中期报告

一.毕业论文主要研究内容，进展情况及取得成果

1.毕业设计内容是进行适用于rel4的网络协议栈设计，而其中rel4是一个用rust改写的sel4微内核，主要应用于嵌入式系统，整体思路是对smoltcp进行改造优化，得到一个性能较好的适配协议栈。

2.当前进展是已经完成了对于开源网络协议栈smoltcp的分析，并在基础上通过分析embassy-net和starry-os中对于网络模块的代码编写完成了对于性能提升的改进方案，并进行了当前情况的性能测试。

首先是smoltcp的协议栈分析，其作为我主要进行修改设计的对象，需要深入分析，其并不是以常见的osi七层模型进行的模块区分，其主要分为了socket，iface，storage，wire和phy五层，其中的关系也不是严格的高层低层封装关系。具体来说

Socket作为最顶层与应用对接，负责提供不同协议的统一接口，如tcp,udp,icmp,dns等，这里封装的是高层的细节，如tcp的监听，碰撞检测等，并不负责底层的数据包封装，具体这部分是由wire层进行处理，而解析也是由wire层负责，其秉持的思想是能构建就能够进行解析，通过一个模块负责去保证不会发生无法解析的情况，iface层是直接与网络设备交互的接口层,负责整合wire和socket模块中函数，storage层中包括了静态的缓冲区实现。最后phy层负责最底层的虚拟设备传输，如loopback,rawsocket,和tuntap传输等，定义了设备的特征。

Smoltcp中是没有给出高层的封装的，也就是并没有一个函数去真正封装了发送和接受的函数，其需要使用者去进行编写，也就提供了进一步进行逻辑修改的可能性。

从smoltcp本身结合示例代码，可以分析得到他的运行逻辑是：一个设备维护多个socket连接，这部分是由socket层定义的，发送是多个函数的调用结果，首先是socket调用send，把数据存储到socket本身的缓冲区中，之后每个设备有一个interface，其会以同步的方式调用poll函数，而poll函数可以分为两个功能，一个是发送，也就是依次取出所有socket中的缓冲区的内容，通过wire层的函数进行数据报的封装，之后存储到设备本身的发送缓冲区中，由设备驱动进行发送；而另一个功能是接受，在完成发送后，会检查设备本身的接受缓冲区，有数据的情况下将其取出并进行解析，将解析结果放入相应的socket的接受缓冲区中，在之后socket通过recv函数进行，如下图所示

图示

AI 生成的内容可能不正确。

其次是对embassy-net和starry-os中网络模块axnet的分析，二者都是在smoltcp的基础上进行了封装，为其提供更好的性能，从上面的分析可以看到socket的send函数，recv函数，还有interface的poll函数都是通过同步的方式，也就是通过轮询控制执行，axnet是通过wouldblock机制来进行了send无法执行条件下的逐层阻塞，从而进行性能提升，减少无效判断，而embassy-net中进行了进一步的提升，使用了异步的方式进行函数调用，具体就是将send和recv定义为async函数，借用smoltcp中本身就有实现的waker机制，在准备好数据的情况下调用waker唤醒async函数，来实现异步执行。两种优化方式为我的后续性能优化改进提供了借鉴思路。

最后进行了当前的性能测试，axnet是在一个完整的操作系统starry-os中的模块，也就可以方便的进行发包收包的带宽测试，通过构建已经封装好的数据报，可以测试得到接口层的传输速率，然后通过iperf来进行qemu中的实际socket发包测试，可以得到如下图的测试结果

（柱形图待补充）

二.存在的问题和拟解决方案

1.不能明确当前的改进方案能有效的提升smoltcp协议栈的性能

拟解决方案：通过对embassy-net中发送的函数调用等进行严格分析，配合各模块的独立测试来了解各模块的开销，从而对改进后的性能进行有效估计

2.smoltcp对于网卡的支持中要求各协程单独使用网卡，不能同时进行收发，这个限制了性能，但是如果进行解决，对于poll函数的修改可能会涉及到同时使用发送函数，从而导致抢占问题

拟解决方案：查看相关对于设备的驱动代码，分析修改难度，在完成基础的异步修改后作为后续改进方案

三．下一步研究任务与进度安排