HELSINKI UNIVERSITY OF TECHNOLOGY
Networking laboratory

Centralized Routing of IP Networks using QRS Routing Simulator

Sampo Kaikkonen

Networking Laboratory, HUT

IRoNet seminar, 17.2.2004

Introduction

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

QoS-routing and TE

- QoS-routing in the Internet would be an excellent solution to enhance the utilization of the network resources
- The present protocol implementations have been more or less in the development phase (though now some intra-area solutions have been emerging)
- Idea: why not to use an existing QoS-routing simulator to do the route calculations!

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

- A totally different approach to do ”the same thing”
- Network’s (one OSPF area) topology information is collected to a centralized place where the routes are calculated for every router
- Free software can be used

But…

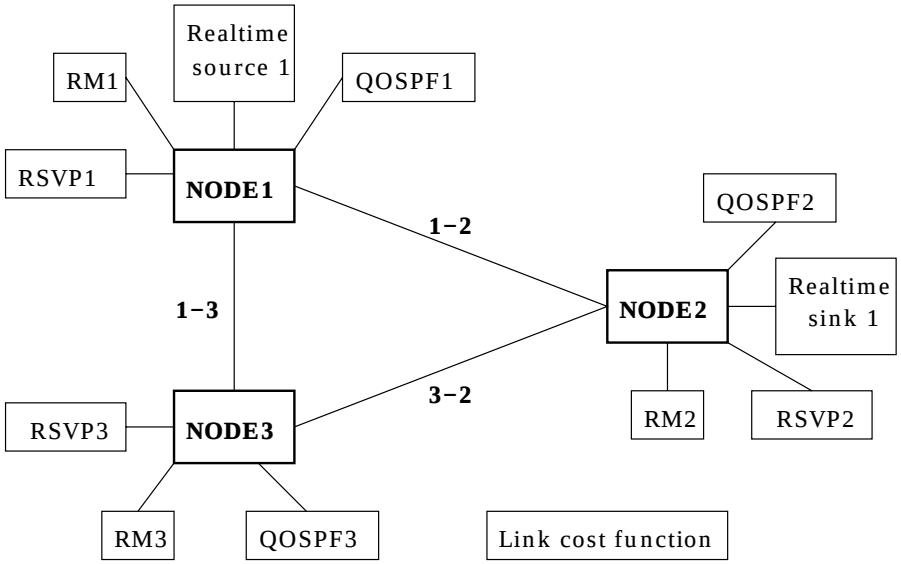
- However, it's complicated to collect the information of the network's **state** periodically and give it to the simulator
- So, in this work the main goals have been:
  - to transfer the network's topology information to a format that the simulator understands
  - to convert the simulator's routing tables to real ones

