Comparison of Cloud Middleware Protocols and Subscription Network Topologies using CReST, the Cloud Research Simulation Toolkit

John Cartlidge & Dave Cliff

University of Bristol, UK

University of Bristol

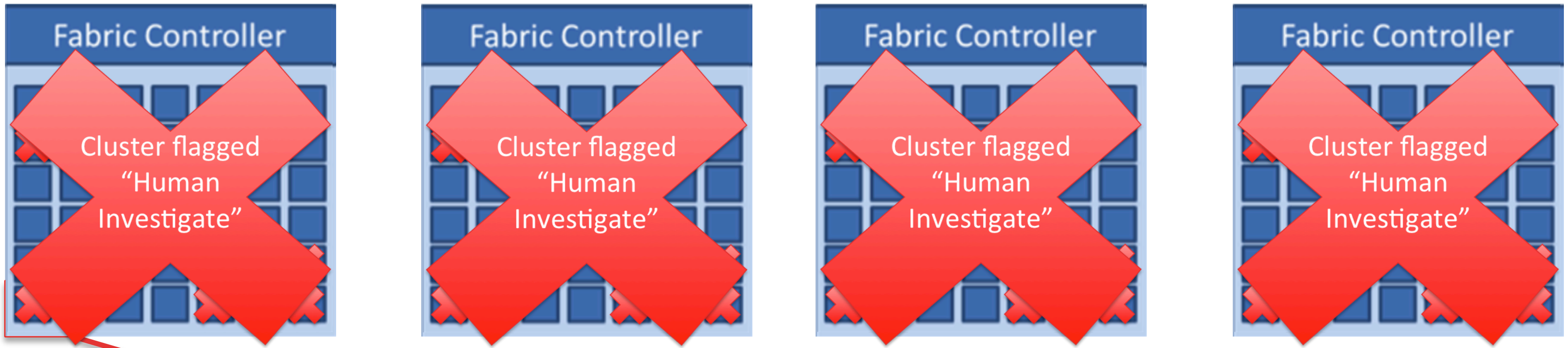
Outline

1. Frame the problem with a real-world example of cascading middleware failure
2. Review simulation tools for modelling cloud provision
3. Introduce and situate CReST – a new simulation tool
4. Problem: Comparison of middleware subscription topologies and communication protocols
5. Review previous findings published in the literature
6. Experiment: Use CReST to test the published findings
7. Results: Revision, rejection, & extension of findings
8. Summary & Conclusion

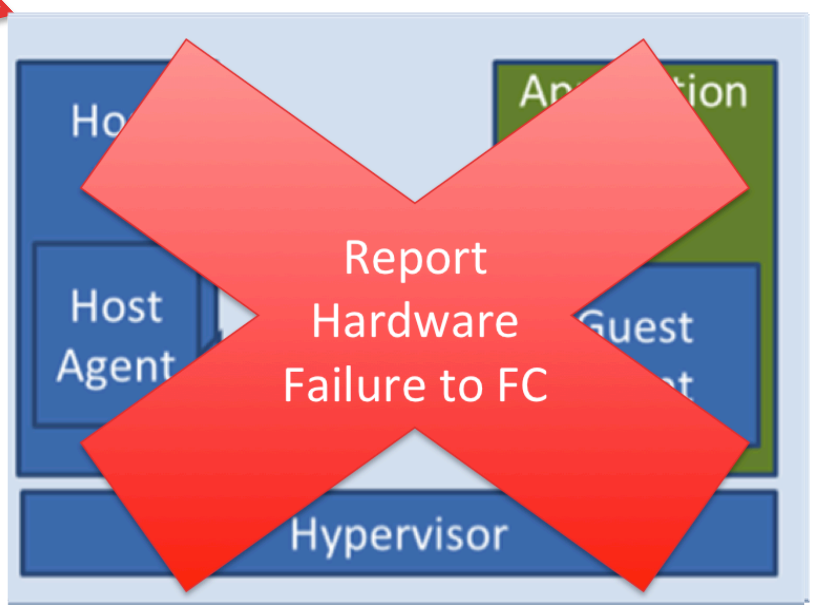
"The three truths of cloud computing are: Hardware fails, software has bugs, and people make mistakes"

Windows Azure Team, 2012

Laing, B. (2012). Summary of Windows Azure service disruption on Feb 29th, 2012. MSDN Windows Azure Team Blog, 09/03/12. http://bit.ly/AfdqyL



MicroSoft disabled service management functionality in all clusters worldwide for **more than 10 hours**



A subsequent series of human errors meant it was **more than 34 hours** before Azure was running at full service availability

Cost: ~3% annual revenue! (Azure issued a 33% refund to all customers for Feb 2012).

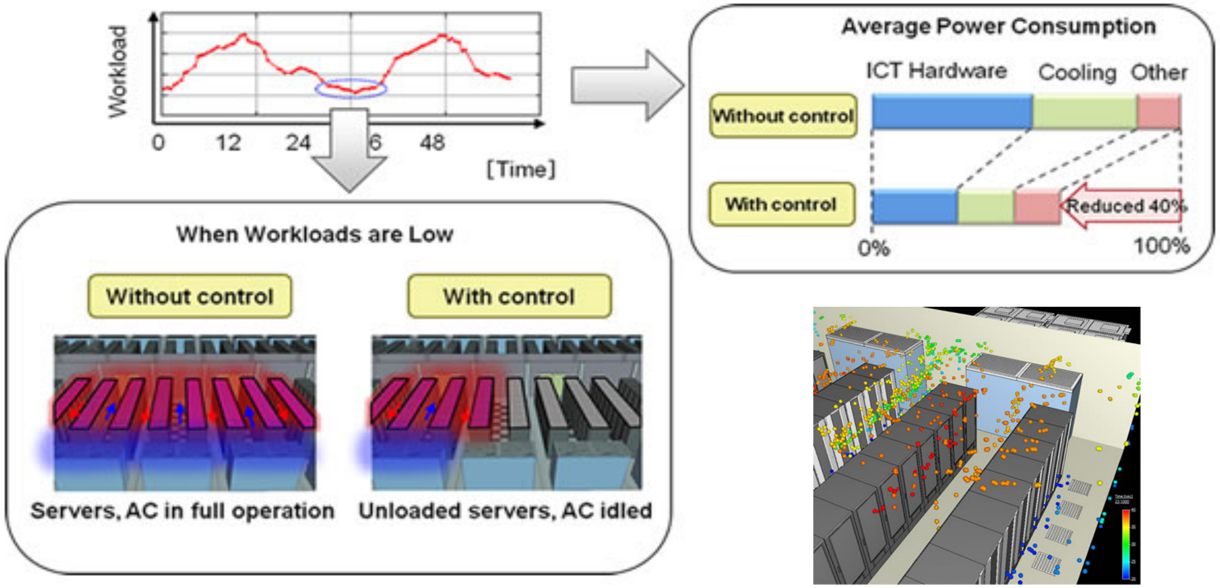
Solution: A consistent Date class!

University of BRISTOL

Simulation?

Fujitsu Labs

2011, Fujitsu Laboratories developed a **proprietary CFD data centre simulation tool**

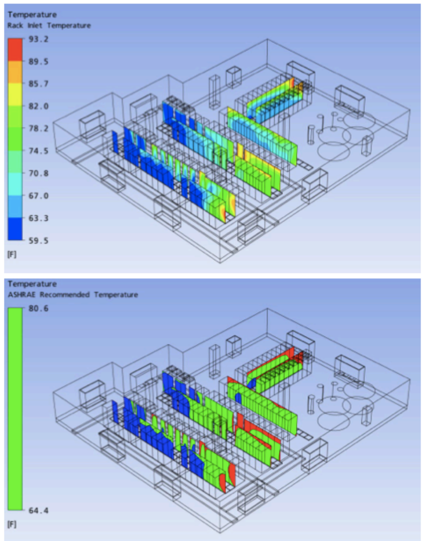


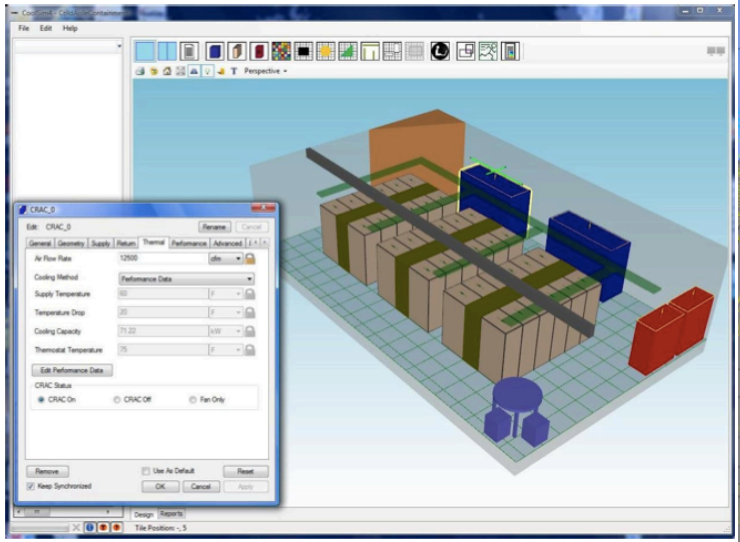
"It is impossible to directly perform tests…using an actual data centre. A promising alternative is to employ computer simulations to check the impact of control measures"

