RSVP と帯域制御：インターナルズ

IP Meeting 97 チュートリアル

長 健二朗

ソニーコンピュータサイエンス研究所

kjc@csl.sony.co.jp

97 年 12 月 17 日

1 はじめに

1.1 チュートリアルの目的

- RSVP の全体像を理解する
- RSVP の RFC や ISI リリースのコードを読むために必要な知識を得る
- トラフィック制御部との関連を理解する

1.2 RSVP の全体像はどうして解りづらいか？

RSVP を理解するには

- マルチキャスト
- ルーティング
- トラフィック制御
- トラフィック・パラメータの定義
- カーネルインターフェイス
- セキュリティ

等の広い知識が必要である。

これらのモジュールはRSVP と深く関連しているが、RSVP とは独立に定義あるいは実装されているため全体像がつかみにくい。

1.3 RSVP の実装はどうして複雑なのか？

- ルーテイングやネットワークインターフェイスの制御等、特殊な操作が多い
- 各プラットフォームでこれらのインターフェイスが異なる
- IPv6 対応でさらに複雑に、、、

2 RSVP の概要

2.1 RSVP の特徴

- RSVP は資源予約プロトコル。ユニキャストとマルチキャストの資源予約を行なう
- シンプレックス （一方向のデータフローに対する予約を行なう）
- レシーバ主導 （レシーバが予約を起動する）
- ソフトステート (マルチキャストのメンバ変更、経路変更に動的に対応)
- ルーティングプロトコルではない（ルーティング情報をもらって利用する）
- QoS は独立のトラフィック制御モジュールで実現
  (クラシファイア, アドミッション制御, パケットスケジューラを想定)
- トラフィック制御とポリシ制御のパラメータを転送、管理するがその内容は RSVP の知るところではない
- 複数の資源予約モデル（スタイル）を提供
  (Fixed-Filter, Shared-Explicit, Wildcard-Filter)
- RSVP をサポートしないルータが途中にあっても動作する
- IPv4 と IPv6 で動作する
- 予約要求はトラフィック制御モジュールとポリシ制御モジュールに渡され、許可の判断が行なわれる

2.2 RSVP のステータス

2.2.1 標準化

- 1997 年 9 月に RSVP 関連の RFC がまとめて承認された
  - RSVP-WG: RSVP プロトコルの仕様
  - IntServ-WG: トラフィックパラメータの定義
    (RSVP 以外のプロトコルも考慮)

2.2.2 ISI による RSVP デーモン実装

- 最新バージョン: rel4.2a1 (Sept 1997)
- platform: SunOS 4.x, FreeBSD, Solaris 2.5, IRIX 5.3/6.2, Digital Unix 3.0
- 新機能： IPv6、Diagnostic Message サポート
- ISI の実装に含まれないもの
  - トラフィック制御
  - ポリシ制御
  - RSVP MIB
  - マルチキャスト・トネル
  - フラグメンテーション
  - 可変リフレッシュ・タイマ
- 部分的にしか実装されていないもの
  - IPSEC 拡張
  - INTEGRITY

3 RSVP のモデル

3.1 RSVP 動作の概要

3.1.1 Path メッセージ

Sender: Path メッセージを送る
　　Sender Tspec: オリジナルのフロー情報
　　Adspec: 中間ルータによって書き換えられるフロー情報
Router: Adspec を変更しながら下流に転送

3.1.2 Resv メッセージ

Receiver: 受信している Path メッセージに対応して、
　　Resv メッセージを送って予約をする
　　Flowspec: 予約の QoS 指定
　　Tspec: （予約要求 Tspec）
　　Rspec: （Guaranteed QoS の指定）
　　Filterspec: フローの指定 (e.g., sender addr/port)
Router: Resv メッセージにしたがって予約を確保し、上流に転送
　　予約は可能であればマージする

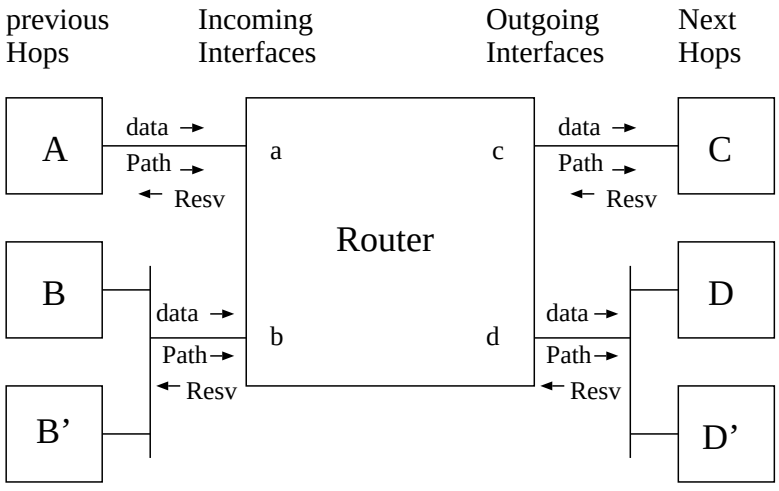

図 1: RSVP ルータ

3.1.3 トラフィックパラメタ

- トークンバケットによる Tspec (RFC 2215)
- RSVP に対するサービスを規定 (RFC 2210)
- Controlled-Load(RFC 2211)
- Guaranteed QoS(RFC 2212)

