Java Pathfinder and Verifying Nested Lock Priority Inheritance in RTEMS

Saurabh Gadia, **Cyrille Artho**, Gedare Bloom
KTH Royal Institute of Technology, Stockholm, Sweden
artho@kth.se

2019-05-07

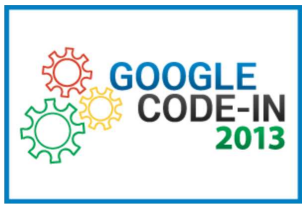
Outline

1. RTEMS
2. Threads, locks, and priorities
3. Priority Inheritance Protocol (PIP)
4. Model of PIP in Java Pathfinder, defect found
5. Results, conclusion

RTEMS, a compact open-source real-time OS

rtems

Real-Time Executive for Multiprocessor Systems



- Industrial-strength real-time operating system and tools.
- Fully open source (all components, tests, documentation).
- Portable, light-weight, modular.

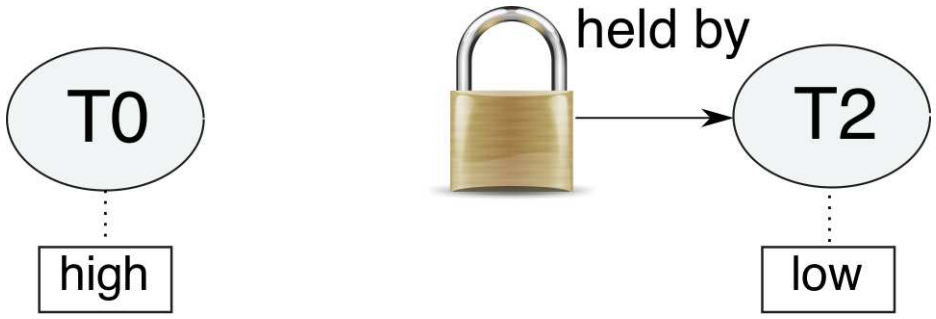
The Orocos Project
Smarter control in robotics & automation!

RTEMS compared to other real-time OSes

Single Process	ARINC Partitions	POSIX Processes		Other Separation
RTEMS	ARINC 653	LynxOS	RT-Linux	GNU/Linux	MS Windows
VxWorks	based RTOSes		Solaris	BSD	
ThreadX				other Unix	
Nucleus				Minix	
Real-Time Focus			General-Purpose Focus		

Adapted from slide © On-Line Applications Research Corporation

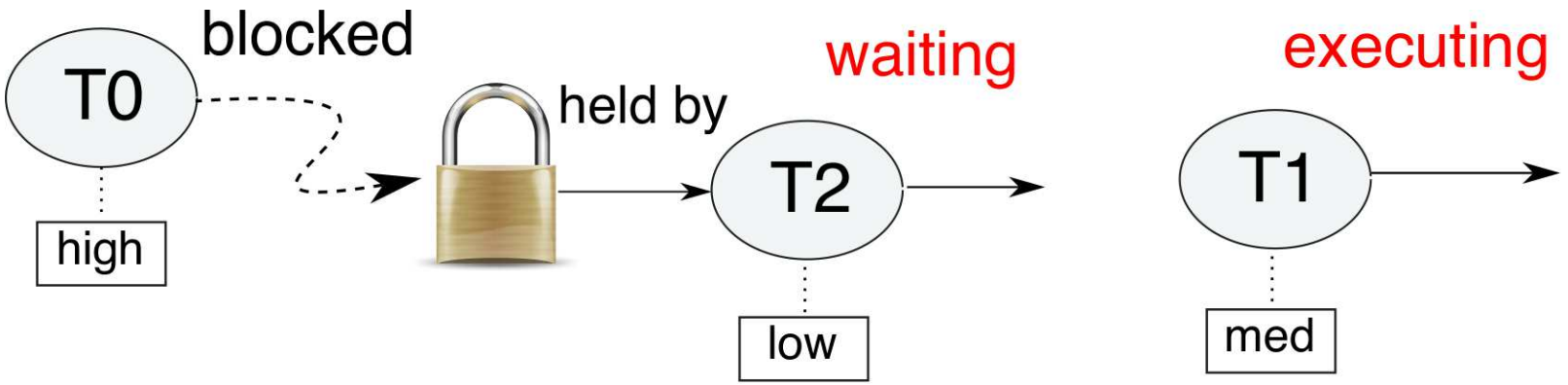
Thread concurrency and thread priorities



- Each thread in a program has
  - priority;
  - access to shared memory;
  - ability to hold locks.

