Yet another long-term stable kernel

introduction of Linux Foundation project named "LTSI"

Hisao Munakata

Renesas Solutions Corp.

August 18th 2012, COSCUP2012

Agenda

1. Why we have initiated LTSI project
   - community long-term maintenance
   - why we needed LTSI
2. LTSI creation process
   - patch collection
   - validation and maintenance
3. Why we suggest you to use LTSI
   - value offering
4. Status Update
   - LTSI status @2012-08
   - next step
   - resources
   - conclusion & call for action

who am I

- From embedded SoC provider company Renesas
- Linux Foundation CE working Gr. Steering committee member
  LF/CEWG Architecture Gr. co-chair
- One of LF/CEWG LTSI project initial proposer
- LinuxCon Japan (and others) steering committee
- At my company, I had been encouraging my team developers to send patch to the upstream
- We have sent hundreds of patch to the LTSI 3.0

Question : How did you choose your kernel version ?

1. No care : part of distribution[a]
2. No care : part of application platform[b]

---

3. No option : development started with SoC vendor's BSP

---

4. Selected : selected previously adopted kernel
5. Selected : selected today's latest fresh kernel[c]

---

[a] i.e. Debian/Ubuntu/Fedora/...
[b] i.e. Android/Tizen/Genivi/...
[c] This is always our preference anyway

