南京大学计算机学院教学观察报告：以学生和助教的视角

南京大学计算机学院 崔博涵 bohan.cui@outlook.com

写在前面

NJU在很早之前就打出了“做中国最好的本科教育”的旗号。在NJUCS的两年半多时间里，也切实的感觉到系里的教学质量是有一定优势的（比如：CSDIY提名最多的国内高校（3次）），但也有很多不好的体验，观察到一些很严重的问题。事实上，出于某些各种各样的原因，NJU现在是华东五校里学生内卷程度最低的学校，因此也是最有可能孕育出“最好本科教育”的土壤，有希望通过提高教学质量来走出一条不同于堆资源、堆经费或者鼓励内卷养蛊的道路来发展院系和校友网络。遗憾的是，从结果上看，整体上似乎远远没有达到预期，也没有和华东友校拉开差距。可以理解，在今天，PI的评价机制和就业机会的收缩是影响教学质量和学生生存环境的核心原因，也是无法改变的原因，但现实当然有大量可以改进的机会。因此本文希望利用自己两年半的学习经历，一年的助教工作经历，以及和多位老师的交流，在yanyan老师的鼓励下，提出一些教学问题的一线观察和改进教学质量的“对症治疗方案”。我个人也愿意为一些教学改革工作贡献自己的一份力量。

本文分为 “关于减轻学生负担的问题”和“关于一些必修课程组的具体建议”两个部分。前者理论分析了现在的学生生存（内卷）环境并指出了一些现有方案存在的问题，后者则是对各类课程提出的一些具体的建议。本文交由南京大学计算机学院教学委员会审阅，恳请委员会老师们批评指正交流。

关于减轻学生负担的问题

正所谓“屁股决定脑袋”，我们讨论策略和改进之前先明确立场和目的。

如果我们的前提假设和根本目的是：

1. 让**少数**有想法的同学获得空间，而不是被整体的内卷裹挟。
   使他们更有可能做出 Something Interesting
   功利的讲，这些同学有机会成为出色的研究生，PI，校友乃至提高NJUCS的整体实力。

2. 让**多数**同学受到 Professional Training 而不是盲目的 Research Training。
   如果没有什么motivation，那么在某些强制要求下可以获得Qualified for Industry的能力。
3. 我们希望**尽可能多**的同学可以获得较好的学习体验。（因为永远都回有一部分能力极强者在任何环境中都可以获得极高的自由度，同时无视一些功利的需求和客观的限制。如果局限于这个群体，那就没有讨论的必要了。）

为什么会产生内卷？

一个核心的因素是现在学历混同。相对低学习能力/驱动力者可以通过一些手段（课程设计的不合理，作弊，抄袭，大语言模型（方便起见，下文我们将其称之为**不当因素**））获得一个“几乎相当”的回报。这导致相对高学习能力/驱动力者不得不通过“制造更高的标准”来自我区分（以保证自身在未来发展路径的选择上保有一定的自由度）。重复上述的过程，最后就会导致所有人的时间投入越来越高。相应的，学历的价值就会越来越低。

Spence, Michael. "Job Market Signaling." The Quarterly Journal of Economics, Volume 87, Issue 3 (1973): 355–374.

我们在这里提出一个核心的关系式

$$Involution \approx RealW = \frac{W}{Reward} = \frac{k_1 \cdot W_{Compul} + k_2 \cdot W_{Extra}}{Value}$$

$$Value \approx f(W_{Compul}) \quad (\text{正相关})$$

W 代表 $Workload$ 亦即学生的工作量。$Value$近似为学历的认可度。

误区1：我们通过减少课程的量就可以减少Workload吗？

表面上 $Workload \propto Content\ of\ course$

(NO, it is true only for maybe GPA 4.0/5.0, but GPA 4.0 somehow means no any good opportunity for normal students)

实际上 $Workload \propto Scoring\ of\ course$

$$W_{compul} = Content + \Sigma\frac{pr_i}{E_i}$$

E_i为每一项（实验、作业、出勤、考试...）评价的期望的得分， pr_i为其分数占比。

实际上最主要的工作量是受到评价机制的影响的。

- e.g. 一个现象：究竟是否有人在听课？

  对于某些课程来说，听课的作用之一更多是获得签到、考试范围的缩减、心理的“免责声明”。有相当一部分课程的内容并没有被elaborate effectively. 所以部分课程真正在课堂上听课的同学可以说甚至是个位数，多数人在写作业，干其他事情或者戴耳机听其他的课。坦白的讲，部分课程限制同学们听平替和上位替代的网课的主要因素仅仅是授课内容的区别从而导致备考成本的增加。所谓考试范围的缩减指的是由于过于宽泛的考试内容（如下一个例子所言），需要从老师上课的只言片语中推断考试的强调程度。所谓“心理的免责声明”，是指大量界定不清楚的概念和不充分甚至相互矛盾的问题需要一个官方说法，以期在考试中有个统一的写法。