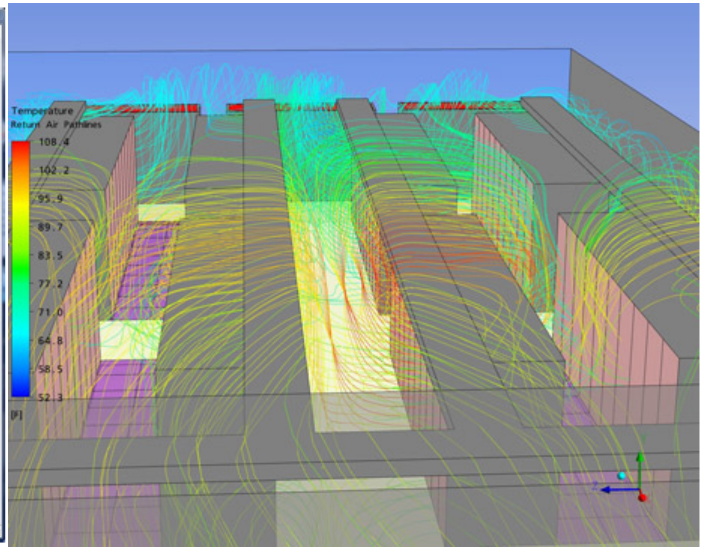
Results: linking together the control of servers and AC equipment may cut overall datacenter power consumption by as much as 40%

CoolSim

Applied Math Modelling Inc., founded 2008, offer CoolSim, a **CFD data centre simulation tool** with a **SaaS delivery model**. Subscriptions start at **$10,000 / year**







Use cases: *"predict cost savings results from DC modifications; determine maximum IT load and placement for a given DC; perform a comparative analysis of cooling system failure models; and optimise the design of a new or existing DC."*

CloudSim

- Developed at University of Melbourne
  - Open-source Java library/API
  - Leverages BRITE to model network topology
- A framework for modelling and simulation of cloud computing infrastructures and services
  - Models data centres at the level of networking and virtualisation rather than at the physical level
  - Has been used in at least 8 (correct Dec, 2012) academic publications

SimGrid

- First released in 1999; developed and maintained at INRIA
  - Open-source C library/API (Java, Liu and Ruby bindings)
- Models data centres at the level of networking and virtualisation rather than at the physical level
- Designed to simulate grid environments, recently extended to accommodate cloud computing framework
  - Documentation of virtual machine typedef states: *"all this is highly experimental and the interface will probably change"*
- Used in 119 journal, conference and PhD theses
  - Only 1 conference paper ostensibly related to cloud computing

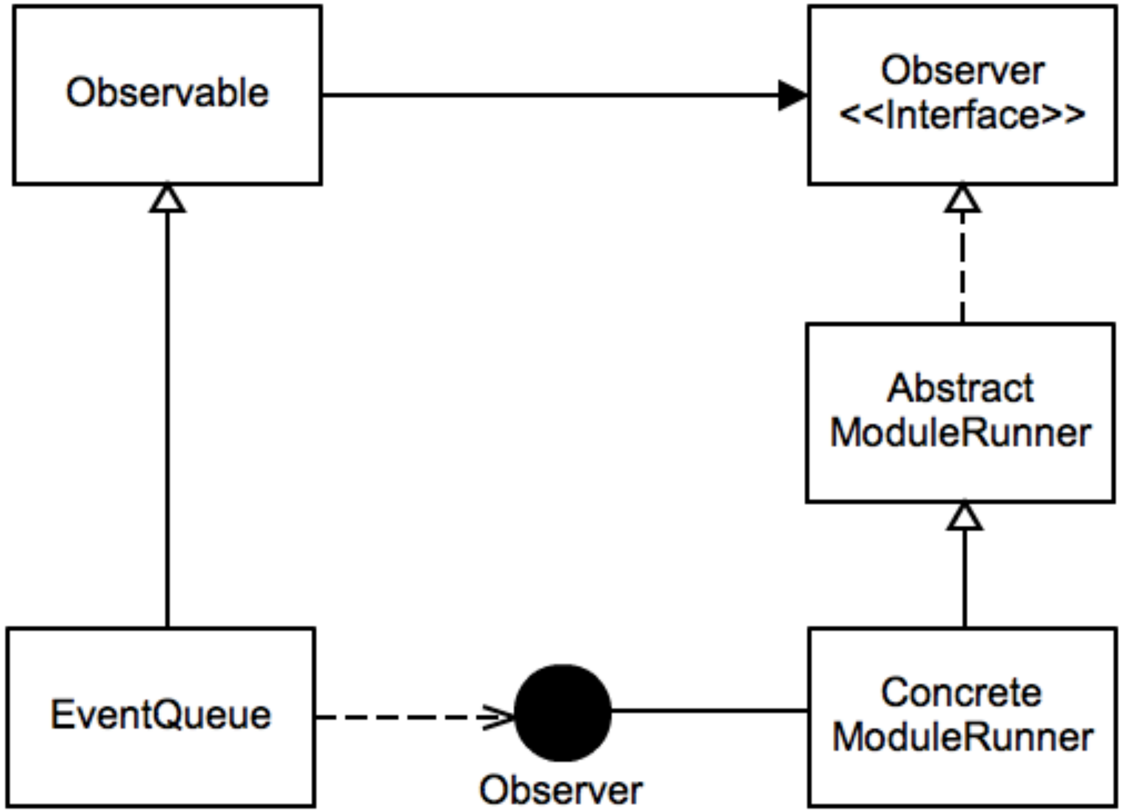
Summary of Cloud Simulation Tools

Name	Type	VM	Network	Physical	GUI	License
Fujitsu Laboratories	App	No	No	Yes (CFD)	Yes	Prop.
CoolSim AMM Inc.	SaaS	No	No	Yes (CFD)	Yes	Subs.
CloudSim UoMelbourne	Java Lib/API	Yes	Yes	No	No	Open Source
SimGrid Inria	C Lib/API	Yes	Yes	No	No	Open Source
CReST UoBristol	**Java App**	**Yes**	**Yes**	**Yes (Simple)**	**Yes**	**Open Source**

