

**本科生毕业设计(论文)**

# 中期报告

**北京理工大学本科生毕业设计（论文）题目**

**The Subject of Undergraduate Graduation Project (Thesis) of Beijing Institute of Technology**

|  |  |
| --- | --- |
| 学 院： | 计算机学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 07112101 |
| 学生姓名： | 戴骏翔 |
| 学 号： | 1120210610 |
| 指导教师： | 陆慧梅 |
| 校外指导教师： | 向勇 |

|  |
| --- |
| 一、毕业设计（论文）主要研究内容、进展情况及取得成果 1.毕业设计内容  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，研究的整体思路是对开源网络协议栈smoltcp进行改造优化，得到一个性能较好的适配于rel4的协议栈。  2.当前进展  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。  3.取得成果，  首先是smoltcp协议栈的分析，其作为我主要进行修改设计的对象，需要深入分析。作为一个网络协议栈那其并不是以常见的osi七层模型进行的模块区分，而是分为了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。  Socket层作为协议栈与应用层的交互界面，承担着多协议抽象与连接管理的核心职责。该层实现了TCP、UDP、ICMP、DNS等协议的Socket抽象，但其设计理念与传统协议栈存在本质区别：它不直接参与数据包的封装与传输过程，而是专注于协议状态机管理与缓冲区调度。以TCP协议为例，Socket层实现了三次握手、流量控制、超时重传等核心机制，但将数据包的实际构建工作委托给下层模块。这种设计使得协议处理逻辑与数据传输路径实现解耦，为协议扩展提供了灵活性。值得注意的是，smoltcp的Socket层并未提供完整的BSD Socket API，而是暴露了更底层的控制接口，这种设计选择既降低了协议栈的复杂度，也为开发者保留了定制协议行为的空间。  wire层是协议栈的协议编解码引擎，其设计遵循"可构建即可解析"的核心原则。该层包含了以太网帧、IP数据包、TCP/UDP报文等各层协议的构建器与解析器，通过严格的类型系统保证协议字段的正确性。例如在构建IP数据包时，wire层会自动计算校验和并填充必要字段，确保生成的报文完全符合协议规范；在解析过程中，则会进行严格的格式验证，丢弃不符合规范的数据包。这种双向一致性保障机制，使得协议栈具备自我验证能力，从根本上避免了畸形报文导致的解析错误。wire层通过零拷贝技术直接操作原始字节流，既提升了处理效率，又减少了内存消耗。  iface层负责协调Socket层与wire层的交互。该层通过同步轮询机制驱动整个协议栈的运转：在发送方向，iface层定期收集各Socket的发送缓冲区数据，调用wire层进行协议封装，最终将完整的数据帧提交至物理设备；在接收方向，则从物理设备获取原始数据帧，经wire层解析后分发至对应Socket的接收缓冲区。这种显式的poll驱动模型，与常见操作系统内核的中断驱动架构形成鲜明对比，虽在实时性方面存在理论劣势，却显著降低了上下文切换开销，更适应资源受限的嵌入式场景。  storage层通过静态预分配的缓冲区管理策略，实现了确定性的内存使用。该层提供环形缓冲区、分片缓冲区等多种数据结构，支持零拷贝的数据传递。例如在TCP传输过程中，应用层数据直接写入Socket的发送环形缓冲区，iface层在poll周期内批量获取这些数据进行协议封装，整个过程避免了数据复制带来的性能损耗。这种设计对内存碎片化问题具有天然免疫力，特别适合长期运行的嵌入式系统。  phy层作为硬件抽象层，定义了统一的网络设备接口。该层支持环回接口、TUN/TAP虚拟设备、原始套接字等多种物理/虚拟设备的接入，通过trait抽象屏蔽具体设备的差异性。开发者可通过实现Device特质，将协议栈移植到定制硬件平台。phy层的异步设计允许在无操作系统支持的环境下，通过忙等待或硬件中断机制驱动数据传输，这种灵活性是smoltcp能适配多种嵌入式场景的关键。  Smoltcp中是没有给出高层的封装的，也就是并没有一个函数真正封装了发送和接收的函数，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，适合我对其进行修改。  