

一体化标识网络下 OSPF 路由协议的实现

马世佳,董平,张宏科

(北京交通大学 电子信息工程学院,北京 100044)

摘要:一体化标识网络与普适服务体系是一种全新的“标识分组网络”体系结构。为了完善一体化标识网络的路由功能,使得 OSPF 路由协议能够支持一体化标识协议栈,文中提出了一体化标识网中 OSPF 路由协议的设计与实现。通过对一体化网络协议栈的分析,给出了一体化网络中 OSPFID 接口以及分离机制的设计思路,然后修改路由协议的代码以及 Linux 内核代码实现了一体化标识网络中 OSPFID 的方案。最后,搭建拓扑对标识网络下 OSPFID 协议进行了测试,验证了 OSPFID 协议能够在一体化标识网络下很好地实现路由功能。

关键词:一体化标识网络;标识协议栈;分离机制;OSPFID

中图分类号:TP393

文献标识码:A

文章编号:1673-629X(2012)11-0022-05

Implementation of OSPF Routing Protocol in Universal Network Protocol

MA Shi-jia, DONG Ping, ZHANG Hong-ke

(School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China)

Abstract: The universal network and pervasive service is a new architecture of the identifier packet network. In order to optimize the routing function of the universal network, making OSPF routing protocol support the universal network, research the design and realization of the OSPF routing protocol in the universal network. By analyzing the universal network protocol stack, give design idea of OSPFID interface and separation scheme in universal network, then modify the routing code and Linux kernel to realize the OSPFID scheme. At last, develop real experiments to validate the OSPFID routing protocol in the universal network and results show that the new routing protocol module can work well on the protocol stack in the universal network.

Key words: universal network; identifier-based protocol stack; separation scheme; OSPFID

0 引言

随着信息技术的快速发展,当今网络开始突显出越来越多的问题。传统网络一般只提供单一业务,对网络建模方式造成了巨大的重复建设和资源浪费,同时由于架构设计等方面的局限性,也很难满足实时性要求高、传输带宽大的服务。为了应对这些挑战,一体化标识网络应运而生,设计出全新的网络体系结构,提出将多种网络设计成一种网络的思想^[1,2]。

一体化标识网络包括核心网、接入网和终端三个部分。核心网和接入网需要通过动态链路协议维持路由标识的可达性。OSPF 协议就是一个典型的动态路由协议,能有效克服距离矢量路由协议存在的缺陷,使

得路由器之间能够正常、快速、实时地交换标识信息。

1 一体化标识网络与 OSPF 协议

在一体化标识网络协议栈中,结构体名称和地址格式发生了改变,从网络的 IP 化转向了网络的标识化,原有的 OSPF 路由协议不能识别这些标识名称,也不能实现一体化网络中的分离映射机制,因此需要在一体化网络下设计 OSPF 协议。

1.1 一体化标识协议栈命名机制

一体化标识协议栈基于 linux-2.6.28 内核开发,提供标准的 BSD Socket API,使得内核协议栈与路由协议有很好的兼容性^[3,4]。

表1 网络协议栈的命名格式

	IPv4	IPv6	IDMP
协议族	AF_INET	AF_INET6	AF_IDMP
地址结构	in_addr	in6_addr	in_id
套接口地址结构	sockaddr_in	sockaddr_in6	sockaddr_id

表1展示了三种不同网络协议栈的命令格式,可以看出一体化网络协议栈的命名特点为^[4]:

收稿日期:2012-03-12;修回日期:2012-06-17

基金项目:中央高校基本科研业务费专项资金(2012JBM010);国家自然科学基金(61100219,60903150)

作者简介:马世佳(1987-),男,硕士研究生,研究方向为交换与路由理论与技术、下一代互联网络;张宏科,博士,教授,博士生导师,研究方向为通信、计算机及信息网络科学。

