Centralized routing prototype

Sampo Kaikkonen
Networking Laboratory, HUT
IRoNet seminar, 8.1.2004

Introduction

- More intelligent routing and Traffic Engineering is needed in the Internet, because
  - users require better service
  - networks' resources are not utilized evenly

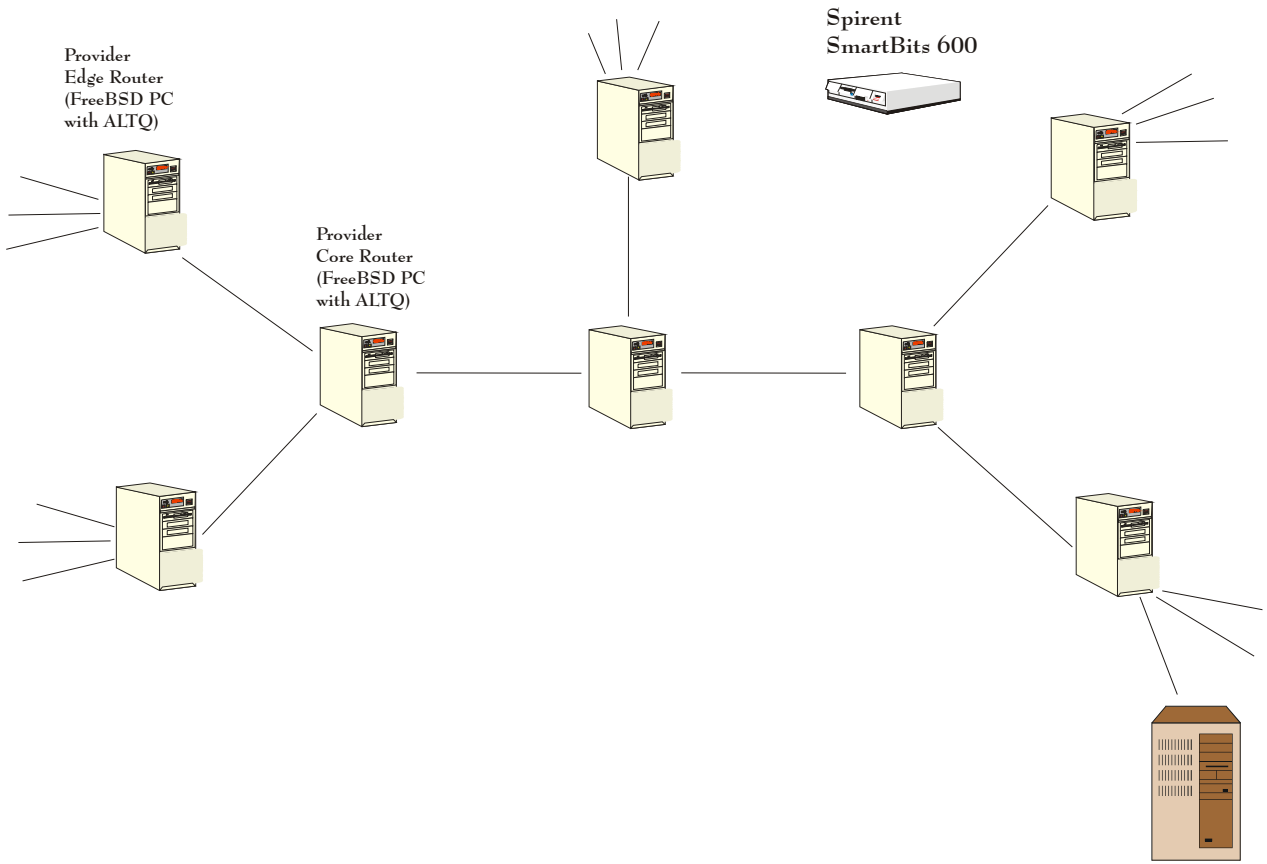
Status of QoSR researching today

- some drafts and RFCs ready (e.g. OSPF-TE, QOSPF)
- lack of systems supporting any QoSR in the Internet
  - now there are emerging some expensive implementations (actually also Zebra would support OSPF-TE, but we don't utilize it entirely)
- researching of algorithms continues
  - testbed needed

Utilizing a simulator to calculate QoS-routes for a real network

- a totally different approach to do ”the same thing”
- network’s topology information is collected to a centralized place where the routes are calculated for every router
- free software can be used

