拓扑消息控制洪泛

拓扑控制（TC ）消息的洪泛具有十分重要的意义。在使用 OLSR 协议的网络中，每个节点都周期性发送*TC* 分组。当一个节点接收到*TC* 消息时，就进入拓 扑信息维护模块。收到*TC* 消息必要时进行转发(实现拓扑泛洪);如果得到网络 中一条有效的链路(拓扑)，则将其添加到拓扑表中，用以计算路由。当检测到 拓扑表发生变化时，就要转到路由建立与维护模块，重新计算路由。当收到数据包时，对其进行转发。

MSSN (16bits)	*Reserved (16bits)*
MPR Selector Address (32bits)	
MPR Selector Address (32bits)	

TC 消息数据包头部结构

MSSN : *MPR Selector* 序列号。与多点中继 *MPR Selector* 集相对应的序列号， 每当节点检测到 *MPR Selector* 集发生变化时，就增加该序列号的值。节点收到 *TC* 分组时，根据 *MSSN* ，决定有关发送者的 *MPR Selector* 的信息是否比已有的要新。
Reserved :保留字段，设置为“00000000000000000000000000”。
MPR Selector Address :多点中继选择节点的地址。包含的是产生该*TC* 分组的节点的 MPR Selector 的地址。

- **TC数据结构**

```
119  /* deserialized LQ_TC */
120  struct lq_tc_message {
121    struct olsr_common comm;
122    union olsr_ip_addr from;
123    uint16_t ansn;
124    struct tc_mpr_addr *neigh;
125  };
126
127  /* serialized LQ_TC */
128
129  struct lq_tc_header {
130    uint16_t ansn;
131    uint8_t lower_border;
132    uint8_t upper_border;
133  };
```

120-125: *lq _ tc _ message* 是封装后的拓扑数据包格式。 *from* :到达目的 地的倒数第二跳地址;*ansn* :记录本节点收到的最近一个*TC* 分组的 *ANSN* 序列 号。当收到一个新的*TC* 分组时，将新的*TC* 分组的 *ANSN* 号与拓扑表中的相对应 的 *ANSN* 序列号比较，以此判断接收还是丢弃该消息。 *neigh* :指向广播邻居集 的地址结构。
127-131: *lq _ tc _ header* 是数据包的头部。 *ansn* :记录本节点收到的最近 一个 *TC* 分组的 *ANSN* 序列号。*lower _ border* 表示下一级的边界，*upper _ border* 表示上一级的边界。

```
77  struct tc_message {
78    olsr_reltime vtime;
79    union olsr_ip_addr source_addr;
80    union olsr_ip_addr originator;
81    uint16_t packet_seq_number;
82    uint8_t hop_count;
83    uint8_t ttl;
84    uint16_t ansn;
85    struct tc_mpr_addr *multipoint_relay_selector_address;
86  };
```

77-86: *tc _ message* 是 *TC* 消息数据包格式。 *OLSR* 路由协议利用拓扑表记 录接收的*TC* 消息内容。拓扑表包含四个部分:目的地址，到达目的地的倒数第 二跳地址， *ANSN* (*Advertised Neighbor Sequence Number*) 序列号和表项有效时 间。 *TC* 分组仅仅包含 *MPR selector*（将本节点选为 *MPR* 节点的邻居节点）的地 址，而不是所有邻居节点的地址。因为 *TC* 分组数据包必须通过 *MPR* 节点被广播到全网中，用以维护网络的拓扑信息、确保链路时刻的连通状态和更新路由表集。

- **TC 消息的生成**

```
81  void
82  generate_tc(void *p)
83  {
84    struct tc_message tcpacket;
85    struct interface *ifn = (struct interface *)p;
86
87    olsr_build_tc_packet(&tcpacket);
88
89    if (queue_tc(&tcpacket, ifn) && TIMED_OUT(ifn->fwdtimer)) {
90      set_buffer_timer(ifn);
91    }
```

84-90:*MID* 消息通过 *olsr _ build _ tc _ packet* ( ) 函数生成之后放在 MID 队 列中。当时间戳期满的时候，调用*set_buffer_timer*( )设置定时器。最后从给定 接口*ifn*释放消息，同时调用*olsr_ free_tc_ packet*( )释放所占内存。

为了构建拓扑信息库，每个被选择为 *MPR* 的节点必须广播拓扑控制 (*TC*) 消息，这些通过*TC* 消息扩散到网络中的信息将有所帮助每个节点计算其路由表。 并且 *TC* 消息必须是根据“默认转发算法”进行转发的。 *MPRs* 使得拓扑信息的 分布具有更好的可扩展性。

当节点的通告链路集变为空时，该节点应当在等于其先前发送的*TC* 消息的 “有效时间”的持续时间期间仍然发送(空) *TC* 消息，以便使先前的 *TC* 消息 无效，直到有节点加入到通告链路集。