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
```

The QRS way to do define topology

- The ideology how to define a topology is totally different in QRS than in the real world
- QRS doesn't use subnets or IP addresses at all → simplifications are needed in conversions
  - pros: route calculation is faster
  - cons: 1) difficult to get working well 2) the routes are not always the most optimal ones, but doesn't matter so much in the real life because the users reside usually in the stub networks

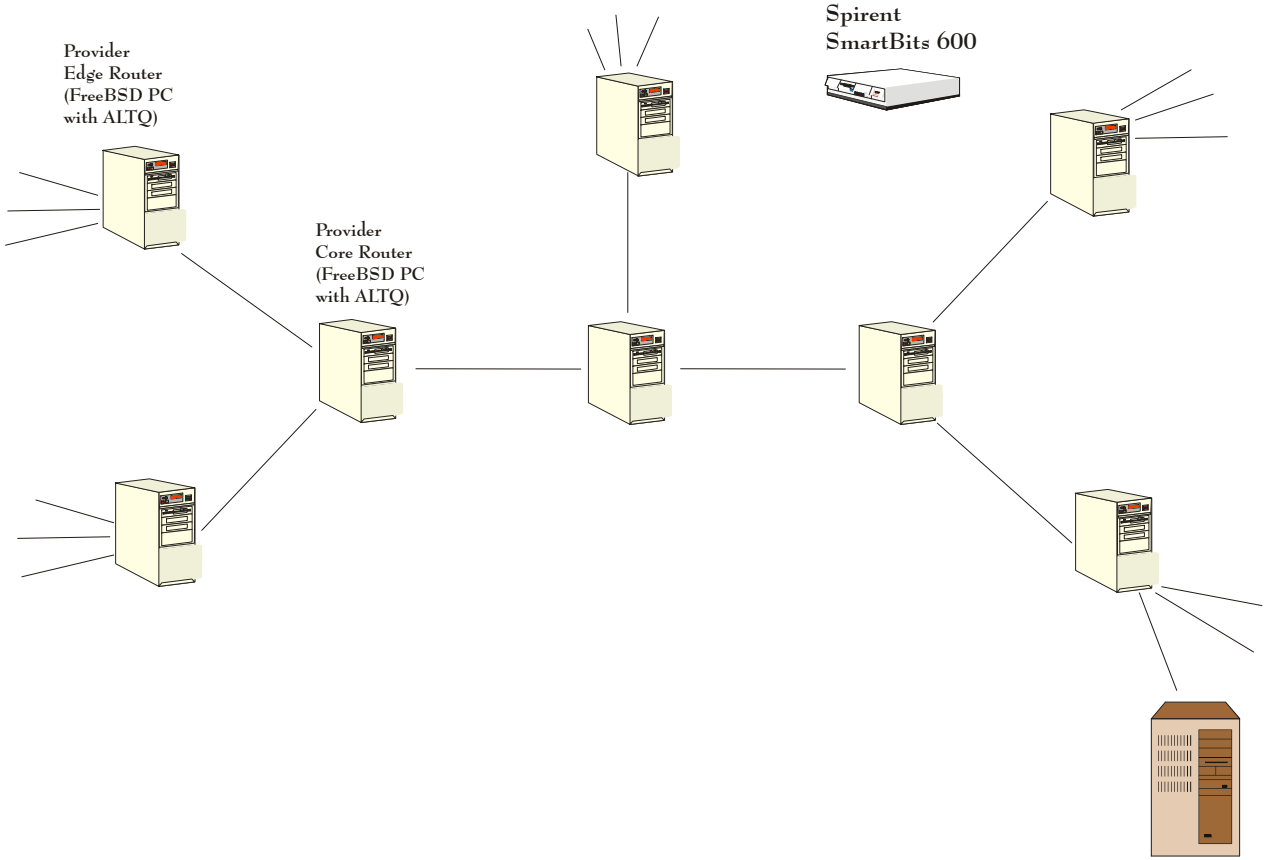
Problems with QRS in this work

- All in all it has felt difficult to fit the QRS to the whole entity, it's not intended to be used in this purpose
- There has also been problems with routing table printings of QRS
- Anyway there is still effort to get the QRS working
- There is nowadays another similar project called TOTEM in Belgium, we may have some co-operation in the future so QRS could be replaced with more suitable simulator

The test network

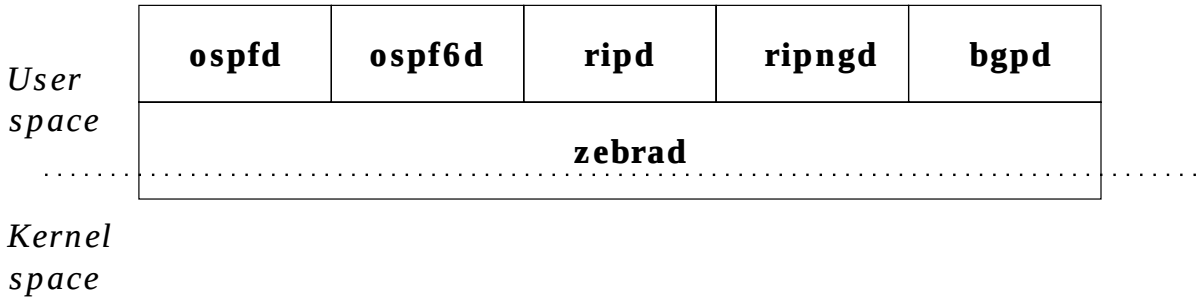
- Route calculation using QRS is done in the Policy Server
- The routers (FreeBSD PC:s) contain Zebras
  - zebra daemons are modified so that they don't update the routing tables

The test network



Zebra

- GNU software, founder Kunihiro Ishiguro
- Several routing daemons, modular architecture
  - zebrad
  - ospfd
  - ripd
  - bgpd
  - etc.



Zebra

- We use ospfd (and zebrad) only
- There is a Command Line Interface in zebra
- Using an `expect` script the LSDB can be fetched to a text file
- Also SNMP (OSPF MIB) could be used and there has been some experiments

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
```

