实验三报告

区块链数据分析

Lab 3: Blockchain Data Analysis

姓名：范舒舰

学号：23121677

专业：网络空间安全

2026 年 4 月 13 日

目录

1 实验目标

1. 理解区块链作为公开账本的数据组织方式，掌握区块数据与费用数据的基本含义。
2. 学会整理和分析链上区块信息，理解区块活跃度、资源使用情况与网络运行状态之间的关系。
3. 围绕区块链费用机制，对连续区块中的费用数据进行统计与可视化分析。
4. 训练利用链上数据描述区块链网络状态的能力，并通过图表和文字分析形成基本的数据分析结论。

2 实验背景与实验要求

2.1 实验背景

区块链不仅是交易执行和智能合约运行的平台，也是一个持续增长的公开账本。链上的每一个区块都记录了时间、交易数量、资源消耗和费用信息，这些数据能够客观反映网络在某一时间段内的运行状态。

本实验要求你从数据分析的视角出发，对连续区块中的基础数据进行整理、汇总和分析。重点关注两类核心问题：一是区块活动情况，二是费用变化趋势。通过本实验，你应能够从原始链上数据中提取有意义的信息，并用直观的图表和严谨的结论对网络状态进行描述。注意：由于本实验需要分析 Base Fee 字段，实验应基于支持 EIP-1559 费用机制的以太坊网络（如以太坊主网或 Sepolia 等测试网）开展。

2.2 实验要求

2.2.1 任务一：区块数据汇总分析

收集最近一段连续区块的数据，并对其中至少 10 个区块进行详细的汇总分析。获取的每个区块至少应包含以下字段：

- 区块高度（Block Number）
- 出块时间（Timestamp）
- 区块内交易数量（Transaction Count）
- 费用接收地址（区块头 beneficiary，对应部分 RPC 返回字段 miner）
- 消耗的 Gas（Gas Used）
- Gas 上限（Gas Limit）

在此数据基础上，完成以下分析任务：

- 描述最近区块的活跃程度变化（如哪些区块交易数量较多，哪些较少）。
- 比较不同区块的资源使用情况，分析 Gas Used 占 Gas Limit 的比例（即区块填充程度）。
- 结合时间与区块数据，说明链上活动在短时间内是否存在明显的波动。
- 对上述数据进行归纳，形成对当前网络基本状态的总体判断。

2.2.2 任务二：费用机制分析

围绕一段连续区块中的费用数据，分析网络费用的宏观变化趋势。请收集至少 20 个连续区块的费用相关数据，重点关注 `Base Fee`、`Gas Used` 和 `Gas Limit`，并完成以下分析：

- 绘制 Base Fee 随区块高度变化的趋势图。
- 观察 Base Fee 的变化是否平稳，指出是否存在明显的上升或下降区间。
- 结合区块的填充程度（Gas Used 占比），分析 Base Fee 的调整与区块资源使用程度之间是否存在逻辑关联。
- 简要说明当前的费用机制反映出了网络怎样的拥堵程度或活跃程度变化。

2.2.3 核心数据与分析要求

本实验不强调具体的数据获取实现方式（可自由选择 Web3 RPC、以太坊浏览器 API 或第三方数据平台），而强调是否能够围绕链上数据完成规范的整理与分析表达。在分析层面，应至少体现以下能力：

- 能够将原始、底层的区块字段整理为可读的结构化信息。
- 能够通过专业的图表和简要的文字分析组织实验内容，做到图文并茂。
- 能够基于客观数据对网络状态作出解释，拒绝单纯的字段罗列。
- 能够从“区块活跃度”和“费用变化”两个独立又相关的角度，对链上运行情况进行立体说明。

3 开发环境说明

本实验开发环境如下：

- 数据来源网络：Sepolia Testnet
- 区块浏览器：Sepolia Etherscan
- 数据获取方式：Python 爬虫 + 区块浏览器页面校验
- Python 依赖库：`requests`、`BeautifulSoup4`、`pandas`
- 数据整理工具：Excel
- 图表绘制工具：Python `matplotlib`
- 文档编写工具：LaTeX

4 理论基础

4.1 区块数据的基本含义

区块是区块链账本中的基本组织单位。每个区块都包含区块头信息和交易列表，其中区块高度用于标识区块顺序，时间戳用于描述出块时间，交易数量用于衡量该区块内的链上活动规模，而 `Gas Used` 与 `Gas Limit` 则反映区块资源使用情况。

4.2 区块填充率与资源使用

区块填充率通常表示为 `Gas Used / Gas Limit`。填充率越高，说明区块资源使用越接近上限；填充率越低，则说明该区块可用资源较为充足。通过对连续区块填充率的观察，可以分析链上负载是否平稳，以及是否存在短时波动。

4.3 EIP-1559 与 Base Fee 机制

在支持 EIP-1559 的以太坊网络中，`Base Fee` 是由协议自动调整的基础费用。若前一区块资源使用偏高，则后续区块的 `Base Fee` 往往上升；反之则可能下降。因此，`Base Fee` 的变化与区块资源使用情况存在内在联系，但这种联系具有一定滞后性。

5 数据设计说明

5.1 任务一所需字段设计

为完成区块数据汇总分析，实验记录了连续 10 个区块的区块高度、出块时间、交易数量、费用接收地址、`Gas Used`、`Gas Limit` 以及由此计算得到的区块填充率。

表 1: 任务一区块基础数据字段说明

字段名	作用说明
Block Number	标识区块顺序，用于构造连续区块样本
Timestamp	表示区块产生时间，用于观察短时间内的活动变化
Transaction Count	表示区块内交易数量，用于衡量区块活跃程度
Fee Recipient	表示费用接收地址信息，对应区块头中的 beneficiary 字段，在部分 RPC 接口中对应 miner 字段
Gas Used	表示区块实际消耗的 Gas
Gas Limit	表示区块允许使用的 Gas 上限
Fill Rate	即 `Gas Used / Gas Limit`，用于描述区块填充程度

