the Availability Digest

Avoiding Notworks

January 2009

A network that doesn't work is a "notwork."

The Impact of a Notwork

Active/Active Systems

In an active/active system,[1] in which two or more nodes cooperate in a common application using a distributed database, network faults can wreak havoc. One set of nodes can become isolated from others, requiring that the isolated nodes be taken down if split-brain operation is to be avoided. If the isolated nodes are not taken down, perhaps data-collision resolution[2] will be required when the network is returned to service.

If synchronous replication is being used between the active/active nodes, network errors or increased channel latency can cause applications to time out and crash.

Active/Backup Configurations

Network faults can cause similar problems even in an active/backup configuration, in which a backup system is available to take over the role of the active system should it fail. The backup system must have access to a reasonably current application database copy that it can use if it has to take over processing. To the extent that the backup database copy is not current, transactions will be lost upon failover. The degree of acceptable data loss is specified by the Recovery Point Objective, or RPO.

If the backup database is being kept current via virtual tape or by data replication, both of which replicate changes from the active system to the backup system over the network (as opposed to the use of magnetic tape), a network failure will cause the backup database to get further and further behind, thus perhaps violating the RPO specification.

User Access

In all of the above scenarios, an available network is required if users are to be moved from an inoperative node or system to a functional one. If the network should fail and then a node in an active/active system or the active system in an active/backup pair should fail, users cannot be reconnected to an operating node or system to continue their service. These users are now down as well.

In this article, we review at a high level some of the steps that should be taken to ensure that long-haul wide-area networks used in an active/active system do not become the limiting factor in

[1] What is Active/Active?, *Availability Digest*; October 2006.
[2] Asynchronous Replication Engines, *Availability Digest*; November 2006.

system availability or performance. Many of these considerations also apply to local area networks that connect users to their local processing nodes.

Network Availability

Split-Brain Operation

Split-brain operation may be acceptable for those cases in which collisions cannot occur or are not important.

In some active/active applications, data collisions do not occur. An example of such an application is an insert-only application. Since no data is being updated, collisions cannot occur.

These applications can work perfectly well in split-brain mode. Should the network fail, operation continues unaffected until the network is restored. The only ability that is lost is the ability to run queries against a current database until the network is restored and the database copies are resynchronized. If query processing is not critical, network availability is not critical. Besides, queries can be satisfied from the local database though they may return stale results.

If data collisions can happen, they can be avoided by partitioning the database so that each partition is owned by one and only one node. Only the owning node can update its partition. Therefore, there can be no data collisions. So long as all transactions that update a partition are generated by local users, the absence of the network only affects queries that need data in partitions resident in remote nodes. Of course, if a user initiates a transaction that needs to be sent to a remote owning node for processing, that transaction will fail.

In some applications, data collisions are unimportant. During split-brain mode, each node will miss some updates. However, when the network is restored, it may be acceptable to realize that eventually all collisions will be resolved simply by new updates that will overwrite the stale values.

Backbone Networks Do Fail

If split-brain operation is to be avoided, the availability of the interconnecting network becomes paramount. Specifically, since it is a component in the availability equation, its availability must be better than the desired availability of the system. If the active/active system is to have an availability of six 9s, the network should have an availability of seven 9s or so (three seconds of downtime per year).

Network connections today are highly reliable. Should a link fail in a communications channel, traffic is often simply rerouted around that link; and the fault is transparent to the users of that channel.

However, networks do fail. A common cause is the “last mile” that connects the network provider’s backbone to the user. This is often a single cable that can break because of natural or man-made actions (a tree limb falling, a backhoe digging).

However, the network backbone can also fail. Undersea fibre-optic cables seem to be particularly vulnerable, being cut by dragging anchors or broken by seismic activity. Submarine cables under the Mediterranean Sea account for the majority of communications between Europe and the Middle East, causing North Africa to be particularly hard hit.

On Wednesday, January 30, 2008, North Africa, the Middle East, and India experienced a massive Internet outage due to several submarine cables being cut. The outage created an Internet *notwork* that was destined to last for several days or even weeks in some cases.[3] Then

[3] What? No Internet?, *Availability Digest*; February, 2008.

on December 19, 2008, disaster struck again when three Mediterranean cables were cut.[4] Days later, efforts were still in progress to restore communication service to Egypt and some Middle East countries.

One of the biggest interruptions of international telecommunication services occurred in December, 2006, when a magnitude 7.1 earthquake broke nine submarine cables between Taiwan and the Philippines and cut connections between southeast Asia and the rest of the world. It took 49 days to restore full capacity.

But submarine cables are not the only culprits. In April of 2007, the Blackberry network went down for over a day due to a faulty upgrade.[5] Its network also went down in February of 2003 and twice in June of 2007.

On December 8, 2008, a Time Warner cable outage denied Internet services to several thousand users for over a day.[6] On December 28, 2008, AT&T service throughout the Eastern United States was down for most of the day due to a power failure.[7] Even major carriers are not immune from *notworks*. And the list goes on.

What will you do when you lose network connectivity? The only way to ensure network reliability is redundancy.