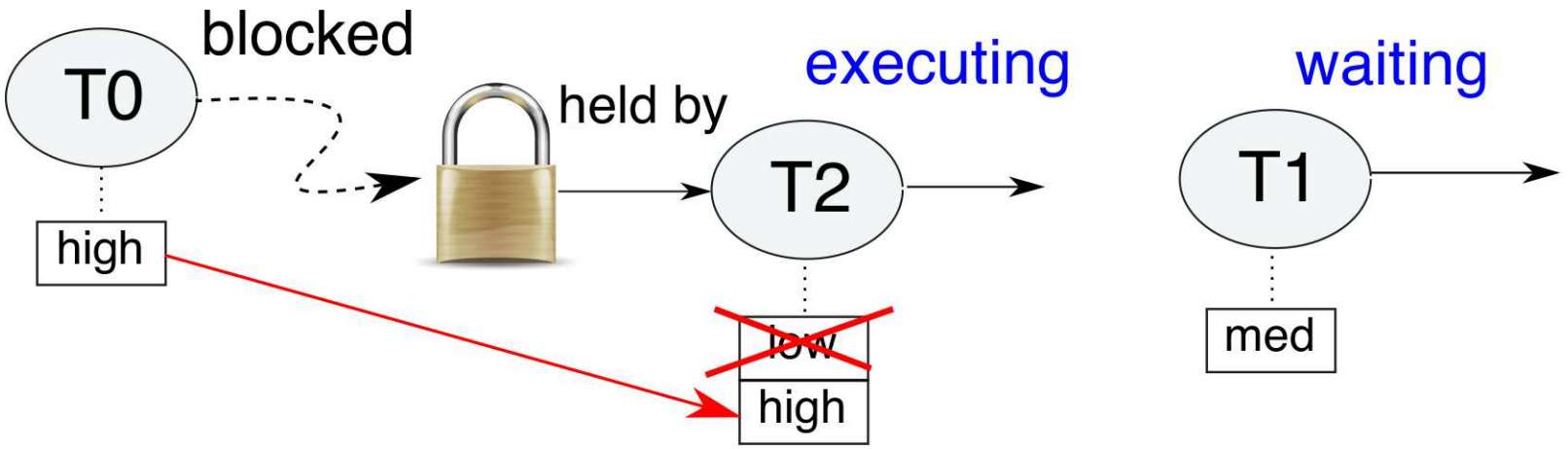
- Low-prio. thread: release lock within worst-case exec. time!

