，RT1 和 RT3 进入 GRHelper 模式，从而进入 GR 重启流程处理。

[RT2] placement reoptimization

7.2.2.2 输出信息观察

（1）GRRestarter 输出信息

1) 进程 GR 信息

RT2 上查看 OSPFv3 进程 GR 重启的相关信息，包括 GR 能力使能情况、GR 状态、GR 周期时间间隔和 GRHelper 计数等信息，如图 7.5 所示。

```
[RT2] display ospfv3 graceful-restart status
                OSPFv3 Process 1 with Router ID 2.2.2.2
 Graceful-restart capability :      Enable
 Graceful-restart Helper capability :      Unable
 Current GR State :      UnderGR
 Graceful-restart period :      120 seconds
 Number of Neighbor under helper :      2
 Number of Neighbor under restarting :      0
```

图 7.5　RT2 正常流程进程 GR 信息

2) GR 状态变迁信息

RT2 上查看 GR 重启过程中 OSPFv3 进程 GR 状态的变迁过程，也即 GR 状态机的运转流程，如图 7.6 所示。

```
[RT2]display ospfv3 graceful-restart state change
( ProcID,   Event,      PreState,   CurState,   Time Sec,    Time mSec,   Id )
-------------------------------------------------------------------------------
(   1       upgrade     normal      prepare     1338343256   459296      0 )
(   1       entergr     prepare     grdoing     1338343256   459302      1 )
(   1       allnbrdone  grdoing     generate    1338343296   462296      2 )
(   1       genover     generate    calc        1338343296   472296      3 )
(   1       calcover    calc        redist      1338343296   549296      4 )
(   1       redistover  redist      flush       1338343296   551296      5 )
(   1       flushover   fush        batch       1338343296   572329      6 )
(   1       batchover   batch       normal      1338343296   583528      7 )
```

图 7.6　RT2 正常流程 GR 状态变迁

3) 调试信息

查看 RT2 在 GR 重启过程中输出的调试信息如图 7.7 所示。

```
11:51:05:282  OSPFv3/7/OSPFv3DEBUG:OSPFv3 1 Neighbor 1.1.1.1( GigabitEthernet 0/1/4) received killNbr
and its state from full -> Down
11:51:05:287  OSPFv3/7/OSPFv3DEBUG:OSPFv3 1 Neighbor 3.3.3.3( GigabitEthernet 0/1/2) received killNbr
and its state from full -> Down
16:51:05:361  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: OSPFv3 Process will enter IETF GR
16:51:05:399  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Create global suppress hello timer，timeout value is
5000ms
16:51:05:401  OSPFv3/7/OSPFv3DEBUG:  OSPFv3 1: Create Grace LSA send timer，timeout value is 1000ms
16:51:05:411  OSPFv3/7/OSPFv3DEBUG:  OSPFv3 1:  Create GR waiting timer，timeout value is 40000ms
16:51:06:410  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Send Grace LSA
16:51:07:410  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Send Grace LSA
16:51:08:410  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Send Grace LSA
16:51:09:410  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Send Grace LSA
16:51:10:410  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Send Grace LSA
16:51:10:415  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete global suppress hello timer
16:51:10:426  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete Grace LSA send timer
16:51:10:841  OSPFv3/7/OSPFv3DEBUG:  OSPFv3 1: Create GR Interval timer，timeout value is 120000ms
16:51:11:338  OSPFv3/7/OSPFv3DEBUG:  Neighbor 1.1.1.1 ( GigabitEthernet 0/1/4)  received HelloReceived
and its state from Down -> Init
16:51:11:341  OSPFv3/7/OSPFv3DEBUG:  Neighbor 1.1.1.1 ( GigabitEthernet 0/1/4)  received 2WayReceived
and its state from Init -> ExStart
16:51:13:188  OSPFv3/7/OSPFv3DEBUG: Neighbor 1.1.1.1( GigabitEthernet 0/1/4) received NegotiationDone
and its state from ExStart -> ExChange
16:51:13:403  OSPFv3/7/OSPFv3DEBUG:  Neighbor 1.1.1.1 ( GigabitEthernet 0/1/4)  received ExchangeDone
and its state from ExChange -> Loading
16:51:14:121  OSPFv3/7/OSPFv3DEBUG:  Neighbor 1.1.1.1 ( GigabitEthernet 0/1/4)  received LoadingDone
and its state from Loading -> Full
16:51:14:889  OSPFv3/7/OSPFv3DEBUG:  Neighbor 3.3.3.3 ( GigabitEthernet 0/1/2)  received HelloReceived
and its state from Down -> Init
16:51:14:921  OSPFv3/7/OSPFv3DEBUG:  Neighbor 3.3.3.3 ( GigabitEthernet 0/1/2)  received 2WayReceived
and its state from Init -> ExStart
16:51:16:125  OSPFv3/7/OSPFv3DEBUG: Neighbor 3.3.3.3( GigabitEthernet 0/1/2) received NegotiationDone
and its state from ExStart -> ExChange
16:51:16:557  OSPFv3/7/OSPFv3DEBUG:  Neighbor 3.3.3.3 ( GigabitEthernet 0/1/2)  received ExchangeDone
and its state from ExChange -> Loading
16:51:17:233  OSPFv3/7/OSPFv3DEBUG:  Neighbor 3.3.3.3 ( GigabitEthernet 0/1/2)  received LoadingDone
and its state from Loading -> Full
16:51:45:455  OSPFv3/7/OSPFv3DEBUG:  OSPFv3 1: Graceful restart，AllNbr have been done
16:51:45:463  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete GR Interval timer
16:51:45:471  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete GR Waiting timer
16:51:45:477  OSPFv3/7/OSPFv3DEBUG:  OSPFv3 1: Flush Grace LSA，Send MaxAge Grace LSA
16:51:45:489  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Graceful Restart finish
```

