

Netherlands OpenVMS Technical Update

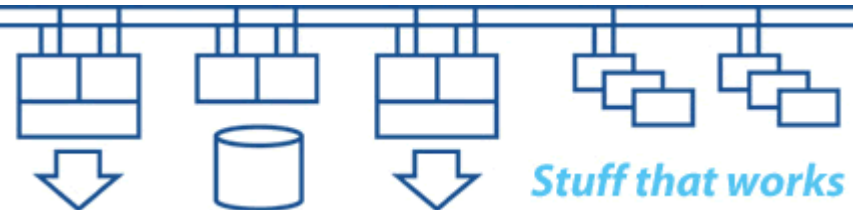
Availability & Performance: Designing and implementing multi-site disaster-tolerant environments

Colin Butcher

Copyright © Colin Butcher, XDelta Limited, October 2007

Web: <http://www.xdelta.co.uk>

E-mail: vms@xdelta.co.uk



Requirements:

What is the minimum we have to do extremely well?

Availability:

Disaster Tolerance = ability to survive major disruption

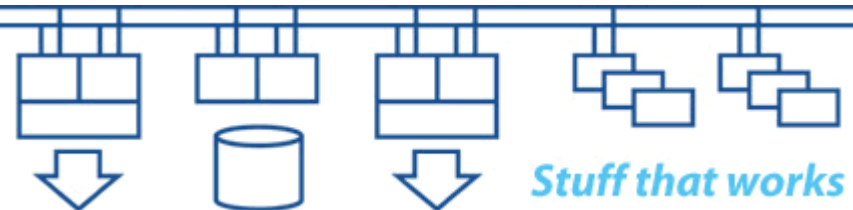
High Availability = ability to survive failures

Performance:

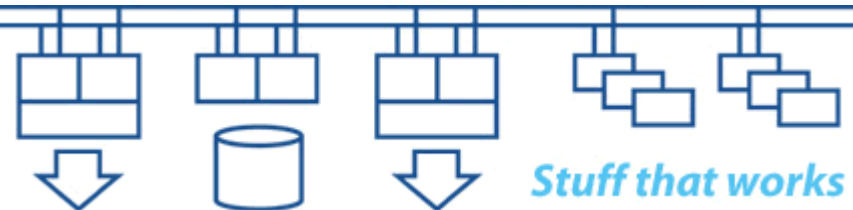
Performance issues are often the cause of transient system failures and disruption

Connectivity:

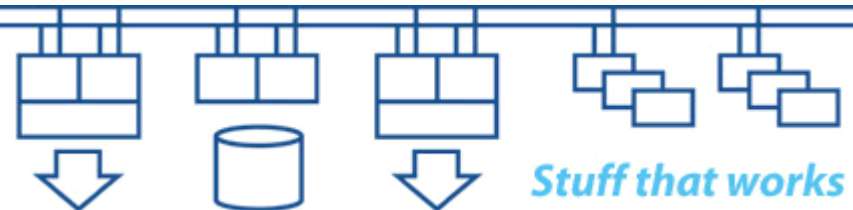
What kinds of access do we need? Users, Management, Remote backups, File and Data access, etc.



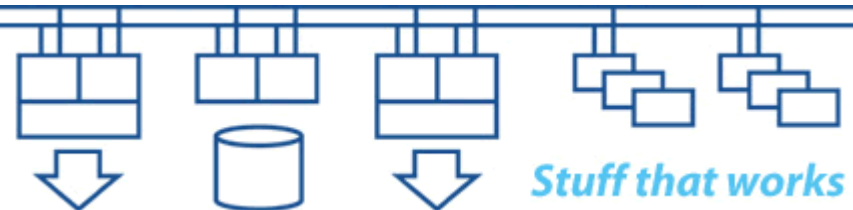
- **Performance**
- **Availability**
- **Connectivity**
- **Security**
- **Overall considerations**
- **Examples and discussion**



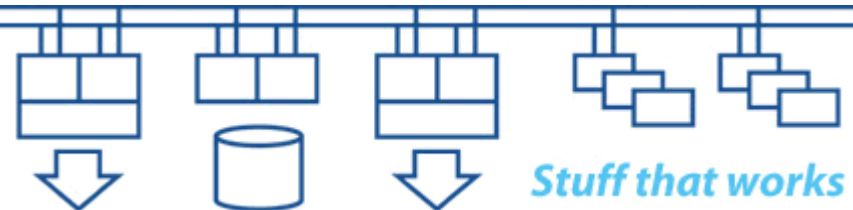
- **What does the business require the systems to do?**
- **What happens to the business if the systems fail?**
- **What happens if you push beyond the limits?**
- **How far from the edge are you?**
- **How do you know?**



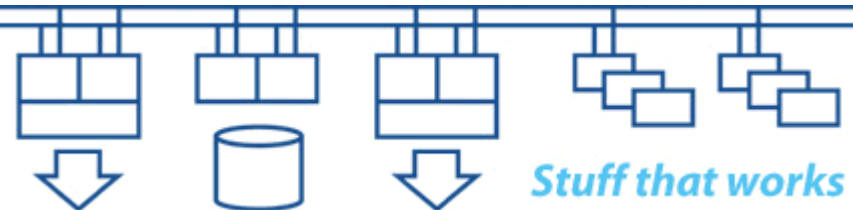
- **What is “fast” or “slow”?**
- **What are the performance requirements of the system we’re designing or investigating?**
- **What can we measure?**
- **What comparisons can we make?**
- **What “footprint” can we look for?**



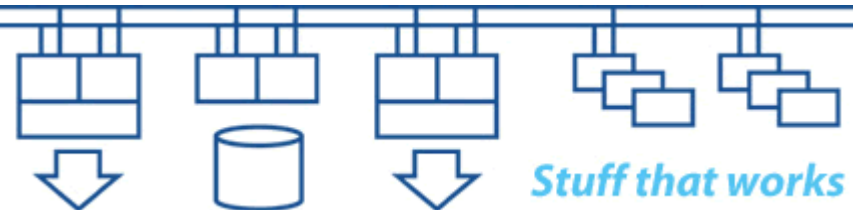
- **Bandwidth – determines throughput**
- **Latency – determines response time**
- **Jitter (“div latency” or variation of latency with time) – determines predictability of response**
- **Think about a motorway (freeway):**
 - **What is throughput?**
 - **What is latency?**
 - **What is jitter?**



