How to cook the LTSI kernel with Yocto recipe

improve your productivity with LTSI & Yocto

Hisao Munakata

Linux Foundation Consumer Electronics working group

February 20th 2013, ELC2013

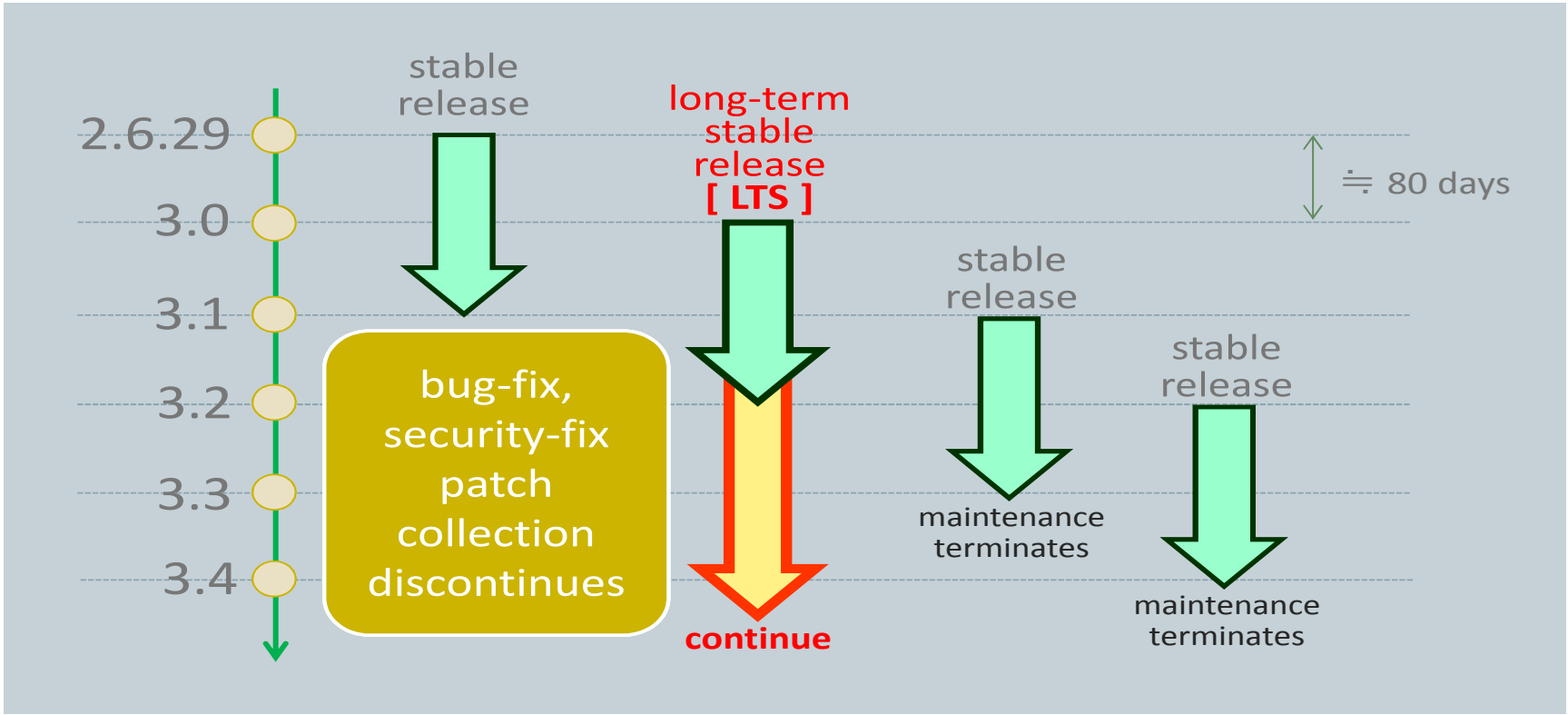
Who am I ?

- From embedded SoC provider company Renesas
- Linux Foundation CE[1] working Gr. Steering committee member, LF/CEWG Architecture Gr. co-chair
- One of LF/CEWG LTSI[2] project initial proposer
- At my company, I had been encouraging my team developers to send a patches upstream
- Also I have supported various CE customers who develop digital-TV, Blu-ray recorder and Smart-phone

[1] CE = consumer electronics
[2] LTSI = Long Term Stable kernel Initiative

Why you should choose LTS & LTSI kernel ?

Linux kernel life-cycle varies according to version



If you choose LTS, you can simply apply serious bug-fix and security-fix patches maintained by the community.

LTS (and LTSI) maintainer, Greg's statement

the 3.4 kernel tree will be -longterm

From: Greg KH
Date: Mon Aug 20 2012 - 18:25:09 EST

- Next message: Andrew Morton: "Re: [PATCH v3 3/9] rbtree: place easiest case first in rb_erase()"
- Previous message: Shirley Ma: "Re: [RFC PATCH 1/1] fair.c: Add/Export find_idlest_perfer_cpu API"
- Messages sorted by: [date] [thread] [subject] [author]

As I'm getting a few questions about this, and I realized that I never sent out an email about this, yes, the 3.4 kernel tree will be the next -longterm kernel that I will be maintaining for at least 2 years.

Currently I'm maintaining the following stable kernel trees for the following amount of time:
3.0 - for at least one more year
3.4 - for at least two years
3.5 - until 3.6.1 is out

Hope this helps clear up any rumors floating around. If anyone has any questions, please let me know.

greg k-h

https://lkml.org/lkml/2012/8/20/675

LTS (long-term stable) kernel rules

Target kernel selection rules

- Maintainer will **choose one LTS version per year**
- **Maintain it for 2 years** from its original release
- Then, we have 2 LTS kernels like 3.0 and 3.4

Patch adoption rules

- Serious security/bug fix small code
- Backport already mainlined code
- No new feature applied to keep 100% compatibility
- See kernel document ``*stable_kernel_rules.txt*'' for detail

We want to use latest device on LTS kernel, but...