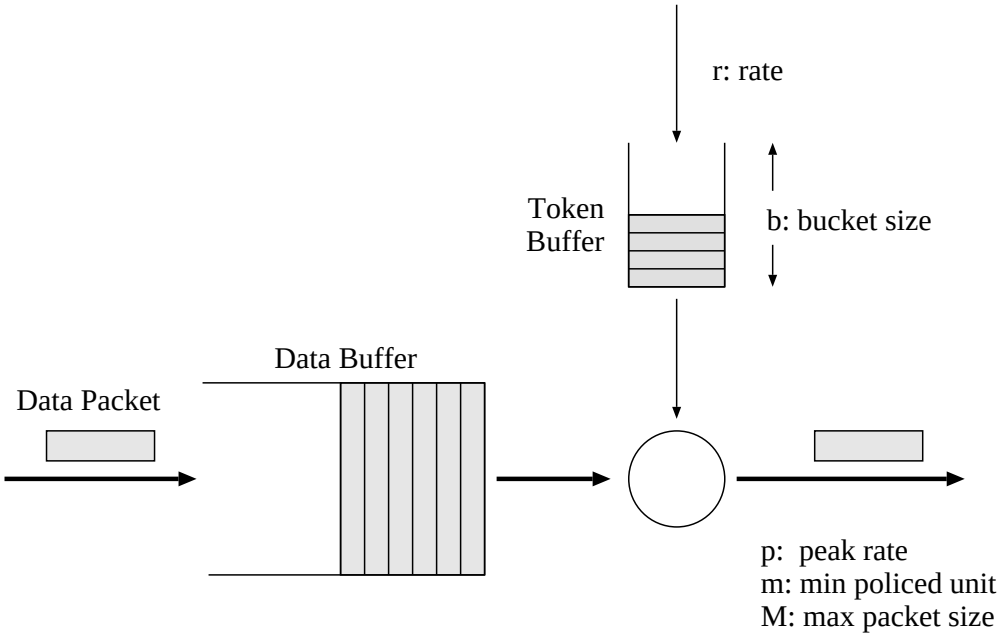

図 2: Tspec パラメータ

- r: 平均レート （平均使用帯域）
- b: バケット長 （バースト長）
- p: 最大レート （e.g, リンクスピード）
- m: 最短パケット （パケット当たりのオーバヘッドから最大遅延の計算に使える）
- M: 最大パケットサイズ （Path MTU）

3.1.4 RSVP ホストとルータ

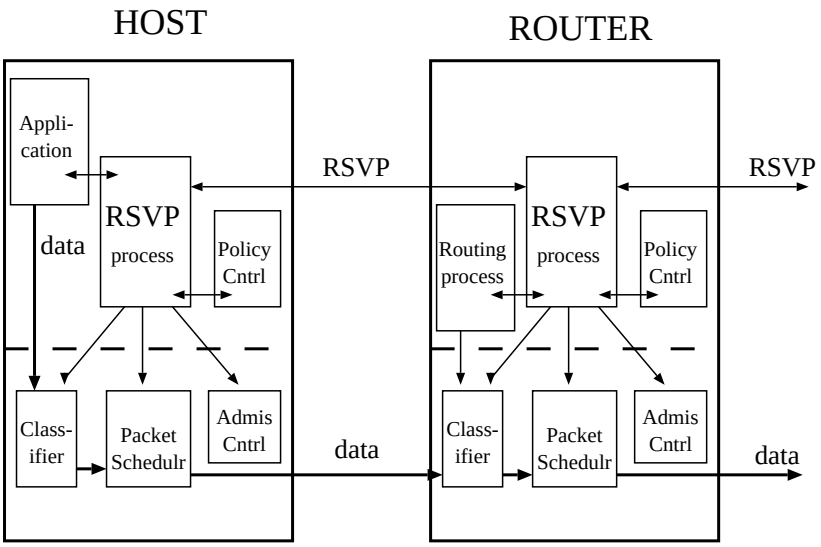

図 3: RSVP ホストとルータ

- ホスト: アプリケーションは RSVP API 使ってローカルな RSVP プロセスと通信する
  - Sender: トラフィックの性質を伝える （Path メッセージ）
  - Receiver: 予約の実行 （Resv メッセージ）
- ルータ: ローカルなルーティング情報をもとに、RSVP メッセージのフォワーディングと予約の実行を行なう
- Policy Control:なんらかのポリシに従ったアドミション制御
- Traffic Control:
  - classifier: パケットヘッダを見てフローを識別
  - packet scheduler: 各フローの帯域を保証
  - admission control: トラフィック制御によるアドミション制御