1、在协议族的属性信息 family 中体现标识协议栈的特点,涉及协议栈属性的信息都使用 IDMP 替代。

2、在协议栈中的地址信息命名都遵循标识化的思想,涉及协议栈的地址信息都使用标识结构替代。

3、在设计命名格式,需要考虑命名的通用性,即设计的名称可以被 IDMP 内核协议识别。

一体化标识网络中 OSPF 需要和协议栈的命名相统一,为了同 IPv4 网络中的 OSPF 路由协议和 IPv6 中的 OSPFv3 路由协议区分,将一体化标识网络下的 OSPF 路由协议命名为 OSPFID,即标识网络中的开放最短路径优先协议(Open Shortest Path First Protocol of Identifier Network)。

1.2 一体化网络中 OSPF 作用

一体化网络基于“身份与位置信息分离”的思想,将网络划分为接入网络 and 核心网络,如图 1 所示^[1,2]。

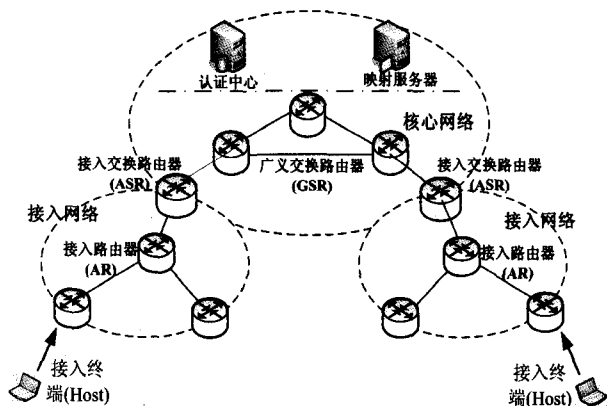


图 1 一体化标识网络模型

接入网络包括接入路由器(Access Router, AR)和接入终端等功能实体,核心网络包括接入交换路由器(Access Switch Router, ASR)和广文交换路由器(General Switch Router, GSR)等交换路由设备以及映射服务器和认证中心等网络管理设备。

一体化网络把传统 IP 地址语义上的双重属性进行解耦,分为接入标识和路由标识^[1,2]。接入标识(Accessing Identifier, AID)在接入网络内使用,代表用户网络中路由器或接入终端的身份信息。路由标识(Routing Identifier, RID)在核心网络内使用,代表核心网络中路由器或终端的位置信息。在接入网和核心网中,需要使用支持标识化的 OSPF 协议维持各自网络信息的正常快速交互。

同时,核心网中的 ASR 是完成分离映射的实体,主要功能是把数据包中的源和目的 AID 映射为对应的 RID,再将映射后的数据包转发发出。基于接入网与核心网分离的思想,AID 的身份信息不能扩散到核心网中,RID 的位置信息也不能扩散到接入网中,这就需要使得标识化的 OSPF 协议能够隔离接入网和核心网

的路由信息,即支持分离映射机制。

2 一体化网络中 OSPFID 设计

一体化标识网络下 OSPFID 路由协议设计框架如图 2 所示^[5-7],图中路由管理模块负责上层路由协议和一体化内核协议栈 IDMP 之间的通信。上层的标识路由协议计算和创建标识化的路由表,并向内核协议栈中的转发表进行写操作,上层的路由表也完成从内核协议栈中的转发表的读操作。路由信息在相邻路由器之间传递,确保路由器相互之间的可达性^[7]。

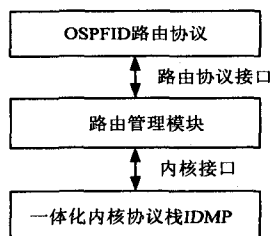


图 2 一体化标识网络 OSPFID 设计框架

在一体化网络中,实现的是 128 位的标识化网络,可以通过对 IPv6 下的 OSPF 进行移植,来实现一体化标识网络的 OSPF 路由协议^[8,9]。基于一体化标识协议栈的 OSPFID 路由协议增加了 AID 和 RID 接口配置模块、分离映射模块以及路由协议接口模块。

2.1 OSPFID 接口设计

在一体化网络中,有 AID 和 RID 两种标识,在接口配置的时候,需要对接口类型进行区分,分为接入口和路由口。接入口表示连接的是接入网,路由口表示连接的是核心网。通过在代表 OSPFID 接口类型结构体 ospfid_interface 中增加了 inftype 字段,表示接口类型。当用户需要改变接口类型的时候,经过一系列判断如果满足要求,则调用钩子函数用来更新 LSA(链路状态通告)。OSPFID 接口设计流程如图 3 所示:

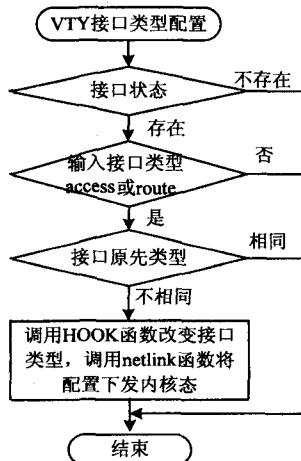


图 3 接口设计流程

2.2 分离机制的设计

用户可以通过同时启动两个不同的路由进程分别

维护接入网和核心网的路由信息,但这样带来了额外的管理负担,消耗更多的设备资源,并且当配置出错时,很容易造成路由信息的泄漏。因此从路由协议的机制入手,才能更好地解决分离映射的问题^[5,7,8,10]。

基于以下四点进行分析:

(1) OSPF 属于链路状态路由协议,相邻路由器并不直接交互各自的路由表,而是通过泛洪机制交互各自学到的 LSA,再根据得到的 LSA 信息计算出全局路由表。因此可以在边界路由器上对链路状态更新消息和泛洪消息加上过滤机制实现路由信息的分离。

(2) 如图 1 所示,在一体化标识网络中,路由口所连的邻居接口一定是路由口,接入口所连的邻居接口一定是接入口,因此在接收 LSA 的时候不用根据邻居类型进行判断是否应该接收以及安装该 LSA。

(3) 在接入网和核心网内部只有一种接口类型,也不需要进行 LSA 过滤,只有在核心网和接入网边界的 ASR 需要进行 LSA 条目的过滤。

(4) 在 OSPF 中,链路状态数据库交互模块完成 LSA 的交互、更新以及同步,通过对链路状态数据库交互模块的修改实现 LSA 的过滤。

链路数据库的交互是通过一种称为泛洪的机制实现,泛洪的过程分为 2 个阶段^[5]。

第一阶段对收到的 LSA 进行处理,判断是否应该将收到的 LSA 安装到自己的数据库中。对收到的每一条 LSA,进行如下步骤处理:

(1) 检测 LSA 是否有效,包含校验和的检查、LS 类别检查和域的检查:如果检验发生错误,则丢弃该 LSA,结束程序;

(2) 检查数据库中无实例、LSU 中的 LSA 的 LS Age 为 MaxAge 且没有邻居处于 Exchange 或 Loading 状态;

(3) 检查数据库中没有实例或者收到的 LSA 较新,收到的 LSA 和数据库中的实例一样新,数据库中的实例较新^[11,12]。

泛洪的第二阶段是将安装到自己的数据库中的 LSA 泛洪到整个路由域的过程。主要工作就是逐一检查路由器的每一个接口,并判断这个接口的类型,来决定收到的 LSA 要从哪些接口泛洪出去。

(1) 检查该 LSA 是否加入到任何邻居重传列表,如果没有加入,则结束程序;

(2) 检查该 LSA 是否从该接口的邻居接口注入且来自 DR 或者 BDR,如果是,则结束程序;

(3) 实现分离机制,进行 LSA 接口类型判断,将那些来自或产生于接入口的 LSA,不向路由口泛洪。实现时使用一个循环,以接口为循环变量,在每一个循环中,对每一个接口分别进行处理。在对每一个接口的

处理中,对接口的每一位邻居进行检查,判断邻居所属接口类型。如果邻居的类型和收到或者产生 LSA 接口类型相同,就要将该 LSA 从邻居接口上泛洪出去。如果类型不同,则不去泛洪该 LSA。

由于 ASR 对需要泛洪的 LSA 进行了过滤,ASR 不能向核心网或者接入网泛洪它收到的所有路由信息,也不能够完成相邻数据库的同步以及达到 FULL 状态,因此需要在 ASR 上修改链路状态描述报文。当 ASR 和它相邻路由器通告其的链路状态信息时,并不进行全局通告,而只是通告属于邻居网络信息的部分。