3.0-LTS = long-term stable for 2011

- development start = 2011.5.18
- merge window close = 2011.5.29
- **release = 2011.7.21**

3.4-LTS = long-term stable for 2012

- development start = 2012.3.18
- merge window close = 2012.3.31
- **release = 2012.5.20**

There is no chance to mainline new device/platform support to LTS kernel, as its development was done.

LTS vs LTSI : What differs ? Why we wanted that ?

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

Discipline of LTSI project management

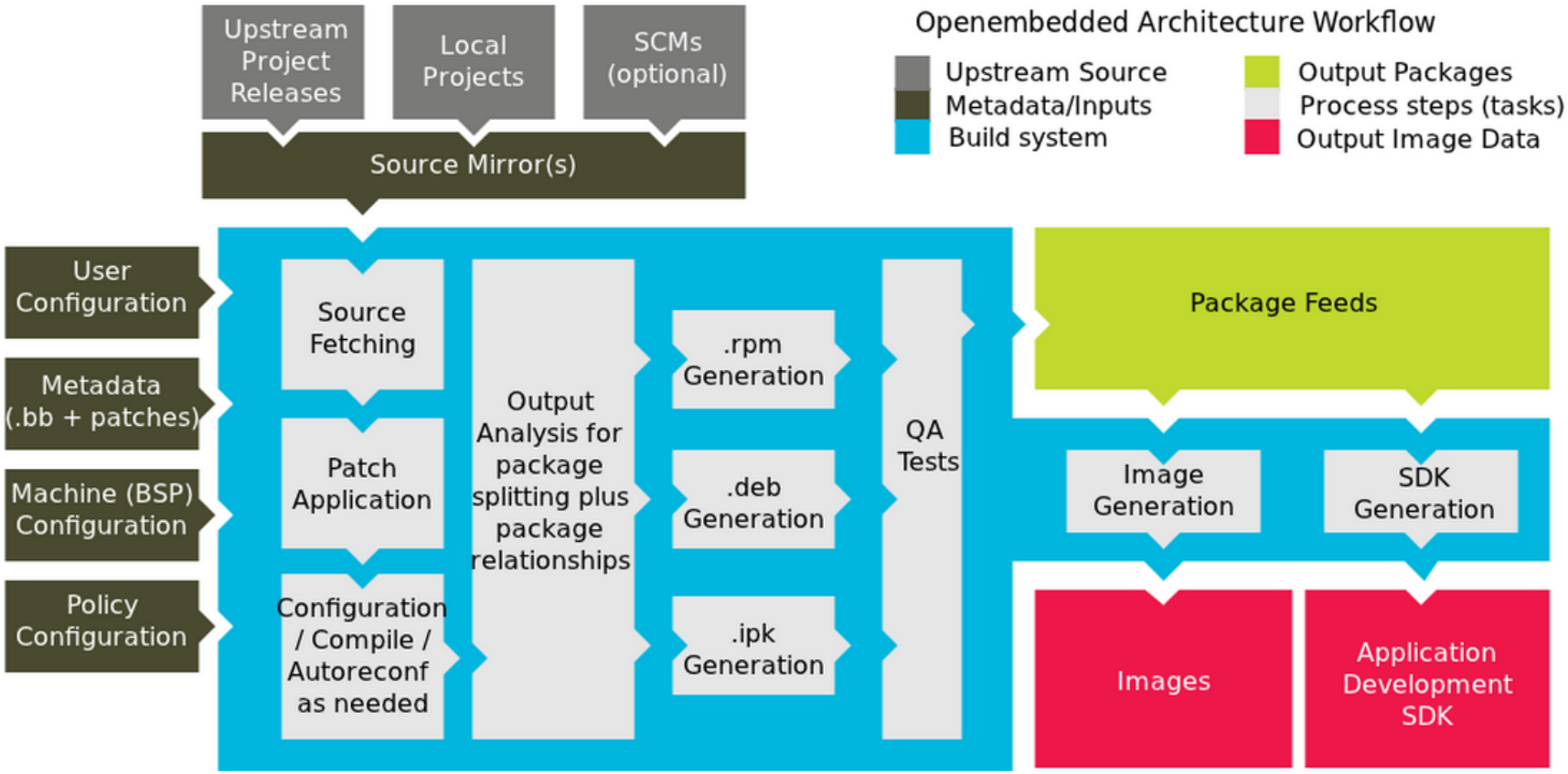
- Community LTS + industry demanded extra patches.
- Governed by LF/CEWG
- Focus on kernel code[a], not aiming complete BSP
- Therefore, can be combined with existing platform[b]
- CPU architecture and platform neutral
- Comply with upstream rules[c]
- Industry friendly acceptance (flexible patch forms,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

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

Yocto project : Yet another LF project for embedded

Introducing the Yocto Project Development Environment

The Yocto Project through the Poky build system provides an open source development environment targeting the ARM, MIPS, PowerPC and x86 architectures for a variety of platforms including x86-64 and emulated ones. You can use components from the Yocto Project to design, develop, build, debug, simulate, and test the complete software stack using Linux, the X Window System, GNOME Mobile-based application frameworks, and Qt frameworks.



Synergy of Yocto + LTSI integration

How LTSI can utilize Yocto infrastructure

- **source code collection (Yocto recipe)**
  - **LTSI kernel**
  - **LTSI off-tree patches**
- build automation
- test automation
- various option for userland
  - tiny-root file system
  - full package system
- https://git.yoctoproject.org/cgit/cgit.cgi/poky/tree/meta

LTSI and Yocto : originally aimed different goal, but..

