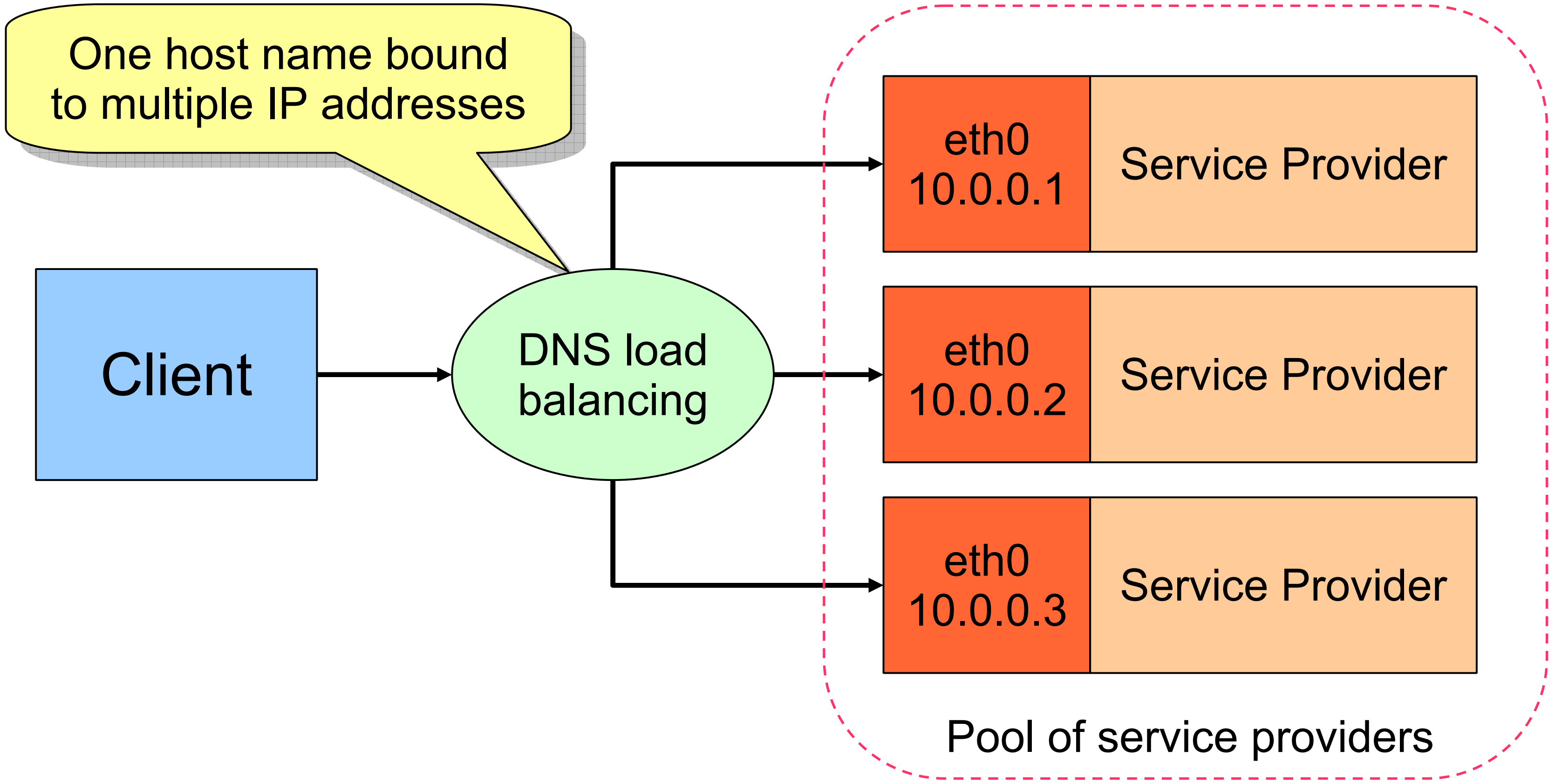
High Availability for core LCG services

Andrey Kiryanov, PNPI.

Solution for stateless services (Web server, BDII, etc.)



