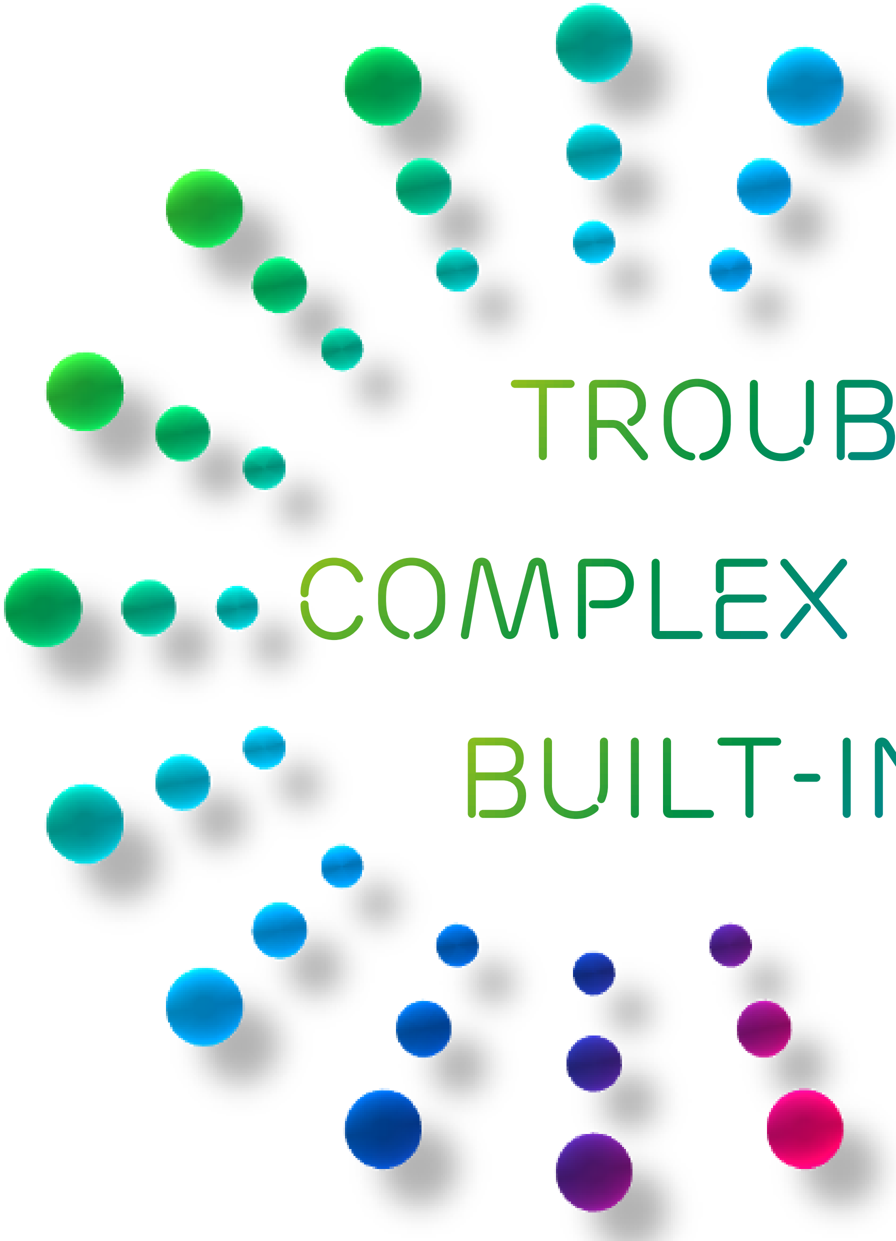


ERICSSON

TROUBLESHOOTING COMPLEX PROBLEMS WITH BUILT-IN DIAGNOSTIC

DOMINIQUE <DOT> TOUPIN <AT> ERICSSON <DOT> COM

COMPLEX DIAGNOSTIC

- Linux in: Safety critical, High throughput/Low latency, High availability, Real time systems

- Usually NO system admin
- Problems can be hard/impossible to reproduced in the lab
- Trace is the only approach to get enough data to pin point the problem and fix it