- e.g. 一个现象：“无限的”知识点。

  有一些考察无限细节的考试，由于没有明确的核心内容和清晰明确的考察范围，而是“可以考任何东西”，从而导致需要复习的内容是无限的。作为同学，在考场上有相当一部分题目的感受就是，“太幸运了，这道题我在某某天某某时候多亏顺手多看了一眼”。而在复习的感受就是如果想要好成绩就“永远复习不完”，比如我的宿舍大家都比较在意成绩，我个人属于“考究派”，会对大量细节进行“考古学一般”的对比（最后往往会因为过于耗时间来不及了越看越粗略，事实证明也是没有用的）而我的另一位室友属于“通读派”，他在考试前往往会读三遍甚至以上的所有ppt（如果时间允许，甚至可能更多）。
  这样的考试在统计上会有非常好看的分布（当然如此，对于每个人都是一张卷子若干题就是一个随机多次重复实验，最后自然近似于正态分布）
  这种情况用某学长的描述很合适，就是
  课程的投入-产出曲线是一个根号曲线。
  在这个情况下，对于想要取得较好分数的同学，减掉一部分课程的内容其实并不有助于负担的减轻。

  > “课堂上承载（以及ppt里）知识点的分辨率是无限的，简直是连续的模拟信号，相比之下复习是数字信号。考试时能够记住的分辨率是极其有限的——脑容量（闭卷）和A4纸正反面的面积（开卷）制约了它的上限。”——某同学

- e.g. 一个现象：无限精细的实验报告。
  e.g. 一个现象：“考试月消失”现象

  为什么往往会出现“考试月”的现象？明明考试集中在几天，学习的东西无非三个月，为什么甚至要复习一个月？以及为什么实验报告总是会无限精细。

一方面就是上面说的考试“无限知识点”的问题，另一方面就是如果有很大的考试占比，而考试的得分又有比较强的随机性。因此同学们就自然会在实验报告这种可以自己掌握且有一定弹性（评价模糊性）的内容上尽可能地完善，以期给学期的得分一个“保底”，达成“尽人事听天命”的效果。自我反思一下，我也具有这个问题。下面贴一个我自己的实验报告作为例子（见文件最后的附件）。

> 实际上，这些形式精美的实验报告当然并没有什么实际的作用，反而会占据大量的时间。
> 作为DLCO的助教，我已经见过了无数的极其精美的实验报告，以计金班的格外突出。但实际上其中并没有什么特别有价值的内容（比如对一些内容自己的思考，或者自留的一些经验记录），甚至有的是来源于抄袭。在明确了以思考题实现为重且不看美观与否的情况下，日常更是有大量的同学询问实验报告事宜，都是多一张图少一张图这种细枝末节，可见大家对这种东西谨慎到了病态的程度，这当然来自于对于高占比考试不确定性的担忧。

在这个背景下，实验的问题本身 和 知识本身 反而成为了最不被关注的东西。

误区2：我们依然很难仅通过降低Workload来降低内卷？

当Workload降低时（相当于是在向不当因素妥协）此时分子虽然下降了, 但正如上面的表达式，由于必要部分的量下降了，学历也就更混同了，Value进一步降低，获得的收益减少了，内卷就更严重了。

降低课程难度的道理相同。如果降低难度到**让人觉得**水（甚至可能不是真正的水），那么这个课就没有存在的必要了，无论上几个学时取得的收益都是有限的，不如放同学们自学。

企图利用不当因素的人的核心逻辑是通过最少的Workload获取最大的Reward，课程和制度的设计永远是有缺陷的，课程和制度的迭代是缓慢的，但是“群众的力量”却是无限的，相应的投机取巧策略很快就会被发现出来。

这件事情告诉我们，在现在的环境下，**永远无法通过课程给分本身**将不同掌握程度不同投入程度的同学区分开。而且在整个社会的对这种区分的认可度低的背景下，更是不现实的。

如何部分解决内卷？个人的一些建议。

个人认为应该结合以下两个：降低Workload和提高Reward来改善这一情况。

降低Workload：

由于企图利用不当因素的人的核心思路是：**最少的工作量**，尽量小的风险，**最多的收益**。

一个比较好的方法就是“弹性分数占比制”：将一些难度较高的实验设置为可选的，高工作量，高收益，开放的或易于学术诚信检查的内容。

这样如果投机取巧的风险成本和工作量的成本，超过其预期的收益，就可以阻止他们利用不当行为。（如普通物理中的制作视频作业，这一作业可以抵扣约15%左右的考试占比，但因为要求比较高且工作量大，绝大多数人都不会选择去完成，自然也很少会出现不诚信现象）同时如果降低了选择完成这些任务的人的人数，也降低了学术诚信检查的难度。

采用弹性分数占比制，允许通过选择这些可选内容而将考试占比降到较低的30%甚至20%以下，就可以减少不确定性带来的考试负担。

差异增加Reward的方式: Extra Credit：

（注：这个credit不是指实际的学分）

根据上面的论述，即使通过弹性给分降低学习压力，还是会造成混同。毕竟成绩单上体现不出一门课的具体的信息，以及同学的参与程度。

$$Involution \approx RealW = \frac{W}{Reward} = \frac{k_1 \cdot W_{Compul} + k_2 \cdot W_{Extra}}{Value + Reward_{Extra}}$$