The test network



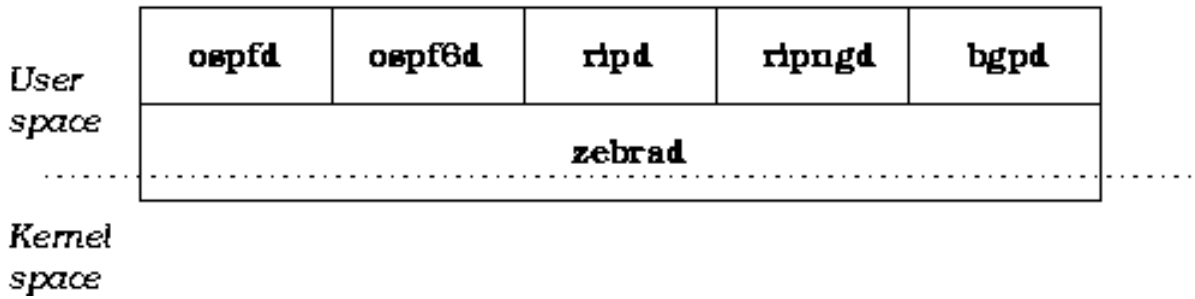
Software used in this work

- Routing software: Zebra
- QoS routing simulator: QRS

In addition to this there are Perl, Tcl(/Tk) and expect –scripts which glue things together and keep the system running

Zebra

- GNU software, founder Kunihiro Ishiguro
- Modular architecture
  - zebrad
  - ospfd
  - ripd
  - bgpd
  - etc.



Zebra's CLI

- well-implemented Command Line Interface resembling Cisco's IOS
- may be used remotely using telnet connection
- daemons can be configured using it and various information (e.g. LSDB) can be printed out

Example from LSDB shown by Zebra (Router-LSA)

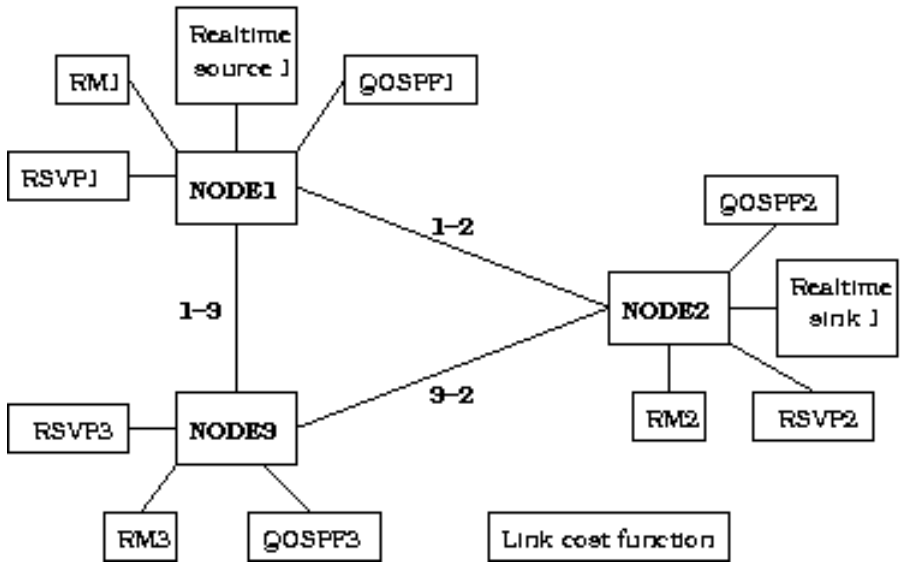
```
ospfd> show ip ospf database router

       OSPF Router with ID (10.10.101.102)


                Router Link States (Area 0.0.0.0)

  LS age: 1267
  Options: 2
  Flags: 0x2 : ASBR
  LS Type: router-LSA
  Link State ID: 10.10.13.1
  Advertising Router: 10.10.13.1
  LS Seq Number: 8000002d
  Checksum: 0xc525
  Length: 72
   Number of Links: 4

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 10.10.10.1
     (Link Data) Router Interface address: 10.10.10.1
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: Stub Network
     (Link ID) Network/subnet number: 10.10.11.0
-      (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metric: 10
```

QRS

- developed by Dr. Peng Zhang (Networking lab.)
- based on MaRS
- network topology is defined in a configuration file containing components

