

DEPT (DEPendency Tracker)

A Path to Mainline and Solving the False Positive Challenge

Linux Plumbers Conference 2026

Byungchul Park @ SK hynix

max.byungchul.park@sk.com | max.byungchul.park@gmail.com

Lockdep's Boundary

Tracks strict lock acquisition order

Shuts down on first
deadlock detection

Fails on cross-context synchronization

The Blind Spot (Untracked Dependencies)

Folio Locks:

Not released within original
context (e.g., folio_lock)

Cross-Context:

`wait_for_completion()` / `complete()`
DMA fences, `swait`, general waitqueues

**[FATAL] Deadlocks masked by complex wait/event patterns
remain completely invisible to mainline tools.**

DEPT shifts the paradigm from lock acquisition to wait-event topologies

The Lockdep
Question:

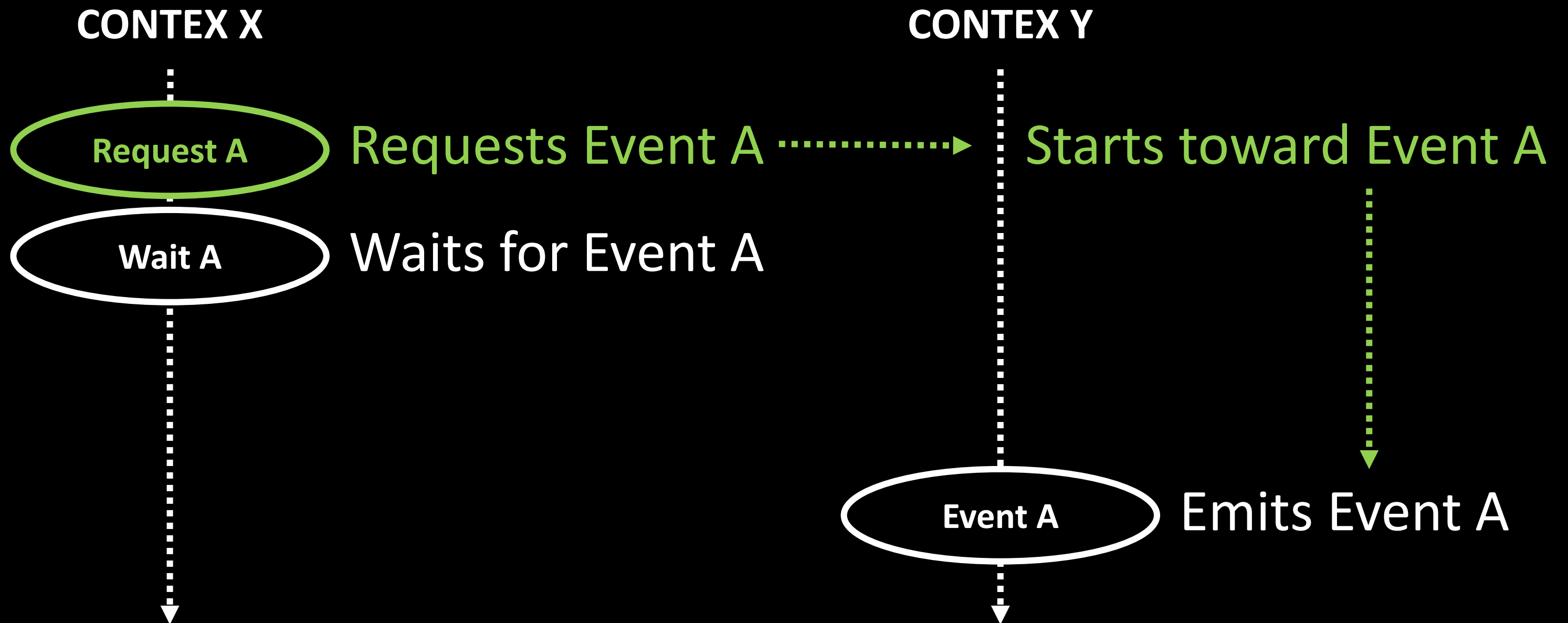
In what order were
these locks acquired?

