THE ARCHITECTURE AND ADMINISTRATION OF THE ATLAS ONLINE COMPUTING SYSTEM

M.Dobson, CERN, Geneva, Switzerland
M.D. Ciobotaru[*†], E. Ertorer[*¤], H. Garitaonandia[§], M. Leahu[*†], L. Leahu[*†], I.M. Malciu[*†], E. Panikashvili[*£], A. Topurov[*‡], G. Unel[*#]

Abstract

The needs of ATLAS experiment at the upcoming LHC accelerator, CERN, in terms of data transmission rates and processing power require a large cluster of computers (of the order of thousands) administrated and exploited in a coherent and optimal manner.

Requirements like stability, robustness and fast recovery in case of failure impose a server-client system architecture with servers distributed in a tree like structure and clients booted from the network.

For security reasons, the system should be accessible only through an application gateway and, also to ensure the autonomy of the system, the network services should be provided internally by dedicated machines in synchronization with CERN IT department's central services.

The paper describes a small scale implementation of the system architecture that fits the given requirements and constraints. Emphasis is put on the mechanisms and tools used to net boot the clients via the "Boot With Me" project and to synchronize information within the cluster via the "Nile" tool.

INTRODUCTION AND REQUIREMENTS

The ATLAS experiment requires a large online computing farm (>2500) for the readout of the detector front-end data, the trigger decision farms (second and third level of trigger) and all the ancillary functions (monitoring, control, etc...). These machines need to be administrated in a coherent and optimal way. The farm should have high availability and should be highly reliable and robust to make best use of available luminosity.

Much of the knowledge which has been used to design the ATLAS experimental area has been gained from experience in the various TestBeam periods over the last few years. This experience was analysed by a SysAdmin Task Force and a report produced, outlining the possible directions and recommendations for the implementation of the final system. The practical realisation of those ideas has produced the final system described here.

A major concern in any high availability computing farms is the mean time between failure. This has been shown to be very dependent on the mean time between failures of the disks in the computers. For this reason, TDAQ decided to try to reduce the need for disks on the data acquisition system, by making nodes diskless. The diskless nodes are booted into Linux over the network from a server. This boot server approach has other advantages, ease of maintenance, reproduceability on a large scale, and alike. The Boot With Me (BWM) project was developed in order to respond to the need for a flexible system to build boot images and configure the booting of the diskless nodes. This project is presented in the section 2.

Another major corner stone of the system is therefore the boot servers. These are designed to serve the DHCP requests and boot images, and to provide network mounted disks for the main part of the OS and TDAQ applications. In this kind of system, the servers are a single point of failure. To overcome this limitation, the system, which would anyway need many such servers to cover the large number of client machines involved, has been made more robust by sharing the responsibility of booting and providing network drives to a machine across two or more of the servers. This redundancy insures the high availability of the clients independently of that of individual servers.

The redundancy and availability of servers and computers is dependent on a performant and redundant network interconnecting the devices. This design of the control network (for the control and service traffic between servers and clients) and the data network (for the transport of the event data) is presented in [1]. Monitoring of these networks is also of primary importance and tools to do this are presented in [2] and later in this paper.

For the trigger farms, performance considerations imposed by the dynamic loading of libraries throughout a run (different algorithms and libraries depending on luminosity conditions) have forced the use of local disks on those nodes, to avoid the impact of large reads from the server network disks. The large number of nodes in these farms might offset the lower reliability.

All the servers and the local trigger farms disks, will hold copies of the same software. All these copies of the software need to be kept synchronized and the mechanisms which have been investigated will be presented in section 4.

[*] CERN, Geneva, Switzerland
[#] Also affiliated to University of California Irvine, CA Irvine, USA
[†] Also affiliated to Politehnica, University of Bucharest, Romania
[‡] Also affiliated to Technical University of Sofia, Bulgaria
[¤] Also affiliated to Gazi University, Ankara, Turkey
[£] Also affiliated to Technion, Haifa, Israel
[§] IFAE, Barcelona, Spain

Another requirement for the system is the ability to run the experiment and take data for up to 24 hours while having lost the connection to the IT department and Tier 0 centre (responsible for long term storage of the data, distribution of it to Tier 1 centres, and also analysis of some of the data). The implications are that the system should replicate any services in IT which are vital such as DNS, NTP, user authentication; and should be able to buffer the event data.

