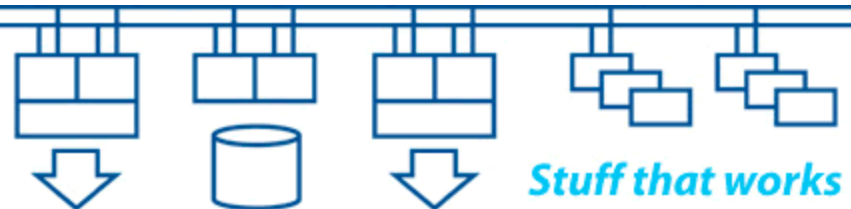


BCS Jersey – 25<sup>th</sup> Jan. 2010

# Designing and implementing mission-critical systems

Colin Butcher



- **Business continuity:**
  - It's not just the systems – it's everything!
- **Disaster tolerance:**
  - Continue operations while surviving major site outages without loss of data
- **High availability:**
  - Continue operations while surviving equipment and software failures at a site without loss of data

- **Disaster recovery:**
  - The process of restarting sufficient operations to run the business after an serious outage, typically from another location
- **Budget and Schedule:**
  - They have to be appropriate for the problems we're trying to deal with. Don't set them first!

	Before	Now	After
Environment			
System			
Component			

See [www.triz.co.uk](http://www.triz.co.uk) for a lot more information!

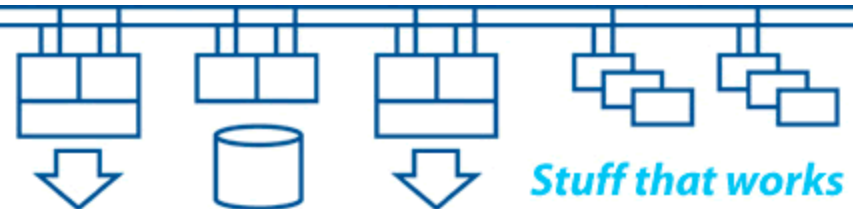


**Risk is a fact of life. We have to deal with it as best we can – or at least well enough for the circumstances we find ourselves in.**

**Think about both project management and technical design as part of systems engineering, then apply techniques from other engineering disciplines to help us analyse the situation and guide our thinking.**

**Disaster-tolerant systems aim to minimise the risk of loss of service and loss of data as much as possible.**

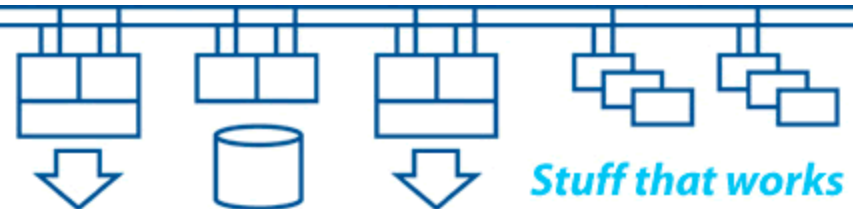
- What is the probability of a situation occurring ?
- What is the impact if that situation occurs ?
- What are the long-term consequences ?
  
- Most projects handle medium risk well enough
- Many projects over-specify to cater for what are in fact low risk issues
- Some projects under-specify and fail to cater for what are in fact high risk issues



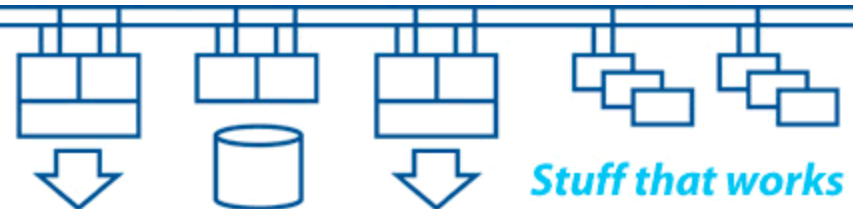
Mission critical systems need to be able to:

- Survive failures (resilience and failover)
- Survive changes (adapt and evolve)
- Survive people (simplify and automate)
- Never corrupt or lose critical data (data integrity)
- Requirements never remain static over an extended period of time, so we need to be able to make changes during the operational lifetime of the system
- Circumstances change, so we often need to be able to extend the operational lifetime and scope of a system

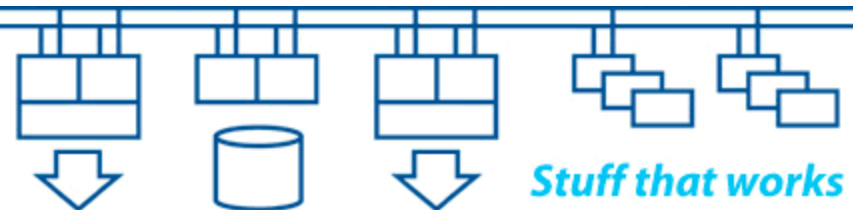
- Safety-critical systems (especially real-time monitoring and control systems such as air traffic control) require exceedingly high levels of availability. They also have to be fail-safe in order not to endanger lives.
- True 24x365 mission-critical systems are fairly rare. With these there is no “downtime window” to take backups, fix faults or to make changes. So, whatever you do has to be done “live” – and very carefully!



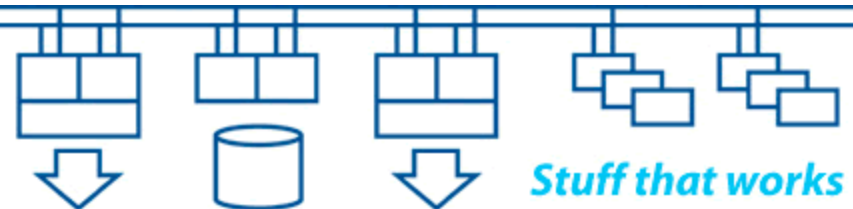
- What does the business require the systems to do ?
- What are the consequences if the systems fail ?
- What happens if you push beyond the limits ?
- How far from the edge are you ?
- How do you know ?
- What can we measure ?
- What comparisons can we make ?
- What evidence can we look at ?



Cause of Outage:	Planned (Maintenance)	Unplanned (Failure)
Hardware	?	?
Operating System	?	?
Network Layer	?	?
Layered Products	?	?
Application Software	?	?
Application Data	?	?
Environment	?	?
People	?	?



- How long have we got ?
- How much data can we afford to lose ?
- How can we model the whole system ?
- How can we predict the overall availability ?
- The closer you get to 100% uptime the more expensive a satisfactory solution will become



## RPO = Recovery Point Objective

- How much data can we tolerate losing ?
- How quickly do we need to react to a failure ?

## RTO = Recovery Time Objective

- What level of service outage can we tolerate ?
- How quickly do we need to recover ?
- How quickly do we need to be ready to deal with a subsequent failure ?

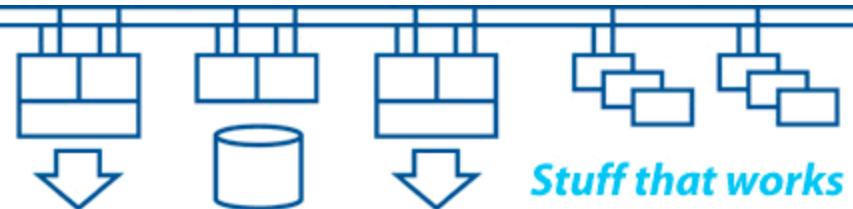
## Availability:

- Probability of system being available for use at a given instant in time within the 'operational window'
- Function of both MTBF (reliability) and MTTR (repair time)

