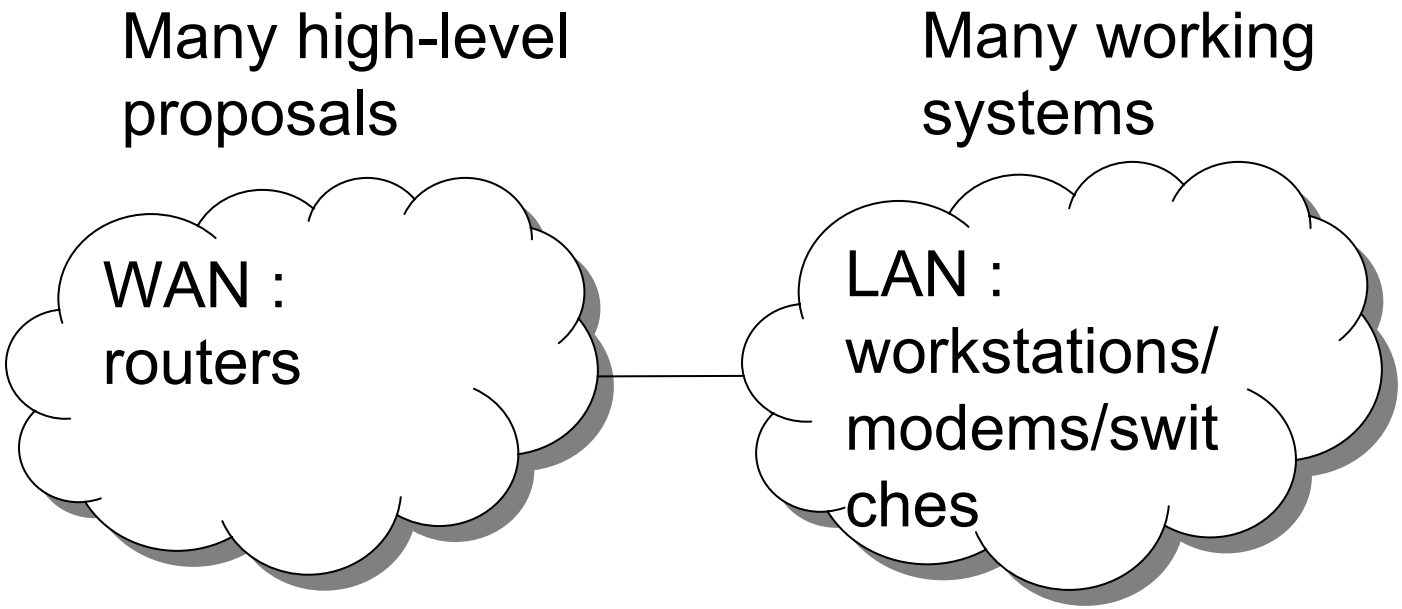
Agent based auto-configuration of OSPF networks

Visa Holopainen
visa.holopainen@tkk.fi

Problem

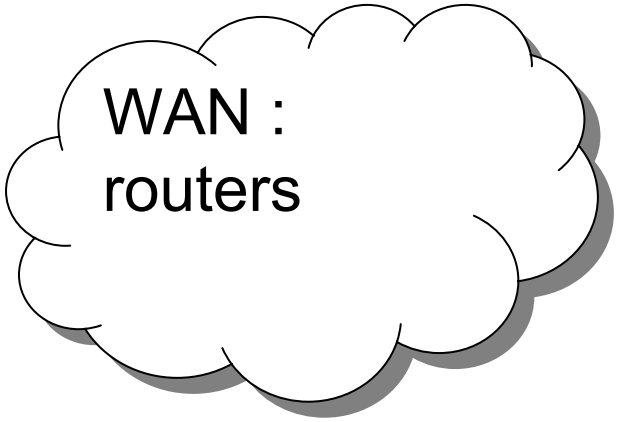
- About 30% - 50% of network outages are caused by configuration error (The Yankee Group's 2003 query)
- It would be nice to automate manual error-prone configuration tasks

Previous work in auto-configuration



Our work

A working system



What is it?