5.2 任务二所需字段设计

为完成费用机制分析，实验进一步记录了连续 20 个区块的 `Base Fee`、`Gas Used`、`Gas Limit` 和区块填充率，并在此基础上完成趋势分析与可视化。

表 2: 任务二费用分析字段说明

字段名	作用说明
Block Number	用于构造横轴并保证样本连续
Base Fee	反映协议自动调整的基础费用水平
Gas Used	用于说明区块资源实际消耗情况
Gas Limit	用于计算区块填充率
Fill Rate	用于分析 `Base Fee` 与资源使用之间的关系

6 实验步骤（任务一）

6.1 步骤一：确定实验网络与连续区块区间

实验网络选择为 Sepolia Testnet。为满足实验要求，选取连续区块 `10643000` 至 `10643019` 作为总体样本，其中区块 `10643000` 至 `10643009` 用于任务一的基础数据汇总分析，区块 `10643000` 至 `10643019` 用于任务二的费用机制分析。

6.2 步骤二：通过区块浏览器确认连续区块信息

首先打开 Sepolia Etherscan 首页，并进入区块列表页面，观察连续区块的区块号、交易数量、费用接收地址、Gas Used、Gas Limit 与 Base Fee 等字段。图1 展示了实验

所选样本附近的连续区块列表页面。

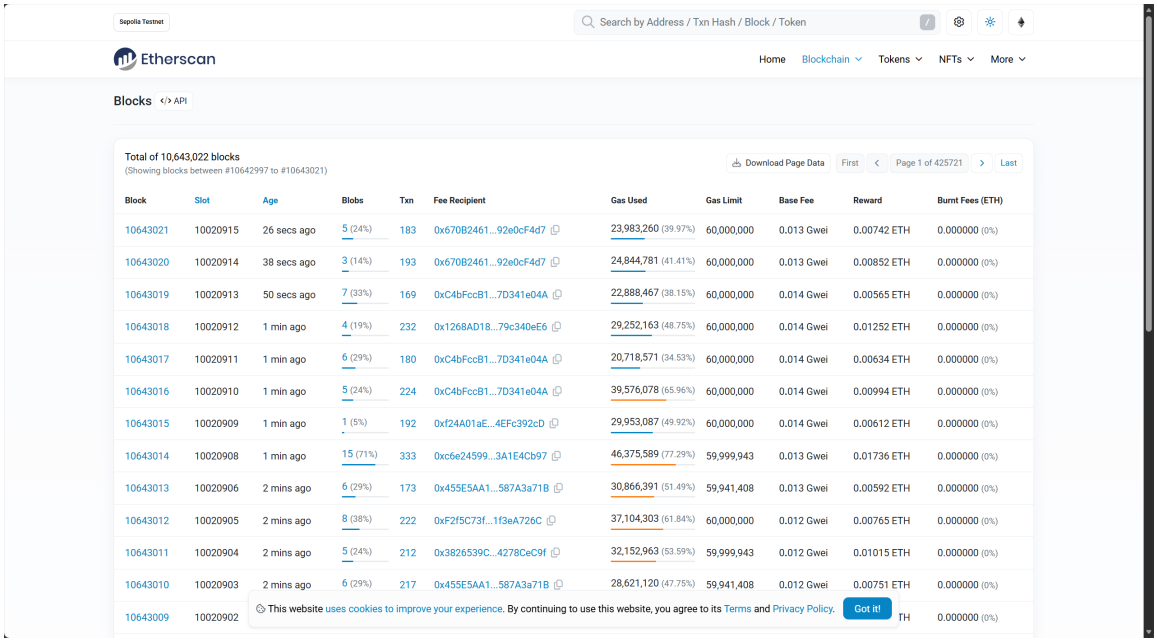

图 1: Sepolia Etherscan 区块列表页面

6.3 步骤三：编写并运行 Python 程序抓取区块数据

在确定区块范围后，编写 Python 脚本自动抓取区块 `10643000` 至 `10643019` 的详细信息，并将结果保存为 Excel 文件。脚本从区块详情页中提取区块高度、时间戳、交易数量、费用接收地址、Gas Used、Gas Limit、Base Fee 等字段，再生成任务一与任务二所需的两个工作表。

6.4 步骤四：整理任务一数据表

将前 10 个区块的基础数据整理如下表所示。该样本时间范围从 `Apr-12-2026 08:58:36 AM +UTC` 到 `Apr-12-2026 09:00:24 AM +UTC`，总时长约为 108 秒，能够反映极短时间窗口内链上活动的波动情况。

表 3: 连续 10 个区块基础数据汇总

区块高度	出块时间	交易数	费用接收地址	Gas Used	Gas Limit	填充率
10643000	Apr-12-2026 08:58:36 AM +UTC	225	0x13CB6AE34A13a0977F4d7101eBc24B87Bb23F0d5	27,486,907	60,000,000	45.81%
10643001	Apr-12-2026 08:58:48 AM +UTC	198	0x9A6034c84cd431409Ac1a35278c7Da36FfDa53E5	31,839,103	60,000,000	53.07%

