

System Configuration

Paul Anderson
dcspaul@ed.ac.uk



[http://homepages.inf.ed.ac.uk/dcspaul/
publications/leuven-2008-talk.pdf](http://homepages.inf.ed.ac.uk/dcspaul/publications/leuven-2008-talk.pdf)

System Configuration

- What is “system configuration” ?
- Automating configuration
- A practical tool - LCFG
- Research challenges

What is “System Configuration” ?

**SERVICE
SPECIFICATION**



What is involved ?

- How do we construct a system to provide the required service:
 - *what software goes in each machine ?*
 - *what goes in all the “configuration files” ?*
- Implement it:
 - *load the software*
 - *create the configuration files*
- Maintain it when things change:
 - *either the requirements*
 - *or the system (failures)*

Why is this hard (I) ?

- It is large problem:
 - $O(1000)$ machines \times $O(100-1000)$ parameters
- The relationships between all the internal services are complex:
 - DHCP, DNS, LDAP, Kerberos, IMAP, SMTP, AFS, NFS ...
- Things change all the time
 - failures are “normal”
- Mistakes are bad:
 - immediate service failures, or (perhaps worse) ...
 - hidden problems (security holes)
 - they are a very significant, real cost

Why is this hard (2) ?

- Many different people are likely to be involved
 - *they will have “overlapping concerns”*
 - *they will have different levels of skill*
 - *they may work for different organisations*
- Deployment is a distributed process
 - *it involves latency and uncertainty*
 - *asymptotic configuration*
- Existing tools are primitive
 - *designed for manual use at a “low-level”*
 - *no support for “top-down” configuration*

An example - a new web server

- ① **Configure the supporting infrastructure**
 - *create a DNS entry*
 - *create a DHCP entry*
 - *create holes in the firewall*
 - *create and sign SSL certificate*
 - *add to backup system*
 - *etc ...*
- ② **Configure the server**
 - *partition the disks, load the software etc ..*
 - *set up the services:*
 - *dns, networking, timeservice, apache, authorisation etc ..*

A few typical problems

- The relationships have to be maintained manually
 - *the machine running the web server & the firewall*
- How do we know the configuration is “correct”?
 - *there is no explicit representation*
- How do we configure a replacement?
 - *some “aspects” of the replacement machine may be different (eg. disk layout, network devices)*
- different people may be responsible for different services
 - *how do we prevent them conflicting*

Automating configuration

**SERVICE
SPECIFICATION**



The motivation

- 👁 Efficiency
 - *obviously*
- 👁 Manage the complexity
- 👁 Have confidence in the correctness
 - *what happens if one machine in 1 000 node cluster has the wrong version of a maths library?*
- 👁 Security
 - *are we confident there are no insecure dependencies?*
- 👁 Reliability
 - *autonomic reconfiguration*

The goal

- A top-down process
 - we care about “fabric-level” policies - not the details
 - eg. “I want 2 DHCP servers on every subnet”
 - refining requirements into detailed specifications
- We cannot (yet) automate at this level
 - but we want a consistent framework to integrate manual and automatic processes
- There are some analogies with programming
 - high-level languages to replace machine code
 - formal techniques & sound engineering practice

Levels of configuration

1. “Copy this disk image onto these machines”
2. “Put these files on these machines”
3. “Put this line in sendmail.cf on this machine”
4. “Configure machine X as a mail server”
5. “Configure machine X as a mail server for this cluster” (and the clients to match)
6. “Configure any suitable machine as a mail server for this cluster” (and the clients to match)
7. Configure enough mail servers to guarantee an SMTP response time of X seconds

Ideal requirements

- 👁 Declarative specifications
- 👁 Support for devolved management
 - *different skill levels, conflict resolution, access control*
- 👁 Support for autonomies
 - *at multiple levels - host & network*
- 👁 Robustness
 - *distributed processing and deployment*
- 👁 Standards and interoperable tools
- 👁 Provably correct implementations of specifications