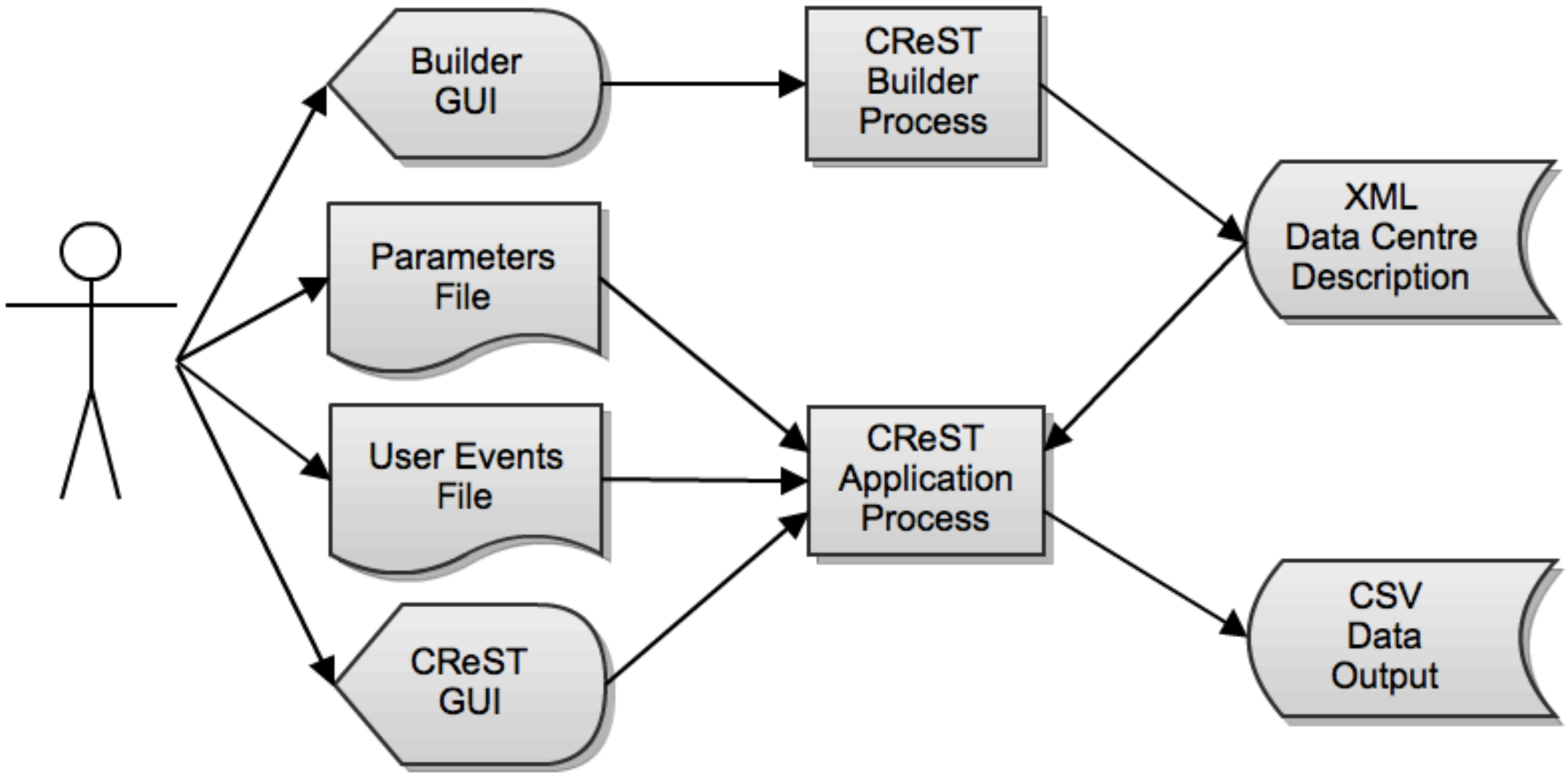
CReST – A modular design

- Open-source application designed for research and teaching
  - http://sourceforge.net/projects/cloudresearch
  - 230+ downloads in first year since release in Apr 2012 (44% in India)
- Designed as a set of *coupled modules* that can be independently switched on or off depending upon the level of abstraction required, including:
  - Thermal – Heat generation, propagation and extraction
  - Energy – Energy used by hardware
  - **Failures** – Permanent and temporary hardware failures
  - Services – Scheduling and allocation of VMs
  - Demand – User demand and market supply
  - **Subscriptions** – Middleware (platform) subscription network
- Extensible: new modules can be added and current modules extended
- Interaction **between** modules produces complex behaviours

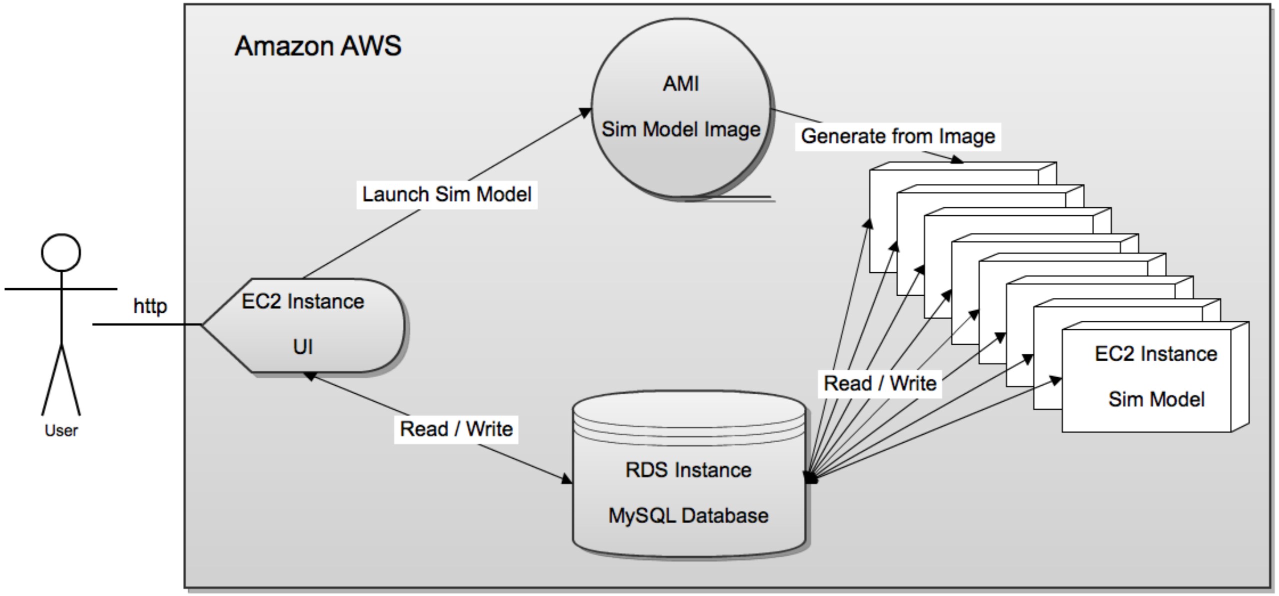
CReST Module Architecture



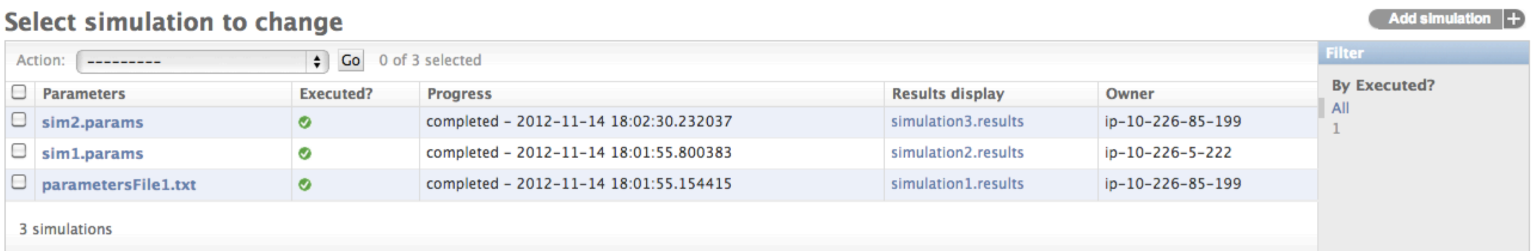
CReST Architecture



Run Simulations in Parallel on AWS



Django/Python on BitNami instance, with MySQL DB, using boto AWS interface



Select simulation to change

Add simulation

Action: --------- Go 0 of 3 selected

Parameters	Executed?	Progress	Results display	Owner
sim2.params	✔	completed - 2012-11-14 18:02:30.232037	simulation3.results	ip-10-226-85-199
sim1.params	✔	completed - 2012-11-14 18:01:55.800383	simulation2.results	ip-10-226-5-222
parametersFile1.txt	✔	completed - 2012-11-14 18:01:55.154415	simulation1.results	ip-10-226-85-199

3 simulations

Filter

By Executed?

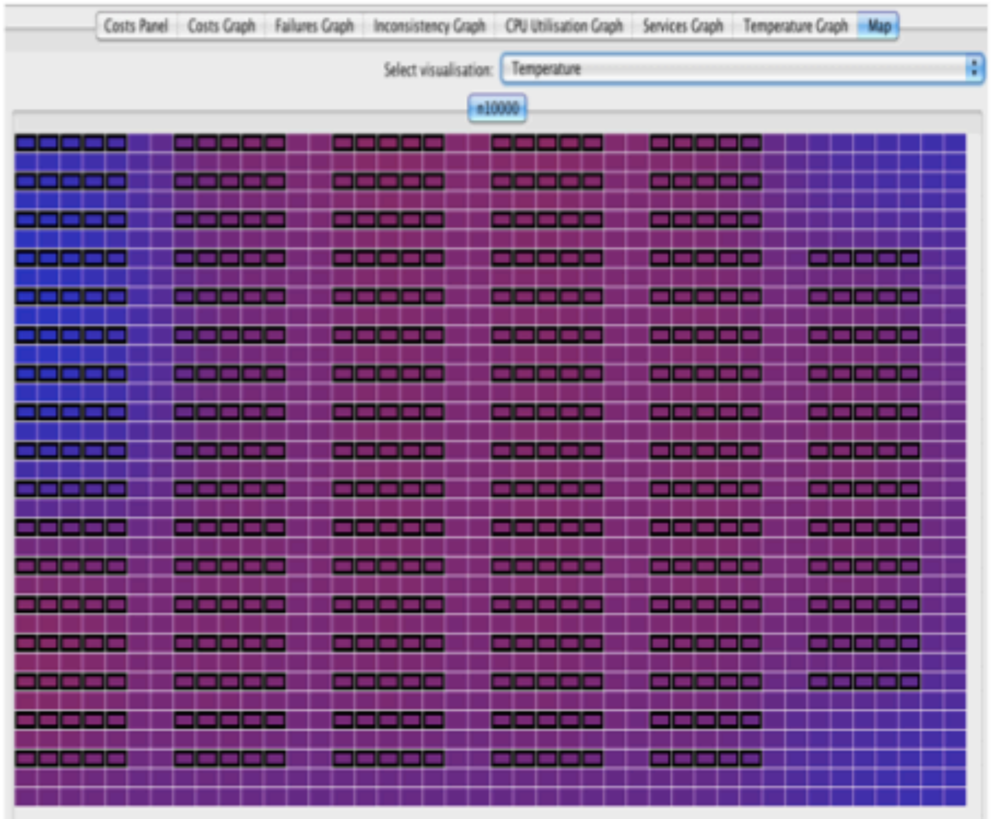
All

1

Admin web page: Upload config-params files, launch simulations, & download results files