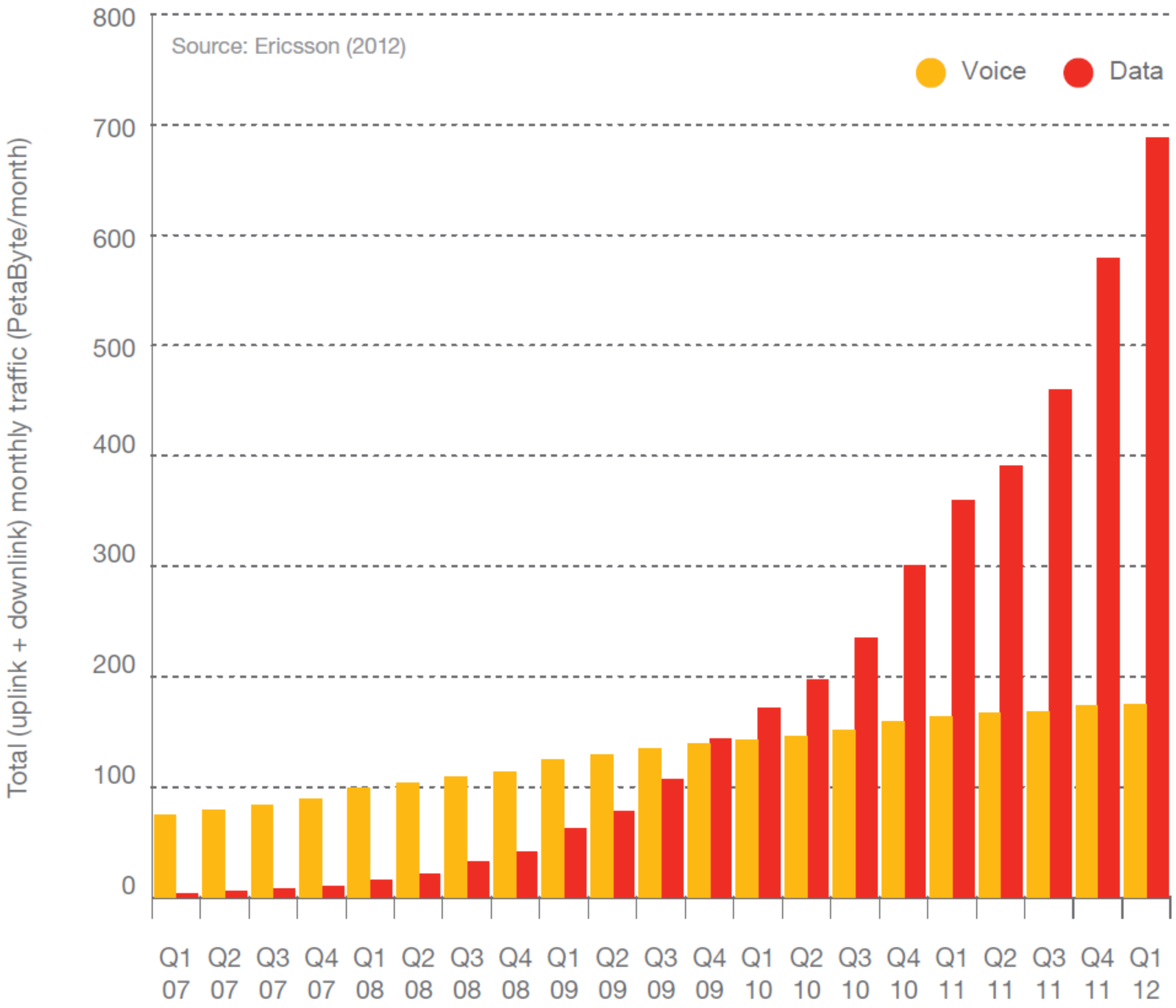
- For enterprise systems with system admin, it eliminates lengthy debug cycles!