The Computing and Network Infrastructure for Controls (CNIC) working group [3] was mandated to look at the requirements of running experiment control systems at CERN. The group has produced a Security Policy document which had implications for the design of the experimental area system. The main one being the isolation of the experimental networks (in our case the ATLAS Technical and Control Network, ATCN) with respect to the CERN general purpose network (GPN) and the use of Applications Gateways to access them.

The exact design and architecture of the system for the ATLAS experimental area is shown in the next section.

ATLAS SYSTEM ARCHITECTURE

The TDAQ System Administration effort is mainly split over the USA15 cavern, which holds the detector front-end readout and main readout system, and the SDX1 building, which houses the trigger farms.

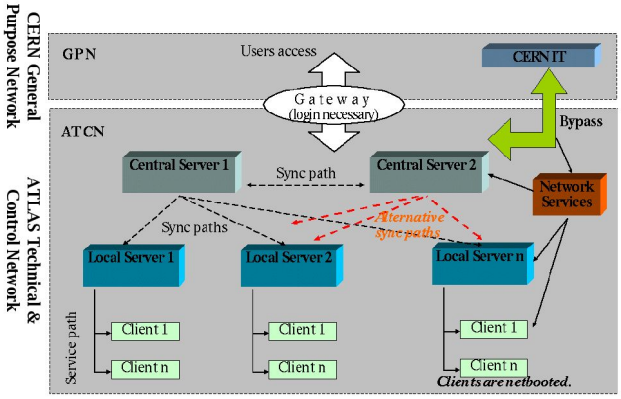

Figure 1: ATLAS System Architecture

Figure 1 shows the ATLAS Point 1 system architecture. As explained before the GPN and ATCN networks are separated by the Application Gateways which form the interface. Users wishing to login to machines in the experimental area have to first login to the Application Gateways and then hop to the desired machines. Another link between the two networks is available in the form of the bypass route indicated in the top right corner. This link is used for the communication with IT services such as DNS, NTP, CASTOR (CERN Advanced STORage manager), etc... The set of services which are visible is restricted to a minimum and is configurable by the Administrators of the ATCN domain. The bypass link also allows machines inside the ATCN to be visible to other machines in CERN. This is not currently used.

Figure 1 shows also the server/client architecture as implemented for the ATLAS system. The Local File Servers (LFS) are in fact the boot/disk servers which are distributed on a per rack basis (or one for multiple racks depending on occupancy of racks). These will be serving of order 30 clients. The holder of the master copy of the software is represented by the Central File Server (CFS) in Figure 1. Again for reasons of reliability and availability, two CFSs are foreseen. These machines will be synchronised to each other in real time, which imposes a different synchronisation method than that used between the CFS and LFSs or the LFSs and the client local disks. For the CFS to CFS synchronisation the heartbeat software [4] is under evaluation. Another one is use of a distributed file system, or SAN/NAS system.

The Network Service box of Figure 1 represents all network services required by the system in order to fulfil the requirement of 24 hour disconnect from the IT and Tier0 systems. These services include NTP, DNS, DHCP all in synchronization with those services in IT. User authentication is discussed later in the paper.

BOOT WITH ME

The Boot With Me (BWM) project was developed to answer the need for a flexible and configurable system for building Linux boot images, and for controlling boot time configurations and post boot configurations.

As mentioned before, the ATLAS experiment will have in the final system, a few thousands of computers split into categories according to their type, function (e.g. ReadOut System, ROS; processing unit) or sub-detector. All these machines will have a variety of hardware configurations, such as remote control mechanisms (IPMI cards, or special management interfaces), sensor monitoring devices, network interfaces, and custom PCI data acquisition cards, which have to be supported. The function of a node adds other requirements on the OS configuration (special kernel settings, network card settings, special IT services required). The time factor also brings in constraints: the system has to be kept up to date and downtime due to updates should be minimal; the system will be more heterogeneous as hardware replacement, renewals and upgrades will occur; the ramdisk should have a reduced set of binaries and libraries because of thin clients (single board computers, SBCs); the administrators of the system will change and the learning curve should therefore be reduced.

The design requirements for such a tool are therefore: flexibility for adding, removing and changing the functionality of a boot image; single point of definition of all the different functionalities, to ensure a fix is propagated to all images using this functionality; ease of use for a system administrator with Linux knowledge; maintainability for ease of changing or adding features.

