
Subject: [patch 3/4] Network namespaces: IPv4 FIB/routing in namespaces
Posted by [Andrey Savochkin](#) on Mon, 26 Jun 2006 09:54:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

Structures related to IPv4 routing (FIB and routing cache)
are made per-namespace.

Signed-off-by: Andrey Savochkin <saw@swsoft.com>

```
---
include/linux/net_ns.h | 9 +++
include/net/flow.h | 3 +
include/net/ip_fib.h | 62 ++++++-----
net/core/dev.c | 7 ++
net/ipv4/Kconfig | 4 -
net/ipv4/fib_frontend.c | 87 ++++++-----
net/ipv4/fib_hash.c | 13 +++-
net/ipv4/fib_rules.c | 114 ++++++-----
net/ipv4/fib_semantics.c | 104 ++++++-----
net/ipv4/route.c | 26 ++++++++
10 files changed, 348 insertions, 81 deletions
```

--- ./include/linux/net_ns.h.vensrt Fri Jun 23 11:49:42 2006

+++ ./include/linux/net_ns.h Fri Jun 23 11:50:16 2006

```
@@ -14,7 +14,16 @@ struct net_namespace {
    atomic_t active_ref, use_ref;
    struct list_head dev_base;
    struct net_device *loopback;
+ #ifndef CONFIG_IP_MULTIPLE_TABLES
+ struct fib_table *fib4_local_table, *fib4_main_table;
+ #else
+ struct fib_table **fib4_tables;
+ struct hlist_head fib4_rules;
+ #endif
+ struct hlist_head *fib4_hash, *fib4_laddrhash;
+ unsigned fib4_hash_size, fib4_info_cnt;
    unsigned int hash;
+ char destroying;
    struct execute_work destroy_work;
};
```

--- ./include/net/flow.h.vensrt Wed Jun 21 18:51:08 2006

+++ ./include/net/flow.h Fri Jun 23 11:50:16 2006

```
@@ -78,6 +78,9 @@ struct flowi {
    #define fl_icmp_type uli_u.icmpt.type
    #define fl_icmp_code uli_u.icmpt.code
    #define fl_ipsec_spi uli_u.spi
+ #ifdef CONFIG_NET_NS
+ struct net_namespace *net_ns;
```

```

+#endif
} __attribute__((__aligned__(BITS_PER_LONG/8)));

#define FLOW_DIR_IN 0
--- ./include/net/ip_fib.h.vensrt Wed Jun 21 18:53:17 2006
+++ ./include/net/ip_fib.h Fri Jun 23 11:50:16 2006
@@ -18,6 +18,7 @@

#include <net/flow.h>
#include <linux/seq_file.h>
#include <linux/net_ns.h>

/* WARNING: The ordering of these elements must match ordering
 * of RTA_* rtnetlink attribute numbers.
@@ -169,14 +170,21 @@ struct fib_table {

#ifdef CONFIG_IP_MULTIPLE_TABLES

extern struct fib_table *ip_fib_local_table;
extern struct fib_table *ip_fib_main_table;
#ifdef CONFIG_NET_NS
extern struct fib_table *ip_fib_local_table_static;
extern struct fib_table *ip_fib_main_table_static;
#define ip_fib_local_table_ns() ip_fib_local_table_static
#define ip_fib_main_table_ns() ip_fib_main_table_static
#else
#define ip_fib_local_table_ns() (current_net_ns->fib4_local_table)
#define ip_fib_main_table_ns() (current_net_ns->fib4_main_table)
#endif

static inline struct fib_table *fib_get_table(int id)
{
    if (id != RT_TABLE_LOCAL)
-   return ip_fib_main_table;
-   return ip_fib_local_table;
+   return ip_fib_main_table_ns();
+   return ip_fib_local_table_ns();
}

static inline struct fib_table *fib_new_table(int id)
@@ -186,23 +194,36 @@ static inline struct fib_table *fib_new_

static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
{
-   if (ip_fib_local_table->tb_lookup(ip_fib_local_table, flp, res) &&
-       ip_fib_main_table->tb_lookup(ip_fib_main_table, flp, res))
+   struct fib_table *tb;
+