节点可以传送附加的 *TC* 消息以增加其链接故障的反应性。当检测到对 *MPR* 选择器集合发生改变并且这种改变可以归因于链路故障时，*TC* 消息应当在短于 *TC _ INTERVAL* 的时间间隔内被发送。

- **TC 消息处理**

```
809   /* We are only interested in TC message types. */
810   pkt_get_u8(&curr, &type);
811   if ((type != LQ_TC_MESSAGE) && (type != TC_MESSAGE)) {
812     return false;
813   }
814
815   /*
816    * If the sender interface (NB: not originator) of this message
817    * is not in the symmetric 1-hop neighborhood of this node, the
818    * message MUST be discarded.
819    */
820   if (check_neighbor_link(from_addr) != SYM_LINK) {
821     OLSR_PRINTF(2, "Received TC from NON SYM neighbor %s\n", olsr_ip_to_string(&buf, from_addr));
822     return false;
823   }
```

810-813:当节点接收到*TC* 消息时，只关心其消息类型。当其类型不等于 *LQ _ TC _ MESSAGE* 或者 *TC _ MESSAGE* 时，直接返回 *false* ，将包丢弃。
814-818:*TC* 消息接收者在接收到消息时判断发送者接口信息，若发送者 并非是对称一跳邻居，那么该包将会被丢弃。
819-822:一旦接收到 *TC* 消息，必须根据消息头的 *Vtime* 字段计算“有效 时间” 。

```
847     if (olsr_seq_inrange_high((int)tc->msg_seq - TC_SEQNO_WINDOW, tc->msg_seq, msg_seq)
848         && olsr_seq_inrange_high((int)tc->ansn - TC_ANSN_WINDOW, tc->ansn, ansn)) {
849
850       /*
851        * Ignore already seen seq/ansn values (small window for mesh memory)
852        */
853       if ((tc->msg_seq == msg_seq) || (tc->ignored++ < 32)) {
854         return false;
855       }
```

851-854:如果该消息中的 *msg _ seq* 和外部变量 *msg _ seq* 相等且 *ignored* 变量小于 32，则代表该消息已经处理过，应该忽视。

```
878   /*
879    * Generate a new tc_entry in the lsdb and store the sequence number.
880    */
881   if (!tc) {
882     tc = olsr_add_tc_entry(&originator);
883   }
884
885   /*
886    * Update the tc entry.
887    */
888   tc->msg_hops = msg_hops;
889   tc->msg_seq = msg_seq;
890   tc->ansn = ansn;
891   tc->ignored = 0;
892   tc->err_seq_valid = false;
```

881-892:如果拓扑表中不存在和*TC* 消息中‘消息产生这地址’字段相同 的条目，则添加新的条目并且保存序列号，之后根据之后获取的*TC* 消息数据包 的头部信息更新 *tc _ entry* 。

```
914   /*
915    * Calculate real border IPs.
916    */
917   if (borderSet) {
918     borderSet = olsr_calculate_tc_border(lower_border, &lower_border_ip, upper_border, &upper_border_ip);
919   }
920
921   /*
922    * Set or change the expiration timer accordingly.
923    */
924   olsr_set_timer(&tc->validity_timer, vtime, OLSR_TC_VTIME_JITTER, OLSR_TIMER_ONESHOT, &olsr_expire_tc_entry,
          tc,
925                  tc_validity_timer_cookie);
926
927   if (emptyTC && lower_border == 0xff && upper_border == 0xff) {
928     /* handle empty TC with border flags 0xff */
929     memset(&lower_border_ip, 0x00, sizeof(lower_border_ip));
930     memset(&upper_border_ip, 0xff, sizeof(upper_border_ip));
931     borderSet = 1;
932   }
```

918-926:调用olsr_calculate_tc_border ()计算borderSet的值，并且重置相关的期满定时器。如果*TC* 消息中的‘消息产生者地址’字段是自己的 *MPR selector* ，并且‘消息存活时间’大于 0，则向周围邻居结点广播该 *TC* 消息。

• 路由计算

节点通过*TC* 消息的扩散获得全网拓扑图，再根据邻居表、两跳邻居表和拓 扑表，独立地按照 *Dijkstra* 算法计算出路由表。每个节点都有一张路由表，通过 路由表寻找路径信息。对于路由已知的网络中的每个目的地，将路由信息记录在 路由表中。所有路由被破坏或仅部分已知的目的地的路由信息不被记录在表中。