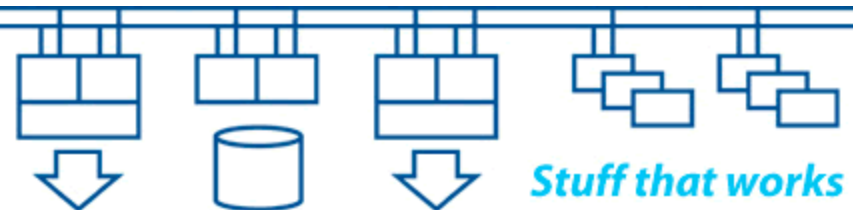
## Performance:

- Performance issues are often the cause of transient system failures and disruption
- The systems have to have sufficient capacity and performance to deal with the workload in an acceptable period of time under normal, failure and recovery conditions

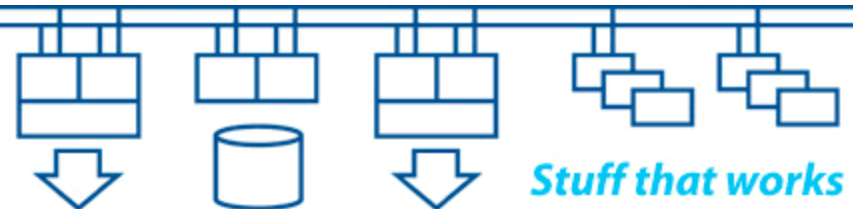
- Why is understanding performance so important in mission-critical systems ?
- How does performance affect availability ?
- What can we do to test a system before it goes into production ?
- What can we do to test changes to a system before we implement them ?
- What can we do to monitor a system once it's in production ?



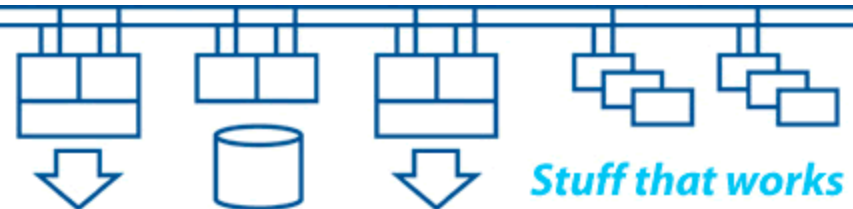
- **Bandwidth – determines throughput**
  - It's not just “speed”, it's throughput in terms of “units of stuff per second”
- **Latency – determines response time**
  - Determines how much “stuff” is in transit through the system at any given instant
  - “Stuff in transit” is the data at risk if there is a failure
- **Jitter (“div latency” or variation of latency with time) – determines predictability of response**
  - Understanding jitter is important for establishing timeout values
  - Latency fluctuations can cause system failures under peak load



- Size systems to cope with peaks in workload
- Minimise “wait states” (caches, parallelism)
- Minimise contention for resources and data structures
- Understand the need for synchronisation and serialisation of access to data structures
- Maximise “User Mode”, minimise the other modes:
  - The fastest IO is the IO you don’t do
  - The fastest code is the code you don’t execute



- Understand how the applications could break down into parallel streams of execution:
  - Some will be capable of being split into many small elements with little interaction between the parallel streams of execution
  - Some will require very high interconnectivity between the parallel streams of execution
  - Some will require high-throughput single-stream processing
- Understand scalability – do as much as possible once only, do little as possible every time



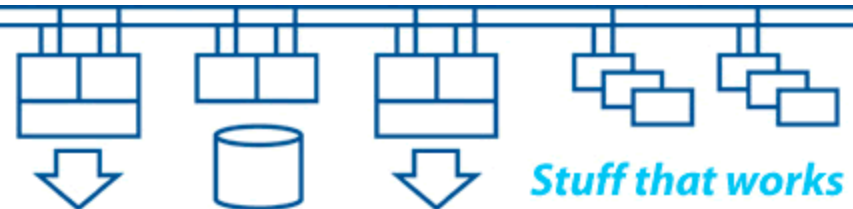
- Which parts of the system are mission-critical ?
- Which parts of the system are safety-critical ?
- What kind of failure do we prefer ?
- What state transitions occur during both failure and recovery ?
- What happens if there's another failure part way through a state transition during a failure ?
- How can we recover from a failure without data loss or data corruption ?
- How will you test your failure and recovery scenarios ?
- How can you make changes while it's working ?