Vintage kernel is not always matured as it is not a wine

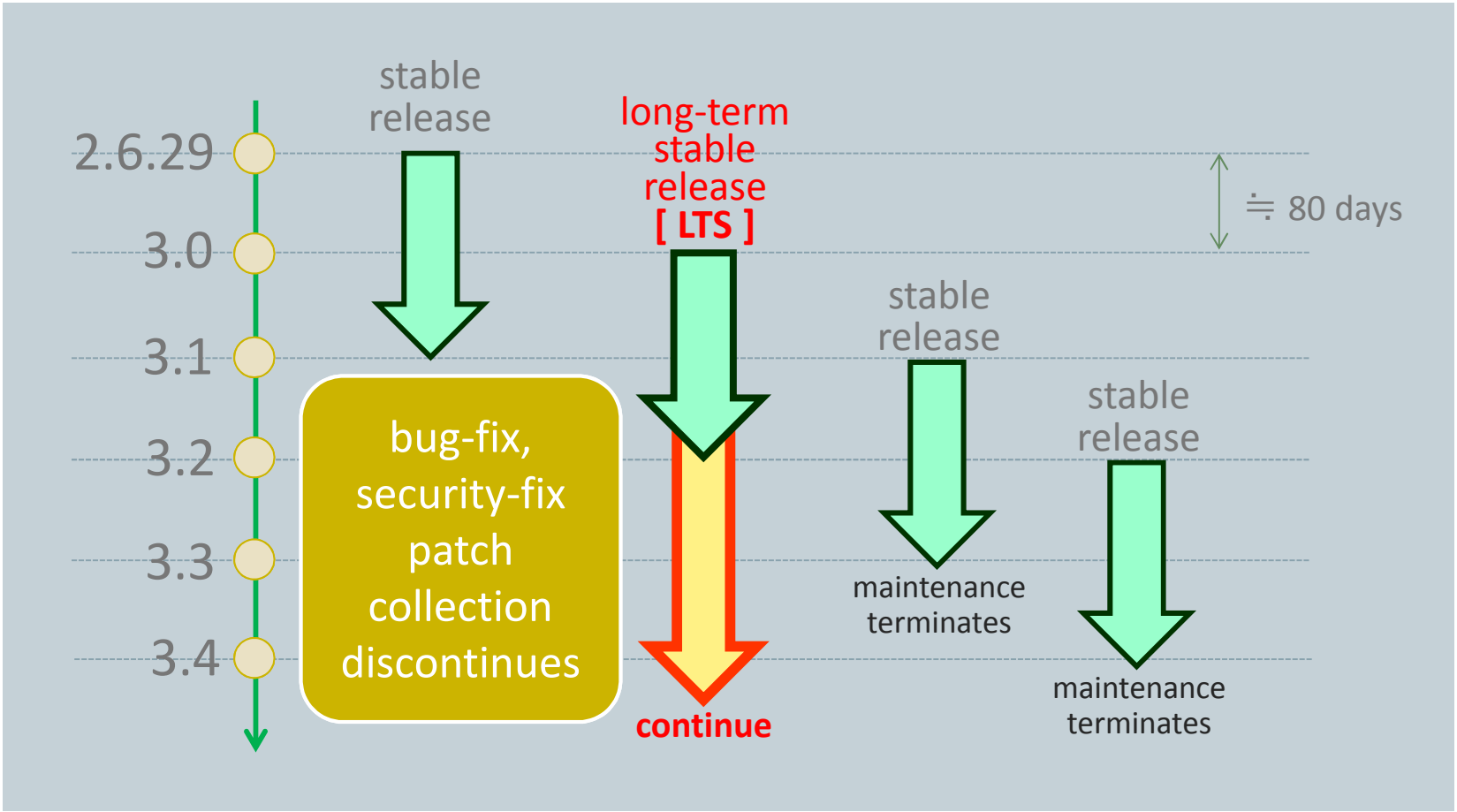
Software lifetime

Software will die when its maintenance becomes discontinued

- modern system software works as a part of connected world.
- application program and data changes (expands) all the time.
- So software need to be maintained throughout its lifetime.
  (Bug-fix, security-fix patch must be collected and applied.)

Some people want to stick on old kernel (like 2.6 series so far), as they think it is enough stabilized. However code maturity is fully depends on how actively that version is kept maintained. In other word, how many patch flow happening against that kernel.

Community maintenance : stable and long-term stable



Upstream kernel maintenance expiration date

Maintenance period = accept bug-fix and/or security-fix patches. Once expired, community will discontinue such patch collection activity. If you want to keep it maintained after expiration, you need to invest someone to continue work on maintenance operation.

regular kernel = roughly 80 days after its release
LTS[a] kernel = normally a few year length

[a] long term stable version kernel

Selecting LTS version kernel should be the best practice to minimize long-term kernel maintenance cost.

LTS kernel version selection criteria

LTS selection rules

- Bug-fix/security-fix patches must be provided to the enterprise server system for 5 years (or longer).
- Then Linux distro requested community to continue patch collection for specific version kernel.
- LTS version = 2.6.16 / 2.6.27 / 2.6.32 / 2.6.34 / 2.6.35...

New stable-kernel rule proposed by Greg Kroah-Hartman

- CE industry demand for LTS is different time-line
- select one version per year for LTS maintenance target
- keep it maintained for 2 year length after its release

OK, LTS seems a good choice, but...

Development speed of consumer product is relatively faster than other industry like enterprise, automotive world. This might cause delay of mainline kernel support catch up even such proposals are already submitted in upstream community.

timing gap challenge for early adoption

- Newly mainlined platform/SoC support is not available on LTS
- Newly mainlined device drivers are still missing on LTS
- Newly mainlined kernel features are still not available on LTS
- LTS kernel might not include device errata fix local patch

Then, we envisioned to create LTSI to solve these embedded industry issues as an extension of community LTS release.

LTS vs LTSI : What differs ?

LF/CEWG LTSI kernel

- **kernel features back-port form latest mainline**
- **device drivers back-port from latest mainline**
- **local patch (=not yet mainlined) integration**

community LTS kernel (is designed to be conservative)

- only accept bug-fix back-port
- only accept security-fix back-port

upstream kernel

- regularly migrated community kernel

Concept of LTSI

- Community LTS + industry demanded extra patches.
- Governed by LF/CEWG
- Focus on kernel code[a], not aiming complete BSP
- CPU architecture and platform neutral
- Can be combined with existing platform[b]
- Comply with upstream rules[c]
- Industry friendly patch collection (flexible patch forms, etc)
- Help CE (and others) industry to utilize Linux

[a] device drivers are part of kernel, of course
[b] Android, Tizen, Yocto, WebOS and others
[c] e.g. signed-off-by process

Who can do what for LTSI

action	membership category			
	CEWG AG voting	CEWG Corporate	CEWG Individual	Other (public)
project web access	yes	yes	yes	yes
code download	yes	yes	yes	yes
ML subscription	yes	yes	yes	yes
attend ICM[1] meeting	yes	yes		
send patch proposal	yes	yes	yes	yes
review patch by ML	yes	yes	yes	yes
discuss patch at meeting	yes	yes	yes	
vote for patch (if needed)	yes			

If you plan to adopt LTSI to your product, we recommend you to join Linux Foundation and become a CEWG/AG member to get full advantage of LTSI project.

