WIND™

Carrier Grade Linux and Wind River

WHEN IT MATTERS, IT RUNS ON WIND RIVER

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Communications and data service networks are seeing revolutionary transformations. Of all the new challenges facing service providers and carriers, driving down costs while still maintaining carrier class characteristics is the hardest; their platforms must provide service and mission-critical applications in an all-IP environment. For many, moving away from specialized proprietary architectures has proven to be the solution—proprietary platforms are expensive to buy, maintain, and scale. Open platforms, on the other hand, based on industry-established standards and common practices, give the needed design flexibility and freedom that the telecom industry needs.

Carrier Grade Linux (CGL) is the standard in terms of moving to open architectures. Today, even tiny embedded Linux distributions such as the Yocto Project include features such as IPsec in the form of Openswan (a mature cousin of strongSWAN), and the larger OpenEmbedded community provides features such as Lightweight Directory Access Protocol (LDAP) via OpenLDAP; Network Information Service (NIS, a system for distributing configuration data to multiple hosts within a network) via the Linux NIS tools; and even Kerberos authentication, a protocol designed to allow computer nodes to securely identify themselves to other nodes across an insecure network. Such technologies have seen widespread adoption, and today they are even available on platforms as lightweight as the Raspberry Pi.

The majority of the requirements laid out in the CGL specification apply equally well to large corporate infrastructures, data centers, and small, highly mobile devices. Based on open standards, Wind River® Linux is a powerful and versatile carrier grade platform capable of meeting stringent infrastructure reliability needs, without compromising any of the core, industrial-strength features that make carrier grade desirable in all parts of the ever-expanding network.

TRANSFORMING THE NETWORK

Today's consumer demands constant access to rich media and enhanced communications features. Ensuring safe and timely traffic for data and high-performance I/O packet processing technologies has become crucial for customer retention, but it can burden networking capacity.

Whether in the data center sectors, where budgets continue to shrink, or in the growing and more competitive cloud and secure environment sectors, vendors are forced to do more with less, work more efficiently, and alter how they develop products without sacrificing performance, safety, and security. Incremental improvements in how these systems are developed are not sufficient; radical changes need to occur for vendors to keep pace. In addition to these pressures is the need to be able to add new capability, as well as manage systems throughout the product lifecycle, especially when keeping systems safe and secure.

This new set of requirements comes with new challenges. Systems that were standalone now need to interact with other systems, not only across multiple vendors but across different hardware and software boundaries as well. Applications have to migrate from legacy standalone systems to globally connected, intelligent systems. Demand for six-nines (99.9999%) availability has become the standard, and security requirements have increased the cost of verification and validation. As such, and accelerated by the necessity of cost reductions, networking companies must make use of commercial off-the-shelf (COTS) technology. As a result, more systems are designed with open architectures in mind. With open architectures, COTS components can drop into systems with little extra integration work, allowing easier and lower-cost technology insertion and greater competition in the marketplace.

This trend flows down from system-level designs to the use of open standards, such as CGL in the networking industry. These open standards enable application portability and ease of integration, increasing the interoperability and scalability of systems and reducing risk throughout the system lifecycle.

If they are to meet increasing traffic requirements, and keep their customers happy, carriers must make the transition from traditional infrastructure and technologies to open and standardized systems based on a modular approach using interoperable hardware and software building blocks. In terms of protecting your networking infrastructure, CGL is an essential standard.

WHAT IS CARRIER GRADE?

Carrier grade is one of those ponderous terms in technology circles. It conjures up images of densely packed equipment rooms, constantly air conditioned and often dimly lit, with ambient noise levels from fans loud enough to discourage casual conversation. It speaks of machines that do the heavy lifting of the Internet with specialized, expensive software and hardware—purpose-built equipment focused entirely on moving data as rapidly and as efficiently as possible from one endpoint to another. Carrier grade says reliable. It says stable. It says long lifecycles, remote locations, and infrastructure. All very true.

It also says slow to adopt, slow to change, proprietary, and boring. That's where the perception is far from the reality.

Carrier grade does mean infrastructure, it does mean the blue collar labor of modern communications (that is, all communications, be they peer-to-peer file transfer, email, instant messaging, micro-blogging, streaming voice and video, even traditional voice communication), but there's been a revolution in the world of communications in the last thirty years. You've probably noticed it. Gone are the days when voice and data channels were laid like highways as permanent fixtures. A year from now, a month from now, even a week from now, the demands on a communications link might be completely different from what they are right now, and everything you know about your customer base might be different. Even what it means to be a carrier has changed, with Google both providing fiber-optic broadband services (through Google Fiber) and preparing to enter the wireless carrier space.[1]

Carrier grade still means moving data as quickly and reliably as possible, but it also means having the foresight to plan for the future and the ability to adapt when your predictions turn out to be off the mark. It means engineering your system to give you the best possible performance for the long term, not just for the next 18 months. It means being ready to change not just your system configuration, but possibly your whole business model, to thrive in the new world—and it means being able to do so on hardware with expected in-service times measured in many years.[2]

WHAT IS CARRIER GRADE LINUX?