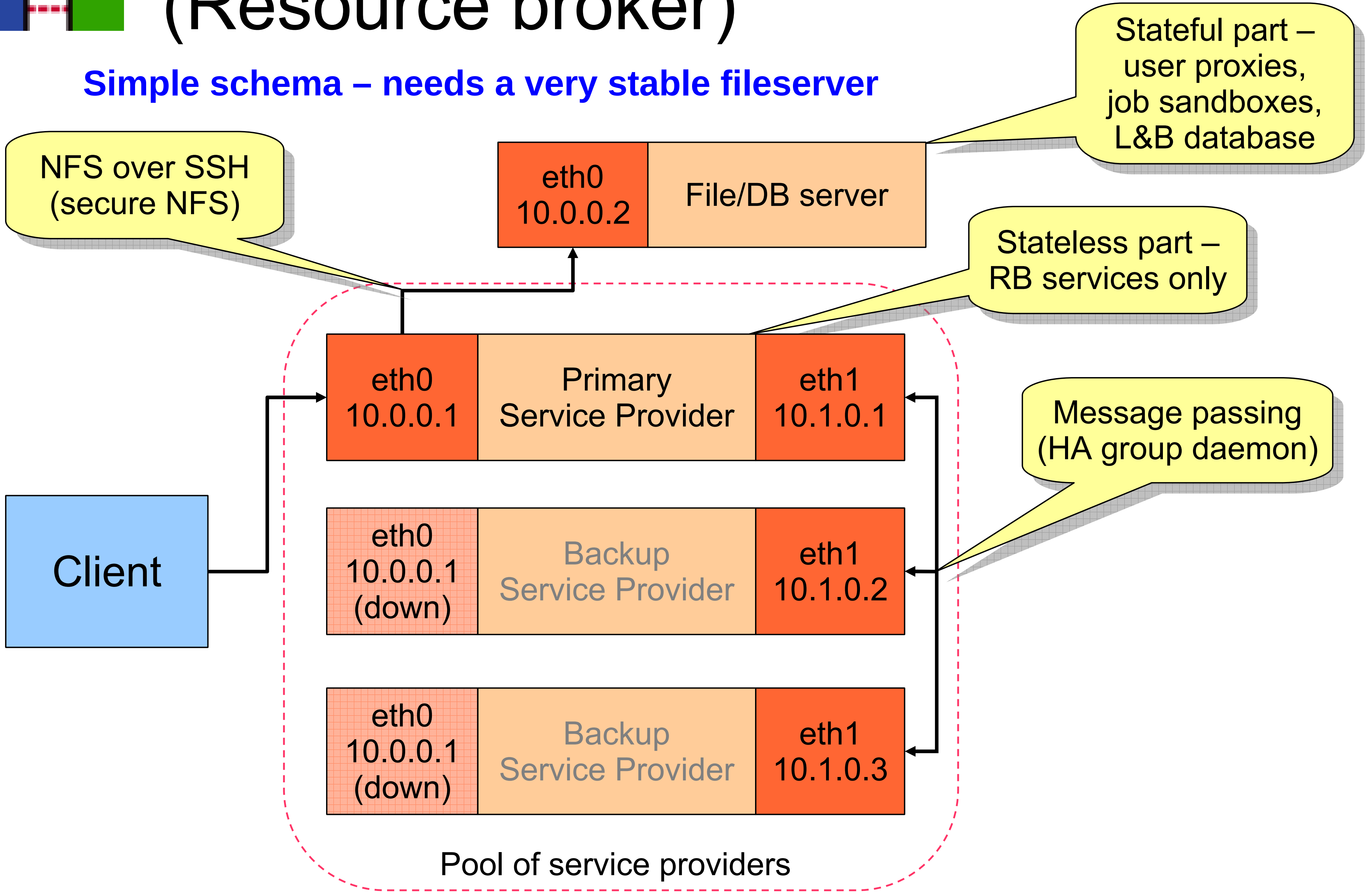
Why the DNS load balancing is not suitable for Resource broker

- User Interface makes several connections to the RB and there is no easy way to guarantee that all of those connections will go to one host.
- Job sandboxes and user proxies need to be synchronized between service providers preserving consistency.
- L&B database needs replication.

Good solution would be to separate service providers from filesystem and database servers.

