Improving Service Availability via Low-Outage Upgrades

Chryssa Dislis
Systems Engineering, Motorola Ireland Ltd., Cork, Ireland
Chryssa.Dislis@motorola.com

Abstract

Service availability is of key importance to operations and maintenance systems for mobile telephony networks. This paper will describe some of the challenges in providing continuous service and the impact of upgrade related outages. Potential methods for low outage upgrades will be discussed from an industrial perspective, including practical and logistic considerations.

1. Introduction

This paper examines low outage upgrades from the industry viewpoint. Reducing upgrade outage time as well as overall upgrade time is important for both customers and manufacturers, and has a significant impact on system availability. High availability is becoming increasingly important, and expected by end users. Although upgrades are planned events, they cause a significant disruption to operations and are a key driver to service availability.

Motorola Operational Support Systems Division (OSSD) provides GSM and GPRS mobile phone network management solutions to a significant number of customers and markets worldwide. The customers of this system constitute mobile service providers worldwide.

The product is an Operations and Maintenance System (OMC) and manages typically very large mobile networks. The OMC is used by the telecom operators to manage their networks and facilitate fault management, configuration management, performance management and load management. The product is of high complexity and makes extensive use of large databases. In spite of the fact that the mobile phone network can operate in the absence of a functional OMC, system availability is still important as the OMC is providing the operator with a view into the network.

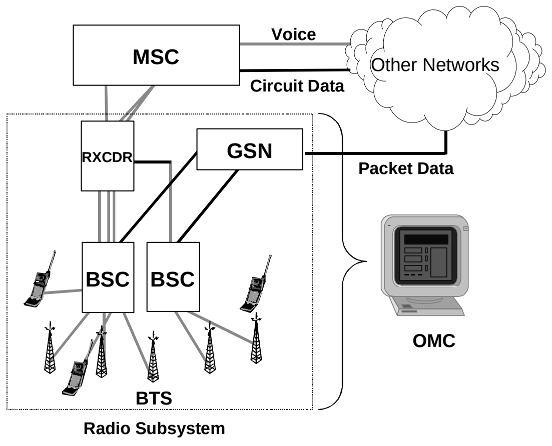

Figure 1. GSM Architecture

Figure 1 shows a typical GSM architecture for a mobile phone network, consisting of the Operations and Maintenance Centre (OMC) and a number of network elements (NEs).

The Base Transceiver Stations (BTS) provide the radio frequency interface for the mobile phones on the network. The BTSs are controlled by Base Station Controllers (BSCs). From the Base Station Controllers, digital voice channels are directed via a transcoder (RXCDR) to the Mobile Switching Center (MSC) and then to the national circuit switched voice networks.

The GSN, is a General Packet Radio Support Node, part of the General Packet Switched Radio System (GPRS). In the GPRS part of the system, packet switched data is directed via GSNs to packet data networks. Control data from all the network elements is directed to the Operations and Maintenance Center (OMC) subsystems. The OMC-R is the Operations and Maintenance Center for the GSM system, while the OMC-G fulfills this function for the GPRS system.

1.1. Availability definition

Availability is related to both the frequency of outages/service interruptions and the time to recover from such events. Outage frequency is expressed on terms of MTBF, or Mean Time Between Failures, which is calculated as follows:

$$MTBF = \frac{Total\ up\ Time}{Number\ of\ \ Failures}$$

Recovery time in expressed in terms of Mean Time To Repair (MTTR)

$$MTTR = \frac{Total\ down\ time}{Number\ of\ \ failures}$$

Availability is calculated as

$$Availability = \frac{MTBF}{MTBF + MTTR}$$

MTTR is generally related to support and logistic issues, which affect the recovery time. Time for an engineer to travel to a site generally outweighs the repair time itself. MTBF is related to product issues and defect prevention, i.e. preventing outages from happening in the first place, or having seamless recovery mechanisms in place so that outages self recover without affecting service.

Note the difference between Availability and Reliability. Reliability relates to the probability that a system will work during a defined time interval. As such, reliability is related to the frequency of outages, while availability is related to both outage frequency and time to repair. Designing a system for high reliability without considering recovery time in the event of failure is unlikely to result in a highly available implementation.

Availability is often expressed as a number of nines (NN). Thus 4NN equates to 99.99% availability. In terms of annual downtime, 4NN equates to 52.5 minutes of downtime per year. 5NN availability equates to a system working 99.999% of the time. If the time interval we consider is one year, the 5NN system would be out of service for no more that 5 minutes and 15 seconds. Strictly speaking, this time would include any scheduled maintenance.