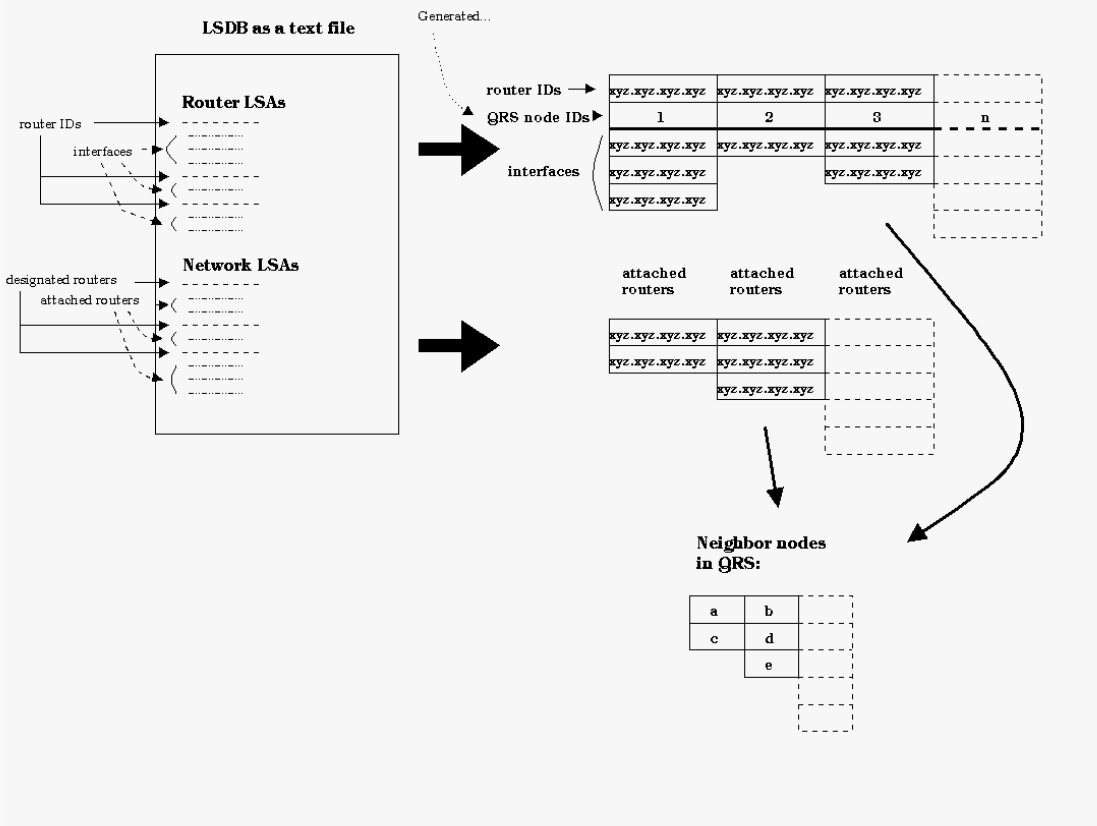
Scripts that glue QRS and Zebra's together

- There are Perl, Tcl/Tk and expect –scripts which glue things together and keep the system running
- The topology info is converted using Perl because it has useful lists as data structures and it's suitable for parsing

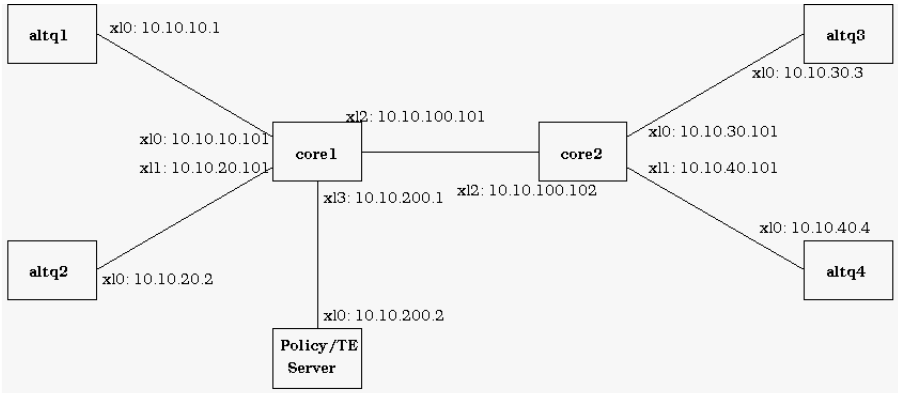
Topology information Conversion ”algorithm”

- First the router-LSAs are parsed:
  - the Router ID of each router is inserted to first line of a list and the corresponding QRS node ids are generated
  - the interfaces of the routers are listed under them
- Then the network-LSAs are parsed:
  - the ”Attached routers” are inserted to another list (one column corresponds to one subnet having a particular Designated Router)
  - a list having corresponding QRS ids is created