图 7.7　RT2 正常流程调试信息

（2）GRHelper 输出信息

RT1 和 RT3 均进入 GRHelper 模式，都能观察到相应的 GR 信息和调试信息，此处以 RT1 的输出信息为例，RT3 类似。

1) GR 信息

RT1 上查看 OSPFv3 进程 GR 信息如图 7.8 所示。

```
[RT1] display ospfv3 graceful-restart status
                OSPFv3 Process 1 with Router ID 1.1.1.1
  Graceful-restart capability :      Unable
  Graceful-restart Helper capability :      Enable
  Current GR State :      UnderHelper
  Graceful-restart period :      119 seconds
  Number of Neighbor under helper :      0
  Number of Neighbor under restarting :      1
```

图 7.8　RT1 正常流程进程 GR 信息

2) 调试信息

GR 重启过程中，RT1 作为 GRHelper 输出的调试信息如图 7.9 所示。

```
17:22:54:778 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received New Grace LSA from Neighbor 2.2.2.2
17:22:54:790 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Enter helper mode for Neighbor 2.2.2.2, neighbor count in
IETF graceful restart is 1
17:22:54:796 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Create GR Period timer for Neighbor 2.2.2.2, timeout value
is 119 (s)
17:22:55:776 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received New Grace LSA from Neighbor 2.2.2.2
17:22:55:808 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Reset GR Period timer for Neighbor 2.2.2.2, timeout value
is 119 (s)
17:22:56:239 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received New Grace LSA from Neighbor 2.2.2.2
17:22:56:245 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Reset GR Period timer for Neighbor 2.2.2.2, timeout value
is 119 (s)
17:22:57:480 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received New Grace LSA from Neighbor 2.2.2.2
17:22:57:489 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Reset GR Period timer for Neighbor 2.2.2.2, timeout value
is 119 (s)
17:22:58:746 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received New Grace LSA from Neighbor 2.2.2.2
17:22:58:751 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Reset GR Period timer for Neighbor 2.2.2.2, timeout value
is 119 (s)
17:23:00:550 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Neighbor 2.2.2.2 (GigabitEthernet 0/1/4) received
SeqNumberMismatch and its state from Full -> Exstart
17:23:02:072 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Neighbor 2.2.2.2 (GigabitEthernet 0/1/4) received
NegotiationDone and its state from Exstart -> ExChange
17:23:02:115 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Neighbor 2.2.2.2 (GigabitEthernet 0/1/4) received
ExchangeDone and its state from ExChange -> Loading
17:23:02:125 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Neighbor 2.2.2.2 (GigabitEthernet 0/1/4) received
LoadingDone and its state from Loading -> Full
17:23:42:262 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Received MaxAge Grace LSA from Neighbor 2.2.2.2
17:23:42:267 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Leave helper mode for Neighbor 2.2.2.2, Neighbor count
in IETF graceful restart is 0
17:23:42:270 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete Grace Period timer for Neighbor 2.2.2.2
17:23:42:278 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Leave all helper mode
```

图 7.9　RT1 正常流程调试信息

7.2.2.3 测试结果分析

通过 OSPFv3 进程下显示的 GR 信息可以看到，RT2 的 OSPFv3 进程使能了 GRRestarter 能力，当前正处于 GR 重启过程中，GR 周期时间间隔为 120s，相应的 GRHelper 计数为 2，即周边有两台路由器协助其进行 GR 重启。RT1 和 RT3 的 OSPFv3 进程使能了 GRHelper 能力，

当前正作为 GRHelper 协助 RT2 进行 GR 重启，相应的 GRRestarter 计数为 1。

进程下记录的调试信息反映了 GRRestarter 设备和 GRHelper 设备在整个 GR 重启过程中的流程处理和状态变迁。可以看到，进行主备倒换后，RT2 首先删除相应的邻居信息，进入 GR 重启流程。然后，创建全局抑制 Hello 报文定时器、Grace LSA 发送定时器、WaitNbr 定时器和 GR 周期定时器。同时，向外发送 Grace LSA 来通告邻居进入 GR 重启流程，默认发送 5 次。RT1 和 RT3 在收到 Grace LSA 后，成为 GRHelper 进入 GR 重启流程，同时创建 GR 周期定时器并增加进程下的 GRRestarter 计数。收到重复的 Grace LSA 时，重置 GR 周期定时器。