区块高度	出块时间	交易数	费用接收地址	Gas Used	Gas Limit	填充率
10643002	Apr-12-2026 08:59:00 AM +UTC	177	0x9b7E335088762aD8061C04D08C37902ABC8ACb87	27,245,817	60,000,000	45.41%
10643003	Apr-12-2026 08:59:12 AM +UTC	209	0x5cC0DdE14E7256340CC820415a6022a7d1c93A35	32,018,107	60,000,000	53.36%
10643004	Apr-12-2026 08:59:24 AM +UTC	201	0xc6e2459991BfE27cca6d86722F35da23A1E4Cb97	31,913,653	60,000,000	53.19%
10643005	Apr-12-2026 08:59:36 AM +UTC	197	0x455E5AA18469bC6ccEF49594645666C587A3a71B	34,390,457	59,941,408	57.37%
10643006	Apr-12-2026 08:59:48 AM +UTC	168	0x25941dC771bB64514Fc8abBce970307Fb9d477e9	22,270,859	59,999,943	37.12%
10643007	Apr-12-2026 09:00:00 AM +UTC	43	0x4dF6EB2EC570B58cC64f540247A8AdFA11F1Cf63	3,226,995	60,000,000	5.38%
10643008	Apr-12-2026 09:00:12 AM +UTC	323	0x670B24610DF99b1685aEAC0dfD5307B92e0cF4d7	52,320,112	60,000,000	87.20%
10643009	Apr-12-2026 09:00:24 AM +UTC	198	0x3826539Cbd8d68DCF119e80B994557B4278CeC9f	31,757,394	60,000,000	52.93%

6.5 步骤五：分析区块活跃度与资源使用情况

为更直观地观察任务一区块活跃度变化，进一步绘制连续 10 个区块的交易数量变化图，如图2 所示。

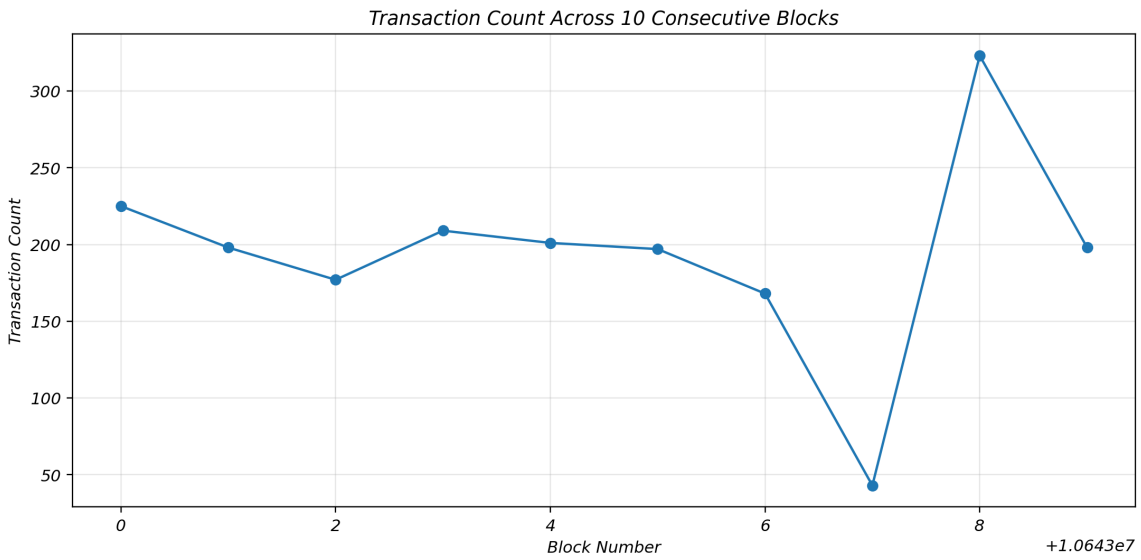

图 2: 连续 10 个区块的交易数量变化图

7 实验步骤（任务二）

7.1 步骤一：整理任务二费用分析表

在任务一基础上，将样本扩展至 20 个连续区块，并记录每个区块的 `Base Fee`、`Gas Used`、`Gas Limit` 与填充率。总时间跨度约为 240 秒。

表 4: 连续 20 个区块费用分析数据汇总

区块高度	Base Fee (Gwei)	Gas Used	Gas Limit	填充率
10643000	0.013325	27,486,907	60,000,000	45.81%
10643001	0.013185	31,839,103	60,000,000	53.07%
10643002	0.013286	27,245,817	60,000,000	45.41%
10643003	0.013134	32,018,107	60,000,000	53.36%
10643004	0.013244	31,913,653	60,000,000	53.19%
10643005	0.013350	34,390,457	59,941,408	57.37%
10643006	0.013596	22,270,859	59,999,943	37.12%
10643007	0.013158	3,226,995	60,000,000	5.38%
10643008	0.011690	52,320,112	60,000,000	87.20%
10643009	0.012777	31,757,394	60,000,000	52.93%
10643010	0.012871	28,621,120	59,941,408	47.75%
10643011	0.012798	32,152,963	59,999,943	53.59%
10643012	0.012913	37,104,303	60,000,000	61.84%
10643013	0.013295	30,866,391	59,941,408	51.49%

区块高度	Base Fee (Gwei)	Gas Used	Gas Limit	填充率
10643014	0.013345	46,375,589	59,999,943	77.29%
10643015	0.014256	29,953,087	60,000,000	49.92%
10643016	0.014253	39,576,078	60,000,000	65.96%
10643017	0.014821	20,718,571	60,000,000	34.53%
10643018	0.014248	29,252,163	60,000,000	48.75%
10643019	0.014204	22,888,467	60,000,000	38.15%

