Subject: [PATCH] Make ip utility veth driver aware
Posted by xemul on Wed, 02 May 2007 10:55:10 GMT
View Forum Message <> Reply to Message

The new command is called "veth" with the following syntax:
* ip veth add <dev1> <dev2>
  creates interconnected pair of veth devices.
* ip veth del <dev>
  destroys the pair of veth devices, where <dev> is either
  <dev1> or <dev2> used to create the pair.

One question that is to be solved is whether or not to create
a hard-coded netlink family for veth driver. Without it the
family resolution code has to be moved to general place in ip
utility (by now it is copy-paste-ed from one file to another
till final decision).

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/ip/Makefile b/ip/Makefile
index a749993..864e04e 100644
--- a/ip/Makefile
+++ b/ip/Makefile
@@ -1,7 +1,7 @@
 IPOBJ=ip.o ipaddress.o iproute.o iprule.o \
     rtm_map.o iptunnel.o ip6tunnel.o tunnel.o ipneigh.o ipntable.o iplink.o \
     ipmaddr.o ipmonitor.o ipmroute.o ipprefix.o \
-    ipxfrm.o xfrm_state.o xfrm_policy.o xfrm_monitor.o
+    ipxfrm.o xfrm_state.o xfrm_policy.o xfrm_monitor.o veth.o

 RTMONOBJ=rtmon.o

diff --git a/ip/ip.c b/ip/ip.c
index c084292..d3d3a8a 100644
--- a/ip/ip.c
+++ b/ip/ip.c
@@ -27,6 +27,7 @@
 #include "SNAPSHOT.h"
 #include "utils.h"
 #include "ip_common.h"
+#include "veth.h"

 int preferred_family = AF_UNSPEC;
 int show_stats = 0;
@@ -46,7 +47,7 @@ static void usage(void)
 "Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }\n"
```

```
 "       ip [ -force ] [-batch filename\n"
 "where  OBJECT := { link | addr | route | rule | neigh | ntable | tunnel |\n"
-"                   maddr | mroute | monitor | xfrm }\n"
+"                   maddr | mroute | monitor | xfrm | veth }\n"
 "       OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |\n"
 "                    -f[amily] { inet | inet6 | ipx | dnet | link } |\n"
 "                    -o[neline] | -t[imestamp] }\n");
@@ -76,6 +77,7 @@ static const struct cmd {
  { "monitor", do_ipmonitor },
  { "xfrm", do_xfrm },
  { "mroute", do_multiroute },
+ { "veth", do_veth },
  { "help", do_help },
  { 0 }
 };
diff --git a/ip/veth.c b/ip/veth.c
new file mode 100644
index 0000000..d4eecc8
--- /dev/null
+++ b/ip/veth.c
@@ -0,0 +1,196 @@
+/*
+ * veth.c        "ethernet tunnel"
+ *
+ *  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; either version
+ *  2 of the License, or (at your option) any later version.
+ *
+ * Authors: Pavel Emelianov, <xemul@openvz.org>
+ *
+ */
+
+#include <stdio.h>
+#include <string.h>
+#include <unistd.h>
+#include <sys/types.h>
+#include <sys/socket.h>
+#include <linux/genetlink.h>
+
+#include "utils.h"
+#include "veth.h"
+
+#define GENLMSG_DATA(glh)       ((void *)(NLMSG_DATA(glh) + GENL_HDRLEN))
+#define NLA_DATA(na)            ((void *)((char*)(na) + NLA_HDRLEN))
+
+static int do_veth_help(void)
+{
```

```
+ fprintf(stderr, "Usage: ip veth add DEVICE PEER_NAME\n");
+ fprintf(stderr, "              del DEVICE\n");
+ exit(-1);
+}
+
+static int genl_ctrl_resolve_family(const char *family)
+{
+ struct rtnl_handle rth;
+ struct nlmsghdr *nlh;
+ struct genlmsghdr *ghdr;
+ int ret = 0;
+ struct {
+  struct nlmsghdr         n;
+  char                    buf[4096];
+ } req;
+
+ memset(&req, 0, sizeof(req));
+
+ nlh = &req.n;
+ nlh->nlmsg_len = NLMSG_LENGTH(GENL_HDRLEN);
+ nlh->nlmsg_flags = NLM_F_REQUEST | NLM_F_ACK;
+ nlh->nlmsg_type = GENL_ID_CTRL;
+
+ ghdr = NLMSG_DATA(&req.n);
+ ghdr->cmd = CTRL_CMD_GETFAMILY;
+
+ if (rtnl_open_byproto(&rth, 0, NETLINK_GENERIC) < 0) {
+  fprintf(stderr, "Cannot open generic netlink socket\n");
+  exit(1);
+ }
+
+ addattr_l(nlh, 128, CTRL_ATTR_FAMILY_NAME, family, strlen(family) + 1);
+
+ if (rtnl_talk(&rth, nlh, 0, 0, nlh, NULL, NULL) < 0) {
+  fprintf(stderr, "Error talking to the kernel\n");
+  goto errout;
+ }
+
+ {
+  struct rtattr *tb[CTRL_ATTR_MAX + 1];
+  struct genlmsghdr *ghdr = NLMSG_DATA(nlh);
+  int len = nlh->nlmsg_len;
+  struct rtattr *attrs;
+
+  if (nlh->nlmsg_type !=  GENL_ID_CTRL) {
+   fprintf(stderr, "Not a controller message, nlmsg_len=%d "
+    "nlmsg_type=0x%x\n", nlh->nlmsg_len, nlh->nlmsg_type);
+   goto errout;
```

