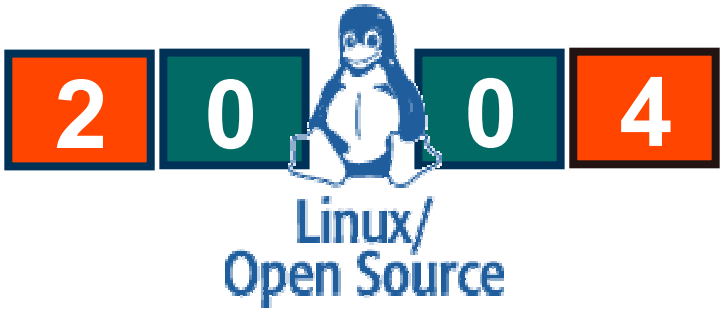
USENIX



Towards Linux Open Telecom Platforms

Ibrahim.Haddad@Ericsson.com
Open Systems Lab – Ericsson Research
http://www.Linux.Ericsson.ca

Outline

- From proprietary to open platforms
- Open, standardized, and components-based platforms
- Carrier Grade Linux: architecture, working group, requirements
- Examples of needed features
- Challenges…

Yesterday vs. Today

Traditionally …

Communications and data service networks were built on proprietary platforms that had to meet very specific requirements in areas of:

- availability,
- reliability,
- performance, and
- service response time.

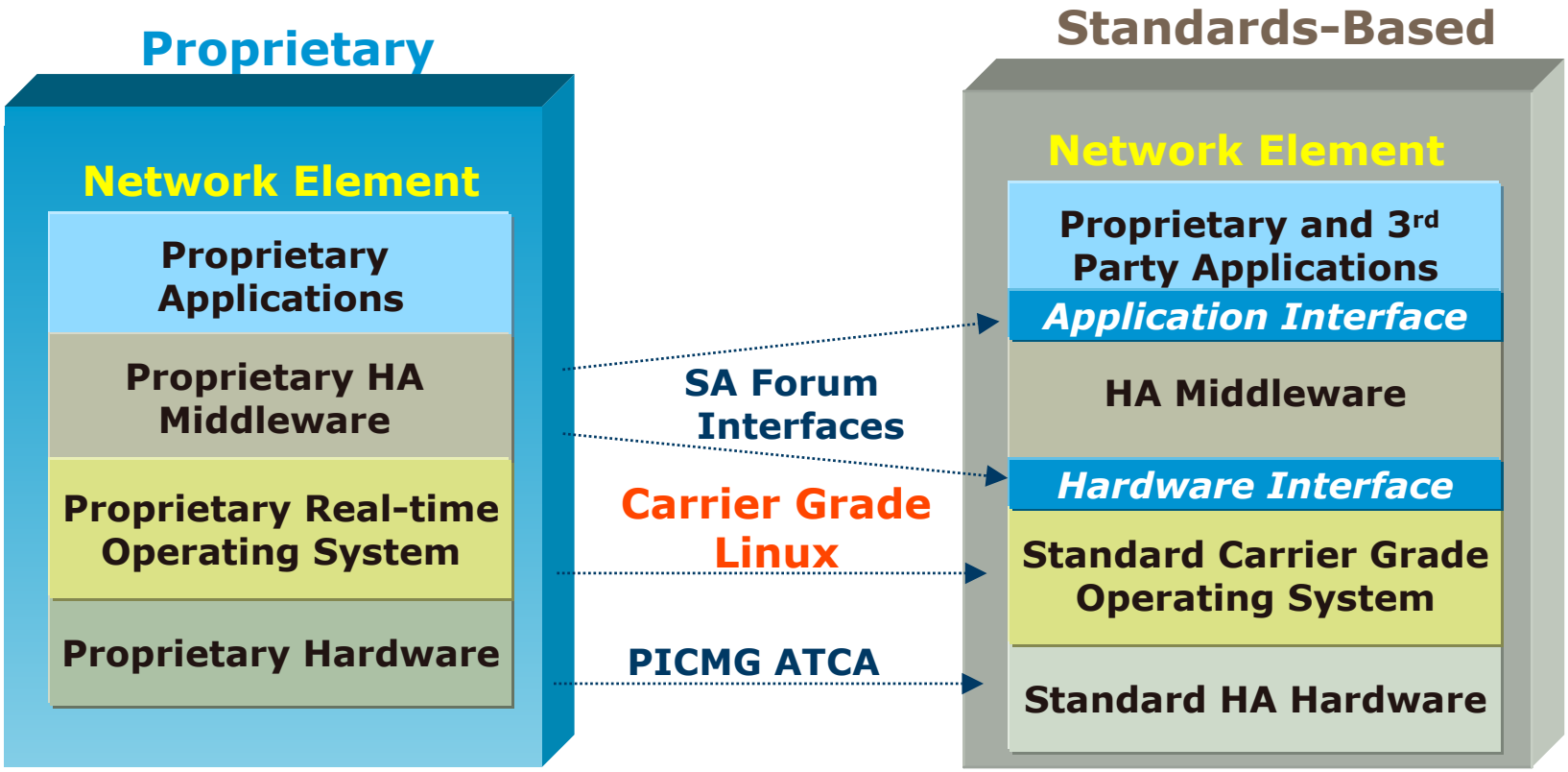
Today …

Communications service providers are challenged to meet their needs cost-effectively for new services and new architectures while maintaining highly available, scalable, secure, and reliable systems that have predictable performance and that are easy to maintain and upgrade.

Proprietary vs. Open Platforms

- Proprietary platforms are:
  - closed systems,
  - expensive to develop, and
  - often lacking support of the current and upcoming standards.
- The current trend is to deliver the next generation communication services using carrier grade platforms that are designed and implemented using common-off-the-shelf SW and HW components as building blocks.

