DPDK SUMMIT
Powering the Future of Networking Software

12–13 May 2026

# Yelled at by LLMs

## Putting a Megaphone to AI Models in DPDK CI

# Agenda

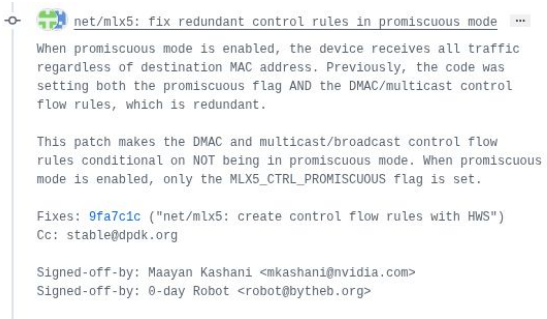
- Why are we even talking about this?
- Timeline of AI Code Review pipeline effort
- Evaluation
- Building the tooling
- Current state
- Future efforts

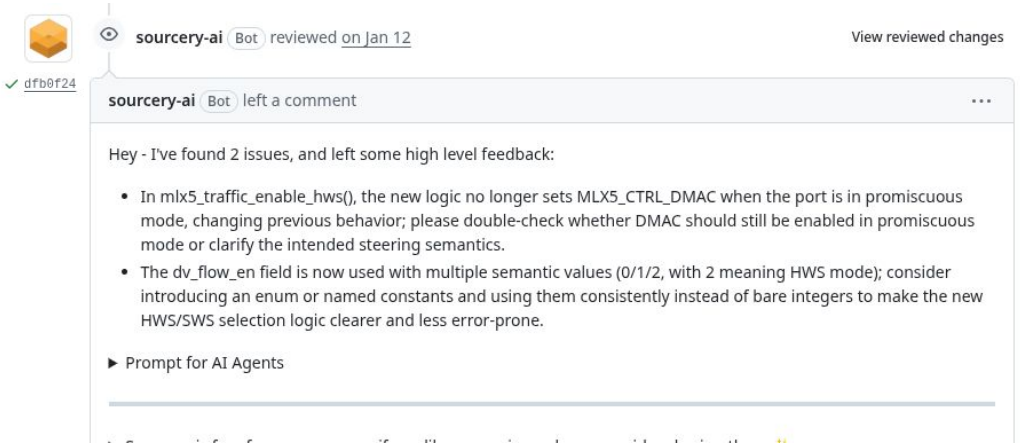
# Why are we even talking about this?

- DPDK large codebase - difficult to do timely review of patches
  - (Over 9k series submissions in 2025)
- High volume of submissions - See 2025 Summit talk
- AI is good at absorbing information and even sometimes generates a useful output
  - Plus there’s no manager to interrupt the workflow, company wide meetings, sick kids, etc.
  - And everyone loves to stamp AI on everything now!
- Can we fit them together?

# Timeline - June 2025, first discussed

- Initial proposal: can we use Sourcery-AI, CodeRabbit, or some other free AI code reviewer on GitHub to inspect code
- Required a bunch of crazy code on the github-robot side, and in the end we got some … interesting results:



# Timeline - Sept 2025, Chris Mason's prompting in Kernel

- Chris Mason develops a prompt suite for Kernel
- Built around a series of hardened prompts for guiding LLM reviews
- As of today used by the Sashiko project
  prompts have been adapted to Gemini, Opus, and Sonnet
- See https://github.com/masoncl/review-prompts for more details on those prompts

# Timeline + Building tooling - Oct 2025 - Jan 2026

- Stephen Hemminger and I work on some scripts.
- See https://mail.openvswitch.org/pipermail/ovs-dev/2025-November/427871.html for the first ‘stab’
- Stephen goes even more in depth and proposes an initial AGENTS.md file
  - AGENTS.md is a ‘standard’ being adopted by lots of different environments - see XKCD 927