所以我们可以在课程内的实验中引入额外的、课程外部的回报方式。也就是说面向同学们的实际需求，使他们获得一些能体现在简历上的，具有一定的社会认可的经历。

比如：作为可选部分的实验，可以提供更多的项目制的，有相当自主性的内容。并不排斥增大内容的难度和标准，以达到可以对接一些竞赛，开源项目，科研课题，自我驱动项目或者说至少是有一定独特性的项目经历。（而不是传统的固定的实验）在课程的评分中折算替代一部分考试占比。

我认为需要做到几点：

- 增量的：在折算到课程得分时，必须是增量的（也就是选择了更有趣的可选项目，分数必须做到只可能比正常更高，当然如果完成度很低可以只增加一点）从而降低不确定性。
- 集中的：相比于传统的 课程+科研实践 双轨，尽可能做到以课程内容为基础，和课程内容的强相关，以部分解决“相互耽误”的问题。
- 公平的：需要与之相适应的评价体系，比方说一个几个任课教师和助教组成的评价委员会（也可以作为常设机构，因为定位就是少量人可选，可以很多课程集中

进行）以防止灌水现象。

- 非组队的：组队是万恶之源。交流机制是很好的，但永远不要低估学生中的“劳动力市场”。

这当然对授课的老师提出了要求，也就是说需要有相当的insight，以设计类似的内容并且提供相关的指导。

更长远的，我们应该构建更完整的有我们自己主导的评价体系或者说Reputation。

> reputation有相当的作用，比方说呢喃的“最好本科教育”口号（虽然更多是被人嘲讽的谈资）但在某些认知和风评中，呢喃本科CS同学确实有更全面的知识体系。这有可能是刻板印象，可能也只是相对普通985有优势，但客观上肯定是有利于同学的生存的。

也就是一定程度上跳出社会评价（比方说竞赛，其实大家都清楚其中的水分），依托课程实验做成一些可持久化的“成果管理”。比方说排行榜或者一些面向社会公开的展示和评价平台，随时可供回顾和管理。（e.g. 比如说“智能计算系统”中的“算子开发”实验自然可以对一些复杂的，面向现实的优化任务设计永久性的排行榜）

> 举一个例子：Codeforce其实就是个第三方机构，但由于其Reputation，学生仅凭在其上拿到一些排名就可以收获相当的认可。而如果Codeforce是我们自己开的，那么在就不用看别人的评价体系的脸色。 传说中的“在CMU上CSAPP课拿到A+”就是一个很好的例子。

关于一些必修课程组的具体建议

- 课程需要有一个明确的“达到学习目标”的标准。明确考核的核心知识点。并不吝惜为达到标准的同学90+的成绩，而不是卡30%的优秀率。（使得同学们的课程学习目标从原来的**达到人群的前30%**变成**达到某固定的要求**）

> 事实上，给定考核知识点并不一定导致整体分数的水涨船高，很多课程都证明了这一点。因为如果课程不水，给定范围也不看/临时抱佛脚以至于看不完的人大有人在。

- 将优质的课程开放给CS大班的同学。比如说拔尖班的PA（如果未来还保留这个区分的话），etone老师更适合CS的概率论等等。

> 现在CS某种程度成为了人下人+人上人
> 人上人是因为具有所有泛计算机专业最为宽松的保研要求和最全面或者说模糊的培养方向。
> 人下人是因为在CS的老师的开设的课程上享有最少的资源，甚至不如其他院系。比如：比较受欢迎的why老师的ICS反而是软院可以选修；etone老师的概率论反而是苏州校区可以选修；而人工智能学院名为一种特化培养，实际上课程基本上是计科课程的全套上位替代...；计科的部分课程想转专业的外院系同学可选，但本系同学却不得不跨专业选还有学分不互通问题等等。
> 并没有拉踩其他院系的意思，但是课程上确实有些厚此薄彼。

- 前面详述的降低Workload和提高Reward的综合策略，此处不赘述。
- 课程最好提供一种“全信息材料”。也就是说能够完整还原出课程的内容的材料。一个更为普遍的的选择是提供完整的课程回放。也可以是Lecture Note等。避免使用“不完全信息”作为一种签到手段。可以采用完全开放或者限制开放的形式，比如开放给本校同学。可以帮助同学们复习或者让优质课程资源突破班级人数的限制。

> Lecture Note等工作表面上会造成老师的额外的工作量，实际上反而能够减少老师的答疑压力。属于一劳一定程度永逸的工作。
> （当然极个别老师完全不答疑，直接搪塞课上课下的全部提问）

> Slides通常不是一种全信息材料（比如ICS的），因为里面有很多内容为了简洁明了只提点了关键词，是由老师补充说明的。有提出的很多的问题答案也没有写在上面。

- 有些课程和评价机制总是有一种导向，即利于本校读研或者说“不耽误本校读研就行”。

e.g. “给分的绝对值又无所谓，反正保研是看排名”
这样并不能将好同学真正留住，以某种限制留住的也没有什么归属感。而且“留住同学”也很难说是一种有利于NJUCS发展的选择。

