

# 面向带内网络的最优路径规划

陈思绵

## 摘要

随着网络复杂性的不断提高，细粒度的网络监视可以提高网络的可靠性与实时的反馈控制<sup>[1]</sup>。带内网络遥测（INT）通过将设备内部状态封装到探测数据包中，实现了更高效益的网络监视。而如何实现全局范围的网络流量遥测至今未有明确的定义<sup>[2]</sup>。在这项工作中，本文复现了论文（INT-PATH）<sup>[3]</sup>中提出的 DFS、Euler 两种路径规划算法，以最少的路径数生成覆盖整个网络的非重叠 INT 路径，同时编写 DFS、Euler 两种路径规划算法的客户端进行展示。根据论文的对比实验，设计了七个实验复现了这两种算法在不同网络拓扑的比较结果，实验结果与原文基本一致，即 Euler 在整体性能上优于 DFS 且适合部署在数据中心网络中。同时搭建遥测系统，复现了论文系统的整体工作，即根据用户指定的路径进行带内网络遥测，让用户更加直观地看出网络的流量状态。针对论文提出的遥测系统中存在的缺点（遥测路径需用户手动指定）进行改进，使得搭建的系统真正地实现全局范围的网络遥测。

**关键词：**带内网络遥测；非重叠 INT 路径；Euler 路径规划；网络流量状态

## 1 引言

随着网络规模的不断扩大，细粒度的网络监视已经成为提高网络可靠性与用户体验的前提之一<sup>[4]</sup>。在传统的网络监视中，广泛采用如 SNMP<sup>[5]</sup>管理协议周期性地轮询路由器或交换机的 CPU，以便每隔几秒或几分钟收集设备的内部状态。然而，由于控制平面和数据平面之间的持续的数据交互以及有限的 CPU，这种粗粒度的查询方式会造成较大的查询延迟。在如今高密度、高动态的数据中心网络无法进行很好地扩展。

为了改善可扩展性问题，P4 语言联盟（P4.org）提出了带内网络遥测（INT），实现细粒度实时数据平面监控<sup>[6]</sup>。INT 允许数据包在通过数据平面时查询设备的内部状态，如队列深度、延迟，而无需控制平面 CPU 的额外干预。通常，INT 依赖于探测分组（具在分组报头中保留的可变长度标签栈）。INT 代理周期性地生成探测分组并将其注入到网络中，其中探测分组将被排列并与后台业务一起转发。在沿着转发路径的每个路由器或交换机中，探测分组将提取设备的内部状态并将它们写入 INT 标签栈。探测包会在最后一跳的时候被转发到控制器进行分析。

INT 本质上是一个底层原语，需要特殊硬件的支持才能获取内部状态。通过调用数据接口，网络运营商可以获取探测路径的实时流量状态。但为了提高网络管理，需搭建 INT 高层机制，以便高效地提取全网流量状态。随着软件定义网络（SDN）的日趋完善<sup>[7]</sup>，控制器可对全局网络范围做出最优的控制策略。此外，通过机器学习的网络自动化管理需要来自外部环境的反馈数据进行训练<sup>[8]</sup>。

本文复现了论文（INT-path）的全局范围的流量遥测框架。将源路由嵌入 INT 探测包中，以允许探测包通过网络路由到达指定的目标点。基于该路由机制，复现了两种路径规划策略来生成覆盖整个网络的非重叠 INT 路径。第一种是基于深度优先搜索（DFS），其实现方式较为简单但时间效率高。第二种算法是基于欧拉的算法，可以生成具有最少路径数的非重叠 INT 路径。

本文的主要工作概述如下：

- 两种图覆盖路径规划策略。基于 DFS 与 Euler 的路径规划算法，可以生成覆盖整个网络的非重叠 INT 路径。基于 Euler 的算法可以生成更优的最少路径数，从而最大限度地减少遥测开销。
- 设计算法对比实验。对比两种算法在不同网络拓扑的实现效果，实验结果表明 Euler 在整体性能上优于 DFS，且适合部署在当今的数据中心网络中。
- 搭建及改进遥测系统。复现论文提出的基于源路由的遥测机制，即将 INT 探测器与源路由标签耦合，通过用户指定的路径进行遥测。同时针对论文提出的遥测系统中存在的缺点（遥测路径需用户手动指定）进行改进，使得搭建的系统真正地实现全局范围的网络遥测。

## 2 相关工作

INT 路径的提出是建立在以往研究的基础之上。Bosshart 等人<sup>[9]</sup>发明了 P4，一种用于协议无关包处理器编程的高级语言，它使网络运营商能够任意修改分组报头。INT<sup>[6]</sup>是一个使用 P4 的遥测应用程序，它可以在不干扰控制器的情况下获取设备的内部状态。而 INT 本身是用于遥测一个或多个设备的底层程序，而没有定义如何实现全局网络范围的遥测，这需要上层的控制设计。有一个非常相关的作品名为“Barefoot Deep Insight”，这也是建立在 INT 的基础之上，使端到端的流量监测<sup>[10]</sup>。然而该类似一种解决方案，没有透露任何的技术细节。Van 等人<sup>[11]</sup>进行了相关的遥测实现，但仍没有涉及触及高层设计。2015 年，Pingmesh<sup>[12]</sup>在微软的一个数据中心进行了类似的全网络遥测。然而 Pingmesh 仅依赖于终端主机的反馈，而不深入网络设备。随着 AI 时代的兴起，许多网络 AI 系统声称依靠实时的流量状态采集来构建闭环式控制系统<sup>[13]</sup>。然而，他们很少谈论遥测的具体细节<sup>liu2021dr</sup>。总的来说，论文提出的方法可以是该领域有力的补充。

## 3 本文方法

### 3.1 本文方法概述

根据论文的设计思路，分步进行复现工作，主要有路径算法复现与系统复现两部分的工作。第一部分根据提出 DFS、Euler 两种图覆盖路径规划策略，生成非重叠的路径覆盖整个网络，进而获取计算路径的结果，对比评估不同算法之间的效果。第二部分根据计算结果，部署 INT 系统根据用户输入的指定路径，进行网络遥测。INT 的实现框架如图 1 所示：