	LTSI	Yocto
project focus	stable kernel	BSP creation
architecture	neutral	ARM,MIPS,PPC,x86
kernel	LTS	latest
toolchain	not combined	provided
userland	not combined	provided
release cycle	yearly	every 6 month
distribution support	yes	yes
hosted by	Linux Foundation	Linux Foundation

Yocto + LTSI can generate stable BSP for embedded

Yocto and LTSI project coordination is just started

yocto · PROJECT™

ABOUT
ECOSYSTEM
DOWNLOADS
TOOLS + RESOURCES
DOCUMENTATION

Ecosystem
Product Showcase
Yocto Project Participants
Member Organizations
Supporting Organizations
Compliance Program
Compliance Program Registrar

SEARCH embedded linux Go

Long Term Support Initiative (LTSI)

LONG TERM SUPPORT INITIATIVE

LTSI is an industry-wide project created and supported by Hitachi, LG Electronics, NEC, Panasonic, Qualcomm Atheros, Renesas Electronics Corporation, Samsung Electronics, Sony and Toshiba and hosted at The Linux Foundation to maintain a common Linux base for use in a variety of consumer electronics products. The project creates and maintains a long-term industry tree, which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2-3 years.

This new initiative is crucial because device makers are doing significant back-porting, bug testing and driver development on their own, which carries substantial cost in terms of time-to-market, as well as development and engineering effort to maintain those custom kernels. Through collaboration in this initiative, these CE vendors will reduce the duplication of effort currently prevalent in the consumer electronics industry.

The LTSI tree is expected to be a usable base for the majority of embedded systems, as well as the base for ecosystem players (e.g., semiconductor vendors, set-vendors, software component vendors, distributors, and system/application framework providers). The LTSI project will combine the innovative features in newer kernels needed by CE vendors with a stable kernel, while helping those vendors get their code upstream to benefit the entire Linux community. The goal is to reduce the number of private trees currently in use in the CE industry and encourage more collaboration and sharing of development resources.

LTSI-3.4 release notes

LTSI-3.4 development history

item	date
Upstream kernel 3.4 release	2012.5.20
Announce of 2012 LTS kernel version	2012.6.6[3]
LTSI-3.4 merge window open	2012.9.19
3.4 becomes LTS	2012.9.30[4]
(merge window open period)	(78 days)
LTSI-3.4-rc1 (=merge window close)	2012.12.6
LTSI-3.4-rc2	2012.12.17
(validation period)	(35 days)
LTSI-3.4 release	2013.1.10

[3] @LinuxCon Japan
[4] @upstream kernel 3.6 release

LTSI-3.4 active contributors (by patch numbers)

developer	host	patch	technical area
Nicolas Ferre	atmel	246	AT91
Simon Horman	renesas	205	Armaddilo, Marzen,..
Damian Hobson-Garcia	igel	71	dma-mapping
Tetsuyuki Kobayashi	kmc	60	KZM-GT
Greg Kroah-Hartman	LF	62	AF_BUS, others
Marco Stornelli	sony	18	pramfs
Aaditya Kumar	sony	15	axfs
Paul Gortmaker	windriver	9	CoDel

677 patches are added on top of community 3.4 kernel

LTSI-3.4 : What is added on top of regular 3.4 ?

- FIX : refreshed to be based on 3.4.24
- BACKPORT : pramfs now builds properly
- BACKPORT : CODEL support patches added
- BACKPORT : CMA backport from v3.7
- BACKPORT : VFIO backport from v3.7
- BACKPORT : AF_BUS patches
- BACKPORT : LTTng
- NEW : big dma-mapping patches
- NEW : azfs (temporary disabled due to build problems)
- NEW : Board support for Armadillo 800, AT91, kzm9d, kzm9g, and Marzen platforms

LTSI-3.4 highlight from news release

The Contiguous Memory Allocator (CMA)

This is extremely useful for embedded devices that have very limited hardware resources and will better handle the large memory requirements of multimedia applications. CMA originally was merged into the 3.4.0 kernel release, but its functionality was quite limited. Since then, the feature has been significantly improved in the kernel.org releases and those fixes have been added to the LTSI 3.4 kernel release.

AF_BUS

AF_BUS is a kernel-based implementation of the D-Bus protocol. This feature was created for systems that required a faster D-Bus speed than the existing userspace method could provide, specifically the automotive entertainment systems.

CoDel (controlled delay)

CoDel is a transmission algorithm that optimizes TCP/IP network buffer control, is backported for LTSI 3.4. This is a feature used to help control the ``buffer bloat'' problem that has been identified by the networking community as an issue that all devices need to be aware of. This feature was backported from the 3.5.0

platform support

Armadillo 800, AT91, kzm9d, kzm9g, and Marzen platforms to work properly with this release.

