Subject: [patch 5/5][RFC - ipv4/udp checkpoint/restart] : c/r the udp part of the socket
Posted by Daniel Lezcano on Wed, 06 Jun 2007 12:18:11 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 udp
option and endpoints. The logic is to extend the netlink message attribute
to take into account these new values.
The ops of struct sock is extended with the dump/restore callbacks, so when
a socket is asked to be checkpointed, the call will fail if the dump/restore
is not implemented in the protocol. That allows to bring C/R functionnality
for each protocol step by step.

 * At dump time : the local binding is retrieve from kernel_getname and
   distant binding is retrieve with kernel_getpeername.

 * At restore time : the local binding is set by the kernel_bind call and
   distant binding is set by kernel_connect
   If the local binding was done with an autobind, the userlock flags, will
   not be set, so the flag are resetted if they were not set during the dump.

One point to be discussed is : should we C/R sendQ and recvQ knowing the
protocol is not reliable ?

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 include/linux/af_inet_cr.h | 9 +++
 include/linux/udp_cr.h | 26 +++++++++
 include/net/sock.h | 6 +-
 net/ipv4/Makefile | 2
 net/ipv4/af_inet_cr.c | 23 ++++++++
 net/ipv4/udp.c | 6 +-
 net/ipv4/udp_cr.c | 119 ++
 7 files changed, 187 insertions(+), 4 deletions(-)

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
@@ -90,6 +90,15 @@
  AF_INET_CR_ATTR_IPOPT_MREQ,
  AF_INET_CR_ATTR_IPOPT_MULTICAST_IF,