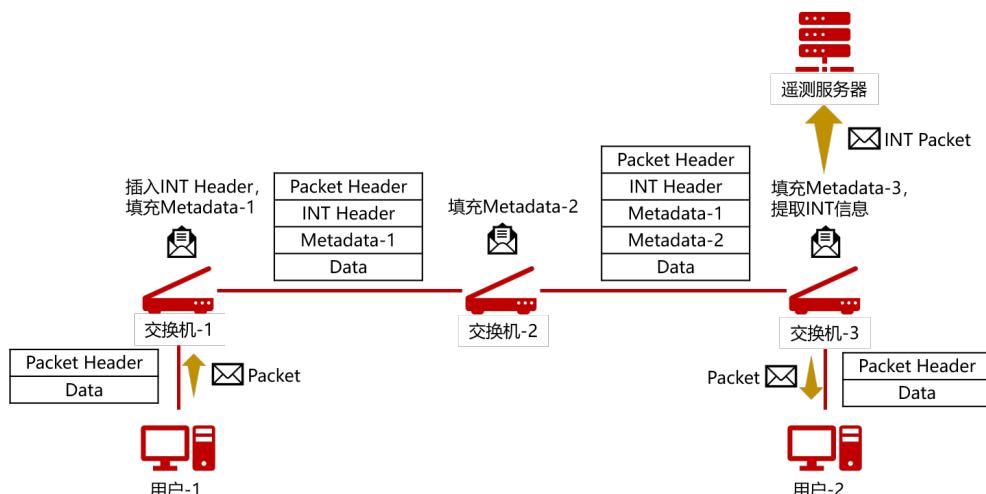


图 1: INT 框架示意图

带内网络遥测（INT）由遥测服务器和具备带内网络遥测功能的交换机组成<sup>[14]</sup>。其数据包处理流程如下：1、数据报文到达带内网络遥测系统的第一个交换节点时，带内网络遥测模块通过在交换机上设置的采样方式匹配并镜像出该报文，根据遥测任务的需要在四层头部后插入 INT 头，将 INT 头所指定的遥测信息封装成元数据（MetaData）插入到 INT 头部之后；2、报文转发到中间节点时，设备匹配 INT 头部后插入元数据；3、报文转发到带内网络遥测系统最后一跳时，交换设备匹配 INT 头插入最后一个元数据并提取全部遥测信息并通过 RPC 等方式转发到遥测服务器；4、遥测服务器解析遥测报文内的遥测信息，上报给上层遥测应用程序。

### 3.2 DFS-based 路径规划

基于深度优先搜索（DFS），基本思想在回溯之前将访问的顶点连续添加进当前的路径中，具体过程如图 2 所示：

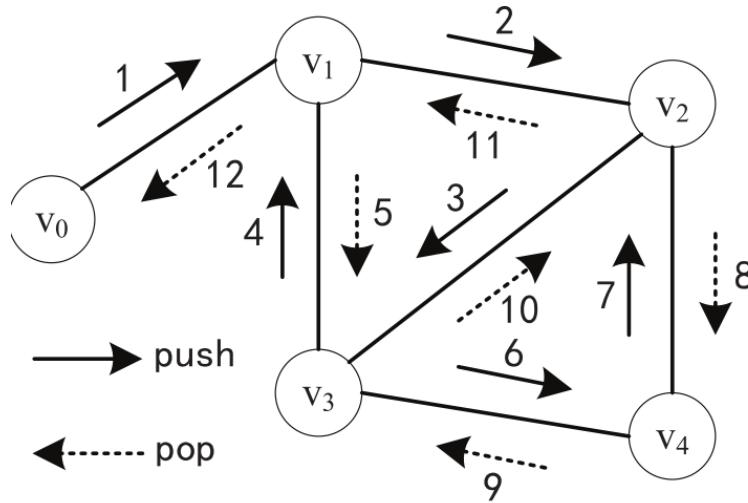


图 2: DFS 图覆盖路径规划

图中  $Path1 = \{v_0, v_1, v_2, v_3, v_1\}$ ,  $V_3$  标识为分叉 s 点，它是回溯路径的第一个顶点，且具有未访问的边。基于  $V_3$  创建一条新路径，即  $path2 = \{v_3, v_4\}$ ，当  $path2$  拓展到  $\{v_3, v_4, v_2\}$  时，就得回溯。当调用的栈为空时递归停止，最后提取了两个非重叠的 INT 探测路径（ $path1$  和  $path2$ ）。

### 3.3 Euler trail-based 路径规划

DFS 不能将生成的路径数降到最低，增加了遥测的开销，并且对平衡路径没有进行优化。其中平衡路径即是对路径的长度进行处理，使得是路径尽可能的相似，利用欧拉环性质精确访问每个边的路径，主要有以下四个图论的基本定理：(1) 没有奇数顶点的连通图具有欧拉回路；(2) 只有一个奇数顶点的连通图不存在；(3) 具有两个奇数顶点的连通图，欧拉轨迹是从一个顶点到另一个顶点；(4) 具有  $2K$  个奇数顶点的连通图，至少包含  $K$  个不同路径，这些路径遍历了图的所有边。其中奇数顶点是指具有奇数度的顶点。为了达到理论上的最小值，选取的路径应是从一个奇数顶点到另一个奇数顶点结束欧拉算法迭代提取一对奇数顶点之间的路径，直到提取完图中的所有边（每个顶点的度数变成 0），对于  $2K$  个奇数顶点图，生成  $K$  个不重叠监测路径，覆盖图的所有边（理论最小值），难点在于提取路径会将一个连通图分割成多个子图。欧拉图覆盖路径算法的主要思想是主要思想是迭代提取一对奇数顶点之间的路径，直到提取完图中的所有边。而提取路径具体分成三个情况：1、如果包含两个奇数点，只需在两个奇数顶点之间提取一条欧拉路径；2、如果有两个以上的奇数点，需要在任意一对奇数顶点之间提取欧拉路径；3、如果没有奇数顶点，则提取一条遍历所有顶点的欧拉路径。其