随后，RT2 开始进行协议报文的收发来重建邻居关系。通过输出的邻居状态调试信息可以观察到，RT1 和 RT3 并没有断开与 RT2 的邻居关系，保持了邻居信息的稳定。在收到 RT2 发送的 DD 报文后，RT1、RT3 和 RT2 的邻居关系降为 ExStart 状态，并且将完整的 LSDB 数据发送给 RT2，协助其进行 LSDB 数据的恢复和重建。

完成 LSDB 数据的同步和邻居关系的重建后，RT2 删除进程下相应的定时器，同时向外发送 MaxAge（LS Age 为 3600s）的 Grace LSA，正常退出 GR 重启流程。RT1 和 RT3 收到 MaxAge 的 Grace LSA 后，退出 GRHelper 模式，删除 GR 周期定时器，递减进程下的 GRRestarter 计数，也正常退出 GR 重启流程。

此外，通过 RT2 的 OSPFv3 进程下记录的 GR 状态变迁信息，可以观察到 OSPFv3 进程在 GR 重启过程中的状态变迁，同时也反映了 GR 状态机的运转情况。

通过对上述测试输出信息的观察和分析，可以说明 GRRestarter 和 GRHelper 在正常的 GR 重启流程中对协议报文、定时器、邻居信息和 LSDB 数据的维护和处理正确可行，同时 GR 状态机运转正常，满足设计方案要求。

7.2.3 异常流程测试

对 GR 异常流程的测试主要关注 GRRestarter 和 GRHelper 在 GR 重启过程中对异常事件的处理以及 GR 状态机的运转情况。

7.2.3.1 测试输入

（1）触发进入 GR 重启流程

RT2 上执行进程亲和力重优化操作，进行进程级主备倒换，触发 RT2 进入 GRRestarter 模式，RT1 和 RT3 进入 GRHelper 模式，从而进入 GR 重启流程处理。

[RT2] placement reoptimization

（2）触发异常事件

在 GR 重启过程中，构造以 RT1 为源端的 1-Way Hello 报文，并打入 RT2，触发 RT2 的 1-Way 邻居事件。

7.2.3.2 输出信息观察

（1）GRRestarter 输出信息

1) 调试信息

RT2 上输出的调试信息如图 7.10 所示。

```
17:21:45:455  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: received neighbor event 1-way, abnormal quit graceful
restart
17:21:45:464  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete GR Interval timer
17:21:45:465  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete GR Waiting timer
17:21:45:473  OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Flush Grace LSA, Send MaxAge Grace LSA
```

图 7.10 RT2 异常流程调试信息

2) GR 状态变迁信息

RT2 上查看 OSPFv3 进程下记录的 GR 状态变迁信息如图 7.11 所示。

```
[RT2] display ospfv3 graceful-restart state change
(ProcID,   Event,     PreState,   CurState,   Time Sec,    Time mSec,   Id)
-------------------------------------------------------------------------------
(   1      upgrade    normal      prepare     1438563471   218687       0)
(   1      entergr    prepare     grdoing     1438563471   218703       1)
(   1      abnormal   grdoing     abnorquit   1438563471   223427       2)
(   1      reset      abnorquit   normal      1438563471   227824       3)
```

图 7.11 RT2 异常流程 GR 状态变迁

（2）GRHelper 输出信息

RT1 上输出的调试信息如图 7.12 所示。

```
17:35:21:354 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Abnormal quit helper mode for neighbor 2.2.2.2
17:35:21:358 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Leave helper mode for Neighbor 2.2.2.2, Neighbor count
in IETF graceful restart is 0
17:35:21:362 OSPFv3/7/OSPFv3DEBUG: OSPFv3 1: Delete Grace Period timer for Neighbor 2.2.2.2
```

图 7.12 RT1 异常流程调试信息

7.2.3.3 测试结果分析

通过输出的调试信息可以看到，GR 重启过程中收到邻居发送的 1-Way Hello 报文后，GRRestarter 触发 1-Way 邻居事件，同时也触发 GR 状态机 Abnormal 事件，删除进程下的定时器，并向外发送 MaxAge 的 Grace LSA，异常退出 GR 重启流程。GRHelper 在收到 Grace LSA 后，也会触发异常退出 GR 重启流程，删除 GR 周期定时器，并递减进程下的 GRRestarter 计数。进程下记录的 GR 状态变迁信息也反映了 GR 状态机的运转正常，整个异常流程处理正确可行，满足设计要求。