NYSE
The world puts its stock in us.
NYSE
NYSE

SPACEX

MOBILE TRAFFIC

Figure 14: Global total traffic in mobile networks, 2007-2012

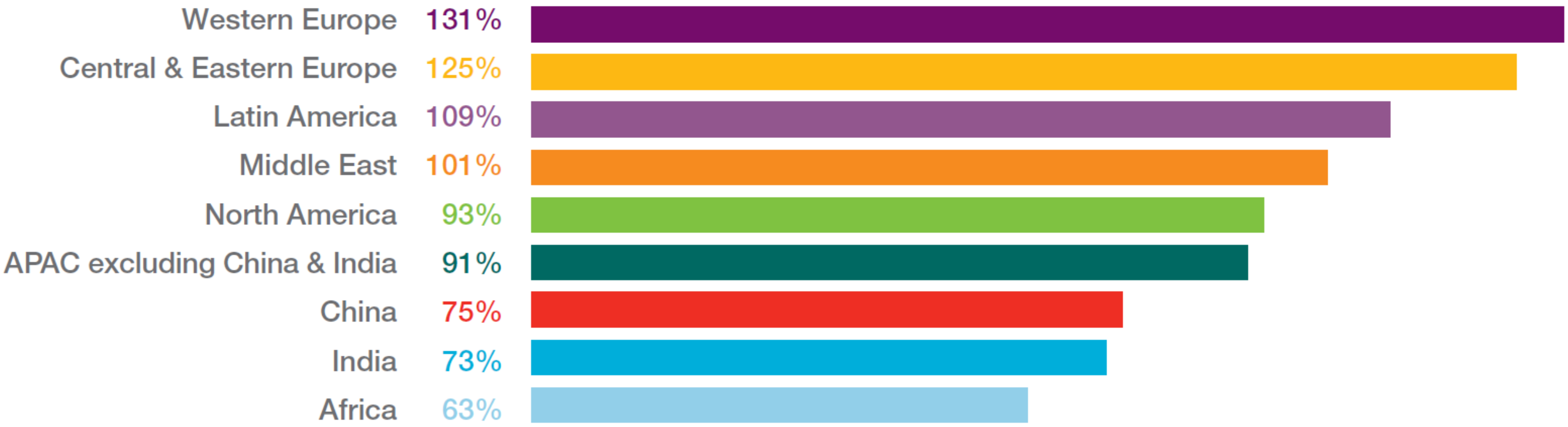


2012-08-28

http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf

MOBILE SUBSCRIPTIONS

Figure 3: Mobile subscriptions penetration in Q1 2012



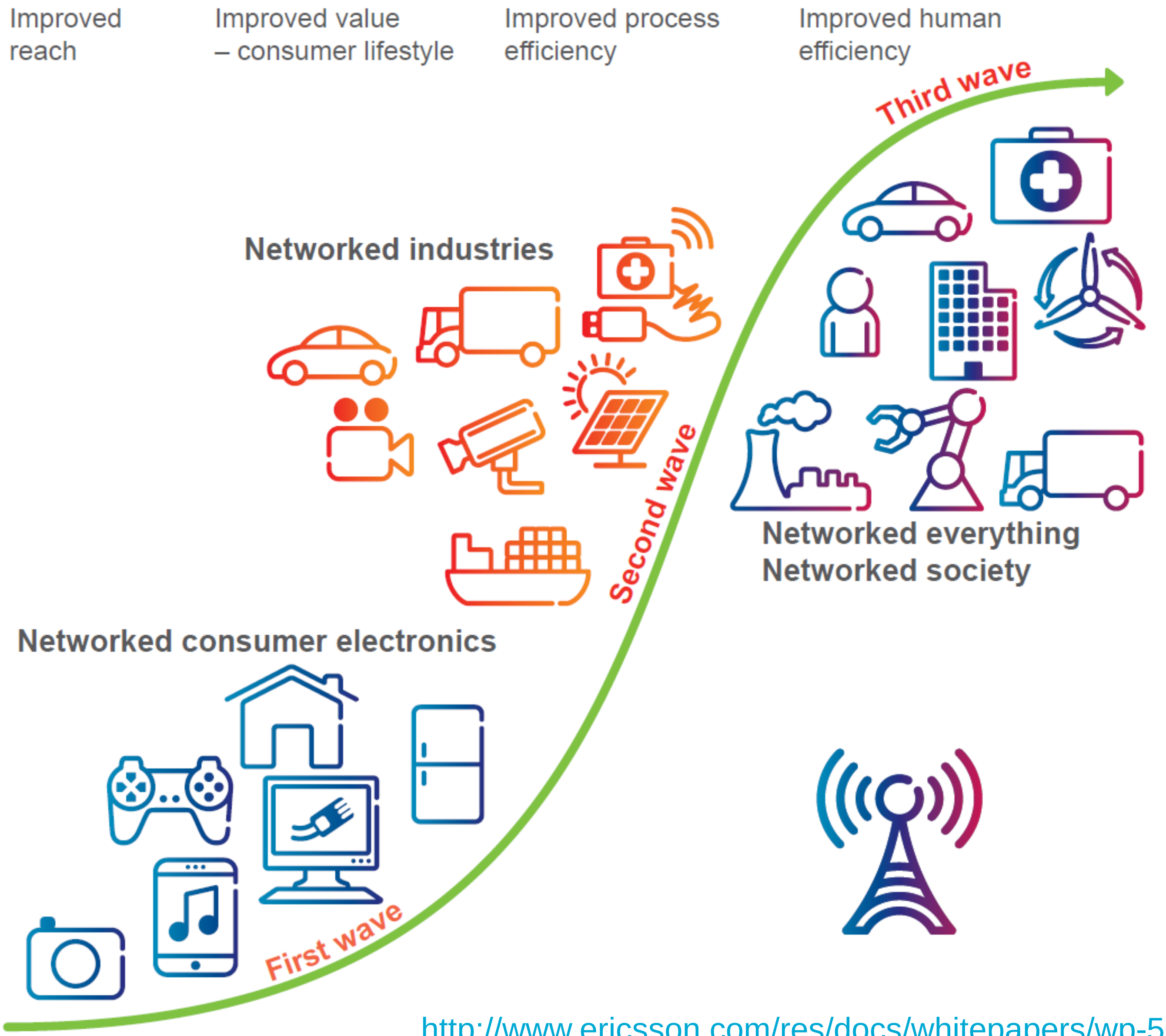
Source: Ericsson (June 2012)

