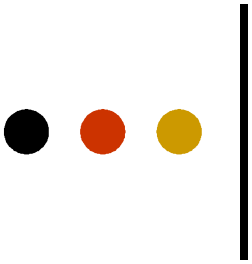


Netfilter Failover

Connection Tracking State Replication

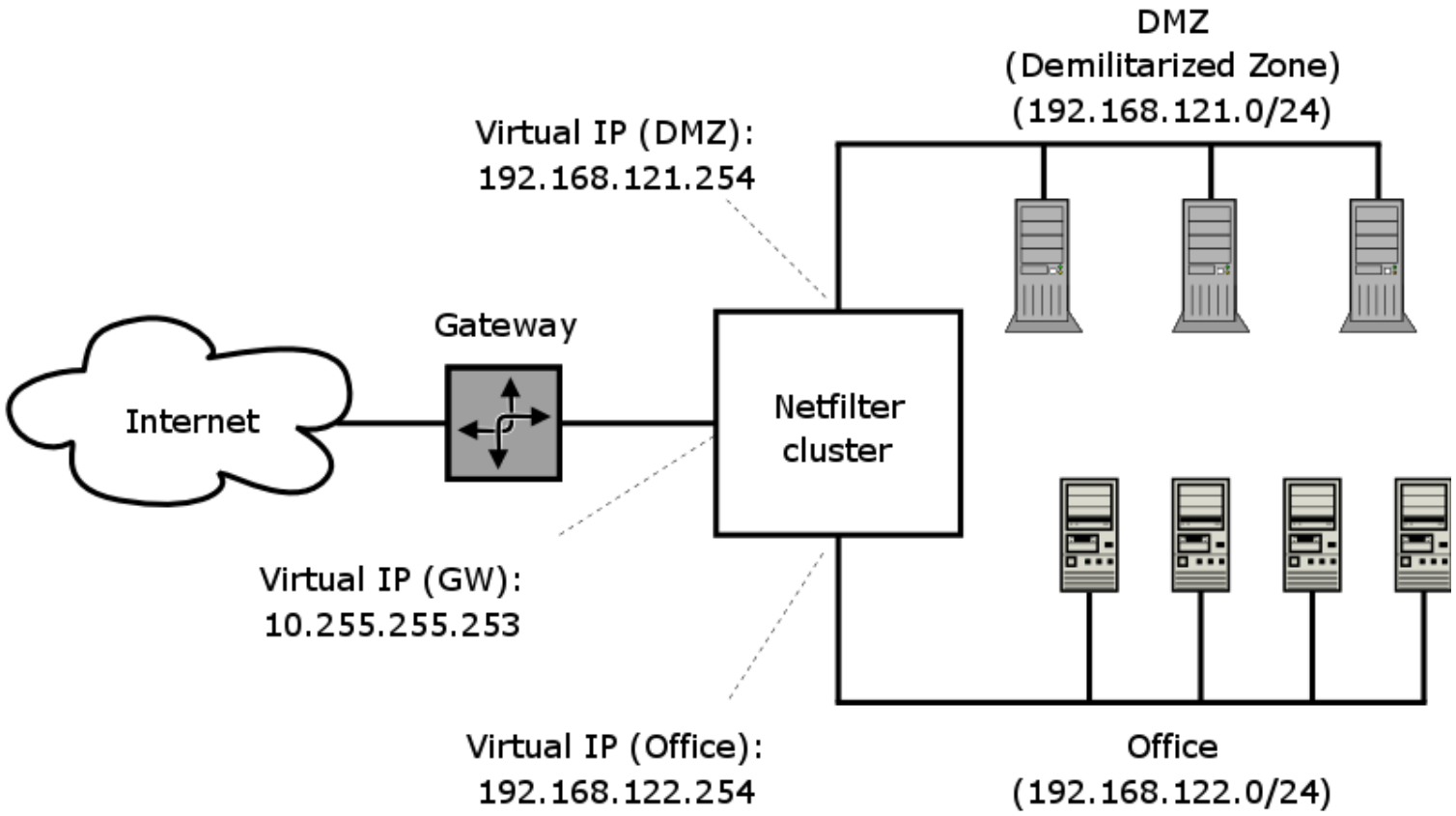
Krisztián Kovács <hidden@sch.bme.hu>

2003.08.17

Original idea

- Harald's OLS 2002 paper: “How To Replicate The Fire – HA For Netfilter Based Firewalls”
- The problem: Netfilter is a *stateful* packet filter, we have to replicate the
  - conntrack entries
  - NAT data

