Deadlock Immunity

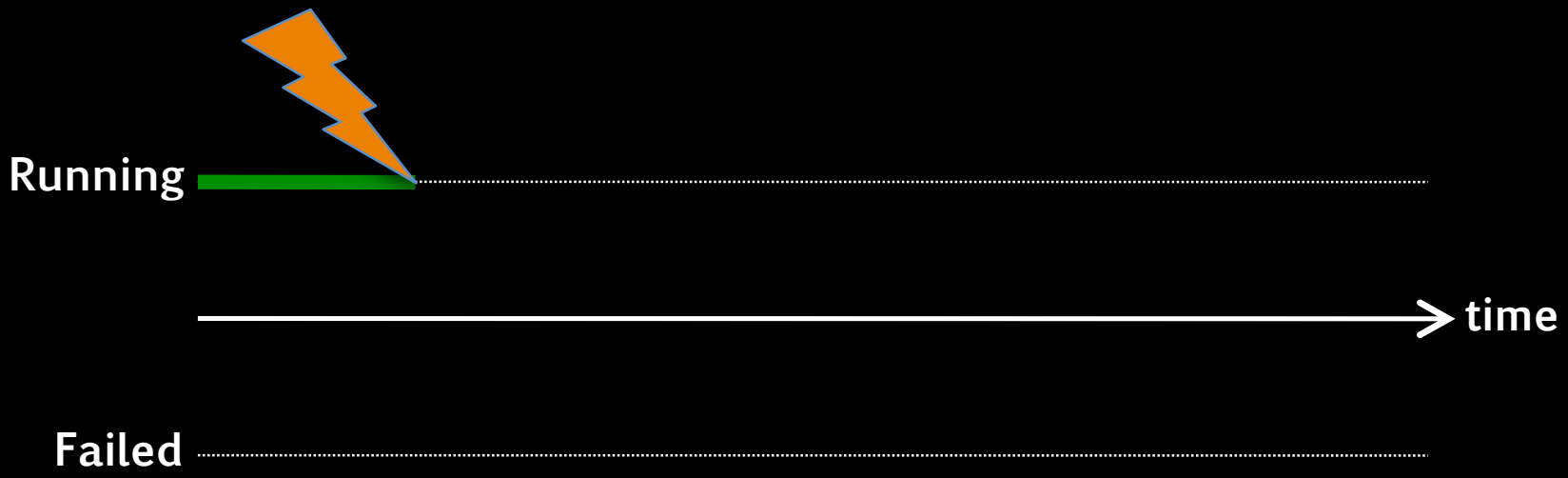
Enabling Systems To Defend Against Deadlocks

