Subject: [patch 4/5][RFC - ipv4/udp checkpoint/restart] : c/r the inet options of the socket
Posted by Daniel Lezcano on Wed, 06 Jun 2007 12:18:10 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

This patch defines a set of netlink attributes to store/retrieve inet
options. The logic is to extend the netlink message attribute to take into
account these new values.

The multicast list is browsed first and the netlink nested attribute is filled
in the reverse order. That allows, when restoring the socket, to keep the
initial order of the multicast list. Not really a big issue if the list are
inverted, but that facilitate the test because the attribute will stay exactly,
the same and comparison with initial socket and restored socket can be done with
a simple "memcmp".

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 include/linux/af_inet_cr.h | 19 ++++
 net/ipv4/af_inet_cr.c | 205 ++-
 2 files changed, 223 insertions(+), 1 deletion(-)

Index: 2.6.20-cr/include/linux/af_inet_cr.h
===
--- 2.6.20-cr.orig/include/linux/af_inet_cr.h
+++ 2.6.20-cr/include/linux/af_inet_cr.h
@@ -71,6 +71,25 @@
  AF_INET_CR_ATTR_SOCKOPT_SNDBUF_ULOCK,
  AF_INET_CR_ATTR_SOCKOPT_RCVBUF_ULOCK,

+ /* ip options */
+ AF_INET_CR_ATTR_IPOPT_OPTIONS,
+ AF_INET_CR_ATTR_IPOPT_PKTINFO,
+ AF_INET_CR_ATTR_IPOPT_RECVTOS,
+ AF_INET_CR_ATTR_IPOPT_RECVTTL,
+ AF_INET_CR_ATTR_IPOPT_RECVOPTS,
+ AF_INET_CR_ATTR_IPOPT_RETOPTS,
+ AF_INET_CR_ATTR_IPOPT_TOS,
+ AF_INET_CR_ATTR_IPOPT_TTL,
+ AF_INET_CR_ATTR_IPOPT_HDRINCL,
+ AF_INET_CR_ATTR_IPOPT_RECVERR,
+ AF_INET_CR_ATTR_IPOPT_MTU_DISCOVER,
+ AF_INET_CR_ATTR_IPOPT_ROUTER_ALERT,
+ AF_INET_CR_ATTR_IPOPT_MULTICAST_TTL,
+ AF_INET_CR_ATTR_IPOPT_MULTICAST_LOOP,

```
+ AF_INET_CR_ATTR_IPOPT_MEMBERSHIP,
+ AF_INET_CR_ATTR_IPOPT_MREQ,
+ AF_INET_CR_ATTR_IPOPT_MULTICAST_IF,
+
  AF_INET_CR_ATTR_MAX
 };
 #endif
Index: 2.6.20-cr/net/ipv4/af_inet_cr.c
===================================================================
--- 2.6.20-cr.orig/net/ipv4/af_inet_cr.c
+++ 2.6.20-cr/net/ipv4/af_inet_cr.c
@@ -13,8 +13,12 @@
 #include <net/sock.h>
 #include <linux/fs.h>
 #include <linux/syscalls.h>
+#include <linux/in.h>
+#include <linux/igmp.h>
 #include <linux/af_inet_cr.h>

+#include <net/ip.h>
+
 /*
  * Netlink message policy definition
  */
@@ -44,6 +48,30 @@
  [AF_INET_CR_ATTR_SOCKOPT_SNDTIMEO] = { .len = sizeof(struct timeval) },
  [AF_INET_CR_ATTR_SOCKOPT_LINGER]   = { .len = sizeof(struct linger)  },
  [AF_INET_CR_ATTR_SOCKOPT_BINDTODEVICE]  = { .len  = IFNAMSIZ },
+
+ /* ip options */
+ [AF_INET_CR_ATTR_IPOPT_PKTINFO]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_RECVTOS]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_RECVTTL]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_RECVOPTS]        = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_RETOPTS]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_HDRINCL]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_RECVERR]         = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_ROUTER_ALERT]    = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_MULTICAST_LOOP]  = { .type = NLA_FLAG },