It then follows that Carrier Grade Linux is some version of Linux that is capable of doing all of these things. It's a version of Linux that is uniquely suited to managing data centers, routing traffic for

large Internet service providers, and switching telephone calls. It can be deployed into either the network core or remote link points that will only be visited when service calls are placed, ideally by the operating system and software itself, but sometimes by passers-by. It knows how to handle highly mobile IP traffic (i.e., IPv4 and IPv6) traversing multiple networks while maintaining constant connectivity. Perhaps most importantly, it resists the constant storm[3] that is the Internet, full of hostile parties and software all intent on finding and exploiting holes that have never been seen or even imagined before.[4]

Well, unless you've been somewhere very, very remote for the last couple of decades or so (the Linux Security website was launched in 1996), you know that pretty much everything we've just established as Carrier Grade Linux is just plain old Linux and has been for a very long time.

So what makes Carrier Grade Linux different from any of the other desktop or embedded distributions out there? Fortunately, the Carrier Grade Linux workgroup at the Linux Foundation has already done much of the work of answering this question for us. It has published several versions of the Carrier Grade Linux specification (the most recent being 5.0, in April 2011), laying out in detail the features and attributes that must be present in a Linux distribution if it is to claim to meet the needs of the carrier space. To define Carrier Grade Linux, then, it makes sense to begin with how the Carrier Grade Linux workgroup defines itself.[5]

We should note first that a combination of hardware and software can be carrier grade—that is, compliant with the Carrier Grade Linux 5.0 specification—without necessarily implementing all of the requirements in the specification. For example, if the combination does not include an Advanced Telecommunications Computing Architecture (ATCA), CompactPCI, or BladeCenter format hardware platform, there is no need for that Linux to be compliant with the Carrier Grade Linux 5.0 hardware requirements.

Note, also, that the latest version of the specification does make a significant change in the way requirements are prioritized. Breaking with history, for this version the full list of requirements were reviewed and re-evaluated, and many of the older requirements were marked deprecated, due either to lack of relevance or to being so commonplace that any modern Linux distribution would already have such features implemented. In a similar vein, what used to be identified as P3 requirements were re-categorized and given the more accurate GAP identifier. All GAP items have been moved into a separate section of the requirements document.

However, this latest version of the specification does maintain the seven broad categories first defined in version 4.0. While the books have been recombined into a single specification, it's still reasonable to consider each category individually.

Standards

In many ways, Carrier Grade Linux can be thought of as a convergence of other specifications. After all, the original intent of the specification was to identify functionality already available in the open source community that is of particular interest to carriers. The best example of this is the Standards Requirements Definition section of the document. The first requirement, STD.1.0, concerns compliance with the Linux Standard Base (LSB) specifications. The section goes on to identify specific pieces of POSIX standards that must be met and references IPv6 and IP Security (IPsec) standards, Requests for Comment (RFCs), and PCI Express specifications. All of these requirements are of interest to a much wider group than just carriers and network administrators—IPv6 and the mobile extensions for it, for example, are key for supporting new, highly mobile Internet-connected devices.

Of course, there are also requirements that are focused more exclusively on carriers, but even some of the more networking-focused requirements are not necessarily the exclusive domain of carriers. Many of the new frameworks that are looking to provide carrier grade capabilities are focused on extensions toward virtualization technologies such as OPNFV, NFV OpenStack, OpenFlow/OpenDaylight, and so forth. These technologies are rapidly becoming the standard for managing large data centers for service providers.

Hardware

Easily the most carrier-centric component of the Carrier Grade Linux specification, the Hardware Requirements Definition describes hardware-specific support considered necessary for carrier environments. Half of the requirements specifically called out in the document are focused on support for the ATCA. Such platforms are typically not well suited to use in large data-center environments. This description also applies somewhat to the other major technology included in the Hardware Requirements Definition, CompactPCI.

But the section also includes requirements that may be of interest to non-carriers. It does, for example, include references to BladeCenter, which comes in a variety of configurations, including some with internal, fast, high-capacity storage such as the BladeCenter S, suitable for mid-sized database and server farm configurations.