3 一体化网络中 OSPFID 实现

3.1 接口实现

(1) 实现 vtysh 配置函数如下所示,其中 idmp_ospfid_interface_type 代表 vtysh 输入的级别,即在 interface 级别。

```
DEFUN ( idmp_ospfid_interface_type,
        idmp_ospfid_interface_type_cmd,
        " idmp ospfid interface type ",
        IDMP_STR
        OSPFID_STR
        " Interface type\n"
        " route or access\n" )
```

(2) 实现判断接口状态函数,首先从 vty 中读入的接口 index 找到对应的接口实体,判断实体是否存在,如果不存在则创建该接口,接下来判断读入接口类型。

```
ifp = (struct interface *) vty->index;
assert (ifp);
o6i = (struct ospfid_interface *) ifp->info;
if (! o6i)
o6i = ospfid_interface_create (ifp);
assert (o6i);
if (argv[0] == "access") .....
```

(3) 调用钩子函数,实现接口的更新以及 LSA 的更新。具体涉及的函数:

CALL_CHANGE_HOOK (&interface_hook, o6i);

用于更新路由器 LSA、网络 LSA、内部 LSA。

CALL_FOREACH_LSA_HOOK (hook_interface, hook_change, o6i);

用于更新链路 LSA。

3.2 分离机制的实现

● 泛洪第一阶段涉及的函数如下:

ospfid_dbex_receive_lsa (lsa_header, o6n);

接收 LSA 并进行一系列检测,其中 o6n 代表该 LSA 从那个邻居接口收到。

● 泛洪第二阶段涉及的函数如下:

(1) ospfid_dbex_flood_linklocal (lsa, ospfid_inter-

face, from);

其中,lsa 代表要泛洪的 LSA 条目,ospfid_interface 代表 LSA 将要泛洪出去的接口,from 代表该 LSA 从哪个接口接收到。

(2) ospfid_dbex_flood_area (lsa, o6a, from);

其中,o6a 代表 LSA 将要泛洪出去的区域。

(3) ospfid_dbex_flood_as (lsa, o6, from);

其中,o6 代表 LSA 将要向哪个自治域泛洪。

(4) ospfid_dbex_flood (lsa, from)。

前三个函数统一由函数 4 调用,在该函数中,根据 LSA 作用范围的不同选择特定函数进行泛洪。

●实现分离机制修改的函数如下:

(1)修改 ospfid_dbex_flood_linklocal 函数,判断该 lsa 收到的函数接口与将要发送的接口类型是否一致。即判断 ospfid_interface->inftype 是否等于 from->ospfid_interface->inftype,如果相等才进行泛洪。

(2)将本地产生的 LSA 强制关联到一个接口。OSPF 调用如下函数实现产生本地 LSA 的,包括产生路由器 LSA、网络 LSA、域间前缀 LSA 等等。

ospfid_lsa_originate (type, id, adv_router, data, data_len, scope);

其中,type 表示本地 LSA 类型,id 表示链路状态 id,adv_router 表示产生该 LSA 的路由器,date 表示该 LSA 的信息,scope 表示该 LSA 的有效范围。该函数调用 ospfid_dbex_flood (lsa, NULL)实现将该 LSA 泛洪。由于该 LSA 是本地产生的,from 为 NULL。通过修改 ospfid_lsa_originate 函数,将这个函数传入一个接口信息,并在这个函数中定义一个 ospfid_neighbor 变量,将其 ospfid_neighbour->ospfid_interface 赋值,传递给 ospfid_dbex_flood 函数。

(3)修改链路状态描述报文。

由于 ASR 实现了分离机制,相互路由器不能获得所有的路由信息,因而不能够达到 FULL 状态。为了让路由器之间仍能达到 FULL 状态,需要修改链路状态描述数据库。涉及到的函数如下:

① ospfid_send_dbdesc_rxmt (struct thread * thread);

该函数定义为一个线程函数,其功能是用于从某接口上重发 DD(数据库描述)包。

② ospfid_message_send (OSPFID_MESSAGE_TYPE_DBDESC, message, &o6n->hisaddr, o6n->ospfid_interface->interface->ifindex);

函数 1 遍历其所有邻居,调用函数 2 将链路状态数据库中 LSA 的头部信息发送给邻居,其中 o6n->ospfid_interface->interface 代表邻居接口。