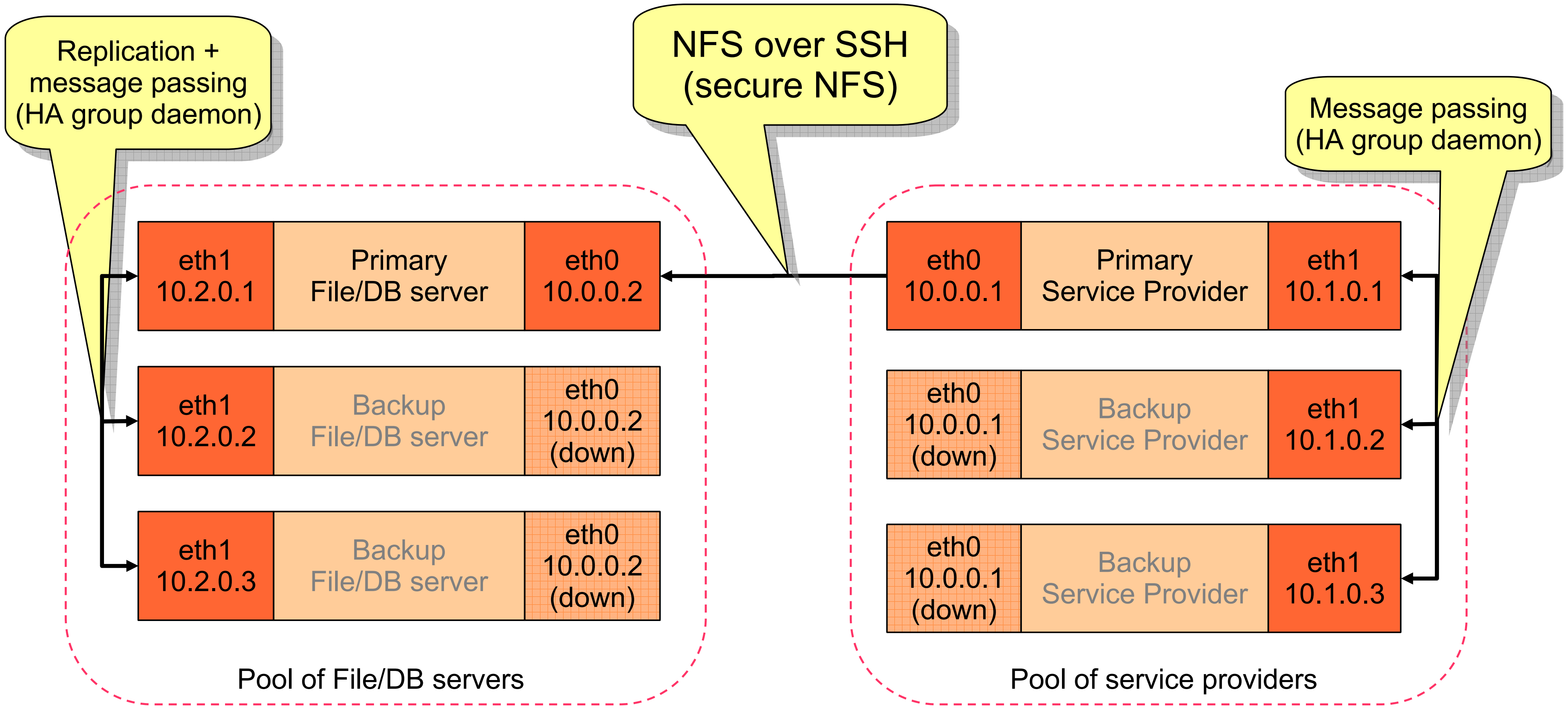
Solution for stateful services (Resource broker)