4 ISI RSVP の実装

4.1 プロセスモデル

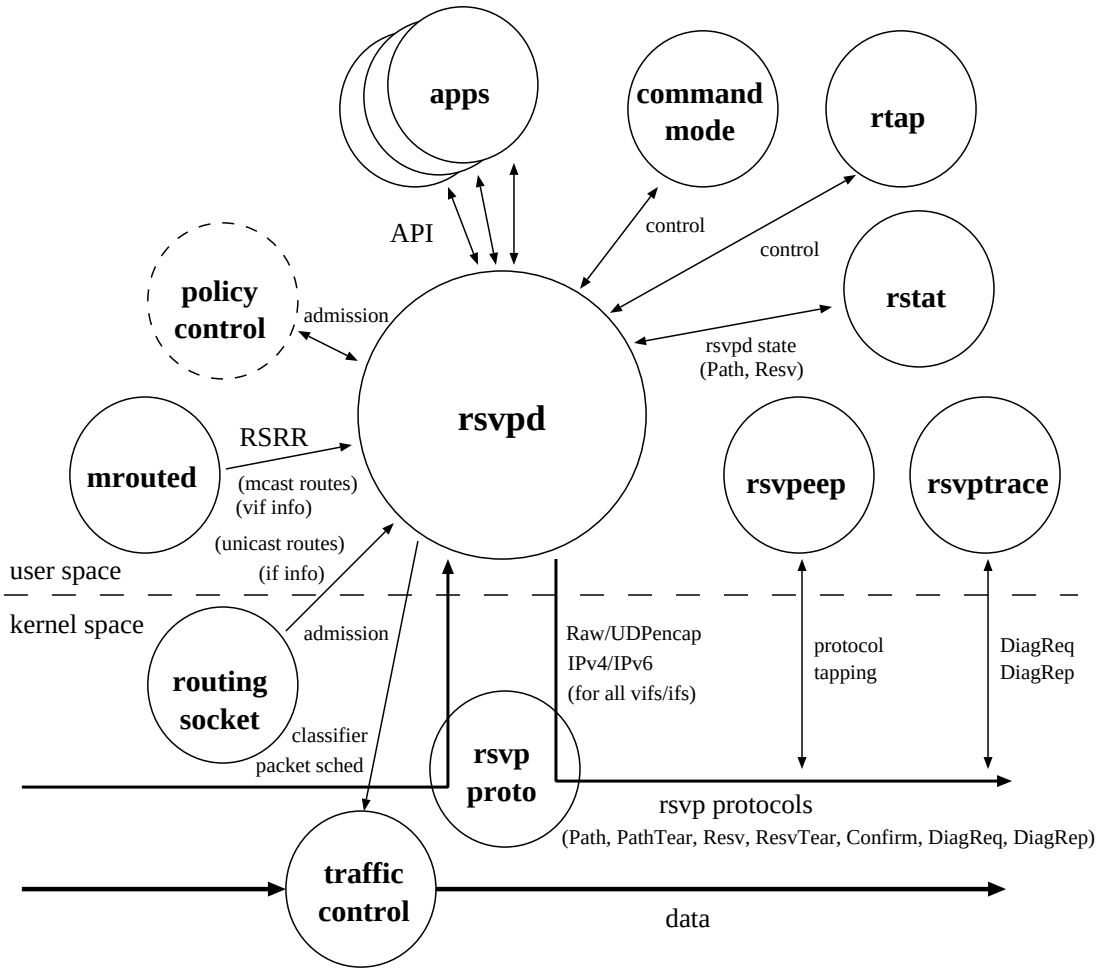

図 4: RSVPD の実装

4.2 rtap による動作例

```
        Sender(172.16.3.117)  ------  Router ----  Receiver
-------------------------------------------------------------------------------

        dest udp 224.100.100.5/9000             dest udp 224.100.100.5/9000


        sender 9001 [t 4000 1000 5000 100]
             (srcport    r   b    p    m)

        **** Path msg が流れ始める ****

                        **** Path msg が届き始める ****

                                                receive
                                                (multicast group に join)

                                                **** Path msg が届き始める ****


                                                reserve ff 172.16.3.117/9001 \
                                                         [cl 4000 1000 100 5000]

                                                (fixed-filter で予約の実行)

                                                **** Resv msg が流れ始める ****
                        **** Resv msg が流れ始める ****
        **** Resv msg が流れ始める ****



                                                close

                                                **** ResvTear msg が流れる ****
                        **** ResvTear msg が流れる ****
        **** ResvTear msg が流れる ****

        close

        **** PathTear msg が流れる ****
                        **** PathTear msg が流れる ****
```