CReST – GUI Screenshot

Aerial view of DC rack layout
Failed servers highlighted in red

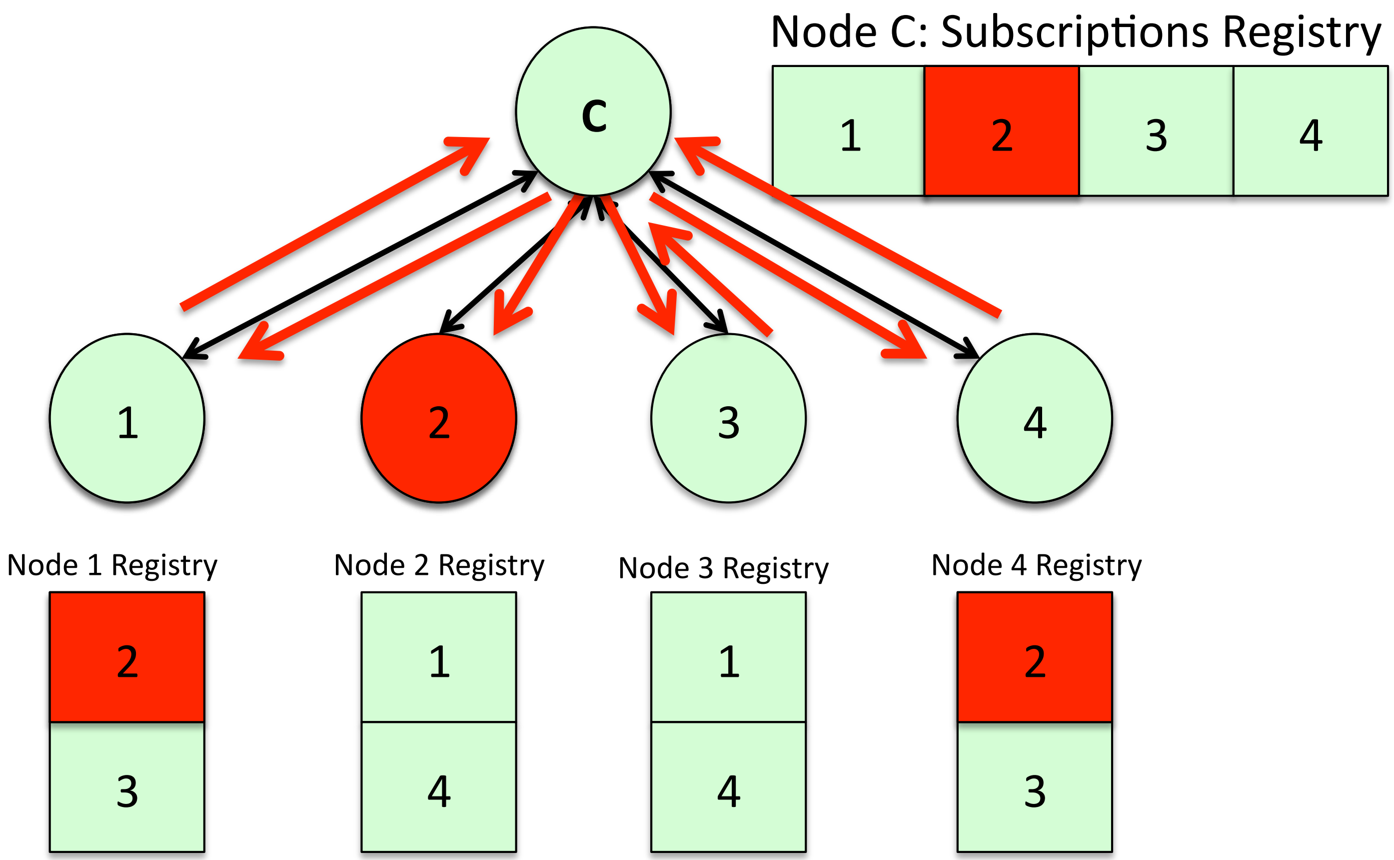


Thermal view of DC
Hotter regions red, colder regions blue

Middleware subscription module

- Describes a communications network (a directed graph), with each node corresponding to an individual server
  - E.g., MS's Autopilot, used for *Index Serving* for Windows Live Search
- Nodes subscribe to other nodes and periodically query for status
- Middleware can be centralised or distributed
  - Consistency is not guaranteed in distributed systems
  - Nodes may form an *inconsistent* view of other nodes
  - E.g., If node *A* thinks node *B* is working, when it has failed
- The percolation of inconsistencies is determined by:
  - the **network topology**; and
  - the **communications protocol**
- We use CReST's subscription module to test and extend some findings that have been presented in the literature

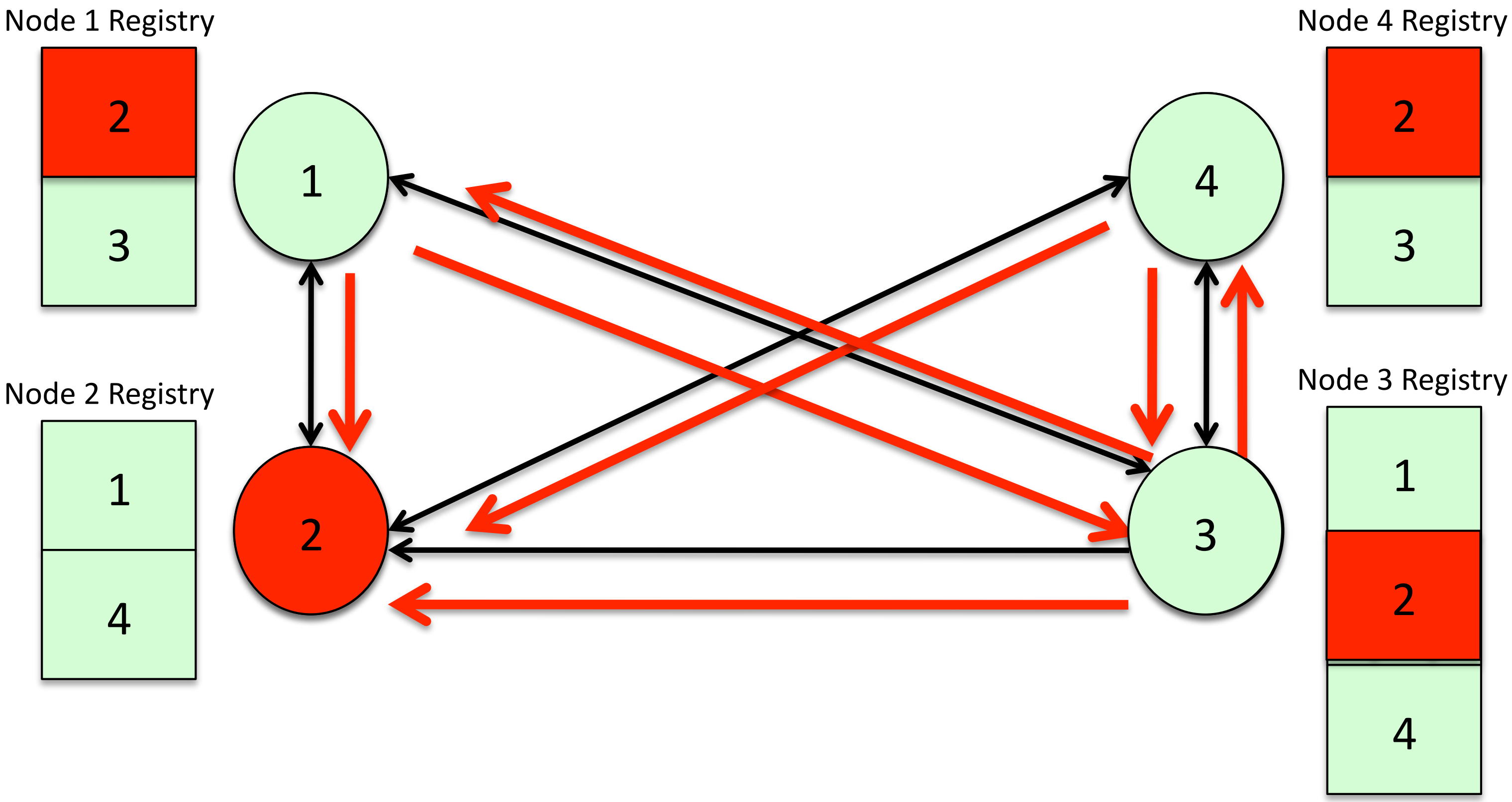
Central/Hierarchical Protocol



A central node periodically requests status info from other nodes in the network. Nodes query the central node for status information of other nodes