Priority inversion



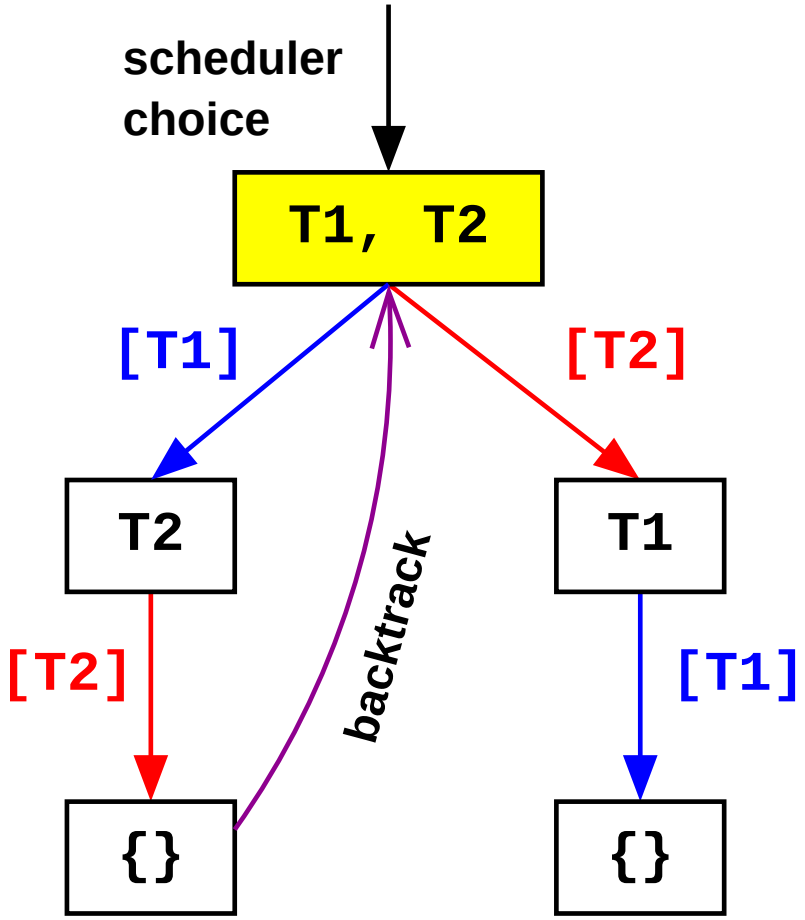
- Mid-priority thread starves low-priority thread.
- High-priority thread never gets lock and runs!

Priority Inheritance Protocol (PIP)



- Increase priority of lock holder when other thread wants lock.
- Simple solution works well when locks are not nested.

Java Pathfinder (JPF): A model checker for Java



- Explore all thread schedules, find all possible failures.
- Backtrack whole program.
- Java as modeling language.

Verify PIP with JPF!

History of Java Pathfinder

- **1999: Project started** at **NASA Ames**.
- **2005: Open sourced** on sourceforge.
- 2008: First participation in **Google Summer of Code.**
- **2017: Moved to github.**
- Many collaborators/users:
  - > 20 univ. (USA, Canada, Japan, South Africa, Europe).
  - Fujitsu, other Fortune 500 companies.

Model of RTEMS kernel code in Java

RTEMS resource	Java/JPF equivalent
Kernel locks	`synchronized` block usage
Thread signaling	`wait` and `notify`
Priority queue	`java.util.PriorityQueue`
Global scheduler lock	`Verify.beginAtomic`, `endAtomic`

- All constructs have logical equivalent in Java/JPF.
- Global scheduler lock used in uniprocessor kernel; class `Verify` is JPF-specific.

Java/JPF very convenient for modeling C code!

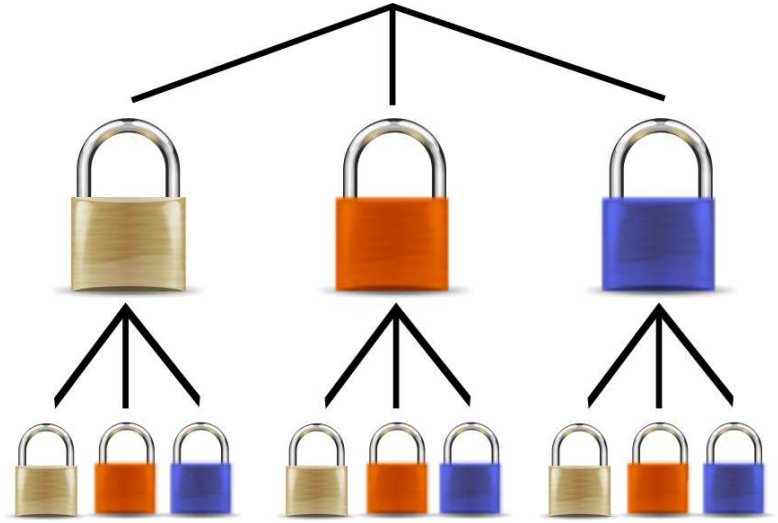
Lock usage model

```
public class TestThread extends RTEMSThread {
  Lock locks[] = { lock(Verify.getInt(3)),
    lock(Verify.getInt(3))};
  // choose two locks at random (out of three)
  // same lock or different locks can be chosen
  setPriority(Verify.getInt(3)); // randomize priority
  public void run() {
    for (int i = 0; i < 2; i++)
      locks[i].lock(); // acquire all (two) locks
    for (int i = 2-1; i>= 0; i--)
      locks[i].unlock(); // release in reverse order
    assert currentPriority==realPriority;
} }
```

Code contains nine properties related to PIP.

