

# REPORT

## 1 macro-idea

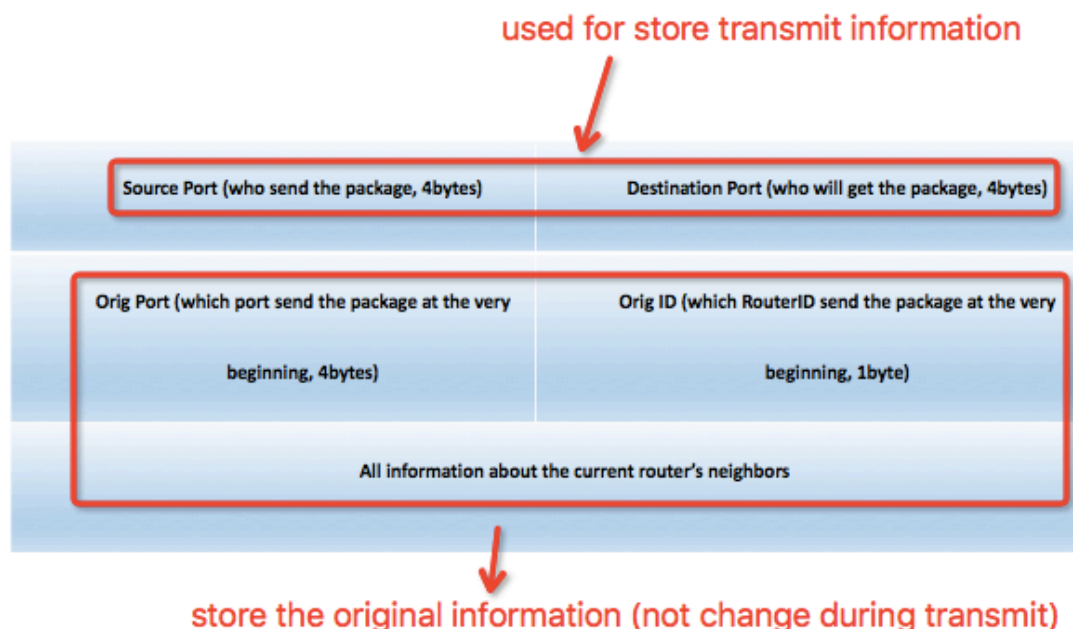
every router need to do these things:

- broadcast itself every 1 second
- keep receiving link state package from all its direct neighbors
- help its direct neighbor router to broadcast (to build a global view of the network topology)

Besides, it need to calculate shortest path to other routers (every 30s, using Dijkstra's algorithm).

## 2 structure

- link state package structure



- router structure  
Router ID;  
Router Port number;  
information about the router's neighbors;  
costMap (it is a ConcurrentHashMap): key is neighbors' router ID, value is the cost;  
portMap (it is a ConcurrentHashMap): key is neighbors' router ID, value is the corresponding port number;