③ ospfid_lsdb_install (lsa);

每当路由器产生或者收到一条 LSA 后,调用上述函数安装到其链路状态数据库。此时对 LSA 进行分类安装,在 ASR 上保存 2 个链路状态数据库,这两个分别用来存储接入网 LSA 和核心网 LSA 信息。当函数 1 调用函数 2 发送其链路状态描述报文时,通过判断 o6n->ospfid_interface->inftype 类型,选择相应的链路状态数据库进行发送。

3.3 路由协议接口模块实现

路由管理模块能够实现 OSPFID 路由协议和一体化网络协议栈相互通信,完成下发路由信息、删除路由信息等操作。通过多种通信操作保证用户空间和内核空间信息传递,包括:ioctl 操作获取接口信息、sysctl 操作检查路由表和接口清单、proc 操作获取内核返回的数据、socket 操作与内核建立套接口^[4,5]。

netlink 套接口提供一种异步双向通信机制,实现增加路由条目、删除路由条目、查询路由条目等功能^[3,4]。通过下面的函数创建一个与标识协议栈中 netlink 通信套接口:

sock_fd = socket (family, socket_type, netlink_family);

family 参数使用 AF_NETLINK 协议族,socket_type (套接口类型) 参数可包括 SOCK_RAW 和 SOCK_DGRAM,netlink_family 参数使用一体化内核协议栈定义的 NETLINK_IDMP_IDMAP 类型,这个类型用于一体化内核协议栈中分离映射内核模块和用户层的通信。并定义了一些列路由操作属性,如下:

#define RTM_IDMP_NEWROUTE (RTM_BASE+8)

#define RTM_IDMP_DELRROUTE (RTM_BASE+9)

#define RTM_IDMP_GETROUTE (RTM_BASE + 10)

分别代表了增加、删除以及获取 IDMP 路由。通过上述方案的设计,实现了一体化标识网络下 OSPFID 路由协议。

4 功能测试

测试拓扑结构及接口配置如图 4 所示:

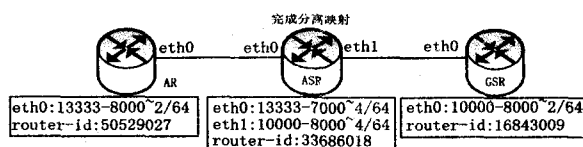


图 4 OSPFID 功能测试拓扑结构

实验结果如下:

图 5 展示了 GSR 的状态信息,包括:接口配置、路由配置,运行状态。为了测试分离机制的实现,在路由配置中使用 idmp route 命令向 GSR 注入了 2 条外部路由信息(用黑线标注)。通过运行状态看到此时 GSR

和 ASR 已经达到了 Full 状态,即已经完成了数据库同步的工作,33686018 和 16843009 分别为 ASR 和 GSR 的 router-id。

```

          接口配置信息
[root@localhost idmp-ripidd]# idconfig
  lo ID-> 1                                prefix 128
  eth0 ID-> 10000-8000~2                    prefix 64
  eth0 ID-> 65152-549-46079-65108-47411    prefix 64
          路由配置信息
GSR# show running-config
Current configuration:
hostname GSR
!
idmp route 10201-300~/64 eth0
idmp route 11111~/64 eth0
!
interface eth0
  idmp ospfid interface type is route
!
router ospfid
  router-id 16843009
  interface eth0 area 0
!
          路由状态信息
GSR# show idmp ospfid neighbor
RouterID  State/Duration  DR          BDR          I/F[State]
33686018  Full/00:01:43          33686018    16843009     eth0[BDR]