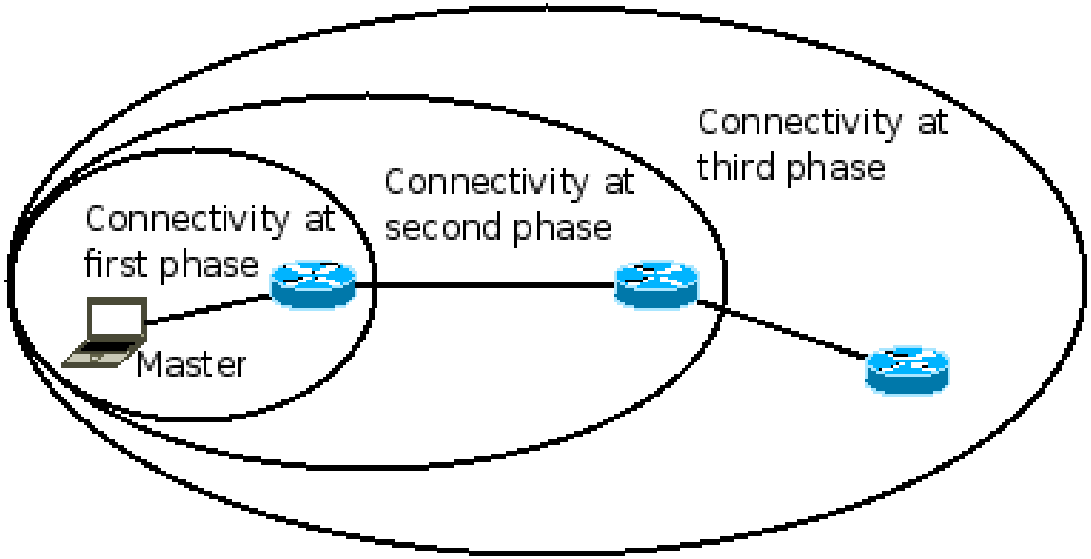
- The main idea in our system is following:
  - A person with very limited or no computer skills can take several commodity PCs, connect their Network Interface Cards (NICs), plug a specifically configured laptop to one of those PCs, and after a while the network will be a fully functional OSPF network.

Problem

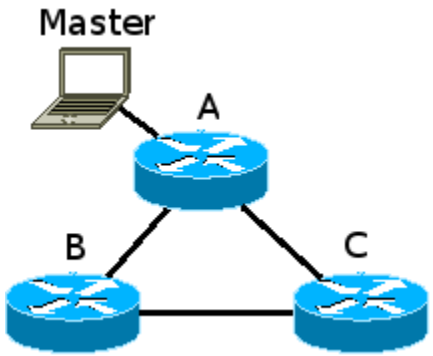
- At first there's no connectivity to routers (unless a separate management LAN is set up)
- How about auto-configuration using CD/USB?
  - possible but may be logistically complicated if routers are in different physical locations

Our solution

- ”Flood” routing protocol to the network -> incrementally increase connectivity range until all routers have been configured