Simple schema – needs a very stable fileserver



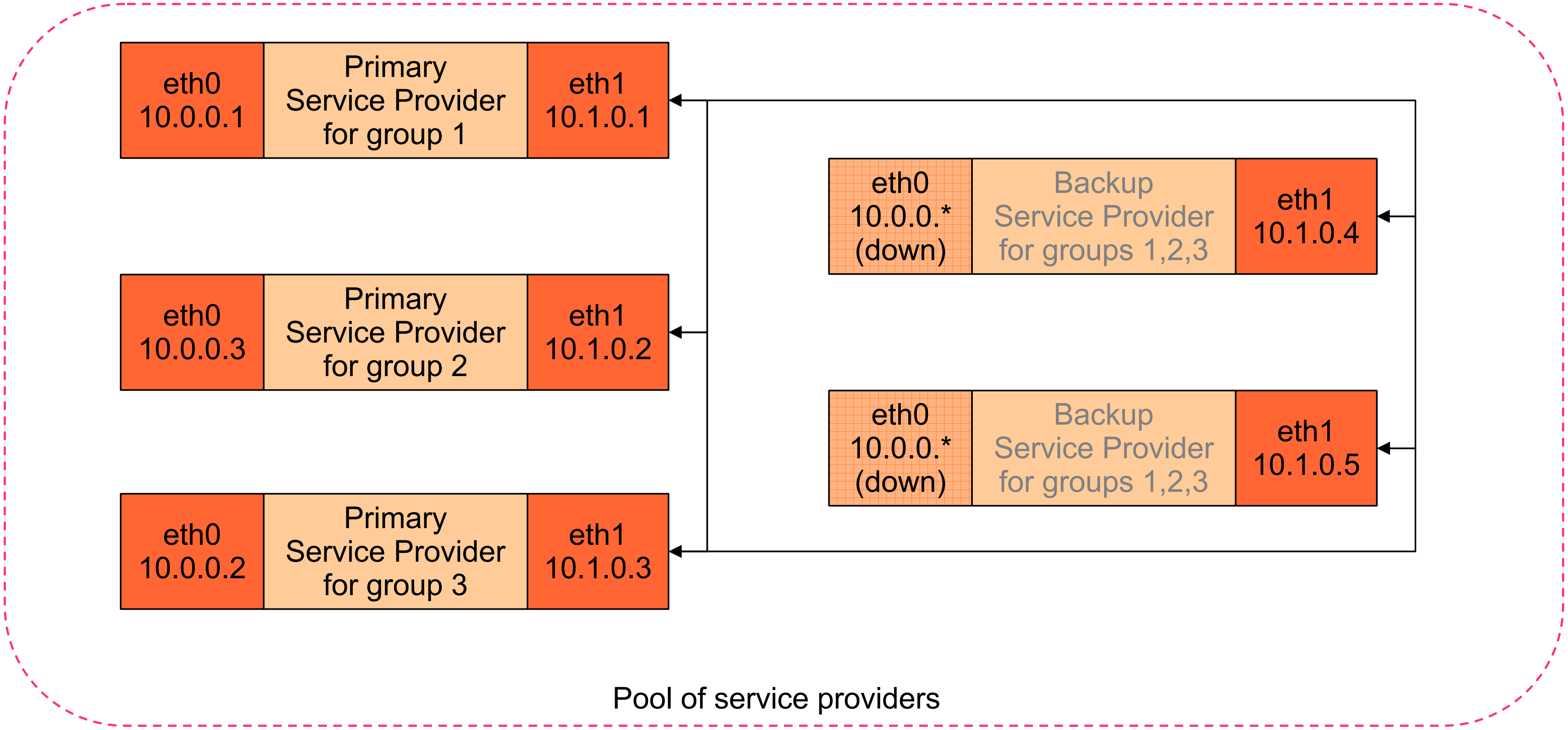
Solution for stateful services (Resource broker)

More complicated schema – can survive a failure of any box



Effective use of resources (HA multigroup)

In HA multigroup any Backup Service Provider can take over any Primary Service Provider thus reducing the number of spare boxes



eth0 10.0.0.1
Primary Service Provider for group 1
eth1 10.1.0.1
eth0 10.0.0.3
Primary Service Provider for group 2
eth1 10.1.0.2
eth0 10.0.0.2
Primary Service Provider for group 3
eth1 10.1.0.3
eth0 10.0.0.* (down)
Backup Service Provider for groups 1,2,3
eth1 10.1.0.4
eth0 10.0.0.* (down)
Backup Service Provider for groups 1,2,3
eth1 10.1.0.5
Pool of service providers

Configuring the Secure NFS

- Install the **sec_rpc** package (SRPM is available from http://cern.ch/kiryanov/sec_rpc-1.54-1.src.rpm) on all nodes. Project homepage: http://www.math.ualberta.ca/imaging/snfs/
- Configure a passwordless (public key) SSH authentication between sevice nodes and the file server(s) for **snfs** user (home directory is /etc/snfs).
- Allocate a filesystem on the file server and export it to allow access **only from local hostname**. Export to localhost is not a good idea due to security issues.
- On each service node create a connection configuration file in /etc/snfs with the following command:

```
# snfshost <remote hostname>
```

- Start the RPC proxy:

```
# rpc_psrv –b /etc/snfs/<remote hostname>
```

- Mount the remote filesystem (do not get confused – it must be mounted as if it’s on a local host):

```
# mount –o nolock,mountprog=201000,nfsprog=251000 <local
  hostname>:<remote filesystem> <local mountpoint>
```

- That’s it. Sounds silly, but the **nolock** parameter is necessary for a locking to work (needed for CondorG).