```

```

+ tb = ip_fib_local_table_ns();
+ if (!tb->tb_lookup(tb, flp, res))
+ return 0;
+ tb = ip_fib_main_table_ns();
+ if (tb->tb_lookup(tb, flp, res))
+ return -ENETUNREACH;
+ return 0;
}

static inline void fib_select_default(const struct flowi *flp, struct fib_result *res)
{
+ struct fib_table *tb;
+
+ tb = ip_fib_main_table_ns();
+ if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
- ip_fib_main_table->tb_select_default(ip_fib_main_table, flp, res);
+ tb->tb_select_default(main_table, flp, res);
}

#else /* CONFIG_IP_MULTIPLE_TABLES */
-#define ip_fib_local_table (fib_tables[RT_TABLE_LOCAL])
-#define ip_fib_main_table (fib_tables[RT_TABLE_MAIN])
+#define ip_fib_local_table_ns() (fib_tables_ns())[RT_TABLE_LOCAL]
+#define ip_fib_main_table_ns() (fib_tables_ns())[RT_TABLE_MAIN]

-extern struct fib_table * fib_tables[RT_TABLE_MAX+1];
+#ifndef CONFIG_NET_NS
+extern struct fib_table * fib_tables_static[RT_TABLE_MAX+1];
+#define fib_tables_ns() fib_tables_static
+#else
+#define fib_tables_ns() (current_net_ns->fib4_tables)
+#endif
extern int fib_lookup(const struct flowi *flp, struct fib_result *res);
extern struct fib_table * __fib_new_table(int id);
extern void fib_rule_put(struct fib_rule *r);
@@ -212,7 +233,7 @@ static inline struct fib_table *fib_get_
if (id == 0)
id = RT_TABLE_MAIN;

- return fib_tables[id];
+ return fib_tables_ns()[id];
}

static inline struct fib_table *fib_new_table(int id)
@@ -220,7 +241,7 @@ static inline struct fib_table *fib_new_
if (id == 0)
id = RT_TABLE_MAIN;

```

```

- return fib_tables[id] ? : __fib_new_table(id);
+ return fib_tables_ns()[id] ? : __fib_new_table(id);
}

extern void fib_select_default(const struct flowi *flp, struct fib_result *res);
@@ -229,6 +250,10 @@ extern void fib_select_default(const str

/* Exported by fib_frontend.c */
extern void ip_fib_init(void);
#ifdef CONFIG_NET_NS
+extern int ip_fib_struct_init(void);
+extern void ip_fib_struct_fini(void);
#endif
extern int inet_rtm_delroute(struct sk_buff *skb, struct nlmsg_hdr* nlh, void *arg);
extern int inet_rtm_newroute(struct sk_buff *skb, struct nlmsg_hdr* nlh, void *arg);
extern int inet_rtm_getroute(struct sk_buff *skb, struct nlmsg_hdr* nlh, void *arg);
@@ -246,9 +271,16 @@ extern int fib_sync_up(struct net_device
extern int fib_convert_rtnetentry(int cmd, struct nlmsg_hdr* nl, struct rtmsg *rtm,
    struct kern_rta *rta, struct rtnetentry *r);
extern u32 __fib_res_prefsrc(struct fib_result *res);
#ifdef CONFIG_NET_NS
+extern void fib_hashtable_destroy(void);
#endif

/* Exported by fib_hash.c */
extern struct fib_table *fib_hash_init(int id);
#ifdef CONFIG_NET_NS
+extern void fib_hash_fini(struct fib_table *tb);
+extern void fib_hash_destroy_hash(void);
#endif

#ifdef CONFIG_IP_MULTIPLE_TABLES
/* Exported by fib_rules.c */
@@ -259,7 +291,11 @@ extern int inet_dump_rules(struct sk_buff
#ifdef CONFIG_NET_CLS_ROUTE
extern u32 fib_rules_tclass(struct fib_result *res);
#endif
-extern void fib_rules_init(void);
+extern int fib_rules_struct_init(void);
+extern void fib_rules_notif_init(void);
#ifdef CONFIG_NET_NS
+extern void fib_rules_struct_fini(void);
#endif
#endif

static inline void fib_combine_itag(u32 *itag, struct fib_result *res)
--- ./net/core/dev.c.vensrt Fri Jun 23 11:48:15 2006
+++ ./net/core/dev.c Fri Jun 23 11:50:16 2006

```

```

@@ -3398,6 +3398,8 @@ static int __init netdev_dma_register(vo
#endif /* CONFIG_NET_DMA */

#ifdef CONFIG_NET_NS
#include <net/ip_fib.h>
+
struct net_namespace init_net_ns = {
    .active_ref = ATOMIC_INIT(2),
    /* one for init_task->net_context,
@@ -3436,6 +3438,8 @@ int net_ns_start(void)
    task = current;
    orig_ns = task->net_context;
    task->net_context = ns;
+ if (ip_fib_struct_init())
+ goto out_fib4;
    err = register_netdev(dev);
    if (err)
        goto out_register;
@@ -3443,6 +3447,8 @@ int net_ns_start(void)
    return 0;

out_register:
+ ip_fib_struct_fini();
+out_fib4:
    dev->destructor(dev);
    task->net_context = orig_ns;
    BUG_ON(atomic_read(&ns->active_ref) != 1);
@@ -3467,6 +3473,7 @@ static void net_ns_destroy(void *data)
    ns = data;
    push_net_ns(ns, orig_ns);
    unregister_netdev(ns->loopback);
+ ip_fib_struct_fini();
    BUG_ON(!list_empty(&ns->dev_base));
    pop_net_ns(orig_ns);

--- ./net/ipv4/Kconfig.vensrt Wed Jun 21 18:53:19 2006
+++ ./net/ipv4/Kconfig Fri Jun 23 11:50:16 2006
@@ -53,7 +53,7 @@ config IP_ADVANCED_ROUTER

choice
    prompt "Choose IP: FIB lookup algorithm (choose FIB_HASH if unsure)"
- depends on IP_ADVANCED_ROUTER
+ depends on IP_ADVANCED_ROUTER && !NET_NS
    default ASK_IP_FIB_HASH

config ASK_IP_FIB_HASH
@@ -83,7 +83,7 @@ config IP_FIB_TRIE
endchoice