7.2 步骤二：绘制 Base Fee 趋势图

基于任务二表格数据，以区块高度为横轴、`Base Fee` 为纵轴绘制趋势图。图3 展示了连续 20 个区块中 Base Fee 的变化情况。

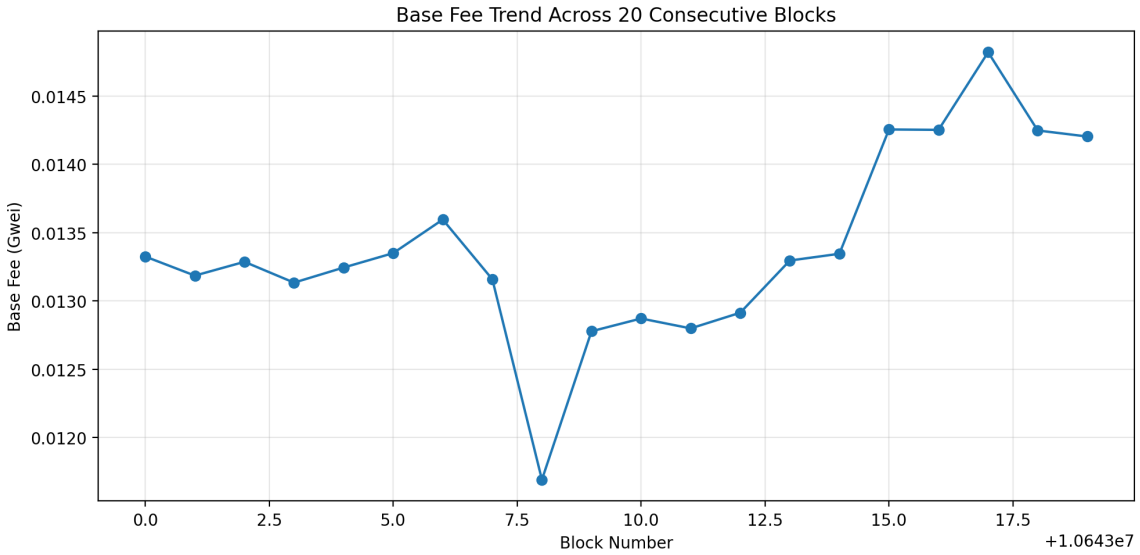

图 3: 连续 20 个区块的 Base Fee 变化趋势图

7.3 步骤三：绘制区块填充率辅助图

为分析费用变化与资源使用之间的关系，进一步绘制连续 20 个区块的区块填充率变化图，如图4 所示。

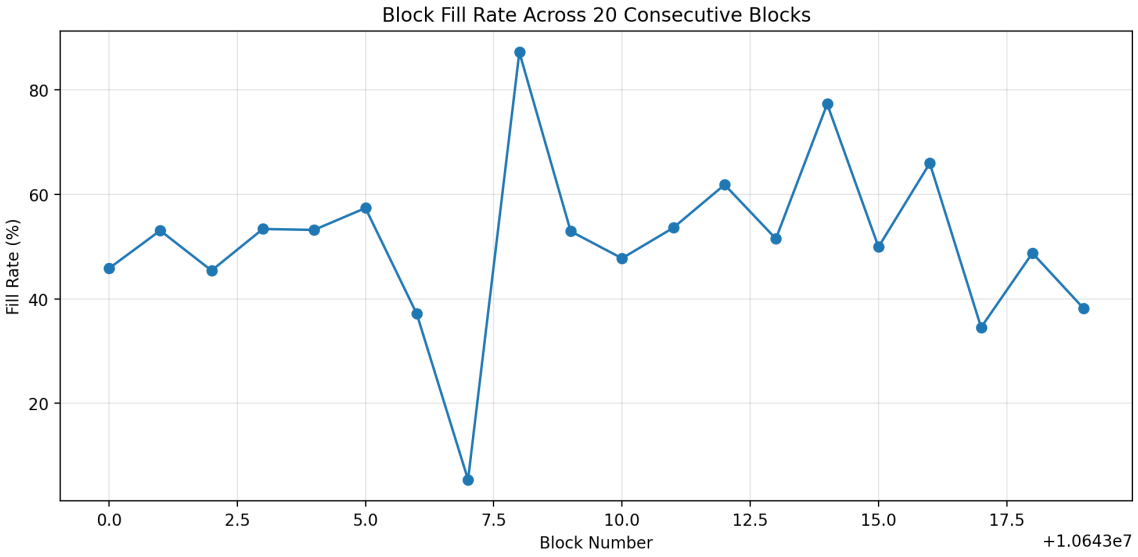

图 4: 连续 20 个区块的填充率变化图

8 结果验证与分析

8.1 数据来源、字段完整性与图表对应关系

本实验的样本区块均来自 Sepolia Etherscan。图1 展示了连续区块列表页面，页面中同时给出了区块号、交易数量、费用接收地址、Gas Used、Gas Limit 与 Base Fee 等字段；表3 与表4 则分别对任务一与任务二所需字段进行了结构化整理。结合 Python 抓取脚本自动导出的 Excel 结果，可以确认本实验已经覆盖任务一要求的区块高度、出块时间、区块内交易数量、费用接收地址、Gas Used 与 Gas Limit 等字段，并覆盖任务二要求的 Base Fee、Gas Used 与 Gas Limit 等费用相关字段。由此可见，数据来源真实、字段完整，图表与表格之间能够相互印证。

8.2 最近一段区块链网络活动整体是否稳定

在任务一选取的 10 个连续区块中，平均交易数量约为 193.9 笔，平均填充率约为 49.08%。从总体水平看，该时间段内的网络活动并非低活跃状态，说明网络在短时间窗口内保持了较为连续的运行状态；但若结合交易数量和填充率的具体变化，则可以发现网络运行虽然整体稳定，但并不是完全平滑，而是伴随着较明显的瞬时波动。

8.3 区块交易数量是否存在明显波动

由图2 可见，任务一区块交易数量存在明显波动。交易数量最高的区块为 10643008，达到 323 笔；交易数量最低的区块为 10643007，仅有 43 笔交易。特别是在区块 10643007 与区块 10643008 之间，交易数量出现了由极低到极高的快速跳变，说明链上活动在极短时间内存在显著波动，而非匀速变化。

8.4 区块资源使用情况与区块活跃度说明

任务一中 10 个区块的填充率在 5.38% 到 87.20% 之间变化，波动范围较大。其中，填充率最高的区块为 10643008，达到 87.20%；填充率最低的区块为 10643007，仅为 5.38%。这一结果表明，在同一短时间段内，不同区块的资源使用水平差异明显。结合交易数量变化可以看出，区块活跃度较高时通常伴随着更高的资源消耗，而交易数量极低的区块则对应明显偏低的资源使用水平。