Proprietary to Open and Standardized Solutions



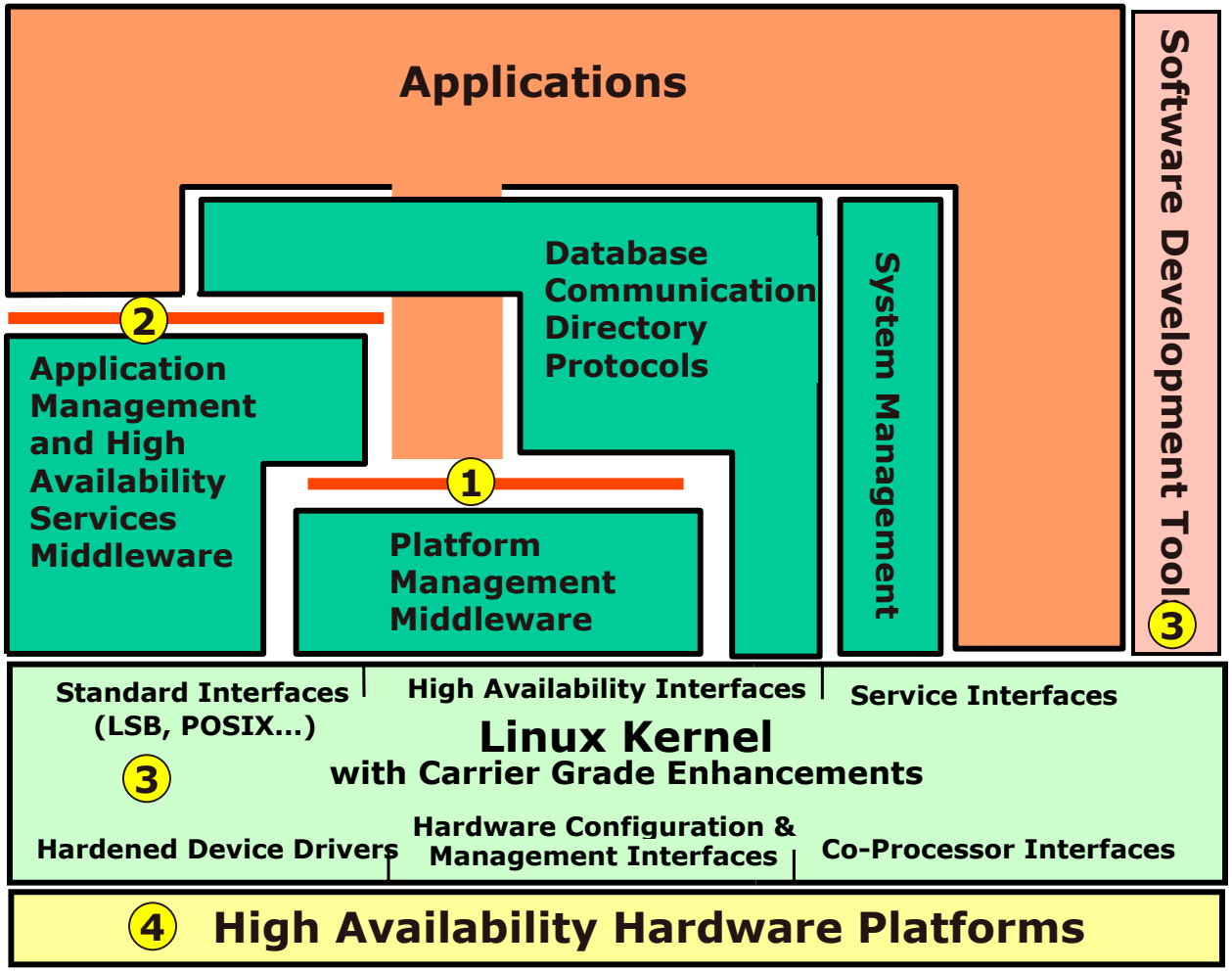
From proprietary to open, standardized, & component based HW and SW Solutions

Motivations for Open Platforms

- There are many motivations behind the trend of migrating towards open platforms that are based on open standards:
  - Ensure *portability*, and *integration* capabilities,
  - Ensure *interoperability* with third-party software,
  - Make *application development easier*,
  - Provide *faster time to market*,
  - Ensure *lower costs* using COTS components, and
  - Help *focus on core competencies* to allow faster innovation.

Who is *defining* what?

1. Hardware Platform Interface Specification — SERVICE AVAILABILITY FORUM
2. Application Interface Specification
3. Scope of SDL
4. PICMG HA HW (ATCA)



Carrier Grade Linux

What, who, why, architecture, specs, roadmap, etc.

What is Carrier Grade?