Concepts, configuration and build

BWM is based on the following concepts: template, project, plug-in and inheritance (see Figure 2 for a diagram showing the relationship of the concepts).

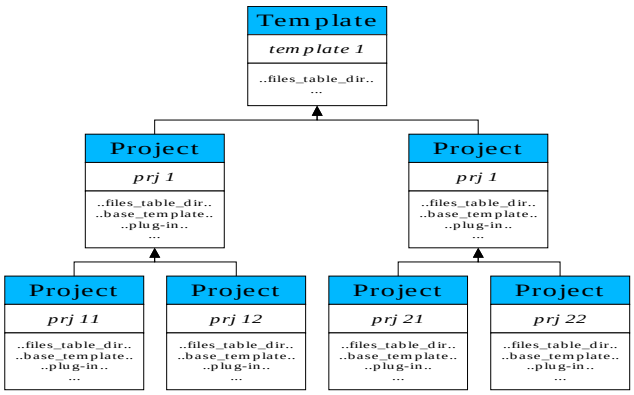

Figure 2: Relationship of BWM Concepts

A template is a minimal specification for a boot image that works. It has no functionalities or services apart from the ones that make the operating system work (e.g. no services like autofs or NFS are available).

A project is the main configuration specification that provides a fully functional boot image. This means for example that the SSHD configuration is properly done, all the necessary mounts from the servers are available, or the users authentication is centrally managed by an LDAP service. Yet this is not a complete Linux installation. A project can be derived from a single template or from another project (only single inheritance allowed).

A plug-in defines a single functionality and must be included in a project (it can not function stand-alone). Multiple plug-ins can be included by a project and multiple projects can include the same plug-in.

These components hold the definition of the directories and files to be added to a boot image, either taken from the reference Linux installation or from a customized area.

A BWM configuration is defined in a set of XML files organized in a directory structure which also hold the customized files. A Python script parses the XML based configuration and populates an area with the mentioned files, producing a file system holding a minimal Linux distribution based on the reference one and customized according to the needs of the netbooted node. The BWM configuration is not dependent on the reference Linux distribution, in as much as with the same configuration it is possible to build boot images based on different kernel versions.

Boot time configuration

With a few exceptions, the cluster nodes are booting from the network using the PXE (Preboot eXecution Environment) boot ROM. The others, mainly older SBCs, use BOOTP.

The LFSs play the role of boot servers for a dedicated set of clients. On a DHCP request, they serve the PXE binary (enhanced version of the SYSLINUX project one [5]), and the kernel and boot image files.

To minimize the number of boot images, some operating system configurations are parametrized inside the boot image and the finalization is done during the local boot process using information provided by the DHCP server and client's PXE configuration on the server (e.g. the name and IP address of the server, kernel arguments).

The most critical operation in the system initialization is to enable the network connection, used to mount network resources (complete /usr area or other software) from the server. The clients have multiple network interfaces and “guessing” which one should be used to communicate with the server (control network interface) or with the data acquisition (data network interfaces) is not trivial. The reasons are that: manufacturers do not use a common method for allocating onboard or expansion bus network devices; it may be necessary to use different medium (copper or fibre) for specific networks while using the same driver to talk to them; sometimes a device on an expansion bus needs to be used instead of the onboard one (e.g. better quality).

An identification is done by looking at the PCI card IDs and matching them to kernel modules. Loading this module, the system might identify multiple cards and multiple devices per card. It remains to be decided which device should be used for the control network and a decision algorithm has been implemented and can be controlled by kernel parameters.

The system configuration files which depend on the name of the local file server (e.g. /etc/fstab file) are generated or configured at boot time.

Post-boot configuration

Once the network connection with the server has been established and the network drives are mounted from the server, the client configuration finishes by running the post boot scripts from a shared repository. These scripts are organized according to the structured machine name which follows the convention of <HW type of the node>-

<detector name>-<function>-<sub-function / detector-area>-<id>(e.g. pc-tdaq-pub-01 for the first public PC belonging to TDAQ).