Horatiu Jula, Daniel Tralamazza,
Cristian Zamfir, George Candea

Dependable Systems Laboratory

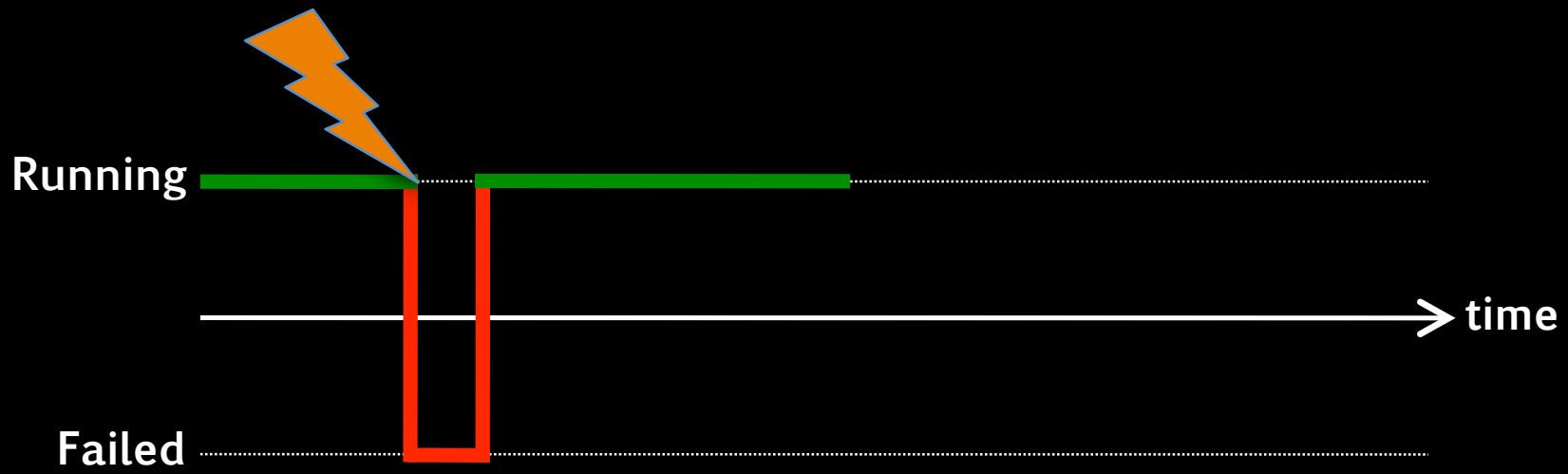
Real Software System

Real Software System

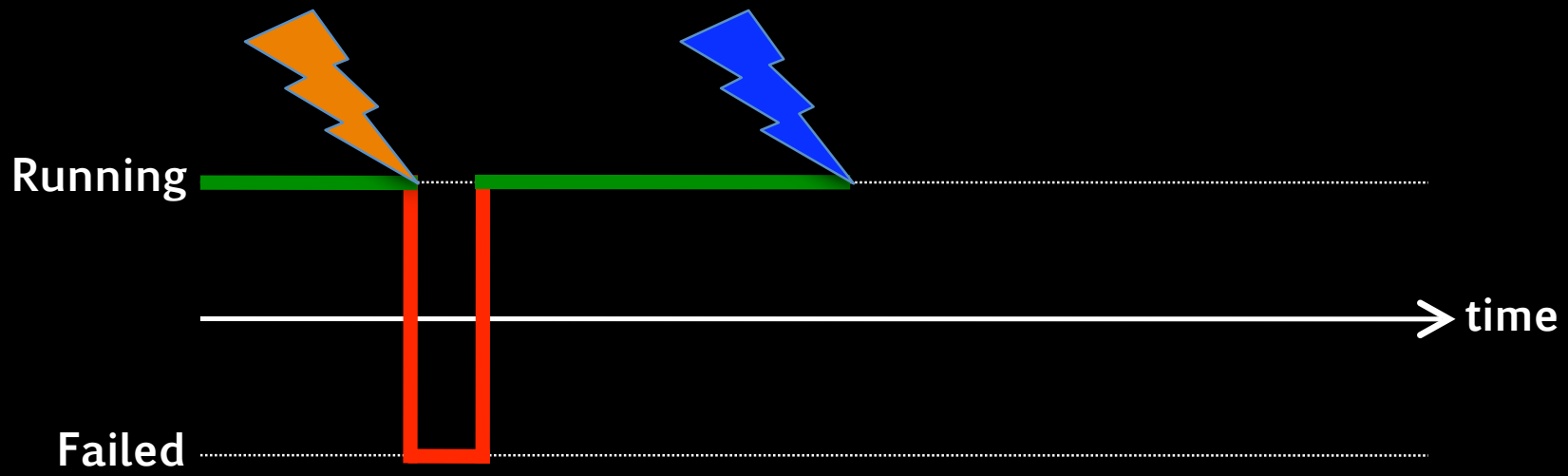


Real Software System

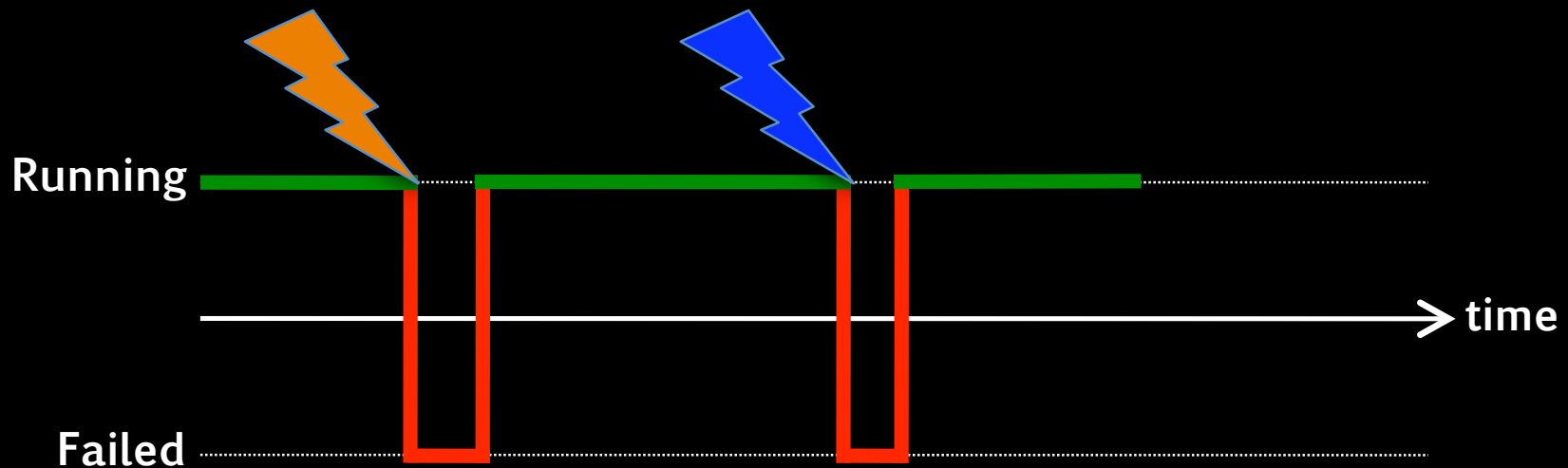