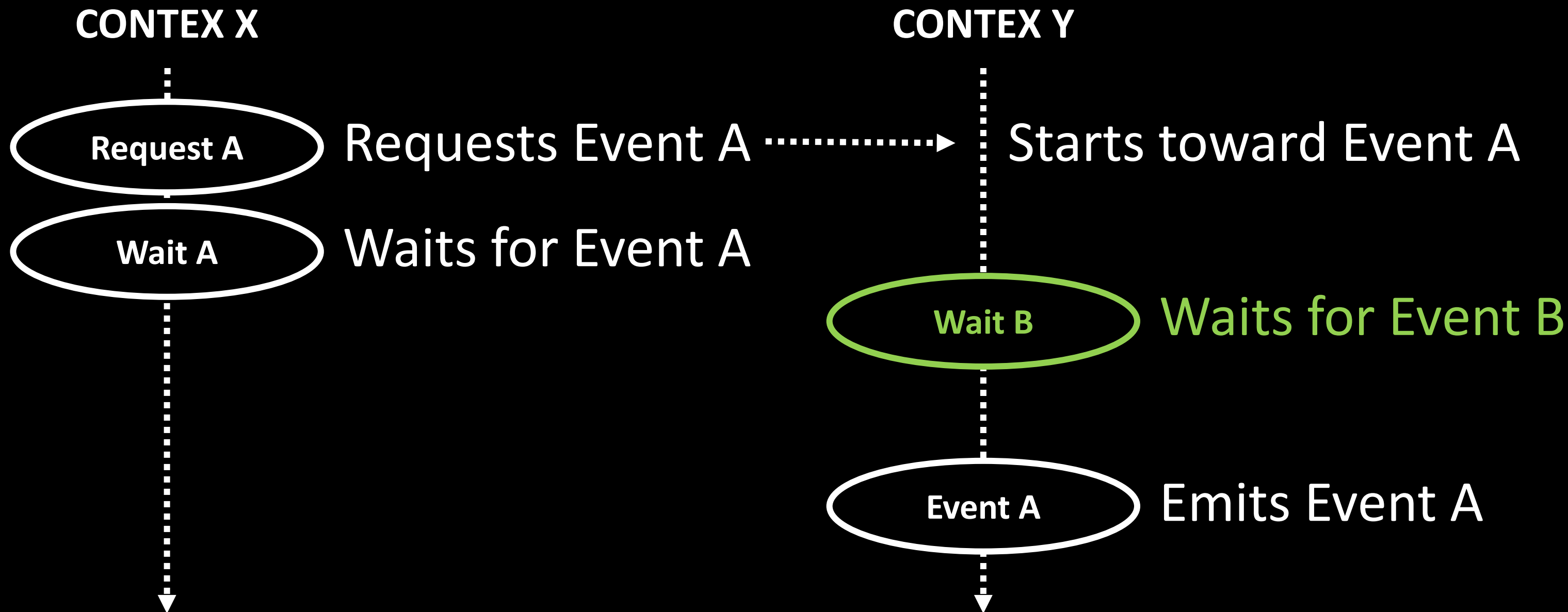


The DEPT
Question:

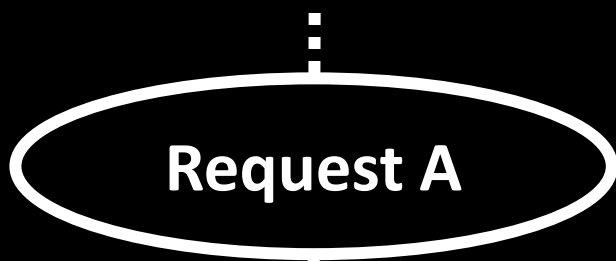
What is this context
waiting for, and what
event will wake it up?

Every deadlock is fundamentally about unreachable events.