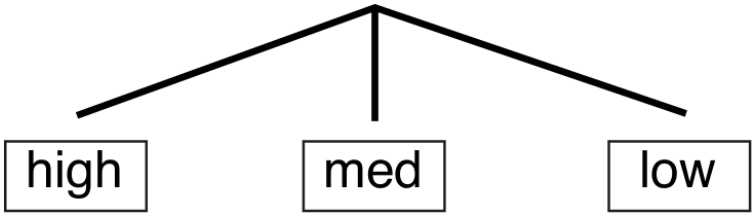
State space explosion

Locks per thread



9 choices per thread

Thread priority



3 choices per thread

Total of $3^{3^3} = 19683$ choices!

Ignore isomorphic configurations

1. Lock index does not matter

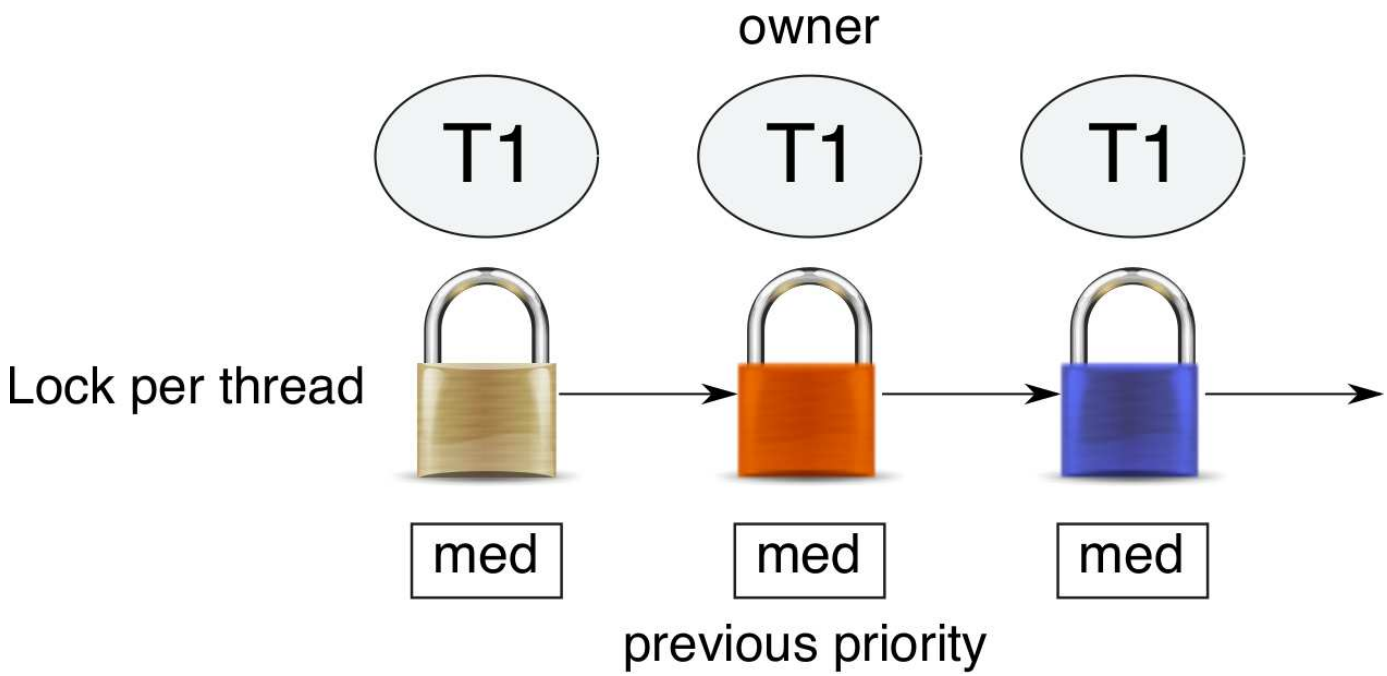
2. Permutation does not matter

3. Thread priorities: only relative priority matters.

From 19683 to 124 configurations.