```

```

config IP_FIB_HASH
- def_bool ASK_IP_FIB_HASH || !IP_ADVANCED_ROUTER
+ def_bool ASK_IP_FIB_HASH || !IP_ADVANCED_ROUTER || NET_NS

```

```

config IP_MULTIPLE_TABLES
    bool "IP: policy routing"
--- ./net/ipv4/fib_frontend.c.vensrt Wed Jun 21 18:53:19 2006
+++ ./net/ipv4/fib_frontend.c Fri Jun 23 11:50:16 2006
@@ -53,14 +53,18 @@

```

```

#define RT_TABLE_MIN RT_TABLE_MAIN

```

```

-struct fib_table *ip_fib_local_table;
-struct fib_table *ip_fib_main_table;
+#ifndef CONFIG_NET_NS
+struct fib_table *ip_fib_local_table_static;
+struct fib_table *ip_fib_main_table_static;
+#endif

```

```

#else

```

```

#define RT_TABLE_MIN 1

```

```

-struct fib_table *fib_tables[RT_TABLE_MAX+1];
+#ifndef CONFIG_NET_NS
+struct fib_table *fib_tables_static[RT_TABLE_MAX+1];
+#endif

```

```

struct fib_table * __fib_new_table(int id)
{
@@ -69,7 +73,7 @@ struct fib_table * __fib_new_table(int id
    tb = fib_hash_init(id);
    if (!tb)
        return NULL;
- fib_tables[id] = tb;
+ fib_tables_ns()[id] = tb;
    return tb;
}

```

```

@@ -80,8 +84,8 @@ struct fib_table * __fib_new_table(int id
static void fib_flush(void)
{
    int flushed = 0;
-#ifdef CONFIG_IP_MULTIPLE_TABLES
    struct fib_table *tb;
+#ifdef CONFIG_IP_MULTIPLE_TABLES
    int id;

```

```

    for (id = RT_TABLE_MAX; id>0; id--) {
@@ -90,8 +94,10 @@ static void fib_flush(void)
    flushed += tb->tb_flush(tb);
    }
    #else /* CONFIG_IP_MULTIPLE_TABLES */
- flushed += ip_fib_main_table->tb_flush(ip_fib_main_table);
- flushed += ip_fib_local_table->tb_flush(ip_fib_local_table);
+ tb = ip_fib_main_table_ns();
+ flushed += tb->tb_flush(tb);
+ tb = ip_fib_local_table_ns();
+ flushed += tb->tb_flush(tb);
    #endif /* CONFIG_IP_MULTIPLE_TABLES */

    if (flushed)
@@ -106,14 +112,15 @@ struct net_device * ip_dev_find(u32 addr
    {
        struct flowi fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
        struct fib_result res;
+ struct fib_table *tb;
        struct net_device *dev = NULL;

        #ifdef CONFIG_IP_MULTIPLE_TABLES
            res.r = NULL;
        #endif

- if (!ip_fib_local_table ||
-     ip_fib_local_table->tb_lookup(ip_fib_local_table, &fl, &res))
+ tb = ip_fib_local_table_ns();
+ if (!tb || tb->tb_lookup(tb, &fl, &res))
            return NULL;
        if (res.type != RTN_LOCAL)
            goto out;
@@ -130,6 +137,7 @@ unsigned inet_addr_type(u32 addr)
    {
        struct flowi fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
        struct fib_result res;
+ struct fib_table *tb;
        unsigned ret = RTN_BROADCAST;

        if (ZERONET(addr) || BADCLASS(addr))
@@ -141,10 +149,10 @@ unsigned inet_addr_type(u32 addr)
        res.r = NULL;
    #endif

- if (ip_fib_local_table) {
+ if (ip_fib_local_table) {
+ tb = ip_fib_local_table_ns();
+ if (tb) {