4.3 実装の詳細

4.3.1 データ構造

- PSB: Path State Block
  PSBはPathメッセージに含まれる情報(SESSION、SENDER_TEMPLATEオブジェクト）を保持
  - Session
  - Sender_Template
  - Sender_Tspec
  - The previous hop IP address and the Logical Interface Handle (LIH) from a PHOP object
  - The remaining IP TTL
  - POLICY_DATA and/or ADSPEC objects (optional)
  - Non_RSVP flag
  - E_Police flag
  - Local_Only flag
- RSB: Reservation State Block
  RSB はResv メッセージの情報を保持する
  - Session specification
  - Next hop IP address
  - Filter_spec_list
  - The outgoing (logical) interface OI on which the reservation is to be made or has been made.
  - Style
  - Flowspec
  - A SCOPE object (optional, depending upon style)
  - RESV_CONFIRM object that was received (optional)
- TCSB: Traffic Control State Block
  TCSB はトラフィック制御モジュールの情報を保持する
  - Session
  - OI (Outgoing Interface)
  - Filter_spec_list
  - TC_Flowspec, the effective flowspec, i.e., the LUB over the corresponding FLOWSPEC values from matching RSB's. TC_Flowspec is passed to traffic control to make the actual reservation.
  - Fwd_Flowspec, the updated object to be forwarded after merging.
  - TC_Tspec, equal to Path_Te, the effective sender Tspec.
  - Police Flags The flags are E_Police_Flag, M_Police_Flag, and B_Police_Flag.
  - Rhandle, F_Handle_list Handles returned by the traffic control interface, corresponding to a flowspec and perhaps a list of filter specs.
  - A RESV_CONFIRM object to be forwarded.

- BSB: Blockade State Block
  BSB はブロケードステートの情報を保持する
  - Session
  - Sender_Template (which is also a filter spec)
  - PHOP
  - FLOWSPEC Qb
  - Blockade timer Tb

以下に実際に ISI の RSVPD(rel4.2a) のコードを抜粋して、その動作を説明する。全体の流れをつかむためエラー処理や実装上の詳細部分を省き、また処理が複雑な部分は%で始まるコメントに置き換えてある。

コードは RFC 2209 RSVP Message Processing Rules をほぼ忠実に実装しているので、比較しながら読むとよい。

なお、引用したコードは以下のコピーライトが付いている。

```
/*****************************************************************************

            RSVPD -- ReSerVation Protocol Daemon

                USC Information Sciences Institute
                Marina del Rey, California

Original Version: Shai Herzog, Nov. 1993.
Current Version:  Steven Berson & Bob Braden, May 1996

  Copyright (c) 1996 by the University of Southern California
  All rights reserved.

  Permission to use, copy, modify, and distribute this software and its
  documentation in source and binary forms for any purpose and without
  fee is hereby granted, provided that both the above copyright notice
  and this permission notice appear in all copies, and that any
  documentation, advertising materials, and other materials related to
  such distribution and use acknowledge that the software was developed
  in part by the University of Southern California, Information
  Sciences Institute.  The name of the University may not be used to
  endorse or promote products derived from this software without
  specific prior written permission.

  THE UNIVERSITY OF SOUTHERN CALIFORNIA makes no representations about
  the suitability of this software for any purpose.  THIS SOFTWARE IS
  PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES,
  INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
  MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

  Other copyrights might apply to parts of this software and are so
  noted when applicable.

*****************************************************************/
```

4.4 Path メッセージ受信時の動作

```
 1 /*
 2  * accept_path() This routine processes an incoming Path message,
 3  *      to build/refresh path state.
 4  *
 5  *      in_vif = incoming vif, or -1 if unknown (multicast UDP encap'd)
 6  */
 7 int
 8 accept_path(int in_vif, struct packet *pkt)
 9 {
10     SenderDesc  *sdp = SenderDesc_of(pkt);
11     Session     *destp;
12     PSB         *psbp;
13
14     destp = locate_session_p(pkt->rsvp_sess);
15
16     if (destp == NULL)
17         destp = make_session(pkt->rsvp_sess);
18
19     psbp = locate_PSB(destp, sdp->rsvp_stempl, in_vif,
20                       pkt->pkt_map->rsvp_hop);
21
22     if (psbp == NULL) {
23         psbp = make_PSB(destp, pkt->pkt_map->rsvp_hop, sdp->rsvp_stempl);
24
25         /*
26          % copy contents of the SENDER_TEMPLATE, SENDER_TSPEC,
27          % and PHOP (IP address and LIH) objects into the PSB.
28          */
29
30         psbp->ps_flags |= PSBF_Prefr_need; /* need path refresh */
31     }
32     else {
33         /* there is a matching PSB */
34
35         if (!match_filter((FILTER_SPEC *)filtp, psbp->ps_templ)){
36             rsvp_path_err(in_vif, RSVP_Err_BAD_SNDPORT, 0, pkt);
37             return(-1);
38         }
39
40         if (!hop_eq(pkt->rsvp_phop,&psbp->ps_phop)) {
41             Move_Object(pkt->pkt_map->rsvp_hop, &psbp->ps_rsvp_phop);
42             psbp->ps_flags |= PSBF_Rrefr_need;  /* need resv refresh */
43         }
44
45         if (Compare_Tspecs(sdp->rsvp_stspec, psbp->ps_tspec) != SPECS_EQL) {
46             free((char *) psbp->ps_tspec);
47             psbp->ps_tspec = copy_tspec(sdp->rsvp_stspec);
48             psbp->ps_flags |= PSBF_Prefr_need;  /* need path refresh */
49         }
50     }
51
52     /*
53      %
54      % update current PSB
55      %
56      */
57
58     /*
59      %
```

```
60      % check policy change
61      %
62      */
63
64      if (!Route_Query(destp, psbp, &rr_vif, &new_routes)) {
65          /* route computation was done synchronously */
66          Route_Update(destp, psbp, rr_vif, new_routes);
67          finish_path(destp, psbp);
68      }
69      return(0);
70  }
71
72  /*
73   *      Finish processing Path message after completion of perhaps
74   *      asynchronous route query.
75   */
76  void
77  finish_path(Session *destp, PSB *psbp)
78  {
79      if (psbp->ps_flags & PSBF_Prefr_need)
80          path_refresh_PSB(destp, psbp);
81
82      if (psbp->ps_flags & PSBF_Rrefr_need)
83          resv_refresh(destp, 0);
84  }
85
86  /*
87   * path_refresh_PSB: This routine is called when path state is created
88   *      or modified, to do an immediate refresh for specified PSB.
89   */
90  int
91  path_refresh_PSB(Session *destp, PSB *psbp)
92  {
93
94      /*
95       %
96       % build a new path msg packet
97       %
98       */
99
100     for (vif= 0; vif < vif_num; vif++) {
101         if (!BIT_TST(psbp->ps_outif_list, vif))
102             continue;
103
104         sdscp->rsvp_adspec = TC_Advertise(vif, psbp->ps_adspec,
105                                 (int) psbp->ps_flags & PSBF_NonRSVP);
106
107         send_path_out_vif(vif, destp, STempl_of(sdscp), pkt);
108     }
109
110     return(0);
111 }
```

