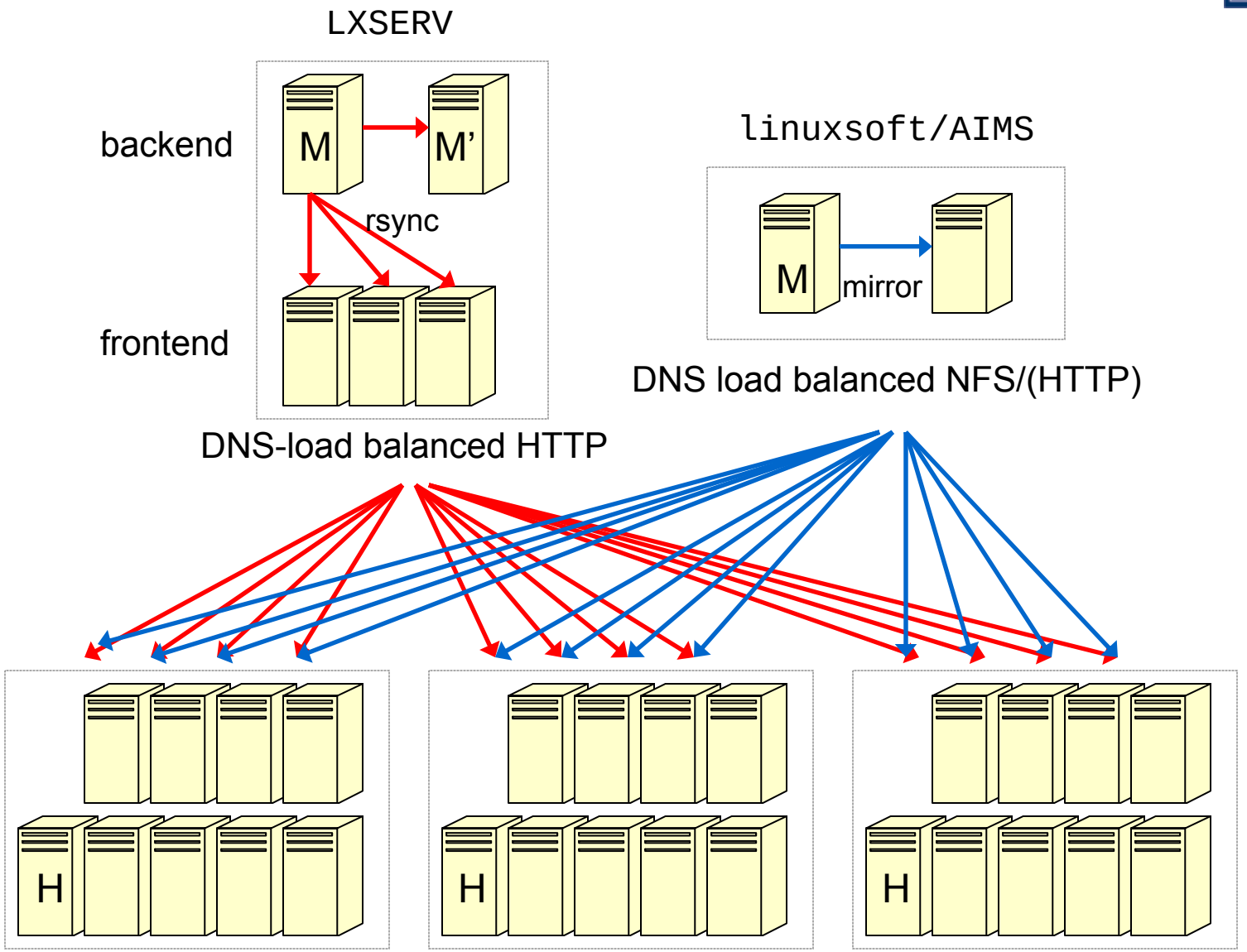
Proxy servers in CERN-CC

German Cancio, 2/3/04

The problem

- Service availability on all (quattor managed) CC nodes
  - Base installation (Anaconda server->client)
  - Software packages (SWRep->SPMA)
  - CDB configuration profiles (CDB->CCM/NCM)
  - User/password information (Regis server->client)
- How to offer a reliable, redundant and load balanced access

Current deployment architecture



Problems with current solution

- Scalability is limited
  - Bottlenecks eg. network and switches
- Efficiency
  - Server->server: All contents need to be replicated (disk size, speed)
  - Server->client: Multiplication of identical transfers
- Reliability
  - Low ratio server/clients (few servers need to cope with 1000's of clients)
  - Requires smart and quick load balancing in case a server becomes unavailable (not always there – eg. swrep.cern.ch round robin)

