Πρώτη Σειρά Ασκήσεων για το ΗΥ-535

Τα προτεινόμενα θέματα είναι τα παρακάτω. Θα έχετε 2 εβδομάδες για την υλοποίηση της άσκησης. Σε όλες τις περιπτώσεις μπορείτε να χρησιμοποιήσετε κώδικα απο άλλα μαθήματα ή open source εαν φυσικά το επιτρέπει το copyright του. Όλες οι ασκήσεις μπορούν να υλοποιηθούν στο προσωπικό σας υπολογιστή (εφόσος αυτός τρέχει Linux). Θα υπάρχει στην διάθεση σας και το εργαστήριο όπου θα μπορείτε να υλοποιήσετε τις ασκήσεις και να πειραματιστείτε. Για την υλοποίηση των ασκήσεων θα χρησιμοποιήσουμε την τελευταία έκδοση (0.99.5) του Quagga (http://www.quagga.net).

Study the event scheduling in Quagga

Study the event scheduling mechanism of Quagga and how OSPF is using it. In particular we want to answer the following questions:

- What kind of event prioritization is allowed by the scheduling? Can I for example prioritize hello over other packets both on recv and send?
- If the prioritization allowed is not enough, propose how you would improve it.
- How scalable is the implementation? Where do I do O(n) operations? For each of these cases propose how you could fix it.
- What kind of tasks do I have in OSPF? Are there any problematic tasks that may take too long and make the implementation unresponsive? For example an SPF that can run for too long, or packet queues that get too large?

Fast convergence in quagga OSPF

Study and report what Quagga does in terms of fast convergence. Implement some of the methods we have seen for improving the convergence time of OSPF. Consider:

- Faster Hellos
- Less pacing of LSAs
- More aggressive SPF computation
- Others…
- How would you prove that you changes work and indeed improve the convergence of the protocol?

OSPF tester LSA generator

Design and write a program that can be used to stress-test an OSPF implementation. The program will run on a box establish an OSPF adjacency with the box we want to test (device under test – DUT) and will feed it the information we want. The program should generate valid HELLO and LS-upd, LS-ack packets at rates that can be configured.

- Use ideas/code from related open source projects if you can find them
- Reuse code from OSPF for packet generation
- Reuse the whole OSPF implementation
    - Find a way to create an artificial topology database (maybe reflecting a particular network topology – see Rocketfuel topologies)
    - Ensure that we can generate streams of various packets at configurable rates
- How would you adapt your program to perform timing measurements: e.g. time to forward an LSA update, time to do an SPF etc…

Implement (a simple) BFD in Linux/Quagga

- Decide where to put it in the Quagga architecture
    - kernel module?
    - New user space daemon?
    - Inside zebra?
- Need to be able to configure it, start/stop it in each interface/neighbor
- When it detects failure in an interface, must notify the protocols that use this interface