7.3 性能测试

本节主要对整体系统进行性能测试[55]，包括 GR 重启过程中数据转发流量持续性的测试和大路由情况下系统稳定性的测试。

7.3.1 数据转发流量测试

OSPFv3 GR 机制的主要功能即在路由器设备控制层面进行协议重启的过程中保证转发层

面数据流量不中断。在对 GR 重启过程中数据转发流量持续性和稳定性的测试中，我们选用 RT1、RT2 和 RT3 三台路由器设备。Ping 命令是检测网络连通性最常用的工具之一，因此我们通过 Ping 命令来检测路由器设备重启过程中数据转发流量是否出现中断。测试过程中，我们选择 RT2 作为 GRRestarter 设备进行 OSPFv3 协议进程重启，同时通过 RT3 设备不断 ping RT1 设备 G0/1/4 接口的 IPv6 地址来检测网络的连通性和数据转发流量的持续性，需要对进程未使能 GR 能力和使能 GR 能力两种情况进行对比测试。

首先对 OSPFv3 进程未使能 GR 能力的情况进行测试。RT1、RT2 和 RT3 之间建立如前文所述的 Full 邻居关系，其中 RT3 的 G0/1/2 接口 IPv6 地址为 3000:2::3/64，RT1 的 G0/1/4 接口 IPv6 地址为 1000:4::1/64。此时 RT2 上执行主备倒换操作，RT3 上 ping 命令执行结果如图 7.13 所示。

```
[RT3] ping ipv6 -C 10000 -m 1000 1000:4::1
 PING6（104 = 40 + 8 + 56 bytes）  3000:2::3 ---> 1000:4::1
56 bytes from 1000:4::1, icmp_seq = 0 hlim = 64 tim = 16.000ms
56 bytes from 1000:4::1, icmp_seq = 1 hlim = 64 tim = 13.000ms
56 bytes from 1000:4::1, icmp_seq = 2 hlim = 64 tim = 17.000ms
56 bytes from 1000:4::1, icmp_seq = 3 hlim = 64 tim = 68.000ms
ping6：sendmsg：No route to host
Request time out
ping6：sendmsg：No route to host
Request time out
ping6：sendmsg：No route to host
Request time out
ping6：sendmsg：No route to host
Request time out
56 bytes from 1000:4::1, icmp_seq = 8 hlim = 64 tim = 8.000ms
56 bytes from 1000:4::1, icmp_seq = 9 hlim = 64 tim = 13.000ms
56 bytes from 1000:4::1, icmp_seq = 10 hlim = 64 tim = 13.000ms
56 bytes from 1000:4::1, icmp_seq = 11 hlim = 64 tim = 11.000ms
56 bytes from 1000:4::1, icmp_seq = 12 hlim = 64 tim = 18.000ms

--- 3000:2::3 ping6 statistics ---
13 packet(s) tranmmited，9 packet(s) received，30.8% packet loss
round-trip min/avg/max/std-dev = 8.000/19.667/68.000/17.333 ms
```

图 7.13　未使能 GR 情况下 ping 命令执行结果

再对 OSPFv3 进程使能 GR 能力的情况进行测试。RT2 的 OSPFv3 进程使能 GRRestarter 能力，RT1 和 RT3 的 OSPFv3 进程使能 GRHelper 能力。RT2 上再次执行主备倒换操作，RT3 上 ping 命令执行结果如图 7.14 所示。

```
[RT3] ping ipv6 -C 10000 -m 1000 1000:4::1
 PING6（104 = 40 + 8 + 56 bytes） 3000:2::3 ---> 1000:4::1
 56 bytes from 1000:4::1, icmp_seq = 0 hlim = 64 tim = 15.000ms
 56 bytes from 1000:4::1, icmp_seq = 1 hlim = 64 tim = 150.000ms
 56 bytes from 1000:4::1, icmp_seq = 2 hlim = 64 tim = 30.000ms
 56 bytes from 1000:4::1, icmp_seq = 3 hlim = 64 tim = 25.000ms
 56 bytes from 1000:4::1, icmp_seq = 4 hlim = 64 tim = 28.000ms
 56 bytes from 1000:4::1, icmp_seq = 5 hlim = 64 tim = 18.000ms
 56 bytes from 1000:4::1, icmp_seq = 6 hlim = 64 tim = 54.000ms
 56 bytes from 1000:4::1, icmp_seq = 7 hlim = 64 tim = 22.000ms
 56 bytes from 1000:4::1, icmp_seq = 8 hlim = 64 tim = 35.000ms
 56 bytes from 1000:4::1, icmp_seq = 9 hlim = 64 tim = 18.000ms
 56 bytes from 1000:4::1, icmp_seq = 10 hlim = 64 tim = 11.000ms
 56 bytes from 1000:4::1, icmp_seq = 11 hlim = 64 tim = 31.000ms
 56 bytes from 1000:4::1, icmp_seq = 12 hlim = 64 tim = 23.000ms

 --- 3000:2::3 ping6 statistics ---
 13 packet(s) tranmmited，13 packet(s) received，0.0% packet loss
 round-trip min/avg/max/std-dev = 11.000/35.385/150.000/34.687 ms
```

图 7.14　使能 GR 情况下 ping 命令执行结果

对比图 7.13 和图 7.14 可以看出，在进程下未使能 GR 能力的情况下，RT2 进行协议重启过程中 RT3 无法 ping 通 RT1，出现数据转发流量中断的现象；而在进程下使能 GR 能力的情况下，RT2 同样进行协议重启，RT3 一直可以 ping 通 RT1，数据转发流量未出现中断。测试结果表明，OSPFv3 GR 机制在路由器设备重启过程中保证数据转发流量不中断的功能得到了实现。