具体实现过程如图 3 所示。

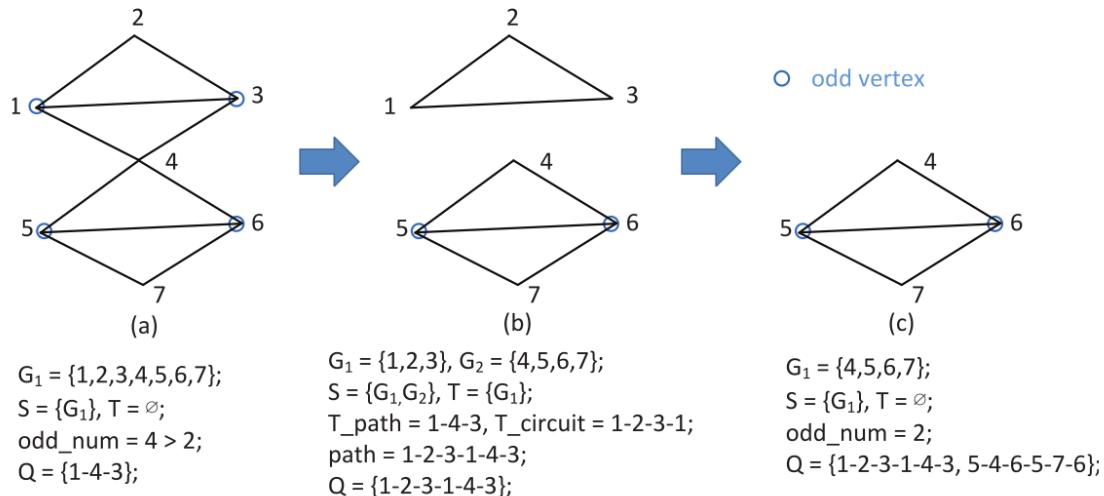


图 3: DFS 图覆盖路径规划

图中的 Q 表示路径集，存放非重叠的 INT 路径，图中的拓扑提取了两条非重叠的路径。传统的 Euler 算法则使用 Dijkstra 返回两个奇数顶点之间的最短路径，会使两个奇数顶点之间生成的路径通常太短，如果之后的路径无法正确加入形成更长的欧拉路径，则会导致最后生成的路径不太平衡。因此，在此基础上进行算法的优化，具体做法不是计算任意两个奇数顶点之间的最短路径，而是计算两个奇数顶点之间的所有路径，并从所有的路径中返回最长的路径。但这种做法也存在一点不足，即由于图更新，需重新计算很耗时。解决这个问题需要进行剪枝操作，即在路径提取过程中删除边，不添加进图中，如果计算出的最短与提取的欧拉路径没有交点则无需重新计算最短路径，这样便可按需更新路径。

### 3.4 遥测系统

该系统提出了三种不同功能的逻辑路由器，分别为 INT 生成器、INT 转发器与 INT 收集器。其系统框图如图 4 所示。

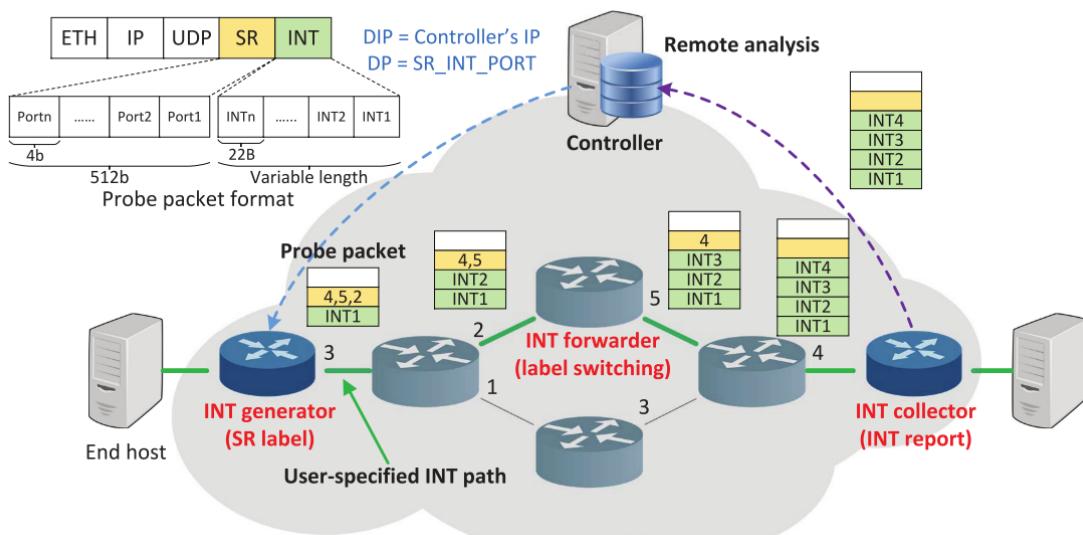


图 4: DFS 图覆盖路径规划

**INT 生成器：**在监测路径的第一条产生 SR-INT 探测数据包，P4 不能直接从数据平面生成数据包，所以由主机定期生成空探针，当探针到达数据平面时，INT 生成器重写数据标头并分配 SR 标签，并在

转发数据包之前使用 `header.setValid()` 添加本地的 INT 信息，INT 生成器会将输出的端口 ID 写入 SR 标签栈中。INT 转发器：根据传入流量的 DP 号执行 SR-INT 探针或流量数据包的转发。如 DP 号是“SR-INT-PORT”，INT 转发器执行标签交换并根据 SR 标签中端口 ID 转发数据包。每一跳将 SR 标签右移 4 位，在路由器上弹出一次 SR 标签。在转发之前 INT 转发器还会将本地的 INT 信息写入 INT 标签中。INT 收集器：将探测数据包转发给控制器进一步分析（此时 DIP 写满了控制器的 IP）。

## 4 复现细节

### 4.1 与已有开源代码对比

该篇论文有开源的核心算法代码进行参考，但在展示两种算法时使用重写了两种核心算法进行演示，如需获取 DFS、Euler 的计算路径节点等，动画的演示采用 C++ 编码的 ImageUI 框架。算法的对比实验需要设计正确的拓扑，此部分与论文开源的代码需要进行较多的改正，进一步编码进行实验数据的采集及展示。遥测系统需自己手动搭建，配置好相关的环境镜像，P4 语言的编写改动及环境的配置会耗费大量的时间。本文在系统部署了六台 P4 交换机以及 5 台主机进行网络遥测，并将遥测队列数据包的数量写入数据库。总的来说，本文对论文已开源的源码进行直接编写及修正的代码量 Python (800 多行)、C (70 多行)、P4 (50 多行)，不包括客户端整体代码的编写。

### 4.2 实验环境搭建

本文使用的系统环境是 Ubuntu16.04。主要需配置好 P4 的编译环境。

### 4.3 图覆盖路径规划界面分析与使用说明

运行启动程序，程序的主界面如图 5 所示。



图 5: 程序主界面

首先展示 DFS 图覆盖路径规划，点击 DFS 初始拓扑按钮，生成随机拓扑图，其中紫色线条表示初始的拓扑关系，红色线条表示寻找的路径，绿色线条表示寻找的结果，最终寻找的路径覆盖整个网络。具体过程如图 6 所示，图中 DFS 算法生成了两条路径来覆盖该网络拓扑。