QRS components

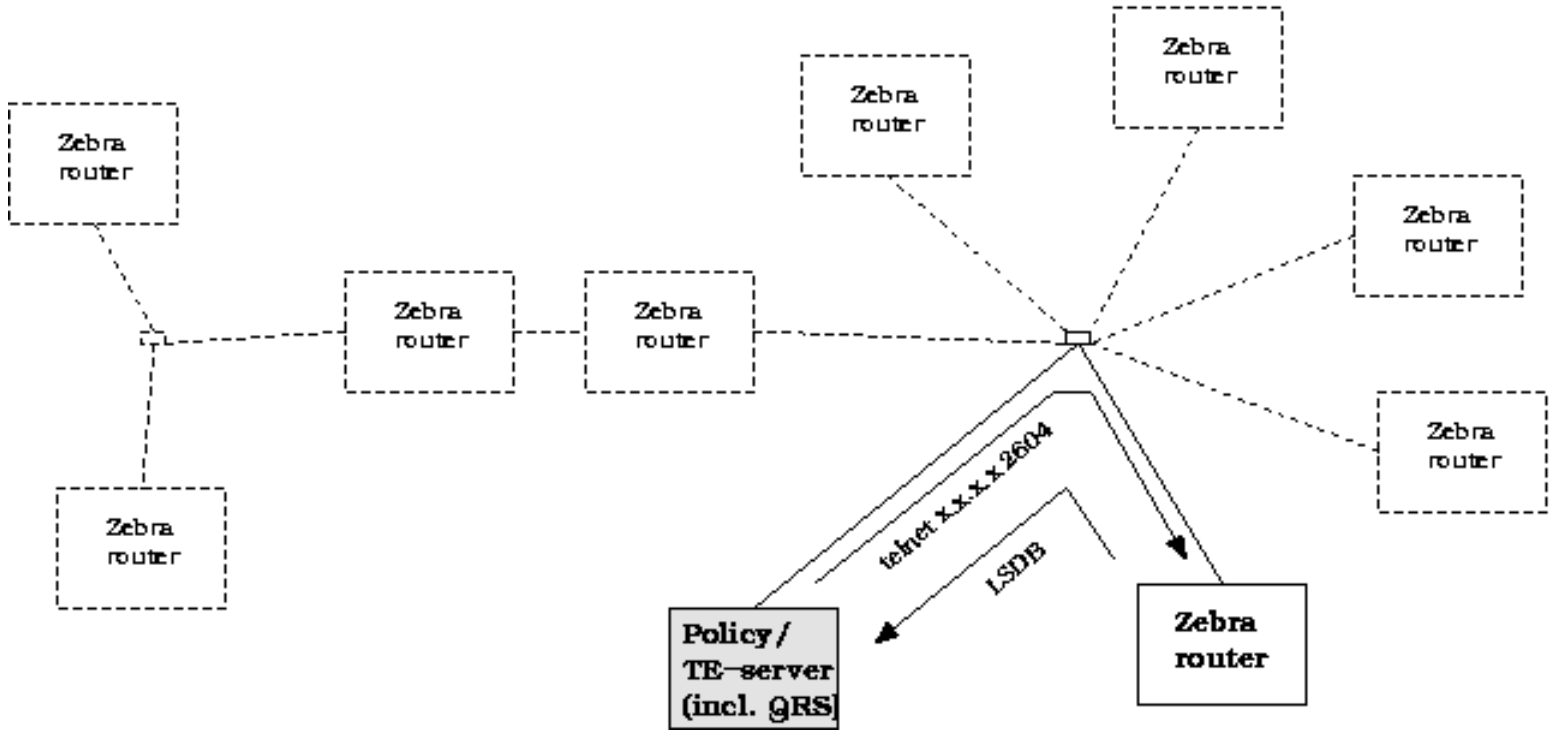
- node
- link
- qospf
- rsvp
- rm
- sources, sinks
- link cost function