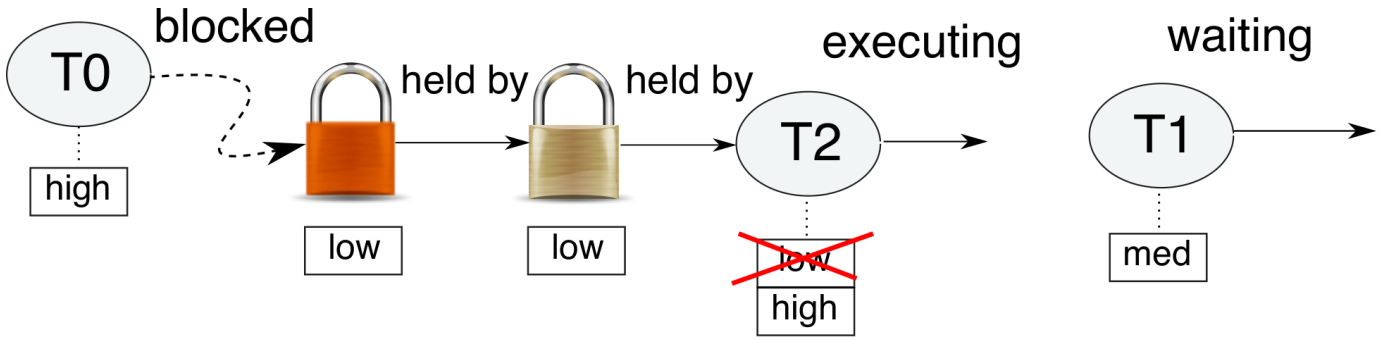
PIP in RTEMS (data structures)

- Each thread has a list of locks being held.

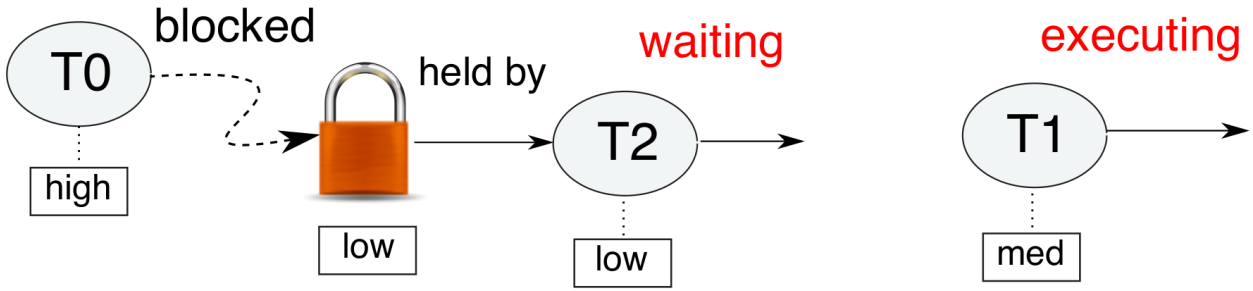


- Each lock knows
  - its owner and
  - the priority the owner had when it first acquired the lock.

