Subject: [patch 2/5][RFC - ipv4/udp checkpoint/restart] : provide compilation option and genetlink framework
Posted by Daniel Lezcano on Wed, 06 Jun 2007 12:18:08 GMT
View Forum Message <> Reply to Message

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

This patchset provide the AF_INET C/R option in the makefile and a generic
netlink framework for passing the socket messages.

It seems that we are encouraged to use netlink instead of the /proc, /sysfs, ioctls:
http://kerneltrap.org/node/6637

I found that there is a lot of advantages to use the netlink:
 * the protocol is secure
 * the protocol will describe the socket attributes and obviously brings a little
   abstraction layer with the internal kernel structure/data type. This is a good
   way to implement ascendant compatibility (eg. move a socket to a kernel with an upper
   version)
 * the amount of socket informations is variable. With netlink, we don't need to take
   care of the size of the data (eg. just read data until there is something)
 * the netlink attributes can be direclty dumped to disk and reused to restore the socket
   or examined for specific processing (I don't have example).

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

```
---
 include/linux/af_inet_cr.h |   15 +++++
 net/ipv4/Kconfig           |    8 +++
 net/ipv4/Makefile          |    1 
 net/ipv4/af_inet_cr.c      |  119 +++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 143 insertions(+)

Index: 2.6.20-cr/net/ipv4/Kconfig
===================================================================
--- 2.6.20-cr.orig/net/ipv4/Kconfig
+++ 2.6.20-cr/net/ipv4/Kconfig
@@ -1,6 +1,14 @@
 #
 # IP configuration
 #
+config IP_CR
+	tristate "IP: checkpoint/restart"
+	help
+	  The checkpoint/restart allows to dump the sockets states and
+	   the associated protocols internals to the userspace land.
+	   The data can be reused to recreate the socket in the same state.
+	   It's safe to say N.
+
```

```
 config IP_MULTICAST
  bool "IP: multicasting"
  help
Index: 2.6.20-cr/net/ipv4/Makefile
===================================================================
--- 2.6.20-cr.orig/net/ipv4/Makefile
+++ 2.6.20-cr/net/ipv4/Makefile
@@ -53,3 +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
Index: 2.6.20-cr/net/ipv4/af_inet_cr.c
===================================================================
--- /dev/null
+++ 2.6.20-cr/net/ipv4/af_inet_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 <net/genetlink.h>
+#include <net/sock.h>
+#include <linux/fs.h>
+#include <linux/af_inet_cr.h>
+
+/*
+ * af_inet_cr_nldump : this function is called when a netlink message is received
+ * with AF_INET_CR_CMD_DUMP command.
+ * @skb  : the netlink packet giving the restore command
+ * @info : the generic netlink message
+ */
+static int af_inet_cr_nldump(struct sk_buff *skb, struct genl_info *info)
+{
+ return 0;
+}
+
+/*
+ * af_inet_cr_nldump : this function is called when a netlink message is received
+ * with AF_INET_CR_CMD_RESTORE command.
+ * @skb  : the netlink packet giving the restore command
```

```
+ * @info : the generic netlink message
+ */
+static int af_inet_cr_nlrestore(struct sk_buff *skb, struct genl_info *info)
+{
+ return 0;
+}
+
+/*
+ * Netlink message policy definition
+ */
+static struct nla_policy af_inet_cr_policy[AF_INET_CR_ATTR_MAX] = {
+ [AF_INET_CR_ATTR_INODE] = { .type = NLA_U32 },
+};
+
+/*
+ * Netlink dumping command configuration
+ */
+static struct genl_ops af_inet_cr_nldump_ops = {
+ .cmd = AF_INET_CR_CMD_DUMP,
+ .doit = af_inet_cr_nldump,
+ .policy = af_inet_cr_policy,
+};
+
+/*
+ * Netlink restore command configuration
+ */
+static struct genl_ops af_inet_cr_nlrestore_ops = {
+ .cmd = AF_INET_CR_CMD_RESTORE,
+ .doit = af_inet_cr_nlrestore,
+ .policy = af_inet_cr_policy,
+};
+
+/*
+ * Generic netlink family definition
+ */
+static struct genl_family af_inet_cr_family = {
+ .id             = GENL_ID_GENERATE,
+ .name           = "af_inet_cr",
+ .version        = 0x1,
+ .maxattr        = AF_INET_CR_ATTR_MAX - 1,
+};
+
+/*
+ * af_inet_cr_init : this function is called at initialization
+ * time. It register the generic netlink family associated with
+ * this module and hang different ops with it.
+ */
+static __init int af_inet_cr_init(void)
```

```
+{
+ int err;
+
+ err = genl_register_family(&af_inet_cr_family);
+ if (err < 0)
+  goto out;
+
+ err = genl_register_ops(&af_inet_cr_family,
+    &af_inet_cr_nldump_ops);
+ if (err < 0)
+  goto out_unregister_fam;
+
+ err = genl_register_ops(&af_inet_cr_family,
+    &af_inet_cr_nlrestore_ops);
+ if (err < 0)
+  goto out_unregister_dump;
+
+ return 0;
+
+out_unregister_dump:
+ genl_unregister_ops(&af_inet_cr_family, &af_inet_cr_nldump_ops);
+out_unregister_fam:
+ genl_unregister_family(&af_inet_cr_family);
+out:
+ return err;
+}
+
+/*
+ * af_inet_cr_exit : this function is called at exit time. It unregister
+ * the generic netlink family.
+ */
+static __exit void af_inet_cr_exit(void)
+{
+ genl_unregister_ops(&af_inet_cr_family, &af_inet_cr_nlrestore_ops);
+ genl_unregister_ops(&af_inet_cr_family, &af_inet_cr_nldump_ops);
+ genl_unregister_family(&af_inet_cr_family);
+}
+
+module_init(af_inet_cr_init);
+module_exit(af_inet_cr_exit);
Index: 2.6.20-cr/include/linux/af_inet_cr.h
===================================================================
--- /dev/null
+++ 2.6.20-cr/include/linux/af_inet_cr.h
@@ -0,0 +1,15 @@
+#ifndef _AF_INET_CR_H
+#define _AF_INET_CR_H
+enum {
```

```
+ AF_INET_CR_CMD_UNSPEC,
+ AF_INET_CR_CMD_DUMP,
+ AF_INET_CR_CMD_RESTORE,
+ AF_INET_CR_CMD_MAX
+};
+
+enum {
+ AF_INET_CR_ATTR_UNSPEC,
+ AF_INET_CR_ATTR_INODE,
+ AF_INET_CR_ATTR_MAX
+};
+#endif

--
```