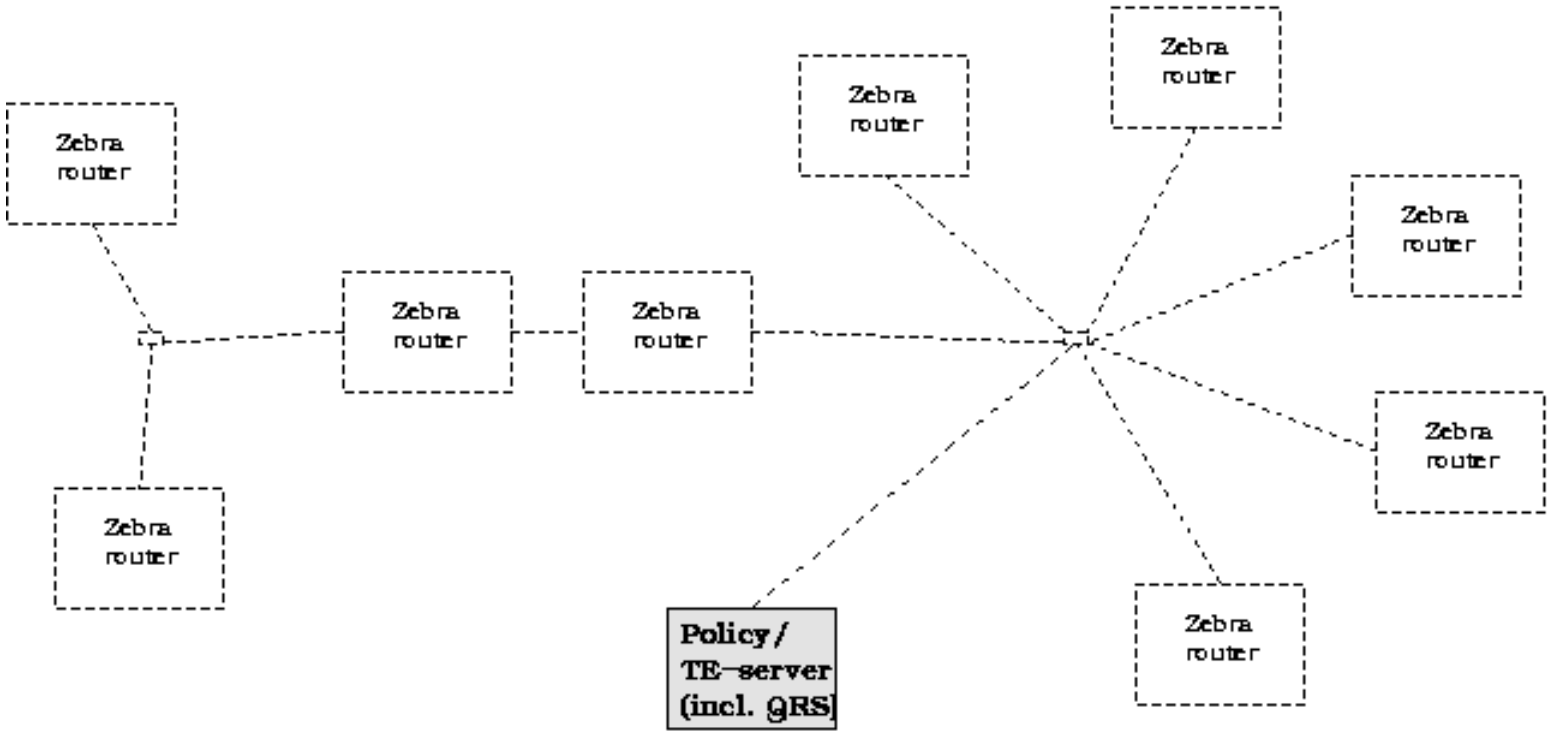
A very short example from QRS parameter file (node comp.)

```
component 'node1' NODE 0 0
param 'node1' 32 0          # node1
param 1000 82 0             # Delay to process a packet (uSec): 1000
pflags 0 0    # Speed of node (uSec/kbyte): 0
param -1 82 0 # Buffer space in bytes (-1=inf): -1
param -1 82 0 # Mean time btw failures (sec): -1
param 1 82 0  # Interfailure dist (0=>EXP, 1=>UNIF): 1
param 1000 82 0             # Enter standard deviation if UNIF: 0
param 1200 82 0             # Mean time to repair (sec): 1200
param 0 82 0  # Repair time dist (0=>EXP, 1=>UNIF): 0
param 1000 82 0             # Enter standard deviation if UNIF: 0
pflags 26 0   # Node status: Up
pflags 2e 4   # Buffer space used: 0
pflags 2e 4   # Max buffer space used: 4920
pflags 2e 4   # Number of packets dropped: 1
pflags 2e 4   # Instantaneous drop rate: 0
pflags 2e 4   # Memory utilization: 0
pflags 2e 4   # Input routing queue has 0 pkts
pflags 2a 8   # flow table
pflags 2e 4   # lk1-2 output queue has 0 pkts
pflags 2e 4   # lk1-4 output queue has 0 pkts
pflags 2e 4   # lk1-7 output queue has 0 pkts
```