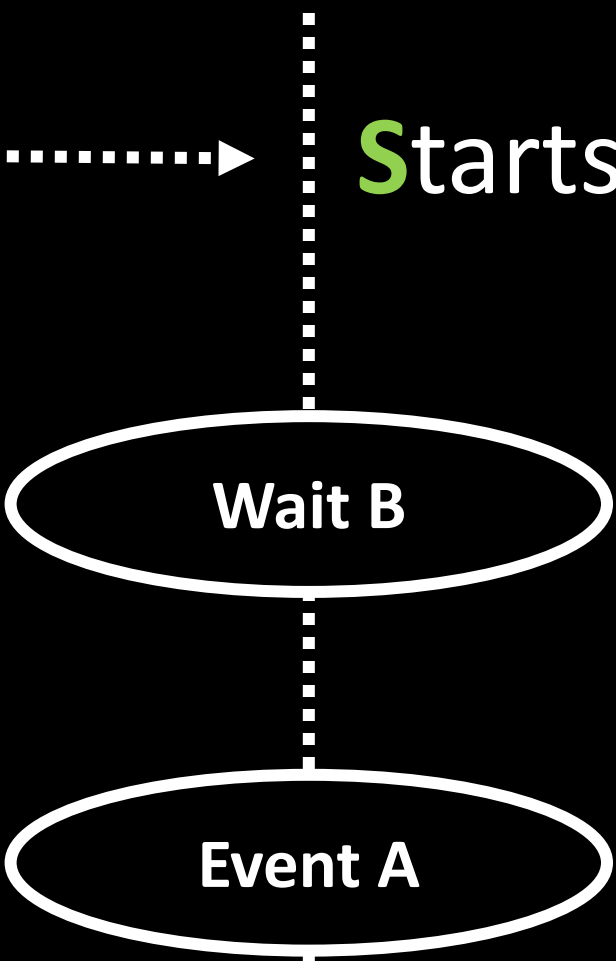
CONTEX X



Requests Event A

Waits for Event A

CONTEX Y



Starts toward Event A

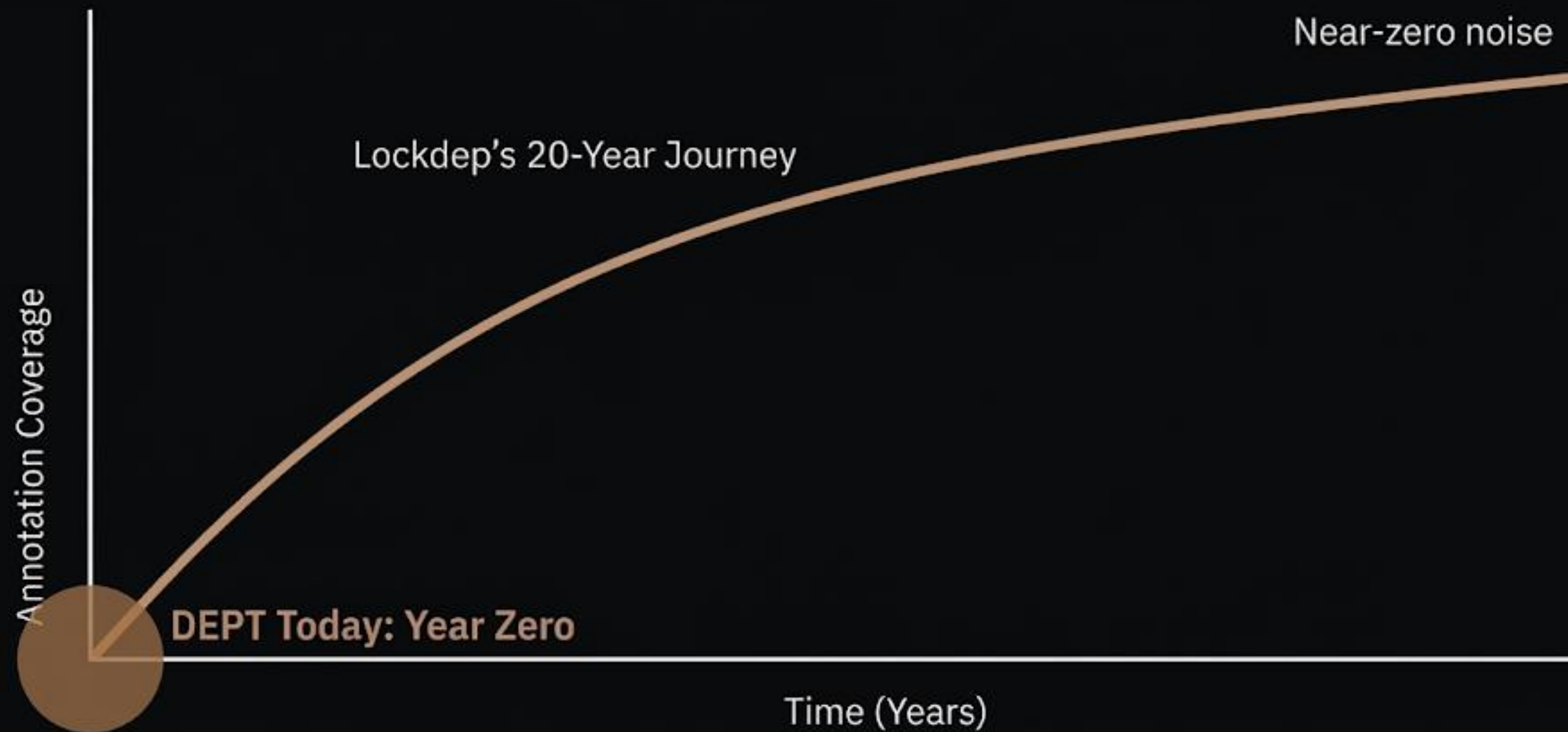
Waits for Event B

Emits Event A

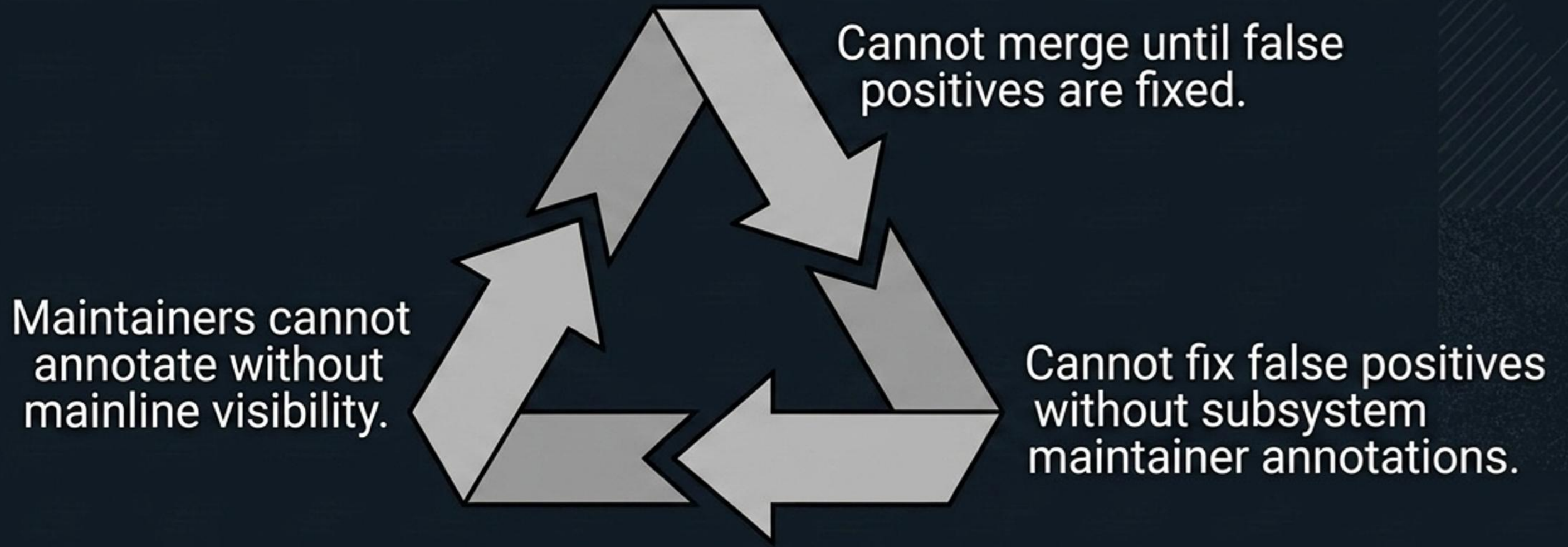
The Start, Wait, Event (S-W-E) maps the context lifecycle.



False positives are a historical phase of **maturity**, not a fatal flaw.



Out-of-tree annotation development is fundamentally unmaintainable.



**DEPT needs mainline exposure for
real-world testing across configurations.**

We propose merging DEPT as an EXPERIMENTAL, opt-in configuration.

CONFIG_DEPT (EXPERIMENTAL)

The Path Forward:

Disabled by default:

Will not affect standard kernel builds or CI pipelines.

Opt-in for debugging:

Available immediately for hunting deadlocks by cross-context synchronization.

Gradual maturation:

Provides mainline foundation to begin subsystem annotation.

The background features a dark blue-grey color with a subtle pattern of light grey circuit lines. In the center, there are three concentric circles: an inner solid orange circle, a middle grey ring, and an outer grey ring.

**Let's build the next 20 years of
deadlock detection together.**

Enable CONFIG_DEPT
and let's map the unseen web of kernel dependencies.

```
root@kernel:~# dmesg
```

Dissecting [S], [W], and [E]

Loading Dependency Tracker graph...