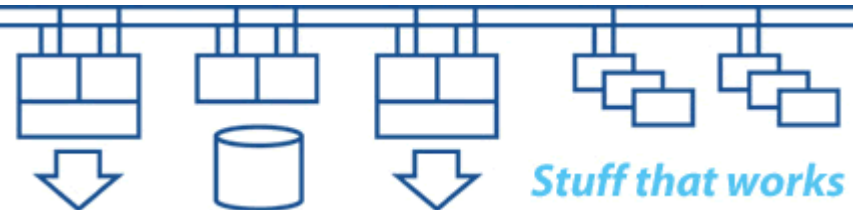
- **Bandwidth – determines throughput**
 - **Worst case is the maximum capacity of the smallest path in the system (the “bottleneck”)**
 - **It’s not just “speed”, it’s throughput in terms of “units of stuff per second”**



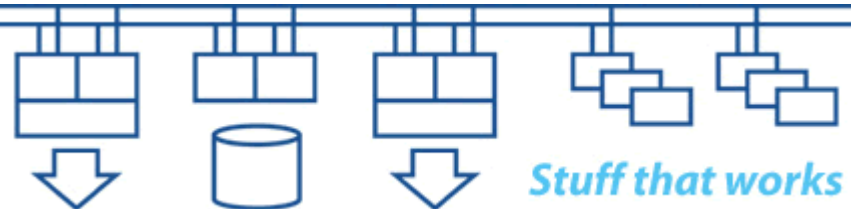
- **Latency – determines response time**
 - **Worst case is the sum of all the delays in the system**
 - **Latency determines how much “stuff” is in transit through the system at any given instant**
 - **“Stuff in transit” is the data at risk if there is a failure**



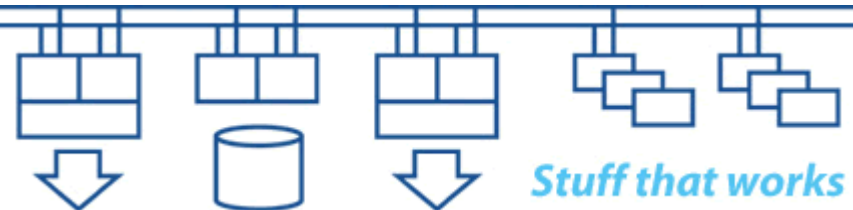
- **Jitter (“div latency” or variation of latency with time)**
 - **Determines predictability of response, ie: timeouts**
 - **Ideally zero**
 - **Wildly fluctuating latency is a “bad thing”**
 - **Latency fluctuations can cause system failures under peak load**



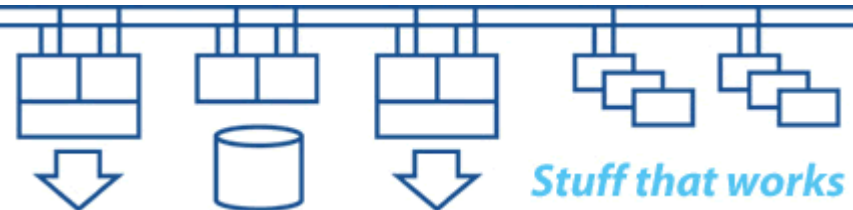
- **Time series performance data is just data – it does not tell us what to do**
- **Time series data sets allow us to make comparisons**
- **You must understand your workload**
- **You must understand what “normal” looks like**
- **There is no substitute for experience to interpret and analyse the data**



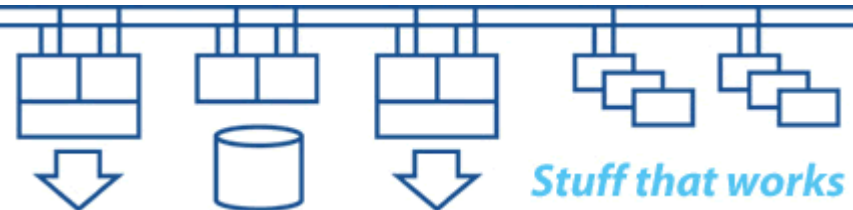
- **Size systems to cope with peaks in workload**
- **Minimise “wait states” (caches, parallelism)**
- **Scalability – do as much as possible once only, do little as possible every time**
- **Maximise “User Mode”, minimise the other modes:**
 - **The fastest IO is the IO you don’t do**
 - **The fastest code is the code you don’t execute**



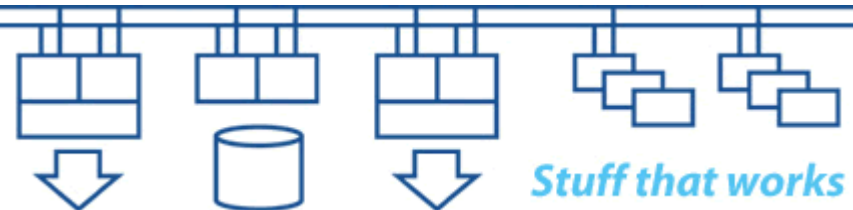
- **Ability to make use of hardware parallelism**
- **Granularity of data structures**
- **Synchronisation techniques**
- **Serialisation techniques**
- **Scalability techniques**
- **Compilers**
- **Application design**
- **Designing and writing very good code requires very good programmers**



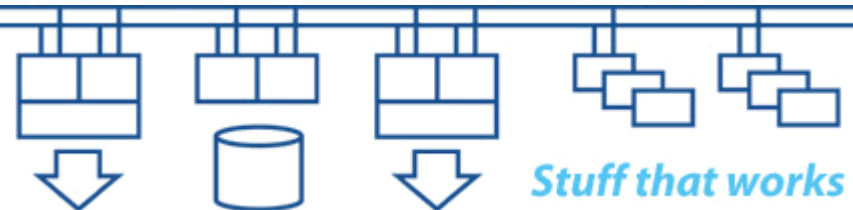
- **Availability is more important than performance**
- **We need to understand how systems fail and what failure looks like**
- **Business continuity:**
 - **It's not just the systems – it's everything!**
- **Disaster tolerance:**
 - **Surviving major site outages without loss of data**
- **High availability:**
 - **Surviving equipment and software failures**