System running - phase 1

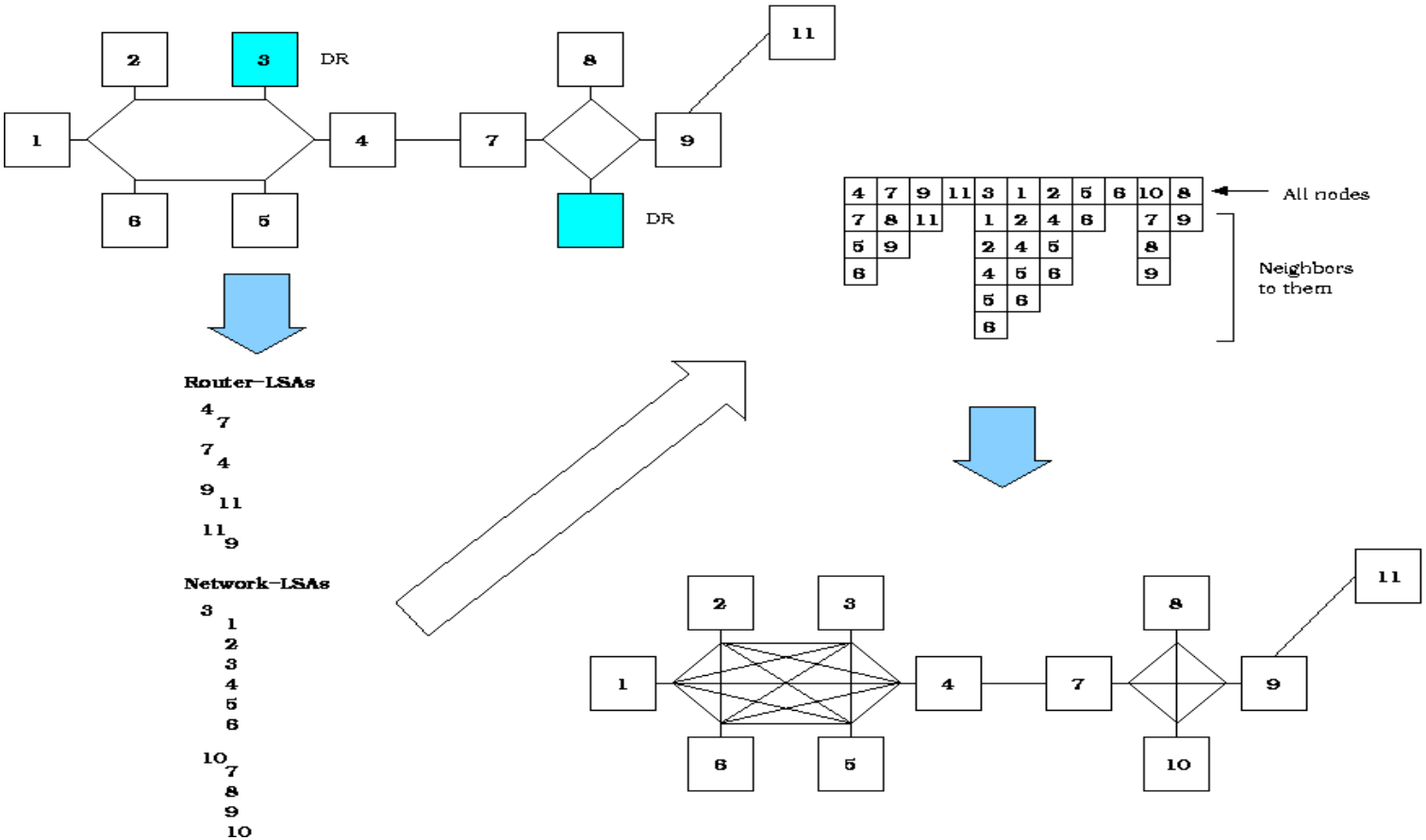


System running - phase 2

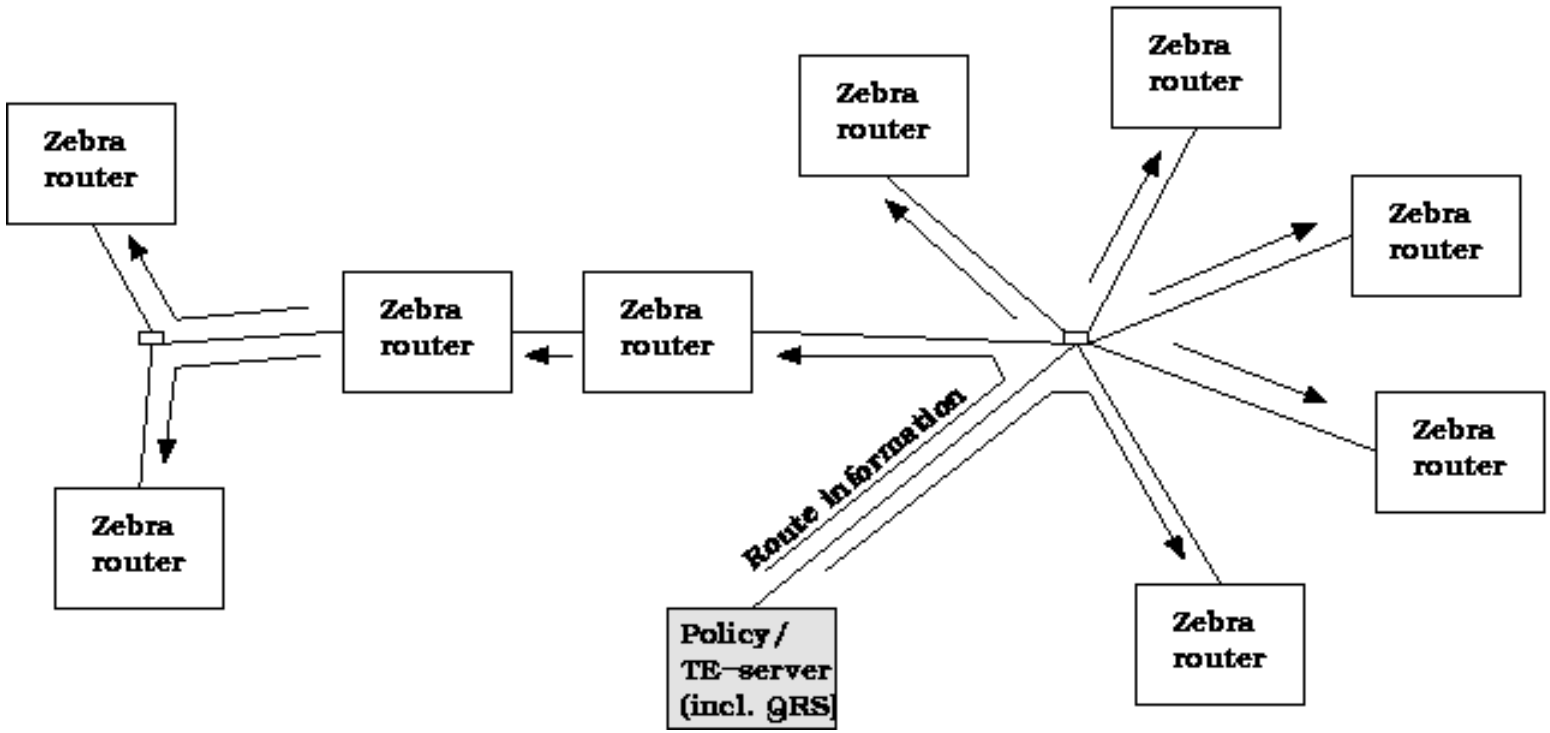


1. Topology information is converted to another format
2. QRS calculates the routes
3. The routes for different routers are written to files