8.5 Base Fee 的变化趋势特征

在任务二的 20 个连续区块中，Base Fee 的平均值约为 0.013387 Gwei，最小值出现在区块 10643008，约为 0.011690 Gwei；最大值出现在区块 10643017，约为 0.014821 Gwei。由图3 可见，Base Fee 在区块 10643000 至 10643007 之间总体围绕 0.013 Gwei 附近小幅波动，可视为相对平稳阶段；在区块 10643008 处出现明显下探，构成显著下降点；在区块 10643008 之后，Base Fee 逐步回升，并在区块 10643015 至 10643017 之间形成较明显的上升区间，随后略有回落。因此，该时间段内的 Base Fee 呈现出“前期相对平稳—中部明显下探—后期持续回升”的趋势特征。

8.6 区块资源使用情况与费用变化之间的逻辑关联

将图3 与图4 对照可以发现，Base Fee 与区块填充率之间并非简单的同区块同步关系，而更符合 EIP-1559 机制下的滞后调节逻辑。典型例子是区块 10643008：该区块填充率高达 87.20%，但其 Base Fee 却处于本样本最低值。这说明该区块继承的是前一区块较低资源压力条件下形成的基础费用；而在随后多个区块中，协议逐步吸收高填充率带来的资源压力后，Base Fee 才表现出持续抬升，并最终在区块 10643017 达到峰值。由此可见，区块资源使用情况与费用变化之间存在明确的逻辑关联，但这种关联表现为短时滞后的动态调整，而不是瞬时同步响应。

8.7 100 条连续区块的扩展验证

为避免仅基于 10 个或 20 个区块样本得出局部结论，实验进一步对区块 10643000 至 10643099 的 100 个连续区块进行了扩展统计分析。扩展样本在区块数量与时间跨度上均显著大于基础样本，因此更适合观察链上运行状态在较长连续区间内的整体趋势与波动特征。该扩展分析不替代实验要求中的 10 区块与 20 区块分析，而是作为对原结论的补充验证。