Serviceability

The core concepts behind the Serviceability Requirements Definition—essential features for servicing and maintaining a system—most definitely arise from the carrier space. The focus of the section is on the ability to remotely monitor, configure, and patch or change software loads on platforms that are in service and may not even be accessible without considerable cost or effort. Highlights from this section include all of the methods familiar to network equipment manufacturers. Meeting Intelligent Platform Management Interface (IPMI) and Hardware Platform Interface (HPI) specifications ensures applications can communicate with the base hardware and remote management controllers to report status and receive critical configuration updates. Additional services extend this functionality to better support multiple machines working in the same chassis to provide hot failover functionality and ensure seamless operation, even during scheduled downtimes or in the event of a hardware failure.

Increasingly, though, these features and monitoring abilities are becoming not just valuable but essential to small, low-power devices with intermittent connectivity. The Internet of Things (IoT) is made possible by machine-to-machine (M2M) technology, which depends on the ability to remotely monitor and manage devices to maintain the health of both the device and the network as a whole.

But the Serviceability Requirements Definition also encompasses things like software installation, upgrades, and rollback of failed upgrades. It addresses support for diskless systems and support for remote booting. There are requirements for providing kernel and user space profiling features that aid in both system development and subsequent tuning, and there are requirements addressing how the system should handle boot failures at different stages and with different levels of severity. There are even requirements specifications for application and system checkpointing and restarting. All these requirements are of great interest to carriers and service providers who have very large, very complex systems, but increasingly these same requirements are cropping up on the very small, highly mobile targets scattered throughout the network.

Performance

The Carrier Grade Linux requirements concerning performance are not aimed at meeting particular benchmarks, but instead address performance measurement and tuning options of the operating system. Many of the requirements outlined in the Performance Requirements Definition were written to ensure timely processing of packet data through a switch. To this end, Carrier Grade Linux specifies a number of attributes of the operating system that must be run-time configurable. These specifications include, for example, how various interrupts are handled and in what priority; they ensure priorities are flexible, based on rules that describe when time-sensitive operations must temporarily receive higher-than-normal priorities. They also include how processes are scheduled on and migrated across different cores or whole CPUs in the system. The Performance Requirements Definition also identifies key real-time functionality and high-resolution timer support that enables this level of flexibility and ensures that service guarantees are met. Finally, some requirements are focused on improving generally high system performance with event-driven message delivery and application pre-loading.

These requirements are written in such a way that they are applicable to many other computing environments, both low power and small size. In fact, many of the requirements described in the performance section are sometimes better suited to non-carrier applications, as these days the default settings for these values in Linux are already well-suited to carrier environments.

Availability

The Availability Requirements Definition identifies itself as "... requirements that apply to the Linux kernel, core libraries, and tools essential to a carrier-grade system. These Availability requirements are related to single system availability, such as support for memory failure detection."

In fact, the availability section does contain requirements regarding detecting, reporting, and taking actions based on hardware and software failure conditions, which is something of interest to all devices—very large or very small—but the section covers much more than that one narrow area. There are requirements that focus on handling low memory conditions, both reporting them and responding to them, up to and including replacing the out-of-memory (OOM) killer entirely. The section also specifies kernel and application monitoring and debugging capabilities. It further includes some of the most complex requirements, those for quickly restarting the system in the event of a catastrophic failure (which are almost universally implemented by distributions through kexec). The requirements in the availability section historically extended to live patching of applications and even the kernel. These have been moved to the GAP section now, but similar requirements around updates and graceful migration from one software load to an updated one are still present throughout the section.

Clusters

Where the Availability Requirements Definition focuses on the individual system operating as an entity, the Clustering Requirements Definition takes a macro view, describing how many Carrier Grade Linux systems can interact with each other to achieve the goal of providing highly available, redundant systems using hardware that may not necessarily have any inherent capabilities to perform in that manner. The Carrier Grade Linux specification groups clusters in four broad categories:

1. High availability (HA) clusters
2. Scalability clusters
3. Server consolidation clusters
4. High performance computing (HPC) clusters

Clustering requirements include such features as arbitrating access to shared storage and ensuring it is rapidly and accurately replicated throughout the cluster; ensuring multiple, redundant communication paths between nodes with rapid failover when necessary; and node health monitoring and management, including STONITH capabilities. While these features are not necessarily of much use on the very smallest of computer systems, their utility is perfectly clear in both carrier environments and in large data centers.

The Clustering Requirements Definition is not just useful for the new networking devices in the IoT era, it concerns their base-level functionality. It addresses the other half of the IoT story: How do all of these intelligent, connected devices share work among themselves?

The open source community is constantly evolving, and as a result highly active projects and groups are looking into expanding data center clustering into a cloud environment in order to keep up with the demand for more bandwidth and reduced operational costs. For many telecom service providers, the high availability requirements have been paired with Network Functions Virtualization (NFV) technology in order to better manage traffic, deploy new services, and operate their networks that guarantee an "always-on" service uptime.

