主要涉及到两个数据结构 proto_ops 和 proto。

Proto_ops 主要内容是 socket 系统调用对应传输层的函数指针，可以视为从套接口系统调用到传输层函数的跳转表。

Proto 中可以视为是从传输层到网络层函数调用的跳转表。Proto 中有些函数和 proto_ops 中的一致，可以称为传输层接口。

Inetsw 记录了 ipv4 协议族中所有类型的协议。每一个协议类型用一个数据结构 inet_protosw 来表示，相同协议类型的所有协议构成一个链表，链表的头部记录在 inetsw[]数组中。所有的 ipv4 协议记录在 inetsw_array 当中。

在创建 socket 时，在函数 inet_create 中会根据协议类型来遍历 inetsw[], 找到那个类型的 inet_protosw,

inetsw
List_head
List_head
List_head

inet_protosw Inetsw_array[]

```
.type =     SOCK_STREAM,
.protocol = IPPROTO_TCP,
.prot =     &tcp_prot,
.ops =      &inet_stream_ops,
```

```
.type =     SOCK_DGRAM,
.protocol = IPPROTO_UDP,
.prot =     &udp_prot,
.ops =      &inet_dgram_ops,
```

```
.type =     SOCK_RAW,
.protocol = IPPROTO_IP,
.prot =     &raw_prot,
.ops =      &inet_sockraw_ops,
```

```
struct proto tcp_prot = {
    .name               = "TCP",
    .owner              = THIS_MODULE,
    .close              = tcp_close,
    .connect        = tcp_v4_connect,
    .disconnect         = tcp_disconnect,
    .accept             = inet_csk_accept,
    .ioctl              = tcp_ioctl,
    .init           = tcp_v4_init_sock,
    .destroy        = tcp_v4_destroy_sock,
    .shutdown           = tcp_shutdown,
    .setsockopt         = tcp_setsockopt,
    .getsockopt         = tcp_getsockopt,
    .sendmsg        = tcp_sendmsg,
    .recvmsg        = tcp_recvmsg,
    .backlog_rcv        = tcp_v4_do_rcv,
    .hash               = tcp_v4_hash,
    .unhash             = tcp_unhash,
    .get_port       = tcp_v4_get_port,
    .enter_memory_pressure= tcp_enter_memory_pressure,
    .sockets_allocated  = &tcp_sockets_allocated,
    .orphan_count       = &tcp_orphan_count,
    .memory_allocated = &tcp_memory_allocated,
    .memory_pressure  = &tcp_memory_pressure,
    .sysctl_mem         = sysctl_tcp_mem,
    .sysctl_wmem        = sysctl_tcp_wmem,
    .sysctl_rmem        = sysctl_tcp_rmem,
    .max_header         = MAX_TCP_HEADER,
    .obj_size       = sizeof(struct tcp_sock),
    .twsk_prot          = &tcp_timewait_sock_ops,
    .rsk_prot       = &tcp_request_sock_ops,
#ifdef CONFIG_COMPAT
    .compat_setsockopt= compat_tcp_setsockopt,
    .compat_getsockopt      = compat_tcp_getsockopt,
#endif
};
```

```
const struct proto_ops inet_stream_ops = {
    .family             = PF_INET,
    .owner              = THIS_MODULE,
    .release        = inet_release,
    .bind               = inet_bind,
    .connect        = inet_stream_connect,
    .socketpair         = sock_no_socketpair,
    .accept             = inet_accept,
    .getname        = inet_getname,
    .poll           = tcp_poll,
    .ioctl              = inet_ioctl,
    .listen             = inet_listen,
    .shutdown           = inet_shutdown,
    .setsockopt         = sock_common_setsockopt,
    .getsockopt         = sock_common_getsockopt,
    .sendmsg        = inet_sendmsg,
    .recvmsg        = sock_common_recvmsg,
    .mmap               = sock_no_mmap,
    .sendpage       = tcp_sendpage,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_sock_common_setsockopt,
    .compat_getsockopt = compat_sock_common_getsockopt,
#endif
};
```

```
struct proto udp_prot = {
    .name               = "UDP",
    .owner              = THIS_MODULE,
    .close              = udp_lib_close,
    .connect        = ip4_datagram_connect,
    .disconnect         = udp_disconnect,
    .ioctl              = udp_ioctl,
    .destroy        = udp_destroy_sock,
    .setsockopt         = udp_setsockopt,
    .getsockopt         = udp_getsockopt,
    .sendmsg        = udp_sendmsg,
    .recvmsg        = udp_recvmsg,
    .sendpage       = udp_sendpage,
    .backlog_rcv        = udp_queue_rcv_skb,
    .hash               = udp_lib_hash,
    .unhash             = udp_lib_unhash,
    .get_port       = udp_v4_get_port,
    .obj_size       = sizeof(struct udp_sock),
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_udp_setsockopt,
    .compat_getsockopt = compat_udp_getsockopt,
#endif
};
```

```
const struct proto_ops inet_dgram_ops = {
    .family             = PF_INET,
    .owner              = THIS_MODULE,
    .release        = inet_release,
    .bind               = inet_bind,
    .connect        = inet_dgram_connect,
    .socketpair         = sock_no_socketpair,
    .accept             = sock_no_accept,
    .getname        = inet_getname,
    .poll           = udp_poll,
    .ioctl              = inet_ioctl,
    .listen             = sock_no_listen,
    .shutdown           = inet_shutdown,
    .setsockopt         = sock_common_setsockopt,
    .getsockopt         = sock_common_getsockopt,
    .sendmsg        = inet_sendmsg,
    .recvmsg        = sock_common_recvmsg,
    .mmap               = sock_no_mmap,
    .sendpage       = inet_sendpage,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_sock_common_setsockopt,
    .compat_getsockopt = compat_sock_common_getsockopt,
#endif
};
```

```
struct proto raw_prot = {
    .name               = "RAW",
    .owner              = THIS_MODULE,
    .close              = raw_close,
    .connect        = ip4_datagram_connect,
    .disconnect         = udp_disconnect,
    .ioctl              = raw_ioctl,
    .init           = raw_init,
    .setsockopt         = raw_setsockopt,
    .getsockopt         = raw_getsockopt,
    .sendmsg        = raw_sendmsg,
    .recvmsg        = raw_recvmsg,
    .bind               = raw_bind,
    .backlog_rcv        = raw_rcv_skb,
    .hash               = raw_v4_hash,
    .unhash             = raw_v4_unhash,
    .obj_size       = sizeof(struct raw_sock),
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_raw_setsockopt,
    .compat_getsockopt = compat_raw_getsockopt,
#endif
};
```

```
static const struct proto_ops inet_sockraw_ops = {
    .family             = PF_INET,
    .owner              = THIS_MODULE,
    .release        = inet_release,
    .bind               = inet_bind,
    .connect        = inet_dgram_connect,
    .socketpair         = sock_no_socketpair,
    .accept             = sock_no_accept,
    .getname        = inet_getname,
    .poll           = datagram_poll,
    .ioctl              = inet_ioctl,
    .listen             = sock_no_listen,
    .shutdown           = inet_shutdown,
    .setsockopt         = sock_common_setsockopt,
    .getsockopt         = sock_common_getsockopt,
    .sendmsg        = inet_sendmsg,
    .recvmsg        = sock_common_recvmsg,
    .mmap               = sock_no_mmap,
    .sendpage       = inet_sendpage,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_sock_common_setsockopt,
    .compat_getsockopt = compat_sock_common_getsockopt,
#endif
};
```