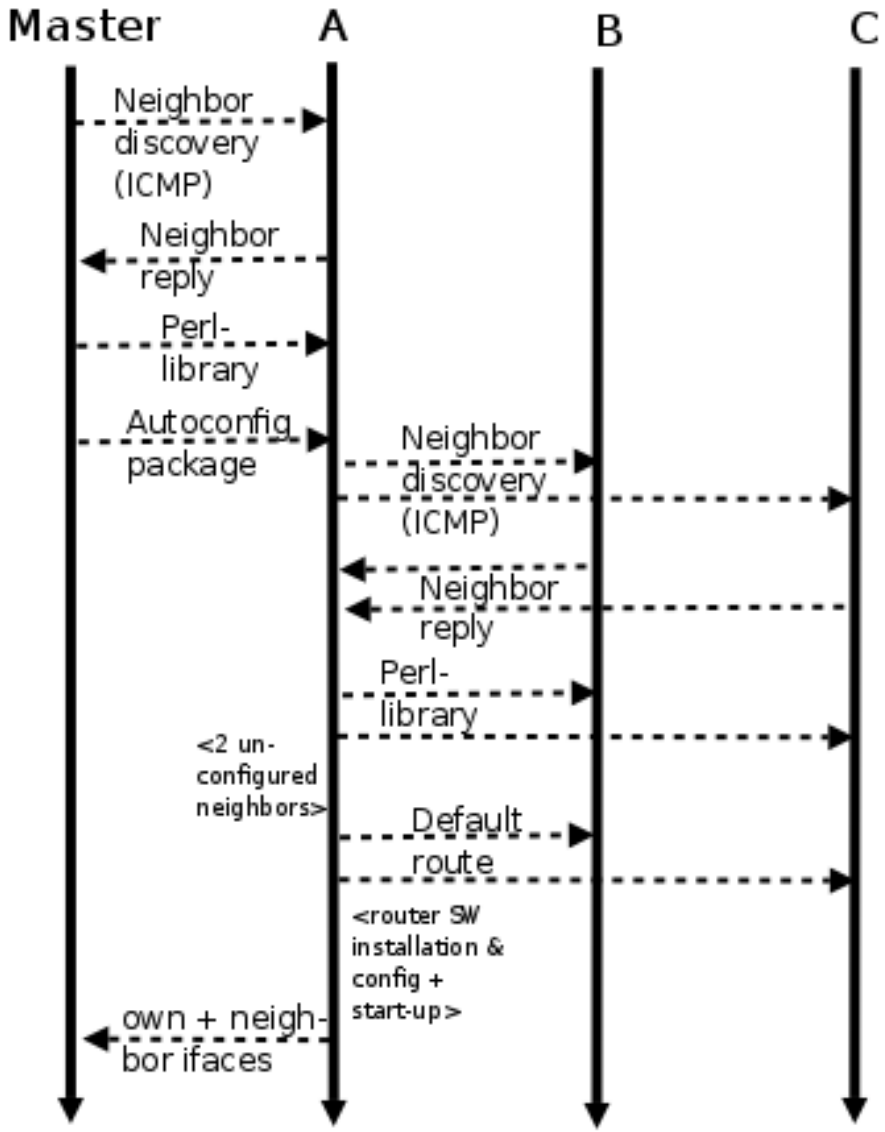


auto-configuration process (1)