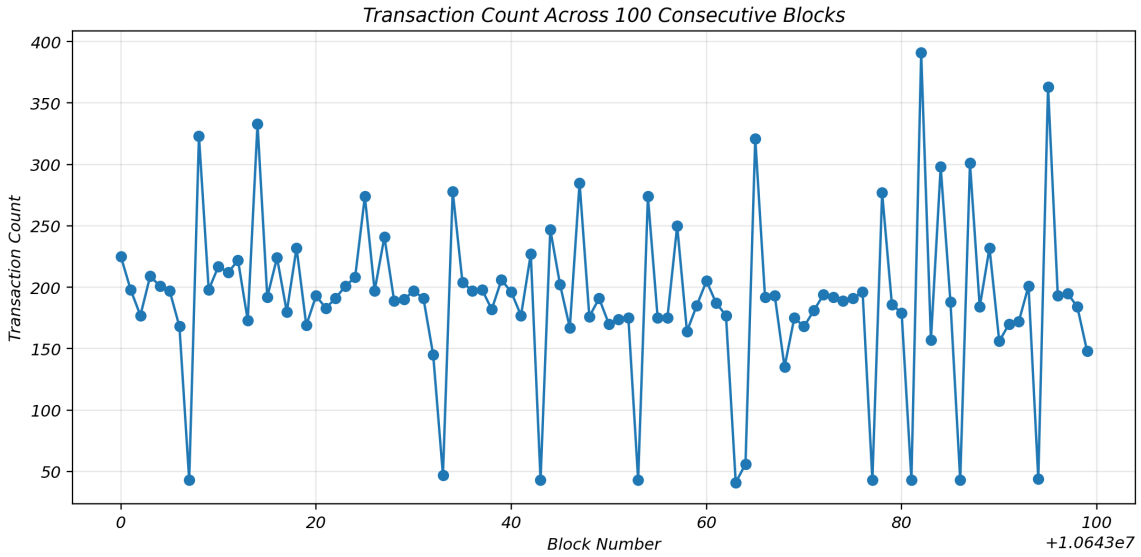

图 5: 100 个连续区块的交易数量变化图

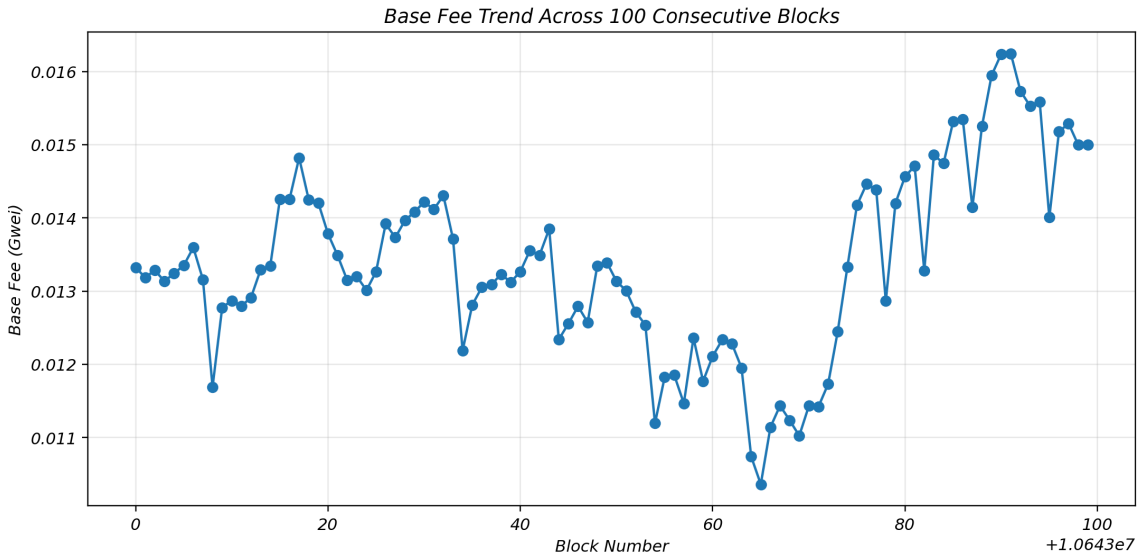

图 6: 100 个连续区块的 Base Fee 变化趋势图

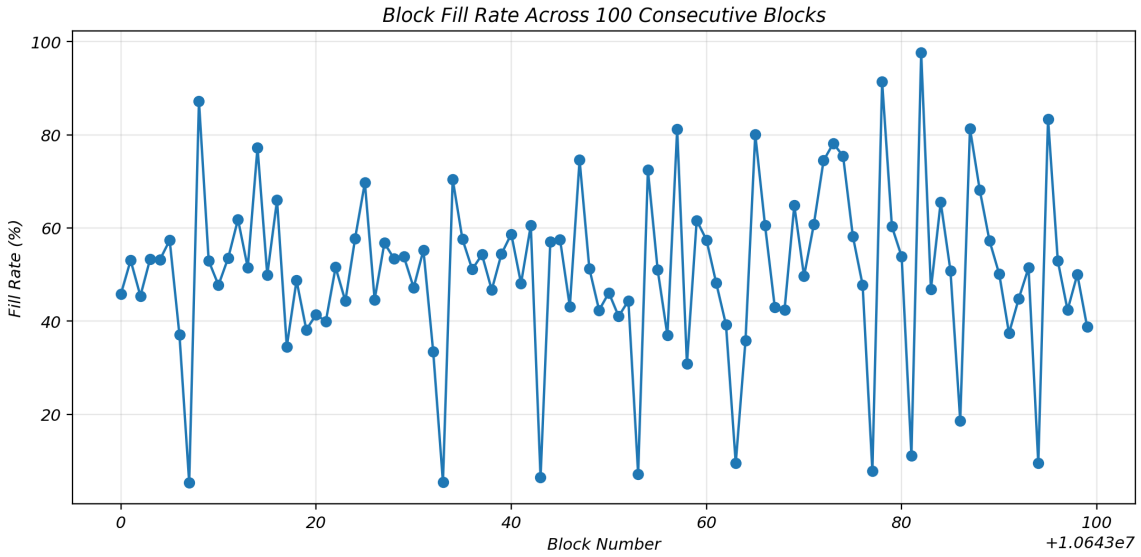

图 7: 100 个连续区块的填充率变化图

8.8 扩展样本下的区块活跃度特征

由图5 可见，在 100 个连续区块范围内，区块交易数量整体围绕某一中等水平上下波动，而不是长期停留在极低或极高状态。这表明在更长的连续区间内，Sepolia 网络总体保持持续运行，区块活跃度具有连续性与可观测性。与此同时，局部区块之间仍可出现明显高低差异，说明链上交易活跃程度在短时间尺度上依然会有较快起伏。因此，从扩展样本角度看，网络整体活动是稳定存在的，但局部波动仍然明显。

8.9 扩展样本下的费用变化趋势

由图6 可见，在 100 个连续区块范围内，`Base Fee` 并未出现无界增长或急剧坍塌，而是围绕一个较低区间持续波动。这说明在更长的观测窗口中，当前网络并未表现出严重拥堵状态，费用机制总体运行平稳。与 20 个连续区块样本得到的结论一致，`Base Fee` 的变化更适合被理解为短时波动中的动态调整，而不是单向剧烈变化。

8.10 扩展样本下的资源使用情况

由图7 可见，在 100 个连续区块范围内，区块填充率存在较明显的上下波动，说明不同区块的资源消耗程度并不一致。有些区块填充率较高，表明资源利用较充分；有些区块则明显偏低，说明该时刻链上负载较轻。与 20 个区块样本相比，100 个区块样本更充分地说明了这种波动并非孤立现象，而是在连续运行过程中持续存在的常态特征。

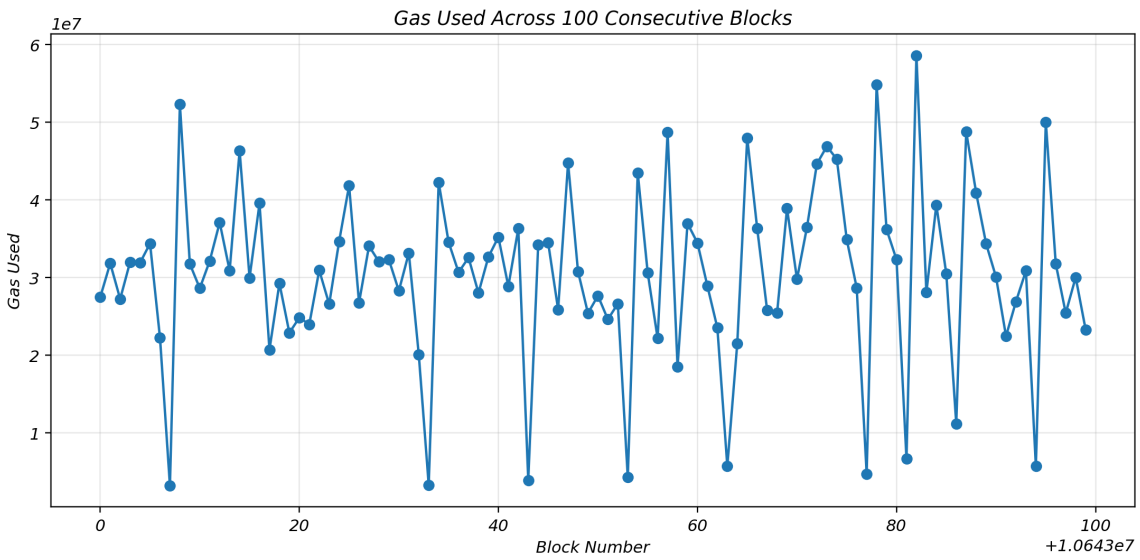

图 8: 100 个连续区块的 Gas Used 变化图

由图8 可进一步看出，`Gas Used` 的变化趋势与填充率变化整体一致。也就是说，区块资源使用水平在扩展样本中同样表现出显著波动，这进一步验证了链上资源需求在短时间内并不恒定，而是随区块活动情况不断变化。