图 6: DFS 图覆盖路径规划

其次展示 Euler 图覆盖路径规划算法，生成动态欧拉拓扑，运行欧拉算法寻找路径，生成最少的路径数来覆盖整个网络。具体过程如图 7 所示。



图 7: Euler 图覆盖路径规划

#### 4.4 遥测系统分析与使用说明

根据论文提出的遥测系统，在虚拟机中搭建了 6 台 P4 交换机以及 5 台主机，其对应的网络拓扑如图 8 所示。

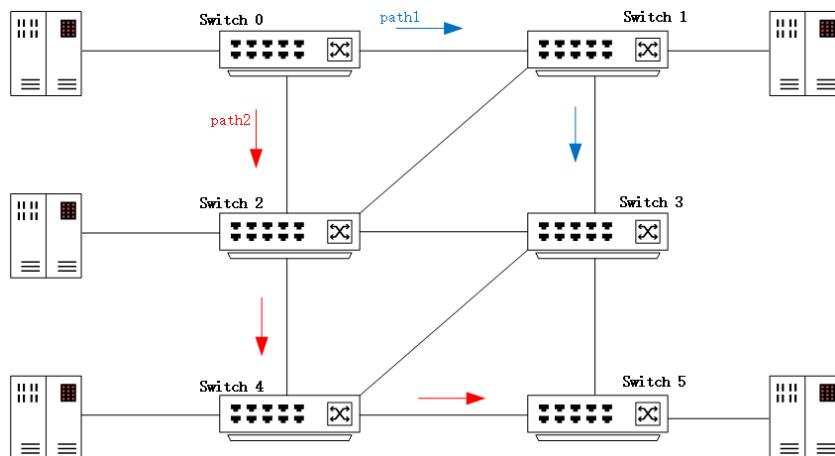


图 8: 系统拓扑及指定的遥测路径

首先启动遥测系统，根据用户输入指定的路径进行遥测，最终成功连接各台主机，各台主机的遥测信息如图 9 所示。

```
*** h0 : ('ifconfig')
eth0 Link encap:Ethernet HWaddr 00:01:00:00:00:00
      inet addr:10.0.0.100 Bcast:10.0.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

*** h1 : ('ifconfig')
eth0 Link encap:Ethernet HWaddr 00:01:00:00:00:01
      inet addr:192.168.0.101 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

h1-eth1 Link encap:Ethernet HWaddr 82:25:9f:30:05:a4
      inet addr:192.168.0.101 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

```
*** h2 : ('ifconfig')
eth0 Link encap:Ethernet HWaddr 00:01:00:00:00:02
      inet addr:10.0.0.102 Bcast:10.0.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:486 (486.0 B)

h2-eth1 Link encap:Ethernet HWaddr 82:25:9f:30:05:a5
      inet addr:192.168.0.102 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:486 (486.0 B)

lo Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

*** h3 : ('ifconfig')
eth0 Link encap:Ethernet HWaddr 00:01:00:00:00:03
      inet addr:192.168.0.103 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

h3-eth1 Link encap:Ethernet HWaddr 82:25:9f:30:05:a6
      inet addr:192.168.0.103 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

```
start connect
('connecting ', '192.168.8.100')
('socket gen success', 'h0')
('192.168.8.100', 'connected')
('connecting ', '192.168.8.101')
('socket gen success', 'h1')
('192.168.8.101', 'connected')
('connecting ', '192.168.8.102')
('socket gen success', 'h2')
('192.168.8.102', 'connected')
('connecting ', '192.168.8.104')
('socket gen success', 'h4')
('192.168.8.104', 'connected')
('connecting ', '192.168.8.105')
('socket gen success', 'h5')
('192.168.8.105', 'connected')
('now act id ', 1)
('action', -1)
(:'portslists', 0, [[1, 2, 5], [2, 3, None]])
(:'addressList', 0, [])
mysql connect success
mysql search success
end
```

A) 系统中各台主机的信息

B) 成功连接各台主机

图 9: 启动遥测系统

并将遥测数据写入了数据库，数据库中的数据如图 10 所示。

```
mysql> select * from fnl_int;
+----+-----+-----+
| id | device_no | deq_qdepth |
+----+-----+-----+
| 1  | 0         | 2          |
| 2  | 1         | 10         |
| 3  | 2         | 3          |
| 4  | 3         | 5          |
| 5  | 4         | 0          |
| 6  | 5         | 7          |
+----+-----+-----+
6 rows in set (0.00 sec)

mysql> █
```

图 10: 启动遥测系统

根据写入的结果，可视化展示探测路径的流量状态，让用户更加直观得看出结果，具体如图 11 所示。从图中可以看出节点 2 的流量状态最为拥挤，与数据库的数据相对应，即节点 2 队列中的探测包最多。

*Telemetry*

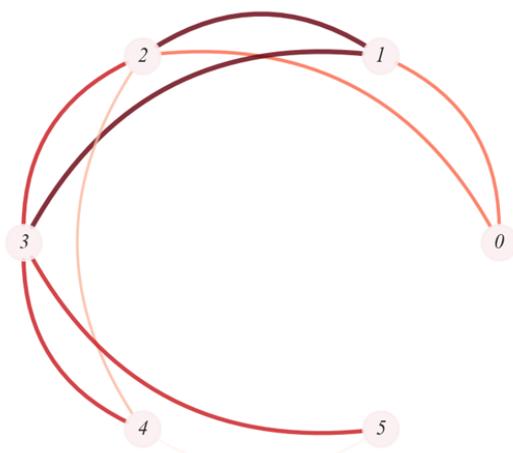


图 11: 遥测流量状态

## 4.5 创新点

本项目的创新点主要在于针对论文提出的遥测系统进行改进，实现动态实时的网络遥测，具体的实现过程即探测的路径不是用户指定输入，而是根据上诉欧拉路径规划算法计算的结果写入文件，系统自动计算结果，并不断获取系统的网络流量状态进行结果展示的更新，其实现效果如图 12 所示。