Topology information Conversion



An example



router_IDs_QRS_IDs_interfaces

router IDs →	10.10.10.1	10.10.20.2	10.10.30.3	10.10.40.4	10.10.100.101	10.10.100.102
QRS node IDs →	1	2	3	4	5	6
interfaces	10.10.10.1	10.10.20.2	10.10.30.3	10.10.40.4	10.10.10.101	10.10.30.101
					10.10.20.101	10.10.40.101
					10.10.100.101	10.10.100.102

neighbors_router_IDs

10.10.10.1	10.10.20.2	10.10.30.3	10.10.40.4	10.10.100.101
10.10.100.101	10.10.100.101	10.10.100.102	10.10.100.102	10.10.100.102

→

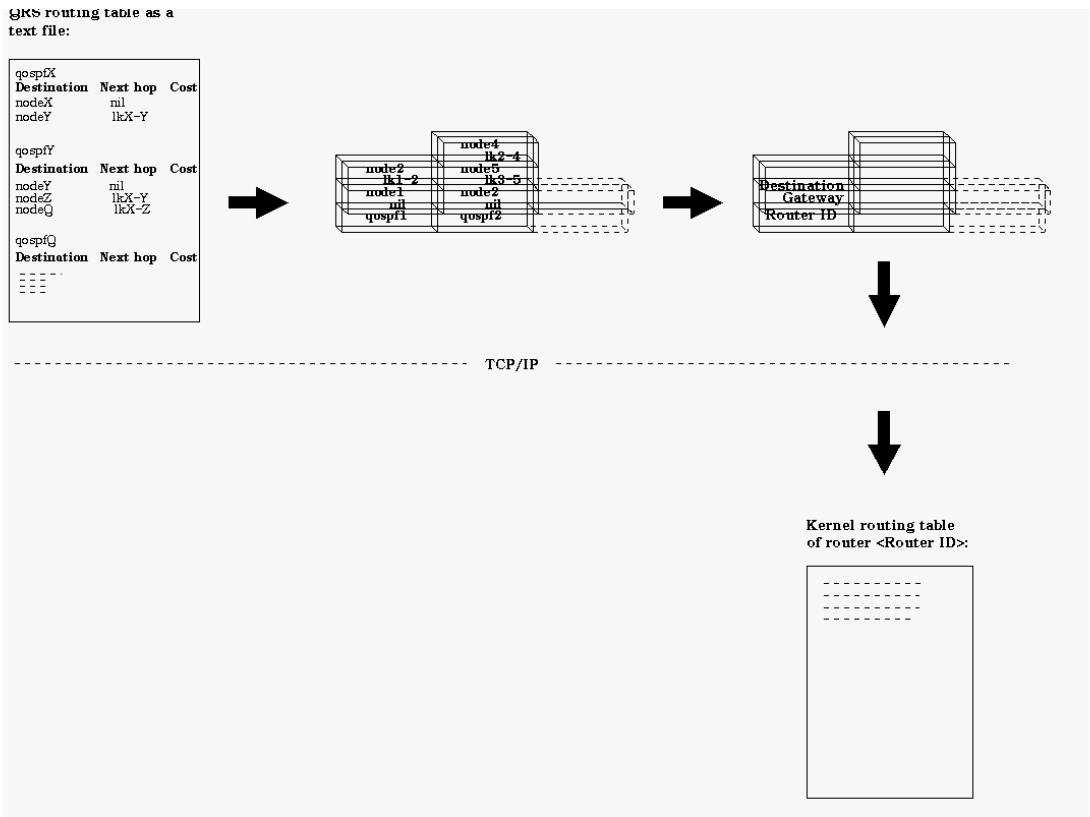
neighbors_QRS_IDs

1	2	3	4	5
5	5	6	6	6

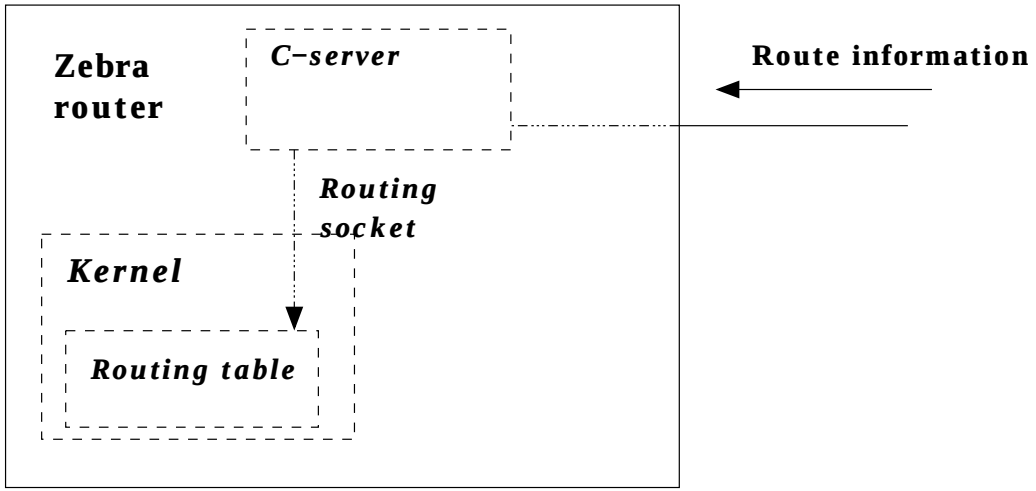
Converting QRS routing tables to real ones, ”algorithm”

