

RapidIO® as the Foundation for Fault Tolerant Systems

By Victor Menasce

At one time fault tolerance was considered the bane of the lunatic fringe, telecom central office applications, the space shuttle, and so on. Today, the requirement for high availability is more like a check-box (an expected feature). It's easy to see why. A packet can commonly traverse over 20 or more network elements going from source to destination. A common way to deal with the availability expectation is to build in hardware redundancy, which carries extra cost, has extra fail over latency, and means that extra capacity must be reserved for failover.

In addition, while the trend in the network is toward a flatter network, significant hierarchy still exists. The problem with this hierarchy is that it adversely affects reliability. Let's look at a simple example:

Our example network element has an availability probability of 0.99. A network made up of two of these elements has an availability of $0.99^2 = 0.9801$. A network made up of 20 of these elements has an availability of $0.99^{20} = 0.82$, which most users would consider blatantly unacceptable.

This fact means that each individual network element must achieve sufficiently high availability so that the product of the availabilities yields an acceptable result at the network level. This has become the generally accepted 5 9's, or 0.99999 availability. It is important to note that this availability is measured at the application level. This level of availability equates to five minutes of outage downtime per system per year. Let's examine a typical analysis of fault tolerant network element failures shown in figure 1. The systems analyzed in figure 1 achieve 5 9's of availability [1].

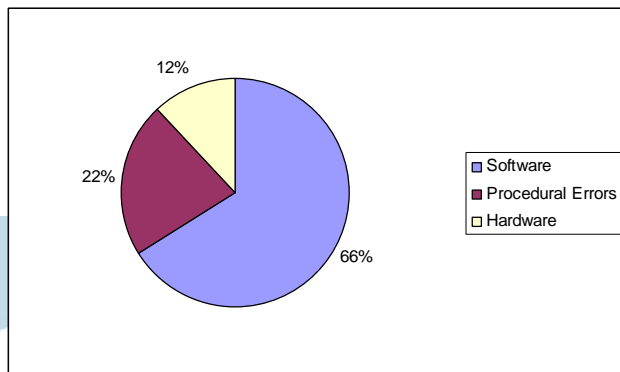


Figure 1: Network element failures

The largest proportion of failures is software, for which redundancy is difficult and costly to design. The second largest element is procedural errors. This class of failure is the result of maintenance people making a wrong decision and causing a network outage as a result. The smallest portion of the failure budget is indeed the hardware. Clearly, in order to achieve 5 9's overall, the hardware needs to be at least an order of magnitude better.

The impact of these outages in terms of outage downtime is affected by the time to repair a given outage. Catastrophic software failures can often be recovered by a system restart. This results in an outage on the order of two to three minutes. However, an outage that requires human intervention to repair the problem can have a significantly longer mean time to repair. Therefore, the frequency of hardware outages must be reduced even further to lessen the impact of the statistically longer mean time to repair.

The outage budget can be broken down further into planned and unplanned outages. Some systems are capable of in-service upgrades, but this is rare. This is analogous to replacing the engine on an aircraft while it is in flight. If we allocate three minutes of the outage downtime to planned outages, then the remaining two minutes can be absorbed by unplanned outages. This means that the hardware needs to deliver better than 24 seconds of outage downtime per system per year. This equates to 7 9's, or 99.99994% uptime. This is extremely challenging to achieve.

Design engineers have traditionally relied on proprietary technology to achieve this kind of performance. However, as companies look to rely increasingly on off-the-shelf technology, industry standards need to incorporate these features. No-where is this more important than in the arena of system interconnect. The interconnect architecture can become the weakest link, and also forms the backbone of the system's fault tolerance architecture.

The need for high availability has expanded to include storage systems, servers, network routers and switches.

The RapidIO interconnect was developed from its inception with high availability applications in mind. This philosophy permeates the architecture and the specifications. However, there are many legitimate architectures for achieving fault tolerance. The different architectures trade off the level of hardware redundancy, system cost, complexity and availability. The interconnect architecture should not assume any one fault tolerance architecture. It should provide the necessary hooks so that any of these fault tolerance architectures can be supported.

Examining System-Level Fault Tolerance

There are six key elements to system-level fault tolerance:

- No single point of failure
- No single point of repair
- Fault recovery
- 100% fault detection
- 100% fault isolation
- Fault containment

Let's examine what each of these means from a system level perspective.