Security

Perhaps the furthest removed from any carrier-specific purposes, the Security Requirements Definition addresses several broad categories of security requirements that are at least as relevant, if not more so, to enterprise and Internet companies as to traditional carriers. The security requirements attempt to address features that make the system both more reliable and more resistant to attacks, both external and internal. For example, mandatory access control and role-based access control (RBAC), often provided by Security Enhanced Linux (SELinux) and a strong security policy, ensure that the system cannot be compromised by accidental or intentional attempts to exceed normal privileges by regular system users.

Requirements regarding buffer overflow protection and file system, memory, and execution quotas ensure that the system is protected from accidental programming errors that could compromise the ability of the system to perform important tasks. System integrity mechanisms such as the Trusted Computing Group's Trusted Platform Module (TPM) and cryptographic hashing of all system binaries, configuration, and user data ensure that the equipment has not been modified in any unauthorized ways. It is also essential for the system to provide reliable communications mechanisms, supporting IP security features as well as Secure Socket Layer (SSL)/Transport Layer Security and Public Key Infrastructure for applications on the system.

These are all features that are just as applicable, and just as desirable, on a tablet or an intelligent sensor in a factory as they are in the networking closet.

WHY DO I CARE?

If you're a carrier it's probably already obvious why you care. You need an operating system that provides high availability, reliability, security, and mechanisms to apply patches in the field and do upgrades during scheduled maintenance periods. And you need all of that as well as all of the tools to actually provide the service you're selling. Ideally you'd rather not build it yourself, too, since you don't make any money off the operating system—the value you contribute is in the applications on your platform, so that's where you want to spend your time.

So you're going to buy an operating system. Which one? There are more operating systems out there than you can ever reasonably evaluate, all of them with their own unique strengths and weaknesses. After some analysis you conclude—as have so many before you—that you want some form of Linux. It's great, it's open source (so you can modify the pieces as you see fit as long as you abide by the terms of the particular open source license), there's an incredibly large community supporting it, and every single hour of every

single day new applications and new features are being developed for it. How can you beat that level of innovation? Best of all, you can buy it, get all of those great features, but also get dedicated support and maintenance. It's basically a dream come true.

All right, Linux it is, then. There's only ... more than six hundred fifty versions to choose from, according to Linux Weekly News (http://lwn.net/Distributions/).

Now you're almost back to square one. How do you know which Linux will meet your needs? Well, now you care about Carrier Grade Linux. Why? Because if you look at a Carrier Grade Linux registered distribution (all such distributions are listed on the Linux Foundation website), you already know whether that distribution supports the parts of Carrier Grade Linux that you care about—and you know how they support it, which patches or packages they integrate, and can do your own evaluation of how well their technology meets your needs before ever picking up a phone or sending off an email.

You're not a carrier? It turns out that's not important, either, because as we saw earlier the bulk of the Carrier Grade Linux requirements are in no way restricted to telcos or network equipment providers. Carrier Grade Linux, at its core, really is about reliability, speed, flexibility, and connectivity, which generally applies to everything you want to run Linux on.

WHERE IS CARRIER GRADE LINUX?

Carrier Grade Linux belongs everywhere, then? Probably not. The distributions that are normally Carrier Grade Linux–registered tend to be strong and well suited to some areas and not so strong and not so well suited to others. So what are the areas to which it is best suited?

Carriers

The first area is so obvious it only warrants brief mention. Of course Carrier Grade Linux belongs in carrier environments—that's what it's built for, after all. Carrier Grade Linux distributions are geared toward voice and data networks that have extremely high throughput and reliability (that is, up-time) demands. Usually these environments standard policy with the addition of * up—often in terms of dollars, sometimes in much more valuable terms. If a phone network goes down due to high usage and a 911 call is dropped, the consequences can be a lot worse than lost revenue.

Secure Environments

Is Carrier Grade Linux suitable in medium and high security environments? Absolutely. The security requirements already laid out in the Carrier Grade Linux specification address all of the basic requirements that are needed to implement a medium-robustness Common Criteria protection profile. A specific profile is not required, but applying one on top of a full Carrier Grade Linux system is a much easier task than retrofitting a general purpose or desktop Linux distribution for use in an environment where Common Criteria standards must be applied.

Note, too, that the term *secure environments* here applies generally to businesses that need to implement high separation of duties amongst employees. Financial institutions, for example, need to have very strict roles and responsibilities for their employees, and there are Carrier Grade Linux requirements in the security section specifically describing what must be present in terms of RBAC.

Data Center

Typically the data center is more of a destination than a stop along the way, so the traditional requirements for a data center operating system have been quite separate from those running on equipment at the network boundaries, the gateways to the data center.