Commit Message

Stephen Hemminger Jan. 9, 2026, 1:41 a.m. UTC

```
Add a structured reference document that enables AI code review tools
to validate DPDK contributions against project standards. This document
consolidates requirements from multiple sources into a machine-readable
format optimized for automated validation workflows.

The AGENTS.md file synthesizes guidelines from:
- DPDK Contributing Code documentation (patches.rst)
- DPDK Coding Style guidelines (coding_style.rst)
- Linux kernel patch submission process (submitting-patches.rst)
- SPDX License Identifier specification (spdx.org)

Key sections include:
- SPDX license identifier requirements
- Commit message format and tag ordering
- C coding style rules and naming conventions
- Patch validation checklists with severity levels
- Meson build file style requirements

The document provides actionable checklists and concrete examples to
support integration with CI/CD pipelines and automated review systems.
Severity levels (error/warning/info) help tools prioritize feedback
appropriately.

This supports the broader goal of maintaining code quality and
consistency as the project scales, while reducing manual review burden
for maintainers on mechanical style issues.

References:
- https://doc.dpdk.org/guides/contributing/patches.html
- https://doc.dpdk.org/guides/contributing/coding_style.html
- https://www.kernel.org/doc/html/latest/process/submitting-patches.html
- https://spdx.org/licenses/

Signed-off-by: Stephen Hemminger <stephen@networkplumber.org>
---
 AGENTS.md | 410 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 410 insertions(+)
 create mode 100644 AGENTS.md
```

# AGENTS.md in DPDK

PHASE 1 — GENERATION FROM SOURCE INPUTS


## AGENTS.md

Machine-readable review policy for DPDK

```
coding_style: linux-kernel
commit_format: subsystem: description
checks: [checkpatch, abi, meson]
review_scope: style, logic, docs, tests
norms: mailing-list precedents
abi_policy: preserve unless versioned
test_required: unit + functional
```

# AGENTS.md - Overview

- Sections detail what the LLM should be reviewing
- Statements should be for things that we can't already script away (don't worry about things caught by checkpatch, etc)
- Large part of the context window for evaluation, so keep it as terse as possible (future optimization - caveman skill?)

# analyze-patch.py - Use AGENTS.md to evaluate

- Currently: Creates system/user prompts and feeds to various LLMs via openai/anthropic APIs
- Can split patches for keeping context windows within limits
- Currently - no toolcall support (forthcoming, but costs $ because each one is an extra turnaround)

```
10:37:31 aconole@RHTRH0061144 {(be73c67afb...)} ~/git/ovs/utilities$ ./analyze-patch.py~ --help
usage: analyze-patch.py~ [-h] [-p {anthropic,openai,xai,google}] [-a AGENTS] [-m MODEL] [-t TOKENS] [-v] [-f {text,markdown,html,json}] [-o FILE] [-l]
                         [--timeout SECONDS] [-D YYYY-MM-DD] [--large-file MODE] [--max-tokens N] [--split-patches] [--patch-range N-M] [--send-email]
                         [--to ADDRESS] [--cc ADDRESS] [--from ADDRESS] [--dry-run]
                         [patch_file]

Analyze DPDK patches using AI providers

positional arguments:
  patch_file            Patch file to analyze

options:
  -h, --help            show this help message and exit
  -p, --provider {anthropic,openai,xai,google}
                        AI provider (default: anthropic)
```

# Timeline - March / April 2026 - Signs of life

AI Reviews start coming

Developers beware!

| | | |
|---|---|---|
| ci/aws-unit-testing | success | Unit Testing PASS |
| ci/iol-broadcom-Performance | success | Performance Testing PASS |
| ci/iol-intel-Functional | success | Functional Testing PASS |
| ci/iol-unit-arm64-testing | success | Testing PASS |
| ci/ai-code-review | success | AI Code Review |
| ci/iol-unit-amd64-testing | success | Testing PASS |
| ci/iol-abi-testing | success | Testing PASS |
| ci/iol-compile-amd64-testing | success | Testing PASS |