```

```

    ret = RTN_UNICAST;
- if (!ip_fib_local_table->tb_lookup(ip_fib_local_table,
-     &fl, &res)) {
+ if (!tb->tb_lookup(tb, &fl, &res)) {
    ret = res.type;
    fib_res_put(&res);
}
@@ -651,19 +659,66 @@ static struct notifier_block fib_netdev_
    .notifier_call = fib_netdev_event,
};

-void __init ip_fib_init(void)
+int ip_fib_struct_init(void)
{
    #ifndef CONFIG_IP_MULTIPLE_TABLES
- ip_fib_local_table = fib_hash_init(RT_TABLE_LOCAL);
- ip_fib_main_table = fib_hash_init(RT_TABLE_MAIN);
+ ip_fib_local_table_ns() = fib_hash_init(RT_TABLE_LOCAL);
+ ip_fib_main_table_ns() = fib_hash_init(RT_TABLE_MAIN);
+ #else
+ #ifndef CONFIG_NET_NS
+ return fib_rules_struct_init();
+ #else
- fib_rules_init();
+ struct fib_table **tables;
+
+ tables = kmalloc((RT_TABLE_MAX+1) * sizeof(*tables), GFP_KERNEL);
+ if (tables == NULL)
+ return -ENOMEM;
+ memset(tables, 0, (RT_TABLE_MAX+1) * sizeof(*tables));
+ fib_tables_ns() = tables;
+ if (fib_rules_struct_init()) {
+ kfree(tables);
+ fib_tables_ns() = NULL;
+ return -ENOMEM;
+ }
    #endif
+ #endif
+ return 0;
+}

+void __init ip_fib_init(void)
+{
+ ip_fib_struct_init();
+
+ #ifdef CONFIG_IP_MULTIPLE_TABLES
+ fib_rules_notif_init();
+ #endif

```

```

register_netdevice_notifier(&fib_netdev_notifier);
register_inetaddr_notifier(&fib_inetaddr_notifier);
nl_fib_lookup_init();
}

#ifdef CONFIG_NET_NS
void ip_fib_struct_fini(void)
+{
+ current_net_ns->destroying = 1;
+ rtnl_lock();
#ifdef CONFIG_IP_MULTIPLE_TABLES
+ fib_rules_struct_fini();
#endif
+ /*
+  * FIB should already be empty since there is no netdevice,
+  * but clear it anyway
+  */
+ fib_flush();
+ rt_cache_flush(0);
#ifdef CONFIG_IP_MULTIPLE_TABLES
+ kfree(fib_tables_ns());
+ fib_tables_ns() = NULL;
#endif
+ fib_hashtable_destroy();
+ rtnl_unlock();
+}
#endif /* CONFIG_NET_NS */
+
EXPORT_SYMBOL(inet_addr_type);
EXPORT_SYMBOL(ip_dev_find);
--- ./net/ipv4/fib_hash.c.vensrt Mon Mar 20 08:53:29 2006
+++ ./net/ipv4/fib_hash.c Fri Jun 23 11:50:16 2006
@@ -629,6 +629,11 @@ static int fn_flush_list(struct fn_zone
struct hlist_node *node, *n;
struct fib_node *f;
int found = 0;
#ifdef CONFIG_NET_NS
+ const int destroy = 0;
#else
+ const int destroy = current_net_ns->destroying;
#endif

hlist_for_each_entry_safe(f, node, n, head, fn_hash) {
struct fib_alias *fa, *fa_node;
@@ -638,7 +643,9 @@ static int fn_flush_list(struct fn_zone
list_for_each_entry_safe(fa, fa_node, &f->fn_alias, fa_list) {
struct fib_info *fi = fa->fa_info;

```

```

- if (fi && (fi->fib_flags&RTNH_F_DEAD)) {
+ if (fi == NULL)
+ continue;
+ if (destroy || (fi->fib_flags&RTNH_F_DEAD)) {
    write_lock_bh(&fib_hash_lock);
    list_del(&fa->fa_list);
    if (list_empty(&f->fn_alias)) {
@@ -819,7 +826,7 @@ struct fib_iter_state {
static struct fib_alias *fib_get_first(struct seq_file *seq)
{
    struct fib_iter_state *iter = seq->private;
- struct fn_hash *table = (struct fn_hash *) ip_fib_main_table->tb_data;
+ struct fn_hash *table = (struct fn_hash *) ip_fib_main_table_ns()->tb_data;

    iter->bucket = 0;
    iter->hash_head = NULL;
@@ -958,7 +965,7 @@ static void *fib_seq_start(struct seq_file
void *v = NULL;

    read_lock(&fib_hash_lock);
- if (ip_fib_main_table)
+ if (ip_fib_main_table_ns())
    v = *pos ? fib_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
    return v;
}
--- ./net/ipv4/fib_rules.c.vensrt Wed Jun 21 18:51:09 2006
+++ ./net/ipv4/fib_rules.c Fri Jun 23 11:50:16 2006
@@ -100,7 +100,12 @@ static struct fib_rule local_rule = {
    .r_action = RTN_UNICAST,
};

-static struct hlist_head fib_rules;
+#ifndef CONFIG_NET_NS
+static struct hlist_head fib_rules_static;
+#define fib_rules_ns() (&fib_rules_static)
+#else
+#define fib_rules_ns() (&current_net_ns->fib4_rules)
+#endif

/* writer func called from netlink -- rtnl_sem hold*/

@@ -110,11 +115,13 @@ int inet_rtm_delrule(struct sk_buff *skb
{
    struct rtattr **rta = arg;
    struct rtmmsg *rtm = NLMSG_DATA(nlh);
+ struct hlist_head *fib_rules;
    struct fib_rule *r;
    struct hlist_node *node;