Real Software System



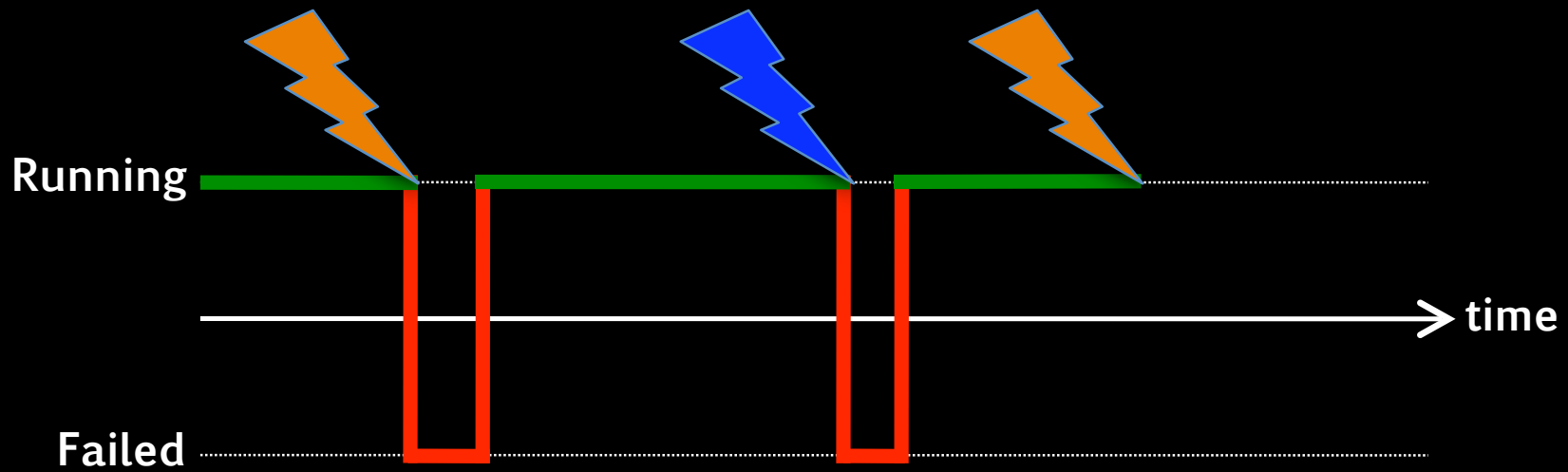
Real Software System



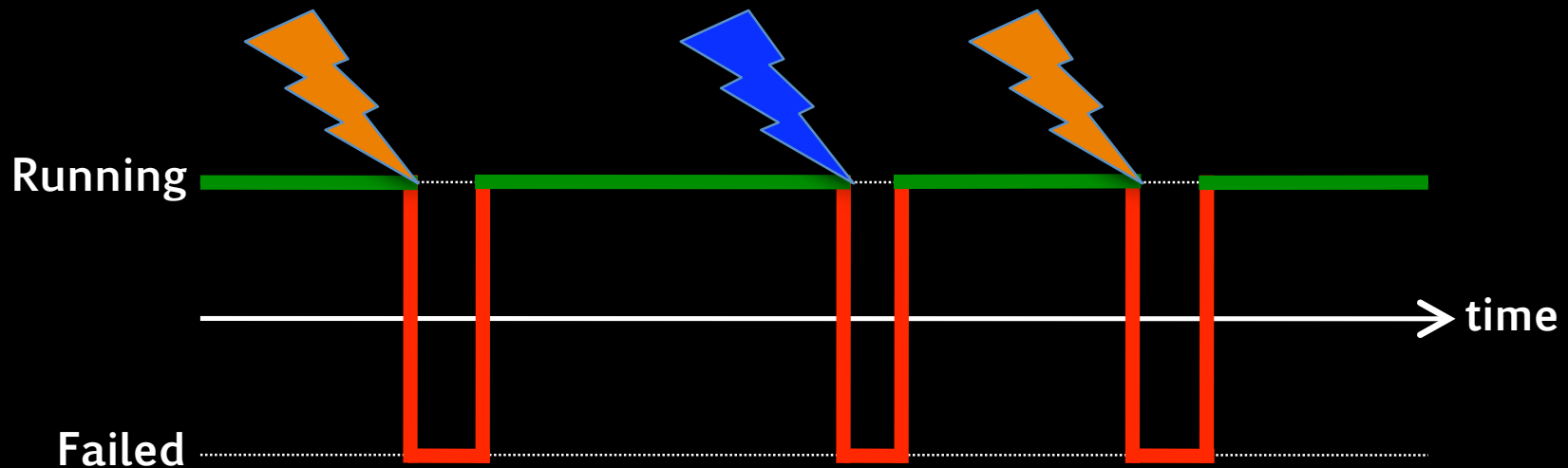
Real Software System



Real Software System



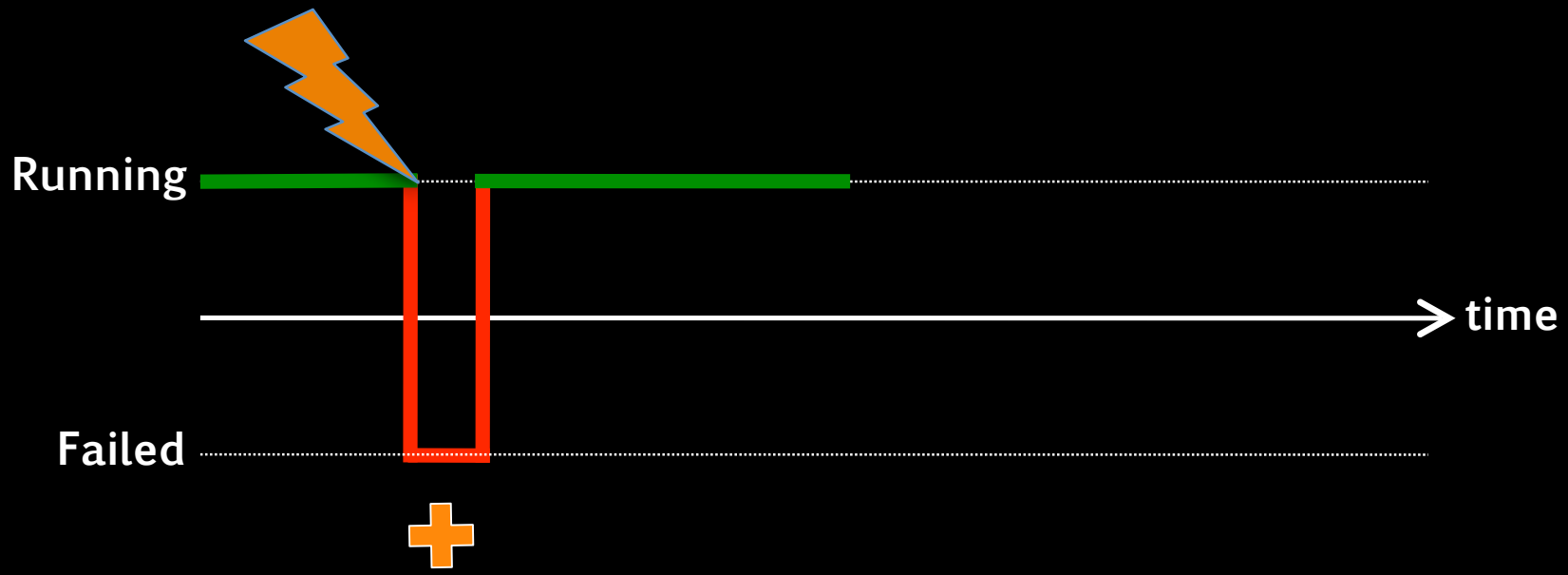
Real Software System



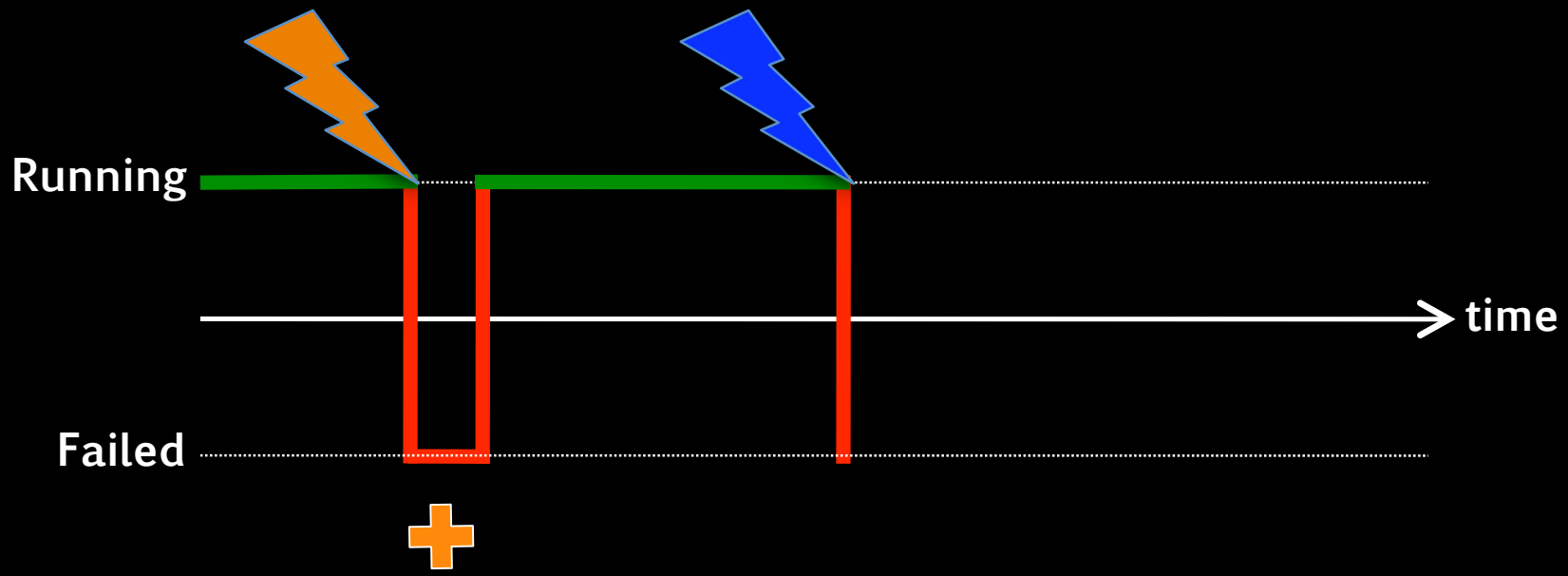