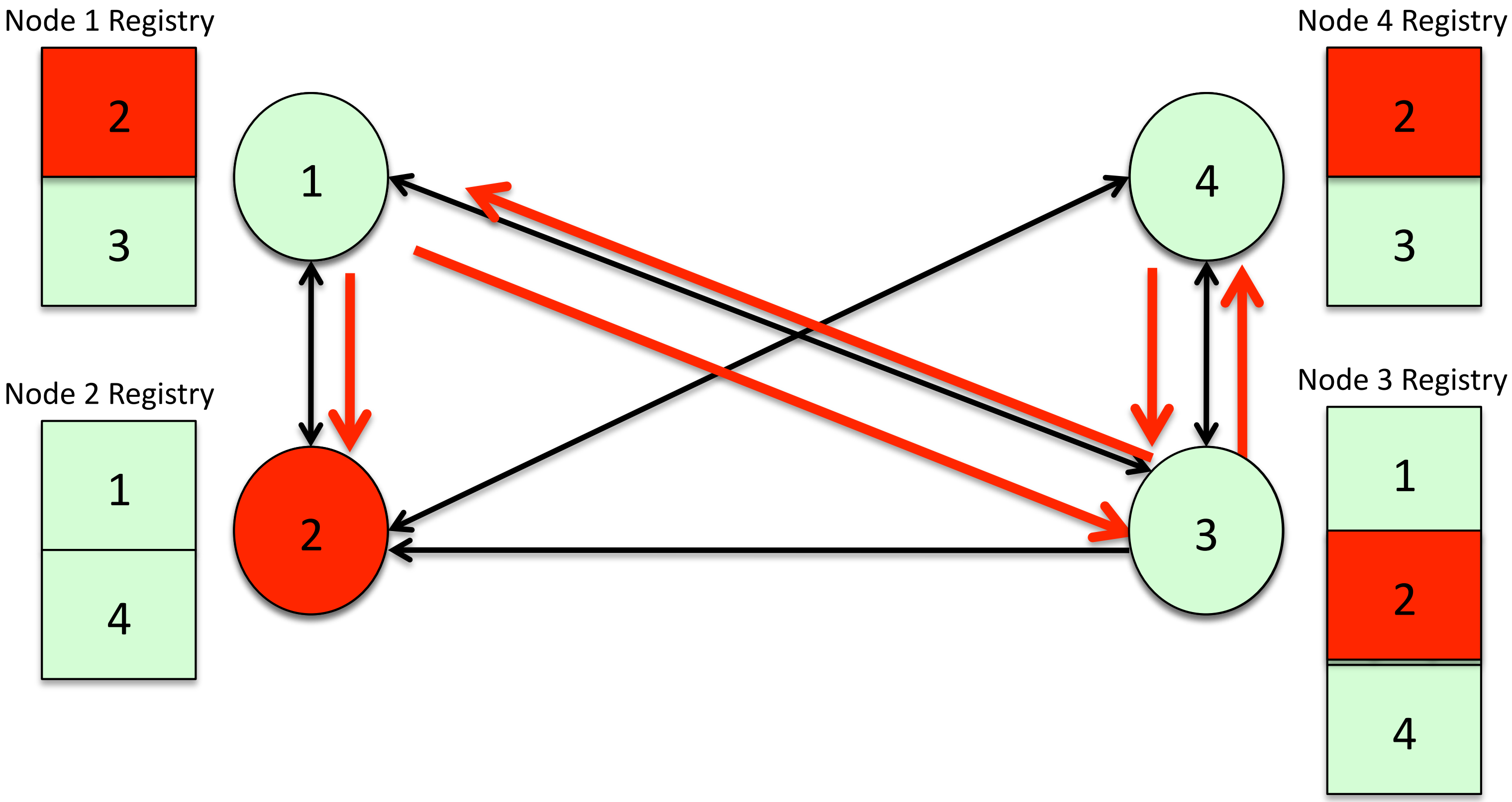
University of BRISTOL

P2P Protocol



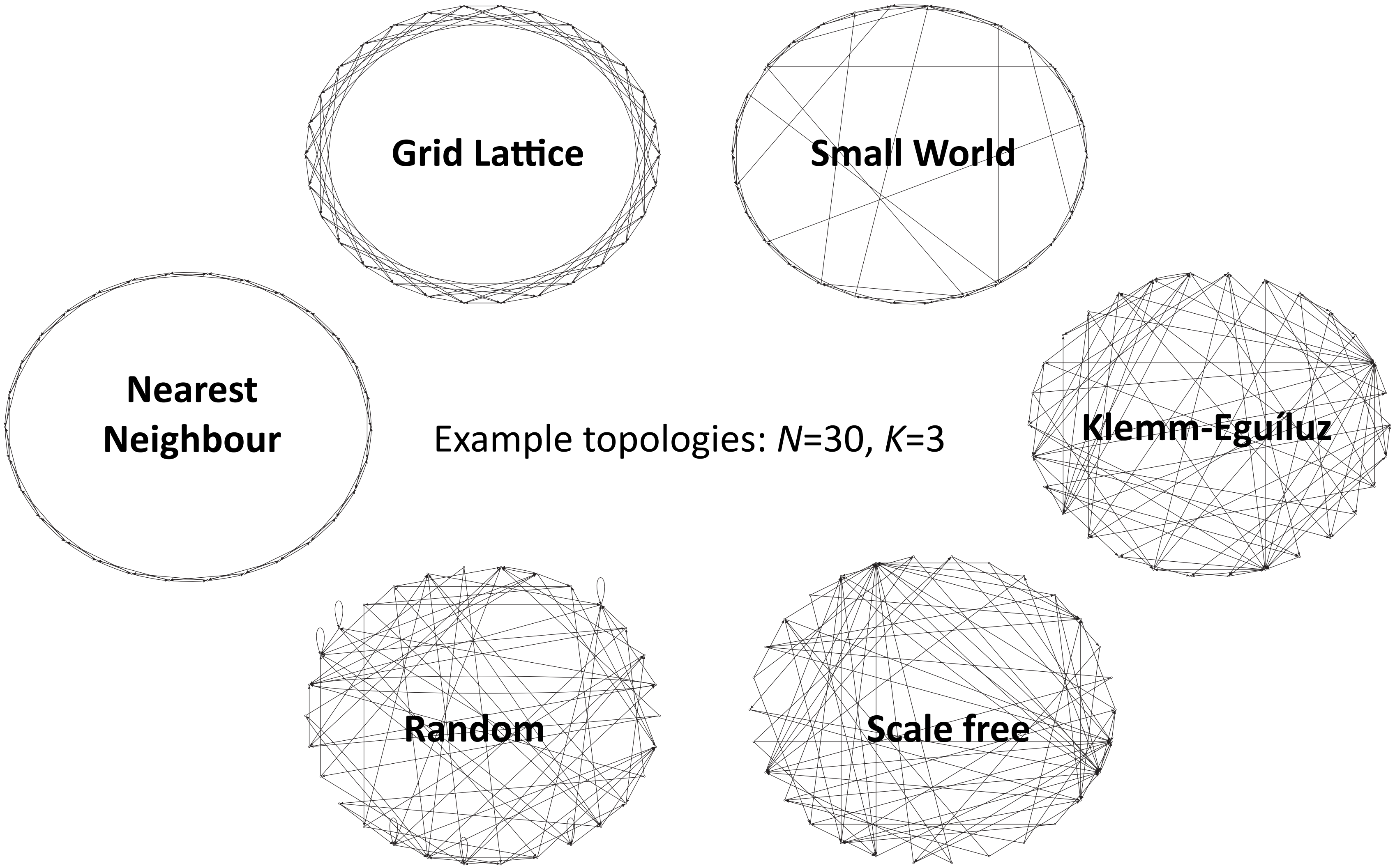
Nodes **directly** request status information of other nodes they are subscribed to