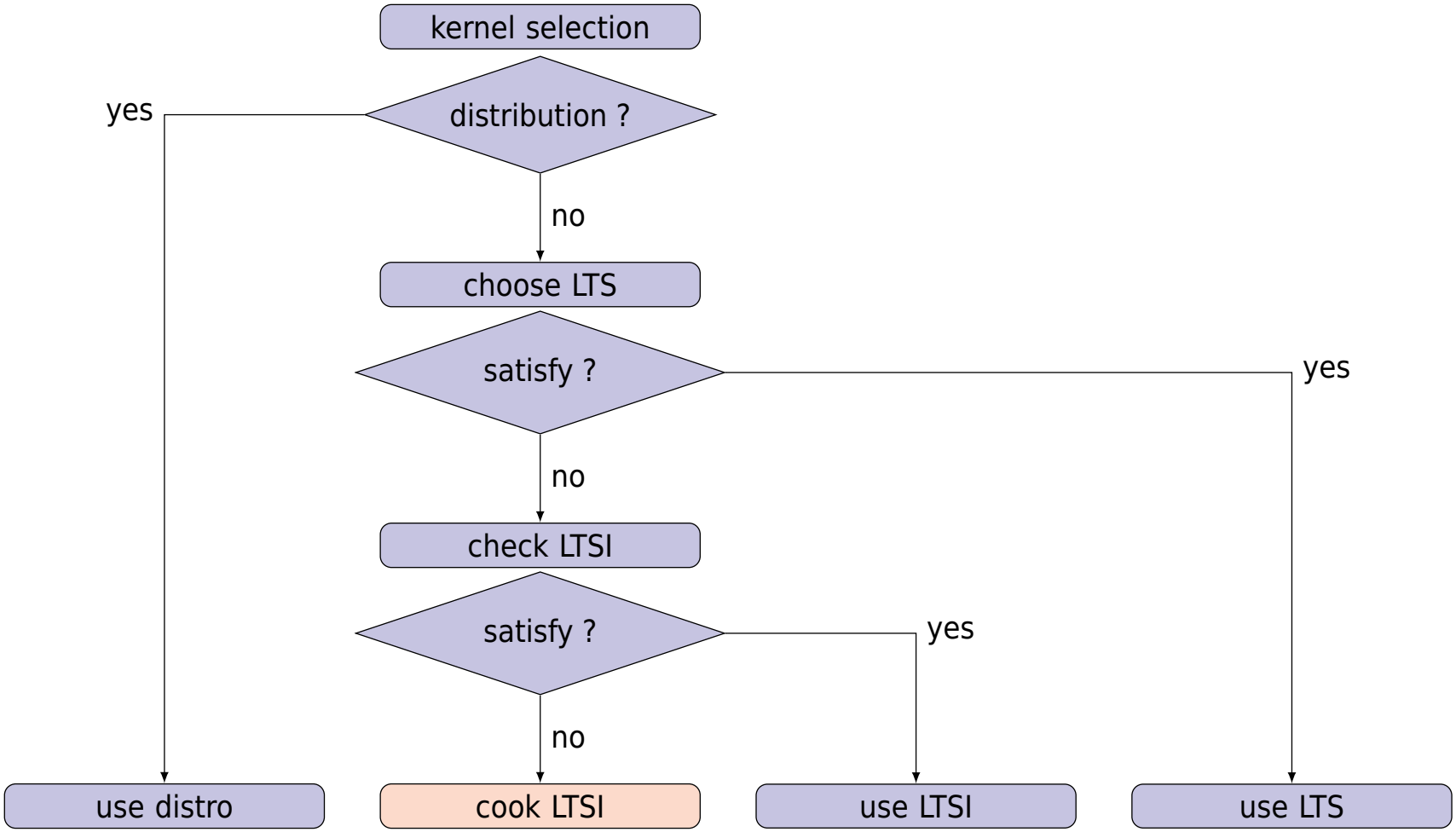
LTSI-3.4 release test by Renesas (pass rate = 99.2%)

board	item	test case (total=137)	/	pass
Armadillo 800EVA	GPIO-KEY	3	/	3
(Cortex A9 single)	Ethernet	6	/	6
	SCIF (serial if)	5	/	5
	touch panel	5	/	5
	LCD controller	2	/	2
	SDHI (SD card)	12	/	11
	MMCIF (MMC)	6	/	6
	FSI (sound)	4	/	2
	CEU (camera)	2	/	2
	USB function	11	/	11
KZM-A9-GT	GPIO-KEY	3	/	3
(Cortex A9 dual)	Ethernet (SMSC LAN911xx)	5	/	5
	SCIF (serial if)	5	/	5
	touch panel	5	/	5
	LCD controller	2	/	2
	SDHI (SD card)	12	/	7
	FSI (sound)	4	/	4
	USB host	25	/	22
	USB function	11	/	11
Marzen	Ethernet (SMSC LAN911xx)	5	/	5
(Cortex A9 quad)	SCIF (serial if)	4	/	1

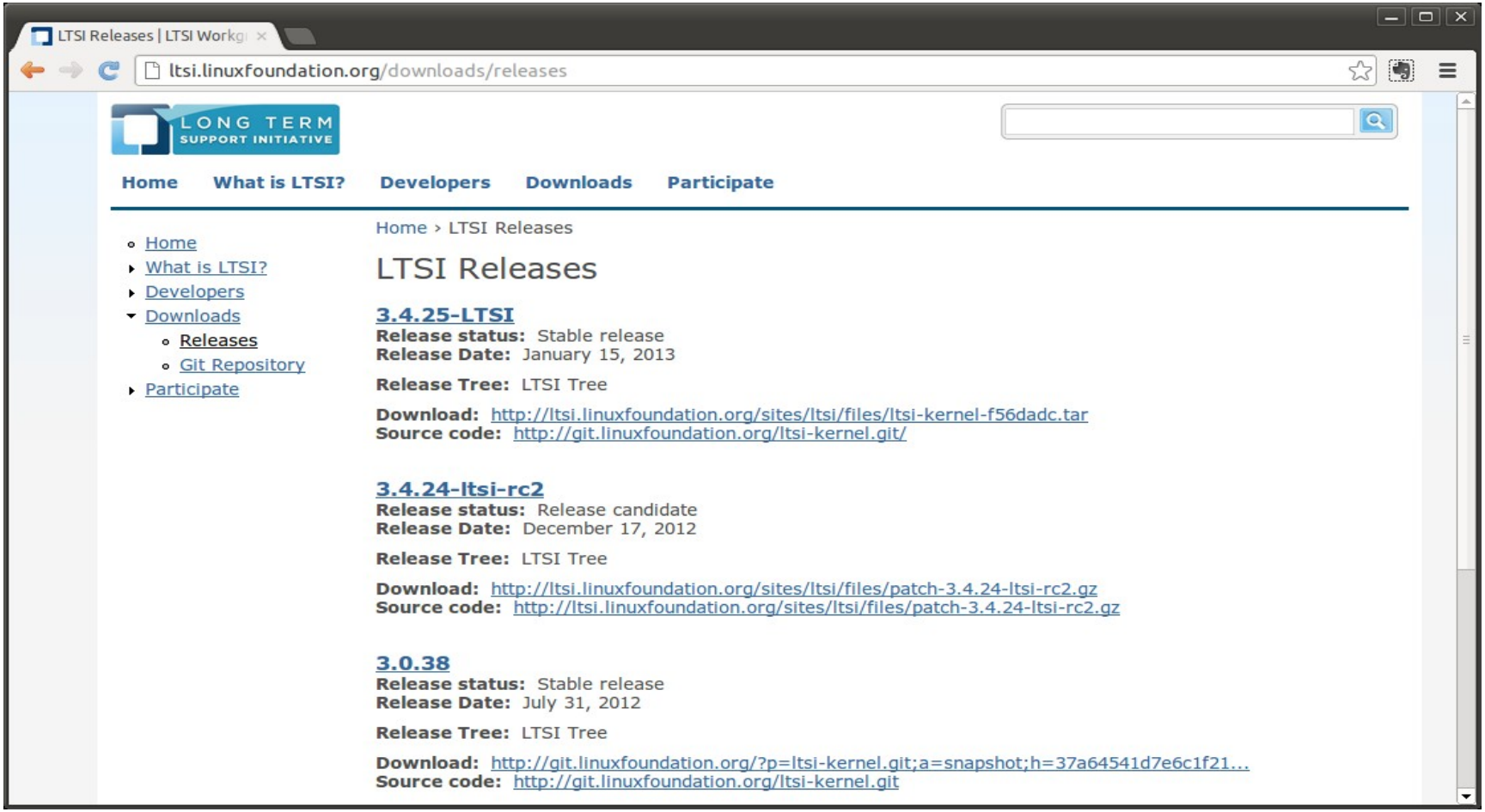
We observed only one degradation from upstream 3.4
Other failures are reproduced also on latest kernel 3.7, not a LTSI problem.