- The QRS’s routing tables are read to a multidimensional list (the nodes in horiz. array and the routes for them in vertical arrays)
- These are converted to real routing tables
  - difficult because QRS doesn’t have IP addresses / subnets
  - however has worked with some simplifications
    - e.g. the ”subnets” are class C, one particular node ”takes care” of the subnet (the one having the largest id), etc…
    - when finding the correct gateway or dest. addr. the interface lists of the neighbors are trawled and when the ntw addresses match that’s the right one…

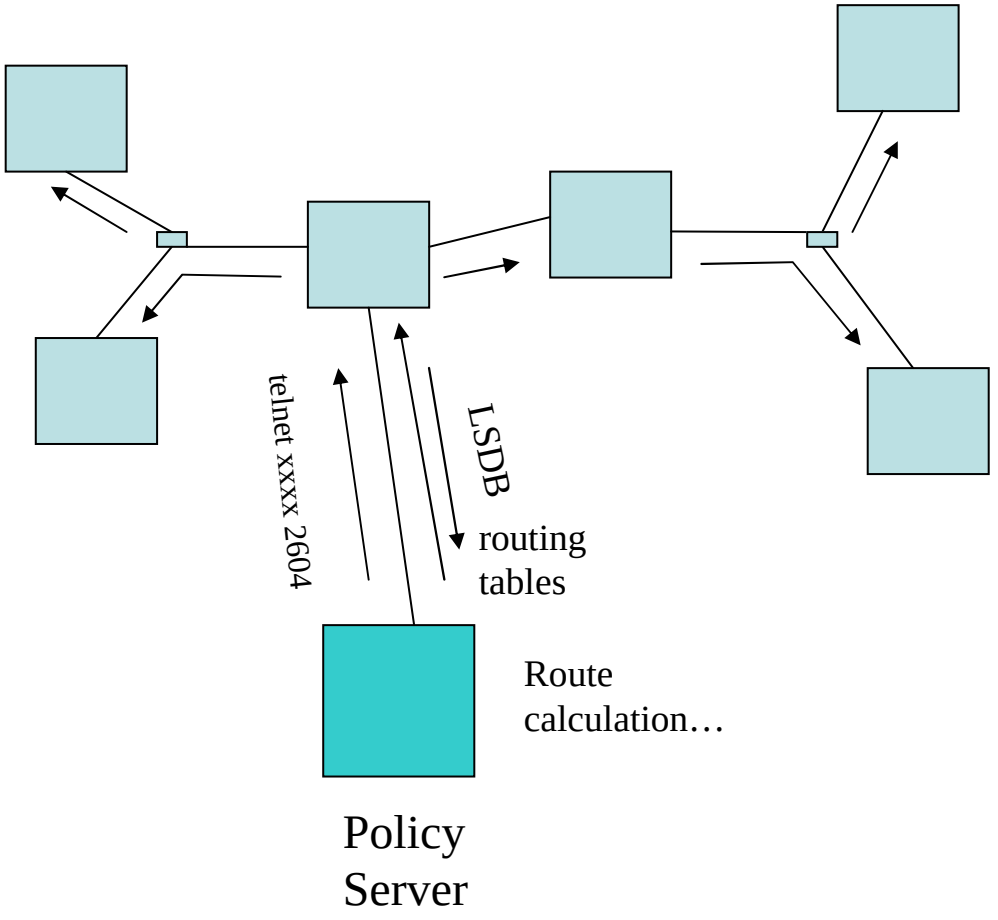
Converting QRS routing tables to real ones