- 对于很多课程：“**其人存，则其政举；其人亡，则其政息。**” ——*《论语》*

关于计算机系统课程组的建议

关于ICS：

- 实际上ICS的内容和后续的OS、前置的DLCO有一定的重合。
而DLCO和OS又都是对所有人必修的，所以完全可以把ICS中的这部分删掉。
- ICS中有一些内容讲的过于简略，比如说中断。这一部分其实如果没有实践很难理解清楚。而且既然在后续课程中要必修，就没有必要通识地讲这一部分。
所以其实还是可以删掉。
- 链接部分内容没有实际的实践（作为一个重点，大班版本课程中只有linklab还是有些简易），虽然讲了很久，但其实很难理解清楚。
- **[同样适用于DLCO理论课]** 课程Slides上有很多non-trivial的值得讨论的问题，但往往授课的时候也没有解释清楚，而是一带而过。Slides上也没有进一步的文字解释，这一部分实际上最关键的内容就被略去了。
- 如果保留x86-PA的话，需要丰富内容和增加难度（原因如前面讨论所述）。黄金版本这种东西，属实没有什么实际的意义，还严重影响了实验的质量。wl老师的实验指导课讲得非常好，但实验指导课这种东西确实不必存在，因为进一步增加了实验的水的程度（为什么不让wl老师来讲理论课呢）。
- 从学生视角来看，根据所见所闻PA实际上也有严重的诚信问题。

关于DLCO理论课：

- 实验本身缺乏insight，过于程序化了，完全不需要思考（一大部分同学可能学完之后对ISA还是几乎没有什么理解）。 从助教的来看，同学们经过实验之后对课程内容的理解可以说是非常非常有限的。
- 极其严重的作弊，从助教的视角和同学们的反馈来看，可能有**至少30%**的作弊率。由于有多个班级+Logisim的特殊性，查重也非常困难。甚至有买卖实验作业的现象。有**50%**甚至更多直接参考现成实现的比例。
- 应该以某种形式改进实验的内容，变成更加灵活、可拓展和易于查重。可以放弃Logisim了（当然连电路图的形式可以得到一定程度保留，毕竟这对于硬件设计是很必要的）

- 讲授了过多的几个指令集的细节，又缺乏进一步的说明，这些东西其实无益于理解指令集的核心设计理念，只是徒增了很多无意义的内容。指令集的细节在实验中涉及一下其实就完全可以了。
- 实验报告中极其模糊的思考题完全可以去掉**[其实还有很多其他的课程中的也类似]**，批改的时候实际上有大量LLM生成的内容。改成明确的评分标准，降低实验报告的形式要求和分数控制要求。
- 对于实验设置，依然是：**其人存，则其政举；其人亡，则其政息。**

关于DLCO实践课：

- 这个课的上限其实很高，可以对接其他任何一门课程。还有很大的发挥空间。

Takeaway messages

- 实际上很难通过直接鼓励同学去做一些interesting的东西，只能降低他们做这些事情在其他量化指标上付出的代价。
- 解决内卷问题的关键是把同学们分到不同的篮子里。也就是根据同学们的切实需求给予相应的指导和reward。如果所有人都在一个篮子里，永远无法解决内卷。
- 只打击浑水摸鱼者是无法改善环境的，必须想方设法给有想法、认真做的同学以奖励。仅在课程内部进行制度设计是不可能的，必须引入外部性（即上述的extra reward）
- 通过“固定标准”，弹性分数制等方法降低workload + 引入extra reward来给同学减轻负担，而不能**只**使用减少课程量的方法，更不能用“喂食”的方法降低课程的难度。提高课程的难度和质量并使得同学获得相应的reward才实际上有机会改善同学们的生存环境。
- 若干条关于全部课程和系统课程组的建议（详见上文）。
- 子曰：其人存，则其政举；其人亡，则其政息。

附件：一个实验报告样例

Lab 5 Report for Computer Network

Bohan Cui
ID: 221220089
NJU, Department of computer science and technology
Nanjing, China
forever_1210@yeah.net
Tutor: Prof.Tian

I. Basic Info

A. Lab Name: Router III

B. Lab Aims:

- to learn the mechanism of ICMP
- to implement router with ICMP Error generation function

II. Echo Response

A. Core Implementation (Briefly)

To better implement this part, I make some adaptations to the original code. I encapsulate the forwarding and ARP lookup procedure into a function call, so that it can be reused by the ICMP error generation part.

```
in handle_packet() ...
    if dst is self and protocol is ICMP ...
        if type is EchoRequest ...
            rplpkt = self.Echo_respond(packet)
            return self.forward_a_packet(rplpkt)
```

Listing 1. In handle (briefly)

The func is below.

```
def Echo_respond(self, pkt):
    reply = Packet()
    ... # constructors of headers
    reply[IPv4].src = pkt[IPv4].dst
    ... # initialize the IPv4 header
    reply[ICMP].icmptype = ICMPType.EchoReply
    reply[ICMP].icmpdata.data = pkt[ICMP].icmpdata.
        data # copy the data domain
    reply[ICMP].icmpdata.sequence = pkt[ICMP].
        icmpdata.sequence
    reply[ICMP].icmpdata.identifier = pkt[ICMP].
        icmpdata.identifier
    reply[ICMP].icmpcode = 0 # 0 for reply
    return reply
```

Listing 2. Echo respond

B. Local Test

There is no local test for this part.

C. Troubleshooting

No trouble arose.

