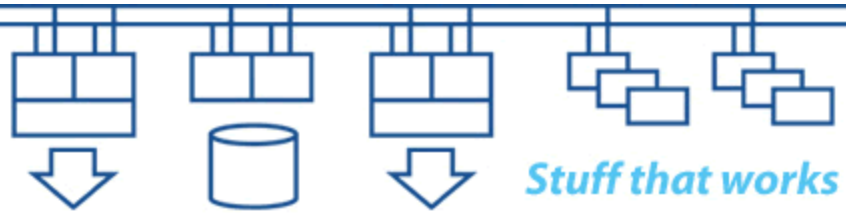
BCS Reading Branch

Disaster-tolerant systems

Colin Butcher



BC, DT, HA and DR (1)

- 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 without loss of data

BC, DT, HA and DR (2)

- 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!

Risk

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.

The risk continuum

- 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

- How can we start to identify what the risks might be?

Systems Engineering and Risk

- We're trying to reduce the probability of failure
- We're trying to minimise the consequences of a failure
- We're looking for critical components / people
- We need to understand how systems fail and what failure looks like when it happens
- We're paranoid about our data

Problem solving concepts - TRIZ

	Before	Now	After
Environment			
System			
Component			

Design decisions

- Big decisions which have long-term implications and constraints
- Small decisions which seem big at the time
- There will be requirements and constraints you don't yet understand or know about
- Need to design in the ability to make changes
- Must establish a meaningful naming convention

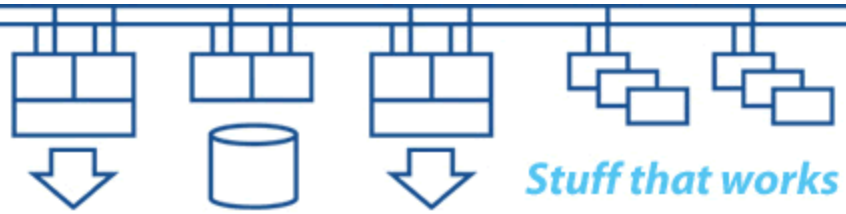
Risk assessment

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

Mission Critical systems

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)
- Be easy to maintain and work on
- 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



Do we really need 100% uptime?

- 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!
- The closer you get to 100% uptime the more expensive a satisfactory solution will become.

RPO and RTO

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?

"Survivability test"

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

The systems "flight envelope"

- 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?

Availability and performance

Availability:

- Business Continuity = ability to continue business operations
- Disaster Tolerance = ability to survive major outages (eg: loss of a complete site)
- High Availability = ability to survive equipment failures (typically within a site)

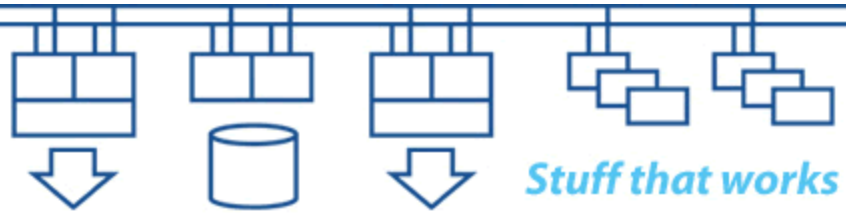
Performance:

- 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

Availability is more important than performance

Throughput and response times

- 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

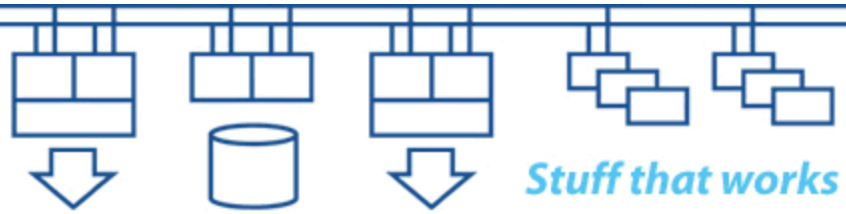


Designing for performance

- Size systems to meet realistic criteria, eg: throughput; response time; minimal loss of "data in transit"; etc.
- Understand how the applications could break down into parallel streams of execution
- Understand scalability – do as much as possible once only, do little as possible every time
  - The fastest IO is the IO you don't do
  - The fastest code is the code you don't execute
- Understand the need for synchronisation and serialisation of access to data structures
- Minimise "wait states" and contention
- How will you generate a realistic load for testing?