TP2P protocol

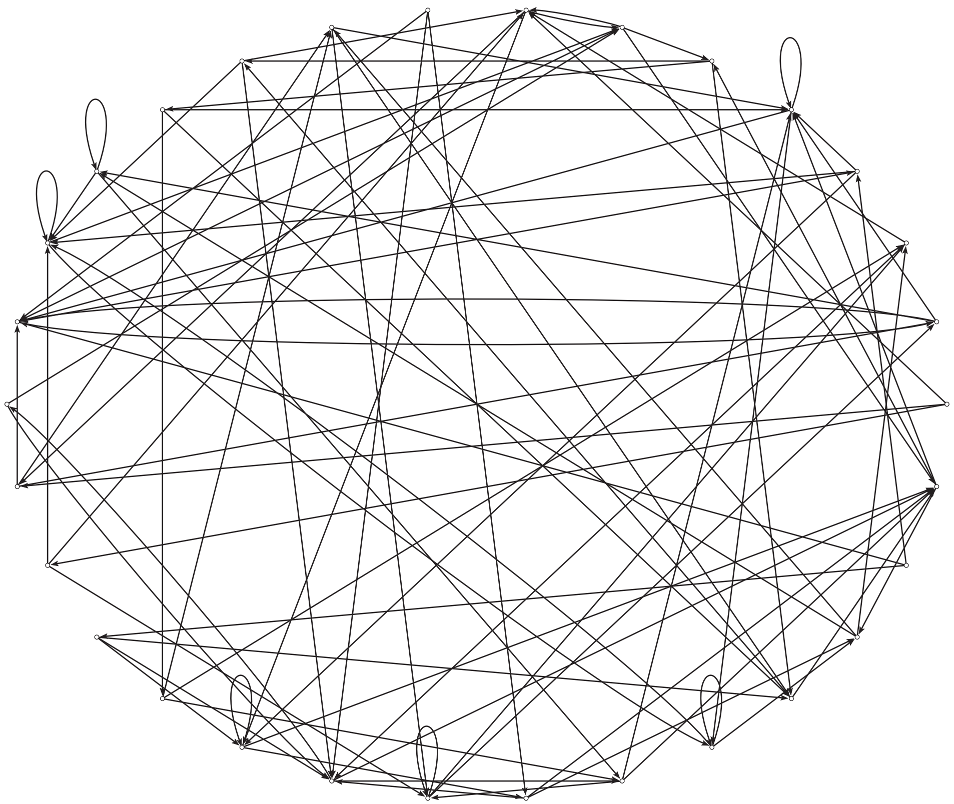


Nodes directly request status information of other nodes and ***also*** pass on information about other mutually subscribed nodes

Network Topologies

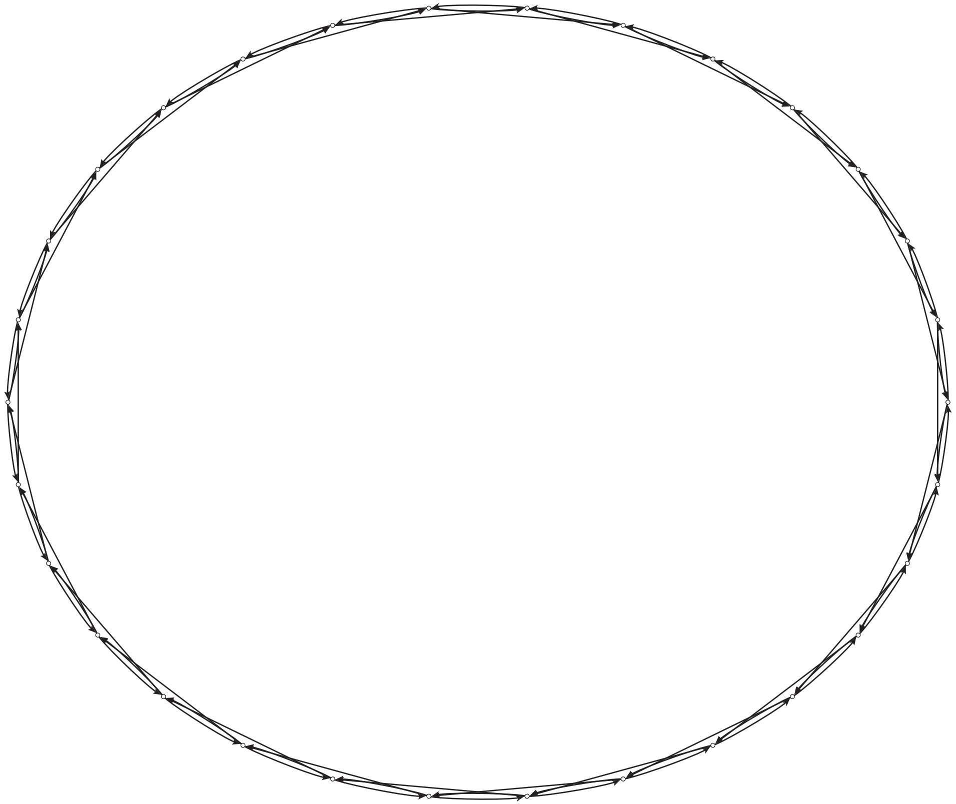


Random Network



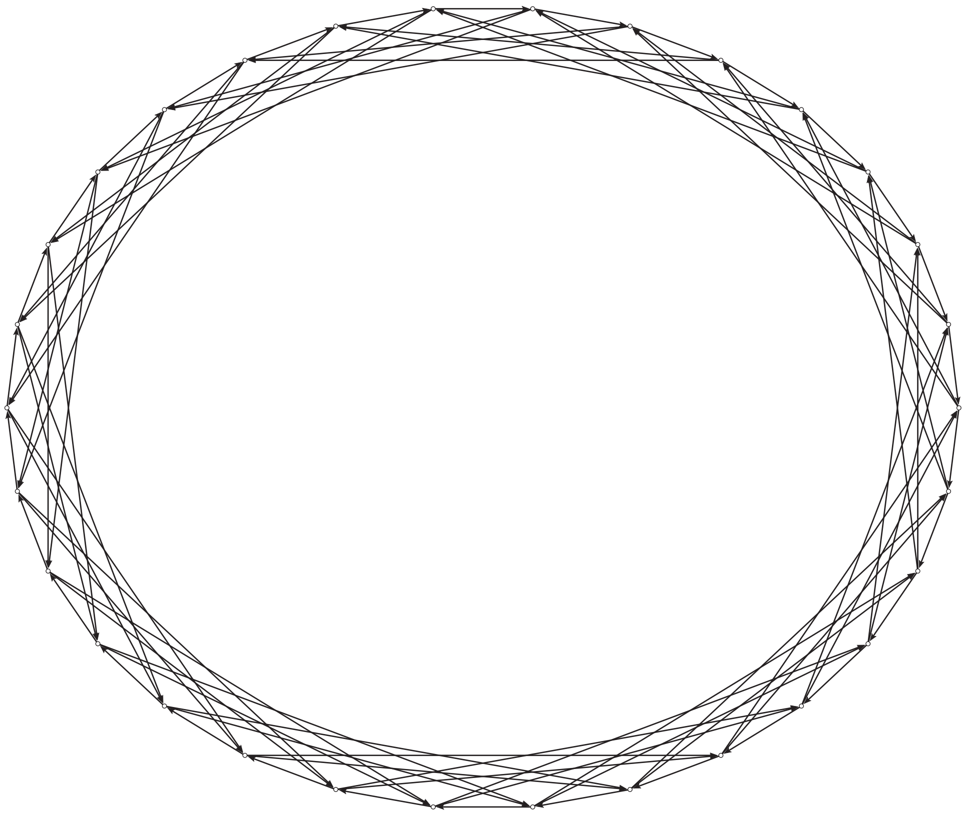
- nodes randomly connected to *exactly K* other nodes
- **small clustering/transitivity** coefficient (i.e., very few neighbours of a node are themselves neighbours)
- **small diameter** (i.e., average path length between any two nodes is very small)