Real Software System

Real Software with An Immune System

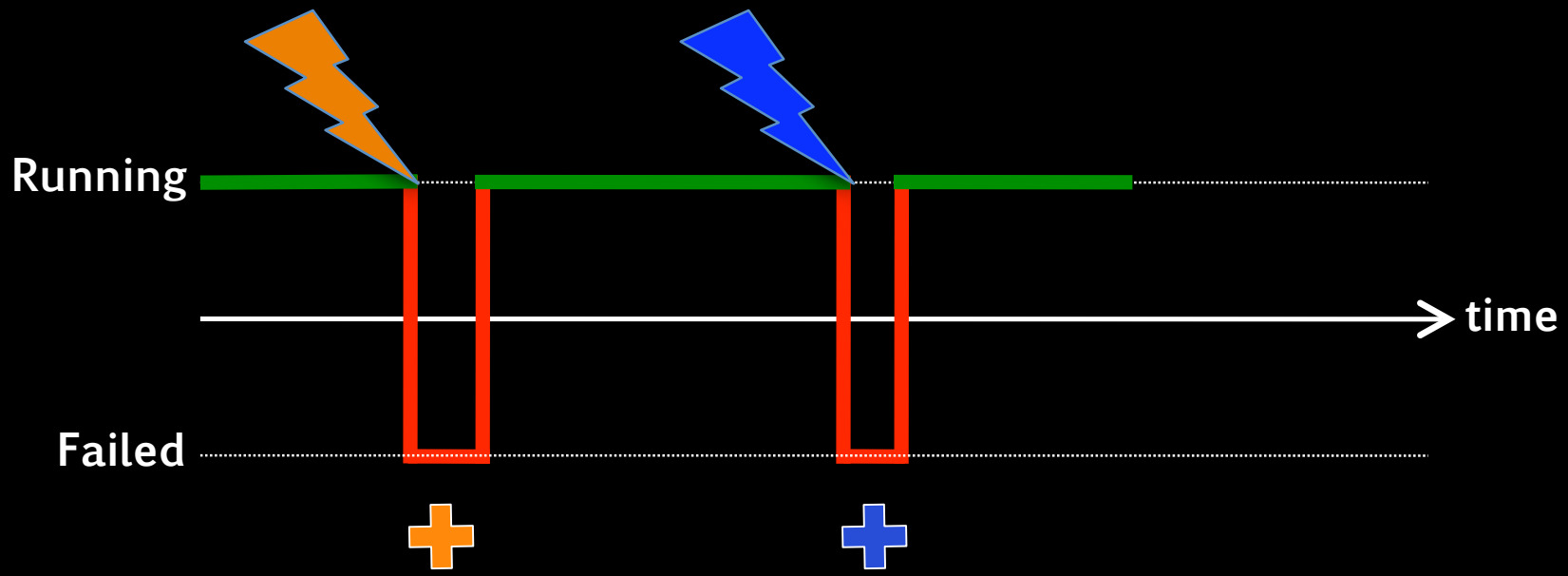
Real Software with An Immune System



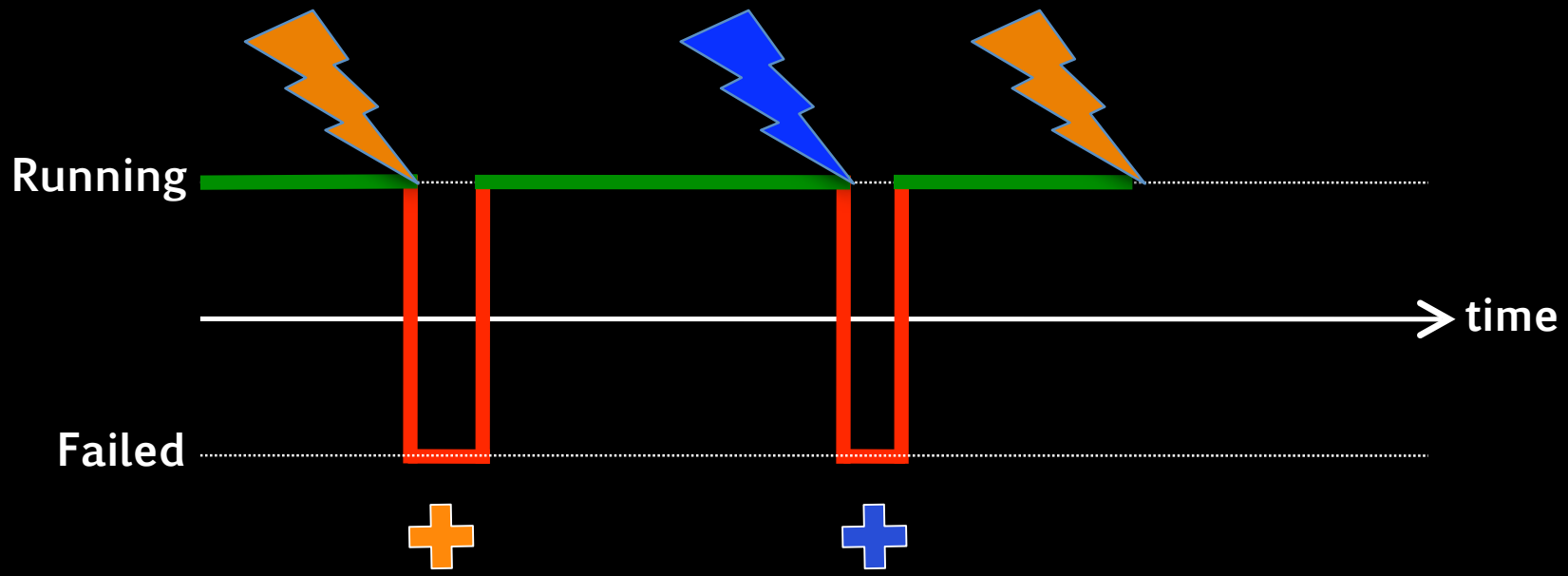
Real Software with An Immune System



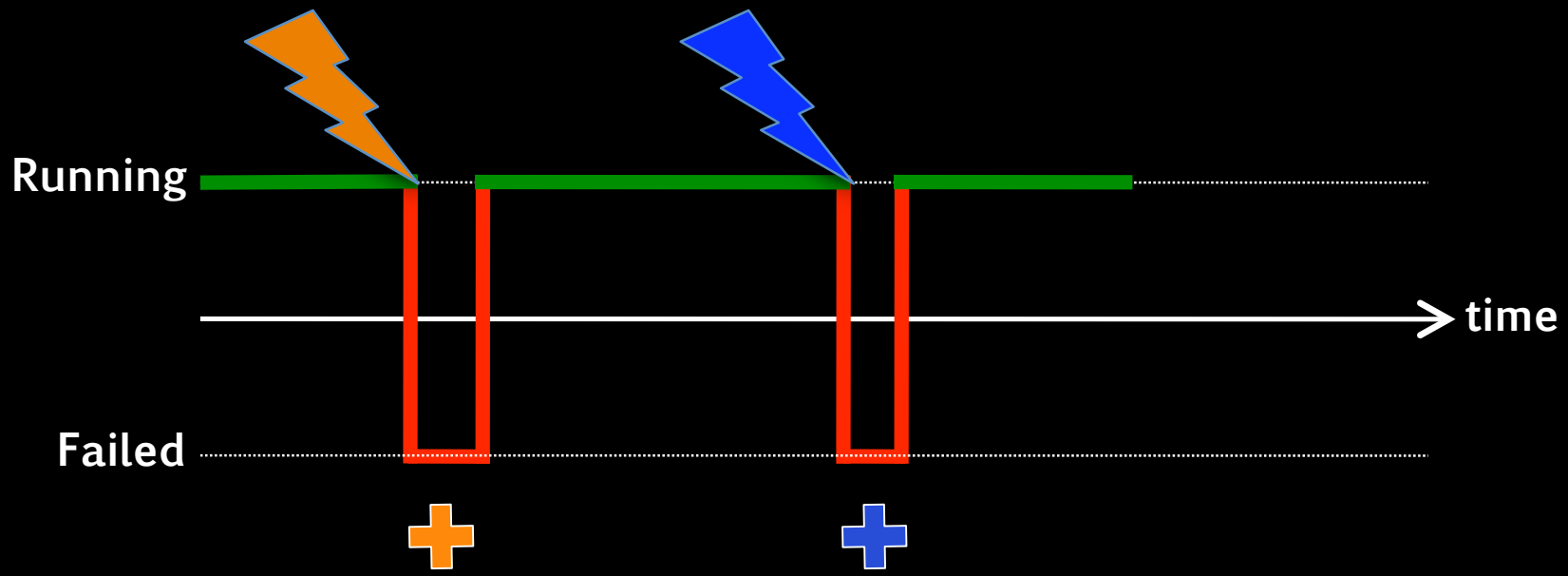