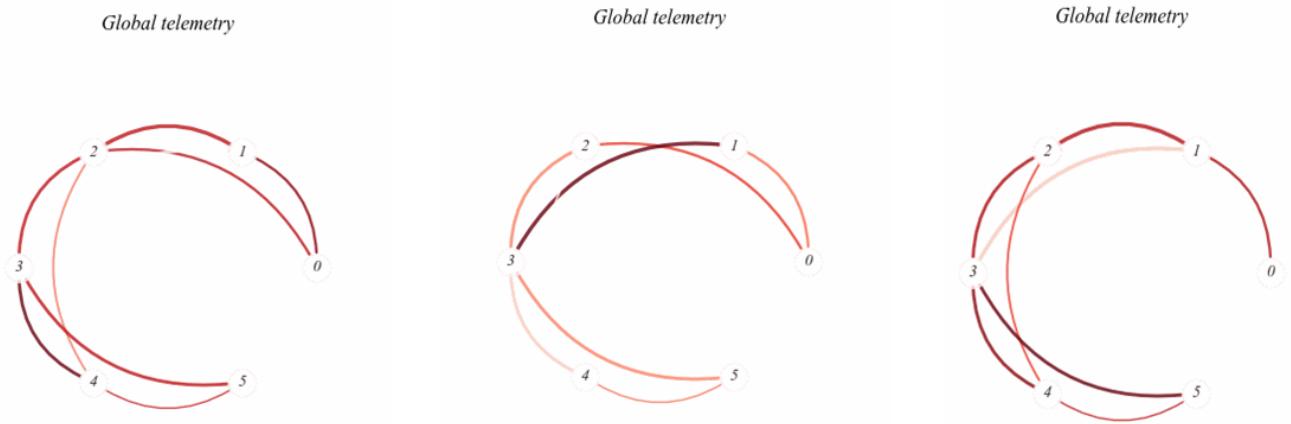


图 12: 全局遥测流量状态

## 5 实验结果分析

算法的对比实验，主要是根据论文的算法对比实验，设计了七个不同的对比实验，来评估 DFS、Euler 两种算法在不同网络拓扑的情况下结果，其最终的实验结果与论文基本一致，即 Euler 在整体效果上优于 DFS，且适合部署在当今的数据中心网络。其中各个算法对比实验的分析如下所示：

实验一：采用随机生成网络拓扑规模的做法，不断增加网络拓扑的顶点数，比较 DFS、Euler 两种算法生成的覆盖整个拓扑所需的路径数量。实验结果如图 13 所示，左侧为自己复现的实验结果，右侧为原文的实验结果，实验结果与原文基本一致，即在不同网络规模拓扑的情况下 Euler 生成的覆盖整个拓扑的路径数量比 DFS 更少。

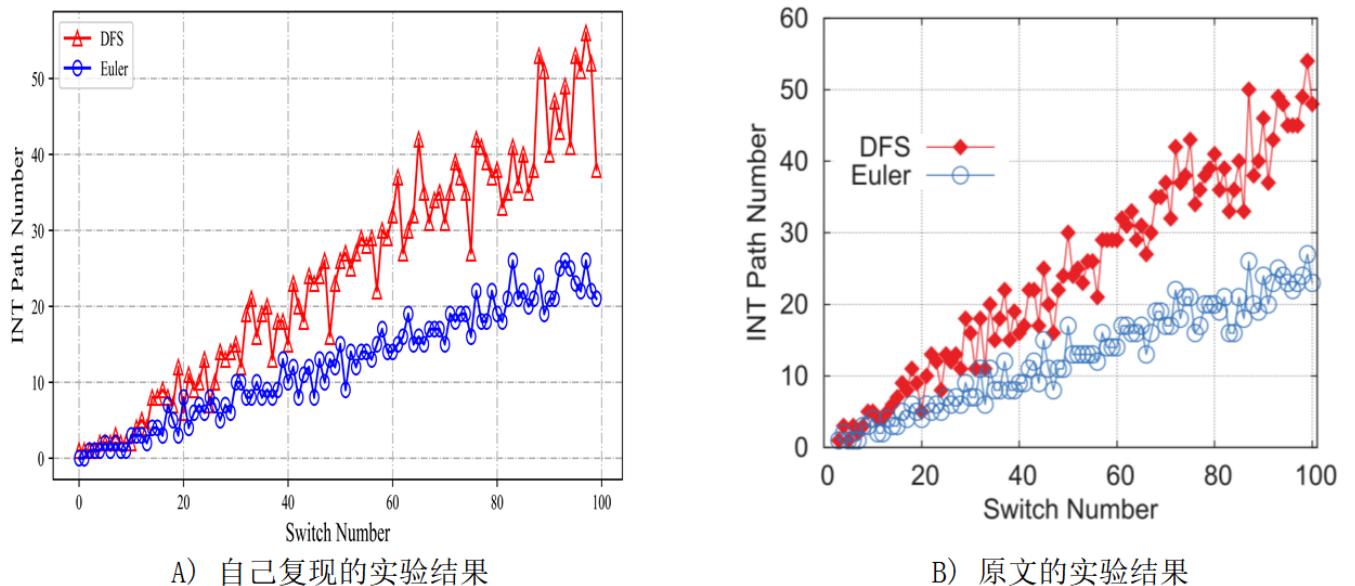


图 13: 实验一

实验二：采用固定奇数，不断增加拓扑顶点数的做法，对两种算法生成的路径数进行比较。实验结果如图 14 所示，实验结果与原文基本一致，即当固定图中奇数顶点的数量时，无论网络增长到什么

