Subject: Re: [RFC PATCH 0/5] net: socket bind to file descriptor introduced
Posted by ebiederm on Wed, 15 Aug 2012 19:49:28 GMT
View Forum Message <> Reply to Message

Stanislav Kinsbursky <skinsbursky@parallels.com> writes:

> This patch set introduces new socket operation and new system call:
> sys_fbind(), which allows to bind socket to opened file.
> File to bind to can be created by sys_mknod(S_IFSOCK) and opened by
> open(O_PATH).
>
> This system call is especially required for UNIX sockets, which has name
> lenght limitation.
>
> The following series implements...

Thinking about this a little more I have serious reservations about this
approach.

Today you are not allowed to bind to an address unless mknod for that
file succeeds. Your patch totally changes those semantics.

Name length limitation does not seeme to justify this at all.

It is possible today to trivially change into a directory and
bind or connect to what would be a long absolute path.

There is also the trick of getting a shorter directory name using
/proc/self/fd if you are threaded and can't change the directory.

The obvious choices at this point are
- Teach bind and connect and af_unix sockets to take longer AF_UNIX
  socket path names.

- introduce sockaddr_fd that can be applied to AF_UNIX sockets,
  and teach unix_bind and unix_connect how to deal with a second type of sockaddr.
  struct sockaddr_fd { short fd_family; short pad; int fd; };

- introduce sockaddr_unix_at that takes a directory file descriptor
  as well as a unix path, and teach unix_bind and unix_connect to deal with a
  second sockaddr type.
  struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
  AF_UNIX_AT

I don't know what the implications of for breaking connect up into 3
system calls and changing the semantics are and I would really rather
not have to think about it.

But it certainly does not look to me like you introduce new systems calls to do what you want.

Eric