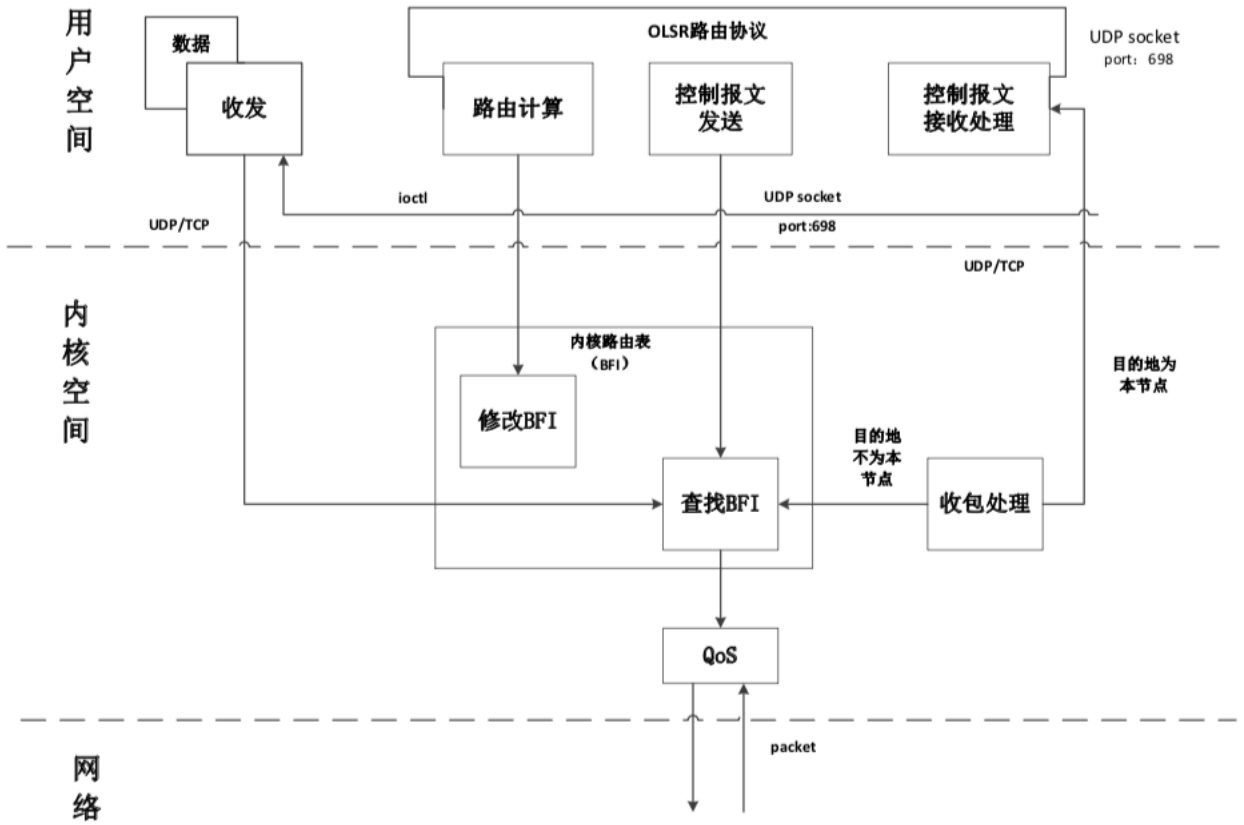
每一条路由信息都包含信息目的地址、下一跳地址、总跳数、下一跳接口地址，格式如下:

R _ dest _ addr	*R _ next _ addr*	*R _ dist*	*R _ iface _ addr*

如果该节点所维护的这些信息表中的任何信息发生改变，则重新计算路由表 以更新关于网络中的每个目的地的路由信息。即更新路由表的条件为:

(1) 邻居表变化，需要重新选择路由，根据路由计算结果更新路由表。(2) 两跳邻居表变化，需要重新选择路由，根据路由计算结果更新路由表。(3) 拓扑表发生变化，需要重新选择路由，根据路由计算结果更新路由表。 路由表的更新既不在网络中，也不在一跳邻居域中生成或触发任何消息。 操作系统的路由体系结构按功能可以分成两个部分。一部分是负责与其它节点交换信息，计算到其它节点的正确路由，称之为“路由功能模块”;一部分则 是根据内核路由表，将需要发送到网络中的数据分组，通过正确的网络接口发送 到下一跳节点，称为“转发功能模块”。这样，操作系统就可以在“转发功能模 块”保持不变的情况下，通过修改“路由功能模块”，从而实现不同的路由协议。

OLSR 协议的实现通过端口号为 698 的*UDP* 端口收发路由控制分组，然后 维护邻居表，进行逻辑计算，最后生成路由表并反映到内核路由表中。数据分组 和协议控制分组按照内核路由表中的最佳匹配表项进行发送和转发。当网络中有分组到达本节点时，内核将判断该分组的目的地是否是自己，如果不是，则“转 发功能模块”根据内核路由表转发该分组;如果是，则根据分组的不同交给相应的模块进行处理，当收到 *OLSR* 协议控制分组时，转由 *OLSR* 路由协议模块处理。


用户空间
数据
收发
OLSR路由协议
路由计算
控制报文发送
控制报文接收处理
UDP socket
port：698
ioctl
UDP socket
port:698
UDP/TCP
UDP/TCP
内核空间
内核路由表
（BFI）
修改BFI
目的地为本节点
目的地不为本节点
查找BFI
收包处理
QoS
packet
网络