The post boot scripts are executed starting from the most generic to the most specific (host one) using the information read from the client's hostname (e.g. for pc-tdaq-pub-01 the following post boot scripts are called in order: pc, pc-tdaq, pc-tdaq-pub, pc-tdaq-pub-01). In this way, an operation in the scope of TDAQ machines can be called in the post boot script pc-tdaq. For example the TDAQ specific software can be mounted only on the TDAQ machines in the pc-tdaq script, while the temperature sensor kernel modules which are hardware specific should be loaded in the most node specific post boot script, namely pc-tdaq-pub-01 in our example. The client specific post boot scripts are configurable by the people responsible for a specific computer.

SYNCHRONIZATION

In the cluster there will be multiple servers (150-200) needed for the many TDAQ/detector clients (> 2500), all of which hold the same software to export to clients. Also some clients with performance requirements will have local disks for the software. Therefore all servers and some client local disks need to be synchronized for the software.

The synchronization studies were originally started in the context of the deployment of the ATLAS software (Offline, HLT and TDAQ) on a large cluster used for scalability tests (of order 6 GB to be synchronized on ~700 nodes) [6]. The problem is the same for the experimental area even if the conditions are different. Possible software distribution or synchronization mechanisms for various cluster sizes and network topologies were investigated [7]. This paper will briefly present the alternatives that were investigated and the results which justify the use of the a worm like tool in the experimental area.

The two possible approaches use either fixed routing points or adaptive distribution. The first approach was implemented in a tool called Nile using worm technology. Nile launches SecureSHell (SSH) connections in a configurable mixture of parallel and cascaded modes, in order to incrementally synchronize software repositories or execute commands. The configuration of the routes is done by a system administrator, and the tool can therefore achieve scalable delivery while being adapted to the network.

The second approach, adaptive distribution, is implemented with peer to peer protocols (P2P). In this system, nodes interested in a file act both as client and server for small pieces of the file. The strength of P2P comes form the adaptive algorithm that is run in every peer or node, the goal being to maximize the peer's own throughput and the overall throughput of the system with no configuration effort. The P2P tool used was BitTorrent.

The studies have shown the P2P tool to work better for unknown networks (where it adapts to the system). However the fixed routing tool (Nile) performs better for the hierarchical nature of the network topology in our experimental area setup (symmetrical nature).

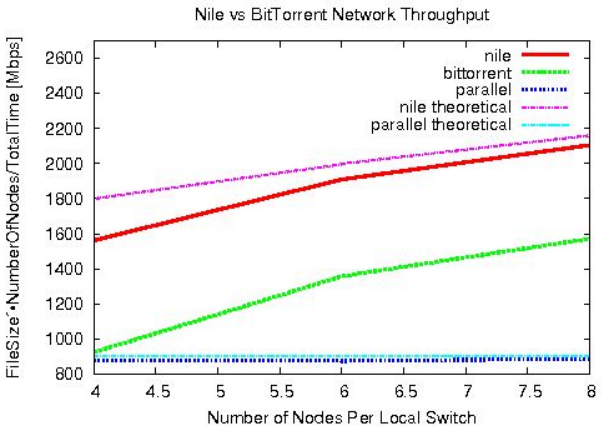

Figure 3: Performance of Nile versus BitTorrent

Figure 3 shows the actual performance of the two tools compared to their theoretical limits and to a complete parallel copy or synchronisation, for a small size test system in the experimental area. It is clear that Nile is performing better than BitTorrent, but it is expected that on a larger system the BitTorrent performance would tend towards that of Nile. However the latter has a major advantage for the experimental area, due to its added functionality of being able to do incremental synchronization, thereby minimizing the time push updates or patches across the system.

AUTHENTICATION

Following the requirement to be independent from IT, and to allow more flexibility, the experimental area has its own user and password database in the form of an LDAP server. The system is standalone but for consistency it is synchronized with IT for the usernames and user IDs.

Contrary to the way some of the previous experiments have been run, it has been decided to have user based authentication and not group based authentication, in order to have accountability and traceability of actions, as well as increased security. The reasons for wanting to use group accounts comes from the natural splitting of users into categories of people and tasks which they are allowed to do, for example some users are shifters, detector experts, TDAQ experts. In order to address this categorization and still have the user authentication for accountability and traceability, it has been decided in TDAQ to implement a Role

Based Access Control (RBAC) authentication system [8]. In such a system, the user would be able to belong to one or more roles (e.g. shifter, detector expert), although currently for ATLAS a user would only be able to be in one particular role at a time. The role will allow the user to do certain tasks at the TDAQ Control system level (e.g. start a run) and at the OS level (e.g. able to start certain processes). The SysAdmin part of the system is to setup the OS level support for this mechanism by using groups, pseudo-users and SUDO (program that controls which applications can be run by users). For the propagation of rights between machines and applications, Kerberos [9] is being investigated as a possible tool.