7.3.2 压力测试

压力测试是对存在大规格路由情况下整体系统稳定性的测试。在该测试中，测试仪 1 与 RT1 设备建立 Full 邻居关系，测试仪 2 与 RT3 设备建立 Full 邻居关系，测试仪 1 发送的 IP 数据包通过 RT1、RT2 和 RT3 的转发能够正常到达测试仪 2，而测试仪 2 发送的 IP 数据包通过 RT3、RT2 和 RT1 的转发同样能够正常到达测试仪 1。测试过程中，测试仪 1 向 RT1 持续打入以测试仪 2 为目的地址的 15 万条路由流量，测试仪 2 向 RT3 持续打入以测试仪 1 为目的地址的 7.5 万条路由流量，RT1、RT2 和 RT3 正常工作情况下数据流量能够被持续转发。此处同样需要对进程未使能 GR 能力和使能 GR 能力两种情况进行对比测试。

首先对 OSPFv3 进程未使能 GR 能力的情况进行测试。在 RT2 未使能 GRRestarter 能力、RT1 和 RT3 未使能 GRHelper 能力时，RT2 执行主备倒换操作，数据流量转发情况如图 7.15 所示。

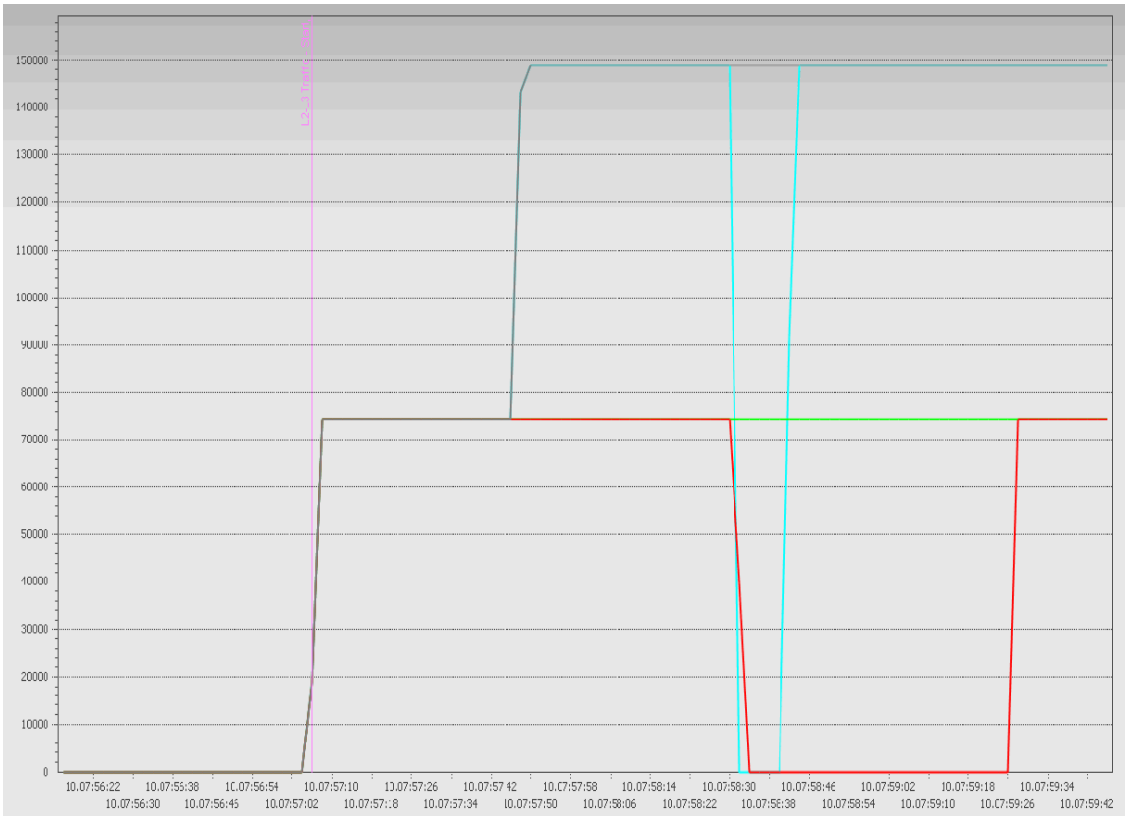

图 7.15　未使能 GR 能力时数据流量转发情况

然后再对 OSPFv3 进程使能 GR 能力的情况进行测试。RT2 进程下使能 GRRestarter 能力，RT1 和 RT3 进程下使能 GRHelper 能力，RT2 执行主备倒换操作，数据流量转发情况如图 7.16 所示。