Defect found



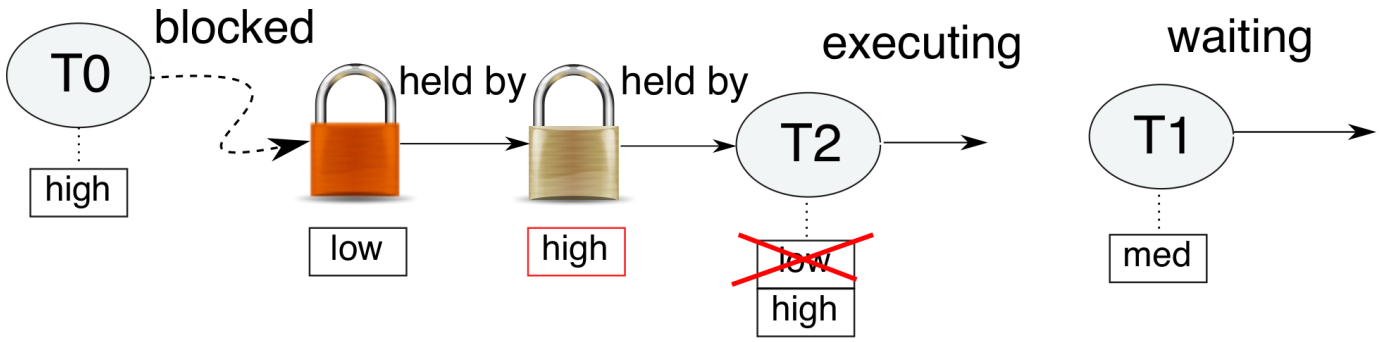
Priority Inheritance Potocol (PIP)



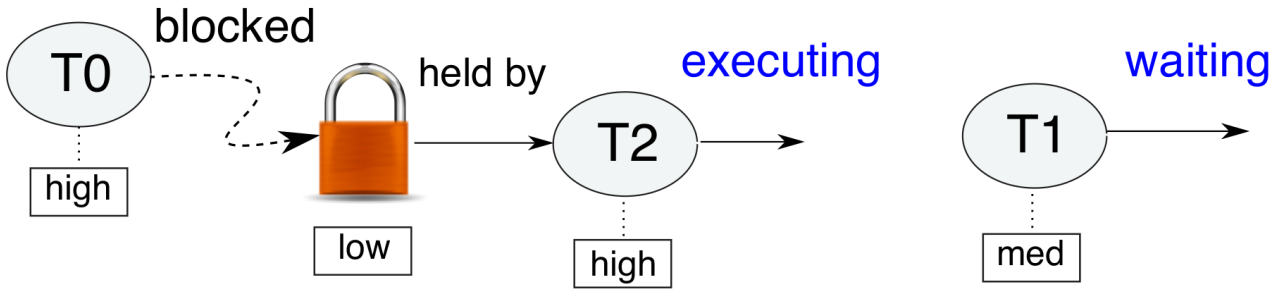
Uncontrolled Priority Inversion

Low priority after releasing first lock → starvation by T1.

Fixed version



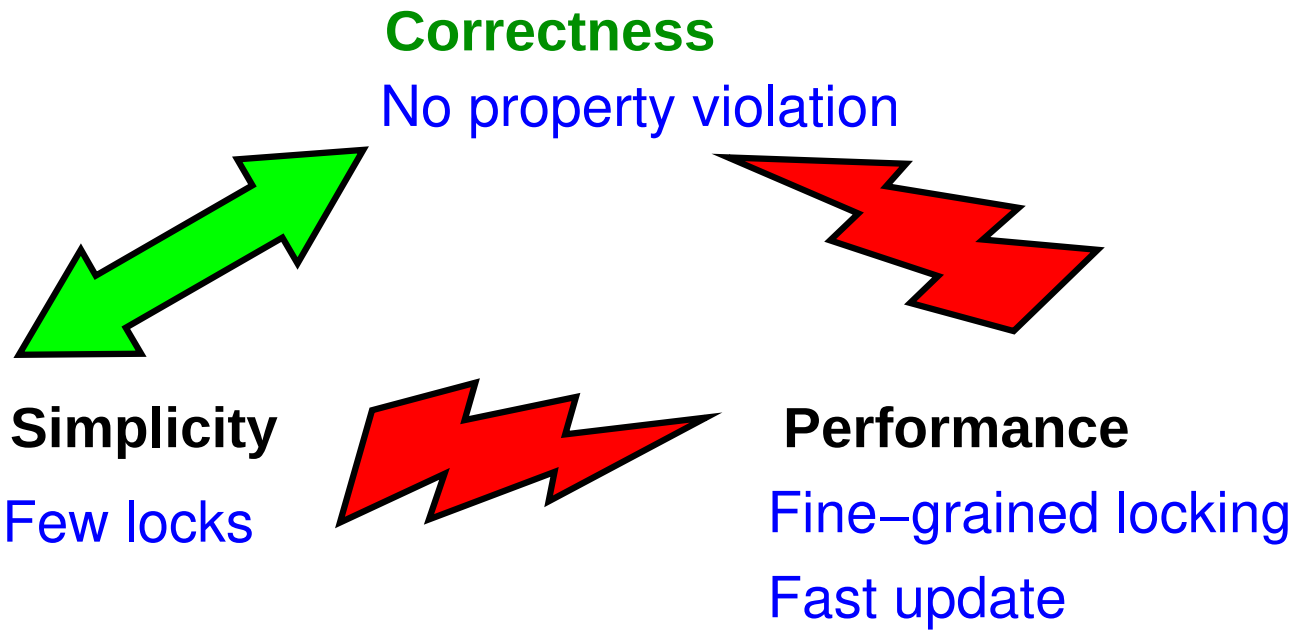
Priority update as per proposed solution