III. Generate ICMP Error

A. Core Implementation (Briefly)

Here I will show the adapted forwarding with error generation. Nested error will be detected everywhere.

```
def forward_a_packet(self, packet):
    ...
    nh, intf = self.fwd_table.lookup(dst_ip)
    if intf == None: # can not find in fwd table
        if nested error:
            drop...
        urb = self.make_nwk_unreachable(packet) #
            Network Unreachable Error
        return self.forward_a_packet(urb)
    if protocol is IPProtocol.ICMP and is_ICMP_error(
        packet) and packet[IPv4].src == IPv4Address("
        0.0.0.0"): # the third one is a hack, that the src
        ip of error packet could still be determined after
        fwd table lookup, tell the by all-0 ip
        packet[IPv4].src = intf.ipaddr
    ...
    if packet[IPv4].ttl <= 1:
        if nested error:
            drop...
        expired = self.make_TTL_expired(packet) #
            TTL Expire Error
        return self.forward_a_packet(expired)
    packet[IPv4].ttl -= 1
    return self.ARP_lookup_and_send(aim_ip, packet,
        intf) # normal
```

Listing 3. Adapted forwarding

The Destination unreachable error is in ARP logic and handle_packet logic, which is like this. We do not display them here.

Now we display the generation codes here.(They are similar, so we only show one of them.) We display a helper function here first.

```
def skin_and_make_trace(self, orgpkt): # what a cruel
    name! lol
    assert orgpkt.has_header(Ethernet), "\033[31mDo not
        have Ethernet packet\033[0m"
    del orgpkt[Ethernet]
    return orgpkt.to_bytes()[:28]
```

Listing 4. Helper function

```
def make_TTL_expired(self, orgpkt):
    assert orgpkt.has_header(IPv4), "\033[31mDo not
        have IPv4 packet\033[0m" # To display
        hierarchic error in terminal, we use ANSI color
        escape sequence here
    expired = Packet()
```

```
4       ... # header constructors
5       expired[IPv4].dst = orgpkt[IPv4].src
6       ... # set IPv4 header here, notice that the src is TBD
            here.(all-0)
7       expired[ICMP].icmptype = ICMPType.TimeExceeded
8       expired[ICMP].icmpcode = ICMPTypeCodeMap[
            ICMPType.TimeExceeded].TTLExpired
9       expired[ICMP].icmpdata.data = self.
            skin_and_make_trace(orgpkt) # set the ICMP
            header
10      return expired
```

Listing 5. Error generation

Please refer to the annotated code for more details.

B. Explain ignoring nested ICMP Error

I think the reasons of ignoring error of ICMP Error could be summarized into to main reasons.

1) To avoid ICMP Error Storm (or 'ICMP Flood')
2) The 'error of error' is always meaningless

Let us discuss them with some examples. We talk about Reason 2 first.

Here is an example.

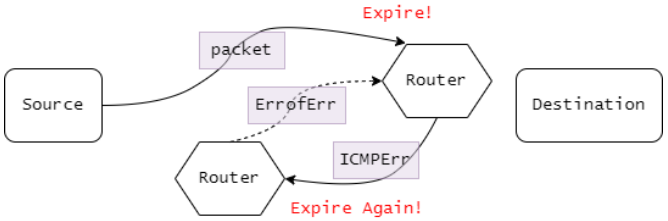

Fig. 1. meaningless

We talk about the router on the right. We could find that in the router view, it will never do any other behaviour whether it sends the TTL Expire message back or not, so it is meaningless for it to know its TTL Expire message is lost. Other type of error is similar.

Then we talk about Reason 1.

First we talk about it in a single router, please refer to Fig.2. We demonstrate it in the router's view. In this flow, the sent packet may circulate in the web for itself, take up the web resources.

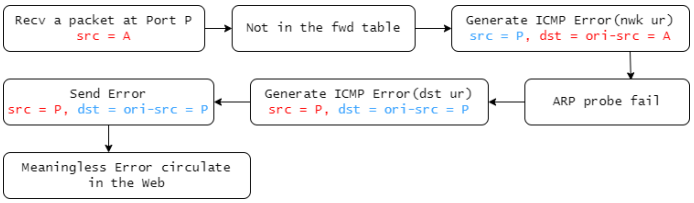

Fig. 2. ICMP Error Storm

Then We talk about within multi routers. Please refer to Fig.3. The solid line is real route, and the dashed line is the expected route.

In this case, the Error, Error of Error, Error of Error of Error, ... Error^n will fall in the loop of router2, router3 and router4. The loop may be common especially in wireless network, which takes up bandwidth permanently.

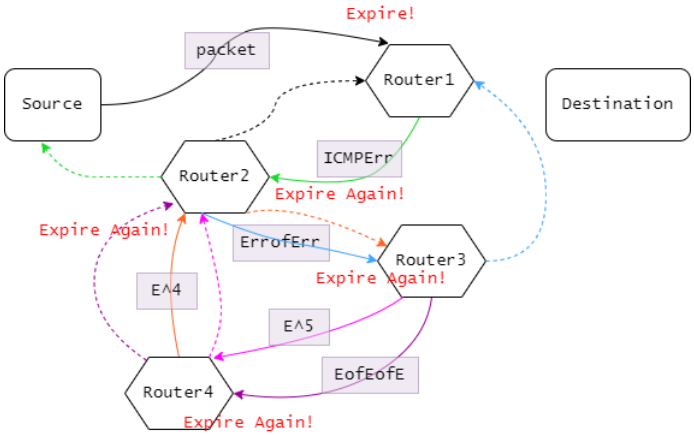

Fig. 3. ICMP Error Storm

So we SHOULD NOT make error response to ICMP Error.

