

Automated Planning for Configuration Changes

Herry

(h.herry@sms.ed.ac.uk)

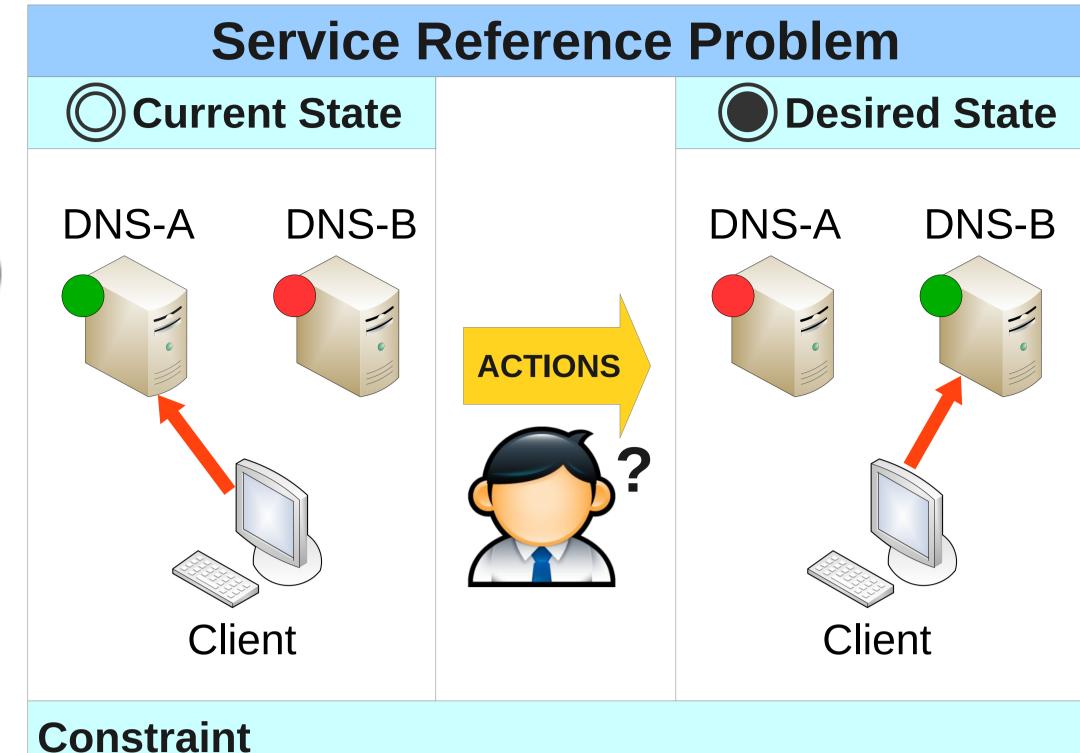
Paul Anderson

(dcspaul@ed.ac.uk)