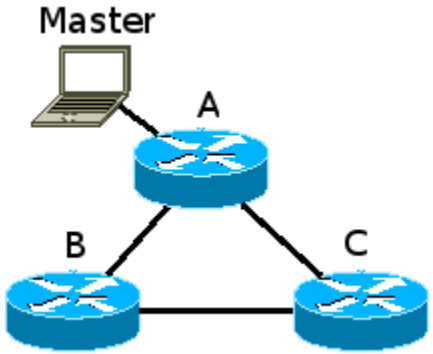


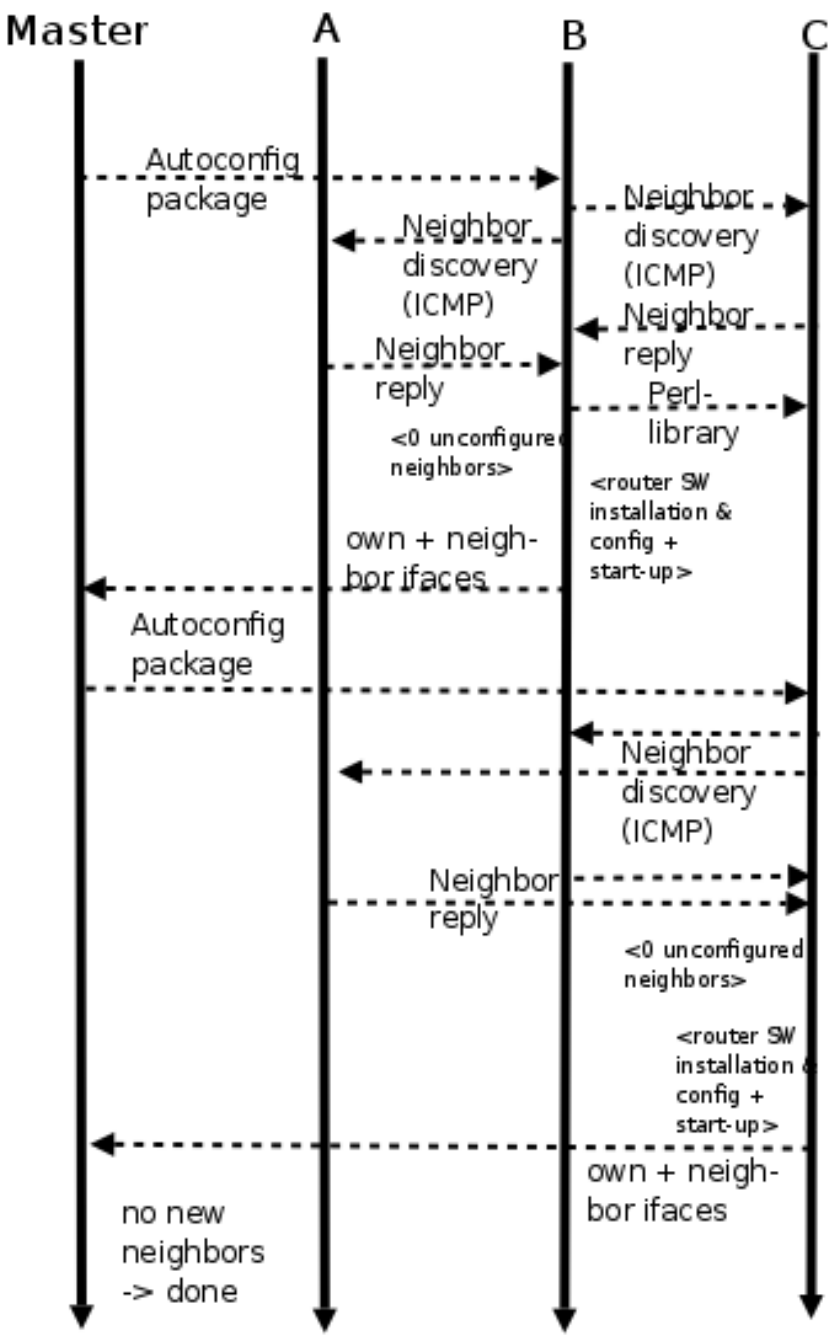
Autoconfig package contains:

1. agent Perl-script
2. routing software tarball
3. a file that contains IP addresses of interfaces that have already been configured to run OSPF (the master script maintains this file)
4. a policy file



auto-configuration process (2)





Policy file

- Plain and simple
- Suitable default values in most parts
- networking novice should be able to fill in suitable values where needed

```
default-internet-gateway-IP-address="10.0.0.1"

MD5-authentication = "yes"

permit-router-remote-configuration = "no"

link-cost-assignment-policy = "I"
# (inverse-of-link-capacity (I)

load-balancing-for-equal-cost-paths = "no"

...
```

Master

- Main purpose of master is to send agents to PCs one at a time and maintain configured and candidate IPs

```
function autoconfiguration_master {

  // initializations
  candidate_IPs = ();
  configured_IPs = ();

  candidate_IPs <- enqueue (discover_IPs());

  // first time there should be 1 candidate
  while(candidate_IPs != empty)
  {
    IP = dequeue (candidate_IPs);
    if (IP exists in configured_IPs)
      next;
    send_configured_IPs_to (IP);
    send_policy_file_to (IP);
    send_routing_software_tarball_to (IP);
    send_configuration_agent_to (IP);
    if (first_time)
      send_perl_library (IP);
    start_configuration_agent_at (IP);
    configured_IPs <- enqueue
                        (feedback_from_agent);
    candidate_IPs <- enqueue
                        (feedback_from_agent);
  }
}
```

Agent

- Agent:
  - Discovery of neighbor PCs
  - Preparation of neighbor PCs
  - Decompression, installation, configuration, and starting of routing software
  - Informing the master about own (configured) and neighboring IPs

```
function autoconfiguration_agent {

  // initializations
  stubs = ();
  own_IPs = ();
  candidate_IPs = ();
  own_IPs = discover_own_IPs();
  interfaces = discover_own_interfaces();
  configured_IPs <- read_from_file();

  for each interface in interfaces
  {
    for each IP in range of interface
    {
      if IP exists in configured_IPs
        next IP;
      if (connect_success(IP))
      {
        candidate_IPs <- IP
        prepare_candidate (IP);
      }
    }
    passive_interfaces <- interface;
  }
  send_to_master (own_IPs);
  send_to_master (candidate_IPs);
  extract_configure_and_install_routing_software
            (own_IPs, passive_interfaces);
  start_routing ();
}
```

Auto-configuration basic stuff

- Each ethernet-interface that the agent detects (using ifconfig-tool) will cause the agent to write the following line to configuration file: *network X.Y.Z.V/N area 0*. Here X.Y.Z.V/N is the IP network configured for the interface

Policy -> Configuration

- The agent writes configuration commands based on the policy file that the master sent
  - **if** (policy_entry *X*) **then** write_configuration *Y*
- Example 1: If the agent notices that one of the interfaces on the PC it is running on has the IP address of default gateway specified in policy file, then the agent writes the following to configuration file: *default-information originate*.