The reality

- Low-level tools only
 - *difficult to interface to higher-level tools*
- Incompatible tools with partial solutions
 - *both vendor tools & open source tools*
- Non-declarative specifications
 - *“copy this file”, “change these parameters”*
- Different communities
 - *network, application (Grid) and system configuration*
- Moving to a more structured configuration technology is big jump for many sites
 - *a perceived steep learning curve & social issues*

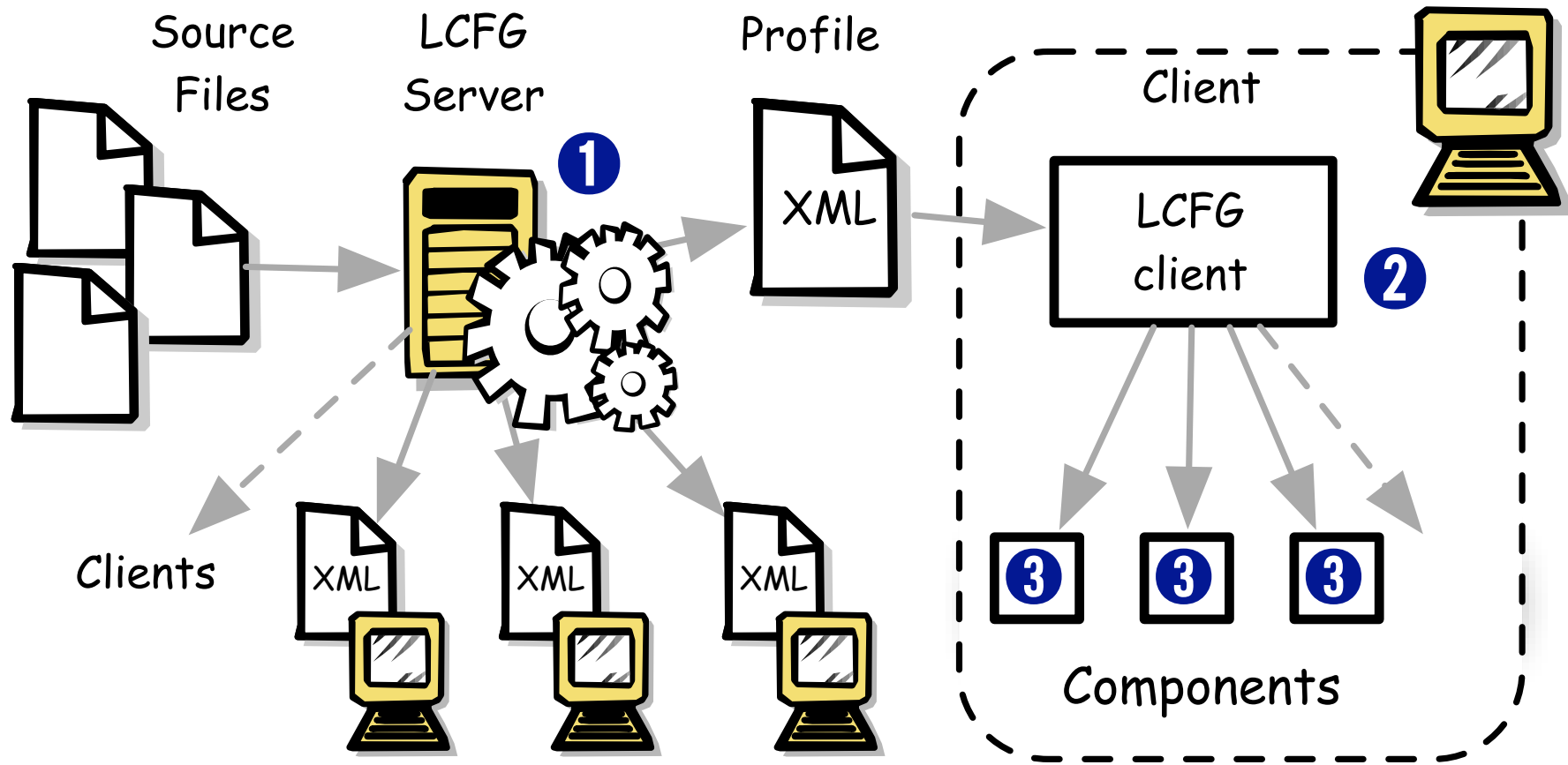


LARGE SCALE UNIX CONFIGURATION SYSTEM

<http://www.lcfg.org>

- Developed around 1994 in Computer Science at Edinburgh University
 - *under regular development since*
 - *used heavily at Edinburgh for production systems*
 - *the basis for many research projects*
- Distributed under the GPL
 - *but not widely adopted elsewhere*
- Currently very stable and being promoted more