Updating the routing table using the routing socket



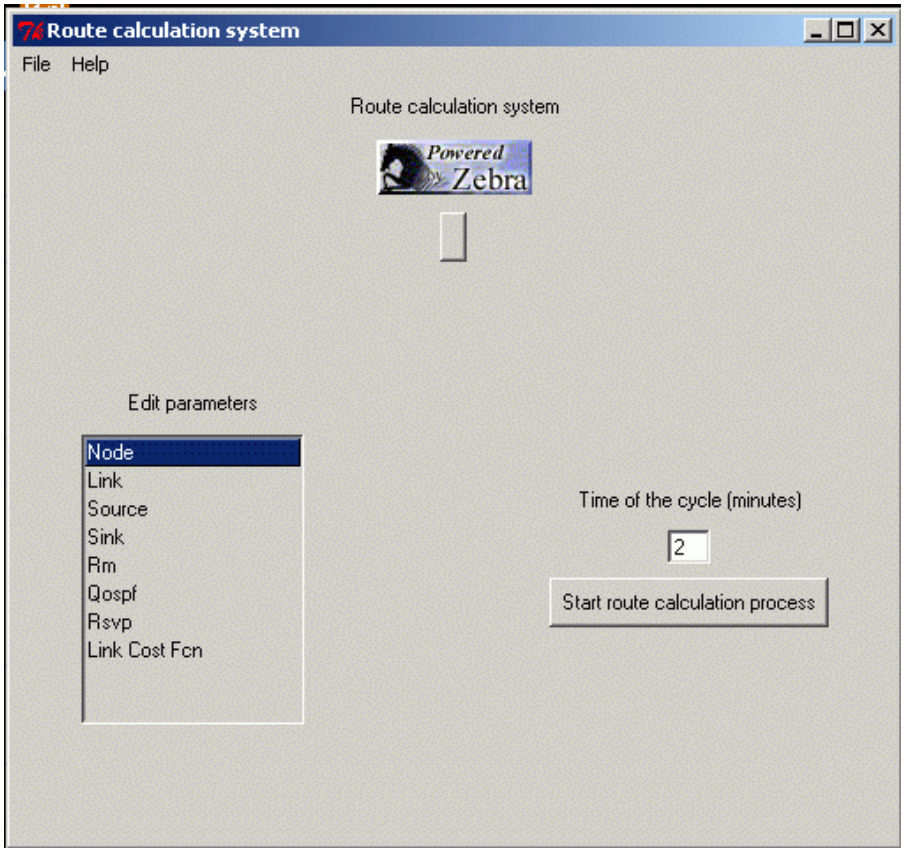
- Note! The *Zebra* has been modified so that it doesn’t update the routing table


