Amir Mohammad Khedri 4023613024

Sadra Khaleghi 4023613022

English Version

Phase 3 Report: Implementation and Simulation of Routing Protocols

1. Project Objective The primary goal of this phase was to transition from abstract network concepts to a practical, observable software simulation. We aimed to simulate Layer 3 routing mechanics—specifically OSPF and Dijkstra's algorithm—without relying on OS-level sockets or IP stacks. The objective was to visualize how independent routers exchange LSA messages to reach a consistent topological view and how a routing table is derived to forward packets dynamically.

2. Step-by-Step Implementation Summary Our development process followed a logical evolution:

1. **Skeleton & Topology:** We first defined the Router and Link classes to model the network graph and initialized a topology with weighted edges.
2. **LSA & Flooding:** We designed the LSA structure (Origin ID, Sequence Number, Neighbors) and implemented the flooding logic to propagate these messages across the network.
3. **Dijkstra & Routing Tables:** We implemented Dijkstra’s algorithm to calculate the Shortest Path Tree (SPT) from the perspective of each router and extracted the Next-Hop for every destination to populate the Routing Table.
4. **Packet Forwarding:** Finally, we implemented a simulation loop to forward a test packet hop-by-hop using the generated tables.
5. **Bonus Features:** We expanded the system to support Distance-Vector (RIP), BGP Path-Vector, and OSPF Multi-Area logic.

3. Key Design Decisions

- **Graph Data Structure:** We utilized a Dictionary of Dictionaries (Adjacency Map) for the Link-State Database (LSDB). This was chosen over an Adjacency Matrix because network graphs are typically sparse, and dictionaries provide O(1) lookup time for neighbors, significantly optimizing Dijkstra's execution.
- **LSA Modeling:** We included a Sequence Number in every LSA. This is critical for the flooding logic to distinguish new information from old or duplicate packets, preventing infinite propagation loops.

- **Modular Architecture:** We separated the core logic (network_logic.py) from the visualization (main_gui.py). This ensures that the routing protocol logic remains pure and testable without dependency on the UI library.
- **OSPF Multi-Area Strategy:** Instead of running separate SPF instances for every area, we used a "Virtual Summary Injection" method. Area Border Routers (ABRs) calculate costs and inject "Summary LSAs" into adjacent areas, allowing routers to see inter-area destinations without needing the full topology of the other area.

4. Key Modules Description

- **_dijkstra(graph, start):** This module is the core computational engine. It takes the global graph and the source router ID as input. Using a Min-Heap (heapq), it computes the minimum cost to all nodes. The output includes a parents dictionary, which is essential for backtracking the path to determine the precise Next-Hop for the routing table.
- **_flood(queue, logs):** This module manages the propagation of LSAs. It enforces the Sequence Number check to discard duplicates. Crucially, it implements the "Area Filtering" logic, ensuring that Type-1 LSAs do not cross area boundaries, while allowing Summary LSAs from ABRs to pass, maintaining the hierarchy required by OSPF.

Persian Version

امیرمحمد خدری 4023613024

محمد صدرا خالقی 4023613022

گزارش فنی فاز سوم: طراحی و پیاده‌سازی شبیه‌ساز پروتکل‌های مسیریابی (Layer-3 Routing Simulator)

۱. مقدمه و تبیین اهداف (Project Objectives)

هدف بنیادین در این فاز، گذار از مفاهیم انتزاعی لایه شبکه به یک پیاده‌سازی عملیاتی و ملموس بود. رویکرد ما ایجاد یک محیط شبیه‌سازی ایزوله و خالص نرم‌افزاری بود تا فارغ از پیچیدگی‌های سطح سیستم‌عامل (نظیر سوکت‌های خام یا پشته IP)، مکانیزم‌های دقیق مسیریابی را مدل‌سازی کنیم. تمرکز اصلی بر درک عمیق رفتار پروتکل OSPF، چگونگی همگرایی روترها از طریق تبادل پیام‌های LSA و نهایتاً تبدیل توپولوژی گراف به جدول مسیریابی (Routing Table) جهت هدایت هوشمند بسته‌های داده است.

---

۲. نقشه راه و گام‌های پیاده‌سازی (Implementation Roadmap)

فرآیند توسعه این سامانه در یک ساختار تکاملی و طی پنج گام اصلی به انجام رسید:

- **گام اول: معماری پایه و مدل‌سازی توپولوژی**

ابتدا زیرساخت شیءگرای پروژه با تعریف کلاس‌های Router و Link بنا نهاده شد. این انتزاع به ما اجازه داد تا توپولوژی‌های پیچیده (شامل گراف‌های وزن‌دار و دارای حلقه) را به سادگی در حافظه مدل‌سازی کنیم.

- **گام دوم: هسته پروتکل OSPF و مکانیزم Flooding**

ساختار بسته LSA (حاوی شناسه مبدأ، شماره توالی و لیست همسایگان) طراحی شد. سپس موتور Flooding با منطق دقیق کنترل انتشار پیاده‌سازی گردید تا اطمینان حاصل شود تمام روترها به یک دید یکسان از شبکه می‌رسند.