[1] industry contact meeting

Which code can be integrated to LTSI and how (1)

from upstream

- newly adopted LTS bug-fix, security-fix (automatic)
- newly mainlined new kernel feature (request basis)
- newly mainlined new kernel driver (request basis)
- newly queued (to -rc version) code (request basis)

from SoC vendor

- curved up from SoC vendor kernel tree (request basis)
- submitted to upstream but still in review (request basis)
- not mainlined chip workaround code (request basis)

Which code can be integrated to LTSI and how (2)

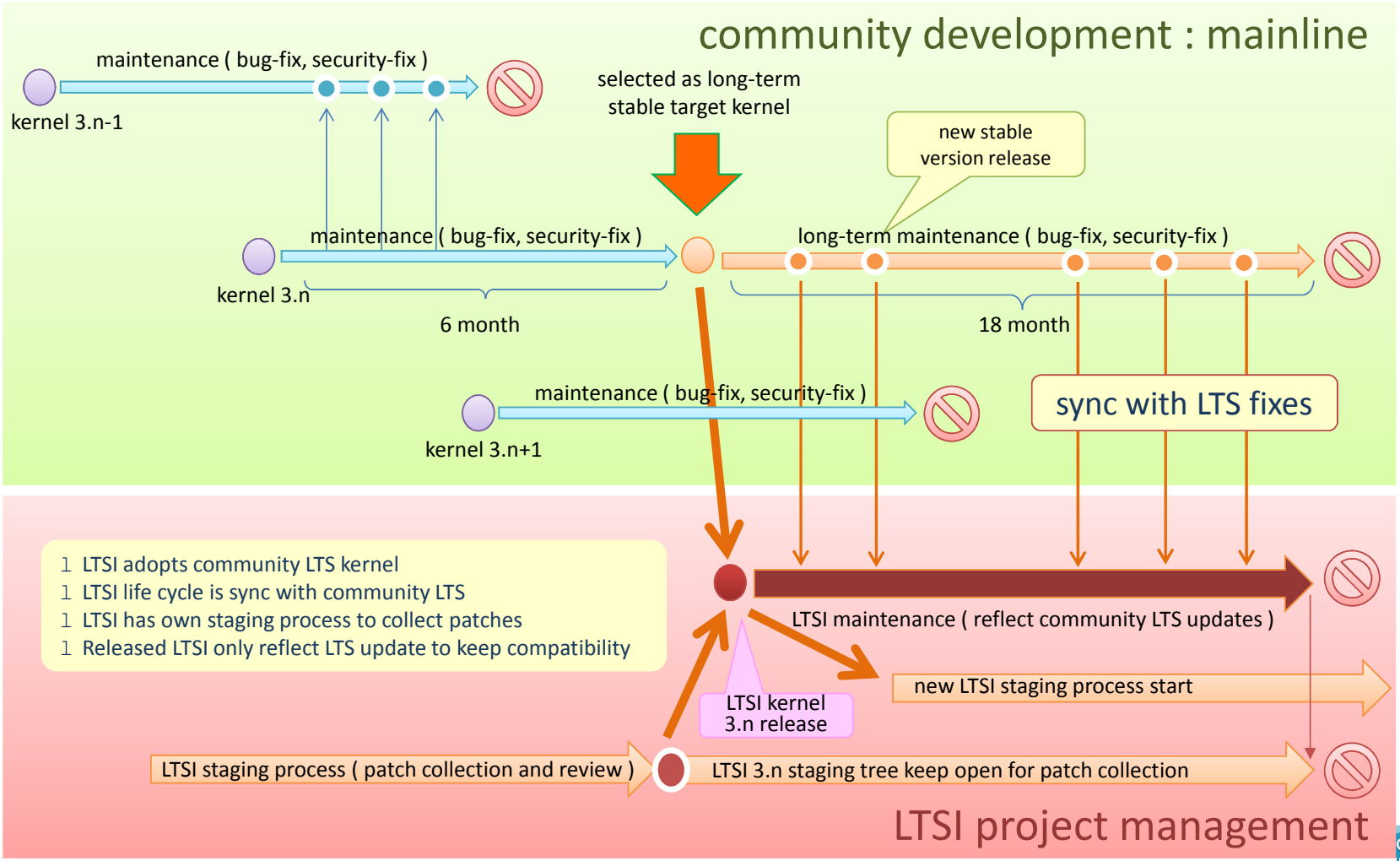
from product producer, distribution

- in-house bug-fix patch (request basis)
- private enhancement (request basis)
  - may require strict compatibility review