- Naming conventions
- Quorum and voting schemes
- Data replication schemes
- Effects of distance on network and storage protocols
- Symmetric or asymmetric operation – how good is your “crystal ball” ?
- Centralised (and replicated) monitoring and alerting
- Remote access for management and operation
- Avoid automation of decision making

- Nothing happens instantaneously - there is always a “state transition”
- How long do state transitions last for ?
- What can we do while a state transition is in progress ?
- Can we ensure that there are no timing windows or other flaws in the design ?
- What possible states can a pair of machines be in ?
- How can we test it ?

A	B
Off to Master	Off
Master	Off to Standby
Master to Off	Standby to Master
Master to Off	Off to Master
Master to Standby	Standby to Master
<b>Master to Hung</b>	<b>Standby to Confused</b>
...	

- We need to demonstrate that service will continue with minimal disruption during failure and recovery
- We need to regularly rehearse and test our procedures and plans to ensure that we stay current
- We need to test scalability as well as functionality
- We need to test every aspect of the system and surrounding infrastructure under normal, failure and recovery conditions
- How will we generate a realistic load for testing ?
- How will we instrument the system and infrastructure ?



- How can we start to identify what the risks might be ?
- How can we look for single points of failure ?
- How can we look for modes of failure ?
- Can we identify specific scenarios of interest ?
- Can we test all the conditions ?
- What happens to our data ?

There are many techniques which have evolved over the years and there are tools to help you apply them.

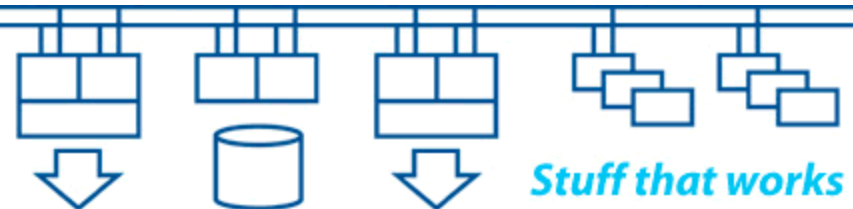
- Reliability Block Diagrams (RBD)
- Fault Tree Analysis (FTA)
- Failure Modes, Effects and Criticality Analysis (FMECA)

These (and many other) techniques can be applied with software tools available from a number of vendors.

- How do we instrument the system ?
  - Application level
  - Operating system level
  - Data storage level
  - Network infrastructure
- Time synchronisation
- How do we generate a typical workload ?
- How do we generate representative data sets ?

- Physical environment (power, cooling, etc.)
- Hardware (revision levels, labelling, etc.)
- Firmware versions and updates
- Operating system updates
- Network infrastructure and configurations
- Storage infrastructure and configurations
- Database versions and configurations
- Application software versions and configurations
- Interoperability and interactions
- Upgrade and backout actions
- Data conversions and one-way processes

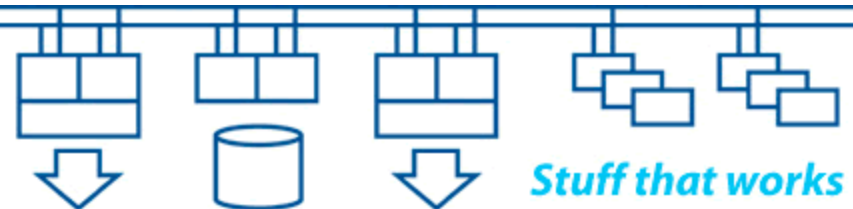
- Mission-critical systems hardly ever fail, so we need the people responsible for its operation to have a good understanding and ‘feel’ for the way it works
- We need to find out how the system behaves when it starts to fail
- We need to know the ‘warning signs’ of incipient failure
- We need to know how to return the system to its normal operational state without data loss or data corruption
- We must have a representative offline environment

