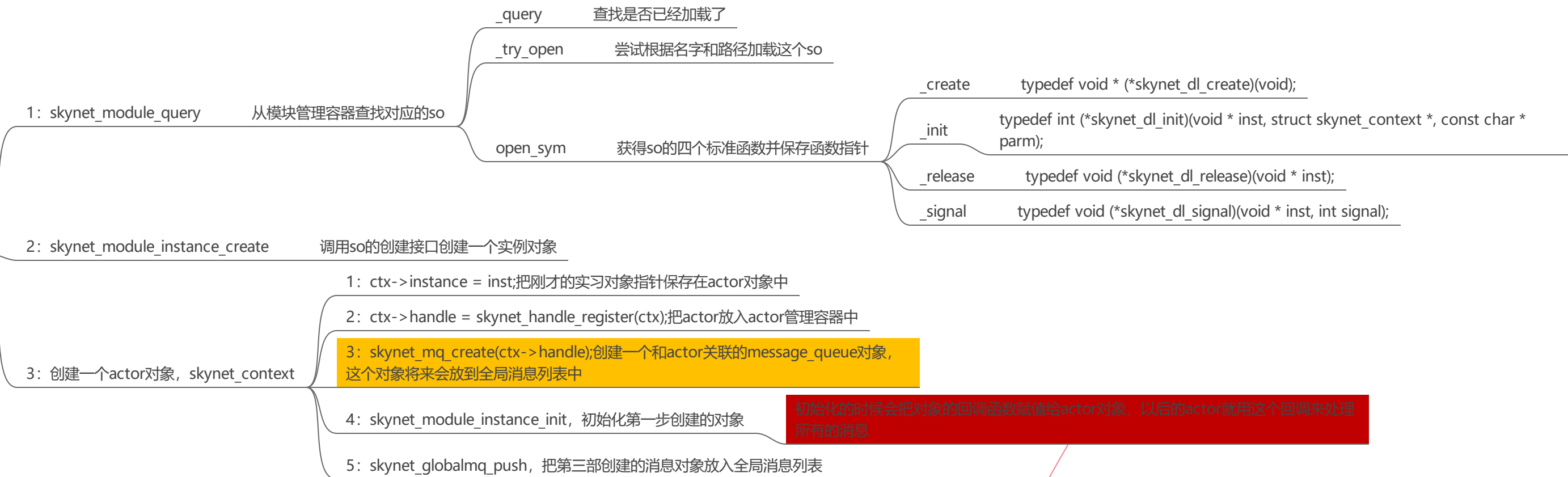


skynet启动流程



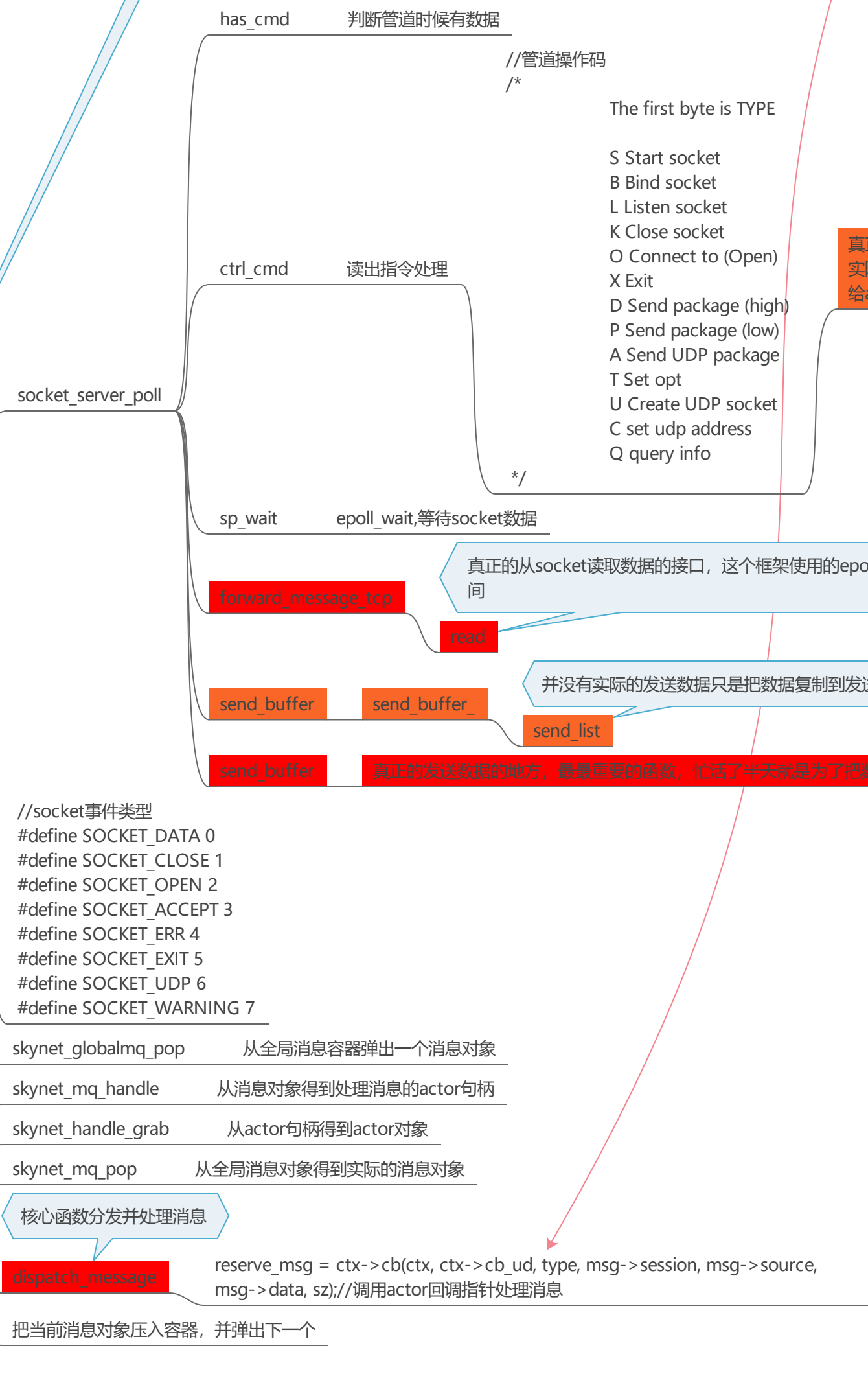
启动框架这个是核心函数



非常重要这个地方开启了核心的工作线程



线程中循环调用这个函数处理管道和socket数据，管道用的select,socket用的epoll水平触发模式



真正的socket的对外连接，监听端口等都是通过actor发送指令到网络层，网络线程来实际执行的，创建完socket就会和actor句柄进行绑定，之后socket的所有事件都是传给actor去处理

从socket读取数据的接口，这个框架使用的epoll水平触发模式这个有改进的空

并没有实际的发送数据只是把数据复制到发送队列

真正的发送数据的地方，最重要的函数，忙活了半天就是为了把数据发出去

```
//socket事件类型
#define SOCKET_DATA 0
#define SOCKET_CLOSE 1
#define SOCKET_OPEN 2
#define SOCKET_ACCEPT 3
#define SOCKET_ERR 4
#define SOCKET_EXIT 5
#define SOCKET_UDP 6
#define SOCKET_WARNING 7
```

skynet_globalmq_pop	从全局消息容器弹出一个消息对象
skynet_mq_handle	从消息对象得到处理消息的actor句柄
skynet_handle_grab	从actor句柄得到actor对象
skynet_mq_pop	从全局消息对象得到实际的消息对象

核心函数分发并处理消息

```
dispatch_message(
    reserve_msg = ctx->cb(ctx, ctx->cb_ud, type, msg->session, msg->source,
    msg->data, sz); //调用actor回调指针处理消息
```

把当前消息对象压入容器，并弹出下一个

7: pthread_join(pid[i], NULL); 阻塞主线程并等待所有线程退出，框架到现在真的运行起来了

12: 退出线程

8: 退出主线程，走到这一步就退出进程了