The LCFG architecture



Abstracting the details

snmpd : localhost 129.215.216. 129.215.58.

stunnel : localhost .inf.ed.ac.uk .dcs.ed.ac.uk

slapd : 127.0.0.1 R\$* < @ [IPv6 : \$+] > \$* \$: \$2 \$| \$1 < @@ [\$(dequote \$2 \$)] > \$3
mark IPv6 addr

R\$+ \$| \$* < @@ \$=w > \$* \$: \$2 < @ \$j . > \$4 self-literal

M R\$+ \$| \$* < @@ [\$+] > \$* @\$ \$2 < @ [IPv6 : \$1] > \$4 canon IP addr

Modeline "1024x768" 65 1024 1032 1176 1344 768 771 777 806 -hsync -
vsync

- 👁 We need to do more than just manipulate configuration files
- 👁 We abstract the **relevant** parts of the contents
- 👁 Specify them in a uniform way

Some *LCFG* features

- Clear declarative specifications
 - *simple, uniform language*
 - *global validation & generation possible*
- Composing requirements from different people
 - *support for “aspect composition”*
 - *usable at different levels*
- Support for relationships
 - *“spanning maps”*
- Modular “components”
 - *easy for people to write and share*
- Support for “prescriptive configuration”

Future developments

👁 Production system

- *documentation & education*
 - *a new SAGE booklet is being published*
 - *we have “quickstart” tutorials & videos*
- *refactoring of production code*
- *better portability & support for more platforms*

👁 Research

- *fundamental issues*
- *a new architecture in the long term*

Beyond LCFG - Some research challenges

- 👁 Languages and specifications
- 👁 Distributed processing
- 👁 Planning
- 👁 Autonomics
- 👁 Human factors

Languages & specifications

- System configuration languages are not programming languages
 - *they describe data, not computations*
 - *but there are some analogies - eg. “aspects”*
- Aspect composition
 - *order-independence in sources*
 - *constraints for composition*
 - *provenance*
- Modularity and scope

Constraints

The language forces explicit values to be specified:

Aspect A

Use server Y

Aspect B

Use server X

This conflict is irreconcilable without human intervention because we don't know the intention

The user really only wants to say ...