从smoltcp各层功能以及相应函数，可以分析得到他的运行逻辑是：一个网络设备维护多个socket连接，这部分是由socket层定义的，而发送是多个函数的调用结果，首先是调用socket的send函数，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，首先进行数据报的封装，这部分由通过wire层的函数进行，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；而另一个功能是接收，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数据报解析，将解析结果放入相应的socket的接受缓冲区中，之后实际处理使用通过socket调用recv函数进行。  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在实现中使用了WouldBlock机制来协调非阻塞I/O，这里它的设计目标并非直接实现异步I/O，而是通过与操作系统的调度协作，在用户态模拟类似异步的效果，从而进行性能提升，减少无效判断，任务在完成的情况下则返回值，未完成则返回WouldBlock。然后分为两种模式，在non-blocking模式下，子任务未完成，父任务立即返回WouldBlock；而在blocking模式下，子任务未完成，父任务轮询到其完成为止。轮询中交替检查网络包状态和任务状态。这样的实现相比与异步会因为重试多次上下文切换，还是会有性能消耗。  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，基于异步运行时，将smoltcp的同步轮询接口改造为异步任务驱动模型。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，减少了空闲时的CPU占用，允许其他任务在等待网络事件时执行，提升系统整体效率。两种优化方式为我的后续性能优化改进提供了借鉴思路。  最后进行了当前版本的性能测试，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，首先benchmark中构建已经封装好的数据报，直接发送可以测试得到接口层的传输速率，作为上限速率，然后通过starry-os中已有的iperf服务器和iperf指令来进行qemu中的实际socket发包测试，可以得到测试结果：在x86\_64的本地测试中，接口层的发送速率约为1000Mbps，传输层约为300Mbps。 |
| 二、存在的问题和拟解决方案 １.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，并且不是很符合正常的网卡工作原理。  拟解决方案：查看相关对于网卡的驱动代码，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，同时如果针对这部分进行修改  2，并不确定poll的修改一定能大幅度的提升性能  该问题存在的原因是smoltcp中是有通过pollat函数去限制poll被循环调用的频率，也就是针对poll函数是有进行同步调用的优化的，而现在我通过异步去进行修改吗，可能并不会像axnet和embassy-net中修改socket的send和recv函数那样有显著的性能上的提升，当然修改为异步函数从逻辑上更简洁，且在资源利用率上有理论优势，所以还是值得进行修正.  拟解决方案:针对一次执行过程中的poll有效情况进行实验测试。而之前进行的性能测试使用的starry-os，接口层的测试是直接构建的数据报，通过设备发送，因为直接构建，所以并没有经过socket的管理，测试代码中是直接调用设备层的接口进行发送，所以不方便进行测试，而传输层所使用的iperf服务器是通过c代码实现的，并通过脚本构建，修改需要的学习成本较高，所以后续考虑直接完成测试代码进行测试 |
| 三、下一步研究任务与进度安排 相较于开题报告中的计划安排，因为添加了embassy-net和axnet的学习，进度有所落后，但是二者在smoltcp优化上的思路会使我后续的编程更加方便，也会使优化更加有实际意义，而不是重复已有工作。  1.研究任务方面  首先完成针对当前版本的poll函数性能的测试，预估修改会产生的性能提升，方便后续修改后进行结果分析。  其次完成poll函数的修改代码，首先将poll函数拆分为发送和接收的两个函数，之后增加设备驱动层的waker，并将两个函数定义为async函数，通过驱动层的waker来唤醒。  之后进行在之前版本的性能分析基础上进行进一步的测试，之前传输层的测试所使用的iperf是在qemu中进行的测试，相当于使用cpu来控制的传输而不是物理显卡，需要增加实际网卡的测试，提升说服力。  2.进度安排  1.Poll函数性能测试及修改（1周）  2.新老版本性能测试比较分析（1周）  3.完成毕设论文（2周） |
| **四、指导教师意见**  签字：  年 月 日 |
| **成绩： ，占比 %** |
| **五、中期审核负责人意见**  签字：  年 月 日 |