Proxies

- Proxy caching: concepts
  - Proxy: Intermediary between client/server interactions
  - Caching proxy: Acts as a transparent store for server objects
- Caching proxy types
  - *Transparent*: client is completely unaware of proxy (same service endpoints). Requires IP routing modifications (eg. ipchains or switches reconfig)
  - *Forward* (or 'cient-side'): client makes server requests via specified proxy server. Caching typically done on (or near to) the client, to reduce outgoing connections
  - *Reverse* (or 'server-side'): client application talks to front-end server(s), which forwards requests to back-end server(s).
- Reverse proxy easiest to set up, as only requires client reconfiguration

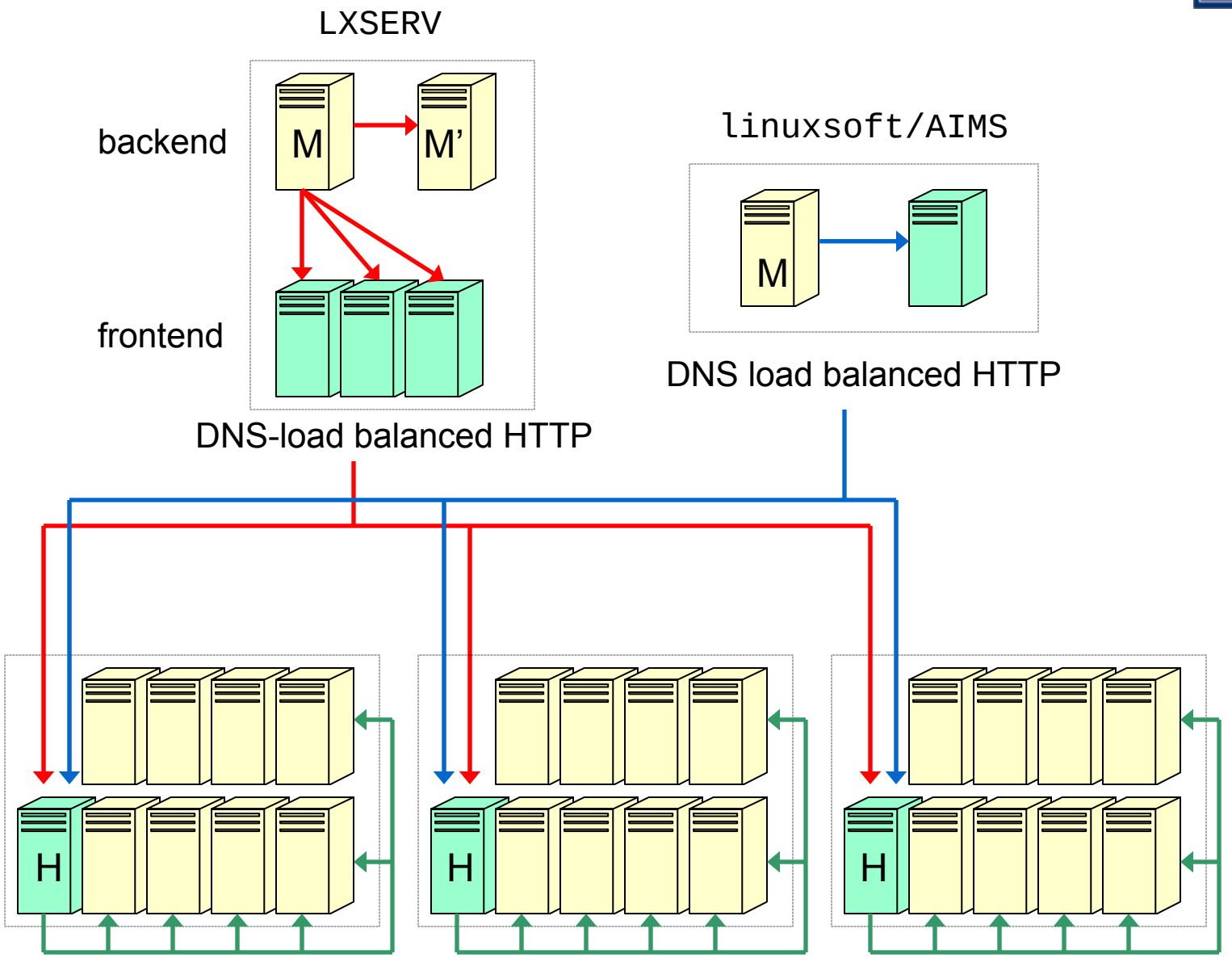
Proxies (II)