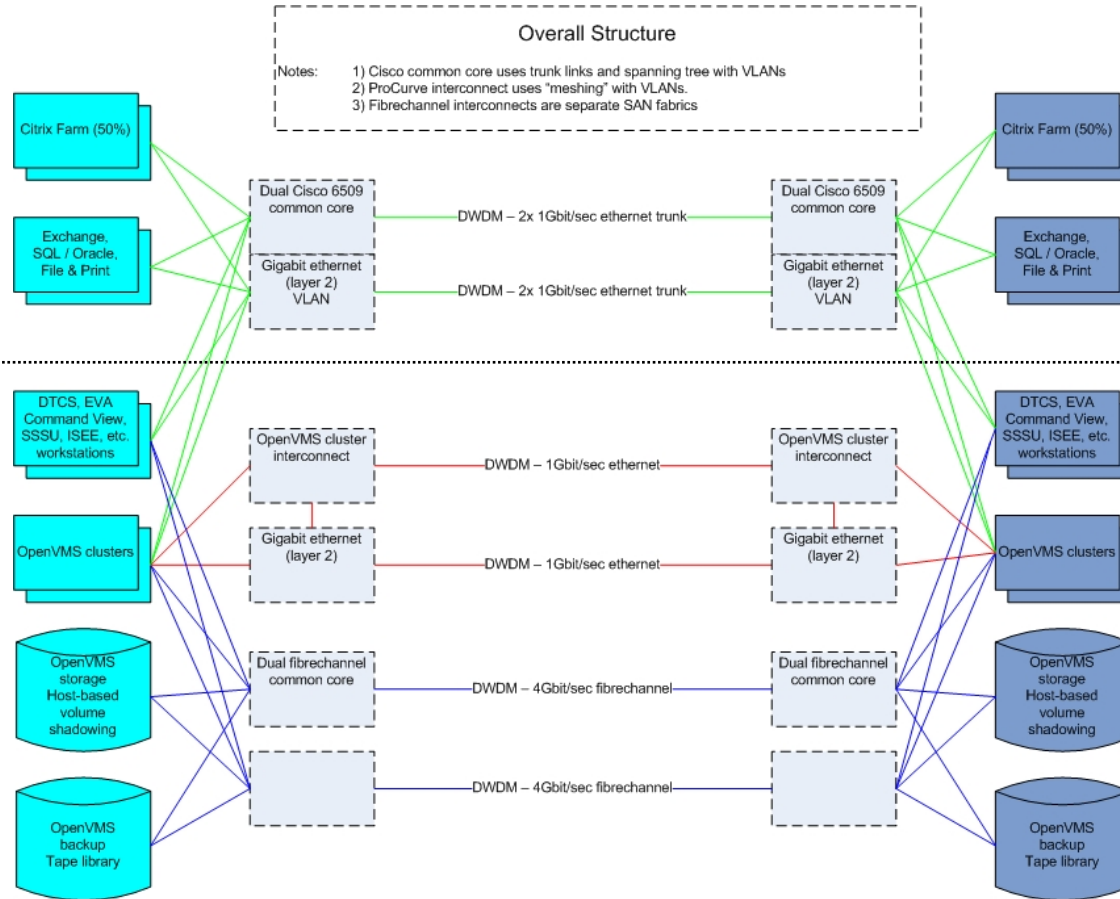


- Where can we do the testing ?
  - Production environment
  - Pre-production Test environment
  - Pre-delivery Test environment
- How might a problem show up – and when ?
- How can we find a problem, eg: data corruption ?
- Can we recreate a problem in a test environment ?
- Continual monitoring and event logging is essential
- Knowledge of the whole system is essential

An overview of how the new NHSBT Pulse systems fit into the NHSBT infrastructure.