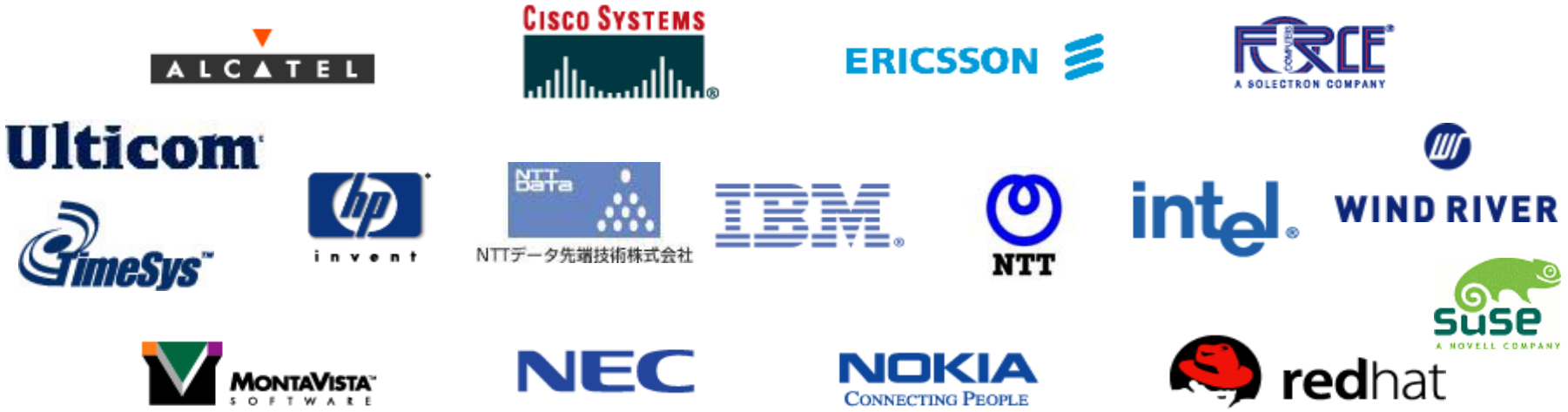
- *Carrier Grade* is a term for public network telecommunications products that require a reliability percentage up to 5 or 6 nines
  - 5 nines (99.999%) -- associated with Carrier Grade servers
    - Less than 5 minutes of downtime per year
  - 6 nines (99.9999%) -- associated with Carrier Grade switches
    - Less than 30 seconds of downtime per year

- *Carrier Grade Linux* is a flavor of Linux targeted for communication platforms.

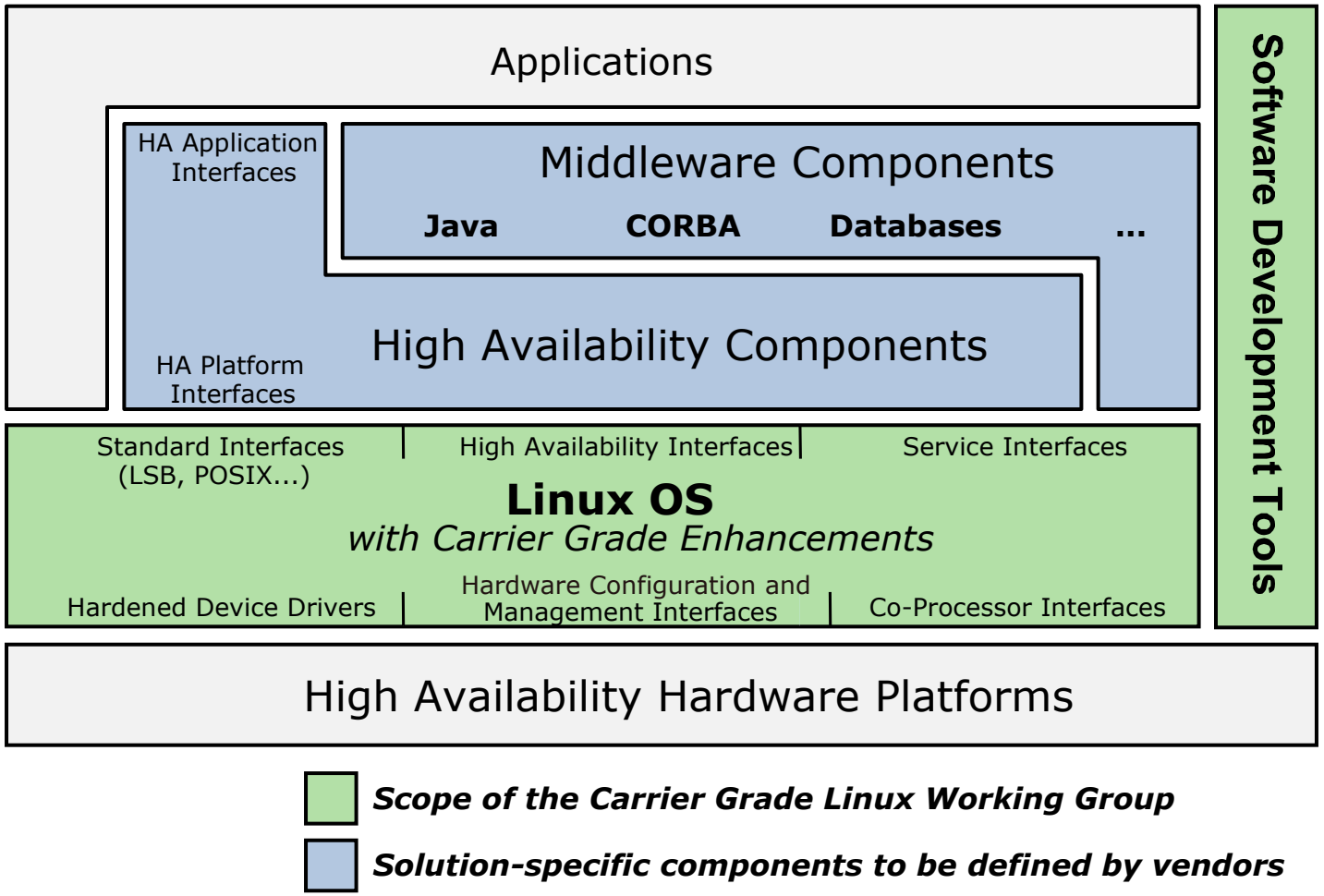
CGL Working Group

An industry forum to support and accelerate the development of Linux functionality for telecommunication applications