telnet xxxx 2604
LSDB
routing
tables
Route
calculation…
Policy
Server

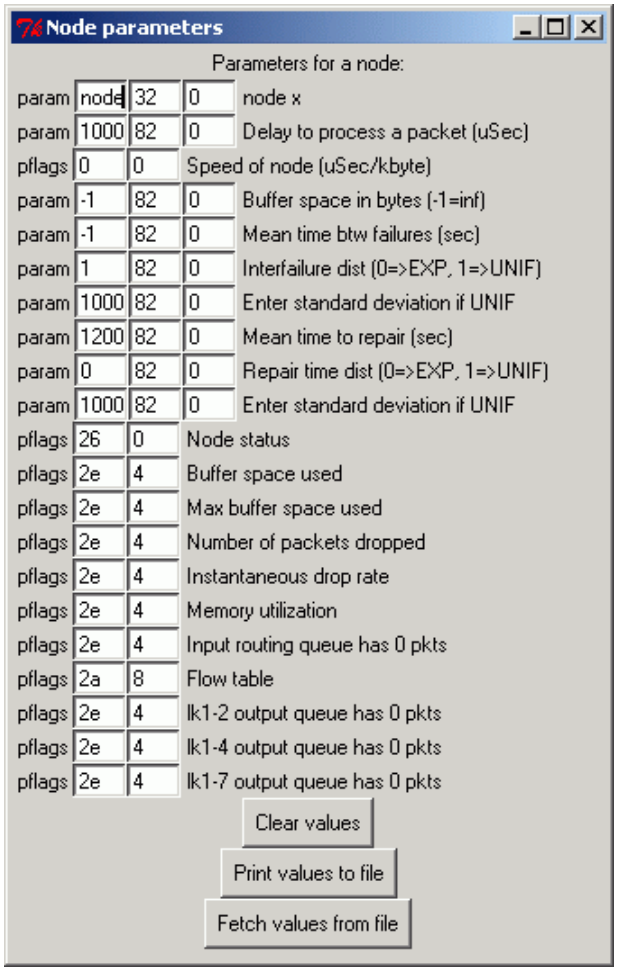
Static information

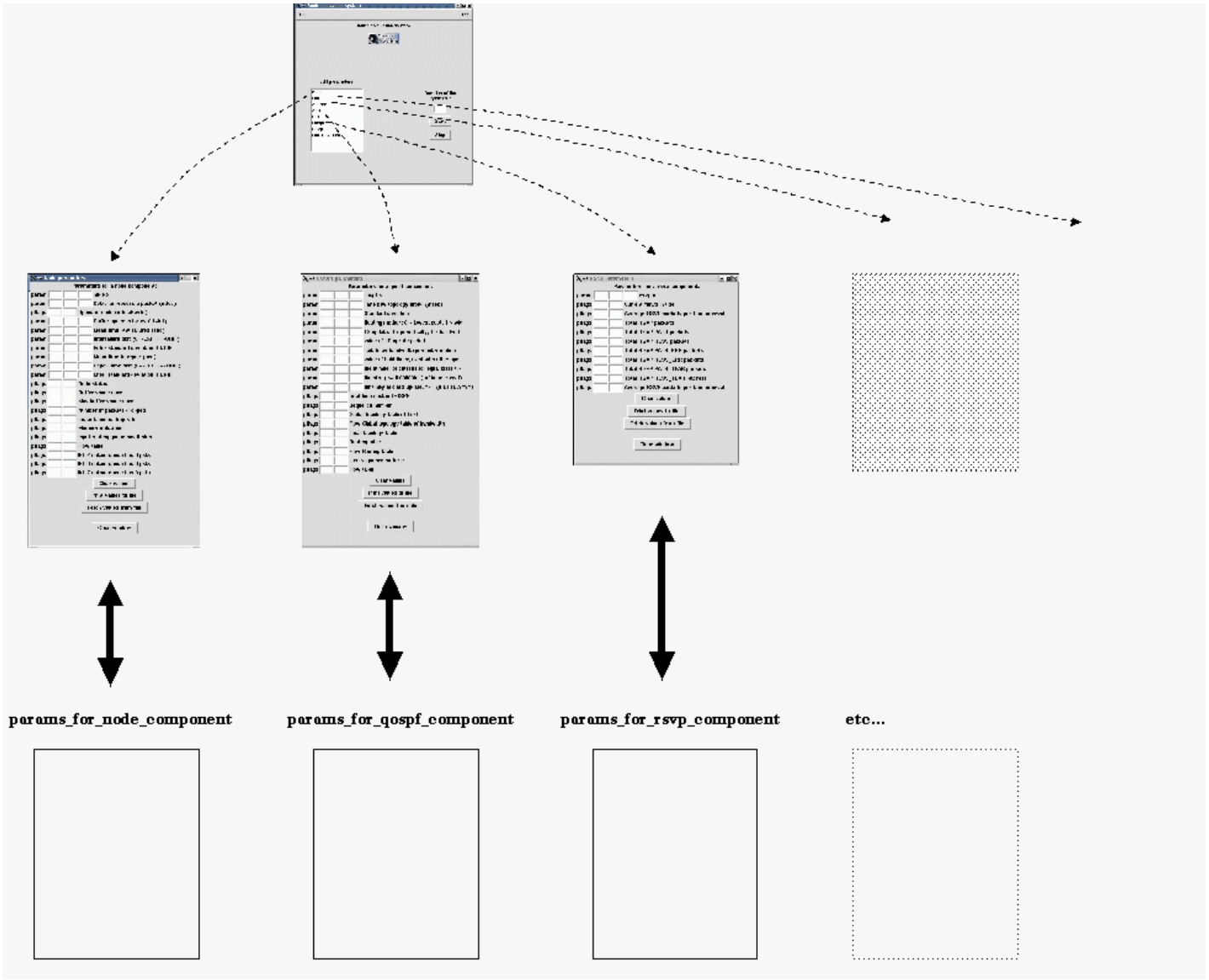
- In this work all the information related to the state of the network is static
- There is a graphical user interface which can be used to insert parameters for QRS components
- Of course should be done dynamically in the future