+ [AF_INET_CR_ATTR_IPOPT_MULTICAST_IF]    = { .type = NLA_U32 },
+ [AF_INET_CR_ATTR_IPOPT_TOS]             = { .type = NLA_U8 },
+ [AF_INET_CR_ATTR_IPOPT_TTL]             = { .type = NLA_U8 },
+ [AF_INET_CR_ATTR_IPOPT_MTU_DISCOVER]    = { .type = NLA_U8 },
+ [AF_INET_CR_ATTR_IPOPT_MULTICAST_TTL]   = { .type = NLA_U8 },
+ [AF_INET_CR_ATTR_IPOPT_MEMBERSHIP]      = { .type = NLA_NESTED },
+
+ [AF_INET_CR_ATTR_IPOPT_MREQ] =
+ { .len =  sizeof(struct ip_mreqn) },
```

```
+
+ [AF_INET_CR_ATTR_IPOPT_OPTIONS] =
+ { .len = sizeof(struct ip_options) + 40 },
+
 };

 /*
@@ -77,6 +105,28 @@
  { SO_BINDTODEVICE, AF_INET_CR_ATTR_SOCKOPT_BINDTODEVICE, 0, SET  },
 };

+/*
+ * ip options association with netlink attribute
+ */
+struct af_inet_cr_optattr ip_options[] = {
+ { IP_PKTINFO,        AF_INET_CR_ATTR_IPOPT_PKTINFO,        1, BOTH },
+ { IP_RECVTOS,        AF_INET_CR_ATTR_IPOPT_RECVTOS,        1, BOTH },
+ { IP_RECVTTL,        AF_INET_CR_ATTR_IPOPT_RECVTTL,        0, BOTH },
+ { IP_RECVOPTS,       AF_INET_CR_ATTR_IPOPT_RECVOPTS,       1, BOTH },
+ { IP_RETOPTS,        AF_INET_CR_ATTR_IPOPT_RETOPTS,        1, BOTH },
+ { IP_TOS,            AF_INET_CR_ATTR_IPOPT_TOS,            0, BOTH },
+ { IP_TTL,            AF_INET_CR_ATTR_IPOPT_TTL,            0, BOTH },
+ { IP_HDRINCL,        AF_INET_CR_ATTR_IPOPT_HDRINCL,        1, BOTH },
+ { IP_RECVERR,        AF_INET_CR_ATTR_IPOPT_RECVERR,        1, BOTH },
+ { IP_MTU_DISCOVER,   AF_INET_CR_ATTR_IPOPT_MTU_DISCOVER,   0, BOTH },
+ { IP_MULTICAST_TTL,  AF_INET_CR_ATTR_IPOPT_MULTICAST_TTL,  1, BOTH },
+ { IP_MULTICAST_LOOP, AF_INET_CR_ATTR_IPOPT_MULTICAST_LOOP, 0, BOTH },
+ { IP_MULTICAST_IF,   AF_INET_CR_ATTR_IPOPT_MULTICAST_IF,   1, BOTH },
+ /* FIXME (for RAW sockets) */
+ /*{ IP_ROUTER_ALERT, AF_INET_CR_ATTR_IPOPT_ROUTER_ALERT, 0, BOTH  },*/
+ /* FIXME */
+       /* { IP_OPTIONS, AF_INET_CR_ATTR_IPOPT_OPTIONS, */
+};

 /*
  * socket_lookup : search for socket using the inode number
@@ -188,7 +238,7 @@
 }

 /*
- * af_inet_cr_opt2attr : convert a netlink attribute to a socket option
+ * af_inet_cr_attr2opt : convert a netlink attribute to a socket option
  *  and set the option to the socket
  *
  * @info : the generic netlink message
@@ -363,6 +413,90 @@
 }
```

```
 /*
+ * dump_inetopt_mc : retrieve the multicast list and store them
+ *  to the netlink message
+ *
+ * @sock : the socket to retrieve the multicast list
+ * @skb : the skbuff containing the netlink message
+ *
+ * Returns 0 on success, < 0 otherwise
+ */
+static int dump_inetopt_mc(struct socket *sock, struct sk_buff *skb)
+{
+ int i, ret = 0;
+ struct sock *sk = sock->sk;
+ struct inet_sock *inet = inet_sk(sk);
+ struct ip_mc_socklist *mc;
+ struct nlattr *mx;
+ struct ip_mreqn **mreq = NULL;
+
+ mreq = kmalloc(sizeof(struct ip_mreqn *)*sysctl_igmp_max_memberships, GFP_KERNEL);
+ if (!mreq)
+  return -ENOMEM;
+
+ i = sysctl_igmp_max_memberships - 1;
+
+ rtnl_lock();
+ for (mc = inet->mc_list; mc; mc = mc->next)
+  mreq[i--] = &mc->multi;
+ i++;
+
+ if (i == sysctl_igmp_max_memberships)
+  goto out;
+
+ ret = -ENOMEM;
+ mx = nla_nest_start(skb, AF_INET_CR_ATTR_IPOPT_MEMBERSHIP);