The LDAP server is foreseen as the repository for the roles (groups, pseudo-users), their authorized tasks, and the user to role mappings.

DISTRIBUTED FILE SYSTEM

So far very little has been said about the different distributed file systems required for the ATLAS experimental area. The one which has already been mentioned was for the distribution of the software from the LFS to the client nodes. This file system will be distributed to a limited set of nodes (those booting from a particular LFS, i.e. up to 32), in read only mode. Therefore the performance requirements are not high and for this purpose NFS has been chosen. NFS is standard on all platforms, is free, and is reasonably performant in read-only mode for a limited number of clients as long as version 3 or better of the protocol is used. So far this system has proven to be reliable and has not shown any performance problems.

The distributed file system which we have not yet mentioned, is the one required to support users home directories. Even if those should not be heavily used, they have to accessible in read/write mode from > 2500 nodes in the final system. There are very few distributed file systems which are able to do this, of which AFS (Andrews File System [10]) and GFS (Global File System [11]) are possible but not the only candidates. In order to have an initial system operational NFS was also used for the home directories, however limitations and performance problems have appeared with approximately 100 to 150 nodes. Investigations are ongoing to find a solution for the home directories. Possible options are: to make the home directories read only in the system (which is fairly controversial but which could be feasible for the final system); to use NFS version 4 which is announced as having better performance than version 3 currently used; to use a more performant distributed file system such as AFS , GFS or some other one.

MONITORING

Monitoring of the ATLAS experimental area system is very important, both for the machines and for the interconnecting networks (control and data).

Host Monitoring

Host monitoring is currently being done using the Nagios software [12]. Nagios allows the monitoring of various services for the machines. For all machines, the following basic services are monitored: ping response, SSH connectivity, ramdisk usage (for netbooted nodes), NTP synchronization, kernel version, BWM version, LFS name, temperature, auto-mount status, HDD state (if present). This list is not fixed, as Nagios allows the administrators to add and configure new features and services. Figure 4 shows a screen shot of one of the status screens which Nagios provides via HTTP. On certain hosts, such as LFS or Application Gateways, advanced features are monitored, such as NTP daemon status, number of users, state of exported file systems, DHCP daemon status, etc...

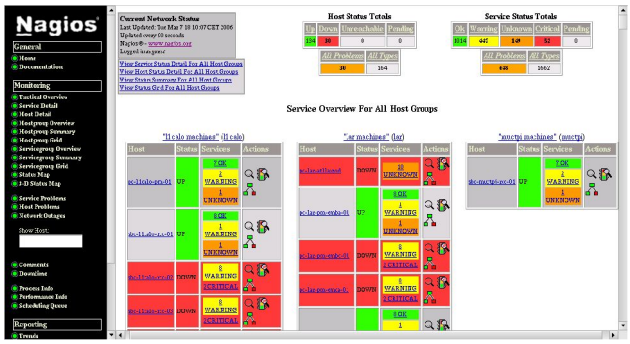

Figure 4: Nagios host group status

Nagios also offers the possibility to trigger events if the results from the service tests are not as expected. This is used to send emails and SMS messages when critical services are no longer in an operation state.

Network Monitoring and Configuration

Monitoring and configuration of the network will also be critical (see [2]). Figure 5 shows the hierarchy of the proposed network management solution. For fault and performance management (monitoring) a commercial network management system will be used. It allows monitoring of the network devices at a relatively low rate (> 5 minutes), but its main strength is its ability to do Root Cause Analysis, that is to find the root cause of a problem which is observed in the network.

For the data network it might be necessary to monitor the network at a much higher frequency than the commercial network management system allows. This would be for a limited amount of time,

in a specific area of the network where a problem is suspected or identified. For this a home grown tool (called Yet another Traffic Grapher, YaTG) to monitor the switches via SNMP was developed.

