**ospfd代码简要剖析**

ospfd的代码就是ospf的rfc的翻译,在读代码之前务必先连接ospf协议的相关内容.

* **ospfd文件说明**

|  |  |
| --- | --- |
| 文件名 | 说明 |
| ospf\_abr.c/h | abr相关事宜 |
| ospf\_absr.c/h | asbr相关事宜 |
| ospf\_flood.c/h | ospf的lsa的洪泛都在这里实现 |
| ospf\_interface.c/h | ospf的接口相关函数 |
| ospf\_ism.c/h | ospf的接口状态机转换 |
| ospf\_lsa.c/h | lsa结构的定义,存储,操作 |
| ospf\_neigbor.c/h | 邻居相关的操作,定义 |
| ospf\_network.c/h | 网络相关操作,如socket的初始化等. |
| ospf\_nsm.c/h | 邻居状态机的实现 |
| ospf\_spf.c/h | 最短生成树,路由的生成 |
| ospf\_te.c/h | 流量工程相关,不用管 |
| ospf\_snmp.c/h | snmp相关,不用管 |
| ospf\_opaque.c/h | 这种lsa比较少见,不管 |

* **ospf状态机入口**

ospfd的代码走向主要要关注两个方面,一个是ospfd的收包,这个非常重要,因为收到的包决定了邻居状态机的转换,lsa的更新,从而触发最短路径的重新计算,路由的重新生成.这里主要是ospf\_read函数.

第二个是ospfd的发包.ospf协议并没有基于tcp,并不是非常可靠,但是区域内各个节点维护一致的链路状态数据库对于ospf至关重要,因此,它设计了一套重传机制,要保证每一个更新包都被邻居接收到了.这个多由状态机和定时器推动.

* **ospf定时器**

struct ospf的相关定时器:

|  |  |  |
| --- | --- | --- |
| 定时器名称 | 函数 | 用处 |
| t\_maxage\_walker | ospf\_new | 检查到达老化时间的lsa |
| t\_lsa\_refresher | ospf\_new | 重新在区域内或者自治系统内部洪泛lsa |

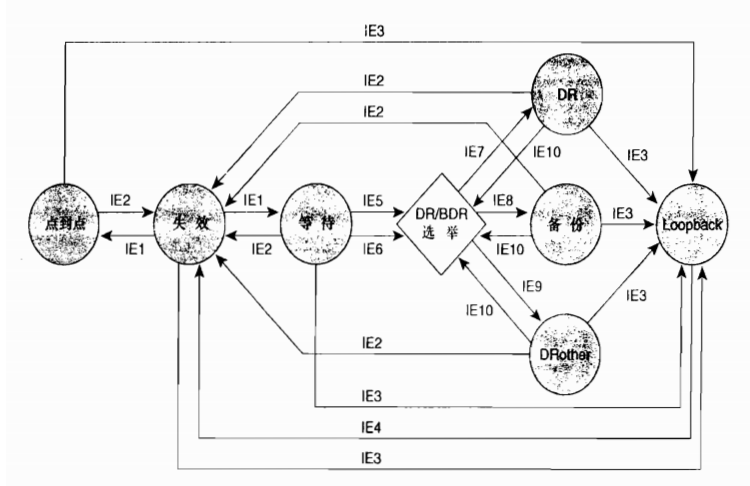
struct ospf\_neighbor相关的定时器:

|  |  |  |
| --- | --- | --- |
| 定时器名称 | 函数 | 说明 |
| t\_inactivity | nsm\_packet\_received | 收到此邻居的包,自动重新开始计时,一旦超时,表示此邻居不再活跃,这个实际就是DeadInterval |
| t\_db\_desc | nsm\_timer\_set | 主要用于重传DBD包 |
| t\_ls\_req | ospf\_ls\_req\_timer  ospf\_ls\_req\_event | 主要用于发送链路状态请求包,链路状态请求链表为空,定时器关闭 |
| t\_ls\_upd | nsm\_timer\_set | 用于重传链路状态更新包,为了保证区域内lsa数据的一致 |
| t\_hello\_reply | ospf\_hello | 用于定时向邻居发送hello报文 |

struct ospf\_area相关的定时器:

|  |  |  |
| --- | --- | --- |
| 定时器名称 | 函数 | 说明 |
| t\_stub\_router | ospf\_stub\_router\_check | 用于判断区域是否为末梢区域 |
| t\_opaque\_lsa\_self | 暂不关心 |  |

* **接口状态机**

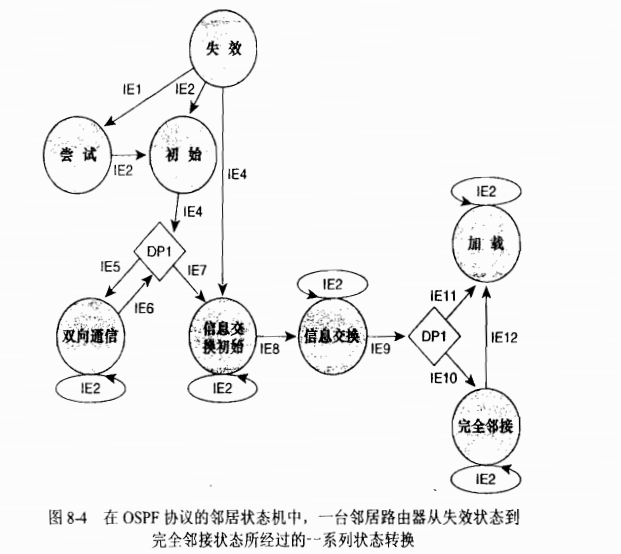


相关事件说明:





* **邻居状态机**



一旦超时,也就是E3事件发生,无论之前处于什么状态,都会转入失效状态.