[S] START_CONTEXT

> Definition:

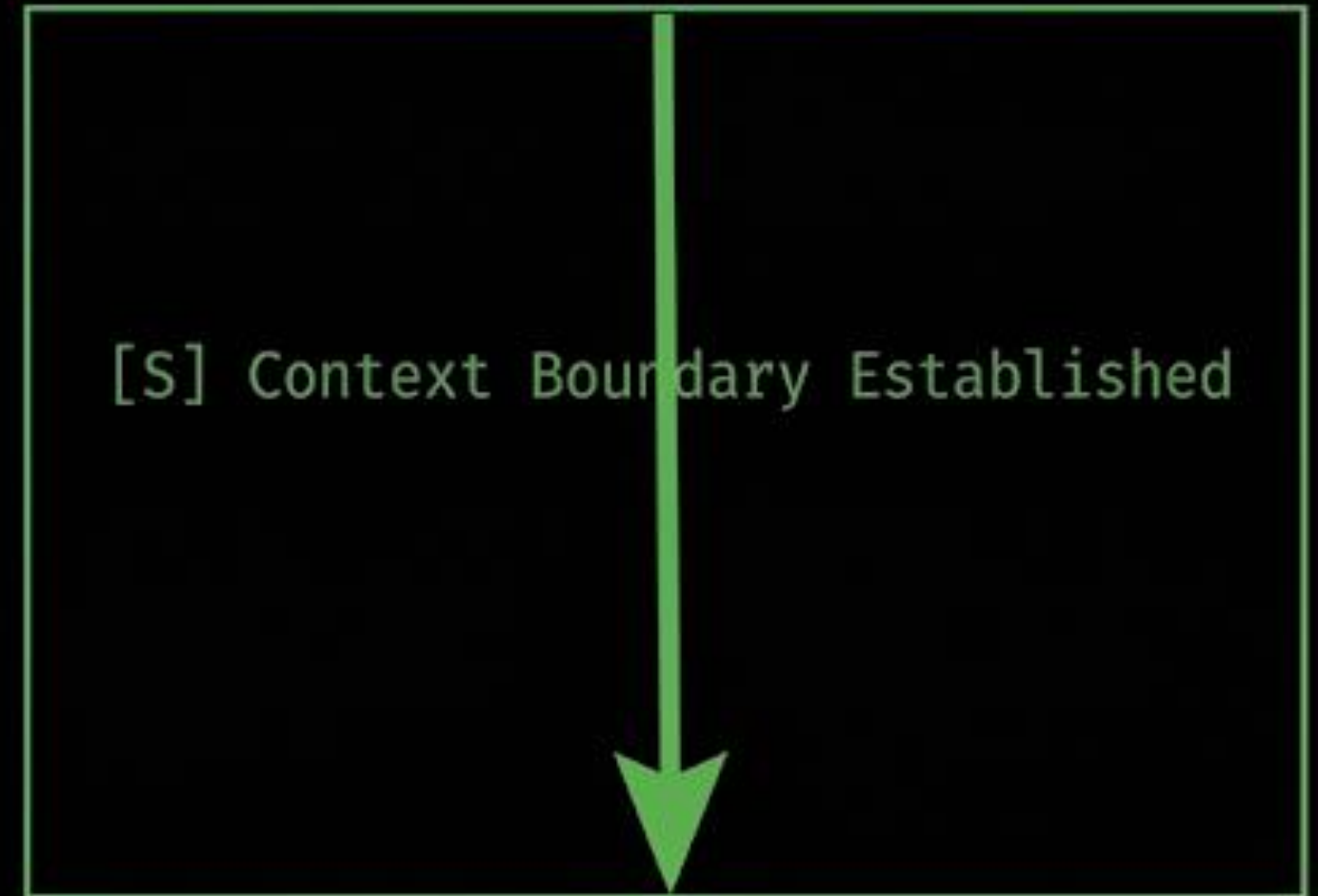
The entry point of an execution path (context) responsible for triggering event [E].

> Role:

Bounds the event context.

> Golden Rule:

Waits inside [S] must NOT block the event [E] this context should produce.