No Single Point of Failure

No single point of failure is quite simple really. It means that you need redundant hardware that can take over in the event of a failure. There are a number of legitimate, but very different sparing architectures.

Examples include:

- Redundant hardware, load shared
- Duplex hardware, hot / cold standby
- Duplex hardware, hot / warm standby
- Triplex hardware with voting
- N + M sparing

The RapidIO architecture supports all of these different sparing schemes.

No Single Point of Repair

No single point of repair is a little more subtle requirement. It means that you shouldn't have to shut down or replace a critical component in order to replace another failed component. An example of a common single point of repair might be a backplane. Live insertion/removal of field replaceable units (FRUs) is an integral piece of meeting this requirement.

The RapidIO specification supports live insertion and removal through both electrical and logical mechanisms. Live insertion is required to ensure that no single point of repair exists in the system.

The links are point to point and as such they do not have the same impact that a shared bus such as PCI may have. Use of extended power pins is common on backplanes but fall short of the required protocol to bring a new FRU into service in a system.

The logical protocol for managing state information is defined in the RapidIO Error Management Specification. Insertion and removal of a blade is considered an exception condition and the software API is defined in this section of the specification. Bringing a new FRU into the system must be done carefully. It must be thoroughly tested and configured by the host in the system, prior to being brought into service. New FRUs are not trusted until proven trustworthy.

At system boot time, the system host identifies all of the unattached links in the machine using the system discovery mechanism and puts them in a locked mode. At this point all incoming packets are rejected, leaving the drivers and receivers enabled. This ensures that whatever new FRU is inserted cannot access the system until the system host permits such access. When a FRU is hot-inserted connecting to a switch device, the now connected link will automatically start the training sequence. When training is complete, the locked port generates a Maintenance port-write operation to notify the system host of the new connection, and sets the Port-write Pending bit.

On receipt of the port-write, the system host is responsible for bringing the inserted FRU into the system in a controlled manner. The system host can communicate with the inserted FRU using Maintenance operations after clearing all error conditions, if any. This procedure allows the system host to access the inserted FRU safely, without exposing itself to incorrect behavior by the inserted FRU.

In order to achieve the first three basic elements, a strong foundation of the last three elements is essential. This is where the features of an interconnect standard can make the task of achieving fault tolerance easy, or nearly impossible. One of the key items is support for live insertion and removal of field replaceable units.

Fault Recovery

Fault recovery requires the ability to switch from a failed piece of hardware to a working piece of hardware with little to no interruption of the application. It can also represent the ability to withstand a transient failure at the transaction level. The ability to recover from a fault is entirely dependent on the ability to satisfy the remaining fault tolerance elements. Without this, fault control hardware cannot make informed decisions on the appropriate corrective action.

100% Fault Detection

This requirement is essential to determining that a piece of hardware has failed or a transaction has been lost. There can be no datagram transmission of data in the system. This means that all data paths, transactions and storage elements need to be protected by parity or some type of error detection code such as a CRC. This detection mechanism needs to be able to report the error to a control entity when the error occurs. As such it cannot rely upon the transport mechanism that itself may be at fault.

Faulty transactions need to be traceable to the offending transaction. If the exception reporting is imprecise, recovery will be virtually impossible and the only remedy will be a system reset which will result in an outage of the duration of the restart time.

The RapidIO interconnect provides a rich array of fault detection mechanisms. First, all transactions are protected by a CRC code. All handshake control symbols are protected by a 5 bit CRC, or are transmitted twice. In addition, all transactions must be positively handshaked and strictly ordered. This ensures that dropped transactions do not permit propagation of logical layer failures.

The serial RapidIO protocol is 8B/10B encoded. This provides an additional layer of fault detection in addition to the CRC protecting the packet. A single bit error can manifest in illegal codes that are detected at the PCS layer of the PHY. In addition, the K-codes that were chosen for use in the RapidIO standard were chosen specifically so that no single bit error could be misinterpreted by the PHY logic as a special use character. Other interconnect standards such as PCI Express, FibreChannel and Ethernet have this vulnerability.