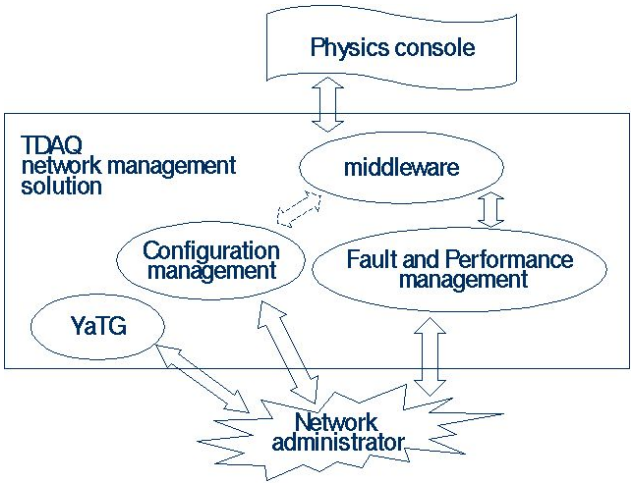

Figure 5: Network Management Hierarchy

The last component which is required is a configuration management tool. The basic functionality will be provided by Rancid [13], and more investigations are ongoing to see which tool could provide more functionality.

REMOTE OPERATION

The ATLAS experimental area cluster is split between underground and surface buildings which are physically at some distance from the control room (especially for those underground). Following a power cut it will be necessary to restart all the machines and continue the data taking. It is not feasible to have to switch on > 2500 nodes manually. Therefore technologies which permit the remote operation (e.g. power on/off, get status) of machines have been investigated and the IPMI (Intelligent Platform Management Interface) technology was chosen for its wide availability and complete set of functions. Therefore all TDAQ machines are being purchased with IPMI interfaces and which can be used to remotely control them. This technology also provides (with version 2.0 of the specification) console redirection and the ability to interact with the BIOS remotely.

The only machines which will be an exception to this are the SBCs which do not have an IPMI device. For these machines it is possible to reset them using a reset connector on the motherboard. In order to remotely reset these devices, the reset connector was interfaces to the parallel and serial ports of a PC. This system suffers from the necessity of having a PC in or near the SBC, the limited number of serial or parallel ports on most modern machines, and the fact that the machine must be booted to allow the reset of the SBCs. An alternative method was to interface the reset connector to the general ATLAS Detector Control System (DCS) which has input/output channels in all racks in the ATLAS experimental area (used for various sensors, or control devices) and which, by design, offers uninterrupted service. This interface is currently being implemented.

CONCLUSION

In conclusion, the netbooted scheme, with the client-server architecture has been demonstrated to work for the small scale system already installed in the ATLAS experimental area. The inherent hierarchy of the system architecture will ensure the scalability to the final ATLAS size (> 2500 nodes).

The BWM project has fulfilled its requirements and has proved a very flexible tool to support many netbooted categories of clients, and to configure all steps in the boot and post-boot phases.

Only one single area of concern over performance has emerged in the area of distributed read-write file systems. However there are solutions which exists at this time and these are being actively investigated as possible solutions to this problem.

Overall the system is felt to be sufficiently flexible and performant for the final system and it is being prepared for intensive use in Cosmic ray and noise runs from mid-2006 to mid-2007.

REFERENCES

[1] S. Stancu et al, “Networks for ATLAS Trigger and Data Acquisition”, CHEP06, Mumbai, India, 2006.
[2] C. Meirosu et al, “Planning for predictable network performance in the ATLAS TDAQ”, CHEP06, Mumbai, India, 2006.
[3] http://wg-cnic.web.cern.ch/wg-cnic/
[4] http://www.linux-ha.org/HeartbeatProgram/
[5] http://syslinux.zytor.com/
[6] D. Burkhart et al, “Testing on a large scale: Running the Atlas Data Acquisition and High Level Trigger software on 700 pc nodes”, CHEP06, Mumbai, India, 2006.
[7] H. Garitaonandia et al, “Worm and Peer To Peer Distribution of ATLAS Trigger & DAQ Software to Computer Clusters”, CHEP06, Mumbai, India, 2006.
[8] http://csrc.nist.gov/rbac/
[9] http://web.mit.edu/kerberos/www/
[10] http://openafs.org/
[11] http://www.redhat.com/software/rha/gfs/
[12] http://www.nagios.org/
[13] RANCID, Really Awesome New Cisco confIg Differ, http://www.shrubbery.net/rancid/