Others

- not mainlined open-source project code (request basis)
  - ideally mainline attempt history already exist.
  - LTSI project will help re-attempt action.
- LF/CEWG funded open project result (request basis)

LTSI patch collection process



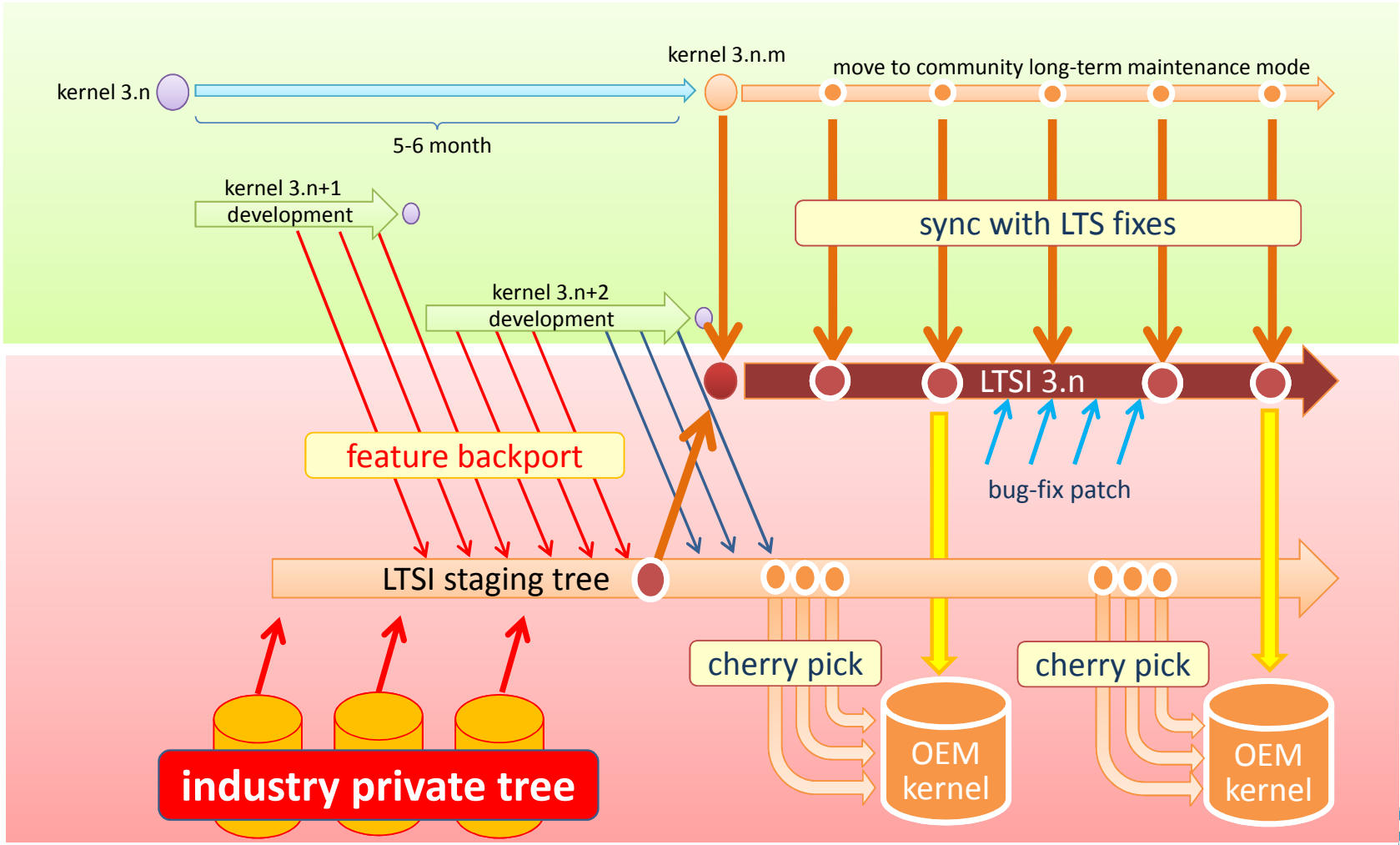
LTSI validation to make actual release

Once patch collection window is closed, LTSI project will provide integrated snapshot for test that includes all approved patches.