8.11 扩展样本下费用与资源使用的关系

为了进一步观察费用变化与资源使用之间的关联，实验对 100 个连续区块绘制了 `Base Fee` 与填充率的散点图，如图9 所示。

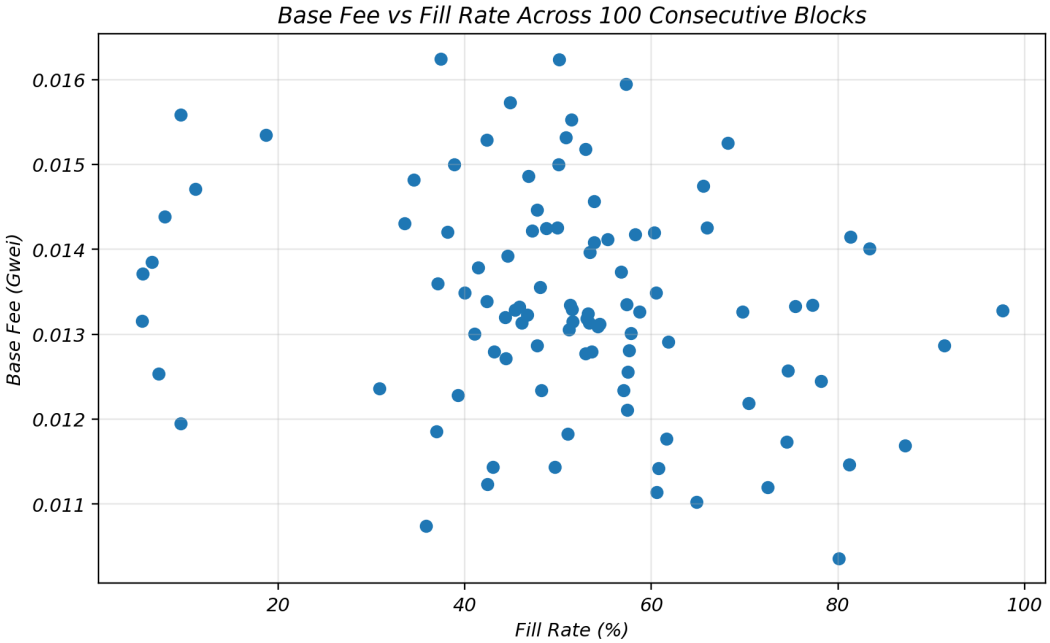

图 9: 100 个连续区块中 Base Fee 与填充率的关系散点图

从图9 可以看出，Base Fee 与区块填充率之间并不是严格线性关系，这与 EIP-1559 机制的实际运行逻辑相符。基础费用并不是单独由当前区块瞬时决定，而是会受到前序区块资源使用情况影响，因此在散点分布上表现为一定相关性，但又不是简单的一一对应关系。这一结果与前文基于 20 个区块样本得到的“存在逻辑关联但具有短时滞后性”的判断保持一致。

8.12 对该时间段内链上运行状态的总体判断总结

综合 10 个区块、20 个区块以及 100 个区块扩展样本的结果，本实验所选时间段内的 Sepolia 网络整体运行稳定，但存在显著的局部波动。区块交易数量、Gas Used 和填充率均在连续区块之间出现明显起伏，说明链上活动强度与资源使用程度在短时间内会发生快速变化；但从更长的 100 个区块样本来看，这些波动总体仍围绕一个有限区间展开，没有表现出持续恶化的拥堵趋势。与此同时，Base Fee 在较长样本中保持低位波动，并在高资源使用区块之后出现阶段性上升，说明费用机制能够持续对网络资源压力作出自适应调节。