The world has changed, though. It's no longer true that you can't have a single operating system that serves the needs of your networking equipment and your data warehouses. In fact, that's not even a particularly desirable situation anymore. Having a common operating system throughout your core network, all the way from the edge to the massive database or file server at the center of it all, can dramatically reduce your maintenance and administration overhead. To be sure, there are benefits to having a heterogeneous network environment, but most of those benefits are related to containing and restricting the damage if a node is compromised. Given the security requirements already present in a Carrier Grade Linux distribution, these concerns are largely alleviated—so you can go back to appreciating how much easier it is to work in an environment where all of your machines behave pretty much in the same way. The major benefit of such an environment is that things just work, and that's tough to beat.

So there are several Carrier Grade Linux requirements that support the data center. They include the availability requirements around ensuring each node is able to monitor itself and take appropriate actions when something starts to go wrong, and a software

redundant array of independent disks (RAID) on systems that don't have hardware RAID support already. They include the clustering requirements such as block-level data replication and technologies such as ATA over Ethernet (AoE) and iSCSI. They also include cluster management tools that ensure the boundaries between the data center and the rest of your network remain up and operational, even if there is a catastrophic hardware failure that brings down an individual node.

Cloud

Everything we said about the data center applies to the cloud too. If the cloud can be thought of as a kind of data center, then the rest of its network extends out to coffee shops and bookstores and apartment complexes and Internet kiosks all over the world. Carrier Grade Linux even has specific clustering requirements to support one of the most exciting possibilities of the cloud, high performance computing (for some fascinating use cases, see the Open Cloud Consortium working groups at http://opencloudconsortium.org/working-groups). Indeed, Carrier Grade Linux is really already providing the framework of the cloud. All that remains is to figure out how best to leverage what's there to achieve the new goals, making sure everyone is connected all the time and can access their personal information both transparently and securely—concepts that are both at odds with each other and essential to the successful migration to the next great computing platform.

Internet of Things

Probably the best fit for Carrier Grade Linux in the IoT space is on the server (or management) side of the network of devices. The devices themselves need to be extremely small, extremely simple, and extremely lightweight. But even on these devices there are parts of the Carrier Grade Linux specification that are important. As we mentioned earlier, the connectivity, load-sharing, and redundancy features that are essential to the operation of a carrier's network are just as essential to implementing a reliable network of monitoring devices to ensure smooth operation on a factory floor, for example. The security features required by carriers also ensure that devices deployed into the field can continue to be trusted and safely managed once they're no longer under the owner's control. The health monitoring features that carriers need to ensure they meet their up-time requirements are either similar or identical to the features that service providers need to know when a customer's set-top box or portable scanner are in need of servicing or a software upgrade. All of these concepts that really took root in the carrier space are finding a new home in the Internet of Things.

CARRIER GRADE PROFILE FOR WIND RIVER LINUX

Carrier Grade Profile for Wind River Linux is a unique and versatile platform that can be used everywhere, from the expected (the network edge) to the commonplace (the corporate network core) to the completely unexpected (for example, high security environments and mobile devices). This flexibility derives both from the specific combination of packages and features in Wind River Linux and from the configuration options presented to developers using it as a base. Let's look at some of its features, starting with the core system (and how it has changed from earlier versions of Wind River Linux and the Carrier Grade Linux profile that has always been a hallmark of Wind River Linux), then moving to individual components, and finally to more general features.

Yocto Project

Wind River Linux is based on the Yocto Project and the bustling OpenEmbedded community, and has a carrier grade profile. Carrier Grade Profile for Wind River Linux is the first product to meet the registration requirements of the Linux Foundation's Carrier Grade Linux 5.0 specification built for a Yocto Project Compatible product. Similar to an OpenEmbedded metadata layer, Carrier Grade Profile builds on top of the core Yocto Project base—but is much more than just the Yocto Project.

The OpenEmbedded meta-networking layer forms an integral component, providing the open source "glue" that allows Carrier Grade Profile to draw upon the Yocto Project while providing the rich feature-set described here. For its part, meta-networking provides services such as File Transfer Protocol (FTP) daemons (and the more lightweight Trivial FTP servers); user space tools suitable for creating robust firewalls; autoconfiguration tools such as zeroconf; virtual private network (VPN) tools such as the Cisco-compatible VPN client vpnc and the fast, powerful VPN server accel-ppp; and network configuration and authentication tools such as autofs and NIS.

Carrier Grade Profile also draws upon the meta-cgl and meta-selinux layers, leveraging the work done by the OpenEmbedded community to integrate SELinux features into the Yocto Project base. Carrier Grade Profile also extends meta-selinux upward, however, in order to provide fully the features required by the Security Requirements Definition in the Carrier Grade Linux specification.