Network architecture



Cluster interface and internals

- Master-slave system
- Cluster is reachable through a *virtual IP*, packets sent to this IP are processed by the master
- An internal replication network is available (high-speed, secure, etc.)
- All nodes have the same configuration
- Two (almost) separate problems to solve:
  - IP failover
  - State replication

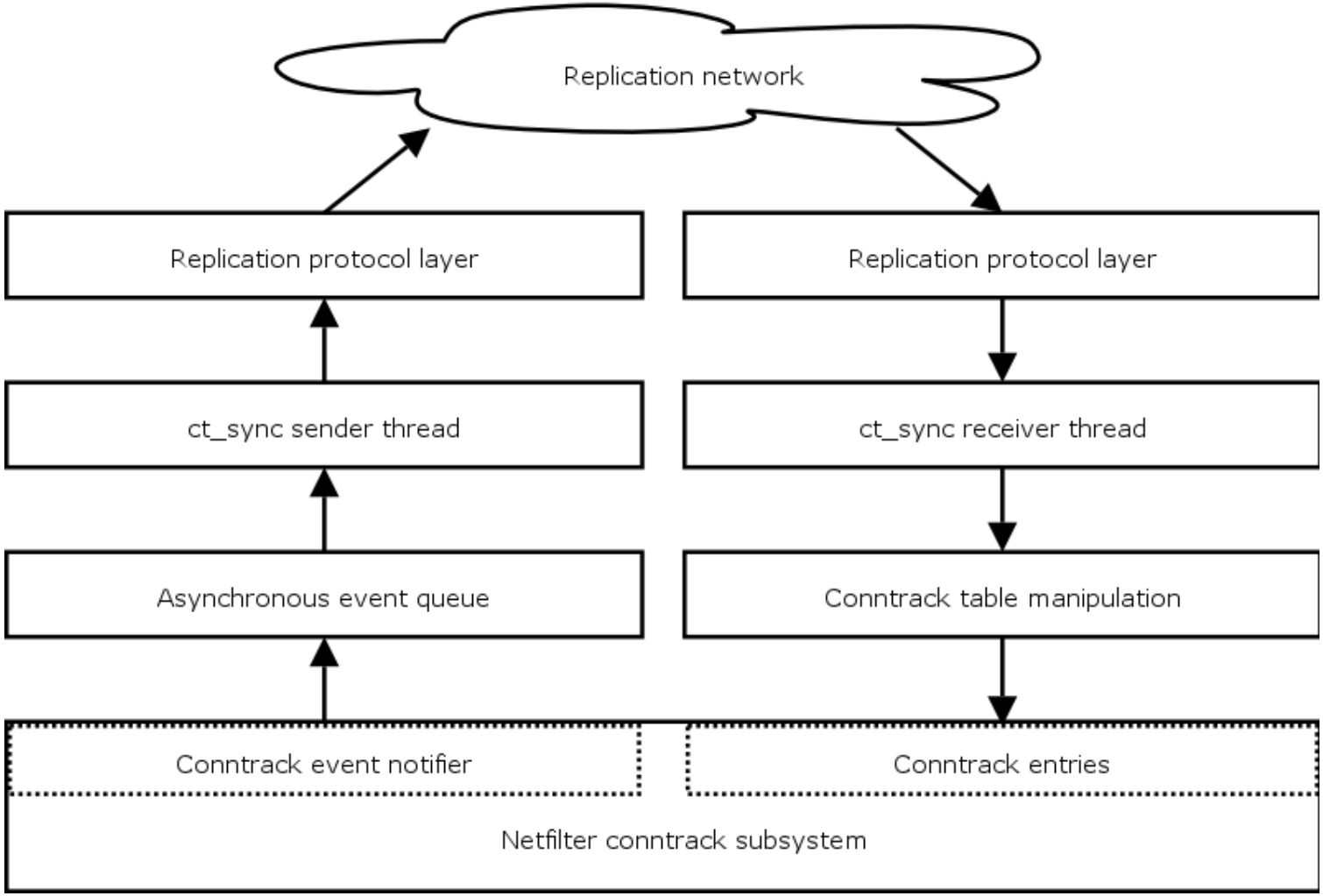
IP failover

- Don't reinvent the wheel: use VRRP (Virtual Router Redundancy Protocol)
- Provides:
  - master election protocol
  - virtual IP configuration on all interfaces
- VRRP implementation: *keepalived* daemon originally developed for LVS
- We need to be able to get notified when state changes occur: this is possible with *keepalived*