Policy -> Configuration examples

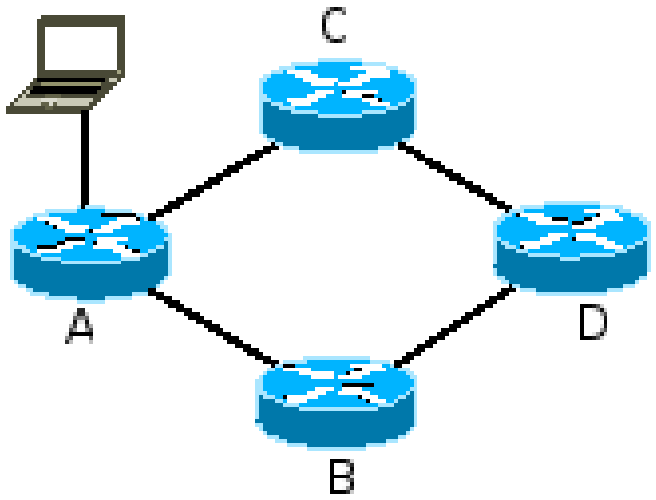
- policy:
  `permit-router-remote-configuration "no"`
  configuration:
  ```
  access-list term permit 127.0.0.1/3
  access-list term deny any
  line vty
  access-class term
  ```

- policy:
  `MD5-authentication = "yes"`
  configuration:
  under each `interface <ifname>` line:
  ```
  ip ospf authentication
  message-digest
  ip ospf message-digest-key 1 md5
  <KEY>
  ```
  under `router ospf` line:
  ```
  area <AREA> authentication
  message-digest
  ```

Features

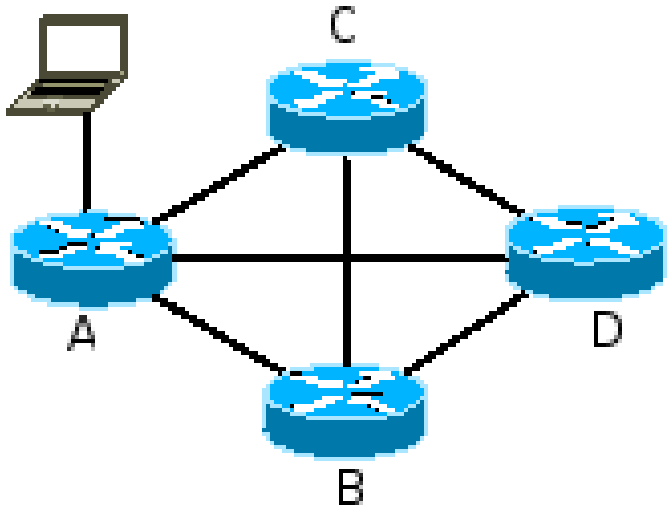
- Currently in addition to the basic routing functionality:
  - **Stubnets**: if there is no reply from an interface it is configured to be passive
  - **Broadcast networks** = switches/hubs are allowed
  - **Default Internet gateway** (not tested)
  - **ECMP load balancing** (not tested)
  - **<your suggestion>**
  - **other protocols like BGP (maybe in the future)**

Performance (1)