What, then, is Carrier Grade Profile building on top of this already very powerful base?

Open-iSCSI

The iSCSI standard is described in a number of RFC documents, the first two of which, RFC 3720 and RFC 3721, describe the core functionality and a discovery method for iSCSI initiators. Essentially, iSCSI allows a system to export a locally connected storage device to remote machines using TCP as a transport mechanism for normal SCSI packet data. This is typically how storage area networks (SANs) are created.

The Open-iSCSI project (www.open-iscsi.org) implements a soft iSCSI initiator and allows Wind River Linux to discover and authenticate to (and, if necessary, de-authenticate from) other iSCSI targets in the SAN, then use remote iSCSI media as if they were locally attached SCSI hard disks. The project consists of both the kernel module necessary to provide an SCSI device and the user space tools necessary to configure and manage these devices, including a management daemon (iscsid).

Cryptodev Interface

The Cryptodev interface is middleware that provides access to the hardware cryptographic modules so user space applications such as OpenSSL can take advantage of cryptographic operations acceleration.

Using this feature, user space processes can use cryptographic algorithms provided by kernel CryptoAPI modules, thus allowing applications to take advantage of hardware accelerators. Extensions into specialized hardware allow cryptodev and ssl to be offloaded to supported platforms.

CGL Test Suite

This test suite provides many options for validating your CGL-enabled system against the CGL 5.0 standards. Tests may be run individually, in groups, or all at once with a single command.

Distributed Replicated Block Device

Distributed Replicated Block Device (DRBD [www.drbd.org]), developed primarily by LINBIT, behaves in a similar manner to iSCSI in that it provides a block-level interface to storage, where some portion of the physical storage media is across the network on a remote machine also running DRBD. It provides a software-based RAID-1 functionality through a kernel module and the associated user space tools, where the RAID mirroring takes place across a network, using TCP as the transport mechanism. Depending on the configuration of the DRBD device, data replication to nodes may be synchronous (which provides high reliability at the cost of some speed) or asynchronous (which is still synchronous in the sense that data has been committed locally, but may not yet be successfully replicated to the peer). DRBD is particularly well suited to high availability clusters and is commonly found on them.

ATA over Ethernet

A related but independent technology, AoE provides high-performance access to Serial Advanced Technology Attachment (SATA) devices using purely Ethernet as the transport mechanism. Removing the IP and TCP layers from the communications stack means AoE cannot be used across networks the way iSCSI can be, but the increase in data throughput can often offset the requirement that the SAN be more limited in scope.

Wind River Linux packages one of the most successful AoE implementations, AoE Tools (http://aoetools.sourceforge.net/). As with Open-iSCSI and DRBD, AoE Tools includes the kernel module necessary to make use of the technology and the vblade utility for configuring the AoE targets.

By offering the combination of AoE, DRBD, and Open-iSCSI, it is possible to create and make use of (or create a large variety of) low-cost network storage solutions.

Linux-HA

Linux-HA (www.linux-ha.org) isn't so much a single technology as the overall name of a much larger project. The goal of the project is to provide modular, reliable, interchangeable components that enable you to build a high-availability cluster. These building blocks include the well-known Heartbeat cluster membership and monitoring daemon and communication layer; the Cluster Glue local resource manager, STONITH mechanism, and error reporting and cluster communication library; and a variety of resource agents. The Wind River Linux-HA solution also packages the Pacemaker cluster resource manager (http://clusterlabs.org). With the combination of Linux-HA and Pacemaker (or Corosync (www.corosync.org), another project included in Wind River Linux) it is possible to completely manage system resources, daemons, and services from a single cluster manager, thus providing all the tools necessary to ensure services remain active and available even during hardware outages.

Virtual Routing and Forwarding

Virtual Routing and Forwarding (VRF) is tackling emerging expansion and security needs by using Linux kernel containers (LXC)

to emulate a number of systems on a single host by allowing you to define a number of virtual machines, each of which has its own independent routing table.

LXC is a lightweight system virtualization, or namespace virtualization—not a full virtualization solution such as KVM or QEMU. The container appears to provide a separate machine, but is actually utilizing the host kernel with a separate set of resources, including a separate file system, routing table, network interfaces, console, and processes.

The VRF package combines LXC, the quagga routing suite (which supports RIP, OSPF, and BGP), a union file system, and some command line utilities. This combination allows you to quickly create and manage many virtual machines, each with their own resources and routing table. This solution minimizes resource usage for each VRF instance, and because of the minimal overhead, setup and teardown of virtual machines is also very quick.

While the assumed use case is virtual routing, the package can be utilized for any application where a lightweight virtual machine is desired. For example, multiple FTP servers, each with a separate set of UNIX user logins and unique file trees, could be hosted in separate VRF instances.