Redundancy

It is easy to say that your network should be redundant. However, it is another matter to ensure redundancy.

Communication-Link Independence

The "last mile" is perhaps the easiest part. The data center should be connected to two independent communication feeds that are electrically and physically separated. Electrical separation begins in the data center. Redundant network equipment in the communications room powered by separate power feeds and connected to separate network ports on the computers should be provided.

Physical separation means that the communication feeds should be routed into the data center over separate paths. There have been too many cases of a backhoe severing all communication lines coming into a data center through a common conduit (the same can be said for separate power sources).

But beyond the "last mile" is the network backbone. Separate communication links into the data center do not help if the backbone network is down. It is equally important to have these separate communication links be fed by different communication carriers so that if one loses its backbone network, the other can still be providing communications.

Redundancy issues don't stop here. Often, a communication carrier will lease lines from another common carrier. If the two carriers that you have selected lease lines from the same third-party carrier, and if that carrier loses its backbone network, then both of your carriers may be down.

And the problems go on. There is one case in which a company took all of these precautions. But what it didn't know was that both carriers routed cables over a common bridge crossing a major river. A flood caused the bridge to buckle, and both networks failed.

[4] Second Internet Outage Hits Egypt, Daily News Egypt; December 21, 2008.
[5] Blackberry Gets Juiced, *Availability Digest*; May 2007.
[6] Time Warner Cable Internet Outage Fixed*, OC Register*; December 11, 2008.
[7] Winter Storm Causes Major AT&T Outages, *Information Week*; December 29, 2008.

Communication Link Usage

Given completely independent communication links that cannot suffer a common failure, there is still an issue of how they are used. One possibility is to have the communication links configured as an active/standby pair. All traffic is routed over the active link. Should it fail, traffic is rerouted over the backup link.

But what if the backup link turns out to be nonoperational? Communication has been interrupted, and in an active/active system we are faced with the decision of running in split-brain mode or of having to down one or more isolated nodes. This is called a *failover fault* and happens all too frequently.

A better configuration from an availability viewpoint is to use the redundant channels in a load-sharing configuration. In this case, both channels are active; and traffic is split between the two channels. Of course, either channel must be able to carry the entire communication load should the other channel fail.

The advantages of this configuration are twofold. First, it is known that both channels are working because each is actively carrying traffic. Therefore, there are no failover faults. Should one channel fail, all traffic is simply rerouted over to the surviving channel.

Second, in normal operation, each channel is carrying only half the traffic even though it is capable of carrying all traffic. Therefore, the channels are less loaded than the active channel in an active/backup configuration, thus improving communication performance.

Load sharing is the active/active equivalent of application processing. All channels are active, and failover is simply rerouting traffic over a known good channel.

Other Network Parameters

Redundancy isn't the only consideration when configuring a network for an active/active system. Equally important parameters are the provided bandwidth, the channel latency, and the error rates of the channel. They should all be specified in the Service Level Agreements (SLAs) negotiated with the communication carriers.

Bandwidth

Needless to say, the bandwidth of the communication channels must be sufficient to carry the communication load. Furthermore, sufficient bandwidth must be provided so communication delays caused by queuing of messages waiting to get to the channel are acceptable.

This typically means that the bandwidth of each communication channel should be twice (or so) of the maximum transmission rates expected (this leads to an average message-queue length of one message).

It may be desirable to specify that a certain percentage of messages will not be delayed by more than a specified amount of time. For instance, one might want to specify that 99% of all messages be delivered within 100 milliseconds. This requires a more critical evaluation of performance and queuing to arrive at the bandwidth that will be required. Reference is made to Configuring to Meet a Performance SLA – Part 3, *Availability Digest*; February, 2009, for the techniques to make this bandwidth calculation.

Channel Latency

Channel latency is the amount of time that it takes for a message to travel from the source system to the target system over the communication channel. Over fibre or copper, channel latency is typically one millisecond per 100 miles.

Channel latency can have insidious effects on active/active system performance. These effects are different for asynchronous and synchronous data replication, used to keep the database copies in the application network synchronized.

Asynchronous Replication

If an asynchronous-data replication engine is used to replicate data changes between nodes in an active/active system, increased channel latency increases what is known as *replication latency*. Replication latency is the time from when a change is applied to a source database to the time that it is applied to the target database. Increased replication latency aggravates two undesirable traits of asynchronous replication:

- *Data Collisions:* If applications at two different nodes in the application network should update the same data item in their local database copy at the same time (by "same time" we mean within the replication-latency interval), each will be unaware that the other is trying to change the same data item. Each will replicate its change to the other node and will overwrite the change originally made at that node. Now the database copies are different, and both are wrong. These data collisions must be detected and resolved.[8]

- *Data Loss Following a Node Failure:* Should a node in an active/active system fail, any changes that are in the replication pipeline will not be delivered to the target system and will be lost. Depending upon the type of communication being used, the replication pipeline may include the communication channel. That is, a change sent by the source system and not yet delivered to the target system (i.e., the change is still in the communication pipeline) may not be delivered to the target system. Therefore, the longer the channel latency, the longer the change will be in the communication pipeline; and the bigger is the chance that the change will be lost following a source-node failure.