规模，欧拉生成的路径数都是一个常数。

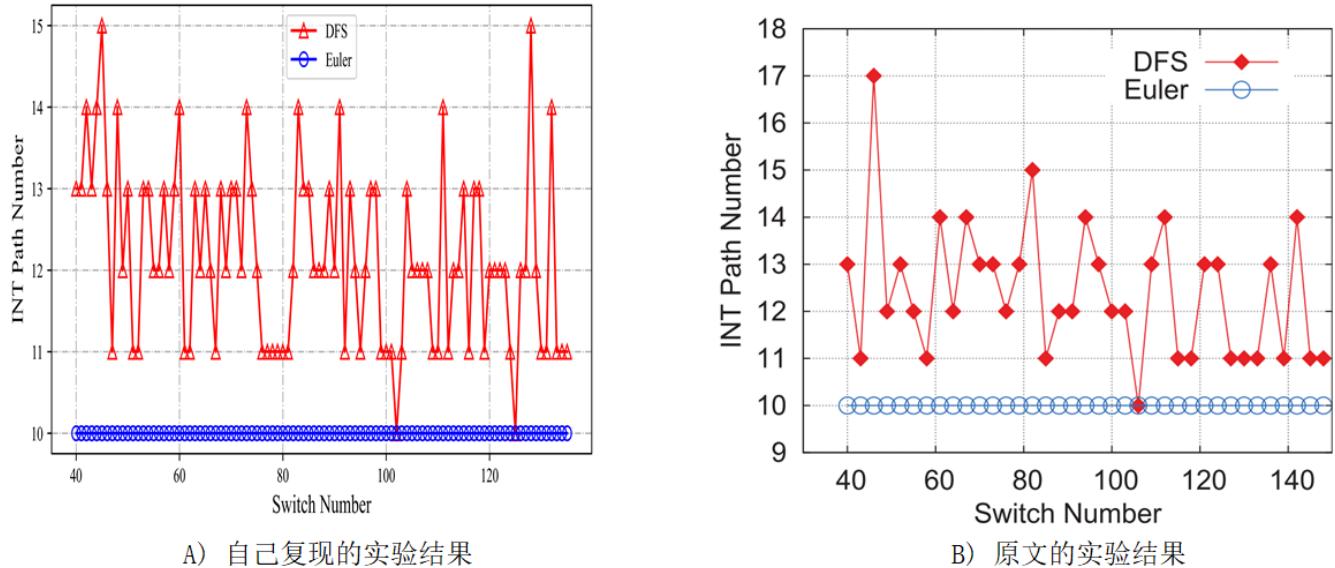


图 14: 实验二

实验三：采用固定顶点数，不断添加边进去的拓扑进行实现，比较两种算法的路径生成数。该实验需注意无向图边数与顶点数的关系，即  $0 \leq k \leq n(n-1)/2$ 。实验结果如图 15 所示。通过结果可以发现由 DFS 和欧拉生成的路径数都是先增长后下降。这是因为在开始时添加更多的边使得图变得更复杂，具有更多的奇顶点，而在结束时，图最终变成没有奇顶点的完全图。欧拉仍然优于 DFS 生成更少的非重叠路径。

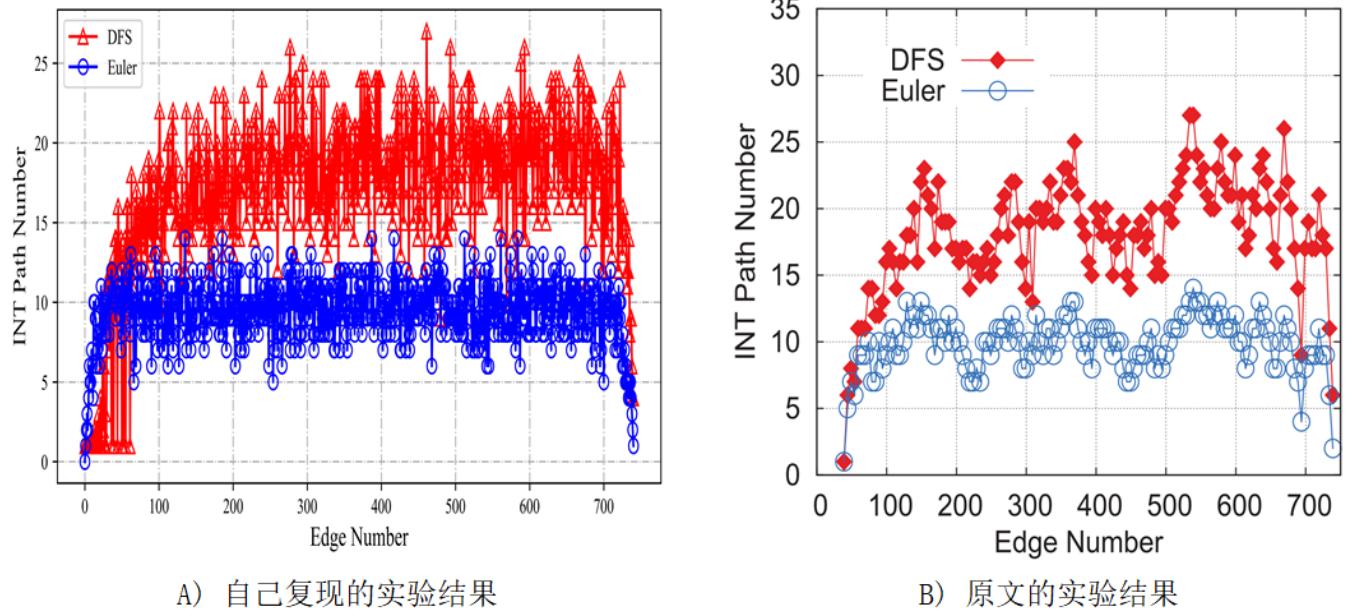


图 15: 实验三

实验四：采用随机生成拓扑的做法，在一个固定顶点的情况下随机生成三个拓扑，并分别采集三组数据进行标准差的计算，比较两种算法的路径长度的标准差。实验结果如图 16 所示，该结果与原文的实验结果基本一致，即在随机生成拓扑的情况下，由优化的欧拉算法生成的路径长度标准差与没有优化欧拉算法相比变得更小。

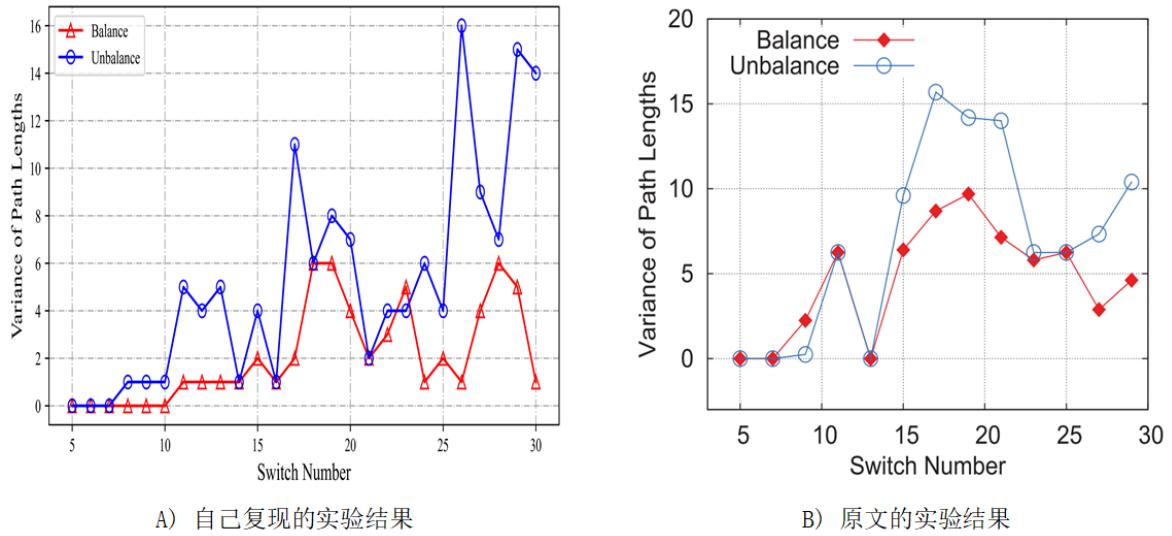


图 16: 实验四

实验五：采用固定奇数顶点，不断增加顶点数的拓扑，比较两种算法的 CPU 执行时间。结果如图 17 所示。该实验结果与原文的数据不太相似，原因在于不同电脑硬件配置的 CPU 时钟是不一致的，但可以得出与原文同样的结论，即欧拉路径的执行时间较大且对奇数顶点更为敏感。