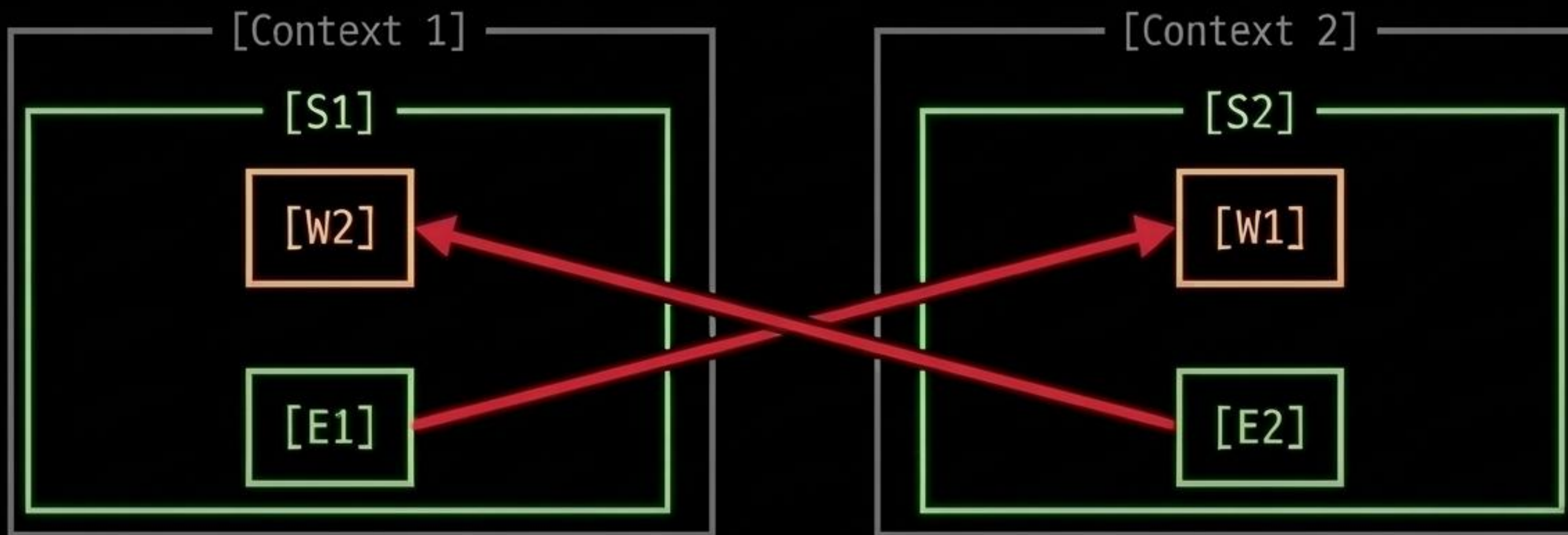
[W] WAIT

> **Definition:** The blocking point where an execution thread suspends itself, preventing the event from being triggered.

Classic Lockdep	DEPT [W] Markers
- Spinlocks - Mutexes - RW Sems	- Sleep points / waitqueues - Folio lock waits (PG_locked) - wait_for_completion - DMA fences & swait

[E] EVENT

- > **Definition:** The actual signal or action (unlock, complete, wake_up) resolving its wait.
- > **Graph:** DEPT records an edge where [E] depends on another [E], under context [S].



FATAL EXCEPTION: Circular Wait-State Detected (Deadlock)

DEPT (DEPendency Tracker)

A Path to Mainline and Solving the False Positive Challenge

Linux Plumbers Conference 2026

Yunseong Kim @ Ericsson Software Technology
yunseong.kim@est.tech



Case study ntfs3: ni_lock vs. PG_locked

Context A: truncate (creat(2) → do_truncate)

ntfs3_setattr

ntfs_truncate

ni_lock(ni) [S]

truncate_setsize

truncate_pagecache

truncate_inode_pages_range

folio_lock() → waits PG_locked [W]

← deadlock: each holds what the other waits for

Context B: read/write via page cache

filemap_read

filemap_get_pages

filemap_update_page

folio_trylock() → sets PG_locked [S]

