ciena

Experience. Outcomes.

How to Build a Router

Prepared by Ciena, 2025

Introduction

Building a router from scratch may sound intimidating! In this presentation, we will de-mystify routers and break down the ins and outs of their design.

What is a router?

- Let’s start simple.
- A router is a box - a box that takes an input, processes it, and produces some

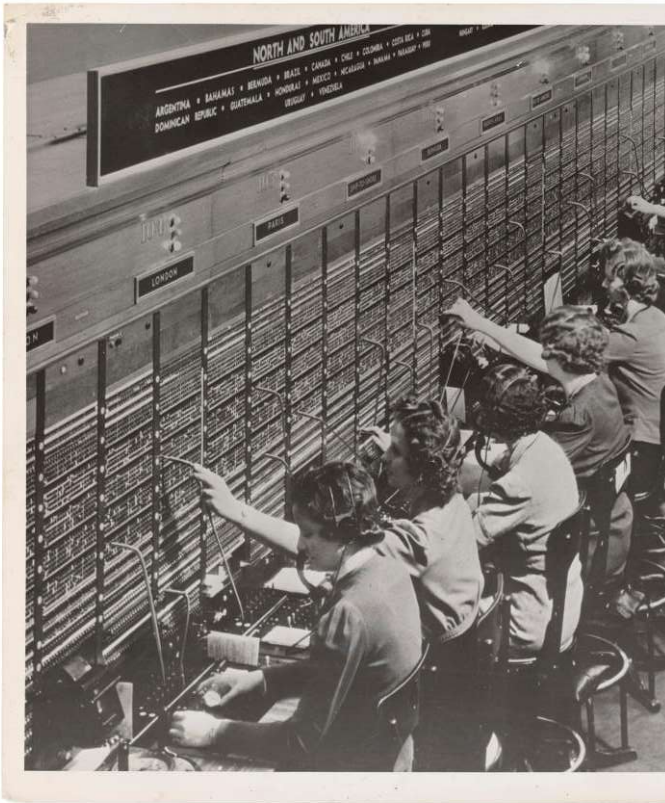
- Mathematically speaking:

Function(input) = output

We’ll soon discuss different types of **input**, **output**, and the properties of this

The Telephone Switchboard

- What came before the router? The Telephone Switchboard!
- In the late 1800s, switchboard operators would connect calls by putting plugs into jacks on a manual switchboard.
- Routers are the new switchboard operator + switchboard.


NORTH AND SOUTH AMERICA
ARGENTINA • BAHAMAS • BERMUDA • BRAZIL • CANADA • CHILI • COLOMBIA • COSTA RICA • CUBA
DOMINICAN REPUBLIC • GUATEMALA • HONDURAS • MEXICO • NICARAGUA • PANAMA • PARAGUAY • PERU
URUGUAY • VENEZUELA
LONDON
PARIS

Source: https://culturexchange1.wordpress.com/2015/06/0
switchboard-the-story-of-a-revolutionary-instrument

The Telephone Switchboard

Visual Example:

User1 **Switchboard Operator**

1. |-----------> User1: Hello Operator, I want to call User2
2. *Switchboard Operator connects the ports of User1 and User2*
3. |--

User1 and User2 can now talk!

Back to Routers

*We can think of a router as a **stateful function**:*

Function(input) = output

We can categorize **input** into two types:

1. Data traffic (e.g., the call between User1 and User2)
2. Control traffic (e.g., traffic that mutates the state of the router)

A **Stateful Function** is one that remembers past events, influencing how it ha
future inputs. For example, the same data packet may take different paths de
on the router’s current state.

The Coding Challenge

You will be provided two files: **router.py** and **simulation.py**.

You will need to implement the logic of **router.py** to cover 5 Router Use Cases
about these next).

- Instead of processing real data packets, router.py will process integers to sim
  behaviour of a real router.

Setup Instructions:

1. Run simulation.py
2. Run router.py

simulation.py

Running **simulation.py** produces a text file called **StatefulHardware.txt**:

StatefulHardware.txt contains 8 -10 integers in the following format:

```
a, b, c, d <----- state values
p, q, r, s <----- control values
x, y       <----- signal values (optional)
```

Every 1 second, simulation.py reads the current state, control, and signal valu
calculates $f(a, b, c, d, p, q, r, s) = a^p \cdot b^q \cdot c^r \cdot d^s$.

Every 6 seconds, simulation.py modifies the current state of the router by ran
mutating a signal value in StatefulHardware.txt.

ciena
Experience. Outcomes.

The 5 Use Cases

There are 5 Router Use Cases you need to know for this challenge:

Case 1: Forwarding Data Traffic

Case 2: Handling Control Traffic

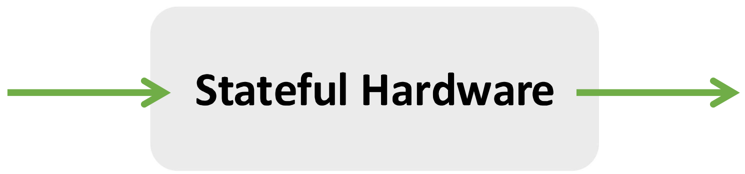
Case 3: Management Functionality

Case 4: Handling Cron Jobs

Case 5: Recovery & Documentation

Case 1: Forwarding Data Traff

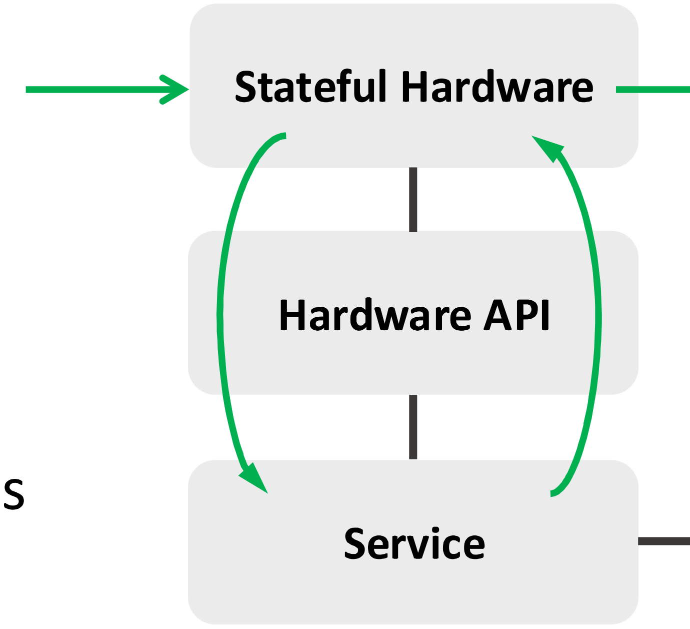
- **Stateful Hardware** forwards **data traffic**.



- *This use case has been already been implemented for you!*
- **simulation.py** simulates this behaviour by:
  1. Reading state, control, and signal values from **StatefulHardware.txt**,
  2. Processing the values $f(a, b, c, d, p, q, r, s) = a^p \cdot b^q \cdot c^r \cdot d^s$,
  3. Outputting the result.

Case 2: Handling Control Traff

- **Stateful Hardware** senses **control traffic**, then notifies **Service** through **Hardware API**.
- **Service** processes this traffic (with help from **Service Database**), then sends instructions back to **Stateful Hardware** through **Hardware API**.
- **Hardware** receives the instruction and changes its state (i.e., changes how it forwards data traffic).



Example: *User1 tells Switchboard Operator "I want to call User2." Switchboar looks at his table and sees the call from User1 comes from Port 1, and User2 i The Switchboard Operator connects Port1 and Port2 with a jack.*

Case 2 Implementation

Case 2: Handling Control Traffic

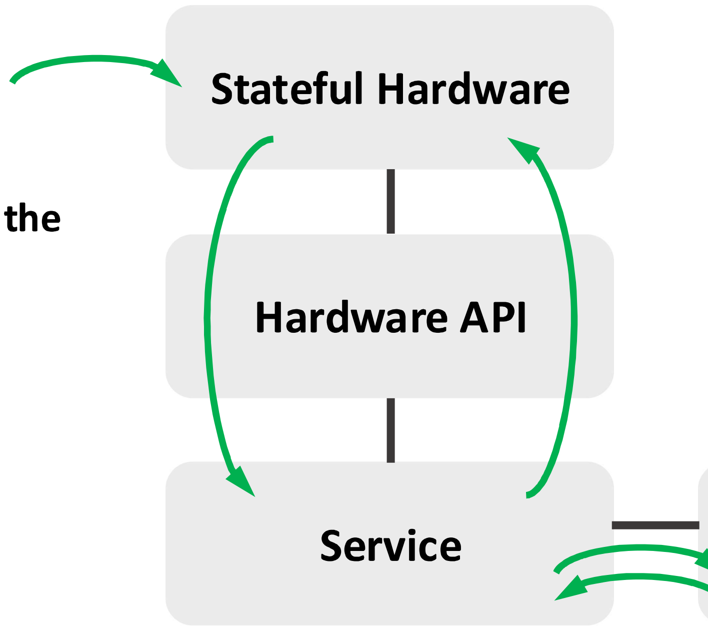
Recall: Format of StatefulHardware.txt:
a, b, c, d <--- *state values*
p, q, m, n <--- *control values*
x, y <--- *signal values*

Your task: Modify the control values using the signal values (e.g., set the *x'th* control value to *y*, where *x ϵ {1, 2, 3, 4}*)

Example:
1, 2, 3, 4
2, 3, 5, 7
1,6

Becomes

1, 2, 3, 4
6, 3, 5, 7
1,6



Case 3: Management Functiona

Scenario:

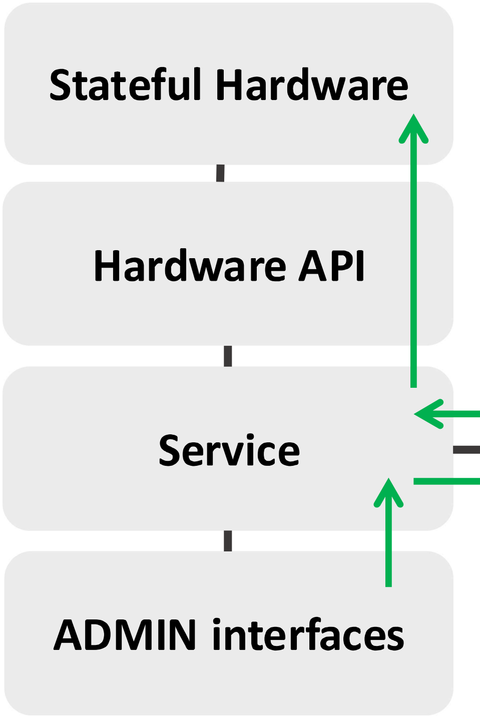
- User1 hasn't been paying their phone bills, and the manager wants to cut th
- We need to introduce an ADMIN interface (CLI) so that the manager can ma modifications to the switchboard.

Switchboard Example:

- Manager tells the Switchboard Operator: "I want to cut User1's service!"
- The Switchboard Operator sees User1 is mapped to Port 1. He unplugs Port though User1 is currently on a call with User2). Then, he disables Port 1, blo control and data traffic.

Case 3: Management Functiona

- The manager sends an ADMIN signal from **ADMIN Interface** to **Service**.
- The **Service** processes the ADMIN signal (with help from **Service Database**), then sends instructions to **Stateful Hardware** through **Hardware API**.
- **Stateful Hardware** changes its state (e.g., to block port 1)



Implementing Case 3

In router.py, **you must implement a CLI so that a manager can send the following command to modify the hardware's state values.**

The command should have the following format:

set j k

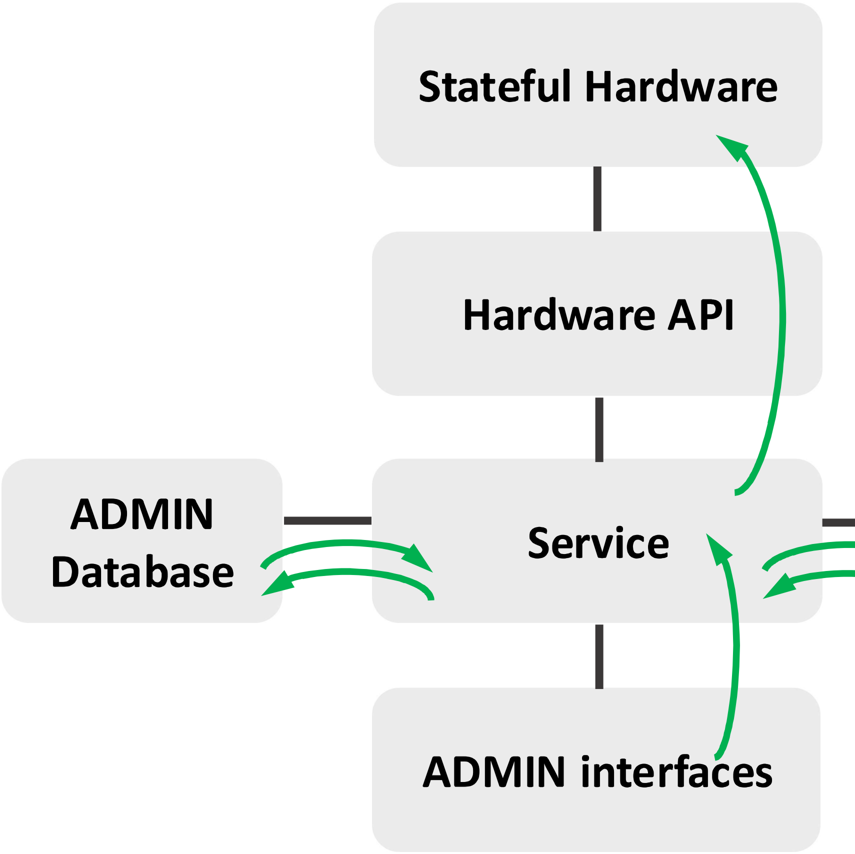
where j = index (1-indexed) of state value
where k = integer the j'th value is set to.

Given the following StatefulHardware.txt:

```
a, b, c, d
p, q, m, n
x, y
```

CLI command: set 1 2

```
2, b, c, d
p, q, m, n
x, y
```

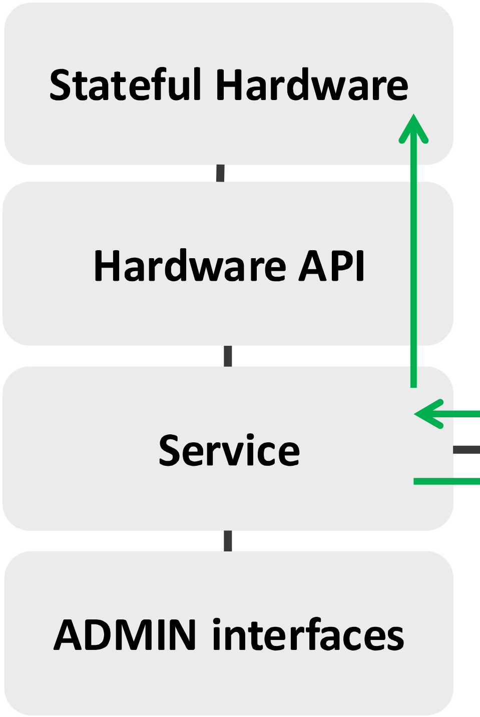


Case 4: Handling Cron Jobs

- To save money and electricity, we don’t want User1 and User2 to stay conne
long after they’ve stopped talking.
- We need a timing feature that will enable us to disconnect User1 and User2
period of silence (e.g., 10 minutes).
- A cron job is a program that schedules tasks at recurring intervals.
- Example: Every night at 2:00 AM, the router needs to perform a backup of it
configuration file and save it to a remote server.

Case 4: Handling Cron Jobs

- The **Service** has an internal clock it’ll use to schedule time-related cron jobs.
- The **Service** reads from (and sometimes writes to) the **Service Database**. It calculates the action it needs to perform, then sends instructions to **Stateful Hardware** through **Hardware API**.
- **Stateful Hardware** changes its state (e.g., performs the cron-job)



Implementing Case 4

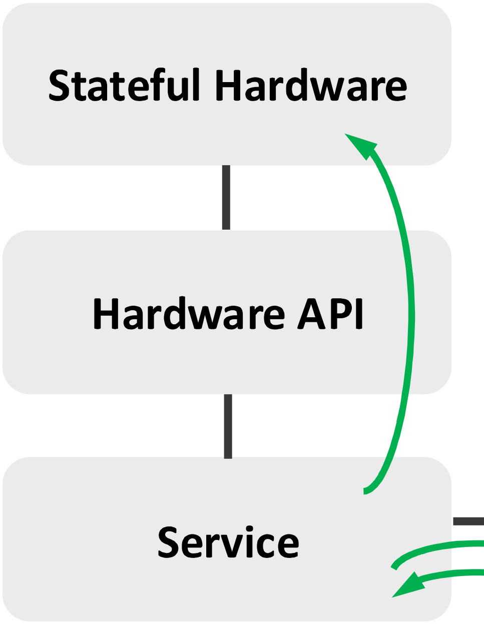
We will simulate time by incrementing a counter *t*.
t starts at 0 and increments by 1 at the start of each iteration

Here is your Cron Job: Whenever *t* is a multiple 10, swap the state values at indices 1 and 2 (1-indexed).

a, b, c, d
p, q, m, n

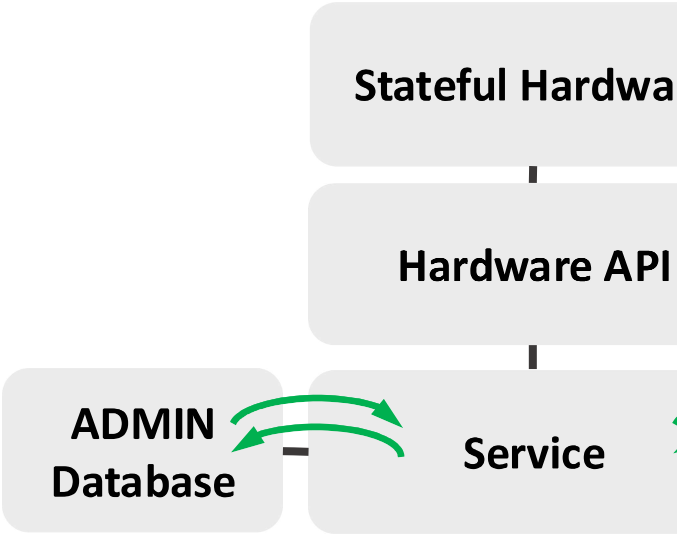
Into
b, a, c ,d
p, q, m, n

(i.e., swapped a and b)



Case 5: Recovery and Documenta

- Our router has crashed.
- After a new **Service** is online, it grabs information from **ADMIN Database**, which is the persistent storage that survived the crash.
- The **Service** updates the **Service Database** using the information from **ADMIN Database**. Then, it sends the configuration to **Stateful Hardware** through **Hardware API**
- **Stateful Hardware** will change its state (e.g., how it forwards data traffic)



Implementing Case 5

router.py contains an empty list called **history**. Use this to store a history of set commands (Use Case 3) and the Cron Jobs (Use Case 4).

Format for Use Case 3:

t set *x y*

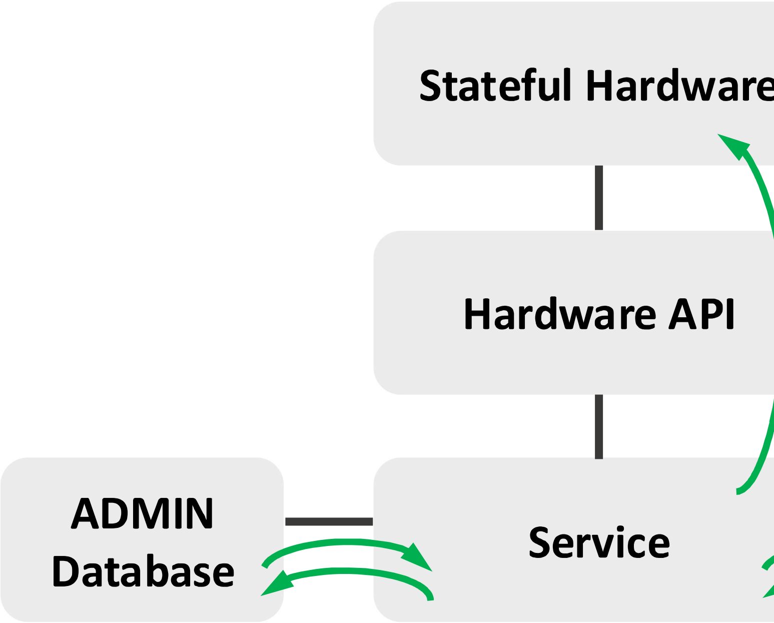
where t = time (while loop interval), x = index, y = value

Format for Use Case 4:

t swap *a b*

where t = time, a = state value at index 1, b = state value at index 2

For example:
['5 set 3 3', '8 set 0 2', '10 swap 3 5', '19 set 0 1', '20 swap 5 3', '30 swap 8 3', '40 swap 3 8', '50 swap 8 3', '60 swap 3 8']


Stateful Hardware
Hardware API
ADMIN
Database
Service