C. Local Test

1) Offcial Test:

Please refer to Fig.4. for the official test result.

```
78  An icmp error message should out on eth0
        Expected event: send_packet(s) Ethernet
        10:00:00:00:00:01->23:33:33:33:33:34 IP | IPv4
        192.168.1.1->192.168.1.2 ICMP | ICMP TimeExceeded:TTLExpired
        28 bytes of raw payload
        (b'E\x00\x00,\x00\x00\x00\x00\x00\x11') OrigDgramLen: 0 out
        router-eth0
79  An UDP message should arrive on eth2
        Expected event: recv_packet IPv4 192.168.1.2->23.33.33.33
        UDP | UDP 10000->10000 | RawPacketContents (4 bytes) b'jntm'
        on router-eth1
80  An icmp error message should out on eth0
        Expected event: send_packet(s) Ethernet
        10:00:00:00:00:01->23:33:33:33:33:34 IP | IPv4
        192.168.1.1->192.168.1.2 ICMP | ICMP
        DestinationUnreachable:NetworkUnreachable 28 bytes of raw
        payload (b'E\x00\x00 \x00\x00\x00\x00\x00\x11') NextHopMTU:
        0 out router-eth0


All tests passed!
```

Fig. 4. Official test passed!

2) (Optional)My Test:

I make a simple trivial test at first. I make a simple trivial test first.(A normal ping reply).

Then I add a special test to the testscenario, which supplements the official test. In RFC 1812, the protocol requires that the router should not respond ICMP Error to packets with broadcast or loopback address. However, our official test does not pay attention to this rule. So I add a test to check this. (Please refer to Fig 5.)

```
1   def is_broadcast_or_lb_IP(self, ip, mask):
2       return ((int(ip) | int(mask)) == int(IPv4Address("
            255.255.255.255"))) or ip == IPv4Address("
            127.0.0.1")
```

Listing 6. The judge

```
Passed:
1   case 1 Input
2   case 1 arp Output
3   case 1 arp Input
4   case 1 Output
5   special test
6   special test

All tests passed!
```

Fig. 5. My test passed!

D. Deploy

1) Procedure:

server1 # ping -c 1 192.168.200.1

server1 # ping -c 1 -t 1 192.168.200.1

server1 # ping -c 1 172.16.1.1

(We must add a set-route entry in the start_mininet.py)

server1 # ping -c 1 10.1.1.3

server1 # traceroute -q 1 –sendwait=1 192.168.200.1

Let me explain the traceroute attributes. There actually exist many implementations of traceroute, I use the directly-installed traceroute tool (named 'traceroute' in Debian packet manager). There will be many many packets after traceroute. (Please refer to Fig.6)

3	0.107724043	192.168.100.1	192.168.200.1	UDP	74 57505 → 33435 Len=32
4	0.107769536	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
5	0.107893097	192.168.100.1	192.168.200.1	UDP	74 55525 → 33436 Len=32
6	0.107902859	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
7	0.108043240	192.168.100.1	192.168.200.1	UDP	74 50183 → 33437 Len=32
8	0.108051438	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
9	0.108206963	192.168.100.1	192.168.200.1	UDP	74 36459 → 33438 Len=32
10	[illegible]	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
11	[illegible]	192.168.100.1	192.168.200.1	UDP	74 36452 → 33439 Len=32
12	0.108373421	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
13	0.108521179	192.168.100.1	192.168.200.1	UDP	74 45849 → 33440 Len=32
14	0.108528635	192.168.200.1	192.168.100.1	ICMP	102 Destination unreachable (Port unreachable)
15	0.108655365	192.168.100.1	192.168.200.1	UDP	74 41664 → 33441 Len=32
16	0.108804289	192.168.100.1	192.168.200.1	UDP	74 56085 → 33442 Len=32
17	0.108949916	192.168.100.1	192.168.200.1	UDP	74 38092 → 33443 Len=32
18	0.109090882	192.168.100.1	192.168.200.1	UDP	74 57357 → 33444 Len=32
19	0.109351509	192.168.100.1	192.168.200.1	UDP	74 34570 → 33445 Len=32
20	0.109472780	192.168.100.1	192.168.200.1	UDP	74 55502 → 33446 Len=32
21	0.109614134	192.168.100.1	192.168.200.1	UDP	74 33430 → 33447 Len=32
22	0.109728703	192.168.100.1	192.168.200.1	UDP	74 41638 → 33448 Len=32
23	0.109842229	192.168.100.1	192.168.200.1	UDP	74 39658 → 33449 Len=32
24	0.109958323	192.168.100.1	192.168.200.1	UDP	74 55306 → 33450 Len=32

Visual Studio Code

Fig. 6. So many packets

These packets are for the detailed mechanism. It will send 3 copies of a TTL value in a row by default. Besides, the sender will keep sending incrementing TTL packets consecutively until the it receive a response.