Designing for availability

- Which parts of the system are mission-critical?
- What kind of failure do we prefer?
- What happens to our data when things go wrong?
- What state transitions occur during failure and recovery?
- How can we recover from a failure without data loss or data corruption?
- Should we automate decision making?
- How can we get good information?
- How will you test your failure scenarios?
- How can we maintain and work on it safely?



Multi-site issues

- Effects of distance on network and storage protocols
- Symmetric or asymmetric operation – how good is your "crystal ball"?
- Avoid booting across inter-site links
- Remote access for management and operation
- Centralised (and duplicated) monitoring and alerting
- Naming conventions
- Quorum and voting scheme
- Host-based volume shadowing scheme
- Full environmental monitoring for lights-out sites
- Avoid automation of decision making when a site fails

Testing - proving that it all works

- We need to test for scale as well as functionality
- We need to test every aspect of the system and surrounding infrastructure under normal, failure and recovery conditions
- We need to understand the underlying cause of problems so that we can fix them or avoid them
- We need to prove that service will continue with minimal disruption during failure and recovery
- We need to know how to recover from failures without loss of service and without data corruption or data loss
- We need to regularly rehearse and test our procedures and plans to ensure that we stay current

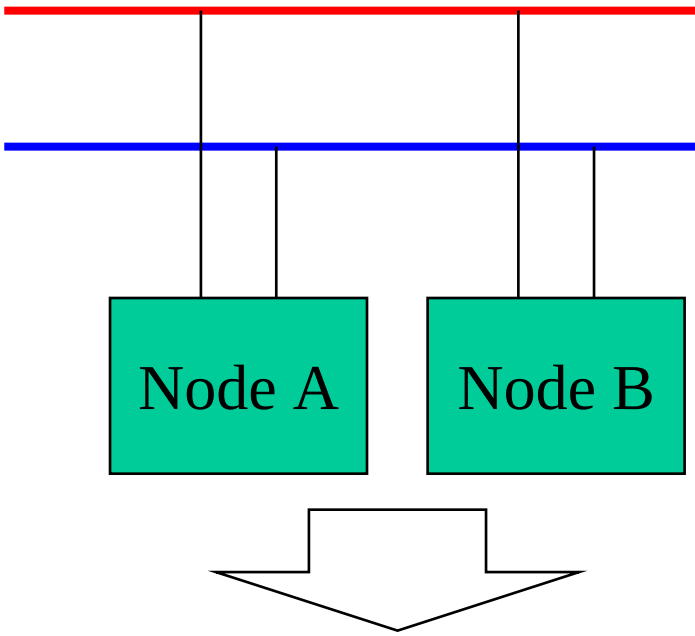
Analysis Techniques

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.