MEMBER COMPANIES



CGL Architecture



Carrier Grade Linux (CGL) Summary

- CGL Working Group Started January, 2002
- CGL Version 1.1 – Released October 2002
  - CGL 1.1 based distributions are already available in the market
- CGL Version 2.0 – Released October 2003
  - CGL 2.0 based distribution are expected to be available by Q1 2005
- CGL Version 3.0 – First public draft was released in May 2004

CGL Specs Requirement Categories

- *Standards*: CGL specifies standards, important to Carrier Grade servers, that are required for compliance.
  - *Examples*: LSB, POSIX, IETF RFCs, and SA Forum compliance, etc.
- *Platform*: Support interactions with the hardware platforms.
  - *Examples*: Hot swap, hot insert, hot remove, hot device identity, Remote boot, support for diskless systems, boot cycle detection, etc.
- *Availability*: Support heightened availability of carrier grade servers and aim to improve the robustness of SW components and support recovery from failure of HW or SW.
  - *Examples*: Watchdog timer, Application heartbeat, RAID Support, Disk and volume management, Hardened driver support, etc.

CGL Specs Requirement Categories …

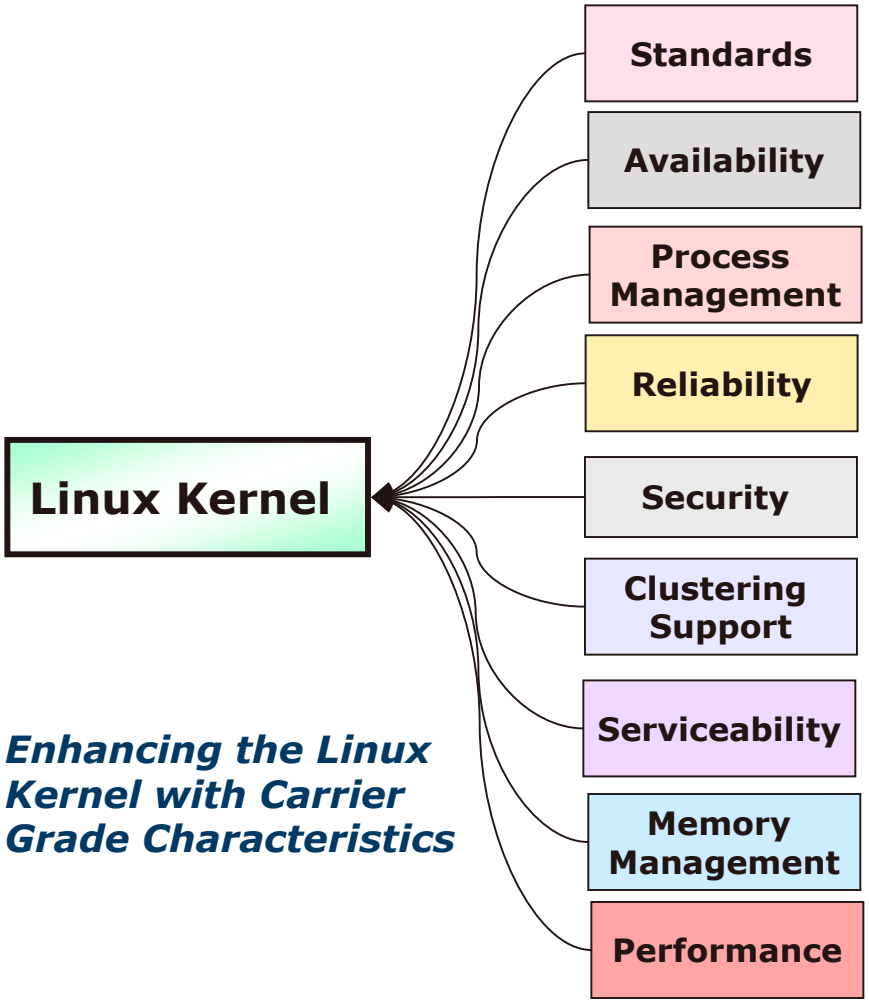
- *Serviceability*: Support servicing and managing HW and SW on carrier server systems.
  - *Examples*: Resource monitoring, Kernel dumps, etc.
- *Tools*: Support auxiliary capabilities not directly involved in normal execution of carrier server systems.
  - *Examples* debuggers used to develop modules, drivers, applications, etc.
- *Performance*: Support performance levels necessary for the environments expected to be encountered by carrier server systems.
  - *Examples*: Soft real time support, Raid 0 support, Application (pre) loading, etc.
- *Security*: Features for building secure systems. “one size fits all” is not achievable; therefore, not all features will always be used together.

CGL Specs Requirement Categories …

- *Clustering*: Requirements that support the use of multiple carrier server systems to provide a horizontally-scaled environment supporting increased throughput and to support higher levels of service availability through redundant resources and recovery capabilities.
- *Scalability*: Requirements that support vertical and horizontal scaling of carrier server systems such
  - Addition of HW resources results in acceptable increases in capacity.

Integration with Mainstream Linux Kernel

- Integration with the kernel takes time
- Some of the enhancements will be or are already proposed for kernel 2.7 integration
- Others will follow in later kernel releases
- All enhancements are available from their SourceForge or project web sites



Examples of Needed Features for Server Nodes Operating in Mission Critical Environments

TIPC, DigSig, AEM

TIPC: Transparent Inter-Process Communication Protocol

Contact info:

Jon Maloy

JonMaloy@users.sourceforge.net

Linux Inter-Cluster Communication Protocol

- TIPC is a specially designed protocol for intra-cluster communication.
  - Supports inter cluster communication too.
  - Provides a framework for supervising and reporting topology changes.
  - It is provided as a portable source code package, ~14000 lines C code
  - It has been used as a part of Ericsson products for years, deployed at hundreds of sites around the globe.
- TIPC is a useful toolbox for anyone wanting to develop/use HA/Carrier Grade Linux clusters.
  - It provides the necessary infrastructure for cluster, network and software management functionality.
  - It provides a good support for designing scalable, distributed, site independent, highly available, and high-performance applications.