Some system failures are visible as a degradation in reliability, prior to a catastrophic failure. For example, the bit error rate on a link may degrade gradually over time. Detection of this trend can provide an early warning of a bad link before the failure becomes a hard failure. The RapidIO interconnect supports a rich array of performance and reliability monitors. These monitors have multiple thresholds. The traffic light analogy works well here. Working hardware is denoted by a green light. A hard failure is denoted by a red light. A degraded link falls into the yellow light category. This can give maintenance software plenty of time to take corrective action prior to the degradation becoming a hard failure. Thresholds can be set by software to determine how what level of protection is optimum for a given system.

100% Fault Isolation

Fault isolation usually requires a strong partnership between hardware and maintenance software. The only exception to this is in the case of triplicated hardware where a simple vote determines which of the three sets of hardware is at fault. This is a costly method. Most fault tolerance architects prefer duplicated hardware or an n+m redundancy approach. Let's examine how this might work.

In a duplex system, two identical hardware components compare against one another executing the same application. As long as they agree, the hardware is assumed to be good. At some point, in the presence of a failure, the two hardware components diverge. This is an excellent way to detect an error. However, how do you know which of the two pieces of hardware to trust? This is where fault isolation enters the picture.

At the point of detecting the failure, maintenance software must take over and look for signs of trouble in the recent history. If one of the two pieces of hardware handled an exception, and the other did not, then there may be something for software to investigate. The only way that software can isolate the failure is if the system maintains a history of what was happening in the recent past. The time between a failure occurrence and the detection of the failure should be very short. A shorter interval minimizes the amount of transaction history that needs to be maintained.

Isolating the error is key to understanding the severity of the error, and hence the severity of the recovery process that must be initiated. This can determine whether a transaction needs to be restarted, a software process, or the entire system. By limiting the scope of recovery, the time impact of the failure can be minimized. This will aid in the uptime statistics. Restarting a software process is much faster than a heavy-handed reset of the entire system.

Transactions in the RapidIO standard are handshaked at the physical layer. The transmitter of a transaction on a RapidIO link is required to maintain a copy of the transaction in its output buffer until it receives positive acknowledgement of receipt from the receiver. If that acknowledgement indicates an error, or the acknowledgement is not received in time, the packet is retransmitted. In this case, the RapidIO link provides detection, isolation and recovery, all at the hardware level with no software intervention.

Fault Containment

All RapidIO links have full protection at the physical layer. This ensures that a failure cannot propagate beyond a single RapidIO link. However, sometimes the failure is in the end-point and is not associated with the link.

In this instance, the failure can propagate beyond the failed component. A rogue transmitter can resist attempts to cease transmission or reset the component. The resulting surge in traffic can cause congestion, buffer overflows, and ultimately data loss. The best protection for this class of failure is to ensure that propagation of the failure does not occur.

RapidIO switches use a table based routing algorithm. The source and destination addresses in the specification get mapped to specific port numbers. These mapping tables can be programmed to isolate the traffic from a rogue transmitter away from harming the system. Switches from most RapidIO vendors are able to support this kind of asymmetric mapping of address ranges to ports.

Conclusion

Fault Tolerance is now a mainstream requirement of many network elements, wireline, wireless, voice and data. The features required of fault tolerant systems must be embedded into the underlying hardware technologies that are used in those systems. The RapidIO interconnect was tailor-made to suit the needs of these demanding applications. And there's a reason why. The architects of the RapidIO interconnect's fault tolerant features (hailing from companies such as Cisco, EMC, IBM, Lucent, Mercury Computer, and Motorola) are experts in high-availability systems and applications.

References

[1] The data for this article was obtained from the US Federal Communication Commission (FCC) ARMIS database by the author. The FCC is an independent US government agency that regulates interstate and international communications by radio, television, wire, satellite and cable. Access to its ARMIS database is publicly available on the internet (www.fcc.gov).

The raw data used in the article is available in the database, although there are no public reports that match the information presented directly here. The analysis was performed independently by the author who consolidated a range of widely distributed data from the database for the purpose of illustration. It is possible, however, to commission private reports from the FCC that detail data for specific vendors.

About RapidIO Technology

The RapidIO interconnect architecture, designed to be compatible with the most popular integrated communications processors, host processors, digital signal processors, and bridge chips, is a high-performance, packet-switched, interconnect technology. It addresses the embedded industry's need for high performance, low overhead, reliable serial interconnect technology.

Interested companies are invited to join the RapidIO Trade Association, a non-profit corporation controlled by its members. Go to: WWW.RAPIDIO.ORG