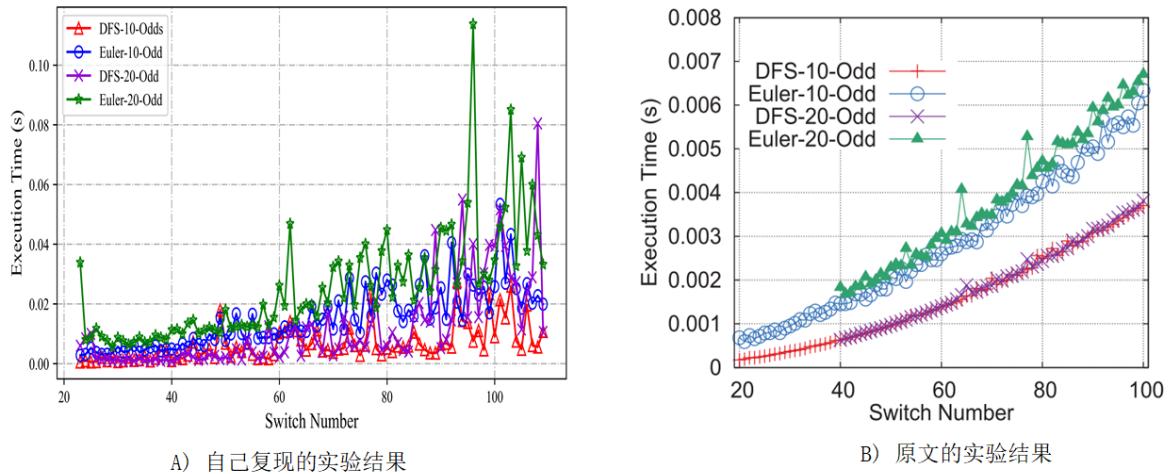


图 17: 实验五

实验六：采用在固定顶点数的情况下，不断增加该拓扑的奇数顶点数，需注意任何图度数为奇数顶点的个数是偶数，记录 Euler 路径算法的执行时间，呈现线性增长的规律，实验结果如图 18 所示。该结果同时也验证了原文中对 Euler 时间复杂度的证明。

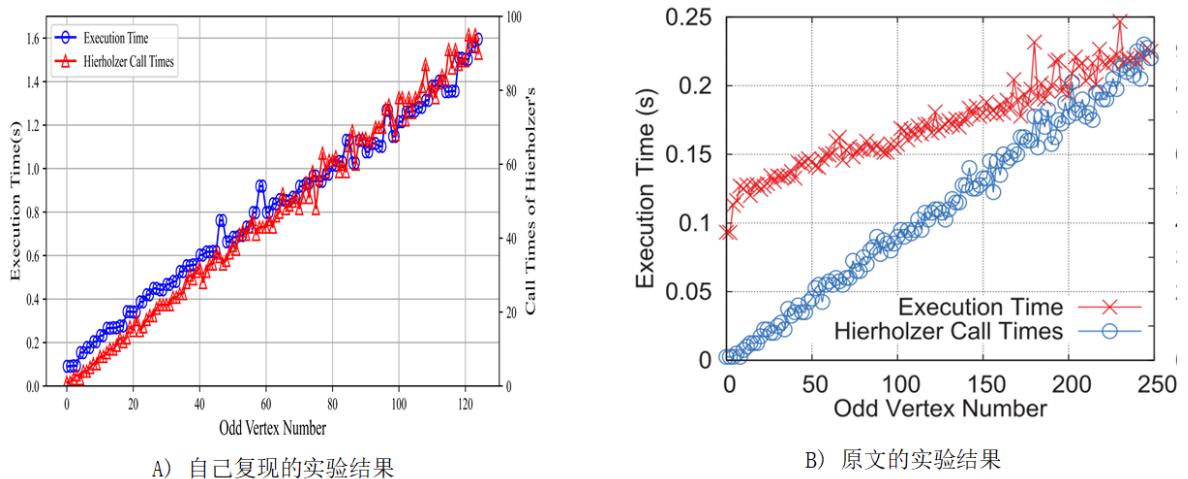
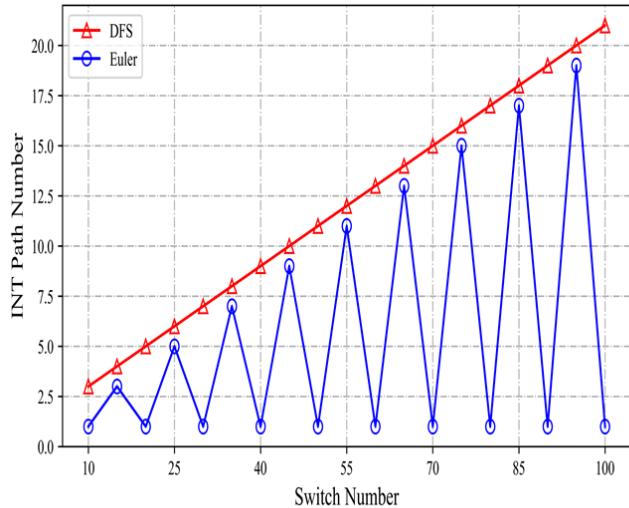
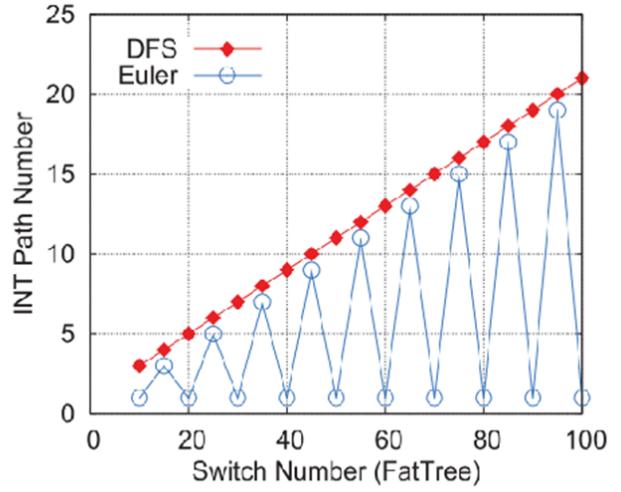