Real Software with An Immune System



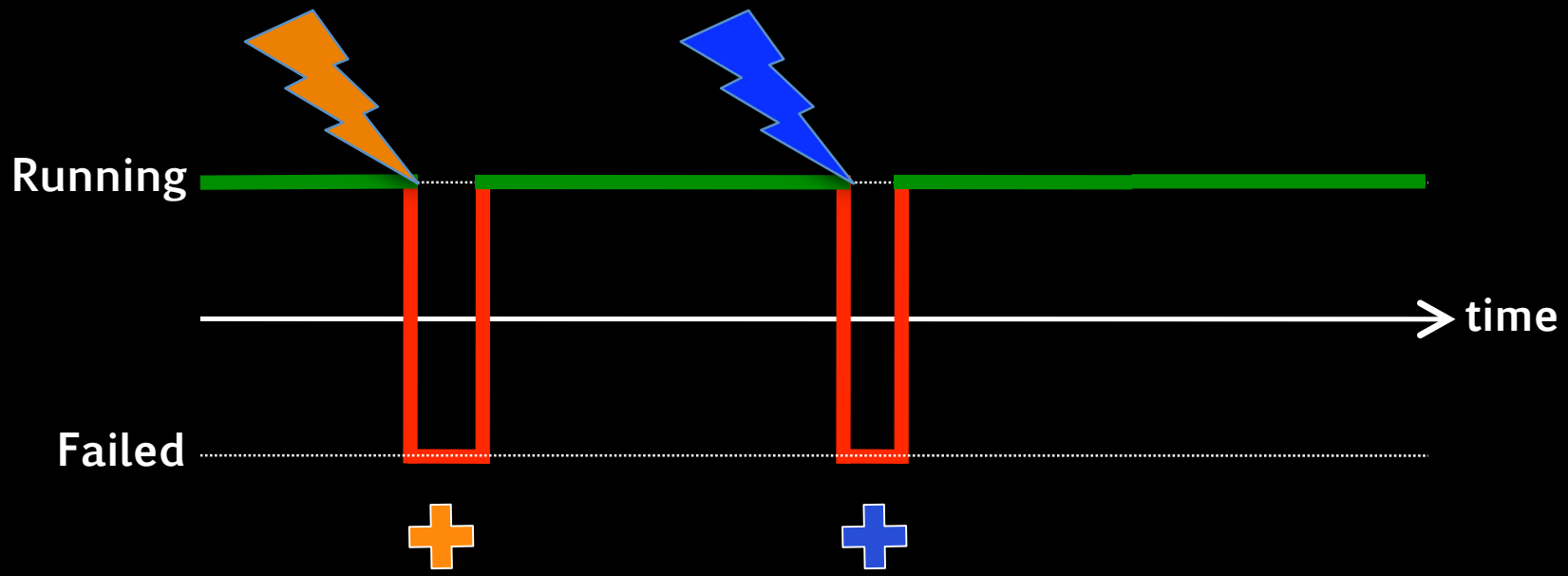
Real Software with An Immune System



Real Software with An Immune System



Real Software with An Immune System



Real Software with An Immune System

Outline

- Immunity Against Deadlocks
- Dimmunix Overview
- Challenges & Solutions
- Discussion & Conclusions

Deadlock Immunity

- Learn executions that lead to deadlock
- Save fingerprints of encountered deadlock patterns into a persistent history
- Avoid executions patterns that have led to deadlock in the past