```
+  }
+
+  if (ghdr->cmd != CTRL_CMD_NEWFAMILY) {
+   fprintf(stderr, "Unkown controller command %d\n", ghdr->cmd);
+   goto errout;
+  }
+
+  len -= NLMSG_LENGTH(GENL_HDRLEN);
+
+  if (len < 0) {
+   fprintf(stderr, "wrong controller message len %d\n", len);
+   return -1;
+  }
+
+  attrs = (struct rtattr *) ((char *) ghdr + GENL_HDRLEN);
+  parse_rtattr(tb, CTRL_ATTR_MAX, attrs, len);
+
+  if (tb[CTRL_ATTR_FAMILY_ID] == NULL) {
+   fprintf(stderr, "Missing family id TLV\n");
+   goto errout;
+  }
+
+  ret = *(__u16 *) RTA_DATA(tb[CTRL_ATTR_FAMILY_ID]);
+ }
+
+errout:
+ rtnl_close(&rth);
+ return ret;
+}
+
+static int do_veth_operate(char *dev, char *peer, int cmd)
+{
+ struct rtnl_handle rth;
+ struct nlmsghdr *nlh;
+ struct genlmsghdr *ghdr;
+ struct nlattr *attr;
+ struct  {
+   struct nlmsghdr n;
+   struct genlmsghdr h;
+   char bug[1024];
+ } req;
+ int family, len;
+ int err = 0;
+
+ family = genl_ctrl_resolve_family("veth");
+ if (family == 0) {
+   fprintf(stderr, "veth: Can't resolve family\n");
+   exit(1);
```

```
+ }
+
+ if (rtnl_open_byproto(&rth, 0, NETLINK_GENERIC) < 0)
+  exit(1);
+
+ nlh = &req.n;
+ nlh->nlmsg_len = NLMSG_LENGTH(GENL_HDRLEN);
+ nlh->nlmsg_flags = NLM_F_REQUEST;
+ nlh->nlmsg_type = family;
+ nlh->nlmsg_seq = 0;
+
+ ghdr = &req.h;
+ ghdr->cmd = cmd;
+
+ attr = (struct nlattr *) GENLMSG_DATA(&req);
+ len = strlen(dev);
+ attr->nla_type = VETH_ATTR_DEVNAME;
+ attr->nla_len = len + 1 + NLA_HDRLEN;
+ memcpy(NLA_DATA(attr), dev, len);
+ nlh->nlmsg_len += NLMSG_ALIGN(attr->nla_len);
+
+ if (peer) {
+  attr = (struct nlattr *)((char *)attr +
+    NLMSG_ALIGN(attr->nla_len));
+  len = strlen(peer);
+  attr->nla_type = VETH_ATTR_PEERNAME;
+  attr->nla_len = len + 1 + NLA_HDRLEN;
+  memcpy(NLA_DATA(attr), peer, len);
+  nlh->nlmsg_len += NLMSG_ALIGN(attr->nla_len);
+ }
+
+ if (rtnl_send(&rth, (char *) &req, nlh->nlmsg_len) < 0) {
+  err = -1;
+  fprintf(stderr, "Error talking to the kernel (add)\n");
+ }
+
+ rtnl_close(&rth);
+ return err;
+}
+
+static int do_veth_add(int argc, char **argv)
+{
+ if (argc < 2)
+  return do_veth_help();
+
+ return do_veth_operate(argv[0], argv[1], VETH_CMD_ADD);
+}
+
```

```
+static int do_veth_del(int argc, char **argv)
+{
+ char *name;
+
+ if (argc < 1)
+  return do_veth_help();
+
+ return do_veth_operate(argv[0], NULL, VETH_CMD_DEL);
+}
+
+int do_veth(int argc, char **argv)
+{
+ if (argc == 0)
+  return do_veth_help();
+
+ if (strcmp(*argv, "add") == 0 || strcmp(*argv, "a") == 0)
+  return do_veth_add(argc - 1, argv + 1);
+ if (strcmp(*argv, "del") == 0 || strcmp(*argv, "d") == 0)
+  return do_veth_del(argc - 1, argv + 1);
+ if (strcmp(*argv, "help") == 0)
+  return do_veth_help();
+
+ fprintf(stderr, "Command \"%s\" is unknown, try \"ip veth help\".\n", *argv);
+ exit(-1);
+}
diff --git a/ip/veth.h b/ip/veth.h
new file mode 100644
index 0000000..4d7b357
--- /dev/null
+++ b/ip/veth.h
@@ -0,0 +1,17 @@
+int do_veth(int argc, char **argv);
+
+enum {
+ VETH_CMD_UNSPEC,
+ VETH_CMD_ADD,
+ VETH_CMD_DEL,
+
+ VETH_CMD_MAX
+};
+
+enum {
+ VETH_ATTR_UNSPEC,
+ VETH_ATTR_DEVNAME,
+ VETH_ATTR_PEERNAME,
+
+ VETH_ATTR_MAX
+};
```