Configuring the Resource Broker

- Remove the *edg-wl-check-daemons* script from /etc/cron.d to avoid unexpected startup of services. Turn off automatic startup of resource broker services from init scripts

  ```
  # chkconfig <service> off
  ```

- Stop the resource broker services (if there are any running)
- Copy the following local folders to the mounted filesystem:

  ```
  # cp -a /var/edgwl <local mountpoint>
  # cp -a /var/myproxy <local mountpoint>
  # cp -a /opt/edg/var/spool/edg-wl-renewd <local mountpoint>
  # cp -a /opt/condor/var/condor <local mountpoint>
  ```

- Replace local folders with symlinks to corresponding locations on a mounted filesystem
- Change the value of the EDG_WL_BKSERVERD_ADDOPTS variable in the /etc/sysconfig/edg file with the following:

  ```
  EDG_WL_BKSERVERD_ADDOPTS="--mysql <user>/<password>@<hostname>:lbserver20"
  ```

- Initialize a **lbserver20** database on a remote host (you can do it just by copying /var/lib/mysql/lbserver20 to the same location on a remote host, do not forget to stop both mysql servers beforehand).

This is sufficient for the service to run in non-autonomous mode (user intervention will be necessary to manually start the backup service provider if the primary service provider fails)

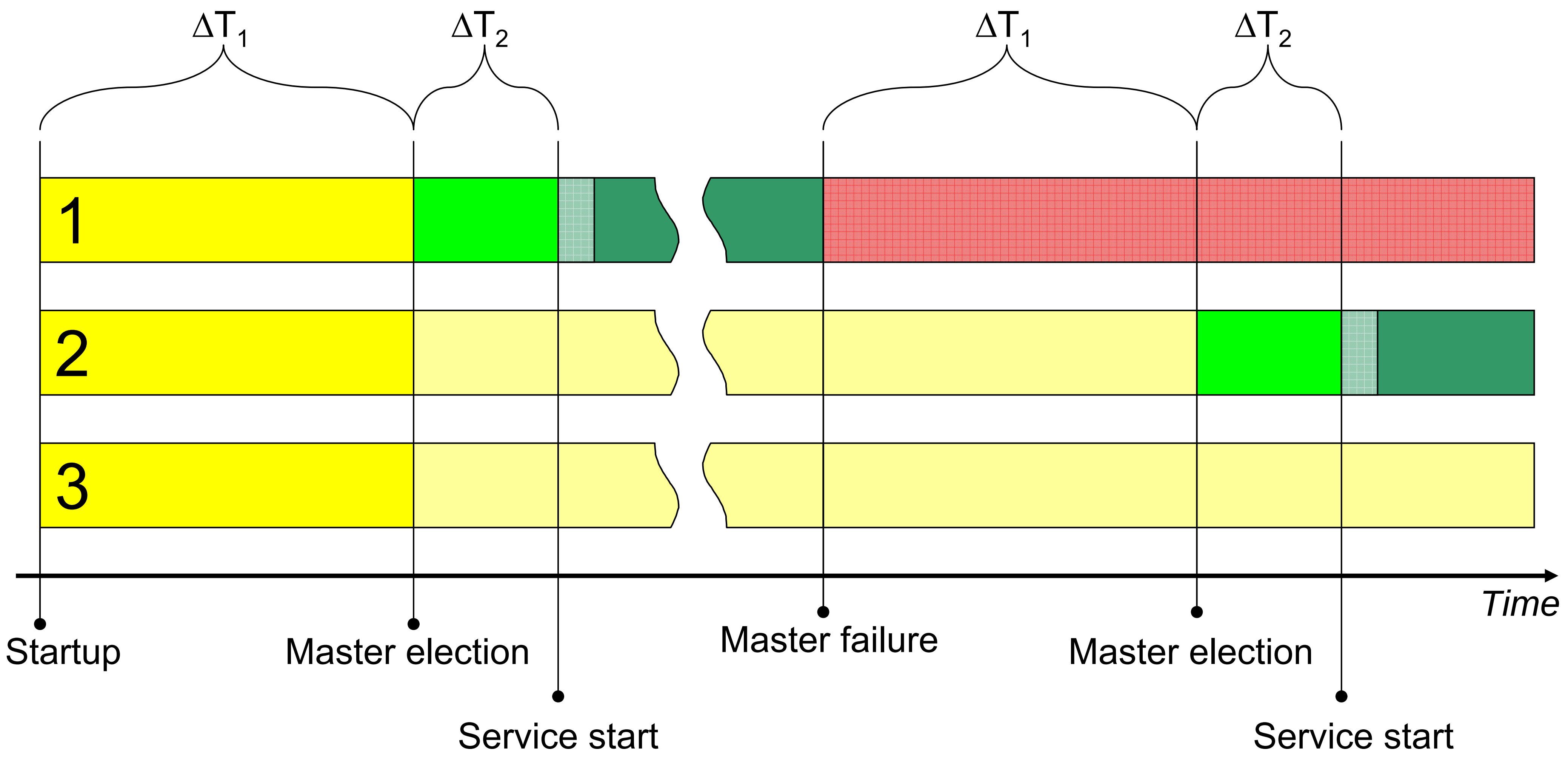
The HA group daemon (HAGD)