- **How can we look for points of failure?**
- **How can we assess the impact of failure?**
- **Which parts of the system are mission-critical?**
- **What kind of failure do we prefer?**
- **What happens to our data?**



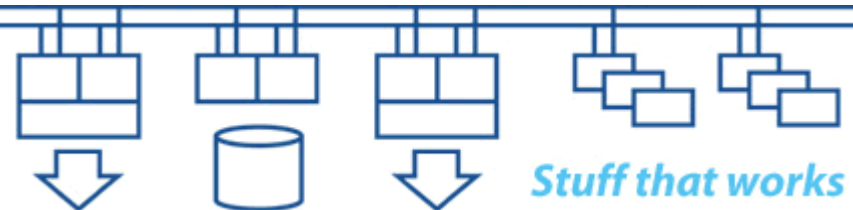
- **Reliability = Probability of failure at a given point in time, usually expressed as MTBF (Mean Time Between Failures)**
- **Availability = Probability of system being up and running at the instant when need it. Function of MTBF and MTTR (Mean Time To Repair), usually expressed as a % uptime, eg: 99.999%**
- **99.999% uptime (five nines) is equivalent to 5.26 minutes loss of service in a year for a 24x365 system. For a 12hour x 5day operational window it's only 1.87 minutes permissible outage in a year. That's difficult to achieve.**

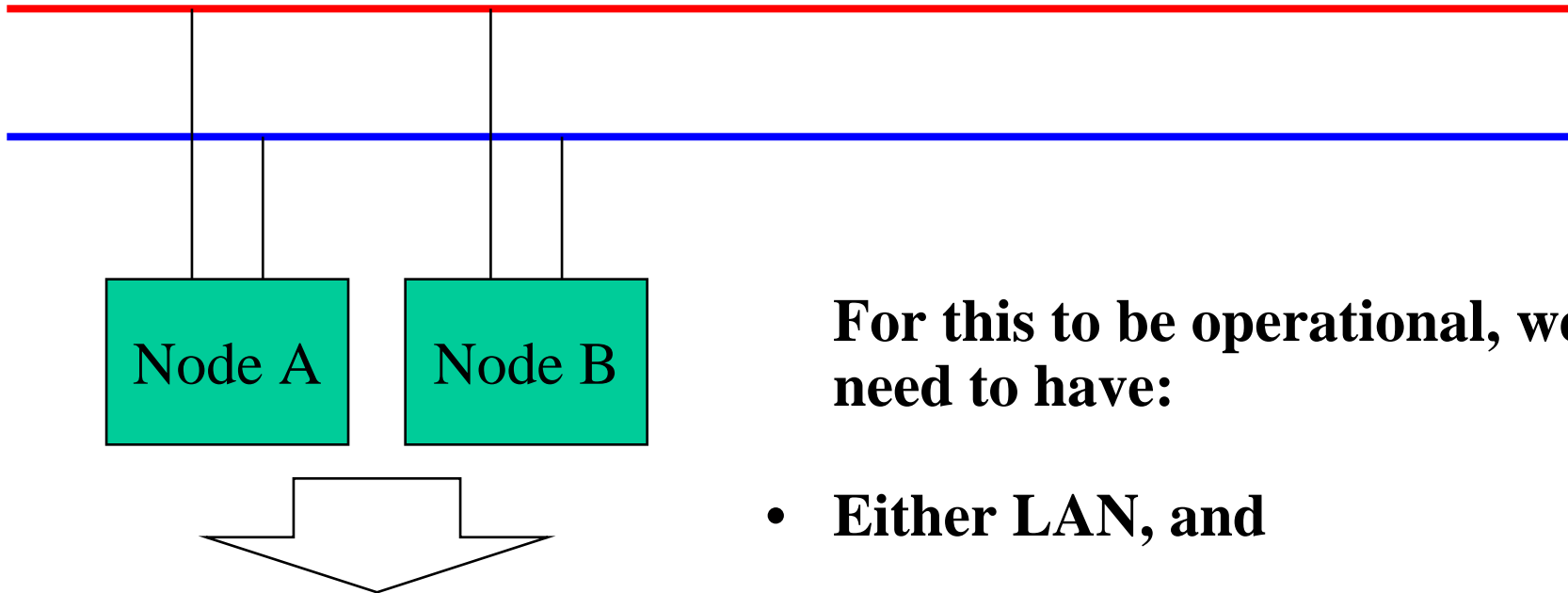


There are many techniques which have evolved over the years and there are tools to help you apply them. For example:

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

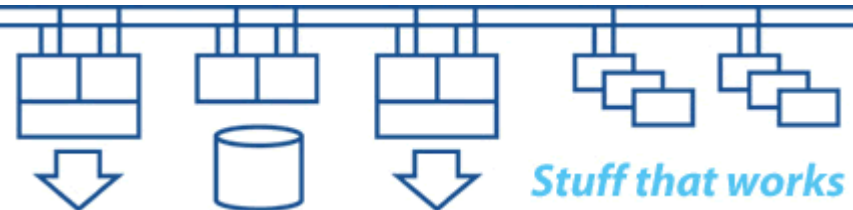
These techniques can be applied with software tools available from a number of vendors.