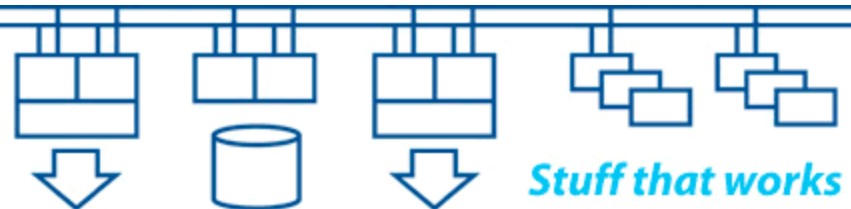
We have Production, Archive and Test environments with a shared common infrastructure, all of which is separated from the rest of the existing infrastructure.

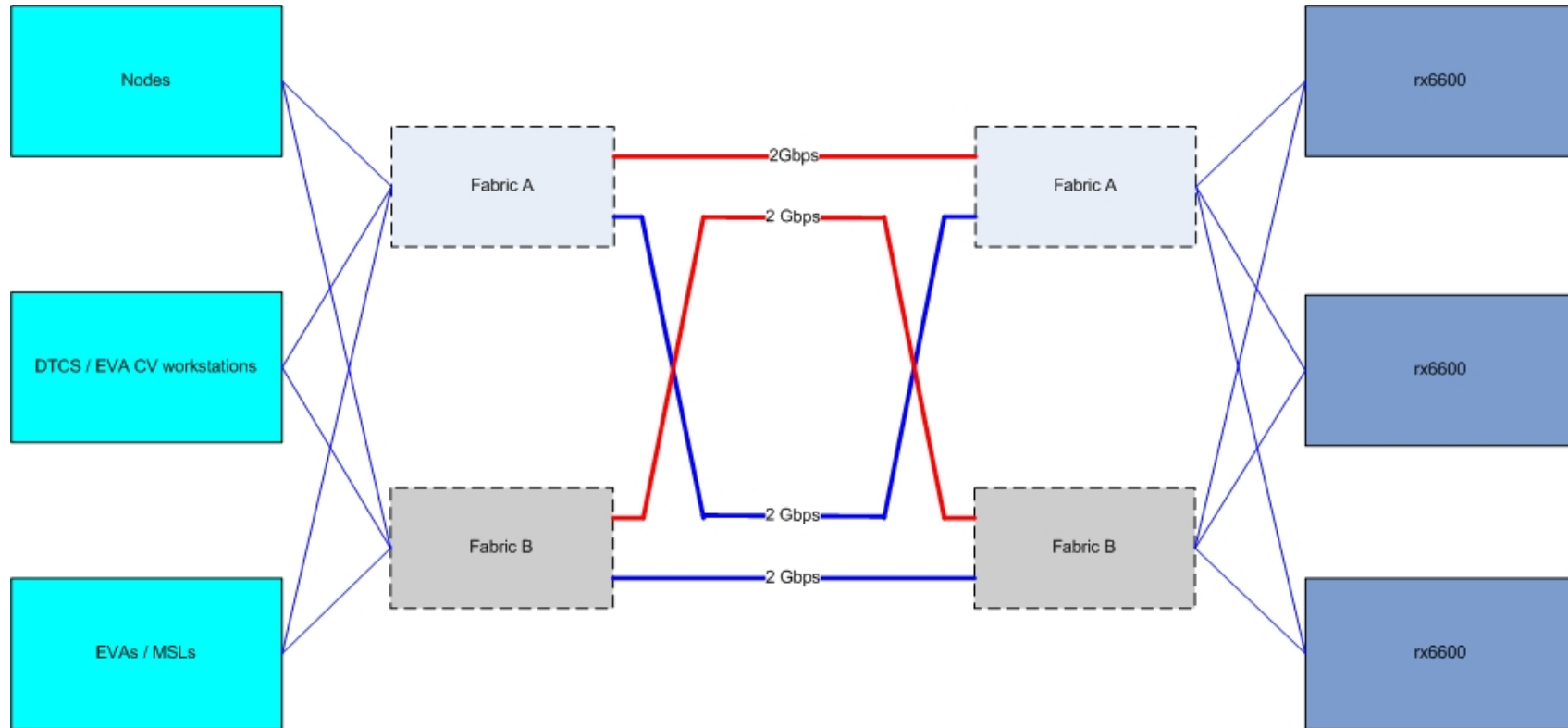




The systems are split up into:

- Common infrastructure (SAN fabrics, private network interconnects etc.)
- Production environment (a split-site cluster with host-based volume shadowed storage)
- Test environment (a split-site cluster on a smaller scale)
- Archive environment (a single node at Site A)
- Duplicated monitoring and reporting facilities
- External connectivity for users





**\*failsafe IP\*:**

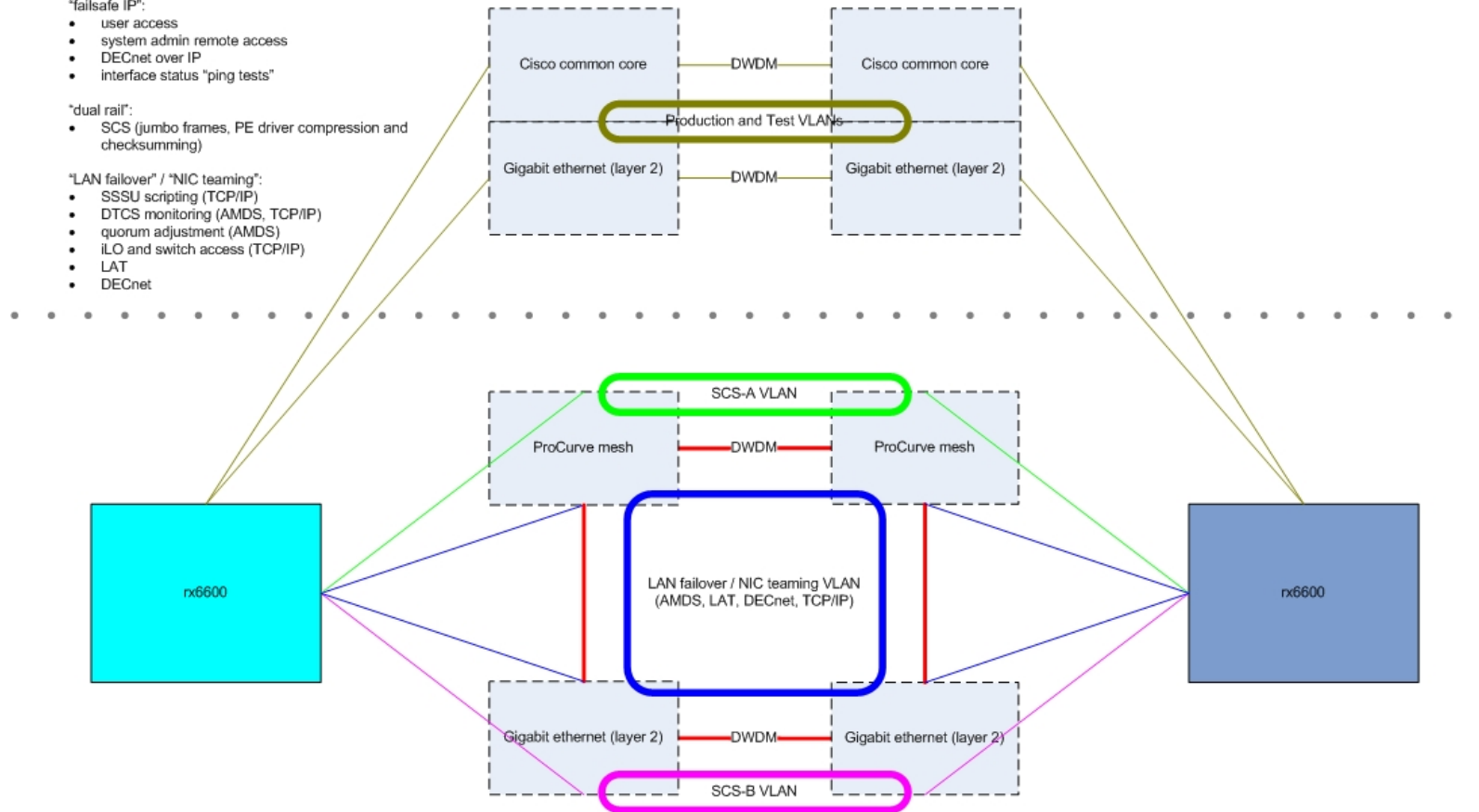
- user access
- system admin remote access
- DECnet over IP
- interface status "ping tests"