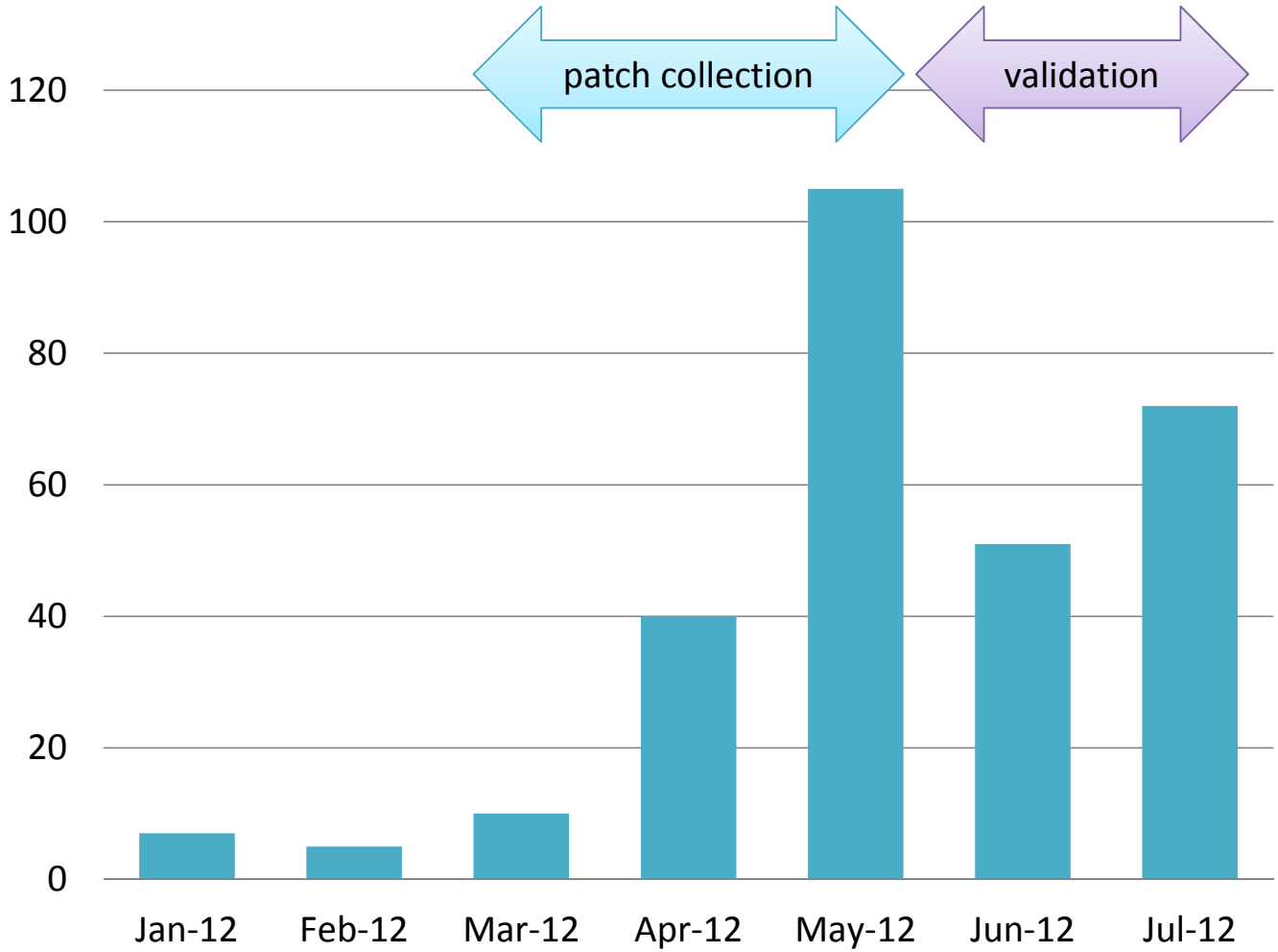
During LTSI-staging period

- Anyone who send a patch should test your code.
- SoC vendor and distribution are expected to validate this snapshot using own reference development environment.
  - LTSI is CPU/platform neutral. You can use your environment.
  - You can publish test result like *"LTSI tested"* or *"LTSI ready"*
  - When you send a feedback, please clarify your environment (kernel config, test target spec ,reproduce procedure,..)
- If you hit any issues (bug, regression,..) please report it to ML. (We did not have dedicated BTS system like Bugzilla so far.)