So I use $-q1$ to make it only send one packet for each TTL value. And I use $--sendwait = 1$ to appoint the gap of sending packets to 1s, so that the host has enough time to be aware of the probing result to stop the consequent probing.

2) Outputs & Analysis:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.066290516	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
3	0.066307538	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3508, seq=1/256,
4	0.374550282	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x3508, seq=1/256,
5	9.407269573	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3509, seq=1/256,
6	9.434389802	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceed
7	38.662331989	192.168.100.1	172.16.1.1	ICMP	98	Echo (ping) request id=0x350a, seq=1/256,
8	38.688129349	192.168.100.2	192.168.100.1	ICMP	70	Destination unreachable (Network unreachab
9	60.132248397	192.168.100.1	10.1.1.3	ICMP	98	Echo (ping) request id=0x350c, seq=1/256,
10	65.142861325	192.168.100.2	192.168.100.1	ICMP	70	Destination unreachable (Host unreachable)
11	96.731794344	192.168.100.1	192.168.200.1	UDP	74	36763 → 33434 Len=32
12	96.814366730	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceed
13	97.731819541	192.168.100.1	192.168.200.1	UDP	74	43730 → 33435 Len=32
14	97.854799938	192.168.200.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)

Fig. 7. router-eth0 trace

- R0 Line 1-4, R1 Line 1-4, S1 Line 1-4, S2 Line 1-4: Server1 send ARP request to next hop(Router),

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:02	Broadcast	ARP	42	Who has 192.168.200.1? Tell 192.168.200.2
2	0.000038492	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	192.168.200.1 is at 20:00:00:00:00:01
3	0.104593107	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3508, seq=1/256,
4	0.104637299	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x3508, seq=1/256,
5	97.584656321	192.168.100.1	192.168.200.1	UDP	74	43730 → 33435 Len=32
6	97.584711261	192.168.200.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)

Fig. 8. router-eth1 trace

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.066299627	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
3	0.066311700	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3508, seq=1/256,
4	0.374560007	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x3508, seq=1/256,
5	9.407270028	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3509, seq=1/256,
6	9.434398872	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceed
7	38.662332681	192.168.100.1	172.16.1.1	ICMP	98	Echo (ping) request id=0x350a, seq=1/256,
8	38.688138115	192.168.100.2	192.168.100.1	ICMP	70	Destination unreachable (Network unreachab
9	60.132249168	192.168.100.1	10.1.1.3	ICMP	98	Echo (ping) request id=0x350c, seq=1/256,
10	65.142870589	192.168.100.2	192.168.100.1	ICMP	70	Destination unreachable (Host unreachable)
11	96.731795636	192.168.100.1	192.168.200.1	UDP	74	36763 → 33434 Len=32
12	96.814376801	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceed
13	97.731820592	192.168.100.1	192.168.200.1	UDP	74	43730 → 33435 Len=32
14	97.854809818	192.168.200.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)

Fig. 9. server1 trace

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:02	Broadcast	ARP	42	Who has 192.168.200.1? Tell 192.168.200.2
2	0.000025740	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	192.168.200.1 is at 20:00:00:00:00:01
3	0.104594323	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x3508, seq=1/256,
4	0.104632434	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x3508, seq=1/256,
5	97.584660847	192.168.100.1	192.168.200.1	UDP	74	43730 → 33435 Len=32
6	97.584706247	192.168.200.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)

Fig. 10. server2 trace

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
2	1.003295623	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
3	2.004485790	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
4	3.005798099	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
5	4.007769392	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2

Fig. 11. client trace

Routereplies to server1. The S1 sends Echo request. Router look up table and find eth1. Then sends ARP request to S2. S2 replies to router. Router forwards Echo request to S2. Then S2 sends Echo reply to S1, forwarded by router.(Please refer to Fig.12)

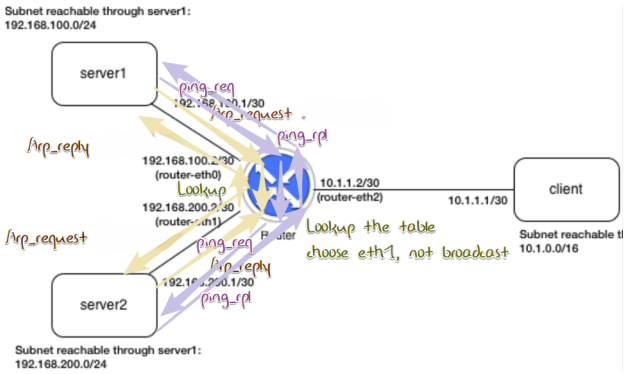

Fig. 12.

- R0 Line 5-6, S1 Line 5-6: Server1 sends ping request to with TTL=1. When it comes to router, the router sends TTL expired error to server1.(Please refer to Fig.13)
- R0 Line 7-8, S1 Line 7-8: Server1 sends ping request to with a nonexistent IP. When it comes to router, the router sends Network Unreachable error to server1. (Please refer to Fig.14)
- R0 Line 9-10, S1 Line 9-10, C Line 1-5: Server1 sends ping request to with nonexistent IP but in the forwarding table. It is aimed at eth2 of router. Router

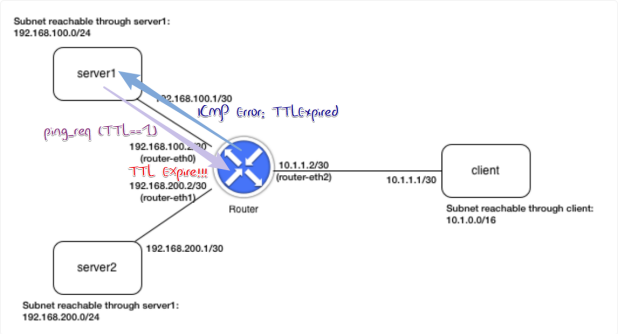

Fig. 13.