5NN is generally the level of availability expected for critical systems. It is unlikely that 5NN would be achievable without some level of redundancy both at the hardware and the software level.

1.2. Impact of upgrades

Upgrades are often ignored in system availability discussions, as they can be planned in advance to minimise operational impact. On a like for like basis, it is true that a long upgrade is preferable to a long unplanned outage. However this is a predictable and regular occurrence; upgrades happen at regular intervals and create disruption to operations. They may be scheduled, but they result in a system that for a period of time is not available for service. Upgrade time and the ensuing disruption to operations are always important to customers and have to be addressed as an availability issue.

As an example, assume that a system has an availability of 5NN for unplanned outage events. This means an average downtime of no more than 5.25 minutes per year. The software in upgraded annually, resulting in an outage of 1 hour duration. The true availability of the system is less than 4NN. This is relatively good, but not in the 5NN league and too low for critical applications. Keeping a system at 5NN necessitates no-outage of very low outage upgrades, bearing in mind that a 5 minute outage uses up most of the downtime 'budget' of a 5NN system.

2. Upgrades – the customer viewpoint

Several aspects of upgrades are important to the customer. A primary issue is of course the added functionality that the upgraded software will provide. This has to be balanced against possible disruption to operations and the possibility of new defects. Mature software is generally stable, as any defects will have been identified and fixed. This disruption to operations means that it is possible for a customer to choose not to take a particular version, if they consider that the disruption outweighs the benefits. For example, software development organisations often do not take up every single new version of an operating system. The possible disruption to the stability of the software has to be balanced against the advantages of the new version.

The disruption to operations has two aspects. One is the outage time, the core time during which the system will be unavailable for service. A superset of that time is the overall upgrade time, including preparation and cleanup. Although this is not strictly outage time, it can be argued that is still impacts the availability of the system for practical use. The challenge is to minimise both the core outage time and the overall upgrade time of the system and associated sub-systems.

Upgrades are generally undertaken during a maintenance window outside working hours, or during light loading of the system. It is very important to the customer that the upgrade is successfully completed within that window. A failed upgrade, a roll-back and then another upgrade at a later time are extremely disruptive. It is preferable to have a longer maintenance window than to schedule another upgrade. In other words, the stability and predictability of upgrade duration are often more important than absolute upgrade outage time.

Customers generally prefer the flexibility of scheduling upgrades in their own time and using their own staff, although specially trained engineers are sometimes preferred for complex upgrades.

Finally, there is a cost to the customer associated with upgrades, in terms of personnel being present outside working hours, additional support required and lost operations during the upgrade.

3. Upgrades – the engineering viewpoint

In planning upgrades, the engineering/company priorities are not entirely the same as the customers'. While there is a key requirement to limit the upgrade duration, there are additional requirements from the engineering perspective.

The first of these requirements is to control engineering development costs. There is significant engineering time in planning the upgrade, writing upgrade procedures and testing them to ensure that upgrades will work.

Stability and repeatability of upgrades (even by non-expert upgraders) are also key requirements and influence the amount of engineering effort required. A high degree of automation (scripting) is often required, making upgrade planning an engineering-intensive operation.

Upgrade procedures need to be thoroughly tested, especially as they will have to cope with the possibility of non-standard configurations on customer systems. It is worth noting that test costs escalate differently to development costs. For example, an upgrade may be developed for two slightly different platforms, with minimal additional development time. However, the test times would double, as the procedures would have to be tested on two, rather than one platform Therefore the standardisation of upgrade procedures and system platforms results in development and test savings.

It is clear from the above that any significant changes to the methodology of upgrades incur additional costs. This in turn can create a conservative approach to software changes, which has to be weighed against the possible advantages to the customer and the company.

If more than one aspect of the software is being changed (platform, operating system, applications) incremental upgrades are less risky but generally these are bundled in the same upgrade procedure to minimize disruption. Roll back in these situations can be much more challenging than in a single application upgrade.

Contingency plans in case of the need to roll back to a previous release are a key part of upgrade planning. In addition, changes to the upgrade process are easier and cheaper in engineering terms than changes to the core software.

4. Reducing upgrade duration

Some methods of reducing both the overall duration of upgrades and the outage duration are described below. These rely on changes to the upgrade process without changing the core application software. Core software architecture changes may be a method of reducing upgrade time but are expensive in terms of engineering and test time, and are likely to introduce additional defects into otherwise stable code.