```

图 5 GSR 的接口配置以及路由配置信息

图 6 使用 show idmp route 命令展示了 GSR、AR 上的路由表条目,由于在 ASR 进行了分离机制使得接入网络信息不能扩散到核心网络,即接入网的路由信息 13333-7000 ~ /64 信息不会泛洪到 GSR 上,核心网路

```

          GSR上的路由信息
GSR# show idmp route
Codes: K - kernel route, C - connected, S - static,
       R - RIPid, O - OSPfid, B - BGP, * - FIB route.

C>* 1/128 is directly connected, lo
O 10000-8000~/64 [110/0] is directly connected, eth0
C>* 10000-8000~/64 is directly connected, eth0
S>* 10201-300~/64 [1/0] is directly connected, eth0
S>* 11111~/64 [1/0] is directly connected, eth0
C * 65152~/64 is directly connected, eth0
          AR上的路由信息
AR# show idmp route
Codes: K - kernel route, C - connected, S - static,
       R - RIPid, O - OSPfid, B - BGP, * - FIB route.

C>* 1/128 is directly connected, lo
O 13333-7000~/64 [110/0] is directly connected, eth0
C>* 13333-7000~/64 is directly connected, eth0
C>* 65152~/64 is directly connected, eth0

```

图 6 三台路由器的路由信息

由信息 10000-8000 ~ /64 以及 2 条外部路由不会泛洪到 AR 上。通过这种机制减少了核心网的路由条目,能够实现快速查询以及转发。

5 结束语

通过对一体化网络下 OSPF 路由协议的设计和实现,完善了一体化网络的路由功能。测试结果表明 OSPFID 已经能够支持标识协议栈,并且能够正常地完成路由功能,具有良好的稳定性及可靠性,能够适合大规模网络的部署。

参考文献:

- [1] 张宏科,苏伟.新网络体系基础研究——一体化网络与普适服务[J].电子学报,2007,35(4):593-598.
- [2] 杨冬,周华春,张宏科.基于一体化网络的普适服务研究[J].电子学报,2007,35(4):607-613.
- [3] 李玮.一体化网络标识网络协议栈设计[D].北京:北京交通大学,2009.
- [4] 姚苏.接入网标识路由协议用户层的设计与实现[D].北京:北京交通大学,2010.
- [5] 张宏科,苏伟.路由器原理与技术[M].北京:高等教育出版社,2010.
- [6] 张宏科,苏伟.IPv6 路由协议栈原理与技术[M].北京:邮电大学出版社,2006.
- [7] Coltun R, Ferguson D. OSPF for IPv6[S]. RFC 5340, 2008.
- [8] Deering S, Hinden R. Internet Protocol Version 6 (IPv6) Address Architecture[S]. RFC3513, 2006.
- [9] Moy J. OSPF Version 2[S]. RFC 2328, 1998.
- [10] Nan Yao. Design and Implementation of Routing Protocol Extensions Supporting Separation of the Core and Access Network[J]. Journal of Internet Technology, 2008, 9(5): 355-360.
- [11] 李建昊,王志克,张春青,等. OSPFv3 详细设计[M]. 北京:北京交通大学,2003.
- [12] 王江林. OSPFv3 概要设计[M]. 北京:北京交通大学,2002.

(上接第 21 页)

- [6] 李雷,罗红旗,丁亚丽.一种改进的模糊 C-均值聚类算法[J].计算机技术与发展,2009,19(12):71-73.
- [7] Mangalampalli A, Pudi V. Fuzzy Association Rule Mining Algorithm for Fast and Efficient Performance on Very Large Datasets[C]//IEEE International Conference on Fuzzy System. [s.l.]:[s.n.], 2009:20-24.
- [8] Lee Y C, Hong T P, Wang T C. Mining Fuzzy Multiple-level Association Rules under Multiple Minimum Supports[C]//Proc. of the 2006 IEEE International Conference on Systems, Man and Cybernetics. [s.l.]:[s.n.], 2006:4112-4117.
- [9] 罗军生,李永忠.基于模糊 C-均值聚类算法的入侵检测[J].计算机技术与发展,2008,18(1):178-180.
- [10] 吴瑛,王秋生.模糊 C-均值聚类算法在 web 使用挖掘上的应用研究[J].计算机技术与发展,2008,18(6):32-35.
- [11] Wu Zhenglong, Xiong Fanlun, Teng Minggui. Mining Fuzzy Association Rules for Numerical Attributes Based on Fuzzy Clustering[J]. MINI-MICRO SYSTEMS, 2004, 25(7): 1295-1297.
- [12] Gyenesei A. A Fuzzy Approach for Mining Quantitative Association Rules[R]. [s.l.]:[s.n.], 2001.