Subject: [PATCH 00/16] core network namespace support
Posted by ebiederm on Sat, 08 Sep 2007 21:07:37 GMT
View Forum Message <> Reply to Message

The following patchset was built against the latest net-2.6.24
tree, and should be safe to apply assume not issues are found during
the review. In the interest of keeping the patcheset to a reviewable
size, just the core of the network stack has been covered.

The 10,000 foot overview. We want to make it look to user space
like the kernel implements multiple network stacks.

To implement this some of the currently global variables in the
network stack need to have one instance per network namespace,
or the global data structure needs to have a network namespace
field.

Currently control enters the network stack in one of 4 major ways.
Through operations on a socket, through a packet coming in
from a network device, through miscellaneous syscalls from
a process, and through operations on a virtual filesystem.
So the current design calls for placing a pointer to
struct net (the network namespace structure) on network
devices, sockets, processes, and on filesystems so we
have a clear understanding of which network namespace
operations should be done in the context of.

Packets do not contain a pointer to a network device structure.
Instead their network device is derived from which network
device or which socket they are passing through.

On the input path we only need to look at the network namespace
to determine which routing tables to use, and which sockets the
packet can be destined for.

Similarly on the output path we only need to consult the network
namespace for the output routing tables which point to which
network devices we can use.

So while there are accesses to the network namespace as
we process each packet they are in well contained spots that occur
rarely.

Where the network namespace appears most is on the control,
setup, and clean up code paths, in the network stack that we
change rarely. There we currently don't have anything except
a global context so modifications are necessary, but since

the network parameter is not implicit it should not require
much thought to use.

The implementation strategy follows the classic global
lock reduction pattern. First all of the interfaces
at a given level in the network stack are made to filter
out traffic from anything except the initial network namespace,
and then those interfaces are allowed to see packets from
any network namespace. Then some subset of those interfaces
are taught to handle packets from all namespaces, after the
more specific protocol layers below them have been made to
filter those packets.

What this means is that we start out with large intrusive
stupid patches and end up with small patches that enable
small bits of functionality in the secondary network
namespaces.

Eric

_____