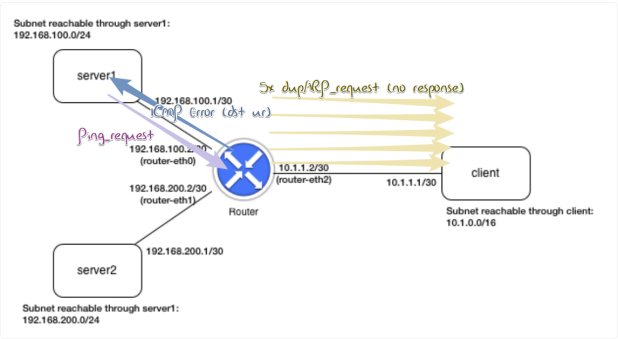

Fig. 14.

sends 5 ARP requests for the IP and aborts. Then it sends Network Unreachable error to server1. (Please refer to Fig.15)

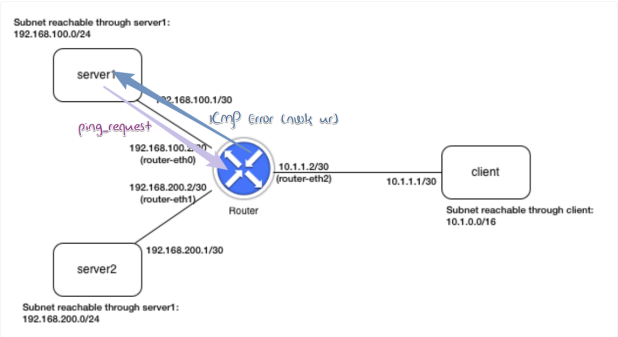

Fig. 15.

- R0 Line 11-14, R1 Line 5-6, S1 Line 11-14, S2 Line 5-6: After calling traceroute, S1 first sends UDP with TTL=1, it will expire at the router, and the router sends TTLExpired to S1. Then S1 sends UDP with TTL=2, it will be forward to S2. S2 sends ICMP Error port unreachable back. Then traceroute stops.(Please refer to Fig.16)
  (The traceroute sender will delibrately send a UDP packet with very large port number, to ensure the receiver will respond with ICMP port unreachable, in case of interrputing normal processes.)

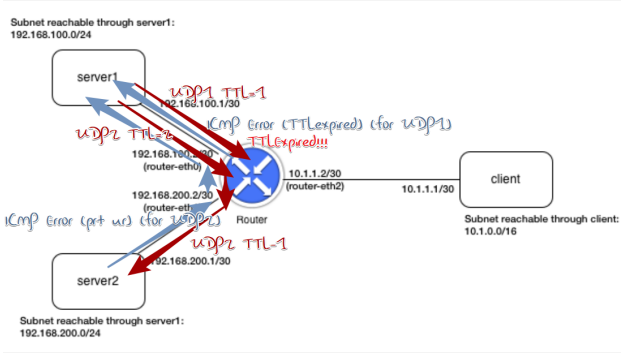

Fig. 16.

E. Troubleshooting

I come across some problem when copy the icmpdata domain from the original packet to the echo packet. Though the name is the same, it could not be copied directly.(It will not cause an error, but set all the icmpdata to 0 quietly.)

IV. (Appendix)Something Interesting

When we talk about ICMP, we could not let alone the DDoS attack with Ping.

So we can add some easy code to prevent basic DDoS attack, as below.

```
1  if time.time() - self.ddostimestamp > 1:
2      self.ddos_cnt = {}
3      self.ddostimestamp = time.time()
4  if packet[IPv4].src in self.ddos_cnt:
5      self.ddos_cnt[packet[IPv4].src] += 1
6      if self.ddos_cnt[packet[IPv4].src] > Router.
           DDoS_THRESHOLD:
7          return log_info(f"\033[31mDDoS attack from {
               packet[IPv4].src}\031[0m")
8  else:
9      self.ddos_cnt[packet[IPv4].src] = 1
```

Listing 7. DDoS prevention

Further, we could count the rate of ICMP traffic in all the traffic within a sliding window, and set a threshold to prevent DDoS attack and detect potential ICMP Error Storm. However, it may cause some problems to pass the testscenario (may has too much ICMP), so I do not add it into the code.

V. Feelings

I refactor my code from Lab 4 for better coping with the ICMP error generation. I reflect on the different control style then. I use the *flow style* in which all the cases need forwarding will pass special process and fall into normal forwarding flow. This kind of control is better when we use *goto* clause. But we can only use function call in Python, so the *branch style* is better.

We should choose proper control style according to the language we use.

Acknowledgment

Thanks for help from professor and TAs.