**\*dual rail\*:**

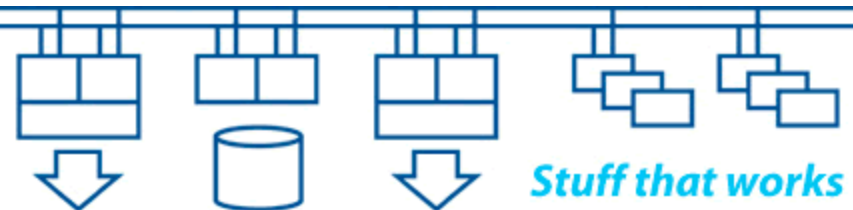
- SCS (jumbo frames, PE driver compression and checksumming)

**\*LAN failover\* / \*NIC teaming\*:**

- SSSU scripting (TCP/IP)
- DTCS monitoring (AMDS, TCP/IP)
- quorum adjustment (AMDS)
- iLO and switch access (TCP/IP)
- LAT
- DECnet



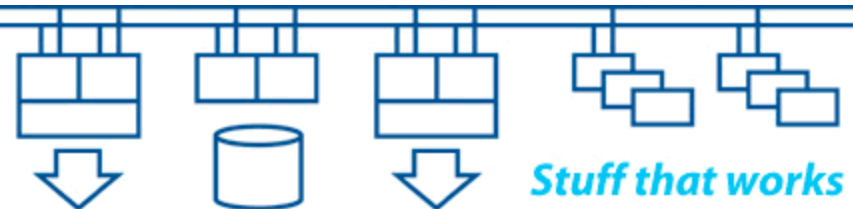
- Multi-site OpenVMS clusters give us “shared everything” access to data with protection from loss or corruption, even in the event of site failure
- Host-based volume shadowing (HBVS) ensures that data is consistent across all members of the shadow sets.
- The quorum scheme lets Site A continue if Site B fails and protects us from data corruption due to a partitioned cluster
- DTCS monitors the systems and (most important of all) controls the formation of storage shadow sets when the systems boot and when nodes rejoin the cluster



- DTCS is a set of HP and 3<sup>rd</sup> party products with installation, configuration and support services
- Remote console access, management and console output logging
- Integrated monitoring and quorum adjustment
- Rule based monitoring of individual systems / nodes
- Rule based SNMP polling of equipment
- Rule based TCP/IP “ping reachability” polling
- GUI and e-mail based alerting
- Scripting of failover and recovery actions across all systems / nodes and storage subsystems

- Small team of committed people
- Clear objectives
- Built 'proof of concept' data migration system first
- Built system 'on paper', discussed it extensively and resolved potential technical problems prior to purchasing equipment and building system platform
- Project management and planning
- Leadership and collaborative working
- Trust between team members
- Sufficient flexibility to cope with issues as they arise

- The art is to select elements that work well together and which provide the bulk of what you need with minimal additional work
- Establish the minimum requirements that have to be met – and do it as well as possible
- Availability and performance have to be designed in to the application
- Monitoring and automation are key components
- Understand the typical behaviour of your systems and be aware of changes





Thank you for your participation

Discussion!