How can you improve productivity with LTSI ?

kernel selection procedure (distro, LTS and LTSI)

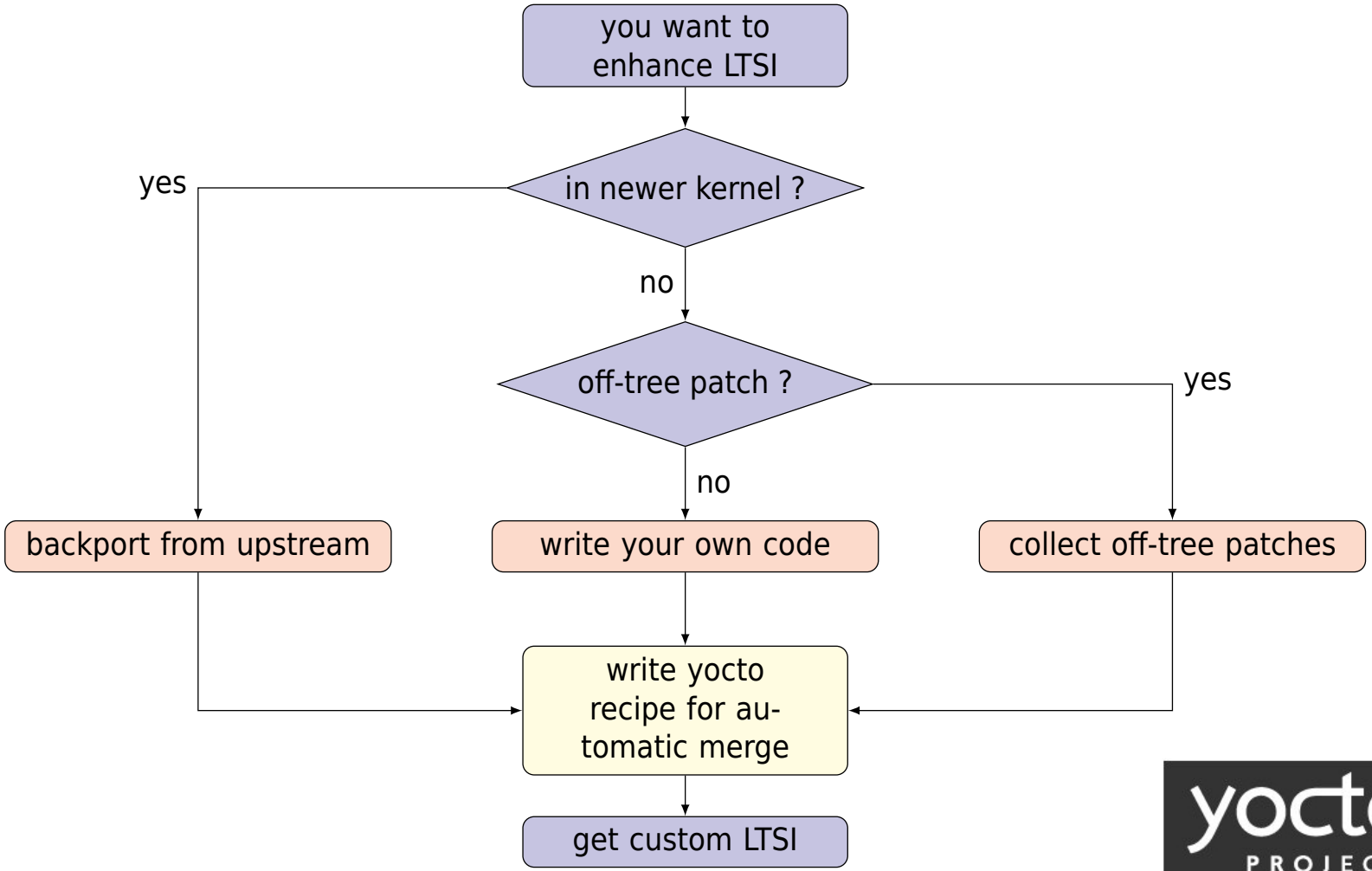


Where can you find the LTSI-3.4 kernel ?