Graphical user interface



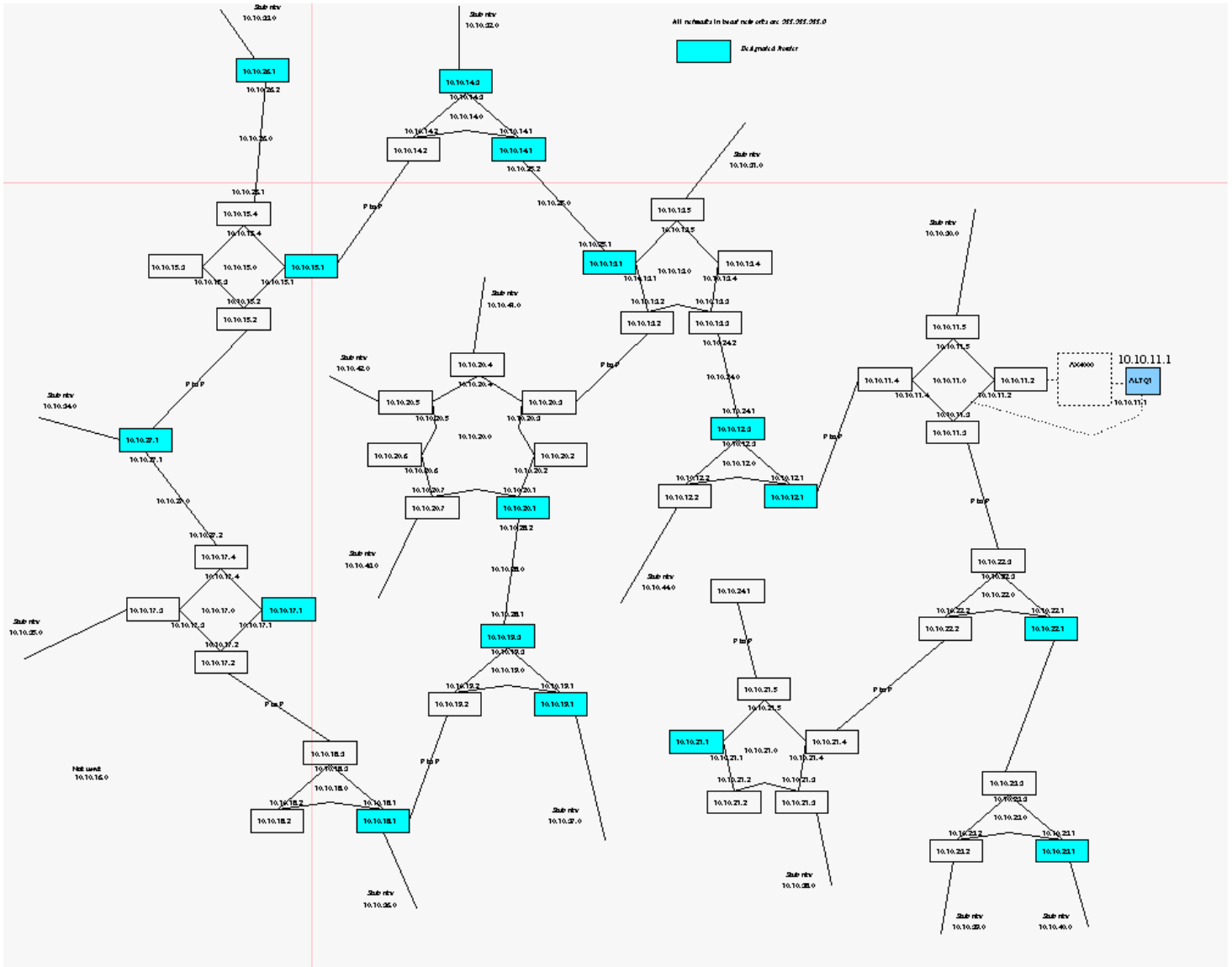
GUI - parameter window




params_for_node_component
params_for_qospf_component
params_for_rsvp_component
etc...

Measurements

- Due to the problems with the QRS (in this work!) there are no accurate measurement results to show
- However the QRS itself calculates routes very fast and the topology conversions happen in the blink of an eye (e.g. for circa 50 routers)
- Some stress test have been done and the conversions seem to work well at least for a topology with 50 routers (Spirent Adtech AX4000's routing emulation is used to create the topology)


10.10.11.1

Conclusions

- It hasn’t been easy to fit QRS into real world OSPF route calculation process
- Anyway the topol. info conversion, routing table conversion, RT update via Routing Socket, GUI etc. are working
- 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 also the QoSR-algorithms could be researched easily