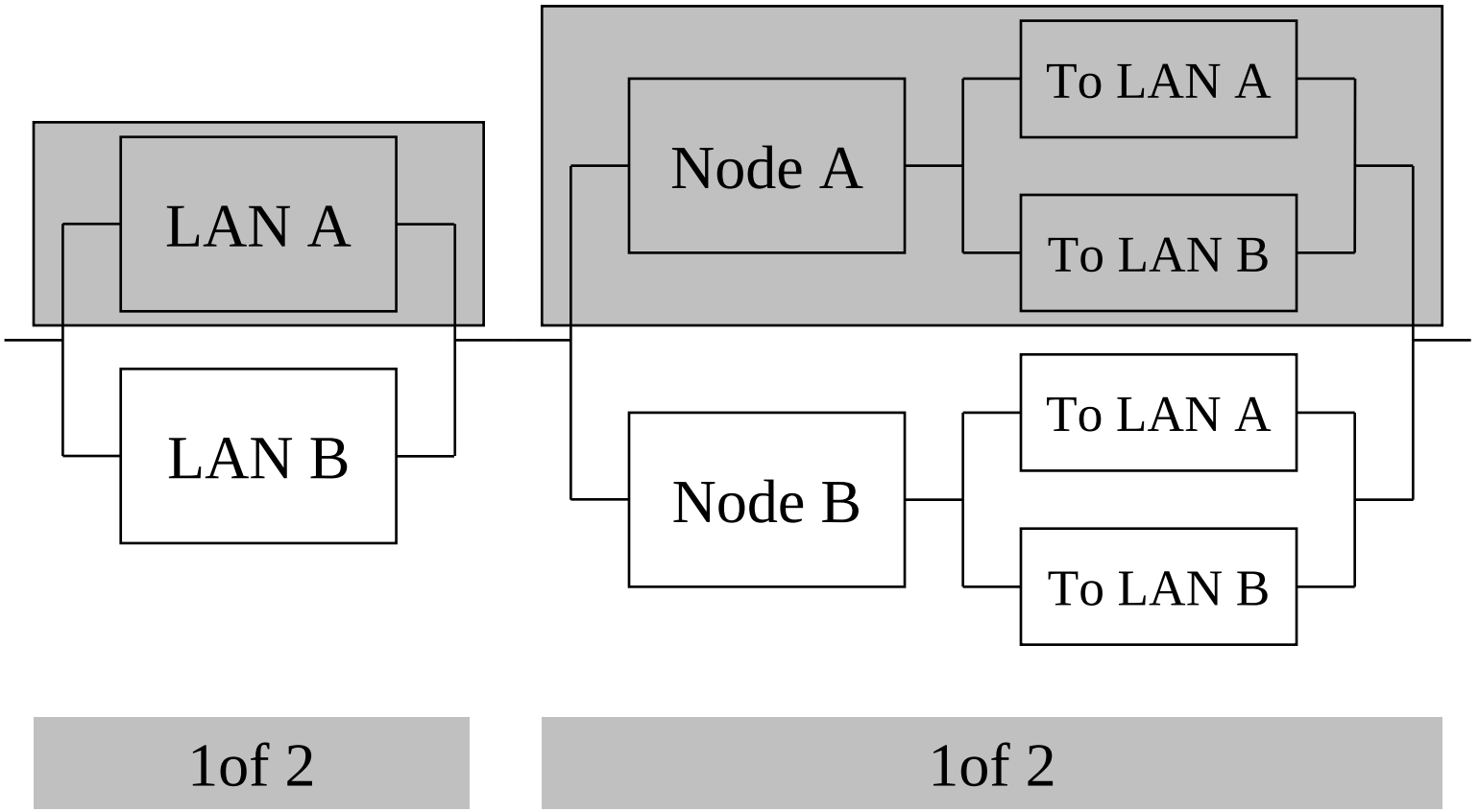
Reliability block diagram (1)



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)



Reliability block diagram (3)

- 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

FMECA and FMEA

- 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

- 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

State transitions (1)

- 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

State transitions (2)

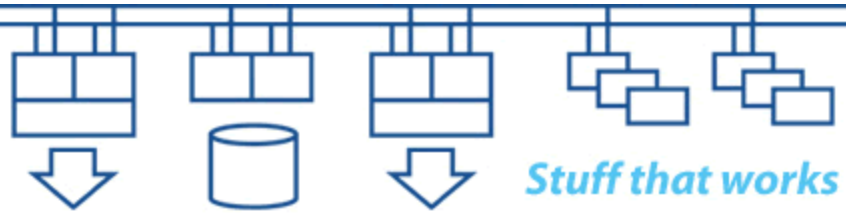
A	B
Off to Master	Off
Master	Off to Standby
Master to Off	Standby to Master
Master to Off	Off to Master
Master to Standby	Standby to Master
Master to Hung	Standby to Confused
And so on…	

State transitions (3)

- 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?

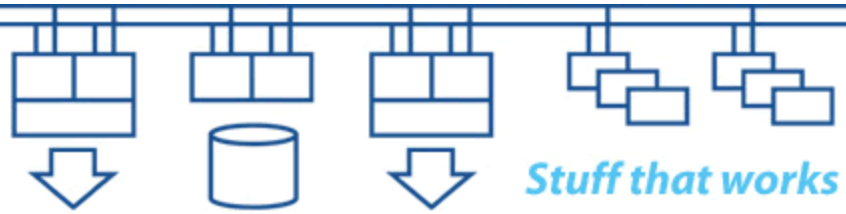
Dealing with failures

- 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?



System configurations

- 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



Key issues

- You can't buy it off the shelf
- Do the minimum you have to do and get it right
- Design for change on-the-fly with no loss of service
- Documentation and configuration control
- Protect the data and ensure that it's consistent
- Monitoring, information and automation
- Testing and continual training
- Procurement
- Project management
- Leadership and collaborative working

Thank you for your participation

Discussion!