A 附录A：Python 数据抓取源代码

以下给出实验中用于抓取 Sepolia 区块数据并生成 Excel 表格的 Python 源代码。考虑到 listings 环境中中文显示兼容性问题，附录中的代码不包含中文注释。

```
import re
import time
from typing import Dict, List, Optional
import pandas as pd
import requests
from bs4 import BeautifulSoup

BASE_URL = "https://sepolia.etherscan.io/block/{}"
START_BLOCK = 10643000
END_BLOCK = 10643019
OUTPUT_FILE = "sepolia_blocks_10643000_10643019_fixed.xlsx"

HEADERS = {
    "User-Agent": (
        "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/124.0.0.0 Safari/537.36"
    )
}
```

```
def fetch_html(url: str, timeout: int = 20) -> str:
    resp = requests.get(url, headers=HEADERS, timeout=timeout)
    resp.raise_for_status()
    return resp.text

def clean_text(text: Optional[str]) -> str:
    if not text:
        return ""
    text = text.replace("\xa0", " ")
    text = re.sub(r"\s*\.\s*", ".", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text

def extract_first(pattern: str, text: str) -> Optional[str]:
    if not text:
        return None
    m = re.search(pattern, text, re.I | re.S)
    return m.group(1).strip() if m else None

def to_float_safe(x: Optional[str]) -> Optional[float]:
    if x is None or x == "":
        return None
    try:
        return float(x.replace(",", ""))
    except Exception:
        return None

def parse_field_rows(soup: BeautifulSoup) -> Dict[str, str]:
    fields: Dict[str, str] = {}
    label_nodes = soup.select("div.text-dt")
    for label_node in label_nodes:
        label = clean_text(label_node.get_text(" ", strip=True))
        if not label:
            continue
        row = label_node.find_parent("div", class_=lambda x: x and "row" in x.
    split())
        if not row:
            continue
        value_node = row.find("div", class_=lambda x: x and "col-md-9" in x.
    split())
        if not value_node:
```

```
            continue
        value = clean_text(value_node.get_text(" ", strip=True))
        fields[label] = value
    return fields

def parse_block_page(html: str, url: str) -> Dict[str, Optional[str]]:
    soup = BeautifulSoup(html, "html.parser")
    fields = parse_field_rows(soup)
    tx_node = soup.select_one("#ContentPlaceHolder1_div_tx_fieldvalue")
    tx_text = clean_text(tx_node.get_text(" ", strip=True)) if tx_node else
    fields.get("Transactions:", "")
    wd_node = soup.select_one("#ContentPlaceHolder1_div_withdrawal_fieldvalue")
    wd_text = clean_text(wd_node.get_text(" ", strip=True)) if wd_node else
    fields.get("Withdrawals:", "")
    size_text = fields.get("Size:", "")
    block_height_text = fields.get("Block Height:", "")
    timestamp_text = fields.get("Timestamp:", "")
    fee_recipient_text = fields.get("Fee Recipient:", "")
    block_reward_text = fields.get("Block Reward:", "")
    gas_used_text = fields.get("Gas Used:", "")
    gas_limit_text = fields.get("Gas Limit:", "")
    base_fee_text = fields.get("Base Fee Per Gas:", "")
    block_number = extract_first(r"(\d+)", block_height_text)
    timestamp_utc = extract_first(r"\(([^()]*)\)", timestamp_text)
    timestamp_raw = timestamp_text if timestamp_text else None
    tx_count = extract_first(r"(\d+)\s+transactions", tx_text)
    withdrawals = extract_first(r"(\d+)\s+withdrawals", wd_text)
    fee_recipient = extract_first(r"(0x[a-fA-F0-9]{40})", fee_recipient_text)
    gas_used = extract_first(r"([\d,]+)", gas_used_text)
    gas_used_pct = extract_first(r"\(([\d.]+)%\)", gas_used_text)
    gas_limit = extract_first(r"([\d,]+)", gas_limit_text)
    base_fee_eth = extract_first(r"([\d.]+)\s*ETH", base_fee_text)
    base_fee_gwei = extract_first(r"\(([\d.]+)\s*Gwei\)", base_fee_text)
    block_reward_eth = extract_first(r"([\d.]+)\s*ETH", block_reward_text)
    size_bytes = extract_first(r"([\d,]+)\s*bytes", size_text)
    return {
        "block_number": block_number,
        "timestamp_raw": timestamp_raw,
        "timestamp_utc": timestamp_utc,
        "tx_count": tx_count,
        "withdrawals": withdrawals,
```

```
        "fee_recipient": fee_recipient,
        "gas_used": gas_used,
        "gas_used_pct": gas_used_pct,
        "gas_limit": gas_limit,
        "base_fee_gwei": base_fee_gwei,
        "base_fee_eth": base_fee_eth,
        "block_reward_eth": block_reward_eth,
        "size_bytes": size_bytes,
        "url": url,
    }

def crawl_blocks(start_block: int, end_block: int, sleep_sec: float = 1.2) ->
    pd.DataFrame:
    rows: List[Dict[str, Optional[str]]] = []
    for block_num in range(start_block, end_block + 1):
        url = BASE_URL.format(block_num)
        ok = False
        last_err = None
        for _ in range(3):
            try:
                html = fetch_html(url)
                row = parse_block_page(html, url)
                rows.append(row)
                ok = True
                break
            except Exception as e:
                last_err = e
                time.sleep(2)
        if not ok:
            rows.append({
                "block_number": str(block_num),
                "timestamp_raw": None,
                "timestamp_utc": None,
                "tx_count": None,
                "withdrawals": None,
                "fee_recipient": None,
                "gas_used": None,
                "gas_used_pct": None,
                "gas_limit": None,
                "base_fee_gwei": None,
                "base_fee_eth": None,
```

```
                "block_reward_eth": None,
                "size_bytes": None,
                "url": url,
            })
        time.sleep(sleep_sec)
    df = pd.DataFrame(rows)
    df["gas_used_num"] = df["gas_used"].apply(to_float_safe)
    df["gas_limit_num"] = df["gas_limit"].apply(to_float_safe)
    df["tx_count_num"] = df["tx_count"].apply(to_float_safe)
    df["base_fee_gwei_num"] = df["base_fee_gwei"].apply(to_float_safe)
    df["base_fee_eth_num"] = df["base_fee_eth"].apply(to_float_safe)
    df["block_reward_eth_num"] = df["block_reward_eth"].apply(to_float_safe)
    df["size_bytes_num"] = df["size_bytes"].apply(to_float_safe)
    df["fill_rate"] = df["gas_used_num"] / df["gas_limit_num"]
    df["fill_rate_pct"] = df["fill_rate"] * 100
    return df

def save_to_excel(df: pd.DataFrame, filename: str) -> None:
    with pd.ExcelWriter(filename, engine="openpyxl") as writer:
        df.to_excel(writer, sheet_name="all_blocks_raw", index=False)
        task1_cols = [
            "block_number",
            "timestamp_utc",
            "tx_count",
            "fee_recipient",
            "gas_used",
            "gas_limit",
            "fill_rate_pct",
            "url",
        ]
        df.iloc[:10][task1_cols].to_excel(writer, sheet_name="task1_10_blocks",
    index=False)
        task2_cols = [
            "block_number",
            "base_fee_gwei",
            "gas_used",
            "gas_limit",
            "fill_rate_pct",
            "url",
        ]
        df[task2_cols].to_excel(writer, sheet_name="task2_20_blocks", index=
```

```
    False)

if __name__ == "__main__":
    df = crawl_blocks(START_BLOCK, END_BLOCK, sleep_sec=1.2)
    save_to_excel(df, OUTPUT_FILE)
```