4.5 PathTear メッセージ受信時の動作

```
 1 /*
 2  *  accept_path_tear(): Processes an incoming PathTear message.
 3  */
 4 int
 5 accept_path_tear(int in_vif, struct packet *pkt)
 6 {
 7     Session     *destp;
 8     PSB         *psbp;
 9     SenderDesc  *sdp = SenderDesc_of(pkt);
10
11     destp = locate_session(pkt->rsvp_sess);
12
13     psbp = locate_PSB(destp, sdp->rsvp_stempl, in_vif,
14                       pkt->pkt_map->rsvp_hop);
15
16     new_pkt = new_packet_area(&data);
17     tear_PSB(destp, psbp, new_pkt, pkt->pkt_map->rsvp_UnkObjList);
18     return 0;
19 }
20
21 /*
22  *      Delete a PSB: forward PTear message to all outgoing vifs and then kill
23  *      PSB.  Common code for accept_path_tear() and cleanup_path_state().
24  *      In the first case, specify list of unknown objects to be forwarded.
25  */
26 int
27 tear_PSB(Session *destp, PSB *psbp, struct packet *pkt, Fobject *fobjp)
28 {
29     /*
30      %
31      % generate PathTear message
32      %
33      */
34
35     /*
36      *  Send PathTear message to each outgoing vif.
37      */
38     if (psbp->ps_ip_ttl > 0) {
39         pkt->pkt_ttl = psbp->ps_ip_ttl - 1;
40         for (vif = 0; vif < vif_num; vif++) {
41             if (BIT_TST(psbp->ps_outif_list, vif))
42                 send_path_out_vif(vif, destp,
43                                   STempl_of(newsdp), pkt);
44         }
45     }
46     return (kill_PSB(destp, psbp));
47 }
```