Nearest Neighbours



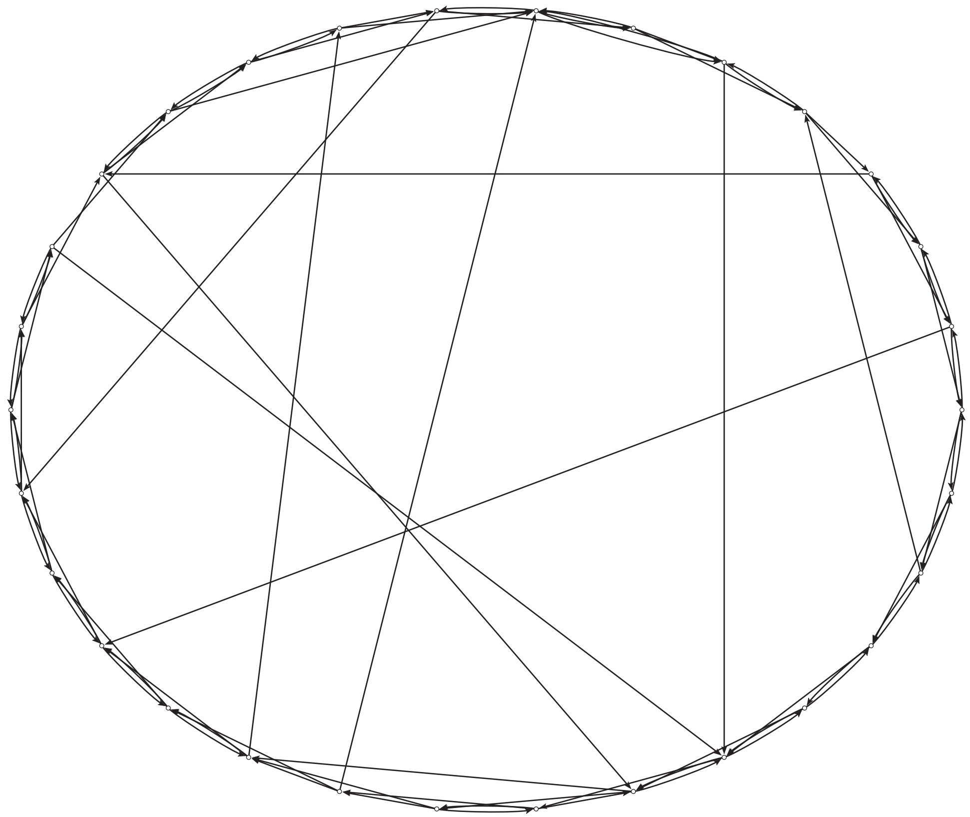
- nodes arranged in a 1D circular array, each attached to *K* nearest neighbours
- **very large clustering/ transitivity**
- **very large diameter/ average path length**

Regular Grid Lattice



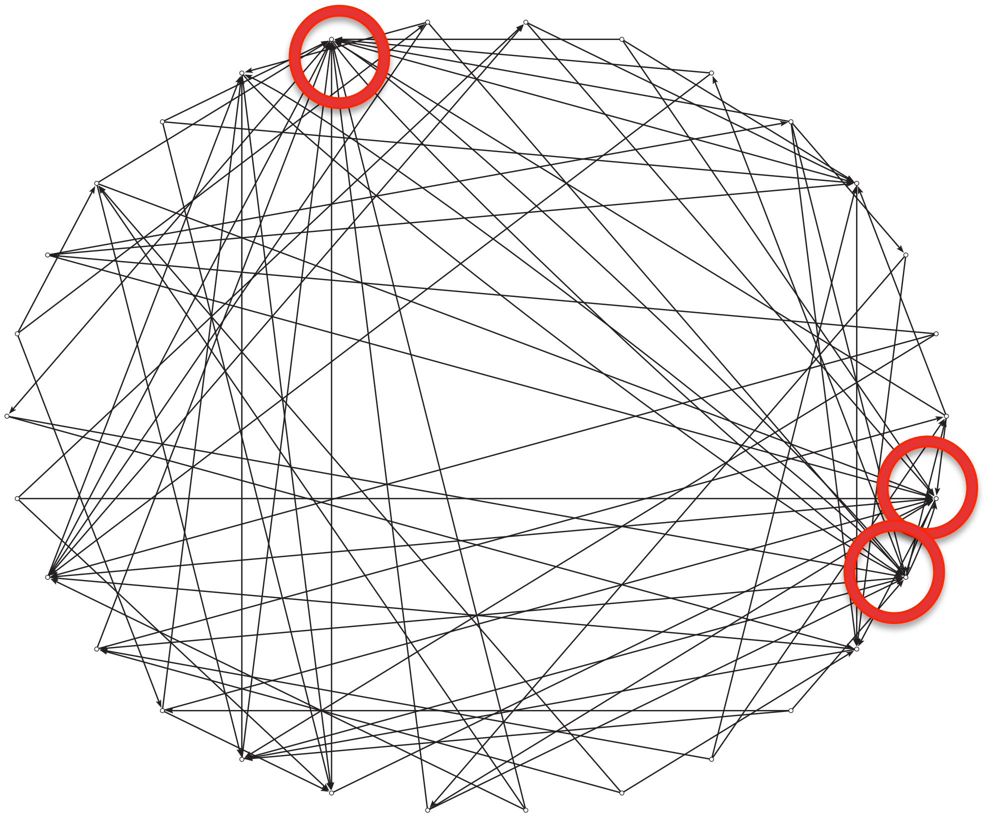
- nodes arranged on a toroidal grid/lattice and connected to *K* nearest neighbours
- **large clustering coefficient**
- **large diameter** (but smaller than Nearest Neighbours)

Watts-Strogatz (Small World)



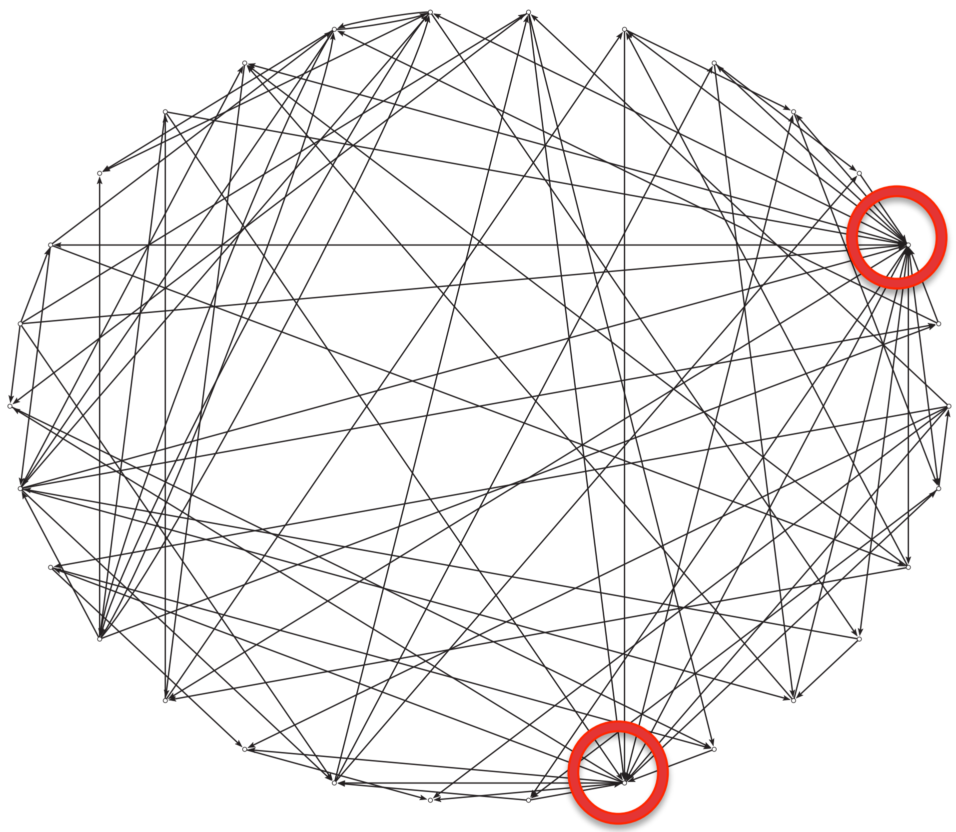
- nodes connected using Watts-Strogatz algorithm
- **large clustering coefficient**
- **relatively small diameter**
- degree distribution sharply peaked around the mean value, *K*

Barabási-Albert
(Scale Free)



- nodes connected using Barabási-Albert algorithm
- **small clustering coefficient**
- **small diameter**
- **distribution of the node degree is scale-free** (i.e., it decays as a power law), producing a hierarchical network organisation

Klemm-Eguíluz (Scale Free / Small World)



- nodes connected using Klemm-Eguíluz algorithm. A "mixing" parameter, $0<\mu<1$, varies network properties between Small World ($\mu=0$) and Scale Free ($\mu=1$)
- At intermediate values, e.g., $\mu=0.15$, the network has a relatively **large clustering coefficient** and **small diameter**, while maintaining a **scale-free distribution of node degrees**

Findings in the literature to use as hypothesis tests

Published findings that we shall test:

1. Inconsistencies grow with DC size [1];
2. Subscription topology has no significant effect on P2P and hierarchical protocols [1];
3. Under TP2P protocol, there is a direct relation between network transitivity and inconsistency [1];
4. Under TP2P protocol, inconsistencies fall as path lengths drop [2].