[1] Mobile broadband is defined as CDMA2000 EV-DO, HSPA, LTE, Mobile WiMAX and TD-SCDMA.

One of the requirements of LTE (4G) is to provide downlink peak rates of at least 100Mbps. The technology allows for speeds more than 300Mbps and Ericsson has already demonstrated the next step of LTE at the MWC 2010 with theoretical peak rates up to 1.2Gbps

http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf

50 BILLION CONNECTED DEVICES BY 2020



Improved reach
Improved value – consumer lifestyle
Improved process efficiency
Improved human efficiency
Third wave
Networked industries
Second wave
Networked everything
Networked society
Networked consumer electronics
First wave

2012-08-28

http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf

THINK BEFORE YOU ACT APPROACH

- Develop with diagnostic in mind
  - Adding ad-hoc trace data leads to useless data
  - Trace data has to be a design activity
  - Company-wide process with design rule
- Trace data should represent the wisdom of developers who are most familiar with the code
- Rest of the world should be able to use it to extract a great deal of useful information without having to know the code
- Low overhead is key, more tracing means faster bug fixes, better system monitoring.

RELEVANT DATA

printf("Program X Init", tracefile) is not enough!

- Settings/Config: port, mode, pins, frame, speed, etc.
- Settings/Config ***changes***!
- Key global variables, state of the state machine, sensor readings, etc.
- Timestamp, time of day

RELEVANT DATA

- Message passing info, e.g.
  - sent or received
  - Who is the sender and receiver
  - Size of message
  - Message type / command
  - First few bytes of message data
- Version of the program
- Where did the tracepoint came from,e.g. main@subdir/test.c:5

WHAT NOT TO TRACE

- Confidential Data
- Classified Data
- Special Military Data

TRACE OVERHEAD

- Trace is not overhead, it adds value to your system
- How much will it cost to fix bugs without trace data?
- Long debug cycles are damaging a company reputation
- Still, try to keep it under 5%
- Optimized tracer can maximize the amount of trace event by a factor of 200!

TRACE DESIGN

- Evaluate amount of total trace storage

- Optimize the trace size
  - Sometimes need to remove less essential trace info
  - Store it less frequently
  - Detect when system is idle
  - Use circular buffer and overwrite oldest data

- Cryptic binary can shrink trace data by a factor of 10!

- Streaming to offload nodes or do live monitoring

TRACE DESIGN

- Test trace data in the lab by injecting problems, is the trace data good enough to fix the problem?
- Do not provide an ability to disable essential trace point
- Name space division in order to guarantee uniqueness of trace-point names and avoid name-collisions
- Structure of trace-points into “layers” in order to give system insight in a certain level (system/function) e.g. com.<company>.<component>.<layer>.<function>.<…>
- Verbosity levels, e.g. lab vs production

A TRACE USE CASE

- Log Levels

Assigned to static tracepoints, with a verbosity level: 0 less verbose to 14, most verbose

- Wildcards

Enable all events under certain hierarchy level: * for all events, libc_* for all events within libc, etc.

The combination of wildcards and loglevels allow users to gradually enable more specific instrumentation, and increase the verbosity level, as they narrow-down the source of their problem.