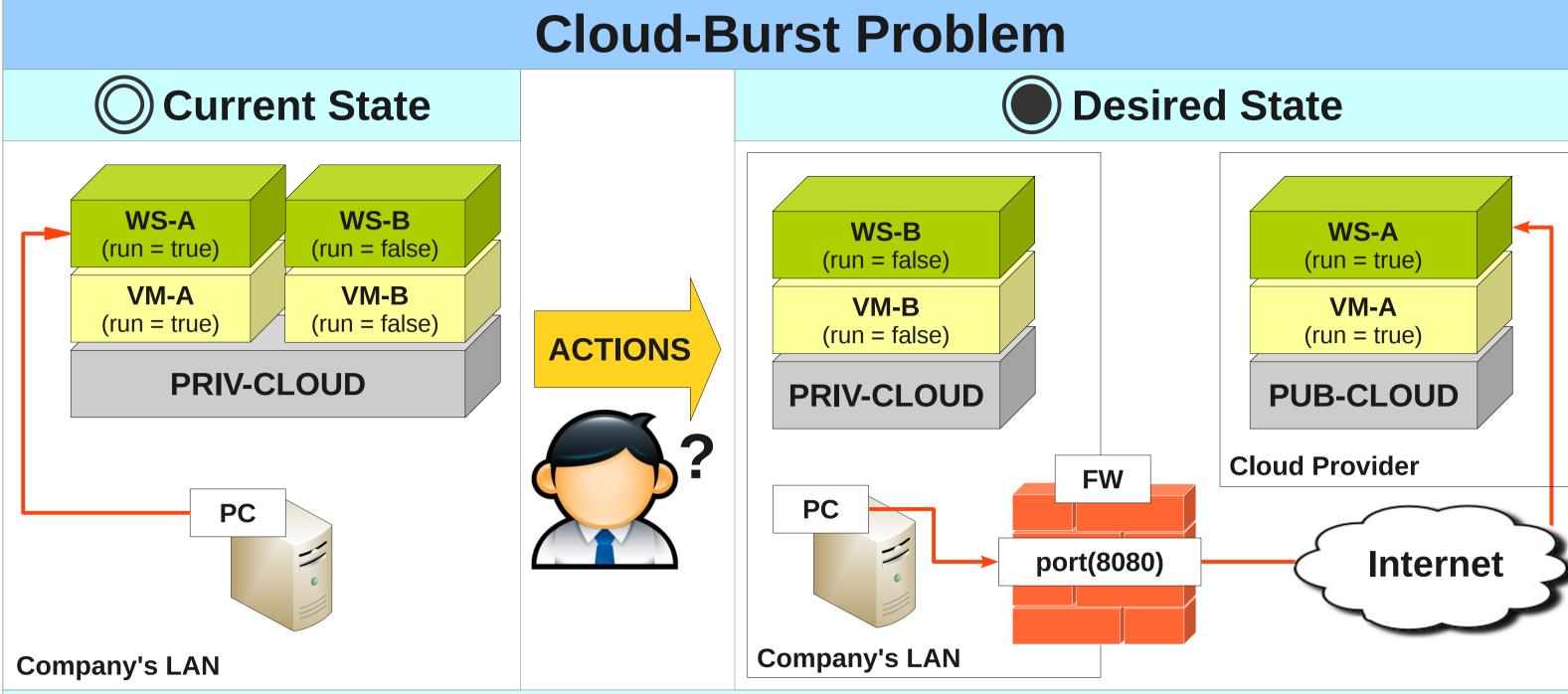
Gerhard Wickler

(g.wickler@ed.ac.uk)





Client must always refer to a running DNS server!

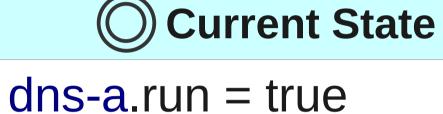


Constraints

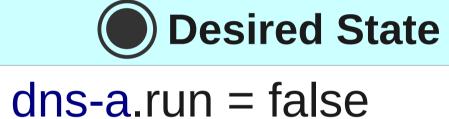
- The service must always available 24-hours per day without any down-time.
- Reconfigured firewall to allow the LAN PCs to have connection with the service on public cloud.
- The web service cannot be installed on any other machines due to the limitation of the license.







dns-b.run = false client.dnsserver = dns-a



dns-b.run = true client.dnsserver = dns-b

Actions

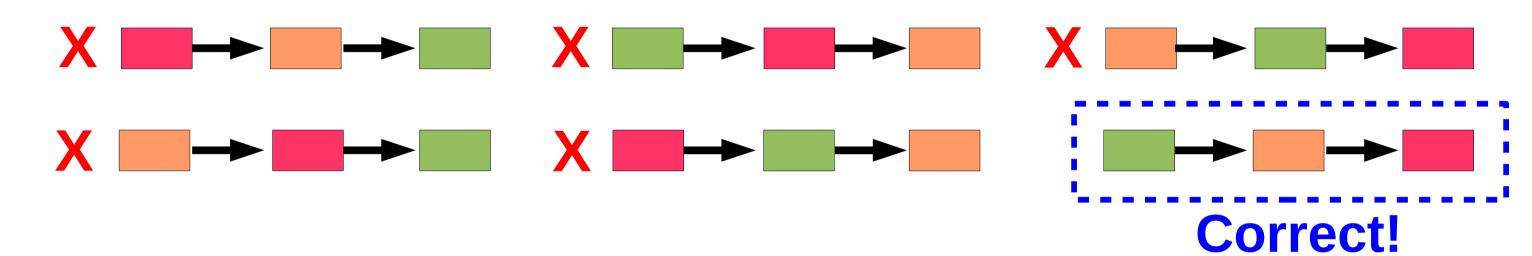
start(dns-b) change(dns-a,dns-b,client)

stop(dns-a)

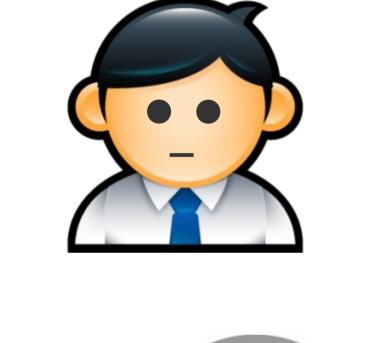
Solution for Service Reference Problem

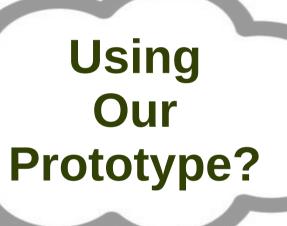
- > Executing the actions in an essentially indeterminate order!
- > 6 possible workflows:

Solution for Service Reference Problem



Highly likely producing the wrong workflow!