LTSI adoption process



LTSI public ML posting status (as of end of July)



LTSI 3.0 git tree status

description	LTSI kernel patches
owner	LF Public Git
last change	Tue, 31 Jul 2012 22:19:13 +0000
URL	http://git.linuxfoundation.org/ltsi-kernel.git
	ssh://git-lf@git.linuxfoundation.org/ltsi-kernel.git

shortlog

2012-07-31	Greg Kroah...	series: some unneeded comments removed master v3.0.38-ltsi
2012-07-31	Greg Kroah...	3 patches to fix sh7757lcr board added
2012-07-31	Greg Kroah...	more defconfig patches
2012-07-31	Greg Kroah...	add kzm9g defconfig patches
2012-07-31	Greg Kroah...	added defconfig for armadillo800eva
2012-07-31	Greg Kroah...	Update to 3.0.38
2012-07-09	Greg Kroah...	add lttng fixup patch
2012-07-09	Greg Kroah...	Merge branch 'master' of git.linuxfoundation.org:ltsi...
2012-07-09	Greg Kroah...	add patches.kzm9g/of-address-add-of_find_matching_node_...
2012-07-05	Greg Kroah...	remove xml file from patches.armadillo800eva/0141-v4l...
2012-07-05	Greg Kroah...	Update LTTNG from 2.0.1 to 2.0.4
2012-07-05	Greg Kroah...	Add patches./arm-fix-rcu-stalls-on-smp-platforms.patch
2012-07-05	Greg Kroah...	Some more kzm9g bugfixes
2012-07-05	Greg Kroah...	more bugfix patches
2012-07-05	Greg Kroah...	added patches for r8a66597-udc driver
2012-07-05	Greg Kroah...	Add sh7757lcr backport patches

LTSI 3.0 extended component

We have added over 800 extra industry demanded patches upon community 3.0 LTS-kernel to create LTSI3.0.

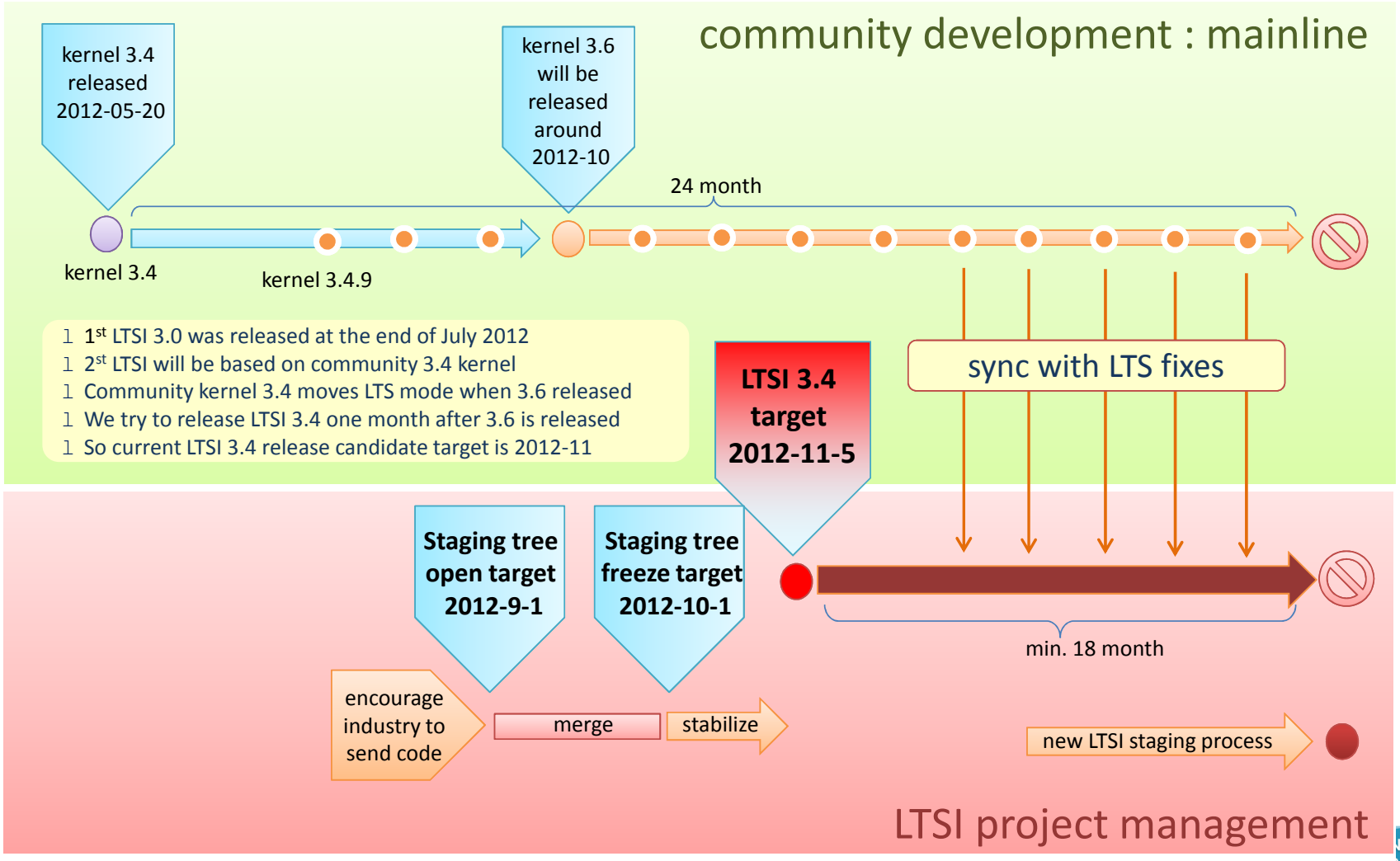
Extra stuff added on top of community 3.0-LTS