General concepts

- Each service provider periodically broadcasts short messages containing its unique ID, rank and state. Messages are signed with password and digest function.
- If a service provider is not broadcasting any messages for some time (ΔT_1) it’s considered dead.
- One of the service providers is a Group Master which currently provides a service. All other (backup) service providers have their services stopped.
- New group master is elected if there is no one alive. It will broadcast takeover messages for a short period of time (ΔT_2) and then start the service.

The HA group daemon (HAGD)

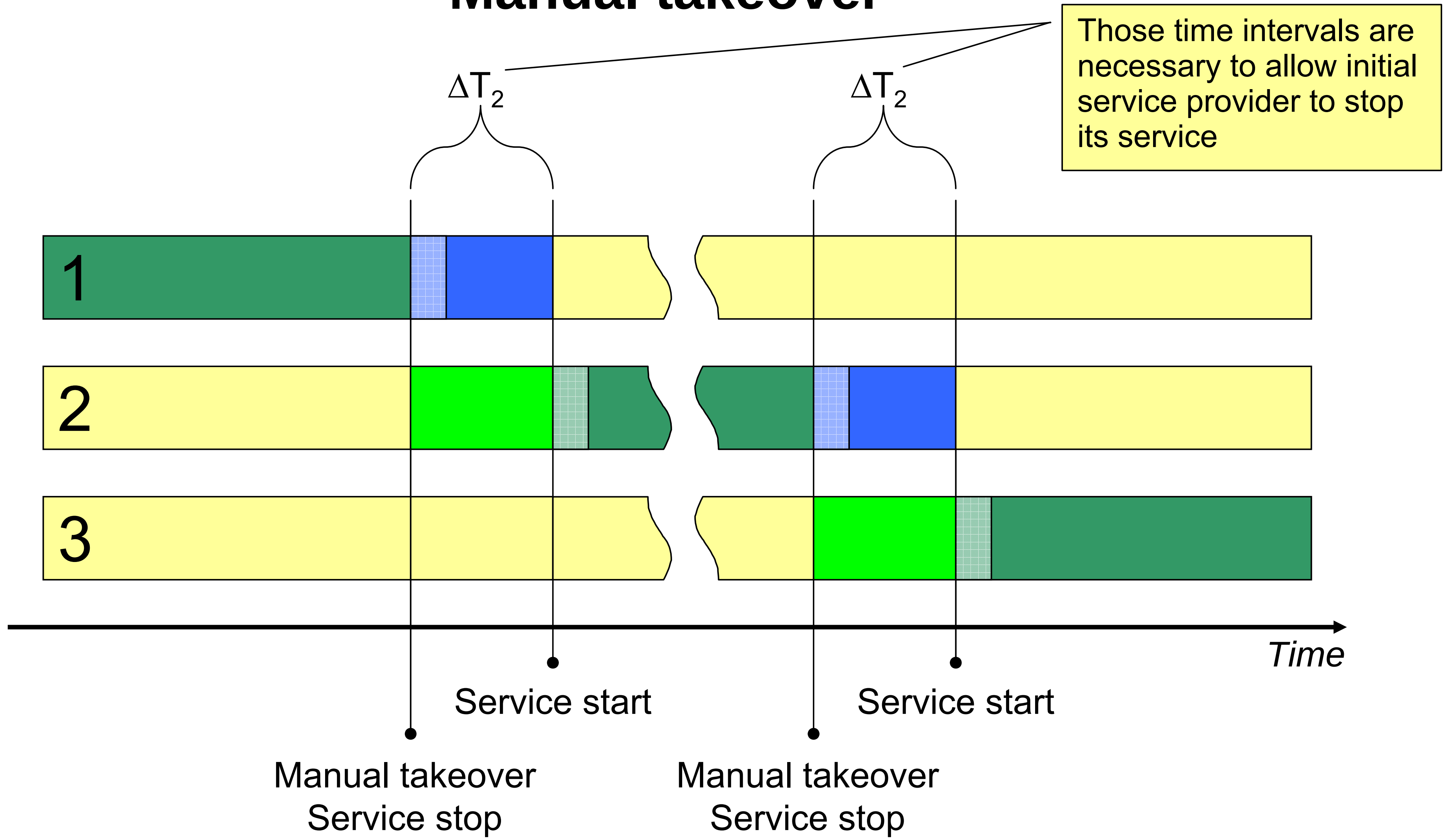
Automatic takeover



🟨 – Initial mode (no group master)	🟩 – Takeover mode (service is down)	🟥 – Failure (offline)
🟨 – Normal mode (group master is known)	🟩 – Group master mode (service is up)	▨ – Service startup

The HA group daemon (HAGD)

Manual takeover



🟦 – **Giveup mode (service is down)** 🟩 – **Takeover mode (service is down)** ▦ – **Service shutdown** ▦ – **Service startup**

The HA group daemon (HAGD)

Installation

- Install the HAGD package (RPM is available from http://cern.ch/kiryanov/hagd-2.10-lcg.noarch.rpm).
- Change the service configuration parameters in /etc/sysconfig/lcg-ha
- Additional post-install configuration of RB nodes may be mostly automated with a helper script which is installed in /opt/lcg/sbin/rb-ha-install-helper.sh by the HAGD package. It will configure the SNFS and do all the filesystem-related job. Network configuration still must be done manually.