4.6 Resv メッセージ受信時の動作

```
 1 /*
 2  * accept_resv():  Process incoming Resv message, which is in host
 3  *      byte order.  It may be real packet from remote node or internal
 4  *      packet from local API.
 5  *
 6  *      pkt struct includes:
 7  *              pkt_data -> packet buffer
 8  *              pkt_map -> map of objects in packet
 9  *              pkt_ttl = TTL with which it arrived (if not UDP)
10  *              pkt_flags & PKTFLG_USE_UDP: was it encapsulated?
11  *
12  *      This routine does the common checking and then calls
13  *      flow_reservation() for each distinct flow descriptor packed
14  *      into the message.
15  *
16  */
17 int
18 accept_resv(int in_vif, struct packet   *pkt)
19 {
20     out_vif = IsHopAPI(pkt->rsvp_nhop) ? API_VIF : hop_lih(pkt->rsvp_nhop);
21
22     destp = locate_session_p(pkt->rsvp_sess);
23     if (!destp) {
24         rsvp_resv_err(RSVP_Err_NO_PATH, 0, 0,
25                       (FiltSpecStar *) -1 /*all*/, pkt);
26         return(-1);
27     }
28
29     /*
30      %
31      % check style
32      %
33      */
34
35     switch (style) {
36     case STYLE_FF:
37
38         for (i = 0; i < pkt->rsvp_nflwd; i++) {
39             /* for each flowdesc in the resv msg */
40             filtss.fst_Filtp(0) = filter_of(FlowDesc_of(pkt, i));
41             specp = spec_of(FlowDesc_of(pkt, i));
42
43             rc = flow_reservation(destp, pkt, out_vif, specp, &filtss);
44             if (rc > 0)
45                 need_refresh = 1;
46         }
47         break;
48
49     case STYLE_WF:
50     case STYLE_SE:
51         /*
52          %
53          % omitted
54          %
55          */
56         break;
57     }
58
59     if (destp->d_RSB_list && (Compute_R(&destp->d_timevalr) == 0))
```

```
60          add_to_timer((char *) destp, TIMEV_RESV, destp->d_Rtimor);
61
62      if (need_refresh)
63          resv_refresh(destp, 0);
64
65      return(0);
66  }
67
68  /*
69   * flow_reservation(): Process a given flow descriptor within a Resv
70   *      msg, by finding or creating an RSB.  But if an error is found,
71   *      create no RSB, send ResvErr message, and return.
72   *
73   *      Returns: -1 for error
74   *               1 OK, refresh needed
75   *               0 OK, no refresh
76   */
77  int
78  flow_reservation(
79          Session         *destp,         /* Session                      */
80          Resv_pkt        *pkt,           /* Packet itself                */
81          int              out_vif,       /* True Outgoing interface      */
82          FLOWSPEC        *specp,         /* Flowspec                     */
83          FiltSpecStar    *filtssp)       /* Ptr to FILTER_SPEC*          */
84  {
85
86      /*
87       %
88       % check path state
89       %
90       */
91
92      rp = locate_RSB(destp, pkt->pkt_map->rsvp_hop, filtssp, style);
93      if (!rp) {
94          rp = make_RSB(destp, filtssp->fst_count);
95          /*
96           % initialize the new RSB
97           */
98      }
99      else {
100         /*
101          % check RSB
102          */
103     }
104
105     rc = update_TC(destp, rp);
106     if (rc < 0) {
107         /* admission control failed */
108         rsvp_resv_err(Get_Errcode(rsvp_errno), Get_Errval(rsvp_errno),
109                       flags, filtssp, pkt);
110         return (-1);
111     }
112     return(rc);
113 }
114
115 /*
116  * resv_refresh():  Sends refresh Resv refresh message(s).
117  *
118  *      Spec says resv refreshes are to be sent to specific PHOPs, or
119  *      to all PHOPs.  This implementation cuts a corner, by sending
120  *      resv refreshes to specific incoming interfaces, selected by
```

```
121  *      bits in d_r_incifs; no bits => all.
122  *
123  *      Parameter IsResvErr is 1 if entered from processing ResvErr.
124  *
125  *      Returns: 0 if timer should be rescheduled, else -1.
126  */
127 int
128 resv_refresh(Session *destp, int IsResvErr)
129 {
130     /* Time out any expired state for this Session, and if it's
131      *  all gone, return -1 => cancel refresh timer.
132      */
133     cleanup_resv_state(destp);
134     if (destp->d_RSB_list == NULL)
135         return(-1);
136 
137     /*
138      %
139      % build a new resv packet
140      %
141      */
142     for (sp = destp->d_PSB_list; sp != NULL; sp = sp->ps_next) {
143         if (!BIT_TST(destp->d_r_incifs, sp->ps_in_if))
144             continue;
145         resv_refresh_PSB(destp, lastsp = sp, pkt, IsResvErr);
146     }
147 
148     return(0);
149 }
150 
151 /*
152  *      resv_refresh_PSB():  Process one PSB to generate Resv refresh
153  *              message for its PHOP.  If this is last PSB for PHOP,
154  *              send the Resv message.
155  */
156 int
157 resv_refresh_PSB(
158         Session         *destp,
159         PSB             *sp,
160         struct packet   *pkt,
161         int             IsResvErr)
162 {
163 
164     for (rp = destp->d_RSB_list; rp != NULL; rp = rp->rs_next) {
165         /*
166          %
167          % try merging the list
168          %
169          */
170     }
171 
172     /*
173      %
174      % do style-dependent process
175      %
176      */
177 
178     /*
179      *  Senders are ordered by phop address.  Return if next PSB has
180      *  the same PHOP, else finish up message and send it.
181      */
```

```
182     if (sp->ps_next && hop_addr_eq(&sp->ps_next->ps_phop,&sp->ps_phop))
183         return(0);
184
185     send_resv_out(sp, pkt);
186
187     return (0);
188 }
```