Security Enhanced Linux

Books have been written on SELinux—many of them, by many different people, with different focuses and different target audiences (to see an example of the breadth of literature, perform a search for SELinux on Amazon.com). That's because SELinux is extremely challenging to use well, but it's also extremely important for ensuring a Linux system remains secure and safe for all users on it.

There are a few hard truths that have to be accepted about software in general:

1. Applications have bugs. Many of these bugs can be exploited by nefarious individuals either for their own personal gain or for other, more serious ends.[6]
2. A mandatory access control system such as SELinux can protect you against such bugs only if you have a security policy in place that describes what the application is supposed to do (as opposed to what it actually does—a critical distinction).
3. The best person to write the security policy for an application is the application developer, since only they know what the real intent of the application is.
4. The person who writes the application will almost never write the security policy for it, because they almost never run their application in a secure Linux environment.

That's pretty bleak news. Fortunately there is good news too. The SELinux project, maintained by both the NSA (see www.nsa.gov/research/selinux) and Tresys Technology (see http://selinuxproject.org), provides ready-made policies for applications commonly found in Linux distributions. In fact, they provide a few different policies with varying levels of restrictions that are suited to more and less controlled environments, including features that provide not just "levels" and "classifications" but also "incomparable categories" for some of the highest levels of containment.

The challenge, then, is making effective use of these canned policies. Even though they're provided reasonably ready to go, layering them on your existing Linux distribution can be a pretty daunting task. That's where Wind River Linux helps. Out of the box, you have the option of choosing one of three security policies. The first is the vanilla reference policy[7] provided by the SELinux project, unmodified except to the extent necessary to have it useful on Wind River Linux with minimal effort on your part. There are also two Wind River targeted policies: the standard policy, which ensures daemons and users are reasonably confined, and the strict policy, which is the standard policy with the addition of multilevel security (MLS) and multi-category security (MCS) for a more tightly controlled system. The Wind River targeted policies are the most useful of the three when deploying a system based on Wind River Linux, but the vanilla reference policy also provides many excellent examples of how to integrate your own applications on top of the carrier grade platform, saving costly development and debugging time.

IP Security

Everyone has an idea of what IPsec means, but at its core the idea is simple: Do IP, but instead of just sending each package out on the wire, authenticate and encrypt each packet on both ends. This gives both sides confidence that they are talking to whom they think they are talking to, because on the Internet it's extremely difficult to know you're seeing what you think you're seeing. So what does this mean to you? It means you can create a trusted communication path through the Internet. It makes things like VPNs possible. It means you don't necessarily even need features like SSL, which must be explicitly enabled in your application, and it makes functions like web-banking safe.

Let's consider two typical IPsec applications. By combining a daemon (or head-on device driver if your kernel supports that) implementing Point-to-Point Protocol (PPP) with an upper-level application such as Layer-2 Tunneling Protocol (L2TP, which is, of course, provided by Wind River Linux), you can create a simple VPN. This ensures that traffic going over your PPP interface will be protected from monitoring and the well-known man-in-the-middle attacks, even if the secure network you are connecting to is on the other side of the world, and you're connecting from some free, Wi-Fi hotspot. Here we're talking mainly about *transport* mode.

Extending this concept, it's also possible to connect two already private networks by means of an IPsec bridge. In this mode, not just the payload of the packet, but all the routing information is encrypted, since the traffic needs to be routed across (potentially) the public Internet, but you are likely connecting from one set of non-routable addresses on the local side of the bridge to another set of non-routable addresses on the far side. This is the other common IPsec mode, *tunnel* mode.

The concept of trust is paramount here, so it's important to understand how trust is established between the two end-points in either of the above scenarios. Wind River Linux supports a number of mechanisms, the most common being Internet Key Exchange (IKE, both version 1 and version 2). IKE sets up a shared session secret via a Diffie-Hellman key exchange using either public key techniques or a pre-shared key. IKE functionality can come from the IPsec-Tools suite in the form of the racoon daemon, from the standalone Racoon2 daemon (a separate project implementing all of IKEv1, IKEv2, and KINK protocols), or from a dedicated IPsec implementation provided by strongSWAN. All of the expected standards (X.509 for single sign-on, AH and ESP for header and payload encapsulation, and IKEv2 for mobile devices via MOBIKE RFC 4555, for example) are available either through IPsec-Tools, Racoon2, or the strongSWAN suite.

Kexec

The primary Carrier Grade Linux requirement concerning kexec is AVL.8.1: Fast Linux Restart Bypassing System Firmware. The full text of the requirement states that a carrier grade platform must provide some mechanism that accelerates system initialization by bypassing the system firmware, specifically when rebooting from one Linux instance into another.