- MyProxy may be removed from the SERVICES variable in /etc/sysconfig/lcg-ha to skip its configuration by the helper script.
- Change the HAGD configuration parameters in /opt/lcg/etc/hagd.conf
- Turn on automatic startup of the HAGD daemon and run it:

```
# /sbin/chkconfig hagd on

# /sbin/service hagd start
```

- RPM package for server-side installation helper which aids MySQL server configuration, export of shared filesystem and SNFS configuration is available from http://cern.ch/kiryanov/rb-state-server-2.0-lcg.noarch.rpm

The HA group daemon (HAGD)

Basic configuration parameters

- **port** – UDP port to broadcast messages (3333 by default).
- **group** – list of HA groups and their ranks. Service provider with highest rank in a group wins the election. Group names never appear in messages as a plain text.
- **script** – script which actually starts/stops the service. Script should accept two arguments ("stop", "start" or "status" as a first one and group name as a second) and should return proper exit code.
- **timeout** – time interval to wait for the script to finish. If the script hangs the service startup is considered unsuccessful. In such case group will elect another master.
- **debug** – log verbosity level (0 – 5, 2 by default).
- **iface** – list of network interfaces to use for broadcasting. If not specified, messages will be broadcasted from all active interfaces excluding loopback.
- **key** – password to sign outgoing messages. Should be the same for all servers in a multigroup. It never appears in messages.