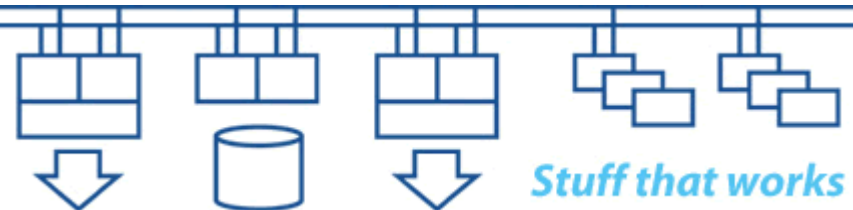
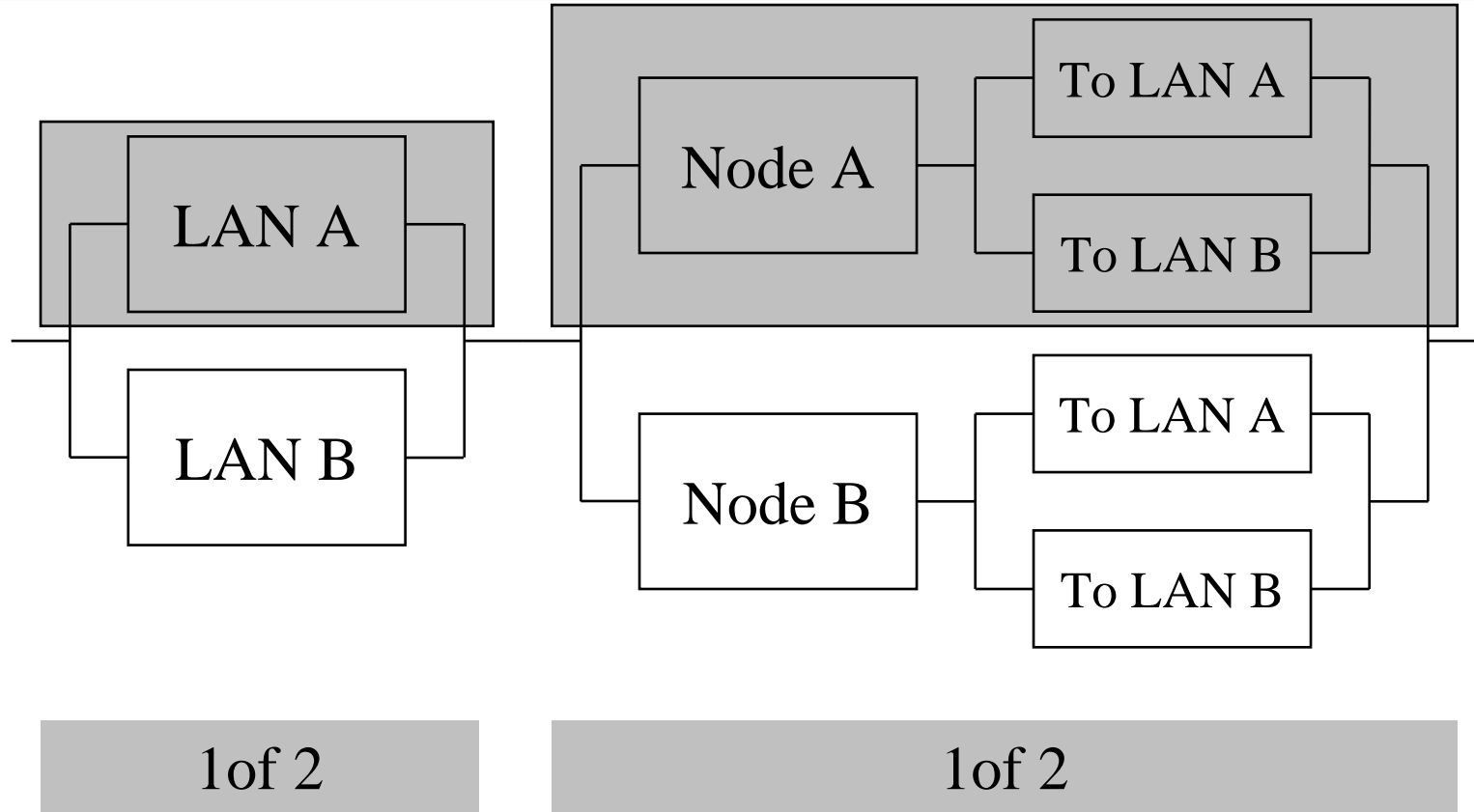


For this to be operational, we need to have:

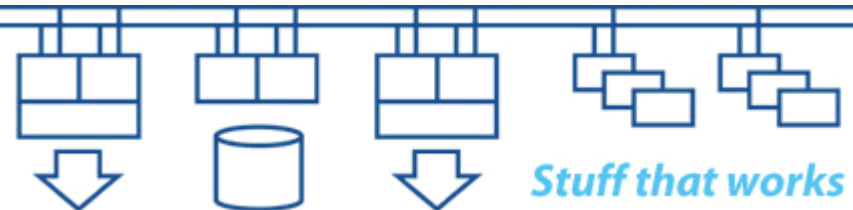
- **Either LAN, and**
- **Either node, which in turn needs either connection to either LAN**



Reliability block diagram (2)



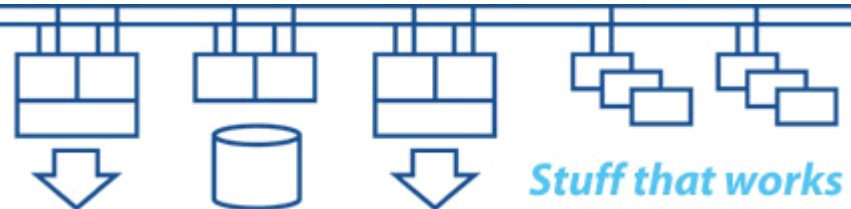
- **Used to identify single points of failure (SPOF)**
- **Can be used to derive an overall theoretical probability of failure for the system by assigning probabilities of failure to individual items**
- **Be aware that theoretical probability of failure calculations are based on statistics and assumptions – however the process is invaluable in understanding the issues**



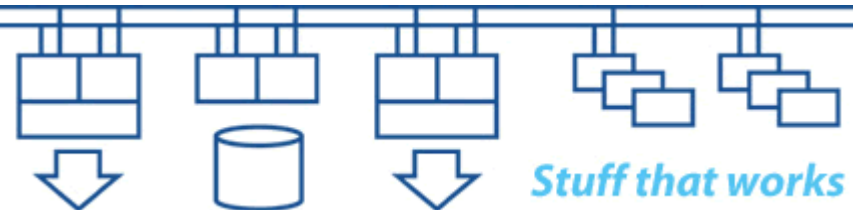
- **Failure Modes, Effects and Criticality Analysis (FMECA)**
- **Failure Mode and Effects Analysis (FMEA)**