(C) Current State

dns-a.run = true dns-b.run = false client.dnsserver = dns-a

Desired State

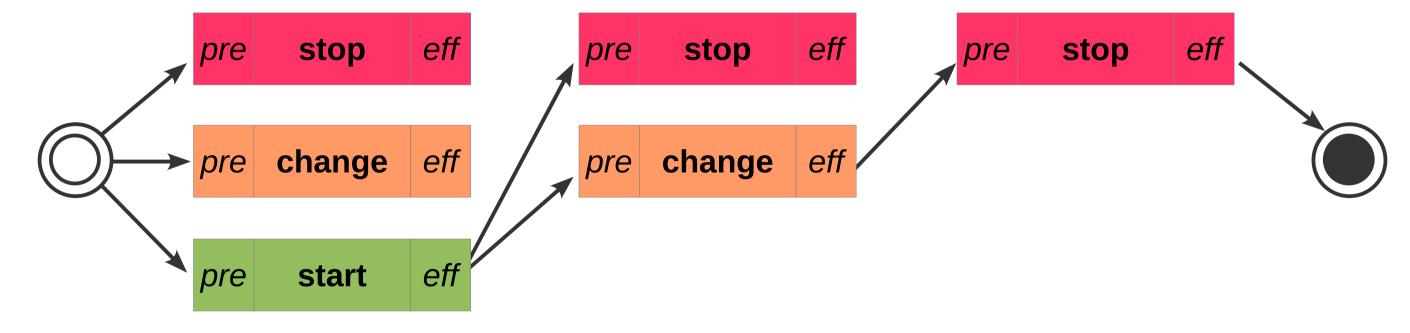
dns-a.run = false dns-b.run = true client.dnsserver = dns-b

Actions

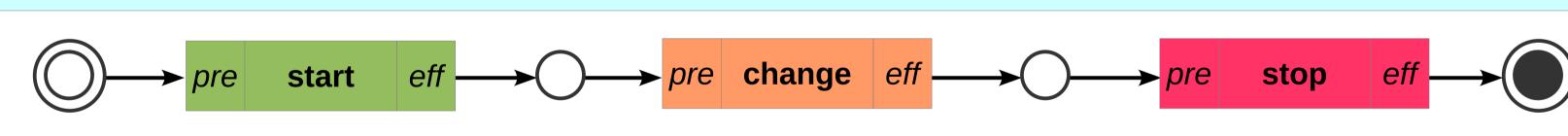


pre = preconditions *eff* = effects





Generated Workflow



(Current State

vm-a.cloud = priv-cloud vm-a.run = true ws-a.on = vm-aws-a.run = true pc.service = ws-a vm-b.cloud = priv-cloud vm-b.run = false ws-b.on = vm-bws-b.run = false

Desired State

vm-a.cloud = pub-cloud ws-a.firewall = fw ws-a.fport = 8080pc.service = ws-a vm-b.cloud = priv-cloud vm-b.run = false

Solution for Cloud-Burst Problem

Generated Workflow open-fport assign-service-fport (fw,8080) (ws-a,fw,8080) stop-service change-ref (ws-a,ws-b,pc) (ws-a) start-service start-vm (vm-b,priv-cloud) (ws-b,vm-b) migrate start-service stop-vm start-vm (vm-a,priv-cloud,pub-cloud) (vm-a,pub-cloud) (ws-a,vm-a) (vm-a) change-ref-fport stop-service stop-vm (ws-b,ws-a,pc,fw,8080) (vm-b) (ws-b)

>> Required computing time < 1s

Our Prototype:

> can automatically generate a workflow between any two declarative states

Conclusions

- > enables unattended, autonomic reconfiguration for failure recovery or other reasons
- > can achieve the desired state as well as preserving system constraints during reconfiguration
- > shows that it is possible to build the practical tool using production-quality tools for the deployment

Future Works

Investigating more distributed and localised approaches to improve the system's resilience

Reference

H. Herry, P. Anderson, and G. Wickler, **Automated** Planning for Configuration Changes, 25th Large Installation Systems Administration Conference (LISA'11), Boston, December 2011. (to be published)