TIPC Status

- TIPC & Linux Kernel:
  - TIPC is supported on both 2.4 and 2.6 kernel series
  - After receiving feedback, TIPC code was changed to make it more suitable for inclusion in the kernel.
  - TIPC was released to Open Source on February 3, 2003 and announced on LKML on June 28, 2004 (http://lwn.net/Articles/91634/)
- TIPC & CGL:
  - TIPC meets several Priority 1 and Priority 2 Cluster Communication Requirements in CGL 2.0.
- General Interest in TIPC:
  - TIPC received a lot of interest from commercial sector and Linux Distributors.
  - The IETF ForCES (Forwarding and Control Element Separation) working group is interested in using TIPC as the standard transport protocol for distributed routers.

DigSig: Distributed Digital Signature

Contact info:
Makan Pourzandi
Makan@users.sourceforge.net

Distributed Signature Verification (DigSig)

- DigSig is part of a larger project called the Distributed Security Infrastructure (DSI).
- DSI started as a research project in Ericsson in 2001 to provide a security framework for real-time distributed applications running on large scale carrier grade Linux clusters.
- DSI was released to Open Source in January 2003 under the GPL license.

DigSig

- DigSig is a kernel module that inserts digital signatures inside the ELF binary and verifies this signature before loading the binary.
  - It is based on the Linux Security Module (LSM) hooks.
- The DigSig approach has been to use the existing solutions like GPG and BSign.
- The local administrator signs binaries they trusts with their private key.
- DigSig guarantees two things:
  - If you signed a binary, nobody else can modify that binary without being detected, and
  - Nobody can run a binary which is not signed, or badly signed.

DigSig Status

- DigSig & Linux Kernel:
  - DigSig is supported on kernel series 2.5 and 2.6
  - It was announced on LKML on Sept 17, 2003 http://lwn.net/Articles/49640
- DigSig & CGL:
  - DigSig meets a Priority 1 security requirement in OSDL CGL 2.0.
- General Interest in DigSig:
  - Patched by Hardened Gentoo Linux distribution developers to be used as a secondary kernel module with SE Linux.
  - Submitted to SE Linux mailing list.
  - There is a lot of interest from commercial companies

Kernel Asynchronous Event Mechanism

Contact info:
Frederic Rossi
FJRossi@users.sourceforge.net

Why we need support for Asynchronous Event Mechanism?

- Communication Applications Requirements include:
  - A high–response rate,
  - A minimum down-time,
  - Scalability w.r.t external requests and hardware,
  - Soft Real-Time capabilities (Hard Real-Time capabilities in some cases)
- Carrier Grade Platform Requirements include:
  - Live software upgrade, hardware hot-swap,…
  - Large database, fail-over, memory utilization…
  - Huge number of processes, fault detection and prevention, application restart, process reload,…

Carrier Grade systems must handle many events quickly

Asynchronous Event Mechanism Project

- AEM implements a native support for asynchronous events in the Linux kernel and aims to bring carrier-grade characteristics to Linux in areas of scalability and soft real-time responsiveness.
  - AEM targets applications scalability
- AEM is implemented as a kernel patch and a set of modules.
  - It was released to Open Source in January 2003 under the GPL license.

AEM Status

- AEM & Linux Kernel:
  - AEM is supported on both Linux Kernel series 2.4 and 2.6.
  - AEM was announced on LKML
  - Received a lot of feedback, and undergone a lot of design/implementation changes since then.
- AEM & CGL:
  - AEM implements a Priority 1 requirement of OSDL CGL 2.0: Efficient Low-Level Asynchronous Events
- Currently …
  - AEM is undergoing testing and stabilization.

Conclusion!

Challenges

- The migration of Carrier Grade servers towards Linux is dependent on:
  - The availability of kernel mechanisms and features needed by such servers operating in mission critical environments, and
  - The integration of these mechanisms into the kernel (including AEM, TIPC, DigSig)
- Other challenges include:
  - Changing ways of working, as part of interacting with Open Source and following more open working methods
  - Avoiding duplicated efforts
  - Harmony and synergy among all efforts
  - Building enablers together

Thank you.
Questions?

ERICSSON

Ibrahim Haddad
Researcher
Research and Innovation

Ericsson Canada Inc.
8400 Decarie Blvd
Town of Mount Royal
Quebec H4P 2N2
Canada

Phone: 1.514.345.7900 x5484
Fax: 1.514.345.6105
Web: http://www.linux.ericsson.ca
Email:Ibrahim.Haddad@Ericsson.com

Resources

- Ericsson Open Systems Lab
  http://www.linux.ericsson.ca
- TIPC
  http://tipc.sourceforge.net
  http://www.ietf.org/internet-drafts/draft-maloy-tipc-00.txt
- DSI
  http://tipc.sourceforge.net
- AEM
  http://aem.sourceforge.net
- OSDL
  http://www.osdl.org

Backup slides

Linux for Telecom Platforms

Will next generation & multimedia communication services be delivered using Linux-based open standard platforms?