+ /* udp options */
+ AF_INET_CR_ATTR_UDPOPT_CORK,
+

```
+ /* udp protocol */
+ UDP_CR_ATTR_BIND,
+ UDP_CR_ATTR_BIND_ADDR_ULOCK,
+ UDP_CR_ATTR_BIND_PORT_ULOCK,
+ UDP_CR_ATTR_PEER,
+
  AF_INET_CR_ATTR_MAX
 };
 #endif
Index: 2.6.20-cr/include/linux/udp_cr.h
===================================================================
--- /dev/null
+++ 2.6.20-cr/include/linux/udp_cr.h
@@ -0,0 +1,26 @@
+/*
+ *
+ *
+ */
+#ifndef _UDP_CR_H
+#define _UDP_CR_H
+#include <linux/skbuff.h>
+#include <linux/errno.h>
+#include <net/genetlink.h>
+
+#ifdef CONFIG_IP_CR
+extern int udp_dump(struct socket *sock, struct sk_buff *skb);
+extern int udp_restore(struct socket *sock, const struct genl_info *info);
+#else
+static inline int udp_dump(struct socket *sock, struct sk_buff *skb)
+{
+       return -ENOSYS;
+}
+
+static inline int udp_restore(struct socket *sock, const struct genl_info *info)
+{
+       return -ENOSYS;
+}
+#endif /* CONFIG_IP_CR */
+
+#endif
Index: 2.6.20-cr/include/net/sock.h
===================================================================
--- 2.6.20-cr.orig/include/net/sock.h
+++ 2.6.20-cr/include/net/sock.h
@@ -55,6 +55,7 @@
 #include <asm/atomic.h>
 #include <net/dst.h>
 #include <net/checksum.h>
```

```
+#include <net/genetlink.h>

 /*
  * This structure really needs to be cleaned up.
@@ -554,7 +555,10 @@
  void   (*hash)(struct sock *sk);
  void   (*unhash)(struct sock *sk);
  int    (*get_port)(struct sock *sk, unsigned short snum);
-
+#ifdef CONFIG_IP_CR
+ int                    (*dump)(struct socket *sock, struct sk_buff *skb);
+ int                    (*restore)(struct socket *sock, const struct genl_info *info);
+#endif
  /* Memory pressure */
  void   (*enter_memory_pressure)(void);
  atomic_t  *memory_allocated; /* Current allocated memory. */
Index: 2.6.20-cr/net/ipv4/Makefile
===================================================================
--- 2.6.20-cr.orig/net/ipv4/Makefile
+++ 2.6.20-cr/net/ipv4/Makefile
@@ -53,4 +53,4 @@

 obj-$(CONFIG_XFRM) += xfrm4_policy.o xfrm4_state.o xfrm4_input.o \
        xfrm4_output.o
-obj-$(CONFIG_IP_CR) += af_inet_cr.o
+obj-$(CONFIG_IP_CR) += af_inet_cr.o udp_cr.o
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
@@ -15,6 +15,7 @@
 #include <linux/syscalls.h>
 #include <linux/in.h>
 #include <linux/igmp.h>
+#include <linux/udp.h>
 #include <linux/af_inet_cr.h>

 #include <net/ip.h>
@@ -72,6 +73,14 @@
  [AF_INET_CR_ATTR_IPOPT_OPTIONS] =
  { .len = sizeof(struct ip_options) + 40 },

+ /* udp options */
+ [AF_INET_CR_ATTR_UDPOPT_CORK] = { .type = NLA_FLAG },
+
+ /* udp endpoints */
+ [UDP_CR_ATTR_BIND] = { .len =  sizeof(struct sockaddr_in) },
+ [UDP_CR_ATTR_PEER] = { .len =  sizeof(struct sockaddr_in) },
```

```
+ [UDP_CR_ATTR_BIND_ADDR_ULOCK] = { .type = NLA_FLAG },
+ [UDP_CR_ATTR_BIND_PORT_ULOCK] = { .type = NLA_FLAG },
 };

 /*
@@ -513,6 +522,9 @@
  void *msg_head;
  int ret;

+ if (!sk->sk_prot->dump)
+  return -ENOSYS;
+
  if (family != AF_INET)
   return -EINVAL;

@@ -542,6 +554,10 @@
  if (ret)
   goto out;

+ ret = sk->sk_prot->dump(sock, skb);
+ if (ret)
+                goto out;
+
  ret = genlmsg_end(skb, msg_head);
  if (ret < 0)
   goto out;
@@ -726,9 +742,12 @@
 static int restore_socket(struct socket *sock, struct genl_info *info)
 {
  int ret;
-
+ struct sock *sk = sock->sk;
  struct nlattr *nla;

+ if (!sk->sk_prot->dump)
+  return -ENOSYS;
+
  ret = -EINVAL;

  nla = info->attrs[AF_INET_CR_ATTR_SOCK_STATE];
@@ -748,6 +767,8 @@
   ret = restore_inetopt(sock, info);
  if (ret)
   goto out;
+
+ ret = sk->sk_prot->restore(sock, info);
 out:
  return ret;
```

```
 }
Index: 2.6.20-cr/net/ipv4/udp.c
===================================================================