Example Uses

- **Software vendors**
  - *Example:* **Protect a deadlock-free Web browser**
    - **Users extend browsers with plugins not controlled by the vendor… these can deadlock the browser**
    - **Deadlock occurs → obtain fingerprint → distribute it to all**
- **End users**
  - *Example:* **Deadlock-prone closed-source legacy DB server**
    - **Develop immunity against deadlocks**
    - **No need to upgrade or wait for patches**

```
NLboolean nlClose(NLsocket socket)
{
  if(driver)
  {
    if(nlIsValidSocket(socket) == NL_TRUE)
    {
      nlLockSocket(socket, NL_BOTH);
      driver->Close(socket);
      nlMutexLock(&socklock);
      nlReturnSocket(socket);
      nlMutexUnlock(&socklock);
      nlUnlockSocket(socket, NL_BOTH);
      return NL_TRUE;
    }
    else
    {
      nlSetError(NL_INVALID_SOCKET);
      return NL_TRUE;
    }
  }
  nlSetError(NL_NO_NETWORK);
  return NL_FALSE;
}
```

```
void nlShutdown(void)
{
  if(--nlInitCount > 0)
        return;
  nlMutexLock(&socklock);
  if(nlSockets != NULL)
  {
    NLsocket i;
    for(i=0;i<nlNextsocket;i++)
    {
      if(nlSockets[i] != NULL)
      {
        nlLockSocket(i, NL_BOTH);
        driver->Close(i);
        nlReturnSocket(i);
        nlUnlockSocket(i, NL_BOTH);
        nlFreeSocket(i);
      }
    }
    free(nlSockets);
    nlSockets = NULL;
  }
  nlMutexUnlock(&socklock);
  if(driver != NULL)
  {
    driver->Shutdown();
    driver->initialized = NL_FALSE;
    driver = NULL;
  }
  else
  {
     ....
```

```
NLboolean nlClose(NLsocket socket)
{
  if(driver)
  {
    if(nlIsValidSocket(socket) == NL_TRUE)
    {
      nlLockSocket(socket, NL_BOTH);
      driver->Close(socket);
      nlMutexLock(&socklock);
      nlReturnSocket(socket);
      nlMutexUnlock(&socklock);
      nlUnlockSocket(socket, NL_BOTH);
      return NL_TRUE;
    }
    else
    {
      nlSetError(NL_INVALID_SOCKET);
      return NL_TRUE;
    }
  }
  nlSetError(NL_NO_NETWORK);
  return NL_FALSE;
}
```

```
void nlShutdown(void)
{
  if(--nlInitCount > 0)
      return;
  nlMutexLock(&socklock);
  if(nlSockets != NULL)
  {
    NLsocket i;
    for(i=0;i<nlNextsocket;i++)
    {
      if(nlSockets[i] != NULL)
      {
        nlLockSocket(socket, NL_BOTH);
        driver->Close(i);
        nlReturnSocket(i);
        nlUnlockSocket(i, NL_BOTH);
        nlFreeSocket(i);
      }
    }
    free(nlSockets);
    nlSockets = NULL;
  }
  nlMutexUnlock(&socklock);
  if(driver != NULL)
  {
    driver->Shutdown();
    driver->initialized = NL_FALSE;
    driver = NULL;
  }
  else
  {
     ....
```