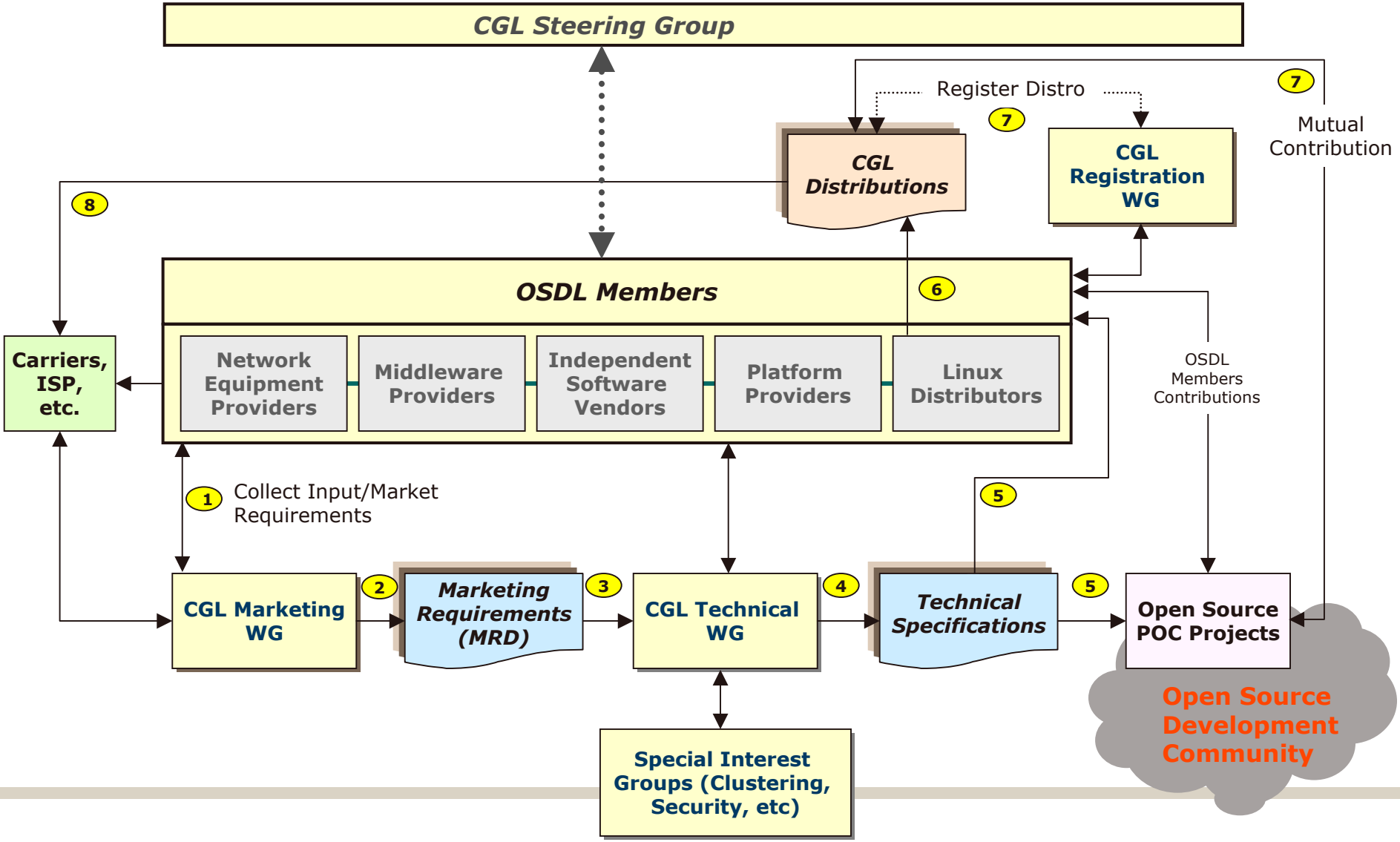
Motivations for Linux

- *Cost:* Linux is available for free. No runtime royalties.
- *Availability of source code:* We have full access to the source code allowing us to tailor the kernel to our needs.
- *Open development process:* The development process of the kernel is open to anyone to participate and contribute.
  - The process is based on the concept of “release early, release often”.
- *Peer review & testing resources:* With access to the source code, people using a wide variety of platforms & compiler combinations can compile, link, and run the code on their systems to test for portability, compatibility and bugs.
- *Vendor independent:* No longer locked-in to a specific vendor.
- *Openness:* In terms of hw, languages, interoperability, 3rd party sw.

Motivations for Linux …

- *High innovation rate:* New features are usually implemented on Linux before they are available on commercial or proprietary systems.
- *Open for all:*
  - People can contribute to Linux the required “hooks” for efficient integration of the upper-layer HA middleware.
  - People can rapidly fix faults or add features to the kernel.
- *Other contributing factors:*
  - Support for a broad range of processors & peripherals
  - Availability of commercial support
  - High performance networking,
  - A proven record of being a stable, and reliable server platform.

CGL Working Method



CGL Benefits

- For NEPs and platform providers:
  - Lower cost
  - Faster TTM
  - Leverage COTS hardware
  - Flexible service models
- For Linux vendors
  - Consolidated customer requirements
  - Customer and community support for open source developments
  - Exchange consideration for reference implementations
- For member companies (in general)
  - chance to communicate members' customers requests to tune the requirements as they go into the ecosystem
  - to reduce individual investments in the development of capabilities
  - to leverage open source expertise in working with the development community

