Network Research Project Proposal

-- CS 397/497 Selected Topics in Computer Networks, Spring 2022

Yunming Xiao
(yunming.xiao@u.northwestern.edu)

Data Center Coflow Scheduling Design

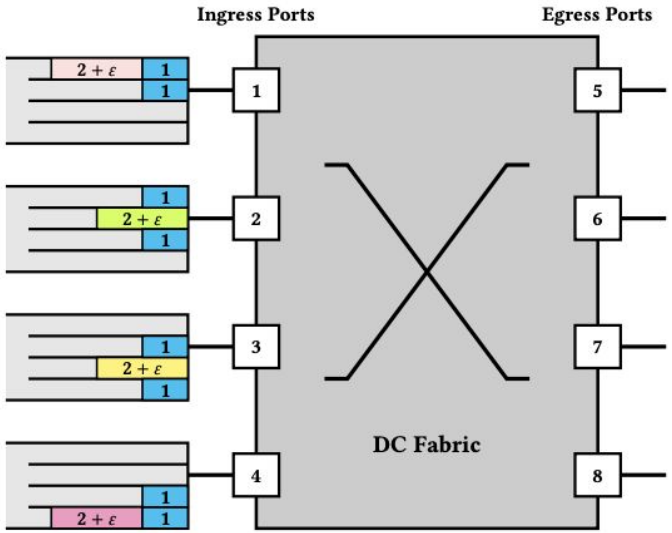
- Background
- Methodology
- Expected Results

Background - Coflow scheduling

Previous study proposed various centralized or decentralized solutions to manage the coflows – a set of subflows that have the same objective, and the objective is achieved only when all subflows are completed.

Model: one-big-switch abstract.

Metrics: coflow completion time



Our Approach

Difficulties:

The coflow scheduling itself is an NP-hard problem.

It has also been proved that decentralized solution cannot be as good as centralized in certain scenarios.

Our Idea:

However, a centralized approach has a **high overhead** for collecting data and distribute the decisions. We instead adopt the decentralized approach based on a new transport-layer protocol, which has an important properties: **without the need to coordinate** with other flows on the same link, the flows are assigned with priorities and can adjust the bandwidth considering the priorities of the other flows.

Overall, we want to design and evaluate a decentralized policy that would minimize the coflow completion time based on the new transport layer protocol.

Expected Results

- Complete the simulations based on the one-big-switch abstract
- Evaluate the proposed decentralized policy. Expectation is that it would generate outcomes that are close to the centralized solutions
- We further want to evaluate our policy on real-world environments and demonstrate its efficiency

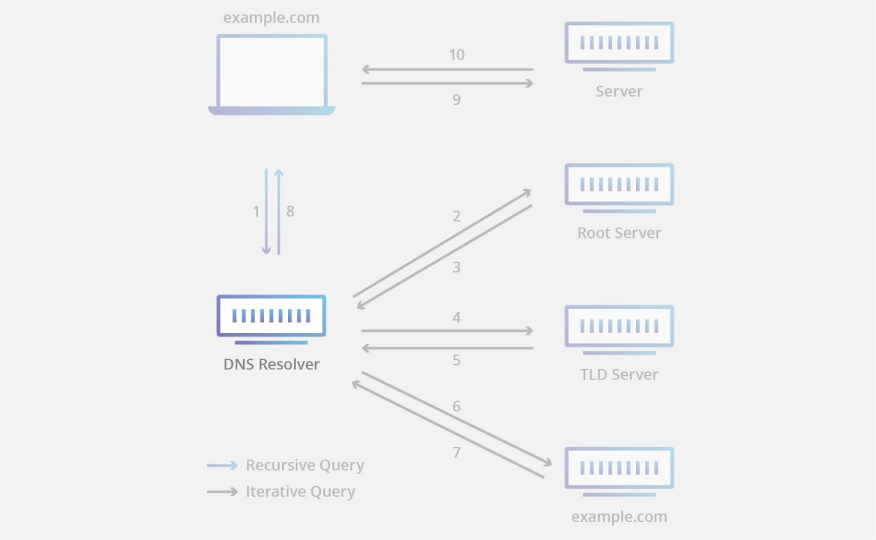
The Privacy Era For DNS

- Background - DNS and Privacy
- Background - Private Information Retrieval
- Methodology and Expected Results (1)
- Methodology and Expected Results (2)

Background - DNS and Privacy

Domain Name System (DNS) is one of the foundation of today's Internet. Before sending requests to any website, the users need to send DNS requests to DNS resolvers, which are hosted either by ISPs or public providers like Google, to resolve the IP address of the domain name.

- DNS with TCP or UDP are in plaintext. This allows anyone in the WAN to observe the activities of the users.
- Later, DNS over TLS (DoT) or HTTPS (DoH) adopts end-to-end encryption to protect the user privacy from any other in the WAN. However, this still allows the DNS resolver to know all the activities of the users.
- A further step is made by Oblivious-DNS (ODNS), which sets up two non-colluding DNS resolvers, where one resolver does not know what the user has requested and the other resolver does not know who sends the request.



Background - Private Information Retrieval

Private information retrieval refers to a series of protocols which allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved. This is achieved by cryptography primitives.

To put it simple, the easiest way is to ask the database to send all items to the user. This apparently brings too much overhead. Instead, PIR is a smart way to do similar things without the need to exchange the whole database.

Complexity: O(n^0.5)

Methodology and Expected Results (1)

Methodology

Our idea is simple: we apply PIR on DNS so that the DNS resolver will have no access to the user activities.

But it comes with a lot of questions:

- how do we know what to cache without knowing what is requested?
- is the performance a issue?
- ……

Therefore, we need to perform a measurement study to understand what are the potential limitation.

Expected Results

Most importantly, we want to understand the cache update policy, domain popularity, CDN impacts, and TTLs

Methodology and Expected Results (2)

Methodology

We build a efficient DNS based on PIR code repository.

Expected Results

Implement a practical DNS resolver with PIR.

Sen Lin
(sen.lin@u.northwestern.edu)

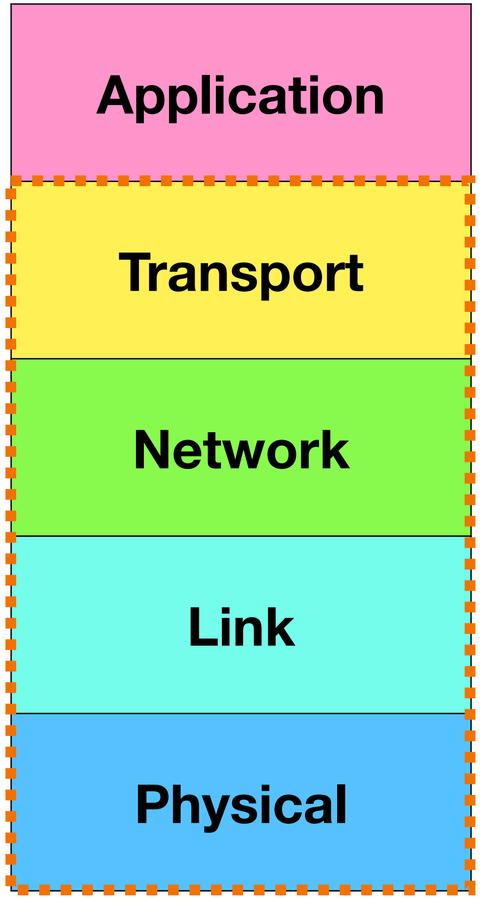
Comparative Analysis of Software Network Programming Techniques

A project proposal of CS 397/497 (Spring'22)

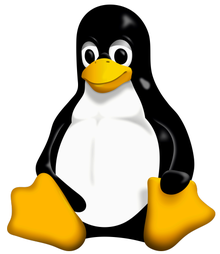
Sen Lin (sen.lin@u.northwestern.edu)

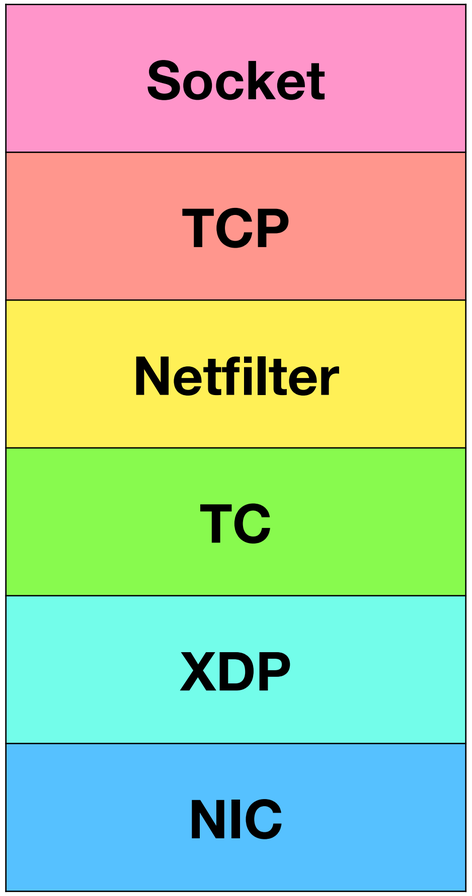
Description

- Network programming aims to operate low-level network packets which is different from socket programming.
  - e.g. a router/switch/load balancer
- We’re focusing on software techniques. Programs are expected to be running on any generic Linux machines.
- In this project, we’re going to explore some popular techniques among the community to develop networks and compare their **performance** and **flexibility** in real production.



A peak of potential techniques





Linux Networking Stack

Tentative plans

- Week 1-2: Get familiar with related concepts
- Week 3: Set up environments
- Week 4: Able to run “hello, world” programs with different techniques
- Week 5: Prepare the midterm presentation
- Week 6-7: Develop a L3-switch (with scheduler) using each technique
- Week 8: Benchmark and collect data
- Week 10: Wrap-up: documents & presentation

A performance-oriented review over QUIC

A project proposal of CS 397/497 (Spring'22)

Sen Lin (sen.lin@u.northwestern.edu)

Description

- QUIC is a new transport layer protocol initially designed by Google and standardized by IETF (RFC 9000) which is the bedrock of the next-generation HTTP/3
  - QUIC is integrated with multiple modern network features (0-RTT, multiplexing, connection migration, etc)
  - Easy-to-deploy as it’s built upon UDP
- However, QUIC implementations are infamous for their poor performance (especially for long flows)
  - **We’re going to explore the reasons and find optimization opportunities**

Potential causes and workarounds

- Kernel's default (UDP) buffer size limitation
- Natural difference between user-space and kernel-space.
- No hardware off-loading
- QUIC libraries need to traverse the whole kernel network stack redundantly. Some serial accesses may even worsen the performance.
- QUIC’s benefits (such as mitigation of handshakes) fade for large stream transmission in comparison with TCP

Increase the buffer

Process QUIC packets before entering the kernel

Can multiplexing or other features compensate?

Tentative plans

- Week 1-2: Get familiar with QUIC and its implementations (quic-go/quiche/etc)
- Week 3: Set up environments
- Week 4: Able to run “hello, world” programs with different techniques
- Week 5: Prepare the midterm presentation
- Week 6-7: Benchmark different implementations and use flame graphs and/or other techniques to find the bottlenecks
- Week 8-9: Try some optimizations
- Week 10: Wrap-up: documents & presentation

Yanzhi Li
(YanzhiLi2026@u.northwestern.edu)

Flash Player:Now & Future

Yanzhi Li

Flash Player

- One of most popular software and plugins before
- Came to the end of life in 2021
- Major Browsers no longer support(However, there are emulators)
- Some websites still rely on Flash
- Flash is still officially supported in mainland China

Try to find out:

- Where are flash still being used?
- Does it cause any problems?
- Can we completely replace them?

What to do?

- Identify where are flash still being used. (Starting from emulators)
- Investigate the reasons why they are still being used
- Investigate flash alternatives

Edge Computing on Low-end Devices

Yanzhi Li

Edge Computing

- Push computation towards the edge of the network, exploiting smart objects, mobile phones, or network gateways to perform tasks.
- Better Response Time and Transfer rates.

Try to find out:

- How well it works on low-end device?
- Compare performance of local computation and overhead of last node delivery.
- How far can we push the edge nodes?
- Future of edge computing on low-end devices.

What to do?

- Identify target applications enabled by edge computing
- Identify some low-end devices
- Analyze latency, bandwidth or other metrics of these applications on these devices with or without using edge computing