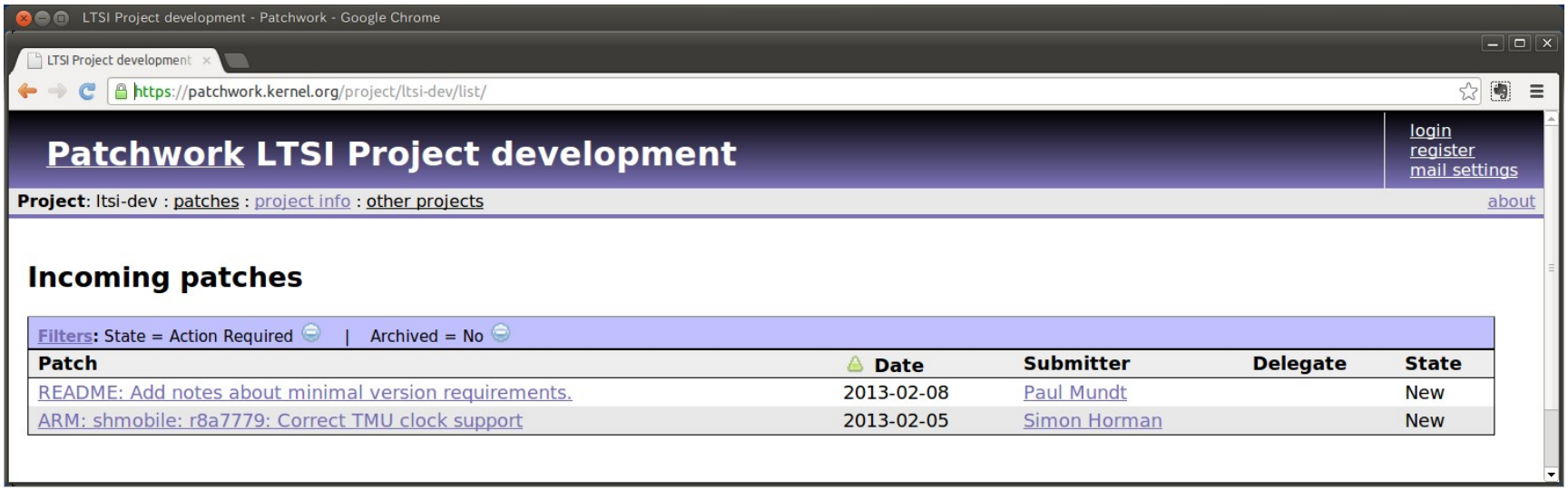
http://ltsi.linuxfoundation.org/downloads/releases

LTSI kernel cooking



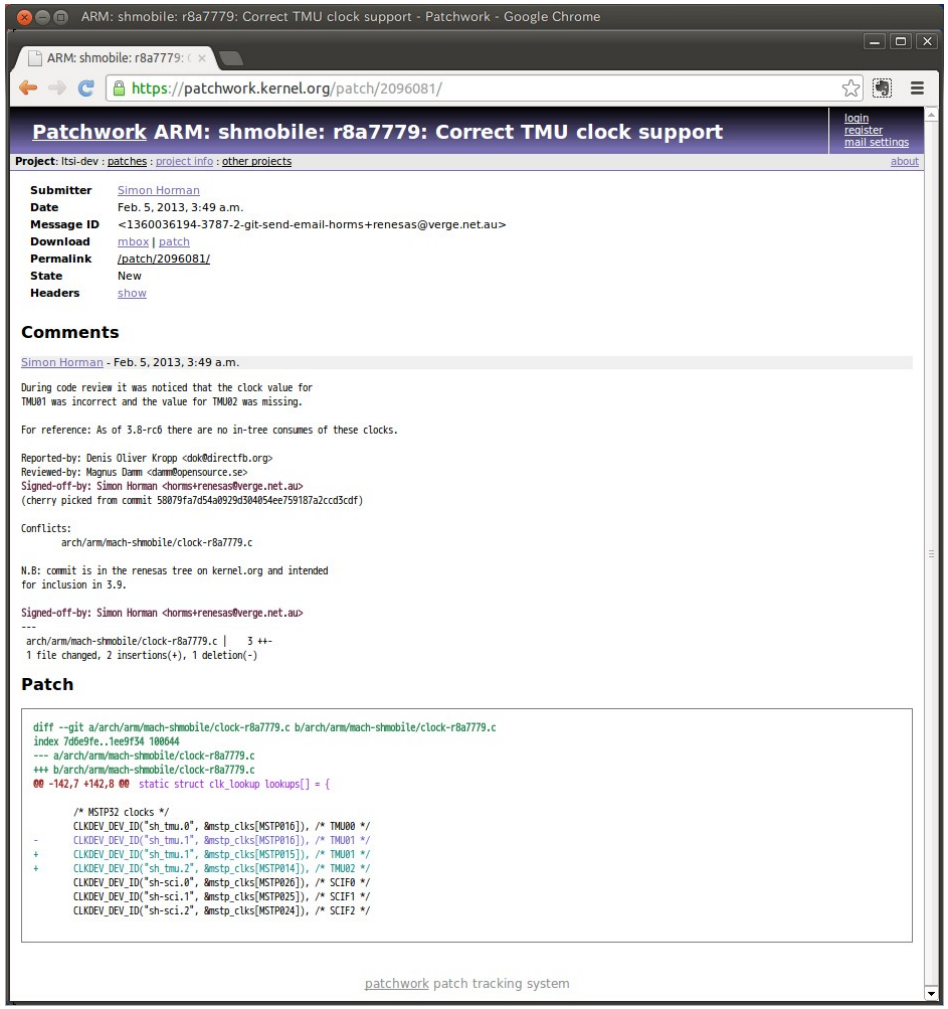
LTSI-patchwork is tracking LTSI-ML incoming message

- You may want to add new platform support to released LTSI.
- Then you sent patch to LTSI-ML, but it might not be merged.
- Patchwork can be the way to collect such off-tree patches.



https://patchwork.kernel.org/project/ltsi-dev/list/

You can cherry pick patch from LTSI-patchwork site



- Patchwork automatically collect message that contains source code (patch)
- Each patch has unique tag and you can identify patch by tag
- You can write yocto recipe to collect patches in patchwork