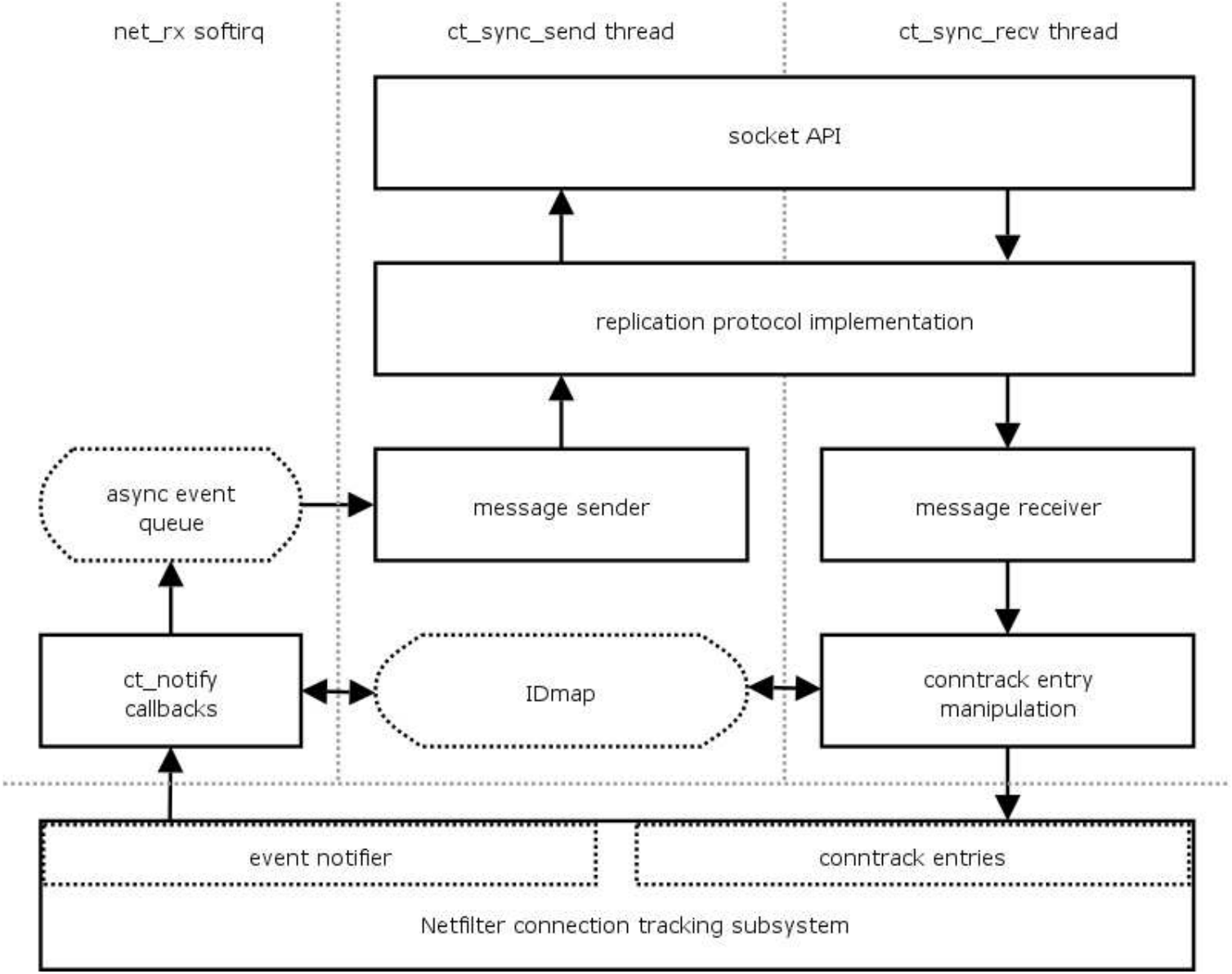
State replication

- Two roles: master and slave
  - Master generates replication protocol messages from conntrack events
  - Slave listens for these messages and updates the conntrack entries accordingly

ct_sync architecture



ct_sync implementation



Problems

- Dynamic structures (linked lists, pointers)
  - Data structures have to be serialized
- Cluster-level unique IDs are needed instead of pointers
  - Harald's approach: since the address of the ip_conntrack structure does not change, use that as an ID
  - This is not correct: when failover happens, the addresses change (they are not guaranteed to be the same on the slaves!)