```

```

int err = -ESRCH;

- hlist_for_each_entry(r, node, &fib_rules, hlist) {
+ fib_rules = fib_rules_ns();
+ hlist_for_each_entry(r, node, fib_rules, hlist) {
    if ((!rta[RTA_SRC-1] || memcmp(RTA_DATA(rta[RTA_SRC-1]), &r->r_src, 4) == 0) &&
        rtm->rtm_src_len == r->r_src_len &&
        rtm->rtm_dst_len == r->r_dst_len &&
@@ -128,7 +135,7 @@ int inet_rtm_delrule(struct sk_buff *skb
    (!rta[RTA_IIF-1] || rtattr_strcmp(rta[RTA_IIF-1], r->r_ifname) == 0) &&
    (!rtm->rtm_table || (r && rtm->rtm_table == r->r_table))) {
    err = -EPERM;
-   if (r == &local_rule)
+   if (&r->hlist == fib_rules->first)
        break;

    hlist_del_rcu(&r->hlist);
@@ -147,9 +154,11 @@ int inet_rtm_delrule(struct sk_buff *skb
static struct fib_table *fib_empty_table(void)
{
    int id;
+ struct fib_table **tbs;

+ tbs = fib_tables_ns();
    for (id = 1; id <= RT_TABLE_MAX; id++)
-   if (fib_tables[id] == NULL)
+   if (tbs[id] == NULL)
        return __fib_new_table(id);
    return NULL;
}
@@ -176,6 +185,7 @@ int inet_rtm_newrule(struct sk_buff *skb
{
    struct rtattr **rta = arg;
    struct rtmmsg *rtm = NLMSG_DATA(nlh);
+ struct hlist_head *fib_rules;
    struct fib_rule *r, *new_r, *last = NULL;
    struct hlist_node *node = NULL;
    unsigned char table_id;
@@ -234,7 +244,8 @@ int inet_rtm_newrule(struct sk_buff *skb
    if (rta[RTA_FLOW-1])
        memcpy(&new_r->r_tclassid, RTA_DATA(rta[RTA_FLOW-1]), 4);
#ifdef
- r = container_of(fib_rules.first, struct fib_rule, hlist);
+ fib_rules = fib_rules_ns();
+ r = container_of(fib_rules->first, struct fib_rule, hlist);

    if (!new_r->r_preference) {
        if (r && r->hlist.next != NULL) {

```

```

@@ -244,7 +255,7 @@ int inet_rtm_newrule(struct sk_buff *skb
}
}

- hlist_for_each_entry(r, node, &fib_rules, hlist) {
+ hlist_for_each_entry(r, node, fib_rules, hlist) {
    if (r->r_preference > new_r->r_preference)
        break;
    last = r;
@@ -273,10 +284,12 @@ u32 fib_rules_tclass(struct fib_result *

static void fib_rules_detach(struct net_device *dev)
{
+ struct hlist_head *fib_rules;
  struct hlist_node *node;
  struct fib_rule *r;

- hlist_for_each_entry(r, node, &fib_rules, hlist) {
+ fib_rules = fib_rules_ns();
+ hlist_for_each_entry(r, node, fib_rules, hlist) {
    if (r->r_ifindex == dev->ifindex)
        r->r_ifindex = -1;

@@ -287,10 +300,12 @@ static void fib_rules_detach(struct net_

static void fib_rules_attach(struct net_device *dev)
{
+ struct hlist_head *fib_rules;
  struct hlist_node *node;
  struct fib_rule *r;

- hlist_for_each_entry(r, node, &fib_rules, hlist) {
+ fib_rules = fib_rules_ns();
+ hlist_for_each_entry(r, node, fib_rules, hlist) {
    if (r->r_ifindex == -1 && strcmp(dev->name, r->r_ifname) == 0)
        r->r_ifindex = dev->ifindex;
    }
@@ -299,6 +314,7 @@ static void fib_rules_attach(struct net_
int fib_lookup(const struct flowi *flp, struct fib_result *res)
{
    int err;
+ struct hlist_head *fib_rules;
  struct fib_rule *r, *policy;
  struct fib_table *tb;
  struct hlist_node *node;
@@ -311,7 +327,8 @@ FRprintf("Lookup: %u.%u.%u.%u <- %u.%u.%

    rcu_read_lock();

```



```

+ if (p == NULL)
+ goto out;
+ memcpy(p, orig, sizeof(*p));
+ if (prev != NULL)
+ hlist_add_after(&prev->hlist, &p->hlist);
+ else
+ hlist_add_head(&p->hlist, fib_rules_ns());
+out:
+ return p;
+}
+
+int fib_rules_struct_init(void)
+{
+ struct hlist_head *fib_rules;
+ struct fib_rule *p, *q;
+
+ fib_rules = fib_rules_ns();
+ INIT_HLIST_HEAD(fib_rules);
+ p = fib_rule_create(&local_rule, NULL);
+ if (p == NULL)
+ goto out_rule;
+ q = fib_rule_create(&main_rule, p);
+ if (q == NULL)
+ goto out_rule;
+ p = q;
+ q = fib_rule_create(&default_rule, p);
+ if (q == NULL)
+ goto out_rule;
+ return 0;
+
+out_rule:
+ while (!hlist_empty(fib_rules)) {
+ p = hlist_entry(fib_rules->first, struct fib_rule, hlist);
+ hlist_del(&p->hlist);
+ kfree(p);
+ }
+ return -ENOMEM;
+}
+
+void fib_rules_struct_fini(void)
+{
+ struct fib_rule *r, *nxt;
+
+ for (r = hlist_entry(fib_rules_ns()->first, struct fib_rule, hlist);
+  r != NULL; r = nxt) {
+  nxt = hlist_entry(r->hlist.next, struct fib_rule, hlist);
+  hlist_del_rcu(&r->hlist);
+  r->r_dead = 1;