These are techniques used to:

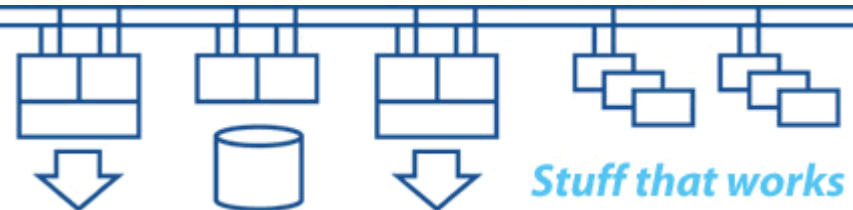
- **identify potential failure modes**
- **assess the risk associated with the failure modes**
- **sort the issues in terms of importance**
- **identify possible recovery actions**



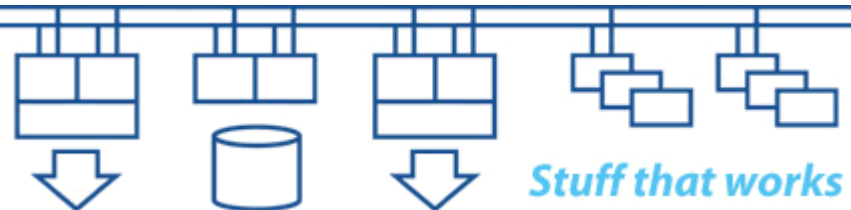
- **Fault Tree Analysis (FTA)**
- **Identify the way that failures can ripple through a system**
- **The “inverse” of a fault tree can help to identify “common mode” failure events and guide fault-finding, eg: loss of one phase can cause loss of power to many devices**



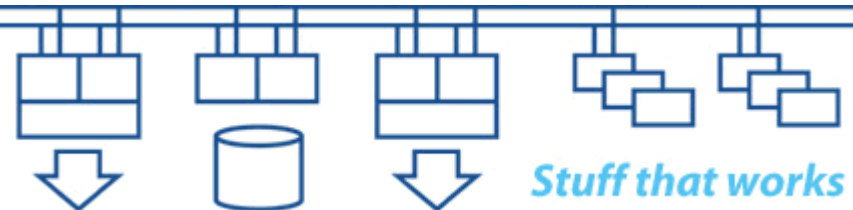
- **Take the example used in the Reliability Block Diagram where we have two machines in hot-standby operation**
- **What states can a pair of machines be in?**
- **We need to identify all possible states and ensure that “invalid states” do not occur (or are handled appropriately) and that the state information is propagated to all other participating machines in the overall system**



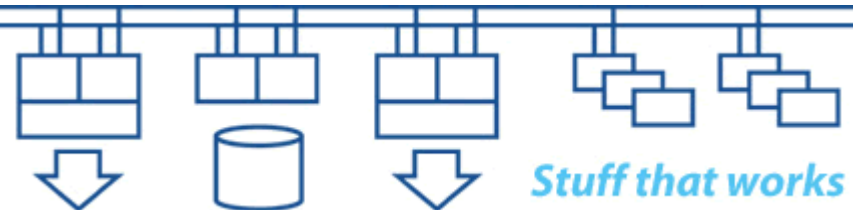
A	B
Off to Master	Off
Master	Off to Standby
Master to Off	Standby to Master
Master to Off	Off to Master
Master to Standby	Standby to Master
Master to Hung	Standby to Confused
And so on...	



- **Nothing happens instantaneously**
- **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 / flaws?**
- **How can we test it?**

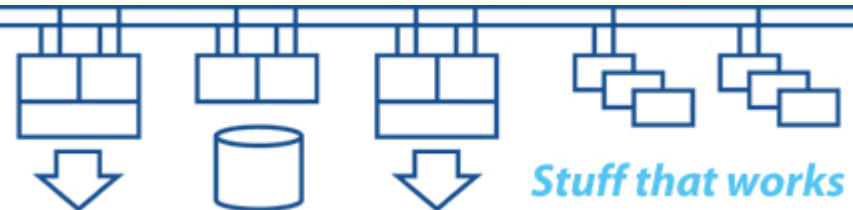


- **How can we get good information?**
- **What does “failure” look like?**
- **How quickly do we need to react to a failure?**
- **Should we automate decision making?**
- **How can we recover from failure?**

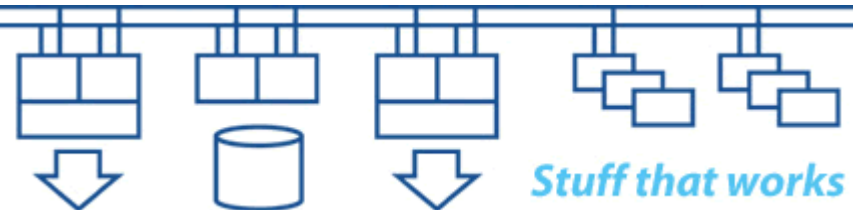


Relative time is more useful than absolute time

- **Need to be able to order events across the network based on timestamps**
- **UTC Timestamp format**
 - **Time value**
 - **Inaccuracy component**
- **External reference clocks**
- **NTP**
- **DTSS**

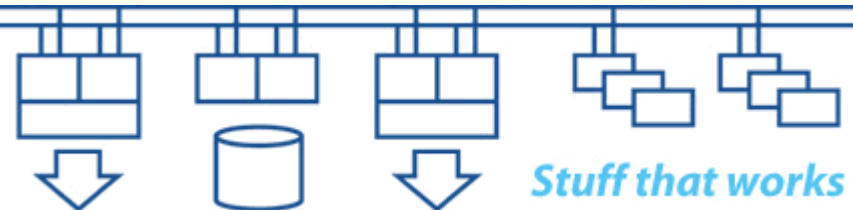


- **Network protocols used with OpenVMS systems**
- **Node naming, addressing schemes and routing mechanisms**
- **Multiple NICs and multiple LANs**
- **Map functions to NICs:**
 - **Management (ILO, SAN appliance, etc.)**
 - **Clustering**
 - **Network backups**
 - **Data transfers (eg: FTP, NFS etc.)**
 - **Interactive users**