Yocto meta file contains .bb (recipe) file

```
munakata@mythen:~/Download/meta-renesas-20130204$ tree recipes-kernel/
recipes-kernel/
├── linux
│   ├── files
│   ├── linux-yocto
│   │   └── armadillo800eva
│   │       ├── armadillo800eva-non_hardware.cfg
│   │       ├── armadillo800eva-preempt-rt.scc
│   │       ├── armadillo800eva-standard.scc
│   │       ├── armadillo800eva.cfg
│   │       ├── armadillo800eva.scc
│   │       ├── defconfig
│   │       ├── missing_required.cfg
│   │       ├── required_redefinition.txt
│   │       ├── specified_non_hdw.cfg
│   │       ├── user-config.cfg
│   │       └── user-patches.scc
│   └── linux-yocto_3.4.bbappend
└── linux-libc-headers
    └── linux-libc-headers-rmobile_git.bb
```

.bbappend can contain a pointer to LTSI off-tree patch

Edit recipe to merge LTSI-patchwork off-tree patch

```
diff --git a/recipes-kernel/linux/linux-yocto_3.4.bbappend b/
recipes-kernel/linux/linux-yocto_3.4.bbappend
index 819c65a..0b89004 100644
--- a/recipes-kernel/linux/linux-yocto_3.4.bbappend
+++ b/recipes-kernel/linux/linux-yocto_3.4.bbappend
@@ -19,7 +19,10 @@ SRC_URI_append_armadillo800eva = `` \
file://missing_required.cfg \
file://required_redefinition.txt \
file://specified_non_hdw.cfg \

+ https://patchwork.kernel.org/patch/1132821/mbox/;
name=patch1;
downloadfilename=patch-1132821.patch;
apply=yes;
striplevel=1 \
''
+SRC_URI[patch1.md5sum]    = ``c5e868f90629a56964c2c6ee731ba1cf''
+SRC_URI[patch1.sha256sum] = ``ea5f81ba7b91c0a1086f7c58f92a9818bae46615c5826aacba842c2aac5222

COMPATIBLE_MACHINE_armadillo800eva= ``armadillo800eva''
KBRANCH_DEFAULT_armadillo800eva = ``armadillo800eva''
```

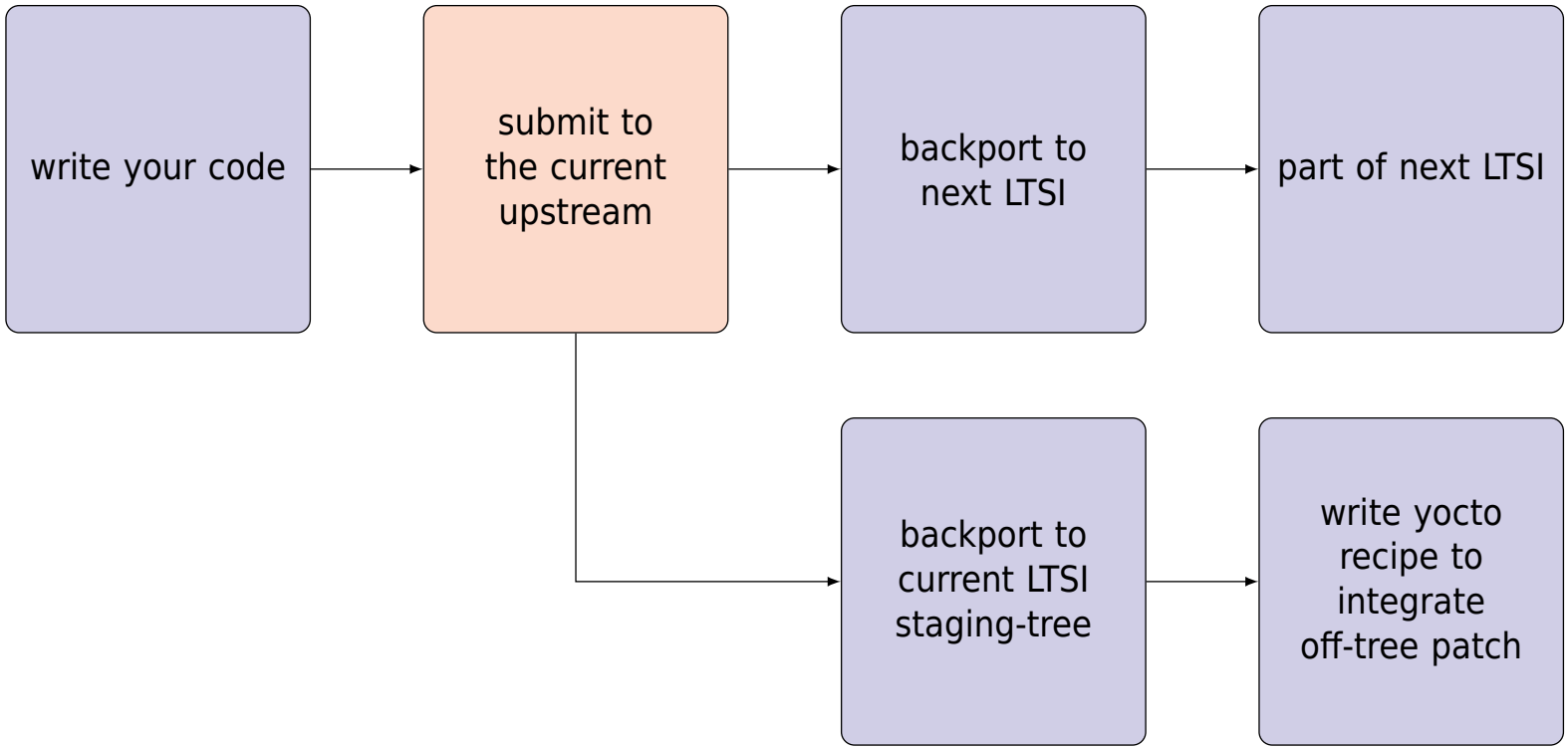
download off-tree patch from patchwork site and apply

Description of patchwork integration recipe

```
+https://patchwork.kernel.org/
            patch/1132821/mbox/;


name=patch1;
downloadfilename=
            patch-1132821.patch;

apply=yes;
striplevel=1 \


+SRC_URI[patch1.md5sum]    =
+SRC_URI[patch1.sha256sum] =
```