Deterministic system with no priority inversion

Step down priority after releasing the last lock.

Discussion of verification results



- Variants: uniprocessor, global lock, fine-grained locking.
- High code complexity (many locks, recursion on unlock).

PIP with nested locks: Not suitable for real-time OS kernel.

Thanks to Google Summer of Code

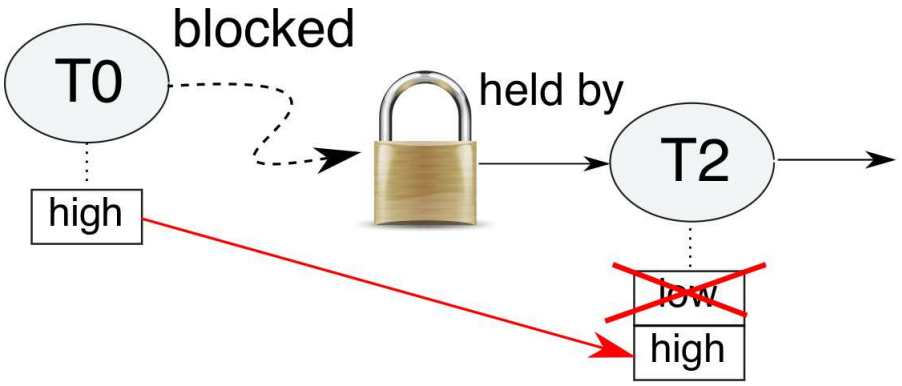
- Promote Open Source Software development.
- About 1,000 students per year, 12 years so far.
- 2016: 1,202 students, 178 organizations.

Google Summer of Code

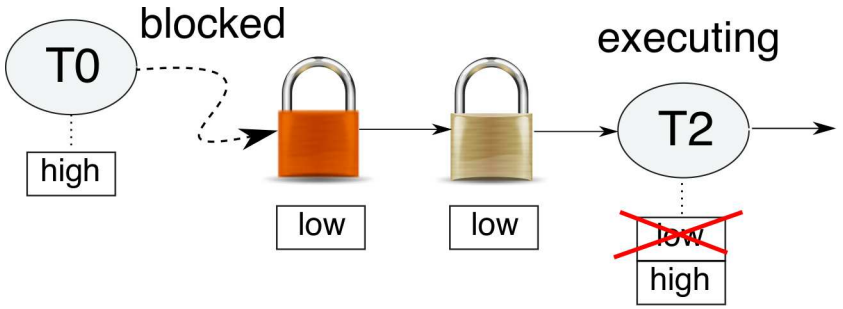
GSoC logo by Google

GSoC supported both RTEMS and Java Pathfinder.