--- 2.6.20-cr.orig/net/ipv4/udp.c
+++ 2.6.20-cr/net/ipv4/udp.c
@@ -93,10 +93,12 @@
 #include <linux/mm.h>
 #include <linux/inet.h>
 #include <linux/netdevice.h>
-#include <net/tcp_states.h>
+#include <linux/udp_cr.h>
 #include <linux/skbuff.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+
+#include <net/tcp_states.h>
 #include <net/icmp.h>
 #include <net/route.h>
 #include <net/checksum.h>
@@ -1523,6 +1525,8 @@
  .unhash     = udp_lib_unhash,
  .get_port    = udp_v4_get_port,
  .obj_size    = sizeof(struct udp_sock),
+ .dump             = udp_dump,
+ .restore          = udp_restore,
 #ifdef CONFIG_COMPAT
  .compat_setsockopt = compat_udp_setsockopt,
  .compat_getsockopt = compat_udp_getsockopt,
Index: 2.6.20-cr/net/ipv4/udp_cr.c
===================================================================
--- /dev/null
+++ 2.6.20-cr/net/ipv4/udp_cr.c
@@ -0,0 +1,119 @@
+/*
+ *  Copyright (C) 2007 IBM Corporation
+ *
+ *  Author: Daniel Lezcano <dlezcano@fr.ibm.com>
+ *
+ *  This program is free software; you can redistribute it and/or
+ *  modify it under the terms of the GNU General Public License as
+ *  published by the Free Software Foundation, version 2 of the
+ *  License.
+ */
+
+
+#include <linux/in.h>
+#include <linux/udp.h>
+#include <linux/udp_cr.h>
```

```
+#include <linux/af_inet_cr.h>
+#include <net/sock.h>
+
+
+/*
+ * udp options association with netlink attribute
+ */
+struct af_inet_cr_optattr udp_options[] = {
+ { UDP_CORK, AF_INET_CR_ATTR_UDPOPT_CORK, BOTH },
+};
+
+int udp_dump(struct socket *sock, struct sk_buff *skb)
+{
+ int i, ret = 0;
+ size_t len;
+ struct sock *sk = sock->sk;
+ struct inet_sock *inet = inet_sk(sk);
+ struct sockaddr_in addr;
+ int addrlen = sizeof(addr);
+
+ len = sizeof(udp_options)/sizeof(struct af_inet_cr_optattr);
+
+ for (i = 0; i < len; i++) {
+  ret = af_inet_cr_opt2attr(skb, sock, SOL_UDP, &udp_options[i]);
+  if (ret)
+   goto out;
+ }
+
+ ret = kernel_getsockname(sock, (struct sockaddr *)&addr, &addrlen);
+ if (ret)
+  goto out;
+
+ if (addr.sin_addr.s_addr || addr.sin_port) {
+  ret = nla_put(skb, UDP_CR_ATTR_BIND, sizeof(addr), &addr);
+  if (ret)
+   goto out;
+ }
+
+ if (sk->sk_userlocks & SOCK_BINDADDR_LOCK) {
+  ret = nla_put_flag(skb, UDP_CR_ATTR_BIND_ADDR_ULOCK);
+  if (ret)
+   goto out;
+ }
+
+ if (sk->sk_userlocks & SOCK_BINDPORT_LOCK) {
+  ret = nla_put_flag(skb, UDP_CR_ATTR_BIND_PORT_ULOCK);
+  if (ret)
+   goto out;
```

```
+ }
+
+ if (inet->dport) {
+  ret = kernel_getpeername(sock, (struct sockaddr *)&addr, &addrlen);
+  if (ret)
+   goto out;
+  ret = nla_put(skb, UDP_CR_ATTR_PEER, sizeof(addr), &addr);
+  if (ret)
+   goto out;
+ }
+out:
+ return ret;
+}
+
+int udp_restore(struct socket *sock, const struct genl_info *info)
+{
+ int i, ret = 0;
+ size_t len;
+ struct nlattr *nla;
+ struct sockaddr_in addr;
+ int addrlen = sizeof(addr);
+
+ len = sizeof(udp_options)/sizeof(struct af_inet_cr_optattr);
+
+ for (i = 0; i < len; i++) {
+  ret = af_inet_cr_attr2opt(info, sock, SOL_UDP, &udp_options[i]);
+  if (ret)
+   goto out;
+ }
+
+ nla = info->attrs[UDP_CR_ATTR_BIND];
+ if (!nla)
+  goto out;
+
+ nla_memcpy(&addr, nla, sizeof(addr));
+
+ ret = kernel_bind(sock, (struct sockaddr *)&addr, addrlen);
+ if (ret)
+  goto out;
+
+ if (!info->attrs[UDP_CR_ATTR_BIND_ADDR_ULOCK])
+  sock->sk->sk_userlocks &= ~SOCK_BINDADDR_LOCK;
+
+ if (!info->attrs[UDP_CR_ATTR_BIND_PORT_ULOCK])
+  sock->sk->sk_userlocks &= ~SOCK_BINDPORT_LOCK;
+
+ nla = info->attrs[UDP_CR_ATTR_PEER];
+ if (!nla)
```

```
+  goto out;
+
+ ret = kernel_connect(sock, (struct sockaddr *)&addr, addrlen, 0);
+ if (ret)
+  goto out;
+out:
+ return ret;
+}
```

--

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers