# Generating routing tables using a relational Database

Lana Abadie, CERN PhD student from University Pierre & Marie Curie (Paris VI) , Laboratoire SAMOVAR

RT2005 Conference

# Objectives & requirements

- Configure a network (routing tables, DHCP config file, DNS configuration …) composed of thousands of switches and networking components.

- Reduce the use of « Automatic» or « self-management » mode: deterministic behavior of any networking devices

- Keep track of all these routing tables, these config files to be able to recover everything in case of a network crash.

# Network configuration use a database

- **Database** : good and safe technology to store information about large systems

- Information stored in the database should be:
  - Complete to be able to start up the network after a crash
  - Scalable to support any extension or removals in the network topology
  - Easy to maintain : (minimize the data inserted by users)

- Develop packages to load and save information required to configure the network

# Gather and analyze use cases

- **2 kinds of use cases in the network configuration**
  - Generate config files (for the DHCP, DNS …)
  - Generate routing tables (for switches)

- **Focuses on generating routing tables**
  - tables stored in a **router** or some other networking device (**switch**) that keeps track of routes to particular network host interfaces
  - 2 type of routing tables IP or MAC
  - One  routing table for each switch in the network
  - Different types algorithms to determine the routes to take (default shortest path, i.e minimize devices crossed)

# Properties of a routing table

- An IP (resp. MAC) routing table consists of the following entries:
  - IP (MAC) address of the next hop
  - Port number to forward the data
  - IP (MAC) address of the host node (or destination)
  - Subnet Mask (VLAN prefix)
  - Path length

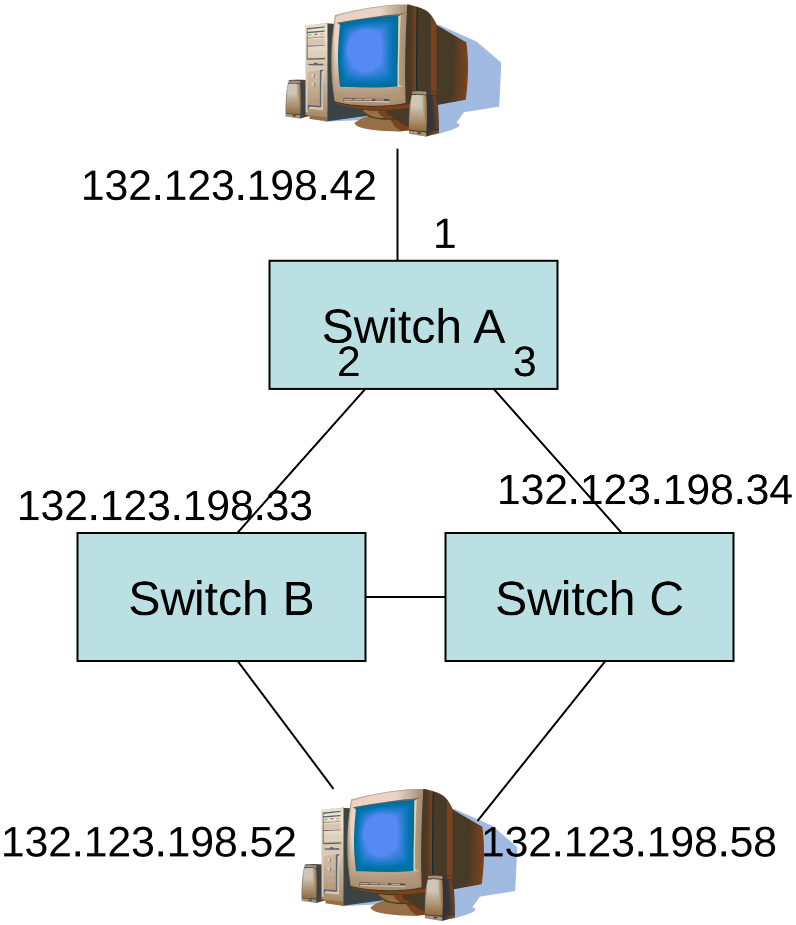
- No ambiguity on the port to forward data: one single route per **destination interface**

- No cyclic paths (  Data Acquisition, no redundant path)

# Routing table concept

## IP routing table for switch A

| Port nb | IP @ Next hop | IP @ destin. | Subnet mask | Path length |
|---|---|---|---|---|
| 1 | 132.123.198.42 | 132.123.198.42 | 255.255.255.0 | 1 |
| 2 | 132.123.198.33 | 132.123.198.52 | 255.255.255.0 | 2 |
| 3 | 132.123.198.34 | 132.123.198.58 | 255.255.255.0 | 2 |



# Network properties

**Node properties:**

- **Host** node : processes data (PCs…)
- **Switch** or intermediate node (switch, router…)
- **Interface** : a MAC and IP @ and a port number of a device

**Link properties:**

- **Start** and **end**
- **Orientation** : unidirectional or bidirectional
- **Type** : data, trigger…
- **Status** (functional or dead)

***Enough information to generate a routing table***

# Database table schema

2 kinds of data stored in the database tables

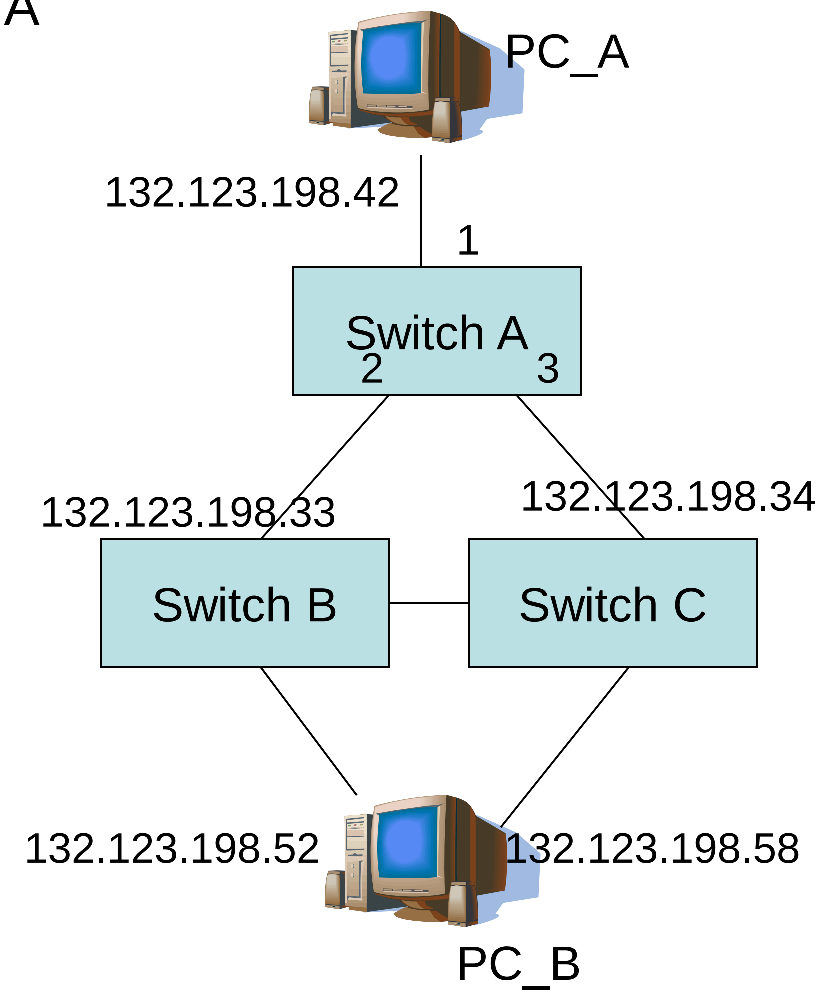
- **Data inserted** by the user (can’t be guessed)
  - – **List of the devices** of the network, by specifying if it’s a host node or not ( device  table)
  - – **All link properties** (connectivity table)
  - – **List of interfaces** (Ethernet & IP @, subnet mask, VLAN prefix, port number and device) (ip_ethernet table)

- **Data derived** from other information stored in the database:
  - – **Routing tables** (routing_table) : one database table per switch

# Ex: IP routing table of Switch A

- Find all paths < round_trip from switch A to PC_A (1 interface)  & to PC_B (2 interfaces)

| Node 1 | Node 2 | Node 3 | Node 4 | Node 5 | length |
|---|---|---|---|---|---|
| **Switch A** | Switch B | **PC_B** | | | 2 |
| **Switch A** | Switch B | Switch C | **PC_B** | | **3** |
| **Switch A** | Switch C | Switch B | **PC_B** | | **3** |
| **Switch A** | **PC_A** | | | | 1 |
| **Switch A** | Switch C | **PC_B** | | | 2 |



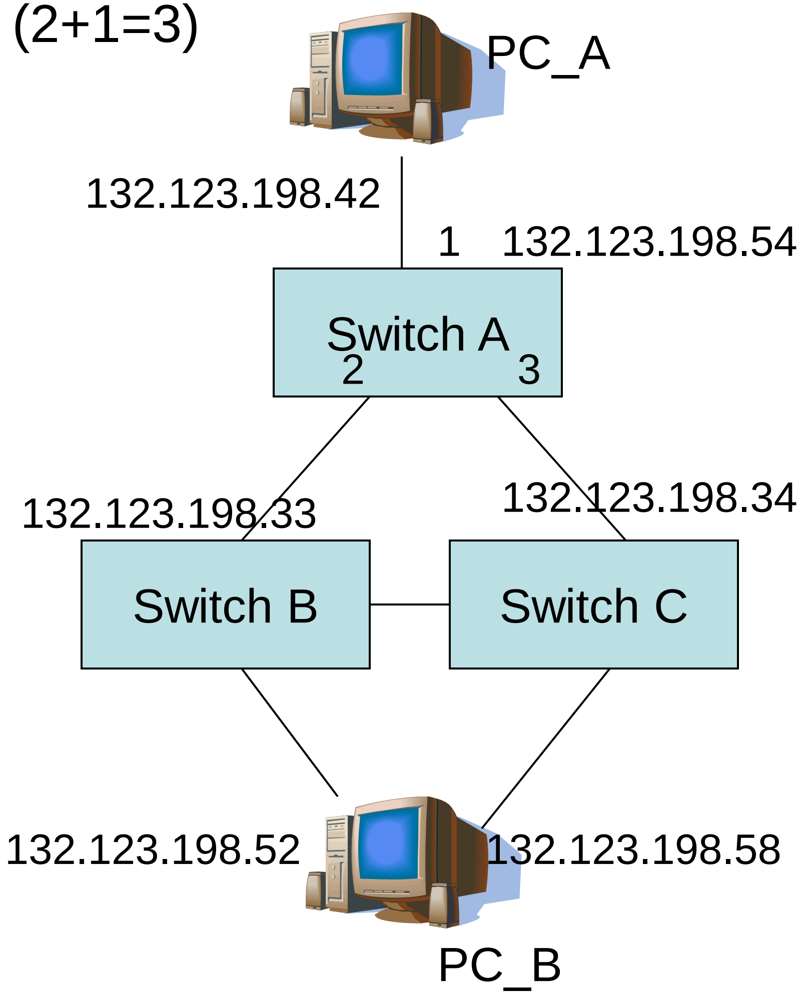
# Ex: IP routing table of Switch A

Select the shortest one per host interface (2+1=3)

| Node 1 | Node 2 | Node 3 | length |
|---|---|---|---|
| **Switch A** | **PC_A** | | 1 |
| **Switch A** | Switch C | **PC_B** | 2 |
| **Switch A** | Switch B | **PC_B** | 2 |

Convert the name to IP @

| Port nb | IP@ next hop | IP @ Destin. | length |
|---|---|---|---|
| 1 | 132.123. 198.42 | 132.123. 198.42 | 1 |
| 2 | 132.123. 198.33 | 132.123. 198.52 | 2 |
| 3 | 132.123. 198.34 | 132.123. 198.58 | 2 |



PC_A

132.123.198.42

1 132.123.198.54

Switch A

2

3

132.123.198.33

132.123.198.34

Switch B

Switch C

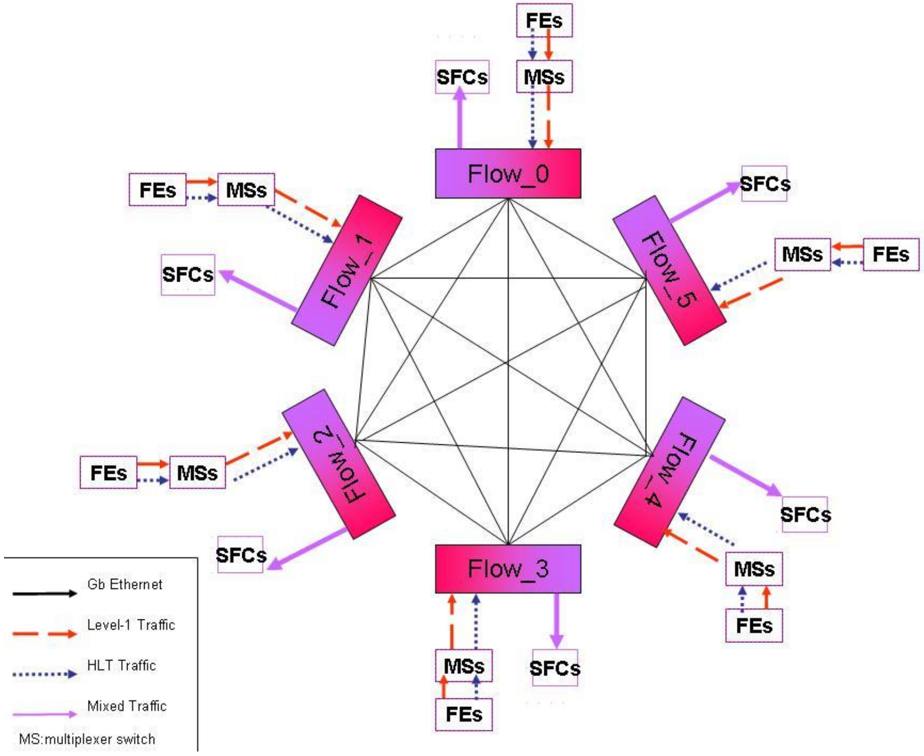
132.123.198.52

132.123.198.58

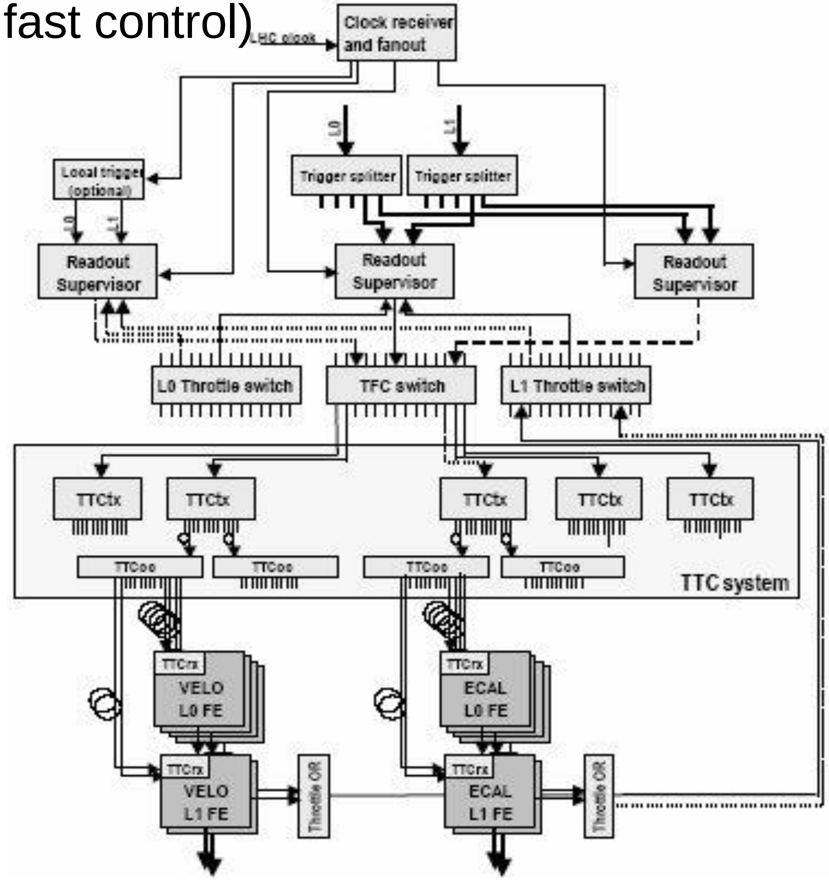
PC_B

# Topology used for tests

DAQ (data acquisition)



TFC (timing fast control)



2350 nodes and 4216 links

11 sec to create the routing table of Flow_3 (store in the database)

2058 nodes and 3063 links

2 sec to create the routing table for the TFC switch

# Conclusion

- Implementation of a PL/SQL package to generate routing
- Tests done for the LHCb DAQ and TFC systems
- Need to do more tests : profiling !
- Try to improve performance: code review
- Extending the model with other routing algorithms