- Filtering

Filtering on specific event fields allow use-cases such as following a call-id

MULTICORE TROUBLESHOOTING

- More execution needs to be done concurrently
- Multicore programs introduce notoriously difficult to find bugs!
- Race condition, deadlocks, non-deterministic behavior,
- Program interactions will produce different results from one run to another

Looking at trace events is a proven way to solve those type of bug

SYSTEM-WIDE TRACING

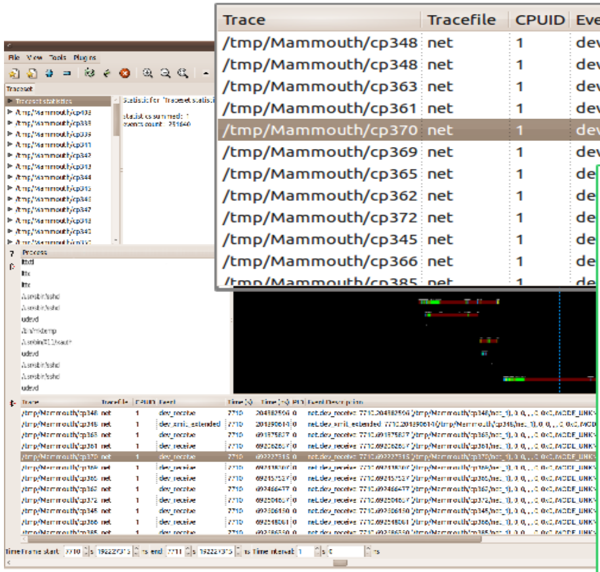
› Correlation of events within a cluster across:

- Nodes,
- CPUs, ASICs, GPUs,
- Hypervisor, Kernel, User-space, Virtual Machines, Libraries, 3PP, Applications.

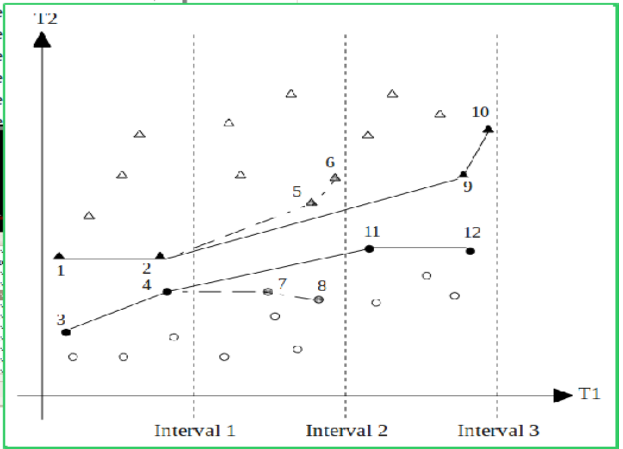
› Based on synchronization information sources:

- Message exchanges,
- Time-stamps,

› Piece-wise algorithms for live cluster traces



Trace	Tracefile	CPUID	Event	Tir
/tmp/Mammouth/cp348	net	1	dev_receive	77
/tmp/Mammouth/cp348	net	1	dev_xmit_extended	77
/tmp/Mammouth/cp363	net	1	dev_receive	77
/tmp/Mammouth/cp361	net	1	dev_receive	77
/tmp/Mammouth/cp370	net	1	dev_receive	77
/tmp/Mammouth/cp369	net	1	dev_receive	77
/tmp/Mammouth/cp365	net	1	de	
/tmp/Mammouth/cp362	net	1	de	
/tmp/Mammouth/cp372	net	1	de	
/tmp/Mammouth/cp345	net	1	de	
/tmp/Mammouth/cp366	net	1	de	
/tmp/Mammouth/cp385	net	1	de	



HETEROGENEOUS TRACE CORRELATION

- Linux on RISC high-level scheduling info: instrumented to collect kernel-level and user space level trace data
- Function call-level detail: HW trace probe collects low-level instruction and data trace on the same RISC processor
- Bare metal DSP events: SW trace instrumentation
- Network “accelerator” hardware block events: HW trace probe

Correlation of traces from different context with IPC between the cores

COMMUNITY TRACING!

- Many layers
  - O&M
  - Application software
  - Middleware
  - Operating system
  - Virtualization

- Developed in different context, i.e. de facto standard needed
  - In house development
  - Consultant
  - Reusable components
  - Third party products

- Many Languages: C/C++, Java, Erlang, etc.
- Cluster wide

LOGGING IS ALREADY EVERYWHERE, WE JUST HAVE TO IMPROVE A LITTLE

IT WILL GIVE THE WHOLE LINUX COMMUNITY MORE RELIABLE SYSTEMS!

ERICSSON