+ if (!mx)
+  goto out;
+
+ for (; i < sysctl_igmp_max_memberships; i++) {
+  ret = nla_put(skb, AF_INET_CR_ATTR_IPOPT_MREQ,
+         sizeof(mc->multi), mreq[i]);
+  if (ret) {
+   nla_nest_cancel(skb, mx);
+   goto out;
+  }
+ }
+ nla_nest_end(skb, mx);
+out:
+ rtnl_unlock();
```

```
+ kfree(mreq);
+
+ return ret;
+}
+
+/*
+ * dump_inetopt : retrieve the inet options and store them
+ *  to the netlink message
+ *
+ * @sock : the socket to retrieve the inet options
+ * @skb : the skbuff containing the netlink message
+ *
+ * Returns 0 on success, < 0 otherwise
+ */
+static int dump_inetopt(struct socket *sock, struct sk_buff *skb)
+{
+ int ret;
+ size_t i, len;
+
+ len = sizeof(ip_options)/sizeof(struct af_inet_cr_optattr);
+
+ for (i = 0; i < len; i++) {
+  if (sock->type == SOCK_STREAM && ip_options[i].unconnected)
+   continue;
+  ret = af_inet_cr_opt2attr(skb, sock, IPPROTO_IP, &ip_options[i]);
+  if (ret)
+   return ret;
+ }
+
+ ret = dump_inetopt_mc(sock, skb);
+ if (ret)
+  return ret;
+
+ return 0;
+}
+
+/*
  * dump_socket : dump the socket state, type and options into a netlink message
  *  and transmit the message to the sender of the dump command.
  *
@@ -404,6 +538,10 @@
  if (ret)
   goto out;

+  ret = dump_inetopt(sock, skb);
+ if (ret)
+  goto out;
+
```

```
  ret = genlmsg_end(skb, msg_head);
  if (ret < 0)
   goto out;
@@ -517,6 +655,67 @@
 }

 /*
+ * restore_inetopt : extract multicast list from netlink message and
+ *  set it to the socket
+ *
+ * @sock : the socket to be restored
+ * @info : the netlink message
+ *
+ * Returns 0 on success, < 0 otherwise
+ */
+static inline int restore_inetopt_mc(struct socket *sock,
+         struct genl_info *info)
+{
+ struct ip_mreqn mreq;
+ struct nlattr *nla, *nested;
+ int nla_rem;
+ int ret;
+
+ nested = info->attrs[AF_INET_CR_ATTR_IPOPT_MEMBERSHIP];
+ if (nested) {
+  nla_for_each_nested(nla, nested, nla_rem) {
+   nla_memcpy(&mreq, nla, sizeof(mreq));
+   ret = kernel_setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP,
+       (char *)&mreq, sizeof(mreq));
+   if (ret)
+    return ret;
+  }
+ }
+ return 0;
+}
+
+/*
+ * restore_inetopt : extract inet options from netlink message and
+ *  set them to the socket
+ *
+ * @sock : the socket to be restored
+ * @info : the netlink message
+ *
+ * Returns 0 on success, < 0 otherwise
+ */
+static inline int restore_inetopt(struct socket *sock, struct genl_info *info)
+{
+ int ret;
```

```
+ size_t i, len;
+
+ len = sizeof(ip_options)/sizeof(struct af_inet_cr_optattr);
+
+ for (i = 0; i < len; i++) {
+  if (sock->type == SOCK_STREAM && ip_options[i].unconnected)
+   continue;
+  ret = af_inet_cr_attr2opt(info, sock, IPPROTO_IP, &ip_options[i]);
+  if (ret)
+   return ret;
+ }
+
+ ret = restore_inetopt_mc(sock, info);
+ if (ret)
+  return ret;
+
+ return 0;
+}
+
+/*
  * restore_socket : restore the socket from the netlink message content
  *
  * @sock : the socket to be restored
@@ -545,6 +744,10 @@
  ret = restore_sockopt(sock, info);
  if (ret)
   goto out;
+
+  ret = restore_inetopt(sock, info);
+ if (ret)
+  goto out;
 out:
  return ret;
 }

--
```