Synchronous Replication

Synchronous replication avoids the asynchronous issues of data collisions and data loss following a node failure. It does this by ensuring that all changes within a transaction are made to all database copies, or that none are. Synchronous replication accomplishes this by acquiring locks on all copies of a database item to be changed before making the change. Therefore, data collisions cannot occur. Furthermore, since a change is guaranteed to be made to all database copies or to none of them, there is no data loss should a source node fail except for transactions in progress.

However, channel latency causes a different problem if synchronous replication is being used. Synchronous replication imposes an additional delay on the completion of a transaction while confirmation is awaited from all nodes that the transaction can be committed. This additional delay is called *application latency*. Channel latency is an important factor in application latency as it determines how long the application must wait before it knows that it can commit the transaction.

Application latency adds to the transaction response time. In addition, it causes other problems, such as an increase in the number of simultaneous transactions in the system, locks being held

[8] P. J. Holenstein, W. H. Highleyman, B. D. Holenstein, Chapter 4, Active/Active and Related Topologies, *Breaking the Availability Barrier: Achieving Century Uptimes with Active/Active Systems*, AuthorHouse; 2007.

longer, and additional transaction aborts due to timeouts. Therefore, any increase in channel latency aggravates all of these problems.

Controlling Channel Latency

One problem with determining channel latency is that it is not dependent upon the physical distance between the source and target nodes. It is the communication distance between the nodes. For instance, the source and target nodes might be ten miles apart. But if the communication link must go through a central office in another city and then to local distribution centers, the communication distance could be hundreds of miles.

Furthermore, channel latency will change if the communication carrier has to route around a network failure. Channel latency could increase dramatically, causing applications to fail as transactions time out.

An extreme example of this occurred with the Mediterranean Sea cable breaks, mentioned earlier. The communication channels connecting Europe with North Africa and the Mideast had to be rerouted over a world-circulating path traveling over Asia, the Pacific Ocean, North America, and the Atlantic Ocean. Hundreds of milliseconds or even seconds were added to each transaction.

Therefore, it is very important to specify the maximum allowable channel latency in the communication SLA.

Error Rate

Modern-day communication networks are very adept at correcting transmission errors. It is generally assumed that a message ultimately will be delivered without error. However, the operative word is “ultimately.” The correction of message errors can often be done via self-correcting error codes embedded in the message. However, in some cases, the only recourse is to retransmit the message to correct uncorrectable errors.

Message retransmission slows down the delivery of the message. Therefore, to the application, uncorrectable errors have the same impact as channel latency. They increase replication latency in asynchronous-replication systems, and they increase application latency in synchronous-replication systems.

Message delays due to communication errors may be more onerous than you think. A simple analysis can serve to illustrate this point. Let

e = uncorrectable packet error rate.
n = number of packets in a TCP/IP message.
p = probability that a message will be retransmitted.

Then

- Probability that a packet will be in error = e.
- Probability that a packet will be error free = $(1 – e)$.
- Probability that a message will be error free = probability that n packets will be error free = $(1 – e)^n$.
- Probability that a message will have an error = probability that a message will be retransmitted = $1 – (1 – e)^n = p$.
- Probability that a message will be sent once = 1.
- Probability that a message will be sent a second time = p.
- Probability that a message will be sent a third time = p^2.
- Average number of times that a message will be sent = $1 + p + p^2 + p^3 + \ldots = 1 / (1 - p)$ $= 1 / \{1 – [1 – (1 – e)^n]\} = 1 / [(1 – e)^n]$.

As an example, let us assume that the packet error rate, *e*, is 1% (one packet out of 100) and that a message comprises 30 packets (*n*). Then the average number of retransmissions per message is

Average number of times a message is sent = $1/(1 - .01)^{30} = 1.35$.

A 1% packet-error rate has extended the message transmission time by 35%!

Communication SLA

As discussed above, it is clear that there are many communication channel characteristics that can have a major impact on the availability and performance of active/active systems. They include:

- Independence of the communication channels making up a redundant group.
- Bandwidth.
- Channel latency.
- Error rates.

Each of these should be clearly specified in the SLA with each communication carrier.

Summary

In an active/active system, the communication links are every bit as important to system availability as are the processing nodes. Therefore, they must be redundant. Redundancy implies that the communication links are totally independent of each other and that they be used in such a way to ensure that there are no failover faults should a channel fail.

In addition, there are many other parameters that are important. They include bandwidth, channel latency, and error rates. Each of these parameters impacts the time that it takes to get a message from a source node to a destination node. As this time increases, replication latency is extended in asynchronously replicated systems; and application latency is extended in synchronously replicated systems. Replication latency in asynchronous systems increases the chance for data collisions and for data loss in the event of the failure of a node. Application latency in synchronous systems extends the transaction response time and may cause applications to fail.

To avoid *notworks*, all of these parameters should be clearly specified in the Service Level Agreement with the communication carriers.