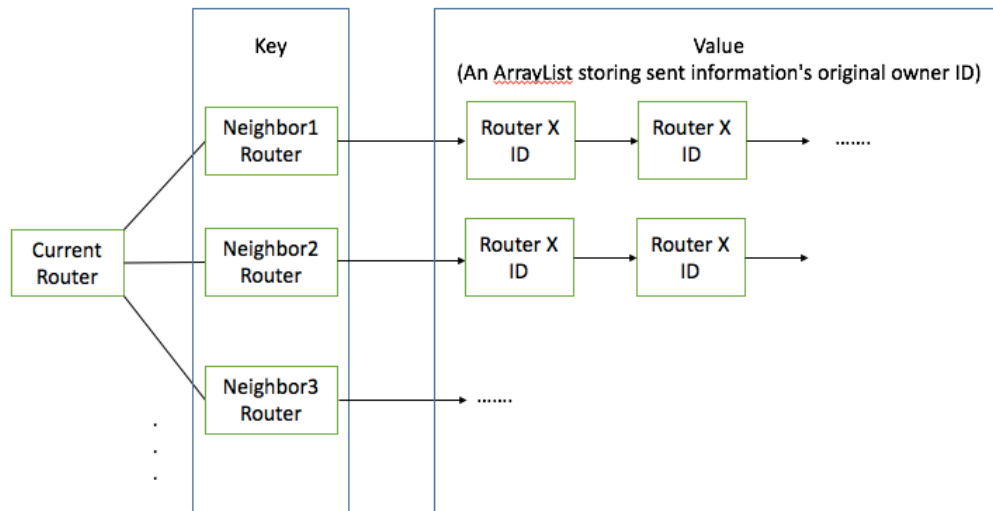
### 3 specific implementation

- multi-thread

Using one thread to keep broadcast itself to its direct neighbor (every 1 second) and using another thread to receive link-state package. Once a router received a package from its neighbor, it also need to broadcast the package to its other neighbors. Besides, using TimerTask to control the process of Dijkstra Algorithm every 30 second.

- restrict Link-state broadcast

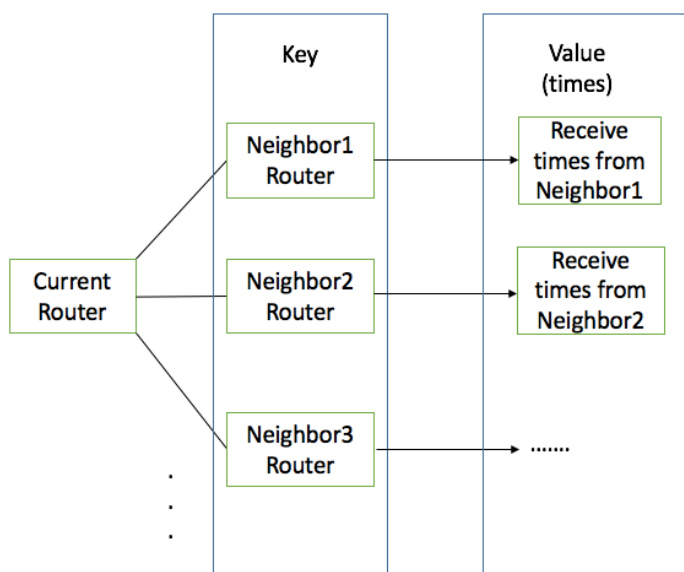
1) recordMap (ConcurrentHashMap):



Note: each key corresponding a array list (to store information sent to key's port)

So, every time we want to send to a Neighbor Router some information, we need to check the information has already in its array list or not. In this way, we can avoid duplicate send.

2)countMap (ConcurrentHashMap):



Once the current router receive package from a neighbor router, the corresponding value is add up 1.

Also in the removing router operation, we just need to check this countMap, if corresponding value is less than 3 times, we could claim that the router has closed.

**Notice: every 30 sec, after using Dijkstra Algorithm, we need to clear these 2 maps.**

- Dijkstra Algorithm

Using an ArrayList S to store coming information (without duplicate). For example: if the current(ID:A) router received a package, and the package's information is owned by router B. And router B is not in S, then we add it into S.

**Notice: every 30 sec, after using Dijkstra Algorithm, we need to clear these S array list.**

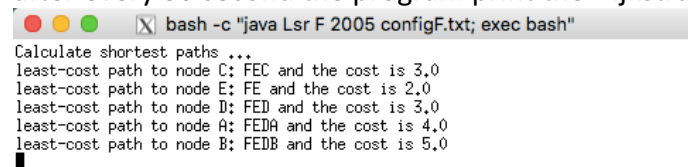
- output format

start with no information:



```
bash -c "java Lsr F 2005 configF.txt; exec bash"
```

after every 30 second the program print the Dijkstra algorithm result and print it out:



```
Calculate shortest paths ...
least-cost path to node C: FEC and the cost is 3,0
least-cost path to node E: FE and the cost is 2,0
least-cost path to node D: FED and the cost is 3,0
least-cost path to node A: FEDA and the cost is 4,0
least-cost path to node B: FEDB and the cost is 5,0
```

using control-c to terminate router and wait at most  $2 * \text{ROUTE\_UPDATE\_INTERVAL}(60\text{s})$ .