The HA group daemon (HAGD)

Advanced configuration parameters

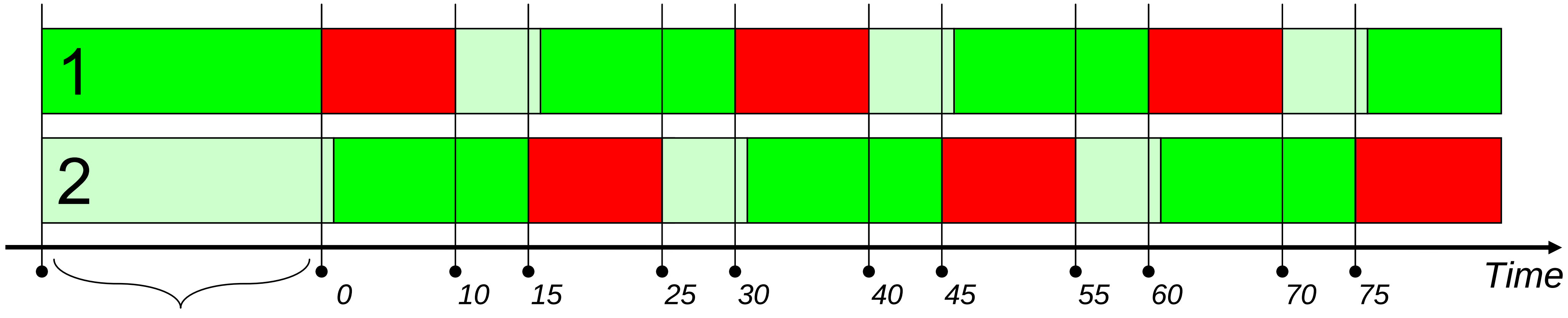
- **digest** – digest algorithm to use for message signing. Usually MD5 but may be SHA1, SHA2, etc. Corresponding Perl module must be installed.
- **tick** – number of seconds between timer ticks. Usually 1 but may be fractional if a high resolution timer module is used.
- **dtime** – if a group member is not broadcasting messages for this number of seconds it's considered dead (ΔT_1).
- **etime** – length of takeover and giveup time intervals in seconds (ΔT_2).
- **sint** – number of seconds between messages in normal mode.
- **fint** – number of seconds between messages in takeover/giveup mode.
- **id** – unique ID of a service provider. It's calculated automatically from the MAC addresses of network cards in a system but may be explicitly set to a specific value for some tricky configurations. It never appears in messages as a plain text.

To reload a configuration send a SIGHUP to a running hagd, SIGUSR2 will initiate a manual takeover to another service provider.

The HA group daemon (HAGD)

Testing

- **Configuration:**
  A “minimal testbed” of two RB/MyProxy nodes (HA group) and one CE/SE/UI/Fileserver node. 281 test jobs (just a one minute sleep + 10 KB of output data) were submitted from the UI to the first RB. Then the “crasher” cron job was activated on each RB. RBs were crashing every half of an hour for 10 minutes with a 15-minutes phase shift.



🟩 **– Primary service provider** ⬜ **– Backup service provider** 🟥 **– Crash (offline)**

- **Results:**
  271 jobs have successfully reached the “Done” status and were successfully cleared from the UI. 10 jobs have filed with “Job RetryCount Hit” error because of RB crash in the middle of data transfer. The fraction of failed jobs should be lower in real life because of less RB crashes and longer job life.

“Unexpected troubles”

- There are two daemons (*interlogd* and *jobcontroller*) which cannot be started without a controlling TTY. They die right after startup with no reason, so it’s impossible to start them from another daemon. Fortunately there is a workaround – *screen.*
- Using *screen* to get a temporary PTY makes BDII unhappy. It dies when *screen* terminates because it doesn’t daemonize properly so it receives an unhandled SIGHUP when *screen* PTY gets destructed. The only easy workaround is to patch the *bdii-update* script with one single line:

```
$SIG{HUP}='IGNORE';
```

So, for the time being it’s impossible to use HAGD with unpatched Resource broker and BDII on one host, but this is not a production configuration and these problems should be easy to fix.

Things to be done

- Fix the problems in RB (from previous slide) – it will simplify the service startup script a lot.
- Test the HAGD in a real life with a real load. Settle down the necessary network configuration issues.
- Write a good automatic installation script (YAIM).
- Maintain a documentation with installation FAQ.
- Tune the HAGD configuration parameters, specifically the timings (timeout, dtime, etime, sint, fint, tick) to guarantee a robust behavior, optimal network load and a minimal service interruption at the same time.
- Check that a newly started RB is in a good shape. RB test might be called from service startup script or by adding a periodic service state poll function to the HAGD.
- Apply this high availability technique to other less critical stateful services like CE, DPM SE, maybe some gLite services.
- Load balancing isn't the same thing as a high availability so it would be nice to merge those techniques to be able to increase performance by utilizing backup servers in a round-robin manner.