Aspect A

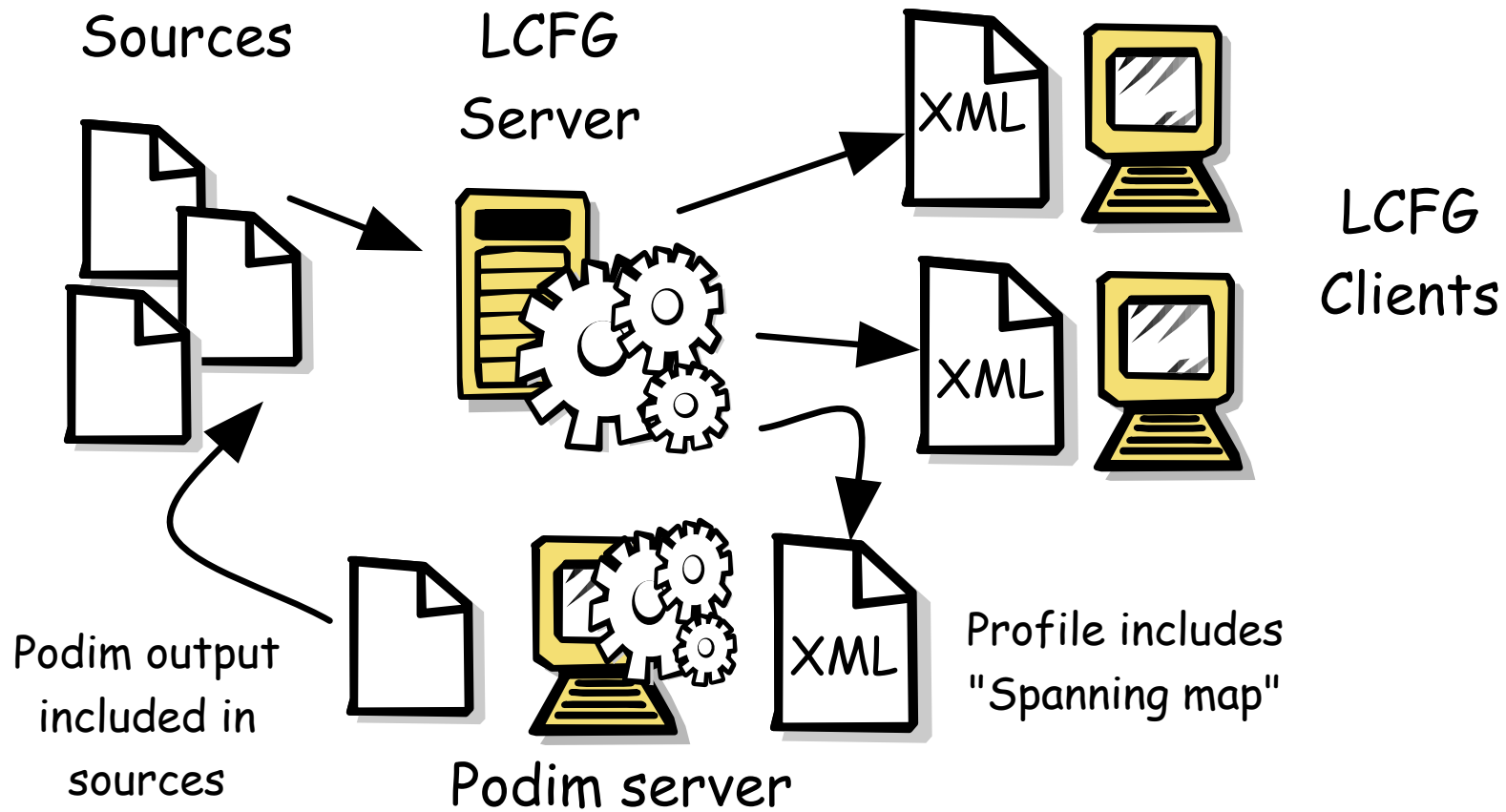
Use any server on my Ethernet segment

Aspect B

Use one of the servers X, Y or Z

These constraints can be satisfied by using Y (assuming Y is on the right segment)

LCFG/Podim



- A Paper with Thomas Delaet (ICN 2008)
 - Solves “2 DHCP servers on each subnet”

Distributed Processing

- Centralised compilation has some disadvantages ..
 - *it may be too slow*
 - *there is a single-point of failure*
 - *the source of the information may be distributed*
- But, distributed configuration “compilation” is hard
 - *the network is unreliable*
 - *communication overhead is likely to be high*
 - *validation before deployment is difficult*
 - *provenance is difficult to establish*
 - *security is harder*