```



```

void free_fib_info(struct fib_info *fi)
{
+ struct net_namespace *ns __attribute_used__ = current_net_ns;
+
  if (fi->fib_dead == 0) {
    printk("Freeing alive fib_info %p\n", fi);
    return;
@@ -154,7 +168,7 @@ void free_fib_info(struct fib_info *fi)
    dev_put(nh->nh_dev);
    nh->nh_dev = NULL;
  } endfor_nexthops(fi);
- fib_info_cnt--;
+ fib_info_cnt(ns)--;
  kfree(fi);
}

@@ -197,9 +211,10 @@ static __inline__ int nh_comp(const stru
  return 0;
}

-static inline unsigned int fib_info_hashfn(const struct fib_info *fi)
+static inline unsigned int fib_info_hashfn(const struct fib_info *fi,
+ struct net_namespace *ns)
{
- unsigned int mask = (fib_hash_size - 1);
+ unsigned int mask = (fib_hash_size(ns) - 1);
  unsigned int val = fi->fib_nhs;

  val ^= fi->fib_protocol;
@@ -211,13 +226,14 @@ static inline unsigned int fib_info_hash

static struct fib_info *fib_find_info(const struct fib_info *nfi)
{
+ struct net_namespace *ns = current_net_ns;
  struct hlist_head *head;
  struct hlist_node *node;
  struct fib_info *fi;
  unsigned int hash;

- hash = fib_info_hashfn(nfi);
- head = &fib_info_hash[hash];
+ hash = fib_info_hashfn(nfi, ns);
+ head = &fib_info_hash(ns)[hash];

  hlist_for_each_entry(fi, node, head, fib_hash) {
    if (fi->fib_nhs != nfi->fib_nhs)
@@ -237,11 +253,15 @@ static struct fib_info *fib_find_info(co

```

```

static inline unsigned int fib_devindex_hashfn(unsigned int val)
{
- unsigned int mask = DEVINDEX_HASHSIZE - 1;
+ unsigned int r, mask = DEVINDEX_HASHSIZE - 1;

- return (val ^
+ r = val ^
    (val >> DEVINDEX_HASHBITS) ^
- (val >> (DEVINDEX_HASHBITS * 2))) & mask;
+ (val >> (DEVINDEX_HASHBITS * 2));
+ #ifdef CONFIG_NET_NS
+ r ^= current_net_ns->hash;
+ #endif
+ return r & mask;
}

/* Check, that the gateway is already configured.
@@ -564,9 +584,9 @@ out:
    return 0;
}

-static inline unsigned int fib_laddr_hashfn(u32 val)
+static inline unsigned int fib_laddr_hashfn(u32 val, struct net_namespace *ns)
{
- unsigned int mask = (fib_hash_size - 1);
+ unsigned int mask = (fib_hash_size(ns) - 1);

    return (val ^ (val >> 7) ^ (val >> 14)) & mask;
}
@@ -595,17 +615,18 @@ static void fib_hash_move(struct hlist_h
    struct hlist_head *new_laddrhash,
    unsigned int new_size)
{
+ struct net_namespace *ns = current_net_ns;
    struct hlist_head *old_info_hash, *old_laddrhash;
- unsigned int old_size = fib_hash_size;
+ unsigned int old_size = fib_hash_size(ns);
    unsigned int i, bytes;

    write_lock(&fib_info_lock);
- old_info_hash = fib_info_hash;
- old_laddrhash = fib_info_laddrhash;
- fib_hash_size = new_size;
+ old_info_hash = fib_info_hash(ns);
+ old_laddrhash = fib_info_laddrhash(ns);
+ fib_hash_size(ns) = new_size;

    for (i = 0; i < old_size; i++) {

```

```

- struct hlist_head *head = &fib_info_hash[i];
+ struct hlist_head *head = &old_info_hash[i];
  struct hlist_node *node, *n;
  struct fib_info *fi;

```

```

@@ -615,15 +636,15 @@ static void fib_hash_move(struct hlist_h

```

```

    hlist_del(&fi->fib_hash);

- new_hash = fib_info_hashfn(fi);
+ new_hash = fib_info_hashfn(fi, ns);
  dest = &new_info_hash[new_hash];
  hlist_add_head(&fi->fib_hash, dest);
}
}
- fib_info_hash = new_info_hash;
+ fib_info_hash(ns) = new_info_hash;