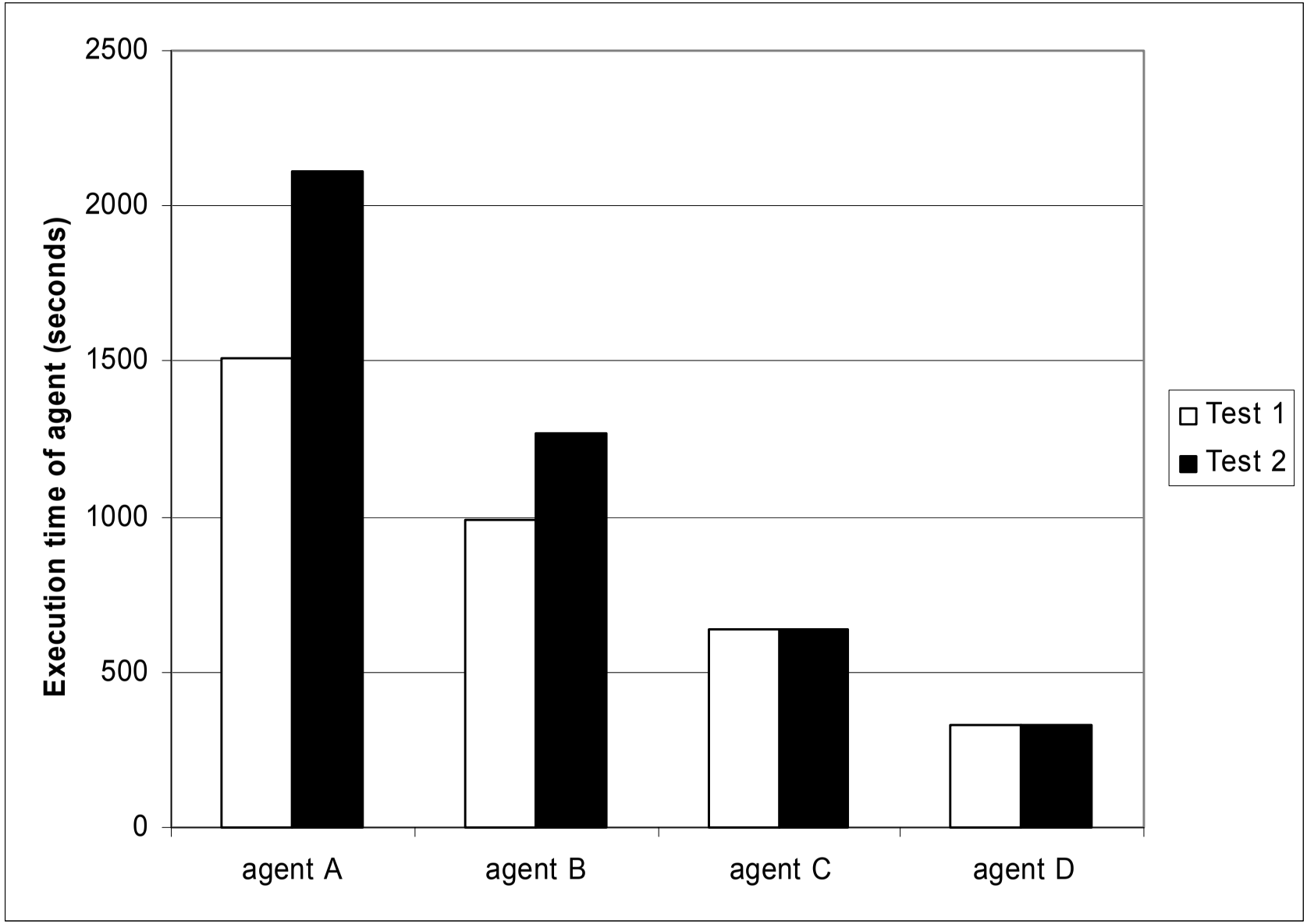


Test started	0 sec
Sent agent to A	291 sec
A done	1799 sec
Sent agent to B	1978 sec
B done	2969 sec
Sent agent to C	3147 sec
C done	3788 sec
Sent agent to D	3967 sec
D done	4298 sec

Performance (2)



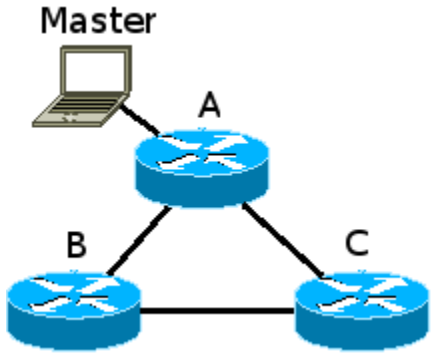
Test started	0 sec
Sent agent to A	292 sec
A done	2401 sec
Sent agent to B	2579 sec
B done	3844 sec
Sent agent to C	4022 sec
C done	4663 sec
Sent agent to D	4842 sec
D done	5172 sec