图 7.16　使能 GR 能力时数据流量转发情况

对比图 7.15 和图 7.16 可以看出，在进程未使能 GR 能力的情况下，路由器设备重启过程中数据转发流量出现了明显的中断现象；而在进程使能 GR 能力的情况下，路由器设备重启过程中数据转发流量保持稳定，未出现中断。该测试结果表明，在大规格路由情况下，整体系统依然能够保证数据转发流量的稳定性和持续性。

7.4 本章小结

本章主要对软件系统的设计结果进行测试和验证，包括功能测试和性能测试。在功能测试中，分别对 GR 正常流程和异常流程进行了测试，验证了 GRRestarter 和 GRHelper 在 GR 重启过程中对协议报文、定时器、邻居信息、LSDB 数据和异常事件处理的正确性。在性能测试中，重点关注了 GR 重启过程中数据转发流量的持续性以及大路由情况下整体系统的稳定性。文中给出了具体的测试步骤和测试输出信息，并对测试结果进行了详细分析。

测试结果表明，本课题中所采用的设计方案正确可行，整体系统运行正常。同时，各线程之间的交互和调度功能正常，对公共数据的访问安全可靠，各个处理流程有序进行，未出现数据访问的冲突和时序问题。

本文设计的软件系统对 OSPFv3 GR 机制进行了较好的实现，能够在路由器设备控制层面协议重启过程中保证转发层面数据流量不中断，满足系统设计要求。

第8章 总结与展望

8.1 全文总结

随着 IP 网络的迅速发展，越来越多的互联网业务依赖于 IP 网络，同时也对 IP 网络运行的可靠性和持续性提出了更高的要求。OSPFv3 协议作为 IPv6 网络中的主流路由协议，其性能直接影响到网络运行质量和数据报文转发效率。由于 OSPFv3 协议自身运行机制的限制，可能导致在协议进程重启时网络中产生路由振荡和数据转发流量中断。GR 机制是一种高可靠性技术，能够保证路由器设备重启过程中数据转发流量不中断，网络中也不会产生路由振荡。因此，在 OSPFv3 协议中提供对 GR 机制的支持已成为一种必然趋势。

本文主要对基于 OSPFv3 协议的 GR 机制进行研究，在实现协议基本功能的基础上，设计实现 OSPFv3 协议对 GR 机制的支持，从而有效地提高了 IP 网络的可靠性。本文的主要工作内容如下：

（1）通过查阅相关文献，了解了 IP 网络中路由协议及其可靠性技术的发展概况。

（2）对 OSPFv3 协议进行了深入研究，详细分析了协议的基本原理和运行机制，主要包括协议报文的收发、邻居关系的建立、LSDB 数据信息和路由信息的维护。

（3）对 GR 机制进行了深入研究，具体分析了 GR 机制的工作原理、技术细节和实现难点。

（4）对整体系统进行了设计实现，包括总体方案设计、具体线程划分，内部功能分解和公用数据结构设计。

（5）对 GRRestarter 模块和 GRHelper 模块分别进行了设计实现，包括 GR 状态机的设计、GRRestarter 和 GRHelper 在 GR 重启过程中对协议报文、定时器、邻居信息、LSDB 数据、路由信息和异常事件的维护和处理。

（6）设计了测试组网，对 GRRestarter 模块和 GRHelper 模块进行了基本功能测试，并对整体系统进行了性能测试和压力测试，从而验证了总体设计方案的正确性和可行性。

实际测试结果表明，本课题中设计实现的基于 OSPFv3 协议的 GR 机制能够在路由器设备控制层面进行协议重启时，转发层面保持数据流量不中断，从而提高了网络的可靠性和持续性，满足交付使用标准。

8.2 展望

网络路由协议通过提供对 GR 机制的支持在提高网络可靠性方面取得了很好的效果，在实际 IP 网络中应用前景十分广泛。与此同时，由于时间和环境所限，本文中设计实现的软件系统还存在一些不足之处和改进空间，因此下一阶段的主要工作有：

（1）根据协议规定，网络拓扑变化将会导致 GR 重启失败，强制退出 GR 重启流程。然

而，实际上网络中部分拓扑的变化并不会对整体网络造成太大影响，GR 重启流程可以继续进行下去。因此，可以对 GR 机制作相应改进，限制这种拓扑变化的影响范围，按接口进行细分，一个接口 GR 失败不影响其他接口 GR 流程的继续进行，不用全局退出 GR 重启流程，即可实现接口级 GR。

（2）本文设计实现的 GR 机制中，重启路由器的控制层面必须具有主备环境，备进程平时作为主进程的备份，主要处理配置数据和少量的运行数据，而大部分运行数据都可以在 GR 过程中从邻居获取。基于此，可以考虑不运行备进程，而把配置数据和少量运行数据储存到非易失性的存储介质中，协议重启后再从中读取备份数据来进行系统恢复。这样就可以实现集中式设备的 GR 机制，即无备 GR。