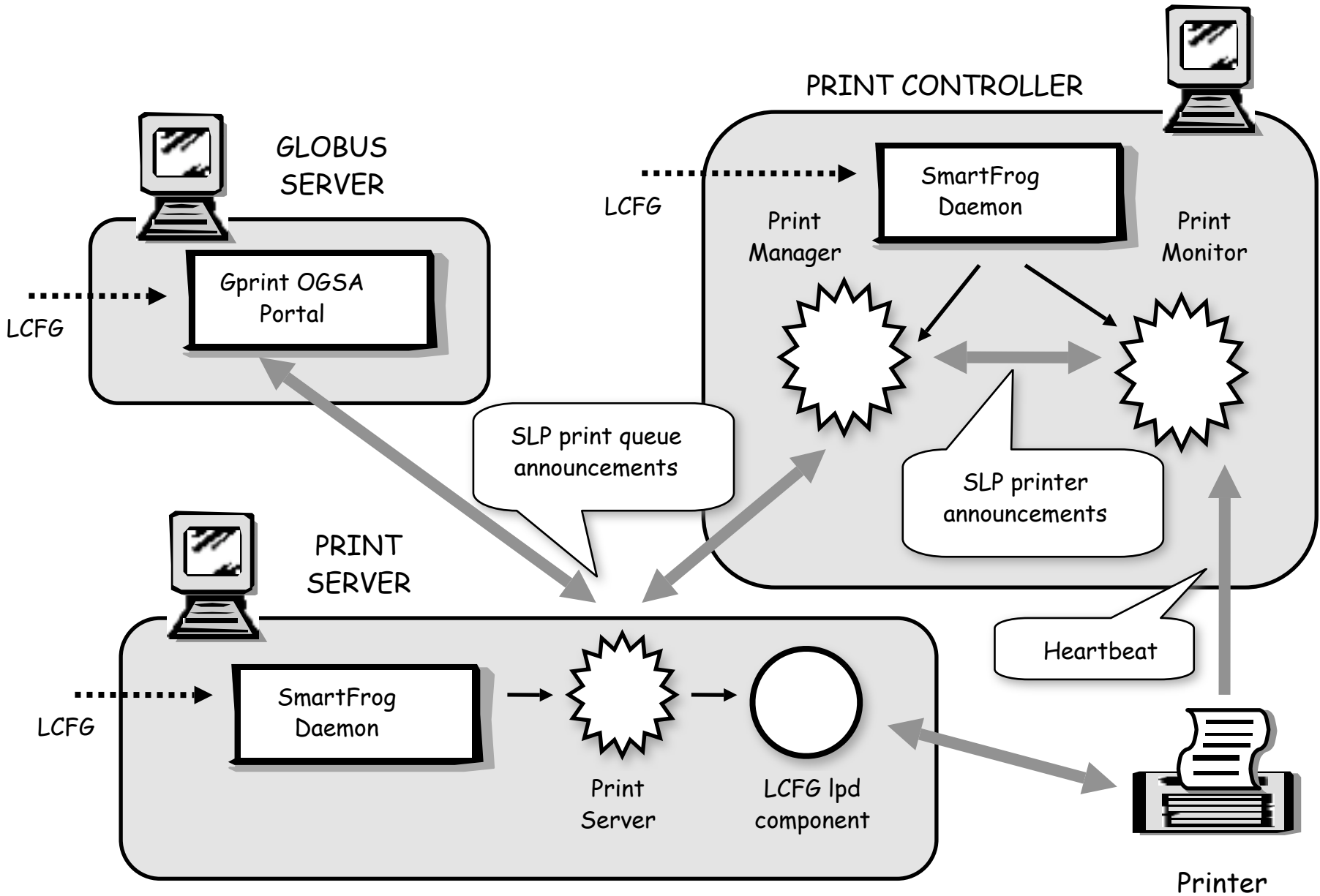
Planning

- Having computed a new configuration, planning and executing the sequence of changes to achieve it is difficult
 - *certain intermediate stages may be invalid*
 - *certain changes may “block” indefinitely*
 - *feedback on the current state may not be reliable*
- For example, to change a server from A to B
 - *deploy server B*
 - *change all clients to point to B instead of A*
 - *undeploy server A*

Autonomics

- Autonomics requires completely unattended, automatic configuration
- It needs to happen at multiple levels
 - *configure a replacement for a broken server*
 - *restart a failed daemon with different parameters*
- Constraints are important here as well
 - *over-constrained autonomic systems are no use*
 - *we need to compose human-generated constraints with system-generated constraints*

LCFG/SmartFrog



Human Factors

- System administrators are familiar with manual processes
 - *a new “mindset” is needed to accept automatic configuration*
- There are no standards for tools
 - *a variety of changing tools to learn*
- A high degree of trust is needed
 - *failures have serious consequences*
- Automatic systems need to explain their reasons

System Configuration

Paul Anderson
dcspaul@ed.ac.uk



[http://homepages.inf.ed.ac.uk/dcspaul/
publications/leuven-2008-talk.pdf](http://homepages.inf.ed.ac.uk/dcspaul/publications/leuven-2008-talk.pdf)