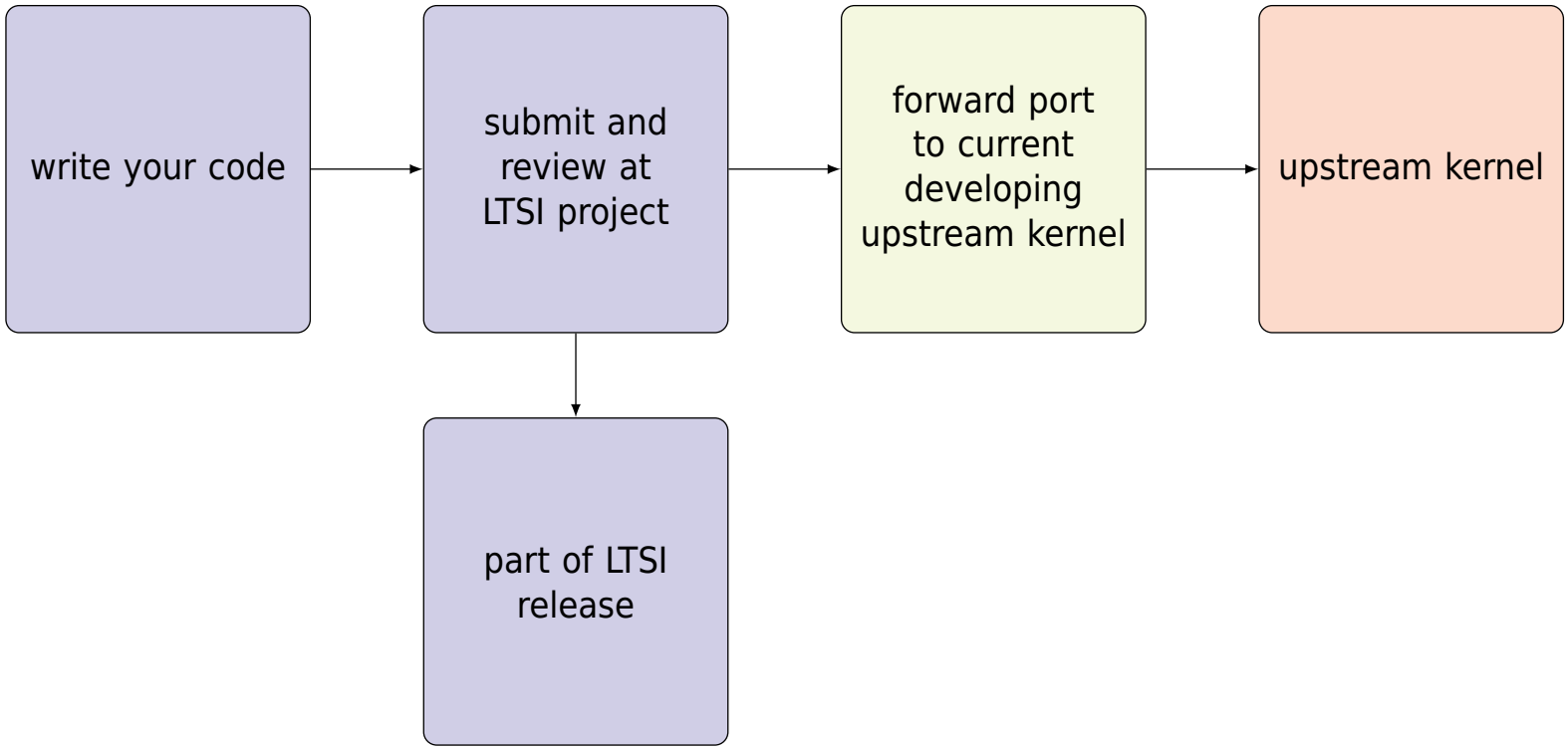
- Define patchwork URI
- You need to define target patch name and assign new name for it, as default download file name is shown as index.html
- You need to calculate SUM after file download (md5 and sha256)

Merge your code into LTSI via upstream (ideal case)



Try upstream first, then backport to LTSI kernel

Upstreaming attempt through LTSI project



LTSI project can help shaping your code for upstream

Conclusion and future action

Conclusion

- If you have a chance to select Linux kernel version, you should choose LTS/LTSI kernel. Because it can reduce your own work to apply security and serious bug-fix patch for maintenance.
- LTSI-3.4 is released now, and it includes various attractive 1)newly mainlined feature from up to 3.7 release, 2)newly developed function for embedded use of Linux, 3)new platform/device support on stable kernel. You can download LTSI-3.4 kernel from Linux Foundation project web.
- If you want to modify LTSI kernel to fit your product demand, you can cook LTSI kernel by yourself and utilize Yocto recipe to integrate your own enhancement with regular LTSI release. You can find off-tree LTSI patch from patchwork web and Yocto recipe can grab them automatically via http connection.

Call for action for LTSI-3.4 (now) & LTSI-next

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

For product producer

- Adopt LTSI kernel with Yocto to reduce your development cost
- Eliminate in-house patch, if any. LTSI patchwork may help.

For software distributor, integrator

- Adopt and support LTSI + Yocto as your BSP foundation.
- Send us your feedback to improve LTSI and future upstream

Need rule for after release patch adoption criteria

Always acceptable patch

- Bug-fix patch for LTSI extended code

Case by case adoption patch

- Add new platform support (self contain stuff only)

Following patches may not be accepted

- New feature backport form new version kernel
- Your own enhancement or local fix
- Out of upstreaming target code

LTSI workshop meeting @ELC2013

LTSI workshop meeting inviting LTSI maintainer

- Feb 21 15:00 - 17:00 @Hearst room, on the 4th floor of the Park55 Hotel
- open to public, anyone can attend this workshop

- Brief Updates of LTSI
- Updates from a partner project: Yocto
- Discussion on after release patch acceptance policy
- Discussion on Super Long Term Support (over 2 years support)
- Discussion on the next LTSI release