DCCP Implementation Status

dccp@vger.kernel.org
Outline

1. Applications & Ports
2. Socket API - Packet Priorities
3. DCCP Nat Traversal
4. CCID-3 changes
5. Current work
6. Further work
Applications and ports

- **Work by Leandro Melo de Sales, Brasil**
- **CCID-4 subtree**
  `git://eden-feed.erg.abdn.ac.uk/dccp_exp`
- **DCCP port for Embedded Phone Project**
  - Maemo kernel with DCCP support
  - for mobile devices such as the Nokia N810
  - [https://garage.maemo.org/projects/ephone](https://garage.maemo.org/projects/ephone)
- **gstreamer DCCP plugin**
  - GNU gstreamer is **the** toolbox for streaming apps
  - facilitates wide range of possible applications/uses
Socket API: Packet Priorities

- Work by Tomasz Grobelny, Poland
- **per-packet priorities**
  - timeout, numeric priority, symbolic priority, ...
  - passed as cmsg(3) parameter to sendmsg()
  - can use different types of priority queue
- **policies** which act on and interpret the priorities
  - drop-lowest-priority first
  - look-at-best-before-date-of-packet
  - send-best-packet-next ?
  - ...
DCCP NAT Traversal

- Work by Patrick McHardy
  - DCCP NAT available already at a Linux near you
  - *Linux the only (stateful) NAT to support DCCP*

- Implementation of draft-ietf-dccp-simul-open:
  - fairly straightforward & already works
  - *need IANA type for DCCP-Listen packet*
  - at the moment supports DCCPv4 and 1 peer
  - easily extended to other scenarios
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>139.133.209.65</td>
<td>139.133.209.176</td>
<td>DCCP</td>
<td>5001 &gt; 1009 [Listen] Seq=0 (service=1179602720)</td>
</tr>
<tr>
<td>2</td>
<td>0.196758</td>
<td>139.133.209.65</td>
<td>139.133.209.176</td>
<td>DCCP</td>
<td>5001 &gt; 1009 [Listen] Seq=0 (service=1179602720)</td>
</tr>
<tr>
<td>3</td>
<td>0.396736</td>
<td>139.133.209.65</td>
<td>139.133.209.176</td>
<td>DCCP</td>
<td>5001 &gt; 1009 [Listen] Seq=0 (service=1179602720)</td>
</tr>
<tr>
<td>4</td>
<td>0.65543</td>
<td>139.133.209.65</td>
<td>139.133.209.65</td>
<td>DCCP</td>
<td>1009 &gt; 5001 [Request] Seq=209913913613052 (service=1179602720)</td>
</tr>
<tr>
<td>5</td>
<td>0.656139</td>
<td>139.133.209.65</td>
<td>139.133.209.176</td>
<td>DCCP</td>
<td>5001 &gt; 1009 [Response] Seq=209801263285276 (Ack=209913913615952) (service=1179602720)</td>
</tr>
<tr>
<td>6</td>
<td>0.656960</td>
<td>139.133.209.65</td>
<td>139.133.209.176</td>
<td>DCCP</td>
<td>1009 &gt; 5001 [Ack] Seq=209913913615952 (Ack=209901263285276)</td>
</tr>
<tr>
<td>7</td>
<td>0.660166</td>
<td>139.133.209.65</td>
<td>139.133.209.65</td>
<td>DCCP</td>
<td>1009 &gt; 5001 [DataAck] Seq=209913913615954 (Ack=209901263285276)</td>
</tr>
<tr>
<td>8</td>
<td>0.660224</td>
<td>139.133.209.65</td>
<td>139.133.209.65</td>
<td>DCCP</td>
<td>1009 &gt; 5001 [Close] Seq=209913913615954 (Ack=209901263285276)</td>
</tr>
<tr>
<td>9</td>
<td>0.662593</td>
<td>139.133.209.65</td>
<td>139.133.209.65</td>
<td>DCCP</td>
<td>1009 &gt; 5001 [Ack] Seq=209801263285277 (Ack=209913913615954)</td>
</tr>
<tr>
<td>10</td>
<td>0.729435</td>
<td>139.133.209.65</td>
<td>139.133.209.65</td>
<td>DCCP</td>
<td>5001 &gt; 1009 [Reset] Seq=209801263285278 (Ack=209913913615955) (code=Closed)</td>
</tr>
</tbody>
</table>

Frame 1 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 3con_7b:e9:a9 (00:60:08:7b:e9:a9), Dst: Dell.4550_e100 (00:07:e9:bd:5d:1f)
Internet Protocol, Src: 139.133.209.65 (139.133.209.65), Dst: 139.133.209.176 (139.133.209.176)
Datagram Congestion Control Protocol, Src Port: 5001 (5001), Dst Port: 1009 (1009) [Listen] Seq=0

Source Port: 5001 (5001)
Destination Port: 1009 (1009)
Data Offset: 5
CCVal: 0
Checksum Coverage: 0
Checksum: 0x7ee3 [correct]
Type: Listen (10)
Extended Sequence Numbers: True
Sequence Number: 0

Service Code: 1179602720
Backwards compatibility in 3 lines

```c
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -809,6 +809,10 @@ static int dccp_v4_rcv(

dh = dccp_hdr(skb);

+ /* Ignore DCCP-Listen packets (NAT Traversal) */
+    if (dh->dccph_type == DCCP_PKT_INVITE)
+          goto discard_it;

dccpd_seq  = dccp_hdr_seq(dh);

dccpd_type = dh->dccph_type;
```
Current work

- Contributions from Wei Yonjung:
  - TAHI tests for DCCP
    - helped uncover several bugs
    - proved very useful input
  - (slowly) adding *changes from rfc3448bis-06*
  - rewriting CCID-3 code to *support ECN*
    - ECN subtree available already
- *Oscillation Prevention* for CCID-3/4
- modularisation of TFRC code
TFRC librarification

The entire CCID-3 Receiver in one slide:

```c
void ccid3_hc_rx_packet_recv(skb)
{
    struct ccid3_hc_rx_sock *hcrx = ccid3_hc_rx_sk(skb);
    const u64 ndp = dccp_sk(skb)->dccpor_ndp;
    const bool is_data_packet = dccp_data_packet(skb);

    if (tfrc_rx_congestion_event(&hcrx->hist, &hcrx->li_hist, skb, ndp, ccid3_first_li, skb))
        send_feedback(skb, skb, CCID3_FBACK_PARAM_CHANGE);

    else if (hcrx->feedback == CCID3_FBACK_NONE && is_data_packet)
        send_feedback(skb, skb, CCID3_FBACK_INITIAL);

    else if (!loss_pending(&hcrx->hist) && is_data_packet &&
                SUB16(dccp_hdr(skb)->ccval, hcrx->last_counter) > 3)
        send_feedback(skb, skb, CCID3_FBACK_PERIODIC);
}
Oscillation Reduction before/after
Further work

- **ECN work to be finished**
  - needs testing & verification

- **CCID-3 needs better RTT estimation**
  - see other slides

- **CCID-2 needs an overhaul**
  - reverse-path congestion not supported
  - very good initial results in using CWND Validation

- **Ack Vectors need new algorithms**
  - on 802.11g links they grow up to 0.5 kilobyte!
Conclusions

- need more testers/contributors
  - TAH1 tests proved very useful
  - code only gets good through frequent review
- still a lot to be done
- Linux DCCP framework is reasonably stable