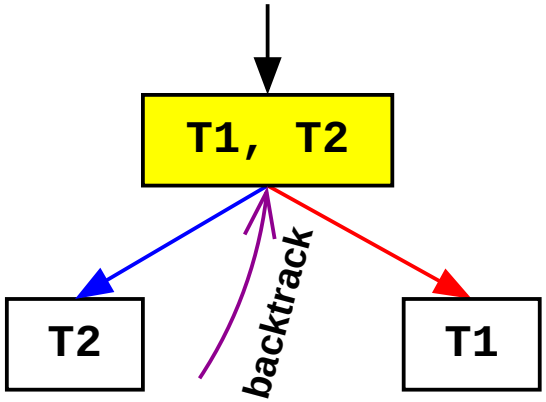
Summary



1. PIP handles priority inversion.



2. Nested locking = difficult



3. Model, verify with Java/JPF.

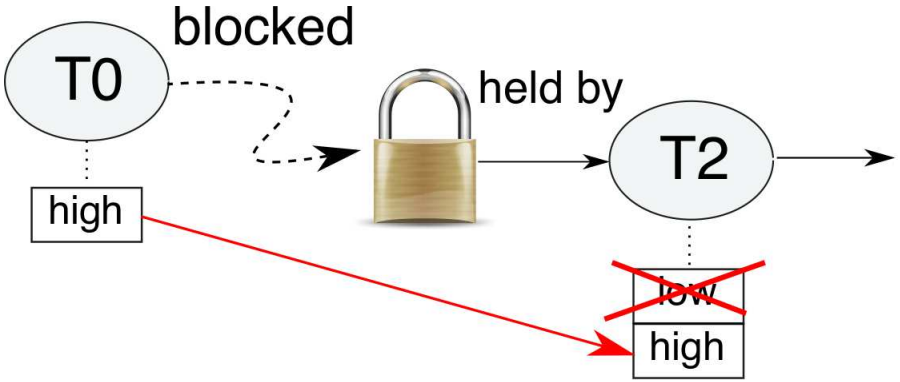
Future work: $O(m)$ Independence-Preserving Protocol (OMIP).

Discussion

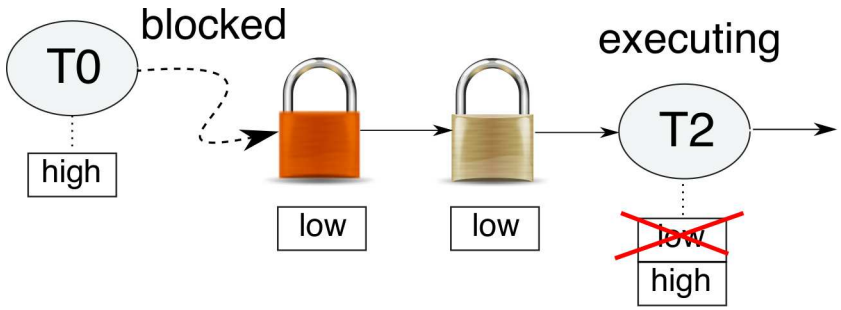
An odd way of using JPF: Java as a modelling language!

	JPF	SPIN or NuSMV	Theorem Prover
Ease of use	easy	medium	hard
Scalability	3 threads, 2 locks	perhaps 4 threads, 3 locks(?)	∞
Use case	bug finding	verification with limits	full verification

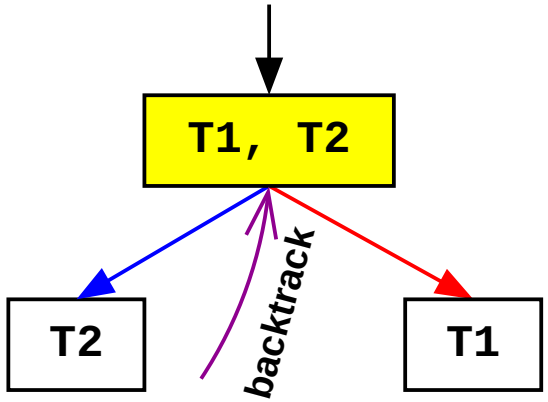
Verifying Nested Lock Priority Inheritance in RTEMS with Java Pathfinder



1. PIP handles priority inversion.



2. Nested locking = difficult



3. Model, verify with Java/JPF.

Future work: $O(m)$ **Independence-Preserving Protocol (OMIP).**