```

```

  for (i = 0; i < old_size; i++) {
- struct hlist_head *lhead = &fib_info_laddrhash[i];
+ struct hlist_head *lhead = &old_laddrhash[i];
  struct hlist_node *node, *n;
  struct fib_info *fi;

```

```

@@ -633,12 +654,12 @@ static void fib_hash_move(struct hlist_h

```

```

    hlist_del(&fi->fib_lhash);

- new_hash = fib_laddr_hashfn(fi->fib_prefsrc);
+ new_hash = fib_laddr_hashfn(fi->fib_prefsrc, ns);
  ldest = &new_laddrhash[new_hash];
  hlist_add_head(&fi->fib_lhash, ldest);
}
}
- fib_info_laddrhash = new_laddrhash;
+ fib_info_laddrhash(ns) = new_laddrhash;

```

```

  write_unlock(&fib_info_lock);

```

```

@@ -647,11 +668,27 @@ static void fib_hash_move(struct hlist_h
  fib_hash_free(old_laddrhash, bytes);
}

```

```

+#ifdef CONFIG_NET_NS
+void fib_hashtable_destroy(void)
+{
+ struct net_namespace *ns;
+ unsigned int bytes;

```

```

+
+ ns = current_net_ns;
+ bytes = ns->fib4_hash_size * sizeof(struct hlist_head *);
+ fib_hash_free(ns->fib4_hash, bytes);
+ ns->fib4_hash = NULL;
+ fib_hash_free(ns->fib4_laddrhash, bytes);
+ ns->fib4_laddrhash = NULL;
+}
+
+
+ struct fib_info *
+ fib_create_info(const struct rtmsg *r, struct kern_rta *rta,
+   const struct nlmsgghdr *nlh, int *errp)
+ {
+   int err;
+
+   struct net_namespace *ns = current_net_ns;
+   struct fib_info *fi = NULL;
+   struct fib_info *ofi;
+   #ifdef CONFIG_IP_ROUTE_MULTIPATH
+   @@ -685,8 +722,8 @@ fib_create_info(const struct rtmsg *r, s
+   #endif
+
+   err = -ENOBUFS;
+   - if (fib_info_cnt >= fib_hash_size) {
+   -   unsigned int new_size = fib_hash_size << 1;
+   + if (fib_info_cnt(ns) >= fib_hash_size(ns)) {
+   +   unsigned int new_size = fib_hash_size(ns) << 1;
+   +   struct hlist_head *new_info_hash;
+   +   struct hlist_head *new_laddrhash;
+   +   unsigned int bytes;
+   @@ -706,14 +743,14 @@ fib_create_info(const struct rtmsg *r, s
+   +   fib_hash_move(new_info_hash, new_laddrhash, new_size);
+   +   }
+
+   - if (!fib_hash_size)
+   + if (!fib_hash_size(ns))
+   +   goto failure;
+   }
+
+   fi = kmalloc(sizeof(*fi)+nhs*sizeof(struct fib_nh), GFP_KERNEL);
+   if (fi == NULL)
+   +   goto failure;
+   - fib_info_cnt++;
+   + fib_info_cnt(ns)++;
+   + memset(fi, 0, sizeof(*fi)+nhs*sizeof(struct fib_nh));
+
+   fi->fib_protocol = r->rtm_protocol;
+   @@ -824,11 +861,11 @@ link_it:

```

```

atomic_inc(&fi->fib_clntref);
write_lock(&fib_info_lock);
hlist_add_head(&fi->fib_hash,
-      &fib_info_hash[fi->fib_info_hashfn(fi)]);
+      &fib_info_hash(ns)[fi->fib_info_hashfn(fi, ns)]);
if (fi->fib_prefsrc) {
    struct hlist_head *head;

-   head = &fib_info_laddrhash[fi->fib_laddr_hashfn(fi->fib_prefsrc)];
+   head = &fib_info_laddrhash(ns)[fi->fib_laddr_hashfn(fi->fib_prefsrc, ns)];
    hlist_add_head(&fi->fib_lhash, head);
}
change_nexthops(fi) {
@@ -1162,15 +1199,16 @@ fib_convert_rtentry(int cmd, struct nlms

int fib_sync_down(u32 local, struct net_device *dev, int force)
{
+ struct net_namespace *ns = current_net_ns;
    int ret = 0;
    int scope = RT_SCOPE_NOWHERE;

    if (force)
        scope = -1;

-   if (local && fib_info_laddrhash) {
-       unsigned int hash = fib_laddr_hashfn(local);
-       struct hlist_head *head = &fib_info_laddrhash[hash];
+   if (local && fib_info_laddrhash(ns)) {
+       unsigned int hash = fib_laddr_hashfn(local, ns);
+       struct hlist_head *head = &fib_info_laddrhash(ns)[hash];
        struct hlist_node *node;
        struct fib_info *fi;

--- ./net/ipv4/route.c.vensrt Wed Jun 21 18:53:19 2006
+++ ./net/ipv4/route.c Fri Jun 23 11:50:16 2006
@@ -267,6 +267,7 @@ struct rt_cache_iter_state {
    int bucket;
};

+static struct rtable *rt_cache_get_next(struct seq_file *seq, struct rtable *r);
static struct rtable *rt_cache_get_first(struct seq_file *seq)
{
    struct rtable *r = NULL;
@@ -279,21 +280,28 @@ static struct rtable *rt_cache_get_first
    break;
    rcu_read_unlock_bh();
}
+ if (r && !net_ns_same(r->fl.net_ns, current_net_ns))