2500
2000
1500
1000
500
0
Execution time of agent (seconds)
agent A
agent B
agent C
agent D
Test 1
Test 2

Conclusions

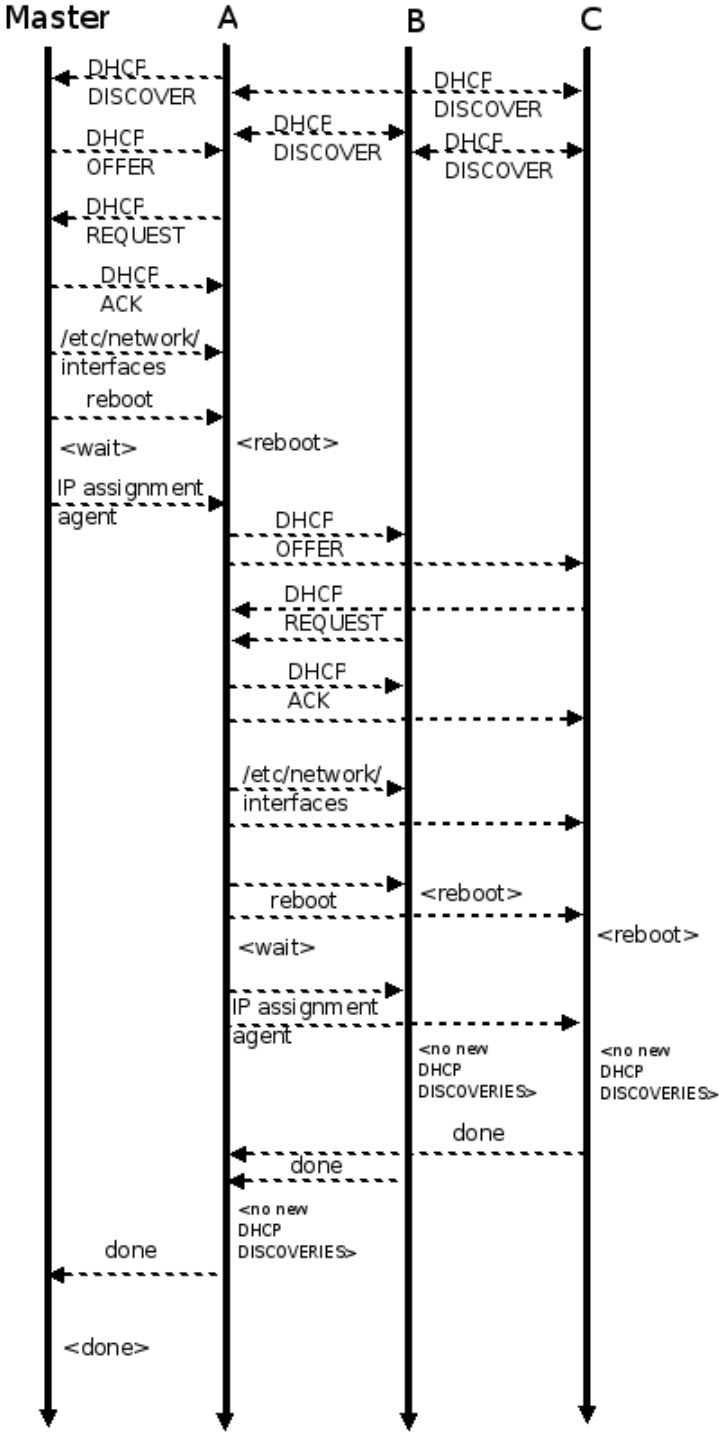
- With Perl it is easy to build a working auto-configuration system for Linux network (however, debugging takes longer)
- Net::SSH::Expect is awfully slow
  - However, if you don't know how to configure routers, an hour is a short time…
  - Also, using SSH instead of Telnet makes it possible to configure the network from a separate physical location

Future work: IP auto-assignment



Issue 1: How to get the PCs to send DHCP-discoveries at startup – need to use a script that does this?

Issue 2: A hack needed to run DHCP on multiple interfaces



Future work: OS auto-installation