This is essentially the definition of the kexec feature as well. It is a mechanism in the Linux kernel that allows a secondary kernel to be loaded and then *exec*'d into, in conceptually the same way a process in user space is able to prepare a program with environment settings and command-like arguments, then with one of the exec( ) system calls substitute that program for the running process. Doing this with your running kernel is exceptionally difficult, but the payoff is enormous. When the kernel feature is combined with the user space kexec tool, it is possible to reboot the machine without going all the way down to the firmware (thereby meeting the Carrier Grade Linux requirement), but it also allows you to prepare a secondary, standby kernel and to boot into that in the event of the primary kernel experiencing a panic. The fast recovery time and benefits to uptime and service level agreements for high-availability systems are obvious, but you also have the ability to capture a core dump from the primary (failed) kernel while in the new kernel for later diagnosis and offline debugging. Note also that the fast-reboot functionality—that is, rebooting without dropping down to the bootloader level—is not restricted to Linux/Linux interactions. It is possible to boot an entirely different operating system (including either a hypervisor or a bare-metal application) if that is required.

LEAVE YOURSELF FLEXIBILITY

Open standards provide a common build system and best tools for building a Linux system. Wind River is a significant contributor to the CGL 5.0 specification. The modular approach of our software and the compatibility with significant open source projects, such as the Yocto Project, improve cross-platform compatibility with continuing support for ARM®, Intel®, MIPS, and Power architectures. In turn, this increased portability for software and middleware translates into decreased cost for maintenance, and better reuse and collaboration between commercial and open source communities. Examples here are around better reuse of existing applications, more efficient software development processes, and lowering of integration and test procedures.

In addition, Wind River software building blocks leverage multiple open source technologies meeting a myriad of system-level requirements. For example, for more stringent security requirements Carrier Grade Profile can be combined with Security Profile for Wind River Linux. Security Profile leverages various kernel features and mechanisms needed for telecom platforms,

and is complemented by constant and vigorous monitoring. To combat vulnerabilities, Wind River is committed to active threat monitoring, rapid assessment, threat prioritization, expedited remediation, response, and proactive customer contact in each of the detected cases. The Wind River Security Response Team comprises members from engineering groups, customer support, field engineers, and other appropriate personnel, who together monitor security issues and formulate rapid responses.

Performance and scalability requirements in a virtualized environment can also be met by the real-time technologies in Wind River Open Virtualization. Switch services provide updates in hours and consolidate network elements, all with zero latency and downtime and no vendor lock-in.

Carrier grade products typically require at least five-nines (99.999%) availability, and rapid network traffic growth and the need for greater data security have resulted in systems that are more reliable and resistant to attacks. Carrier grade is thus a hard requirement for networking devices, but also addresses the needs of large corporate infrastructures, data centers, military communications, and highly mobile devices.

CONCLUSION

Carrier grade isn't just for carriers. No doubt you can think of many other examples where carrier grade features—either the examples we've focused on or some of the others described in the formal specification—are in use today or could benefit new devices or existing applications in new ways. Whether you are a carrier or not, though, Carrier Grade Profile brings a compelling, cohesive feature set to implement these requirements, always with an eye toward what the requirement is intended to accomplish, not only what it says literally. Carrier Grade Profile for Wind River Linux is, in short, *the* Carrier Grade Linux.

1 For example, see Burns, Chris, "Google Mobile Carrier real possibility with tipped Dish agreement," SlashGear, November 16, 2012. www.slashgear.com/google-mobile-carrier-real-possibility-with-tipped-dish-agreement-16257358/.

2 Huawei has indicated expected ATCA life cycles of 8–10 years, Emerson Network Power has similarly long expectations, and RadiSys recognizes ATCA as a long-term solution for meeting future scalability requirements.

3 As a vivid illustration of this concept, the Internet Storm Center records survival times for various operating systems in the wild in minutes.

4 For perspectives on the current and future states of cyberwarfare, see http://techcrunch.com/2013/01/21/eugene-kaspersky-and-mikko-hypponen-talk-red-october-and-the-future-of-cyber-warfare-at-dld/.

5 For the actual specification, see https://www.linuxfoundation.org/sites/main/files/CGL_5.0_Specification.pdf.

6 For more on national cybersecurity threats, see http://blogs.windriver.com/parkinson/2010/10/national-cybersecurity-strategy.html.

7 Detailed information on all SELinux reference policy types can be found at http://selinuxproject.org/page/NB_PolicyType.

WIND

Wind River is a global leader in delivering software for the Internet of Things. The company's technology is found in more than 2 billion devices, backed by world-class professional services and customer support. Wind River delivers the software and expertise that enable the innovation and deployment of safe, secure, and reliable intelligent systems.