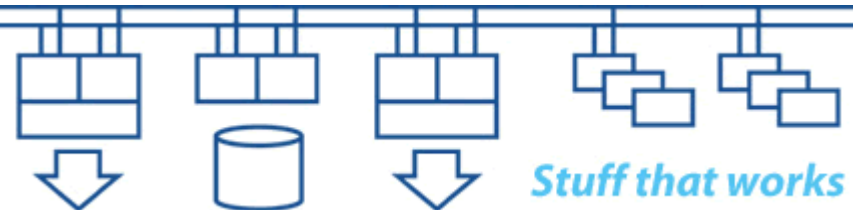
Typical network protocols in use with OpenVMS systems:

- **SCS (clustering)**
- **TCP/IP (and all it's components)**
- **DECnet-Plus (NSP,OSI and DECnet over IP) or DECnet Phase IV (NSP only)**
- **DECdns (not to be confused with TCP/IP's DNS/BIND)**
- **LAT (DECserver terminal access etc.)**
- **MOP + Remote Console (DECserver, LAVC boot etc.)**
- **DTSS (can be disabled)**
- **LAD + LAST (Infoserver etc.)**

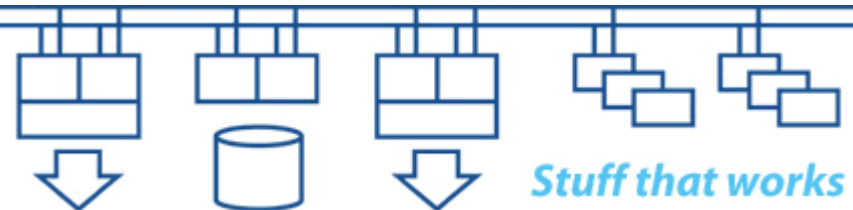


Using VLANs to segment a network:

- **Implemented within core switches**
- **Layer 2, Layer 3 etc.**
- **Port based VLANs**
- **Protocol based VLANs**
- **Connectivity between VLANs**
- **VLAN tagging of packets (802.1Q)**
- **VLAN tagging out of the NICs**

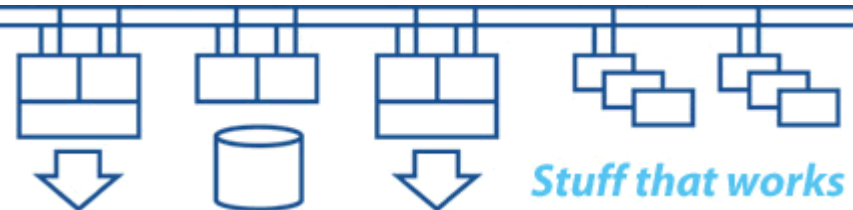


- **LAN failover (LLdriver)**
- **TCP/IP - Failsafe IP**
- **SCS – stopping and starting per adapter (SCACP)**
- **DECnet Phase IV and V - load balancing**
- **MOP and LANCP (network booting)**
- **LAD / LAST (InfoServer)**
- **LAT (DECservers)**

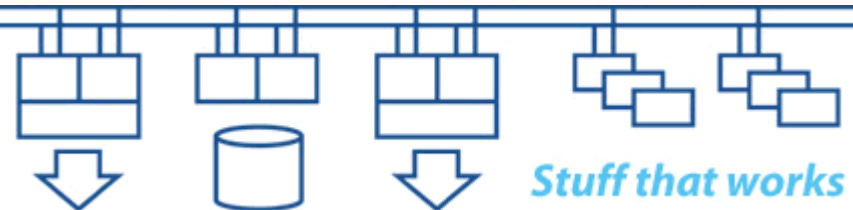


Mission critical systems need to be able to:

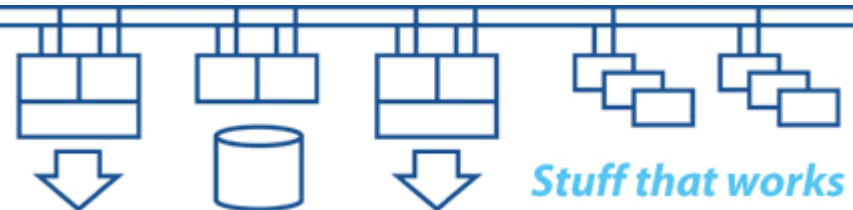
- **Survive failures (resilience and failover)**
- **Survive changes (adapt and evolve)**
- **Survive people (simplify and automate)**
- **Never corrupt or lose critical data (data integrity)**
- **Requirements never remain static over an extended period of time, so we need to be able to make changes 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**



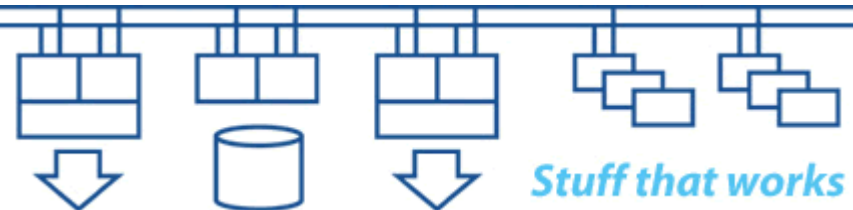
- **Safety-critical systems (especially safety-critical real-time monitoring and control systems such as air traffic control) require exceedingly high levels of availability. They also have to be fail-safe in order not to endanger lives.**
- **True 24x365 mission-critical systems are fairly rare. With 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!**
- **The closer you get to 100% uptime the more expensive a satisfactory solution will become.**



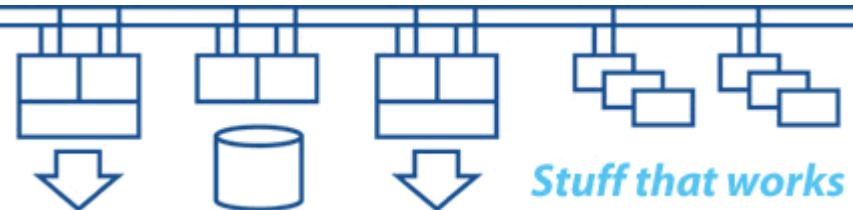
Cause of Outage:	Planned (Maintenance)	Unplanned (Failure)
Hardware	?	?
Operating System	?	?
Network Layer	?	?
Layered Products	?	?
Application Software	?	?
Application Data	?	?
Environment	?	?
Staff	?	?