What does OSDL/CGL expect from the community

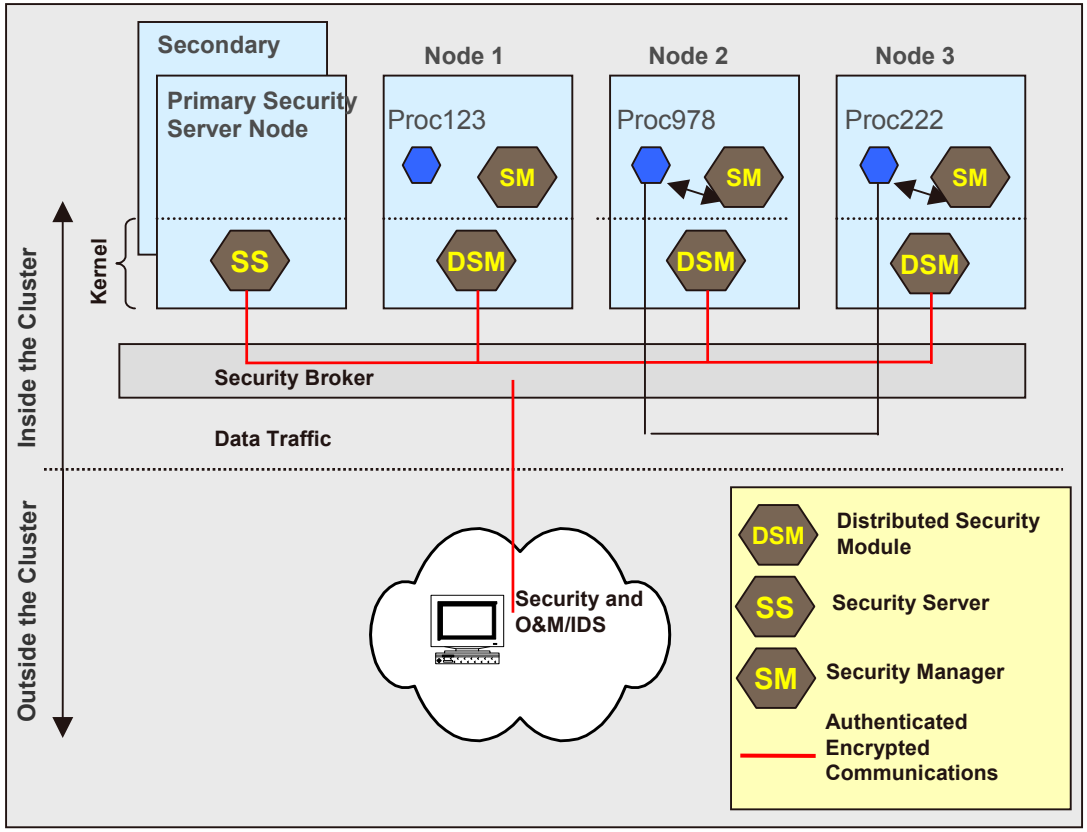
- From the development community at large: Nothing. Making demands on the development community is not only like pushing a chain, it creates animosity.

- From the member companies:
  - well defined capabilities that represent real customer requirements
  - sample/reference implementations that deserve consideration for acceptance in the open source community.
  - cooperation with other member companies in determining common objectives and capability requirements.

Changes to TIPC

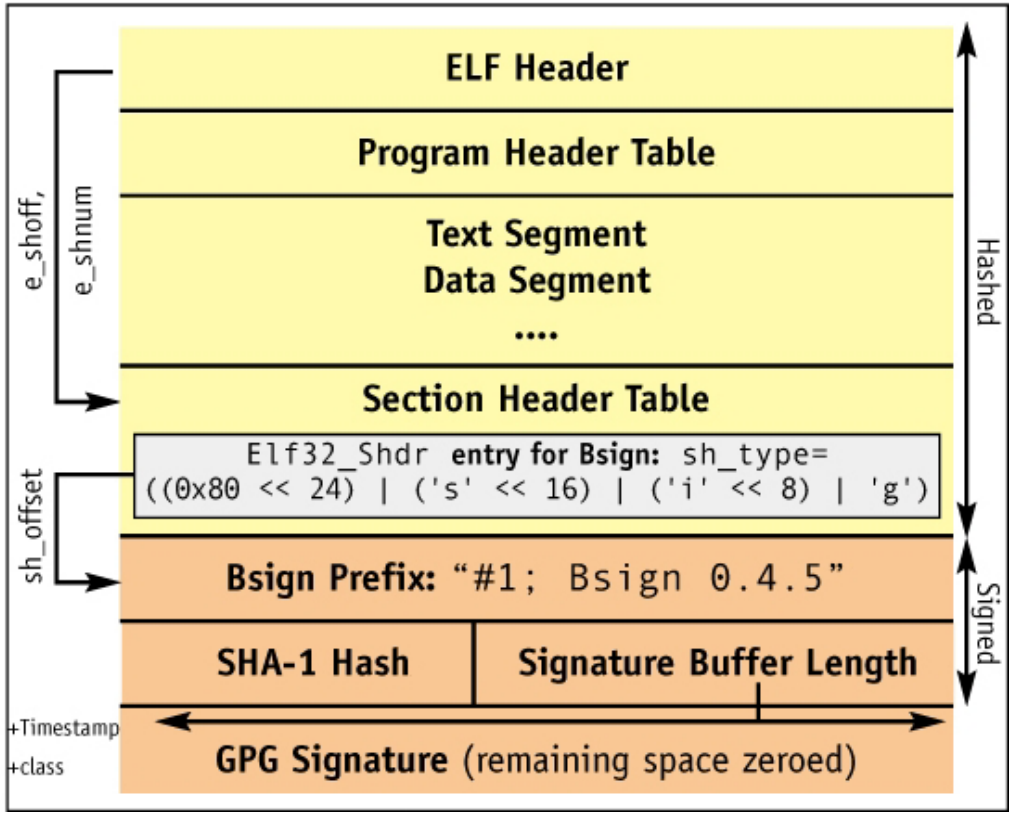
- Code style, code and directory structure
- Memory management, lock handling, and debug support has been completely rewritten.
- Adhering to the standard mechanisms and techniques used in the Linux kernel (for example TIPC now relies entirely on the linux native memory management)
- Changing configuration support,
- The API has been rewritten, and is now as conformant to POSIX
- We have modified the protocol header and added the framework for providing reliable multicast, among other things.
- We have also added an “unreliable transfer mode" which can be set per socket.
- etc…

Distributed Security Infrastructure



DigSig ...