4.7 ResvTear メッセージ受信時の動作

```
 1 /*
 2  *  accept_resv_tear(): Process an incoming ResvTear message
 3  */
 4 int
 5 accept_resv_tear(
 6         int             in_if,  /* Alleged outgoing iface (IGNORED)  */
 7         struct packet   *pkt)
 8 {
 9
10     out_vif = hop_lih(pkt->rsvp_nhop);
11
12     destp = locate_session(pkt->rsvp_sess);
13     if (!destp)
14         return(0);
15
16     /* tear down local reservation state in style-dependent mannar */
17     switch (style) {
18     case STYLE_FF:
19         filtss.fst_count = 1;
20         for (i = 0; i < pkt->rsvp_nflwd; i++) {
21             filtss.fst_filtp0 = filter_of(FlowDesc_of(pkt, i));
22             Refresh_Needed |=
23                 tear_reserv(destp, nhopp, &filtss, style);
24         }
25         break;
26
27     case STYLE_SE:
28     case STYLE_WF:
29         /*
30          %
31          %
32          %
33          */
34         break;
35     }
36
37     /*
38      %
39      % create RecvTear msg to be forwarded
40      %
41      */
42
43
44     /*
45      *  1.  Select each PSB whose OutInterface_list includes the
46      *  outgoing interface OI (more strictly, that routes to NHOP).
47      *  For distinct style, SENDER_TEMPLATE must also match a
48      *  filter spec in the received ResvTear.
49      */
```

```
50      for (sp = destp->d_PSB_list; sp != NULL; sp = sp->ps_next) {
51          if (out_if-doesnt-match)
52              continue;
53
54          /* 2.   Select a flow descriptor Fj in the ResvTear message
55           *      whose FILTER_SPEC matches the SENDER_TEMPLATE in the
56           *      PSB.  If no match is found, return to select next PSB.
57           */
58          for (j = 0; j < pkt->rsvp_nflwd; j++) {
59              if (FILTER_SPEC--match-SENDER_TEMPLATE)
60                  /*
61                   * Do PSB-specific processing, building flow
62                   * descriptor list in outpkt.
63                   */
64                  resv_tear_PSB(destp, sp, Style(pkt),
65                                filter_of(FlowDesc_of(pkt,j)), outpkt);
66          }
67
68          /*
69           * 3. If the next PSB is for a different PHOP or the last
70           *      PSB has been processed, forward any ResvTear message
71           *      that has been built.  Senders are ordered by phop
72           *      address.
73           */
74          if (sp->ps_next == NULL ||
75              !hop_addr_eq(&sp->ps_next->ps_phop,&sp->ps_phop)) {
76              if (outpkt->rsvp_nflwd)
77                  send_resv_out(sp, outpkt);
78              outpkt->rsvp_nflwd = 0;
79          }
80
81      }
82
83      if (nmatch > 0 && (Refresh_Needed))
84          resv_refresh(destp, 0);
85
86      return(0);
87  }
```

5 CBQ (Class-Based Queueing)

ここからはトラフィックコントロールの実装例としてCBQをみていく。

5.1 CBQ の特徴

- インターフェイスごとに、階層的に構成したクラスによるリンク共有を実現
- 各クラスの平均送出パケット間隔を測定して、使用帯域制限を行なう
- 上位のクラスから利用帯域を借り入れできる
- QoS 保証が可能
  (RSVP のトラフィックコントロールカーネルとして使える。Controlled Load サービスのみ)

5.2 CBQ の構成要素

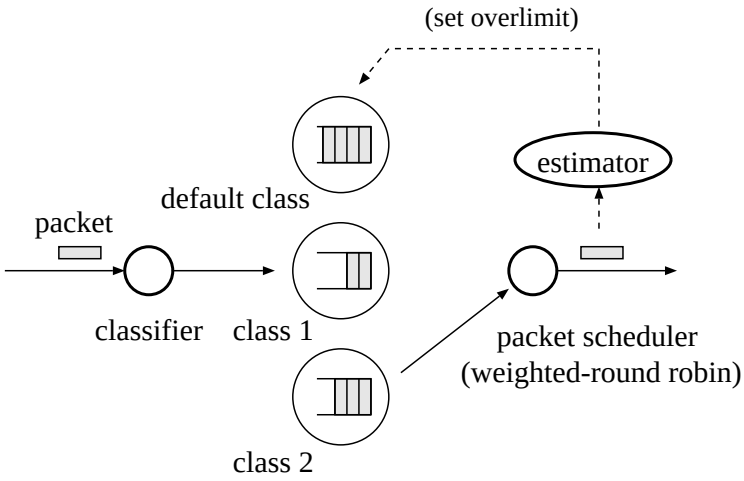

図 5: CBQ の構成要素

- classifier: パケットヘッダを見て対応するクラスのキューにパケットを入れる
- estimator: 各クラスの平均パケット送出間隔を測定する
- packet scheduler: 次にどのクラスからパケットを送るか決定する プライオリティとウエイテッド・ラウンドロビンの併用
- delayer: オーバリミットのクラスをサスペンドする

5.3 CBQ のトラフィックコントロールの仕組み

CBQ は送出パケット間隔を使って、トークンバケットと同様の仕組みを実現する。

5.3.1 加重平均パケット間隔

割り当て帯域が等間隔パケット送出で使われる場合の理想のパケットインターバルとの差分を加重平均する。

```
t: target packet interval
idle: measured idle time
W = 0.03125 (1/32)

diff = t - idle
avgidle = (1 - W) * avg + W * diff
```

- クラスが割り当て帯域以上を使っていると avgidle は上昇、割り当て帯域を使っていないときは avgidle は下降する。
- (avgidle ¡ 0) でクラスがオーバリミットだと判定し、そのクラスからのパケット送出をストップする。
- avgidle の最大値を制限すると、トークンバケット長と同様の効果がある。
- サスペンドされたクラスをリジュームするタイミングは最悪カーネルタイマの粒度（通常 10ms）。
- 帯域制限が効いている場合は、数パケット送出しては停止する、ややバースティなパケット送出サイクルとなる。

5.4 ALTQ (Alternate Queueing for BSD Unix)

- さまざまなキューイングを実装するためのフレームワーク（CBQ、RED、WFQ 等が実装されている）
- CBQ を ISI の RSVP と連動させるためのスタブが含まれる
- FreeBSD で動作する