Problems

- Refresh and timers
  - Refresh events occur too often, so they cannot be replicated
  - This causes inconsistency: if the timers are not refreshed on the slaves, conntrack entries may timeout too early
  - Because of this, timers are not activated on slaves, they are initialized at creation, and started only if a slave->master transition occurs

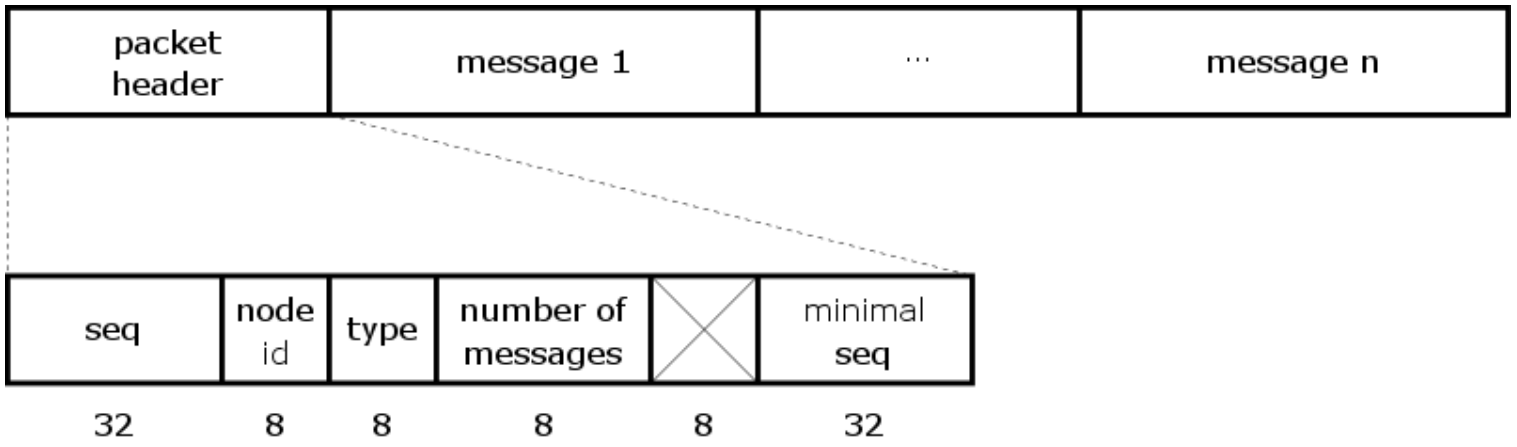
Replication protocol

- Protocol messages must not be tracked
  - NOTRACK should be used
- Multicast UDP based
- NACK (Negative ACKnowledgement) based error recovery
  - Each packet has a sequence number, master has the last N packets sent in memory
  - If a slave detects that the received packet has an invalid sequence number, it requests retransmission

Error recovery

- Every packet sent by the master contains the sequence number of the oldest packet which can be retransmitted
  - Slaves can detect some cases when recovery is trivially impossible
- Retransmission request suppression
- The protocol is very stupid
  - Not scalable: a few missing packets cause retransmission “storms” (slaves do not store packets with too recent sequence numbers)