# Timeline - March / April 2026 - Signs of life

```
> +       "
> +        * residence_time = tx_ts - rx_ts
> +        *
> +        * Remaining untracked delays:
> +        *   - Pre-RX:  NIC DMA - rx_burst return  (~1-5 us, unavoidable)
> +        *   - Post-TX:  tx_ts - kernel TAP write   (~1-2 us)
> +        * Both are symmetric for Sync and Delay_Req so they largely
> +        * cancel in the ptp4l offset calculation.
> +        */
> +      uint64_t tx_ts = sw_timestamp_ns();
> +      int64_t residence_ns = (int64_t)(tx_ts - rx_ts);
```

Does this code handle the case where rx_ts > tx_ts? If the system clock is adjusted backward

# Different models = different results

Using FIB fix tbl8 *https://patchwork.dpdk.org/project/dpdk/list/?series=38075*

| **Tool** | True positives | False positives | Walked the code? |
|---|---|---|---|
| *Claude Web Opus 4.7* | 1 big | 0 | Yes with trace |
| *Claude CLI* | 0 | 1 (mild) | Partial |
| *Analyze-patch (Sonnet)* | 1 minor | ~5 | No |
| *Analyze-patch (Grok)* | 0 | ~5 | No |
| *Gemini Web* | 0 | 4 | No |
| *Analyze-patch (Gemini)* | 0 | 0 | No |

# Policy - Don’t DoS our wallet!

- Please run review locally on your own before submitting
- Code reviews don’t run on every patch!
- Series must pass most of the ‘fast-tests’ unit test suite
- Series must be on the list for >24h and be in a ‘good’ state
  - NEW, REVIEW IN PROGRESS, etc.
- Series must not modify AGENTS.md file
  - Protect against malicious changes to the AGENTS file - that requires manual evaluation.
- Maintainers will have a way to manually request review (similar to recheck-request)

# UNH IOL - The gatekeepers to the API

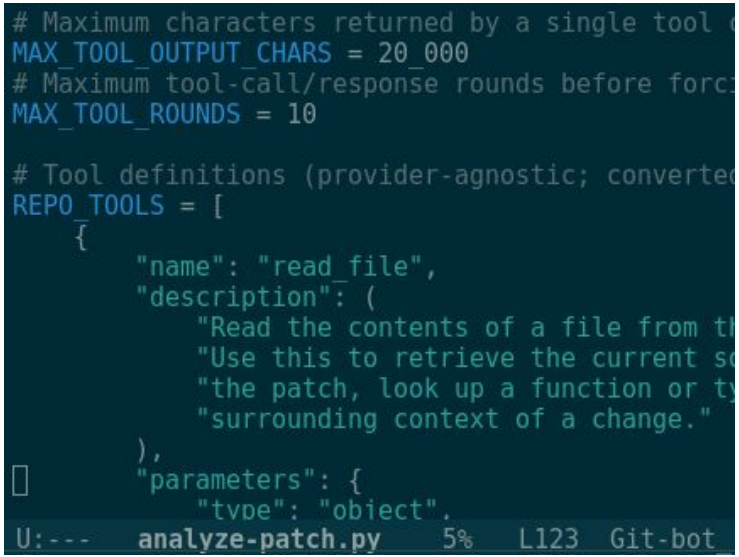
- We have UNH running the scripts - currently using pre-accepted versions of framework
- They will switch once the patches are merged in-tree (requires a bit more evaluation time)

# “Future” - immediate future work

- Get enough confidence to merge the AI review framework from Stephen.
- Update the prompts to have the AI assess whether to mark a review as WARN, or SUCCESS.  FAILURE should indicate that the AI infrastructure is not working for some reason
- Community feedback on discussing the AI review (should it go to dev mailing list, or should that stay with humans only).

# Future work - tool calling

- Add support for grep, and file read
  - Initial testing shows even better review quality
  - Major consideration is Cost
  - Each tool call will eat some more tokens, and that might impact the budget
  - BUT it may be worth it for better results
- Have code for it already




DPDK
SUMMIT
Powering the Future of Networking Software