filemap_read_folio

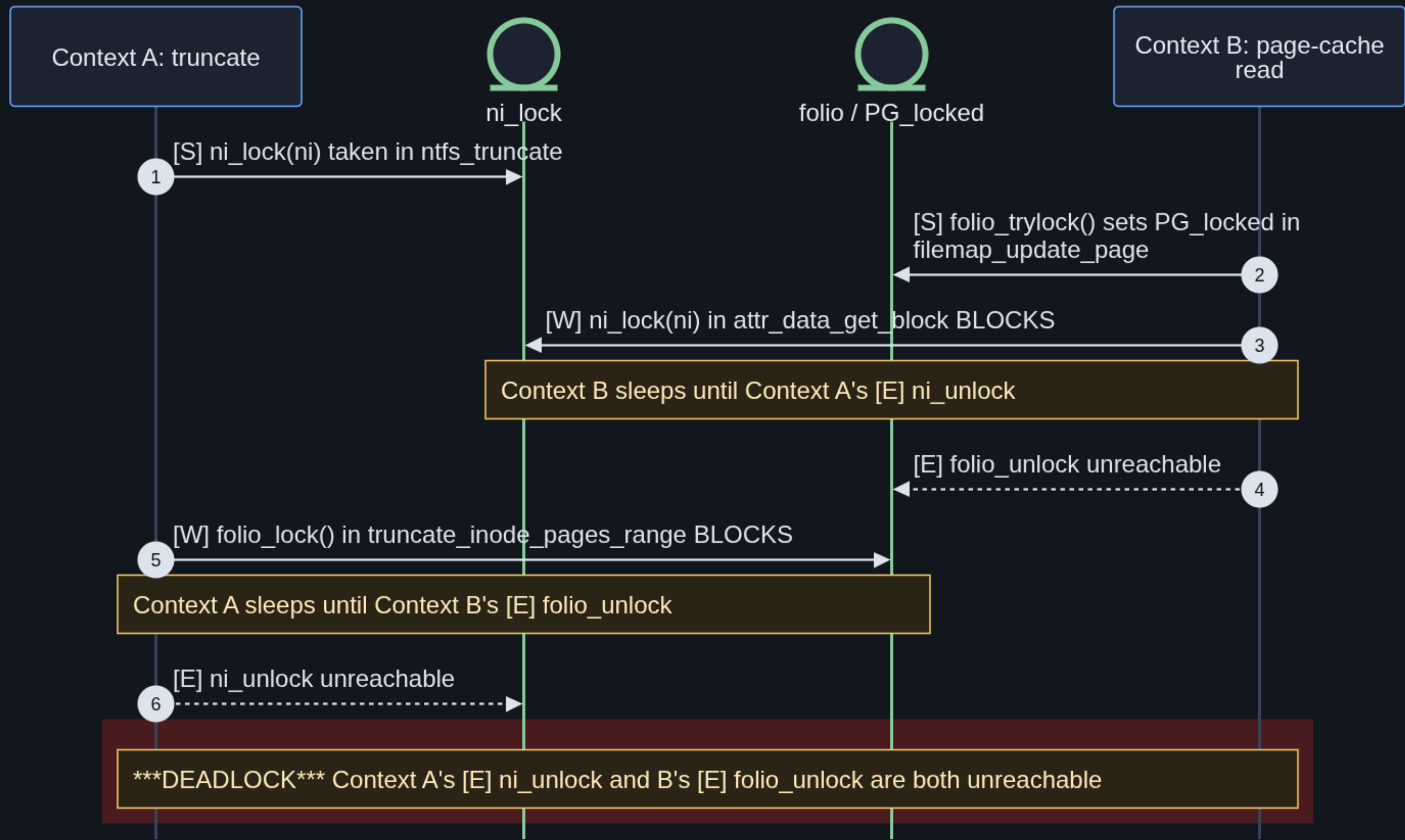
mpage_read_folio

do_mpage_readpage

ntfs_get_block_vbo

attr_data_get_block

ni_lock(ni) → blocks [W]



***** DEADLOCK ***** A holds `ni_lock` and waits for the `folio` · B holds the `folio` and waits for `ni_lock`

Case study hugetlb: folio lock vs. hugetlb_fault_mutex_table

Context A: hugetlb_wp

hugetlb_fault

mutex_lock(B)

filemap_lock_hugetlb_folio

filemap_lock_folio

folio_lock(A) [S]

hugetlb_wp

mutex_unlock(B) ← B released, A still held

...

unmap_ref_private

...

mutex_lock(B) -> blocks [W] ← Context A waits B (held by Context B), Context B waits A (held by Context A)