- Proxy hierarchies
  - Possible to have chains of proxies, which can be of different type
- Protocol types
  - Protocols eg. HTTP(S), FTP used within stateless services
- Proxy implementations:
  - Apache (via plug-in modules (mod_proxy/mod_rewrite/mod_expire)
  - Squid
- Apache used for tests
  - + standard, and reliable
  - clear and flexible configuration
  - no functionality duplication (can be used as proxy and non-proxy)

Data caching

- Invariant objects
  - Same object ID (file name) -> same contents
  - No expiry (excepting object deletion). Lifetime = ∞
  - Example: Software packages (RPM's), Linux base install images (in principle…)
- Dynamic objects
  - The same object may change over time
  - Expiry lifetime can be < ∞, or even 0
  - Requires server revalidation check after expiry
  - Example: CDB XML profiles, passwd files
  - Cannot be cached: cgi scripts, ASP pages
- Objets on a proxy can be forced to expire independently of their remaining lifetime
  - Useful for regular garbage cleanup (..and unexpected updates)
  - Can be done per object type or location

Proxy architecture



Proxy support

- quattor client/servers are by design **reverse** proxy compatible
  - Stateless, no server-based processing or queries, in order to work with any proxy/replication system
  - Server location is configurable per client
- Anaconda (Linux installer) as well (using HTTP installs)
  - Per-node KS file contains server location
- SPMA enhanced support (since v1.9.1):
  - Multiple proxy servers – uses the first one available, if none it reverts to the original package SW repository locations
  - *Forward* proxies (via delegation to syspackager)
  - Configurable via CDB
- Other applications eg. regis client, GPG keypairs compatible as well (check if location is *configurable*)

apache configuration

- Apache set up as reverse proxy on LXSERV front end and head nodes.
- Most important settings:
  - Load and enable `mod_proxy` (`libproxy.so`)
  - Enable cache (/var/cache/httpd), size ~ 6GB
  - Cache garbage collection runs every 4h
  - Cache max expiry: 24h
  - Set father server for each proxy directory (/xml,/swrep, /redhat, /regis,..)
  - Force complete file download in case of partial requests (eg. rpmt/SPMA asking for rpm header information)
- LXSERV master (not a proxy server!)
  - Set expiry type (via mod_expire: ExpiresActive and ExpiresDefault rules) for dynamic objects (/regis, /encrypted/sensitive-files, /xml). Set to 'now', but could be fine-tuned
- Enhanced verbosity:
  - Add `X-Cache` header contents into HTTPD log file (cache operations, HIT and MISS)
  - Enable server-status pages
- No changes done to linuxsoft (to be discussed with ADC). No merge SWRep+Linuxsoft needed anylonger.

Current experience

- LXSERV front-end nodes: proxy in production since ~ 2w, no problems experienced
  - Lxserv01, 02, 03 pointing to lxservb01
  - SWRep, XML profiles, regis
  - Rsync disabled:
    - speedup of 50% on complete CDB recompilation
    - SWRep nightly upload speedup (negligible)
  - Lxservb02 not in proxy mode since not everything is backed up on lxservb01
- Head nodes: tests on lxc1m603
  - ccconf DNS alias (XML), lxservb01 and lxserv01 (SWRep), linuxsoft (base installation)
  - Complete reinstallation of test node only using lxc1m603 as proxy
  - Lxplus001,lxb0001 using lxc1m603 as proxy since 2w
- Server interruptions in the proxy chain:
  - If cached and invariant -> OK
  - (dynamic AND expired) OR not cached -> proxy error. Not problematic in case of CCM requests, as local cache exists

Remaining issues, next steps

- Making remaining services head-node aware in terms of cfg
  - Regis
  - NCM component for CCM configuration
  - KSGenerator
- Apache NCM configuration component
- Possible extension: adding failover capabilities to other services than SPMA …
  - Improve SPMA failover procedure (currently using `ping`)
  - Implement SPMA retry
- .. or provide DNS aliases to redirect head node requests
- Entering client->head node info into CDB, including failovers
  - Requires head nodes to be known
  - Will require adapting CDB schema to accommodate other head node info into CDB
- Operator alarms and procedures for head nodes
  - Single-big-point of failure -> many-small-point-of failures
  - Future extensions, eg. dynamic experiment software distribution