[1] I. Sriram & D. Cliff (2010) "Effects of component-subscription network topology on large-scale data centre performance scaling." http://bit.ly/10IxfEs
[2] I. Sriram & D. Cliff (2010) "Hybrid complex network topologies are preferred for component-subscription in large-scale data-centres." http://bit.ly/TA5rQU

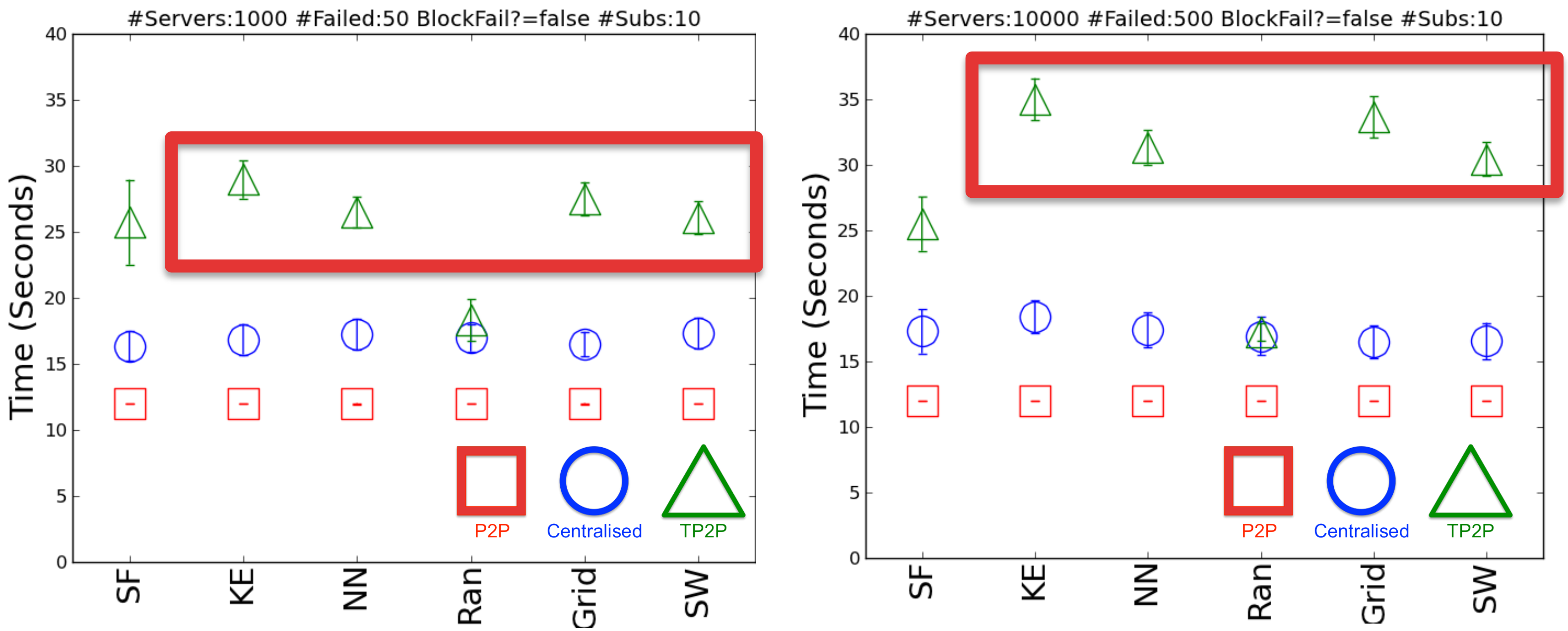
Experimental Design

- Configure a network. Repeat the following:
  - Initially set all server nodes to working
  - After a short random time period, fail a subset of servers
    - Fail servers randomly
    - Fail servers correlated with geographic location (e.g., full rack failure)
  - Calculate the maximum number of nodes in the network that become inconsistent
  - Calculate the time until the network becomes consistent
  - Calculate the load (in network hops) to become consistent

Test 1: *"inconsistencies grow with DC size"*

The Effects of Scaling (*n*)

1,000 nodes   10,000 nodes

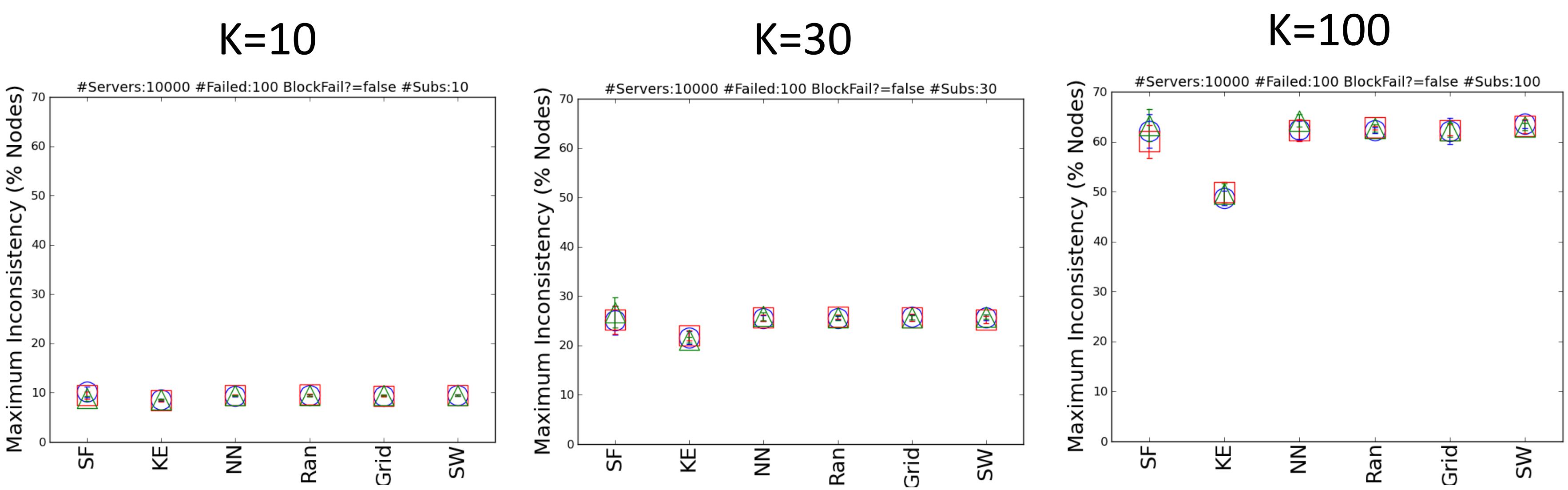


- Central & P2P, inconsistencies ***do not*** increase significantly with network size
- TP2P, inconsistencies ***only*** increase when the network is highly clustered, since "stale" information is more likely to be passed on

University of BRISTOL

Test 1: *"inconsistencies grow with DC size"*

The Effects of Network Density (*K*)



10,000 nodes, random failure

For all topologies and protocols, *inconsistencies increase with K*

Test 1: "*inconsistencies grow with DC size*"

- Central/P2P: *n* has *no effect* on inconsistency
- TP2P: inconsistencies *only* increase with *n* when the network topology has a high clustering coefficient
- For all topologies and protocols, inconsistency increases with network density, *K*
- Conclusion: **finding 1 is incorrect**
  - Inconsistencies grow with subscriptions, *K*, *not* DC size, *n*.
- In the original work, $K=\sqrt{n}$ and thus automatically scales with DC size, *n*.
  - Therefore, the original finding is an experimental artifact!

Test 2: *"Topology has no effect on P2P and hierarchical protocols"*

- Results show that topology has a significant effect on the network load of P2P
  - Load decreases as clustering and average path length between two nodes increases
- Conclusion: **finding 2 needs to be refined**

Test 3: *"Under a TP2P protocol, there is a direct relation between network transitivity and inconsistency"*

- Under correlated failure, topologies with *higher* clustering remain *more* consistent
  - Reason: in highly clustered networks, localized failure is less likely to percolate the network
- Conclusion: **finding 3 has been extended**
  - The original work considered only random failures

Test 4: *"Under a TP2P protocol, inconsistency falls as path lengths drop"*

- Inconsistency is sensitive to *K*
  - E.g., TP2P: when *n=10000 & K=100*, SF networks take *longer* to become consistent than NN networks, despite a *shorter* average path length
- Conclusion: **finding 4 needs to be refined**

Summary & Conclusions

- We have introduced CReST, a new open-source cloud simulation tool
- We have used CReST to test 4 findings in the published literature
  - 1 was rejected
  - 2 were extended
  - 1 was refined
- In future, we hope to run further experiments to tease out the detailed relationships hinted at in these results
- Also, we would like to simulate more real-world scenarios, such as Azure's *Leap Day Bug*
- Finally, we aim to use CReST in different problem areas: (brokerage models, markets of competing providers, market based allocation of resources, …)

Questions?

Dr John Cartlidge, Research Associate,
University of Bristol, BS8 1UB, UK
Email: john@john-cartlidge.co.uk
Web: http://www.cs.bris.ac.uk/~cszjpc