- The DigSig approach has been to use the existing solutions like GPG and BSign rather than reinventing the wheel.
- To reduce the overhead in the kernel, DigSig took the minimum code necessary from GPG.
  - This helped reduce the amount of code imported to the kernel in source code of the original (only 1/10 of the original GnuPG 1.2.2 source code has been imported to the kernel module).



aem vs. epoll

- AEM and epoll are quite difference mechanisms:
  - AEM provides a generic asynchronous support to applications
  - epoll is a synchronous (polling) system call for read/write/exception changes on standard descriptors.

- AEM provides many modules/functionalities that are not in the scope of select/poll/epoll:
  - POSIX timers
  - File change notification
  - TIPC asynchronous interface
  - Asynchronous accept()
  - Asynchronous closing socket notification
  - Process death notification
  - More coming soon …

On Performance …

- AEM targets application scalability
- epoll is a select/poll optimization

- Benchmarks:
  - A benchmark is available on the web site of AEM `http://aem.sourceforge.net`
  - AEMhttpd is an adaptation of dphttpd written to benchmark epoll and figures are available from: `http://lse.sourceforge.net/epoll/index.html`

Support for Multi-FIP (Multiple Forwarding Information Bases)

Background

- Routers are core elements of modern telecom networks.
  - They propagate and direct billion of data packets everyday
  - They must operate as fast as the medium in order to deliver excellent quality of service and have a negligible effect on communications.
- The Linux IP stack works fine for home or small business routers.
  - We are able to achieve around 2.000 routes/sec.
- However, with the high requirements and the new HW capabilities, it appears as barely possible to use Linux as an efficient forwarding and routing element of a high-end router for large network (core/border/access router) or a high-end server with routing capabilities.
- Our target with Linux is to achieve is a predictable performance from 10.000 to 500.000 routes per second.

Support for Multi-FIP -- Problem Statement

- Lack of support for multi-FIB with overlapping interface's IP address
- Lack of appropriate interfaces for addressing FIB
- Limited scalability of the routing table
- Lack of predictable performance: In environments that require predictable performance, different kinds of problems arise:
  - We are not able to predict access time, because of the chaining in the hash table of the routing cache.
  - The route cache and the routing table are not kept synchronized most of the time (path MTU, just to name one).
  - The route cache flush is executed regularly; therefore, any updates on the cache are lost and they need to be rebuilt.

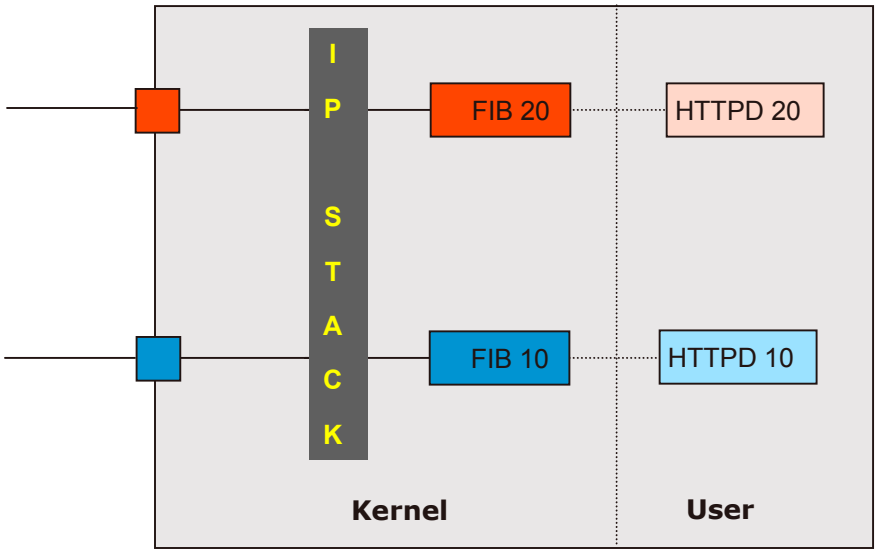
What is the solution?

- Solution: Provide support for multi-FIB with overlapping IP address, as such, we can have on different VLAN or different physical interfaces, independent network in the same Linux box.

- Advantages: We can have one Linux box serving two different customers using the same IP address. ISPs adopt this approach by providing services for multiple customers sharing the same server (server partitioning), instead of using a server per customer.

Example

- We can have two HTTP servers serving two different networks with potentially the same IP address. One HTTP server will serve the network/FIB 10, and the other HTTP server will serves the network/FIB 20.



How to achieve this?

- The way to achieve this is to have an ID (an identifier that identifies the customer or user of the service) to completely separate the routing table in memory.

- Two approaches exist:

  1. Have separate routing tables, each routing table is looked up by their ID; within that table the lookup is done on the prefix.
  2. Have one table, and the lookup is done on the combined key

     Key = ID + Prefix

Support for Multi-FIP -- What is needed?

- To support routing requirements of server nodes operating in high performance mission critical environments, Linux should support:
  1. An implementation of multi-FIB using tree (radix, patricia, etc.):
     - It is very important to have predictable performance in insert/delete/lookup from 10.000 to 500.000 routes.
     - It is favorable to have the same data structure for both IPv4 and IPv6.
  2. Socket and ioctl interfaces for addressing multi-FIB
  3. Multi-FIB support for neighbors (arp)
- Affected areas in the kernel:
  - Network layer (large changes): `net/core`, `net/ipv4`, `net/ipv6`
  - Transport layer (minimal impact): socket, TCP, UDP, RAW, NAT, IPIP, IGMP, etc.