图 18: 实验六

实验七：构建两种数据中心拓扑结构（Fattree、LeafSpine），不断增加这两种拓扑的顶点数，比较 Euler 与 DFS 两种路径规划算法的路径生成数。图 19 为 Fattree 网络拓扑的实验结果，图 20 为 LeafSpine 网络拓扑实验结果。从图中显示的结果可以看出 Euler 在两种数据中心网络拓扑的效果仍然比 DFS 更好，即生成的路径数比 DFS 少。

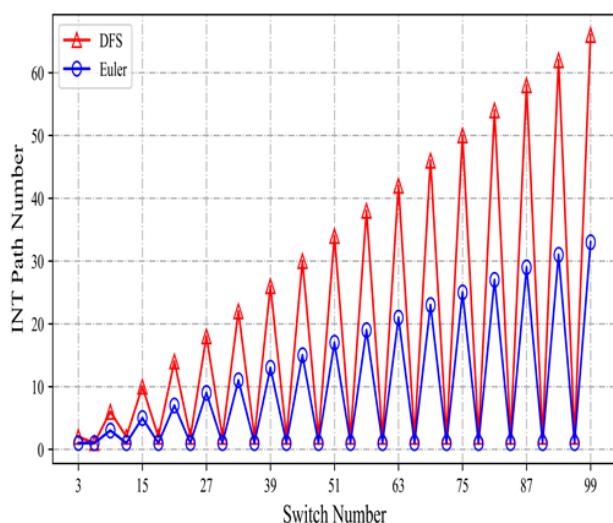


A) 自己复现的实验结果

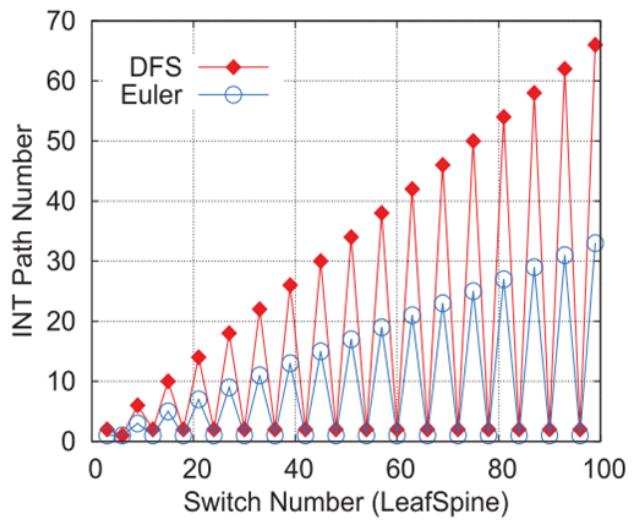


B) 原文的实验结果

图 19: 实验七 (Fattree)



A) 自己复现的实验结果



B) 原文的实验结果

图 20: 实验七 (LeafSpine)

## 6 总结与展望

本文基本复现了论文的整体工作，通过所做的路径算法对比实验可以得出与原文一致的结论，即 Euler 在整体效果上优于 DFS。但通过实验四的实验发现原文对 Euler 算法的优化并不明显，源码细节是对没优化的 Euler 算法采取 fleury (佛罗莱) 的计算方式，而不是采用普遍的 Euler (欧拉) 计算方式来计算，才能把这两种算法的实验数据拉开。

本次课程的复现作业让我有机会接触到了网络遥测领域的研究，由于复现及研究时间有限，对 P4 的研究并不深入，可编程交换机的研究在网络遥测领域具有很高的学术价值。同时该论文也给了我很多启发，例如论文的解决方案是用于探测，但如何进行调度并未展开进一步的研究，因此未来的研究方向可沿着该方向进行探索。

## 参考文献

- [1] THORUP M. Undirected single-source shortest paths with positive integer weights in linear time[J]. *Journal of the ACM (JACM)*, 1999, 46(3): 362-394.
- [2] GUO C, YUAN L, XIANG D, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. 2015: 139-152.
- [3] PAN T, SONG E, BIAN Z, et al. Int-path: Towards optimal path planning for in-band network-wide telemetry[C]//IEEE INFOCOM 2019-IEEE Conference On Computer Communications. 2019: 487-495.
- [4] HIERHOLZER C, WIENER C. Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren[J]. *Mathematische Annalen*, 1873, 6(1): 30-32.
- [5] CASE J D, FEDOR M, SCHOFFSTALL M L, et al. Simple network management protocol[J], 1988.
- [6] KIM C, SIVARAMAN A, KATTA N, et al. In-band network telemetry via programmable dataplanes[C] //ACM SIGCOMM: vol. 15. 2015.
- [7] YAN Q, YU F R, GONG Q, et al. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges[J]. *IEEE communications surveys & tutorials*, 2015, 18(1): 602-622.
- [8] MESTRES A, RODRIGUEZ-NATAL A, CARNER J, et al. Knowledge-defined networking[J]. *ACM SIGCOMM Computer Communication Review*, 2017, 47(3): 2-10.
- [9] BOSSHART P, DALY D, GIBB G, et al. P4: Programming protocol-independent packet processors[J]. *ACM SIGCOMM Computer Communication Review*, 2014, 44(3): 87-95.
- [10] LYNN B M. Barefoot Sanders[J]. *SMU Law Review*, 2016, 62(5): 1709.
- [11] VAN TU N, HYUN J, HONG J W K. Towards onos-based sdn monitoring using in-band network telemetry[C]//2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). 2017: 76-81.
- [12] BANERJEE S. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis–Public Review[J].
- [13] SUNSHINE C A. Source routing in computer networks[J]. *ACM SIGCOMM Computer Communication Review*, 1977, 7(1): 29-33.
- [14] MARQUES J A, LUIZELLI M C, TAVARES DA COSTA FILHO R I, et al. An optimization-based approach for efficient network monitoring using in-band network telemetry[J]. *Journal of Internet Services and Applications*, 2019, 10(1): 1-20.