（3）本文主要针对广播网络进行研究实现，后续可以增加对其他网络类型的 GR 机制实现，包括 NBMA 网络、虚连接等。

相信通过对整体系统的不断改进和完善，GR 机制在实际网络中一定能够得到更为广泛的应用和发展。

致　谢

在本论文即将完成之际，我要向我的导师、同事、同学和亲友表示由衷的感谢，感谢他们给予我耐心的指导和无私的帮助。

首先，感谢我的导师胡建萍教授，在这两年半的研究生学习生活中，胡老师一直给予我耐心细致的指导和帮助。在整个毕业论文过程中，胡老师花费了很多时间精力，提出了许多宝贵的意见和建议，使我能够顺利完成毕业论文工作。胡老师严谨治学的学术风格、一丝不苟的工作作风以及宽以待人的生活态度，都是我今后工作和生活中学习的榜样。

其次，感谢实习单位为我提供了良好的工作环境以及公司同事对我的指导和帮助，使我在实习期间学习到了许多有用的知识，提升了自身的专业技能和职业素养，积累了宝贵的工作经验，为我今后走上工作岗位打下了坚实的基础。

此外，感谢实验室的杨军海、陶映旭、于鹏、于少华、陈瑞森、胡敏翔等同学，在研究生阶段相互学习、相互交流、共同进步，陪我一起度过了这段充实而愉快的日子。作为实验室的一员，我感到无比的荣幸和骄傲。

最后，感谢我的父母和亲人，正是他们多年来对我的付出、支持和理解，才使我能够顺利完成学业。

在此，请接受我最真心、最诚挚的谢意！

参 考 文 献

[1] 郭祥本. IP 网络可靠性技术研究与实现[D]. 成都：电子科技大学, 2009：1-2.

[2] 刘端增, 于涌源. 计算机网络可靠性设计[J]. 电脑学习. 2010,24(1)：44-45.

[3] 卢毅. 计算机网络可靠性的分析研究[J]. 合肥师范学院学报. 2009,27(6)：55-56.

[4] 韦乐平. 三网融合的发展与挑战[J]. 现代电信科技. 2010,39(2)：1-5.

[5] 秦东霞. 基于 IPv6 的 OSPFv3 路由协议的原理和发展趋势研究[J]. 网络财富. 2010,23(12)：151-153.

[6] 褚旭辉. IPv6 技术发展及应用前景[J]. 信息与电脑. 2010,21(8)：71.

[7] Sun Jian, Yin YaFang. Research and implement of Ospfv3 in Ipv6 network[J]. Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011,1(6)：743-746.

[8] 黄瑜岳, 梁伟. 基于 IPv6 的 OSPFv3 协议的研究和实现[J]. 常熟理工学院学报. 2006,20(4)：109-112.

[9] 王小龙. 模块化分布式路由器数据平面研究与实现[D]. 北京：北京邮电大学, 2010：1-4.

[10] 孙宗锋, 肖志辉, 孙健. 基于分布式路由器的 IPv4/IPv6 转发控制架构研究[J]. 电信科学. 2012,56(6)：42-46.

[11] Walter Cerronil, Carla Raffaelli, Michele Savi. Optical Router Architecture to Enable Next Generation Network Services[J]. Transparent Optical Networks (ICTON), 2011：1-4.

[12] Li Zimu, Peng Wei, Liu Yujun. An Innovative Ipv4-ipv6 Transition Way for Internet Service Provider [J]. Robotics and Applications (ISRA), 2012：672-675.

[13] 周倩, 邹婷, 蒋胜. IPv6 发展及演进技术[J]. 电信网技术. 2010,35(7)：27-29.

[14] 黄海峰. IPv6 规模商用契机显现[J]. 通信世界. 2012,12(14)：27-29.

[15] Chandra Wijaya. Performance Analysis of Dynamic Routing Protocol EIGRP and OSPF in IPv4 and IPv6 Network[J]. Informatics and Computational Intelligence (ICI), 2011, (10)：355-360.

[16] 唐拥政, 周大为. 基于 IPv6 的路由协议的研究[J]. 盐城工学院学报. 2011,24(1)：55-58.

[17] 梁华. 关于下一代互联网 IPv6 路由协议的研究[J]. 黑龙江科技信息. 2007,10(18)：88-89.

[18] 洪亮. IPv6 路由协议分析[J]. 软件导刊. 2011,10(12)：129-130.

[19] 罗娇. 浅谈网络可靠性设计[J]. 中国科技信息. 2011,37(23)：76-77.

[20] 秦东霞. 基于 IPv6 的 OSPFv3 路由协议的原理和发展趋势研究[J]. 网络财富. 2010,23(23)：151-153.

[21] 刘邦桂, 刘冰. OSPF 动态路由协议的研究[J]. 电脑知识与技术. 2010,6(12)：3285-3286.

[22] 张涛. OSPFv3 和 OSPF 的差异分析[J]. 内蒙古科技与经济. 2006,9(7)：111.

[23] William V.Wollman. Overview of OPEN SHOREST PATH FIRST (OSPF v2) routing in the tactical environment[J]. Military Communications Conference, 1995：925~930.

[24] 贾佳, 张思东, 张宏科. OSPFv3 动态路由协议及其实现[J]. 现代电信科技. 2003,32(8)：19-22.