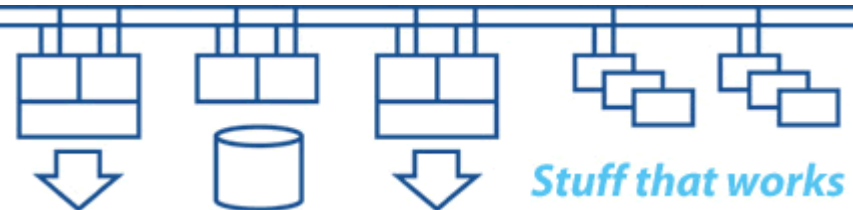
- **Big decisions which have long-term implications and constraints, eg: disc block size = 512bytes**
- **Small decisions which seem big at the time, eg: memory page size = 512 bytes, now variable**
- **Requirements you don't yet understand or know about**
- **Design for change**

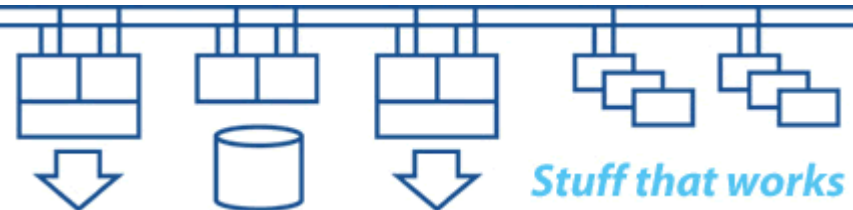
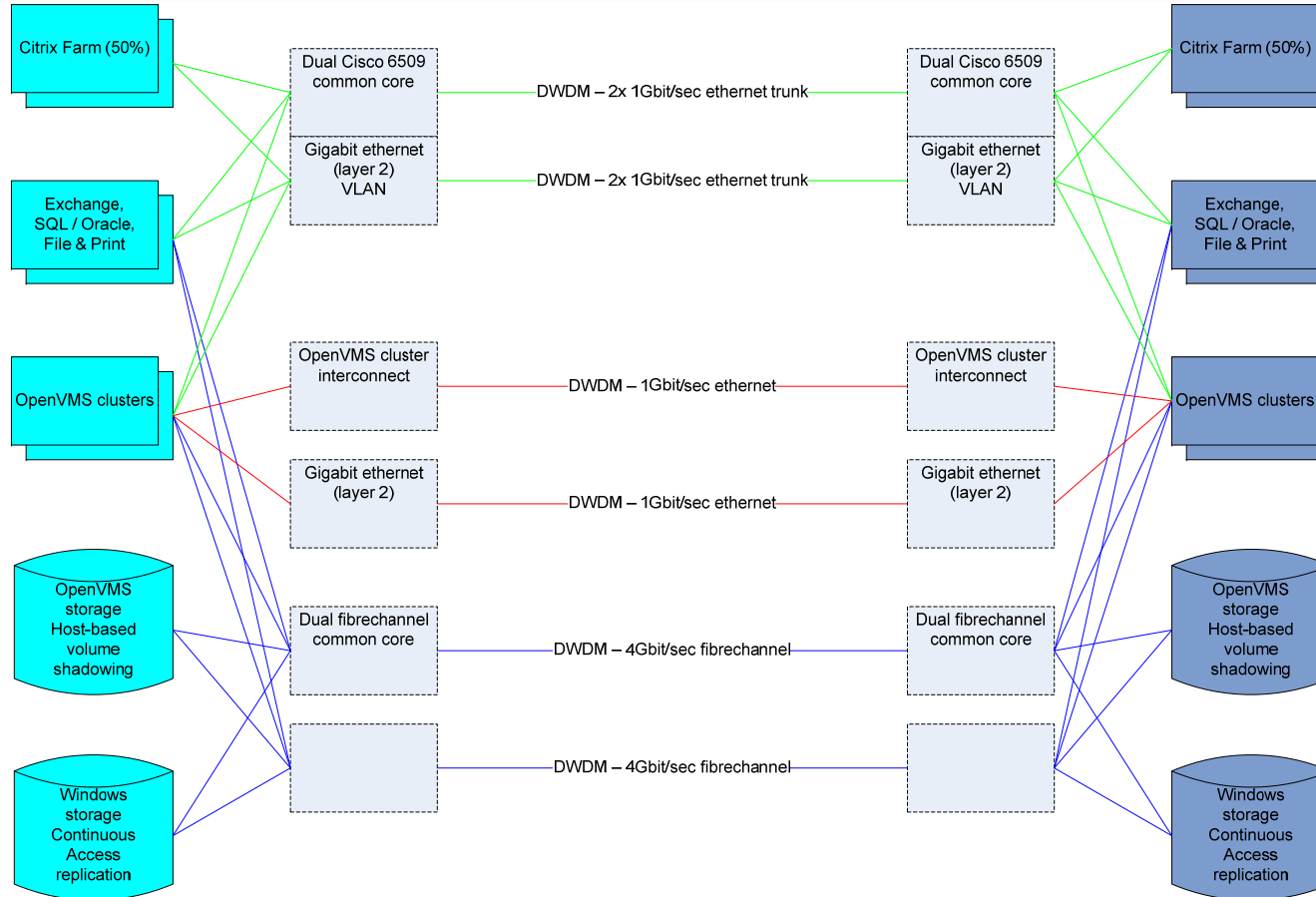