- patches.android
- patches.ltsi
- patches.lttng
- patches.pramfs
- patches.runtime_pm
- scripts
- patches.armadillo800eva

Important LTSI milestone

2011.10.26	project launch @ELCE2011
2012.02.01	Greg Kroah-Hartman becomes LTSI chief maintainer
2012.06.08	1st Industry Contact Meeting @Yokohama
2012.06.30	~~LTSI 3.0 code release candidate date~~
2012.07.31	LTSI 3.0 released (w/one month extra test)
2012.08.29	2nd Industry Contact Meeting @San Diego
2012.09.01	LTSI 3.4 staging-tree open
2012.10.01	LTSI 3.4 staging-tree close, start validation
2012.11.05	LTSI 3.4 release candidates @ELCE2012

LTSI 3.4 release schedule (tentative)



LTSI resources

source tree (git)

- release tree = will open at the end of June
- staging tree
  `http://git.linuxfoundation.org/ltsi-kernel.git`
- upstreaming staging tree = will open soon

communication method

- project web = `http://ltsi.linuxfoundation.org/`
- mailing list
  `http://lists.linuxfoundation.org/pipermail/ltsi-dev/`
- social media = `http://twitter.com/#!/LinuxLTSI`

Conclusion

- LF/CEWG launched LTSI project to develop and distribute specially enhanced LTS kernel named LTSI for Embedded industry use. It contains some feature backport from current latest kernel, off-tree kernel patches owned by SoC vendor/product developer and others.
- We believe LTSI can dramatically reduce your own effort (=cost) for in-house kernel management. Also we hope LTSI can encourage CE company developer to send more code to upstream.
- Very first LTSI release 3.0 has released at the end of July 2012. LTSI3.0 will be maintained (at least) until 2013 June timing.
- Community maintainer decided 3.4 is the next LTS. Following this decision we started preparation for next LTSI 3.4 staging process to make LTSI3.4 release targeting ELCE2012@November.

Call for action for LTSI3.4 (targeting 2012-11 release)

For SoC vendor, CPU core provider

- Send your not-yet-mainlined (AKA vendor tree) code to LTSI
- Test LTSI kernel on your environment and feedback test result
- If any software workaround exist, please share that with us.

For product producer

- Adopt LTSI kernel to reduce your in-house maintenance cost
- If you have not-yet-mainlined bug-fix, send it to LTSI

For software distributor, integrator

- Adopt and support LTSI as your base kernel.
- Send us your feedback to improve LTSI and future upstream