[25] 关天柱, 吴丰. 基于 IPv6 的 OSPFv3 路由协议的研究[J]. 电脑知识与技术. 2010,6(7)：1557-1558.

[26] John T,Moy. OSPF Version 2[S]. IETF RFC2328, 1998.

[27] 吴许俊，朱长水，王巍. IPv6 网络 OSPFv3 路由协议的研究与仿真[J]. 电子设计工程. 2012,20(13)：71-72.

[28] R. Coltun, D. Ferguson, J. Moy, A. Lindem. OSPF for IPv6[S]. IETF RFC5740, 2008.

[29] A.Zinin, A.Lindem, D.Yeung. Alternative Implementations of OSPF Area Border Routers[S]. IETF RFC3509, 2003.

[30] A. Retana,L. Nguyen,R. White,A. Zinin,D. McPherson. OSPF Stub Router Advertisement[S]. IETF RFC3137, 2001.

[31] P.Murphy. The OSPF Not-So-Stubby Area (NSSA) Option[S]. IETF RFC3101, 2003.

[32] 高国奇，段慧军，周波勇. OSPF 路由技术及其应用[J]. 中国金融电脑. 2008,19(11)：53-54.

[33] 周贵鲁. OSPF 协议中 ISPF 算法及其实现的研究[D]. 南京：南京邮电大学, 2011：16-17.

[34] Jeff Doyle, Henry Benjamin. Routing TCP/IP[M]. Beijing：Posts & Telecom Press, 2009：289-305.

[35] Faraz Shamim, Zaheer Aziz, Johnson Liu, Abe Martey. Troubleshooting IP Routing Protocols[M]. Beijing：Posts & Telecom Press, 2008：309-354.

[36] 杨岚兰. 基于 IPv6 的 OSPF 技术及实现[D]. 四川：四川大学, 2005：33-34.

[37] 刘斌. 基于 IPv6 的 IGP 路由协议分析与应用[D]. 南昌：南昌大学, 2010：15-16.

[38] Jie Wang , Bing Chen, Yuebo Dai, Lijuan Zhou. The Network Topology Discovery System Based On OSPF Protocol[J]. IEEE International Conference on Computer Science and Information Technology, 2011：635~638.

[39] Yuichiro Hei, Tomohiko Ogishi, Shigehiro Ano, Toru Hasegawa. OSPF Failure Identification based on LSA Flooding Analysis[J]. IFIP/IEEE International Symposium, 2007：717~720.

[40] 范伦挺. OSPFv3 路由监控技术的研究与实现[D]. 北京：北京林业大学, 2008：30-36.

[41] J.Moy, P.Pillay-Esnault, A.Lindem. Graceful OSPF Restart[S]. IETF RFC3623, 2003.

[42] 张丹，商云飞，张显峰. 基于 OSPF 的 Graceful Restart 技术的研究与实现[J]. 仪器仪表用户. 2007,14(6)：21-22.

[43] A.Zinin, A.Roy, L.Nguyen, B.Friedman, D.Yeung. OSPF Link-Local Signaling[S]. IETF RFC5613, 2009.

[44] L.Nguyen, A.Roy, A.Zinin. OSPF Out-of-Band Link State Database (LSDB) Resynchronization[S]. IETF RFC4811, 2007.

[45] L.Berger, I.Bryskin, A.Zinin, R.Coltun. The OSPF Opaque LSA Option[S]. IETF RFC5250, 2008.

[46] P.Pillay-Esnault, A.Lindem. OSPFv3 Graceful Restart[S]. IETF RFC5187, 2008.

[47] Neil Matthew, Richard Stones. Beginning Linux Programming 4th Edition[M]. Beijing：Posts & Telecom Press, 2010：394-400.

[48] 宋敬彬，孙海滨. Linux 网络编程[M]. 北京：清华大学出版社, 2010：455-460.

[49] David R.Butenhof，于磊，曾刚. POS1X 多线程程序设计[M]. 北京：中国电力出版社, 2003：124-130.

[50] W.Richard Stevens, Stephen A.Rago. Advanced Programming in the Unix Environment[M]. Beijing：Posts & Telecom Press, 2006：297-312.

[51] 张海藩. 软件工程导论[M]. 北京：清华大学出版社, 2008：212-233.

[52] 叶青. 基于 OSPF 的不间断转发与节能技术的研究与实现[D]. 北京：北京交通大学, 2010：26-29.

[53] 张晓海. OSPF graceful restart 协议的分析与实现[D]. 四川：西南交通大学, 2005：50-55.

[54] 刘昕. 运用 OSPFv3 实现 IPv6 的通信[J]. 微型电脑应用, 2012,28(5)：39-42.

[55] 林川, 施晓秋, 胡波. 网络性能测试与分析[M]. 北京：高等教育出版社, 2009：3-47.

附 录

作者在读期间发表的学术论文及参加的科研项目

学术论文：

[1] 胡建萍，平诞. OSPFv3 协议中平滑重启机制的实现[J]. 计算机系统应用, 2012, 21(11):208-211, 193.