	Before	Now	After
Environment			
System			
Component			

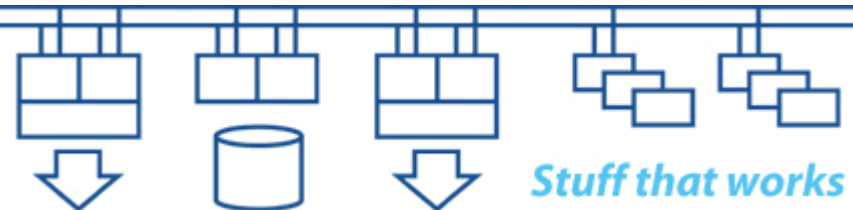


- **Safety-critical and mission-critical system:**
 - **Port from Alpha to Integrity**
 - **Move from 3x clusters to single Integrity cluster**
 - **Move from HSG80s to EVA4100s**
 - **Move to multiple NIC connectivity**
- **Similar principles apply in many other cases, eg:**
 - **Satellite flight control**
 - **Finance data processing**
 - **Process control**

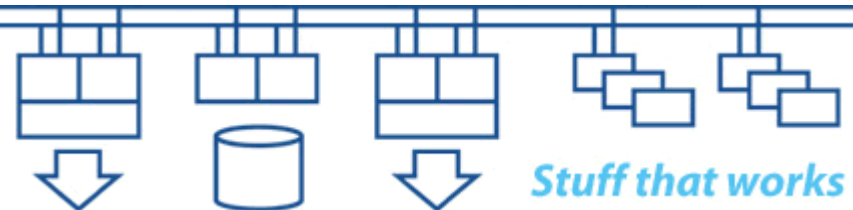




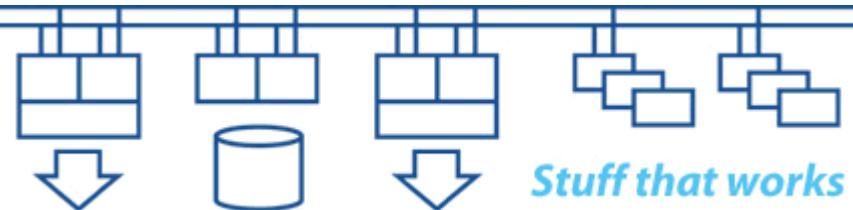
- **System configuration – the art is to select components that work well together and which provide the bulk of what you need with minimal additional work**
- **Establish the minimum requirements that have to be met – and do it as well as possible**
- **Availability and performance have to be designed in to the application**
- **Monitoring and automation are key components**
- **Understand the typical behaviour of your systems and be aware of changes**
- **Configuration control – hardware, settings, software etc.**



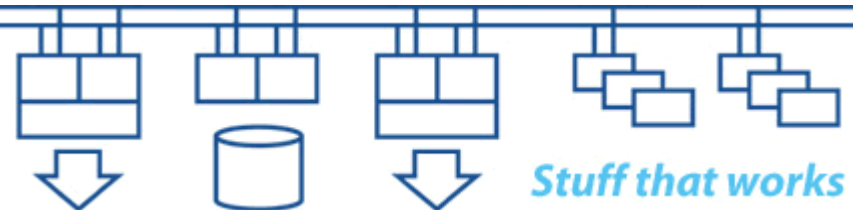
- **Use multiple systems within a site – and have appropriate physical separation between them**
- **Use multiple sites with appropriate geographical separation**
- **Understand what happens when a failure occurs and how the overall system configuration is likely to respond**
- **Avoid the risk of more than one system thinking that it's in charge when a failure has happened**
- **Ensure that the surrounding infrastructure and environment is appropriate to the needs of your systems**
- **Configure the individual components of the systems in a manner that maximises interchangeability**



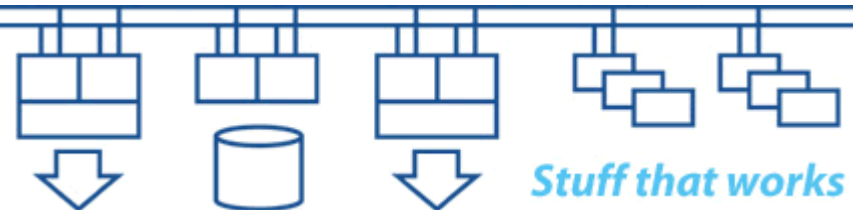
- **Losing data is a disaster**
- **Availability is more important than performance**
- **Size storage subsystem based on minimum components and maximum estimated throughput**
- **Segment storage subsystem to provide gradual degradation rather than wholesale failure**
- **Need adequate backup capacity and throughput in order to meet permissible backup windows**
- **Understand application behaviour and storage performance requirements (bandwidth and latency)**



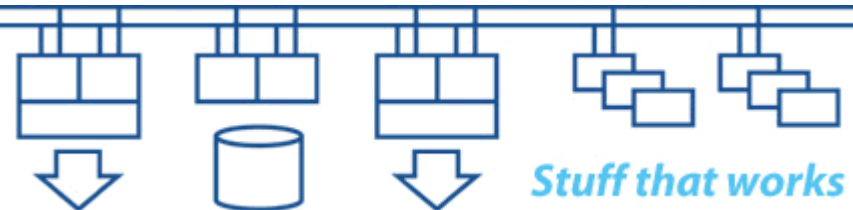
- **Scale network so that overall performance is based on minimum essential number of paths and maximum estimated traffic**
- **May wish to take advantage of installed bandwidth capacity to provide additional functionality when everything is working**
- **There is no such thing as a single protocol network**
- **Understand the behaviours of the different protocols under failure conditions**
- **Segment the network to provide gradual degradation rather than wholesale failure**



- **Operating system design**
- **System configuration**
- **Performance tuning**
- **Application design (scalability, data integrity, availability)**
- **Compilers generate code better than you can**
- **Use the OS features (eg: lock manager)**
- **Portability**
- **Reliability**
- **Supportability**
- **Testing and documentation**



- **Think big, implement small**
- **Documentation**
- **Portability**
- **Modularity**
- **Reliability**
- **Scalability**
- **Software build process**
- **Software installation process**
- **Configuration control**
- **Testing (development, pre-deployment, post-deployment)**
- **Support and planning**



Thank you for your participation

Q & A

