UKCMG Annual Conference 2007

Principles of Availability

Colin Butcher

Overview

Overall system availability depends on matching the system configuration to the requirements of the application. In order to build a successful high-availability system it is necessary to understand the causes and effects of system failures and disruption.

Performance and capacity constraints are often a major cause of system failures. High availability systems require the performance characteristics of a system to be well understood, especially during recovery from failure.

Principles of Availability

An overview of availability analysis, state transitions and failure detection.

- **We're looking for single points of failure**
- **We're looking for critical components**
- **We need to understand how systems fail and what failure looks like when it happens**
- **We're paranoid about our data**

Availability in context

- **Availability is more important than performance**
- **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**

Availability 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?
- What kind of failure do we prefer?
- What happens to our data?

Reliability and Availability

- **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.**

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)**

- **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 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.**

"Survivability test"

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

Design decisions

- **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**
- **There will be requirements and constraints you don't yet understand or know about**
- **Need to design in the ability to make changes over time**

Problem solving concepts

	Before	Now	After
Environment			
System			
Component			

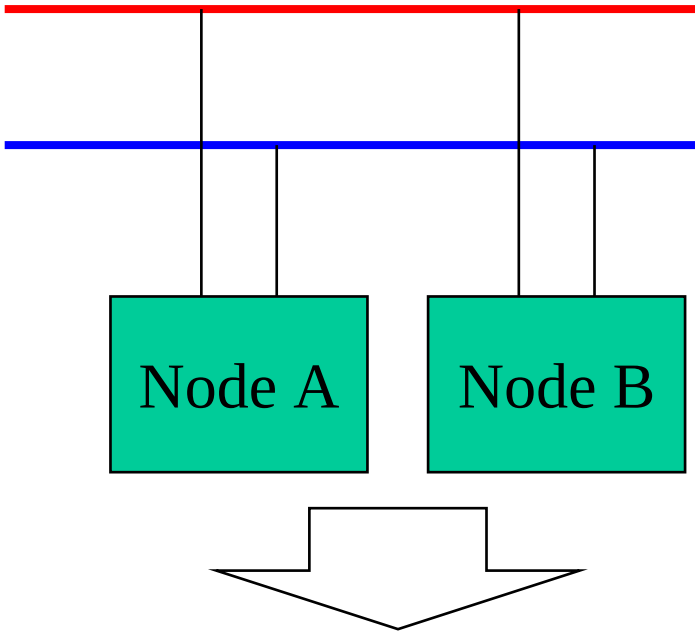
Techniques

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.

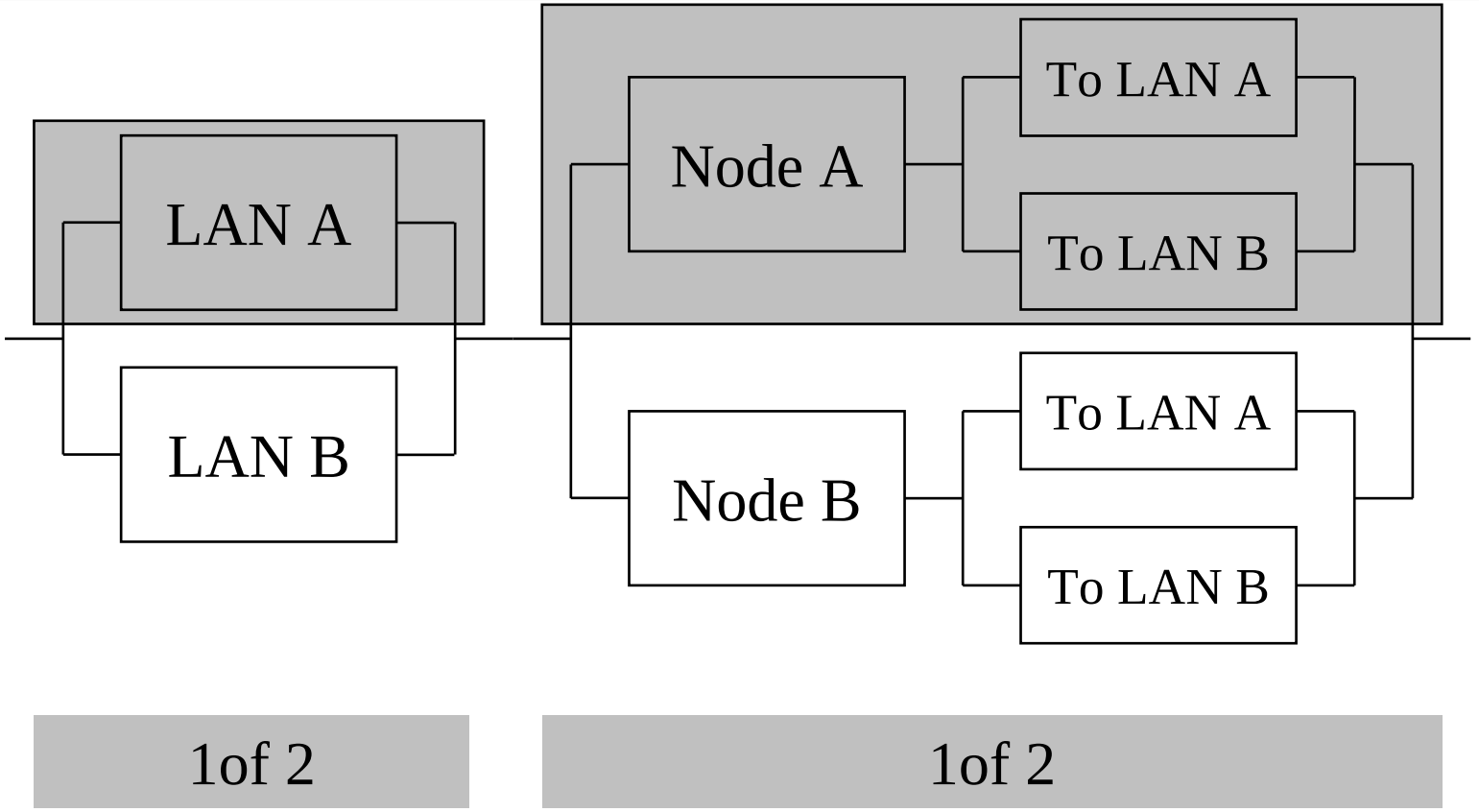
Reliability block diagram (1)



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

- 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.

Systems infrastructure

- **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**

Storage infrastructure

- **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)**

Network infrastructure

- **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**

Summary

- **Think big, implement small**
- **Documentation is vital (the “what”, “how” and “why”)**
- **Concentrate on the essentials**
- **Testing (development, pre-deployment, post-deployment)**
- **Planning and “what if” scenarios**

Thank you for your participation

Q & A