Protocol messages



- Message grouping is not implemented yet… :(

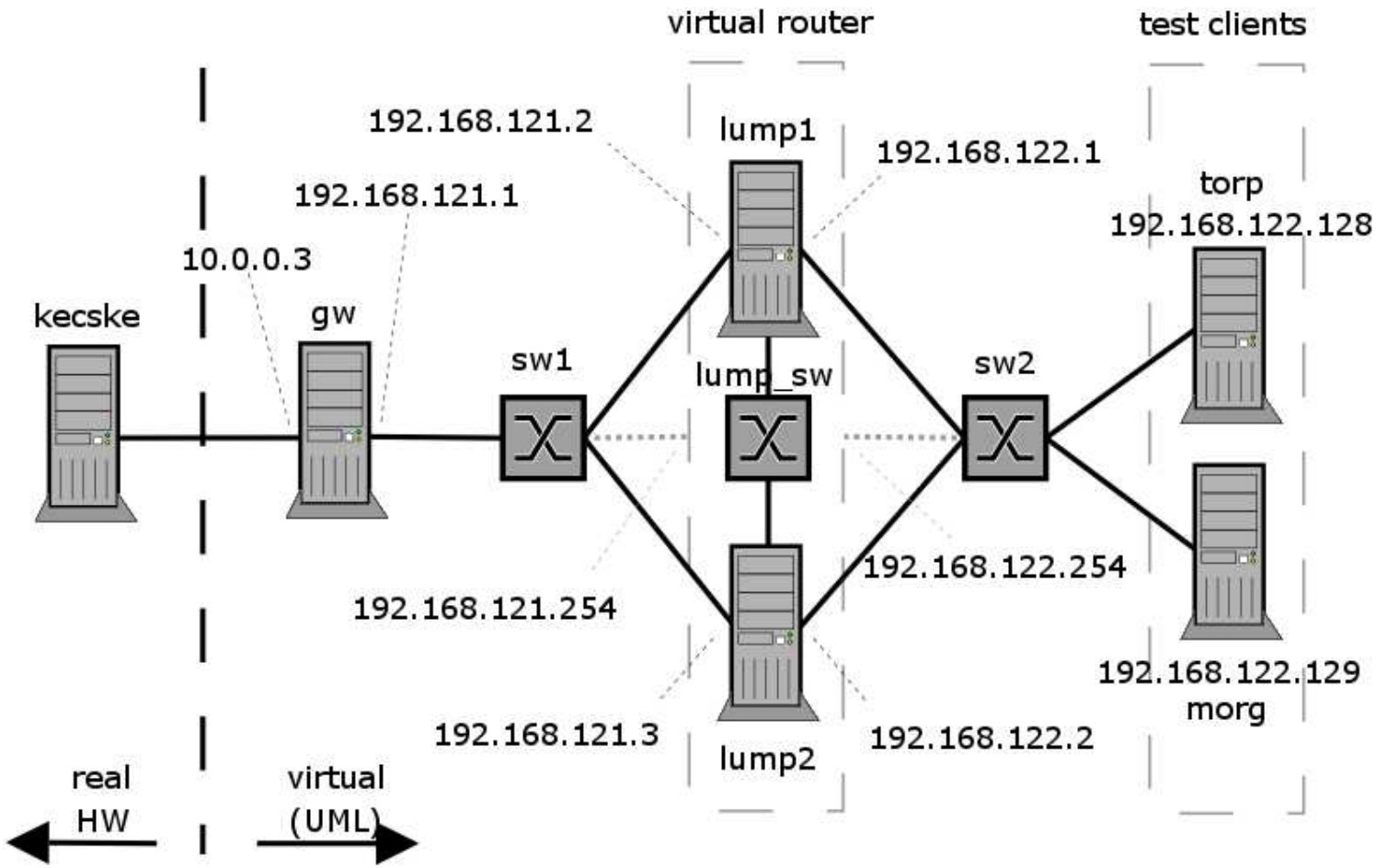
Protocol messages

- Every message is self-contained (not incremental): contains every information about a given conntrack entry or expectation
- update/delete messages for conntrack entries/expectations

Configuration

- Module parameters
  - Initial node status (master/slave)
  - Node ID
  - Replication interface name
- procfs entries
  - Change node status based on *keepalived* events

Test system


virtual router
test clients
192.168.121.2
lump1
192.168.122.1
192.168.121.1
torp
192.168.122.128
10.0.0.3
kecske
gw
sw1
lump_sw
sw2
192.168.121.254
192.168.122.254
192.168.122.129
morg
192.168.121.3
lump2
192.168.122.2
real
HW
virtual
(UML)

Future work I.

- Complete rewrite and cleanup :) (for a new and stable ctnetlink, maybe…)
- Better (scalable, more intelligent) replication protocol
  - Should it be architecture and version independent?
  - Message grouping: more messages in one packet

Future work II.

- Full resynchronization
  - Maintenance of the cluster is really hard without it
  - Retransmission of the whole conntrack table is impossible (slow, locks up master)
  - Possible solution: transmission of fake update messages (eg. for unchanged entries) when master is idle

Future work III.

- Real testing
  - Due to lack of hardware, I was unable to test the system on real HW
  - UML was used throughout the development and testing
  - In its current state it's not architecture independent

Open problems

- Is a userspace solution based on *ctnetlink* feasible? (With performance in mind)
- Load balancing instead of failover