5.5 RSVP とのインターフェイス

5.5.1 クラス階層

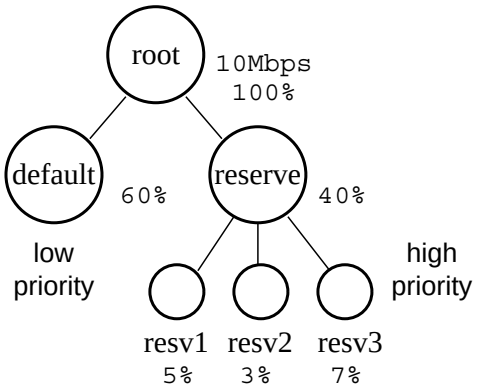

図 6: RSVP 利用時のクラス構成

5.5.2 RSVP TC インターフェイス

- 予約の作成
  `TC_AddFlowspec( Interface, TC_Flowspec, TC_Tspec, TC_Adspec, Police_Flags ) -> RHandle [, Fwd_Flowspec]`
  CBQ: アドミッション制御、OK なら対応する CBQ クラスを作る

- 予約の変更
  `TC_ModFlowspec( Interface, RHandle, TC_Flowspec, TC_Tspec, TC_Adspec, Police_flags ) [ -> Fwd_Flowspec ]`
  CBQ: 変更が可能であるかチェック、可能であれば新しいスペック
  に対応するクラスを作って差し替える

- Flow Spec の削除
  `TC_DelFlowspec( Interface, RHandle )`
  CBQ: 対応するクラスを削除する

- Filter Spec の追加
  `TC_AddFilter( Interface, RHandle, Session , FilterSpec ) -> FHandle`
  CBQ: クラスににフィルタを設定する

- Filter Spec の削除
  `TC_DelFilter( Interface, FHandle )`
  CBQ: クラスからフィルタを削除する

- Path 情報の更新
  `TC_Advertise( Interface, Adspec, Non_RSVP_Hop_flag ) -> New_Adspec`
  CBQ: ローカル情報にしたがって、Adspec の帯域、遅延、MTU 等
  のパラメータを変更する

- Preemption Upcall
  `TC_Preempt() -> RHandle, Reason_code (Up Call)`
  CBQ: 実装されていない

A 関連情報

A.1 RSVP related RFC list

RSVP-WG

- Proposed Standard:
  - 2205 Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. September 1997.
  - 2206 RSVP Management Information Base using SMIv2. F. Baker, J. Krawczyk, A. Sastry. September 1997.
  - 2207 RSVP Extensions for IPSEC Data Flows. L. Berger, T. O'Malley. September 1997.
- Informational:
  - 2208 Resource ReSerVation Protocol (RSVP) – Version 1 Applicability Statement Some Guidelines on Deployment. A. Mankin, Ed., F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang. September 1997.
  - 2209 Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. R. Braden, L. Zhang. September 1997.

IntServ-WG

- Proposed Standard:
  - 2210 The Use of RSVP with IETF Integrated Services. J. Wroclawski. September 1997.
  - 2211 Specification of the Controlled-Load Network Element Service. J. Wroclawski. September 1997.
  - 2212 Specification of Guaranteed Quality of Service. S. Shenker, C. Partridge, R. Guerin. September 1997.
  - 2213 Integrated Services Management Information Base using SMIv2. F. Baker, J. Krawczyk, A. Sastry. September 1997.
  - 2214 Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2. F. Baker, J. Krawczyk, A. Sastry. September 1997.
  - 2215 General Characterization Parameters for Integrated Service Network Elements. S. Shenker, J. Wroclawski. September 1997.
- Informational:
  - 2216 Network Element Service Specification Template. S. Shenker, J. Wroclawski. September 1997.

A.2 RSVP ベンダ・リスト

1997 年 7 月に RSVP-WG によって行なわれたサーベイから、RSVP 対応の実装に取り組んでいるベンダのリストである。(ほとんどまだ製品化されていない)

- ルータ系: Cisco, 3Com, Bay Networks, Fore Systems, Furukawa Electric, IBM, Pivotal Networking, Inc.
- ホスト系: Sun Microsystems, Silicon Graphics, Inc., Digital Equipment Corp., Hewlett-Packard, Microsoft Corp., FTP Software, Inc., Intel Corp., NetManage, CLASS Data Systems, Precept Software, Inc., CEFRIEL/Politecnico di Milano, GMD Fokus - STEP, Sony CSL

A.3 トラフィック関連参考書

- An Engineering Approach to Computer Networking. S. Keshav. Addison-Wesley. ISBN 0-201-63442-2.
- Gigabit Networking. Craig Partridge. Addison-Wesley. ISBN 0-201-56333-9.

A.4 関連リンク

- RSVP: `http://www.isi.edu/rsvp/`
- CBQ: `http://www-nrg.ee.lbl.gov/floyd/cbq.html`
- RFC: `http://www.internic.net/ds/rfc-index.html`
- ALTQ: `http://www.csl.sony.co.jp/person/kjc/programs.html`