- **گام سوم: پایگاه داده وضعیت لینک (LSDB) و موتور محاسباتی**

کلاس LinkStateDB جهت نگهداری نقشه شبکه در هر روتر ایجاد شد. سپس الگوریتم Dijkstra به عنوان موتور محاسباتی روی این پایگاه داده پیاده‌سازی شد تا درخت کوتاه‌ترین مسیر (SPT) برای هر گره استخراج گردد.

- **گام چهارم: استخراج جدول مسیریابی و لایه ارسال (Forwarding)**

با پیمایش معکوس (Backtracking) روی خروجی الگوریتم دایجسترا، پارامتر Next-Hop برای تمامی مقاصد محاسبه و در جدول مسیریابی درج شد. صحت عملکرد این جداول با ارسال یک بسته آزمایشی و ردیابی گام‌به‌گام آن (Packet Tracing) تأیید گردید.

- **گام پنجم: توسعه قابلیت‌های پیشرفته (Bonus Features)**

در فاز نهایی، معماری سیستم برای پشتیبانی از پروتکل‌های RIP (رویکرد Distance-Vector)، BGP (رویکرد Path-Vector) و قابلیت OSPF Multi-Area توسعه یافت.

---

۳. تصمیمات کلیدی در طراحی و معماری (Key Architectural Decisions)

برای تضمین کارایی، مقیاس‌پذیری و خوانایی کد، تصمیمات فنی زیر اتخاذ شد:

- **انتخاب ساختار داده گراف (Adjacency Map):**

برای پیاده‌سازی LSDB، از ساختار Dictionary of Dictionaries استفاده شد.

  - ***چرایی:*** **این ساختار در گراف‌های شبکه که معمولاً خلوت (Sparse) هستند، نسبت به ماتریس مجاورت حافظه کمتری مصرف می‌کند و دسترسی $O(1)$ به یال‌ها را برای الگوریتم دایجسترا فراهم می‌سازد.**

- **مدل‌سازی LSA با شماره توالی (Sequence Number):**

فیلد seq_num به بسته‌های LSA افزوده شد.

  - ***چرایی:*** **این فیلد حیاتی‌ترین مکانیزم برای جلوگیری از Loop و Broadcast Storm در فرآیند Flooding است، زیرا به روترها اجازه می‌دهد اطلاعات جدید را از داده‌های تکراری یا منسوخ تمیز دهند.**

- **تفکیک منطق از نمایش (MVC Pattern):**

هسته محاسباتی (network_logic.py) کاملاً از رابط کاربری گرافیکی (main_gui.py) جدا شد.

- ***چرایی:*** **این جداسازی (Decoupling) موجب شد تا منطق پروتکل‌ها مستقل از کتابخانه گرافیکی قابل تست و توسعه باشد و کد پروژه ساختاری حرفه‌ای و ماژولار پیدا کند.**

- **استراتژی پیاده‌سازی Multi-Area:**

به جای اجرای چندین نمونه مجزای SPF، از تکنیک "تزریق نود مجازی" (Virtual Summary Injection) استفاده شد.

- ***چرایی:*** **در این روش، روترهای مرزی (ABR) هزینه رسیدن به مقاصد ناحیه دیگر را محاسبه کرده و آن‌ها را در قالب یک Summary LSA به ناحیه مجاور تزریق می‌کنند. این رویکرد ضمن کاهش پیچیدگی پیاده‌سازی، رفتار سلسله‌مراتبی OSPF را به دقت شبیه‌سازی می‌کند.**

---

۴. تشریح ماژول‌های حیاتی (Core Modules Description)

- **ماژول محاسباتی _dijkstra(graph, start):**

این تابع قلب تپنده مسیریابی Link-State است.

- ***ورودی:*** **گراف کامل شبکه (مستخرج از LSDB) و شناسه روتر مبدأ.**
- ***مکانیزم:*** **با بهره‌گیری از صف اولویت (Min-Heap)، نودها بر اساس کمترین هزینه پیمایش می‌شوند.**
- ***خروجی:*** **دیکشنری parents که مسیر بازگشت بهینه از هر مقصد به مبدأ را مشخص می‌کند و مبنای ساخت جدول مسیریابی است.**

- **ماژول مدیریت انتشار _flood(queue, logs):**

این تابع وظیفه همگام‌سازی پایگاه داده روترها را بر عهده دارد.

- ***نقش ویژه:*** **علاوه بر انتشار پیام‌ها، منطق Area Filtering در اینجا اعمال می‌شود. این ماژول هوشمندانه مانع از نشت اطلاعات جزئی (Type-1 LSAs) به نواحی دیگر شده و تنها به اطلاعات خلاصه شده (Summary LSAs) اجازه عبور از مرزهای Area را می‌دهد، که این امر اساس مقیاس‌پذیری در پروتکل OSPF است.**