```

```

+ r = rt_cache_get_next(seq, r);
  return r;
}

static struct rtable *rt_cache_get_next(struct seq_file *seq, struct rtable *r)
{
  struct rt_cache_iter_state *st = rcu_dereference(seq->private);
+ struct net_namespace *ns __attribute_used__ = current_net_ns;

+next:
  r = r->u.rt_next;
  while (!r) {
    rcu_read_unlock_bh();
    if (--st->bucket < 0)
-   break;
+   goto out;
    rcu_read_lock_bh();
    r = rt_hash_table[st->bucket].chain;
  }
+ if (!net_ns_same(r->fl.net_ns, ns))
+   goto next;
+out:
  return r;
}

@@ -564,6 +572,7 @@ static inline u32 rt_score(struct rtable
static inline int compare_keys(struct flowi *fl1, struct flowi *fl2)
{
  return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 &&
+   net_ns_same(fl1->net_ns, fl2->net_ns) &&
+   fl1->oif == fl2->oif &&
+   fl1->iif == fl2->iif;
}

@@ -1127,6 +1136,7 @@ void ip_rt_redirect(u32 old_gw, u32 daddr
  struct rtable *rth, **rthp;
  u32 skeys[2] = { saddr, 0 };
  int ikeys[2] = { dev->ifindex, 0 };
+ struct net_namespace *ns __attribute_used__ = current_net_ns;

  if (!in_dev)
    return;

@@ -1158,6 +1168,7 @@ void ip_rt_redirect(u32 old_gw, u32 daddr

  if (rth->fl.fl4_dst != daddr ||
      rth->fl.fl4_src != skeys[i] ||
+   !net_ns_same(rth->fl.net_ns, ns) ||
      rth->fl.oif != ikeys[k] ||
      rth->fl.iif != 0) {

```

```

    rthp = &rth->u.rt_next;
@@ -1643,6 +1654,9 @@ static int ip_route_input_mc(struct sk_b
    dev_hold(rth->u.dst.dev);
    rth->idev = in_dev_get(rth->u.dst.dev);
    rth->fl.oif = 0;
#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
#endif
    rth->rt_gateway = daddr;
    rth->rt_spec_dst= spec_dst;
    rth->rt_type = RTN_MULTICAST;
@@ -1786,6 +1800,9 @@ static inline int __mkroute_input(struct
    dev_hold(rth->u.dst.dev);
    rth->idev = in_dev_get(rth->u.dst.dev);
    rth->fl.oif = 0;
#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
#endif
    rth->rt_spec_dst= spec_dst;

    rth->u.dst.input = ip_forward;
@@ -2087,6 +2104,7 @@ int ip_route_input(struct sk_buff *skb,
    struct rtable * rth;
    unsigned hash;
    int iif = dev->ifindex;
+ struct net_namespace *ns __attribute_used__ = current_net_ns;

    tos &= IPTOS_RT_MASK;
    hash = rt_hash_code(daddr, saddr ^ (iif << 5));
@@ -2096,6 +2114,7 @@ int ip_route_input(struct sk_buff *skb,
    rth = rcu_dereference(rth->u.rt_next)) {
    if (rth->fl.fl4_dst == daddr &&
        rth->fl.fl4_src == saddr &&
+    net_ns_same(rth->fl.net_ns, ns) &&
        rth->fl.iif == iif &&
        rth->fl.oif == 0 &&
#ifdef CONFIG_IP_ROUTE_FWMARK
@@ -2235,6 +2254,9 @@ static inline int __mkroute_output(struc
    rth->u.dst.dev = dev_out;
    dev_hold(dev_out);
    rth->idev = in_dev_get(dev_out);
#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
#endif
    rth->rt_gateway = fl->fl4_dst;
    rth->rt_spec_dst= fl->fl4_src;

@@ -2560,6 +2582,7 @@ int __ip_route_output_key(struct rtable

```

```

{
    unsigned hash;
    struct rtable *rth;
+ struct net_namespace *ns __attribute__((used)) = current_net_ns;

    hash = rt_hash_code(flp->fl4_dst, flp->fl4_src ^ (flp->oif << 5));

@@ -2568,6 +2591,7 @@ int __ip_route_output_key(struct rtable
    rth = rcu_dereference(rth->u.rt_next)) {
    if (rth->fl4_dst == flp->fl4_dst &&
        rth->fl4_src == flp->fl4_src &&
+    net_ns_same(rth->fl.net_ns, ns) &&
        rth->fl.iif == 0 &&
        rth->fl.oif == flp->oif &&
#ifdef CONFIG_IP_ROUTE_FWMARK

```