4.1. Preparation

Good preparation for the upgrade is very important. Ensuring that the system is free of unnecessary files and data and that a backup has been taken reduces the risk of problems with the upgrades, and facilitates roll-back in case of problems. This pre-upgrade clean-up can be automated or made a strict pre-requisite for continuing with the upgrade.

Attention to detail at the early stages can save a lot of time later. If a specialist engineer is not present, the customer should also insure that the engineer undertaking the upgrade has sufficient experience and that expert support is available should things go wrong.

4.2. Minimisation of operations

For the upgrade itself, minimizing interaction with the user is a good method of minimizing upgrade time. This removes the potential for error, and speeds up any input. Generally it necessitates using scripts to replace the human interaction. These scripts can get very complex, as they have to account for different system configurations, initial cleanup, and the ability to recognize when something has gone wrong and initiate roll-back.

In the absence of these scripts, even a simple action such as pre-typing the required commands into a file, checking them for spelling, and then pasting them to the command line during an upgrade, creates significant time savings.

4.3. Use of Mirrored disks

A widely used method is to make use of mirrored disks. Many systems use mirrored disk configurations as a method of improving availability. If a disk fails, its mirror is used to ensure continuous service.

Using mirrored disks in an upgrade means temporarily breaking the link, leaving the core system to operate from one disk set only (set 1). The other set of disks (set 2) is used to load the upgraded software. The system is then restarted from disk set 2, and the mirror configuration restored. This arrangement has the advantage that a quick roll back to set 1 is possible should there be a problem.

The advantages are clear: outage time is minimized, ideally to the restart time from set 2 (some synchronization of data may still be necessary). It also incurs no additional hardware costs, as it merely makes use of the existing configuration.

However, the reduction in outage time can only be realised if the core system can continue operation while the upgrade takes place. This assumption is not true for all systems and depends on the software architecture. However, the advantages of a good rollback mechanism remain, making this a good solution for mirrored disk systems.

Disadvantages are the engineering effort in creating, debugging, testing and executing a more complex procedure. There is a small risk that a disk failure will occur during the upgrade, when no mirroring is available. The overall upgrade time may also be longer due to the breaking and recreating the mirrored configuration.

4.4. Additional disk set

A related method is to introduce an additional disk set, used only during upgrades. The new version is loaded onto this set (set 3). The system is them restarted from set 3, and if the system uses disk mirroring, this is set up with either of the other disk sets. This leaves one disk set with the old version (which can be used for roll back if necessary). This spare disk set will them be used for the next upgrade.

In terms of planning the upgrade, this is slightly simpler than the method above, and results in a small reduction in the overall upgrade time. The outage time is reduced as above. There is an additional cost of the extra disk set. This solution is suitable where a large number of applications are upgraded at once, or where it is necessary to keep the old version on the system.

4.5. Clustered systems

Both the above methods make use of redundancy in the system. In fact, redundancy is the key to improved availability. At the system level, critical systems can be run in a clustered configuration. This can be a standby system, which is a complete duplicate of the live system and can take over if the main system fails, with minimal loss of service. Clustered configurations can also be used for load sharing, which increases complexity but makes use of the added capacity to increase performance.

A clustered system is easily utilized for a low outage upgrade. The upgrade can take place on one side of the cluster. This then becomes the live system Once a stable upgraded system has been achieved, the cluster configuration is re-created. Loss of service in this case is limited to the fail-over time of the cluster. Roll back is significantly simplified; the upgraded system can be checked to ensure that all is in order before it goes into service.

The main disadvantage of this method is the additional cost of the redundant system. This can be reduced if the cluster is used for load sharing, or in a N+1 configuration, but these are more expensive from a development and systems administration perspective.

4.6. Domain systems

Some servers can be set up in a number of different domains. Each domain is in effect a separate machine which can run independent applications. From an upgrade perspective, setting up two domains with identical system configurations would be the first step. The upgrade takes place on one domain while system operation continues on the other. The upgraded domain then becomes the live system.

Whether the two domains are maintained during normal operation depends on the system load, applications and availability requirements. It is possible to recombine the two domains into one for enhanced performance during normal operation.

5. Conclusions

Upgrade time is a significant contributor to reduced availability and general disruption of normal operations. Although not as damaging as an outage resulting from a software or hardware failure, upgrade outages are important to the end customers. Therefore, upgrade outage time minimisation needs to be designed for in both the hardware and the software architecture and taken into account at the system requirements stage.

Different methods of minimising upgrade time were examined from the customer and industry viewpoint. Generally these methods make use of hardware redundancy to minimise upgrade time. Architectural changes to the software were not examined in this paper, although they may be a good solution for particular systems.