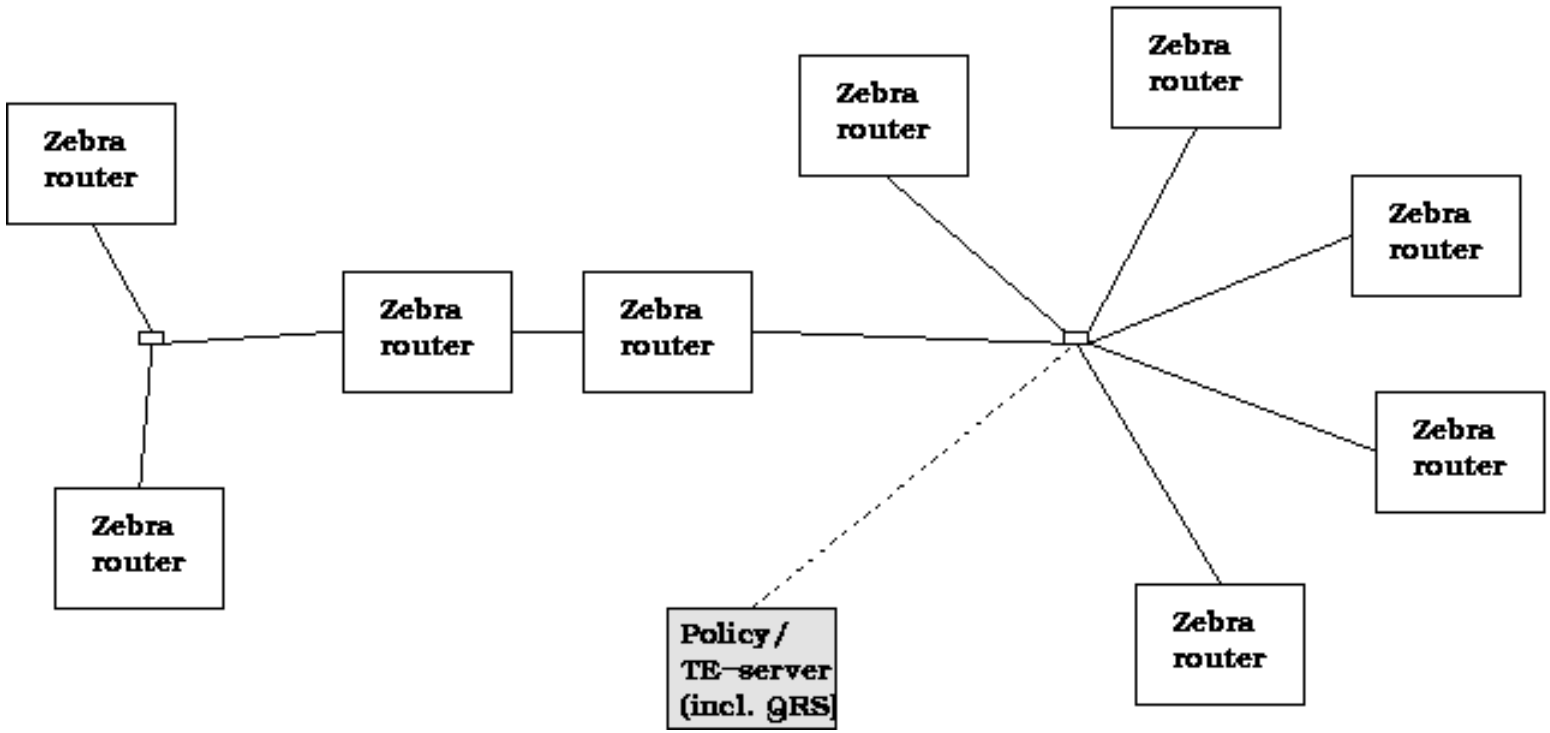
Topology information Conversion



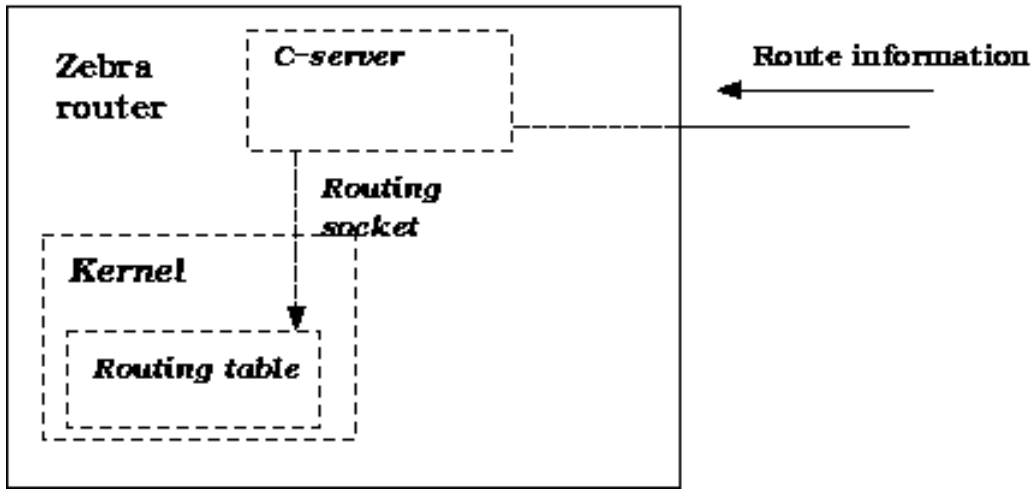
System running - phase 3



Routers running until the next route calculation

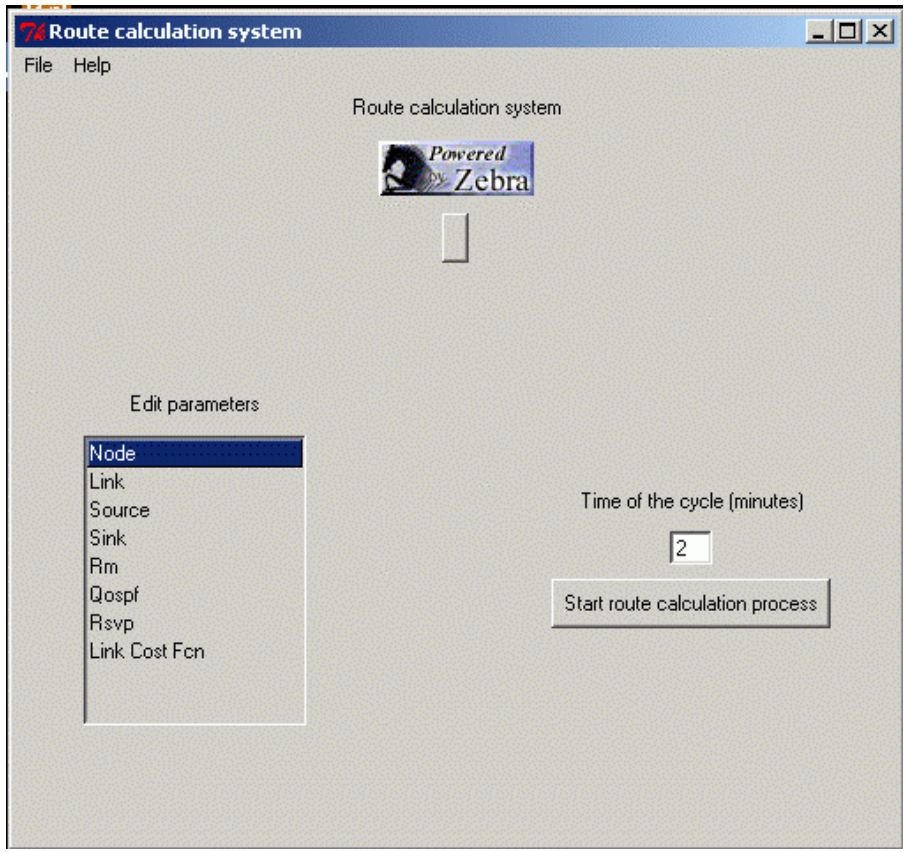


Updating the routing table using the routing socket

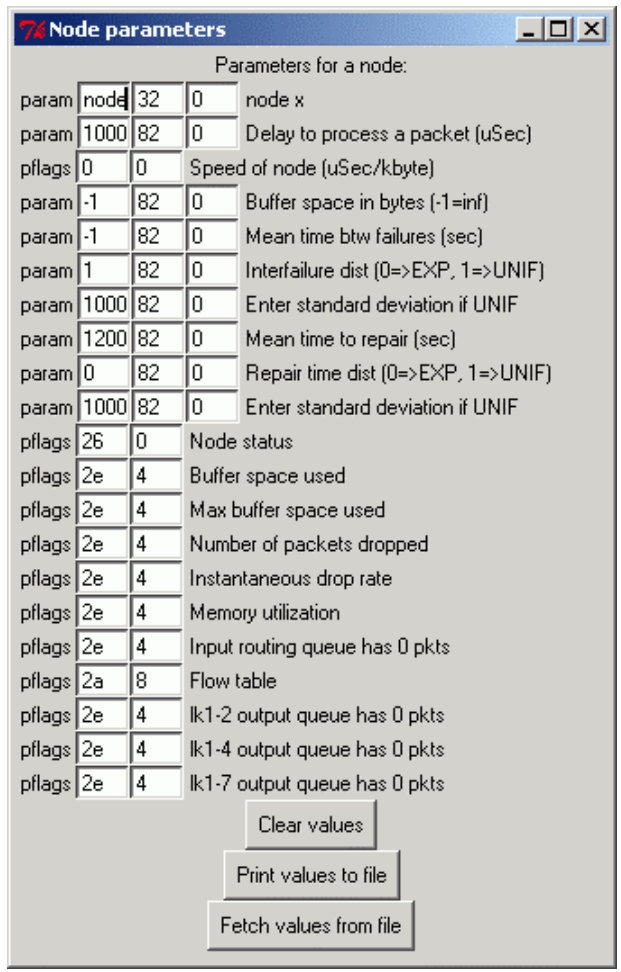


- Note! The *Zebra* has been modified so that it doesn't update the routing table

Graphical user interface



GUI - parameter window



Conclusions

- only a simple prototype which can be developed further by others
- more parameters should be updated automatically (e.g. using SNMP)
- with this kind of system the QoSR-algorithms could be researched easily
- still measurements/stress tests left

Questions?