Context B: filemap_lock_hugetlb_folio

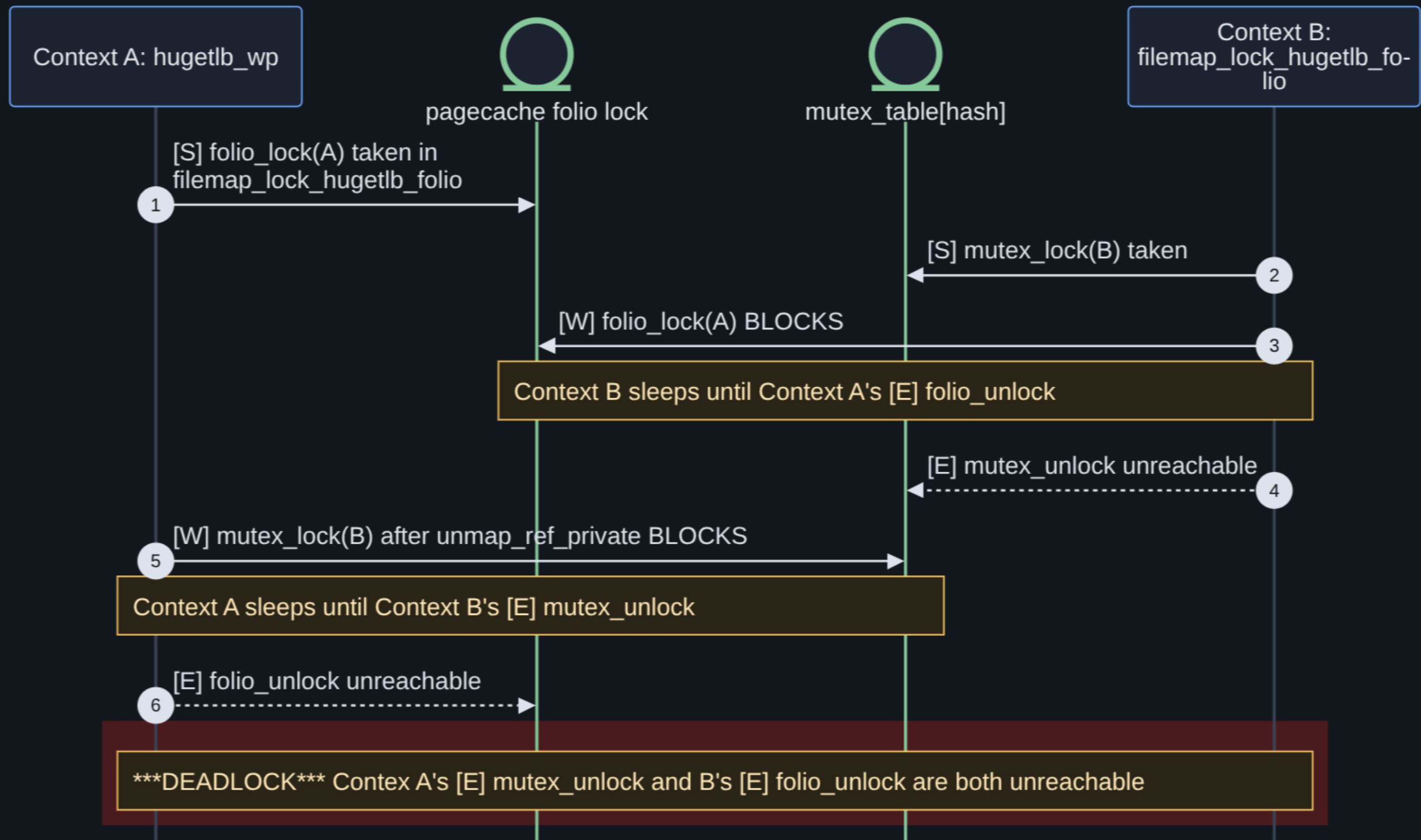
hugetlb_fault

mutex_lock(B) [S] (gets it)

filemap_lock_hugetlb_folio

filemap_lock_folio

folio_lock(A) → blocks [W]



***** DEADLOCK ***** Context A holds folio A and waits for mutex B · Context B holds mutex B and waits for folio A

References

- Fixes for case study (1)
 - [0226635c304c](#) "fs/ntfs3: don't hold ni_lock when calling truncate_setsize()" Tetsuo Handa, co-developed with Hilf Danton, debugged by Linus, applied directly by Linus for v6.2. Two lines moved, no new locking.
- Fixes for case study (2)
 - [2ae1ab9934c7](#) "mm,hugetlb: change mechanism to detect a COW on private mapping" (v6.17) Reported-by: Gavin Guo
 - Fixes: [40549ba8f8e0](#), Suggested-by: Peter Xu.



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026