Thread 1

```
nlLockSocket(socket, NL_BOTH);
nlMutexLock(&socklock);
```

Thread 2

```
nlMutexLock(&socklock);
nlLockSocket(socket, NL_BOTH);
```

Thread 1

```
nlLockSocket(socket, NL_BOTH);
nlMutexLock(&socklock);
```

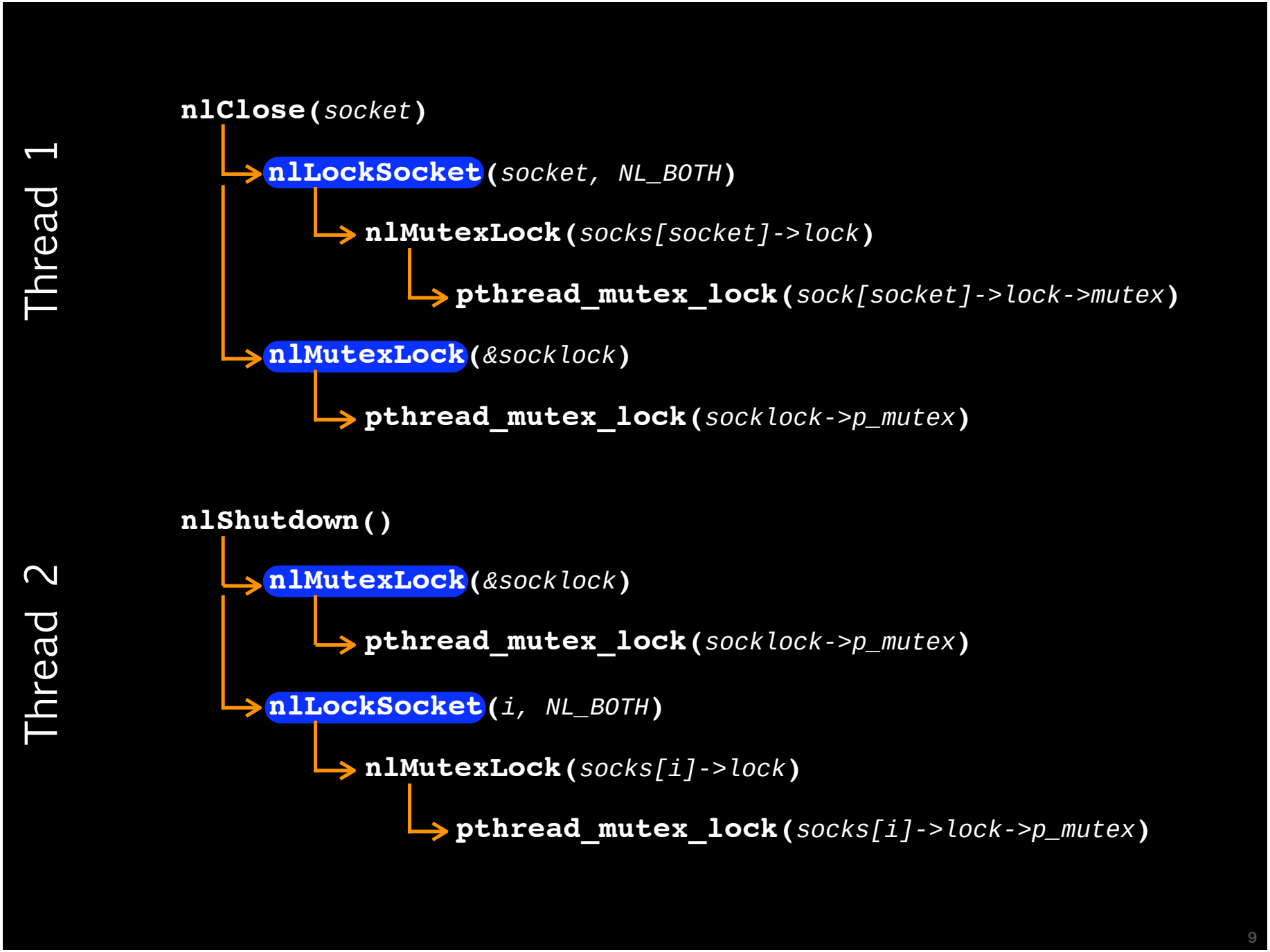
Thread 2

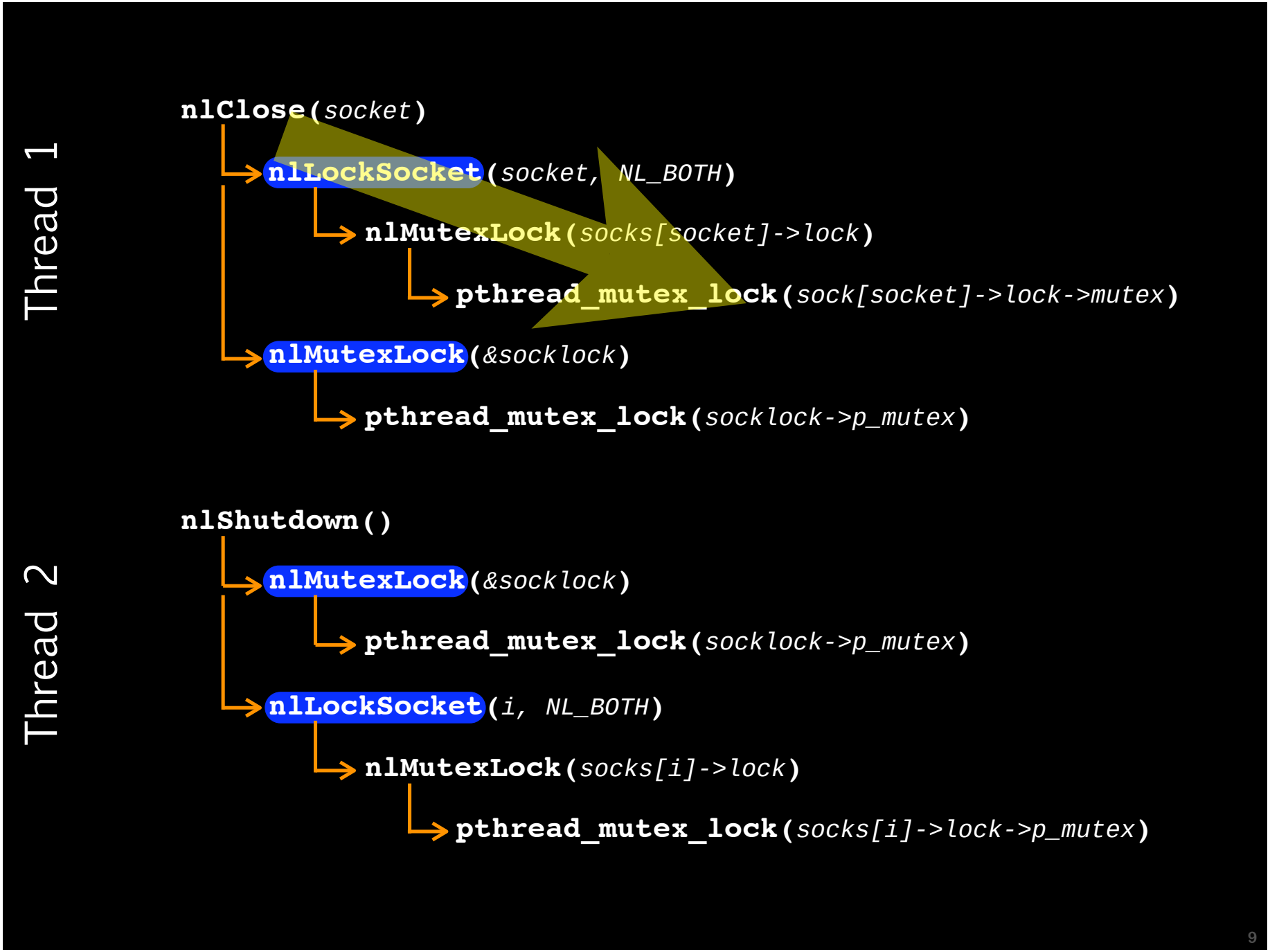
```
nlMutexLock(&socklock);
nlLockSocket(socket, NL_BOTH);
```

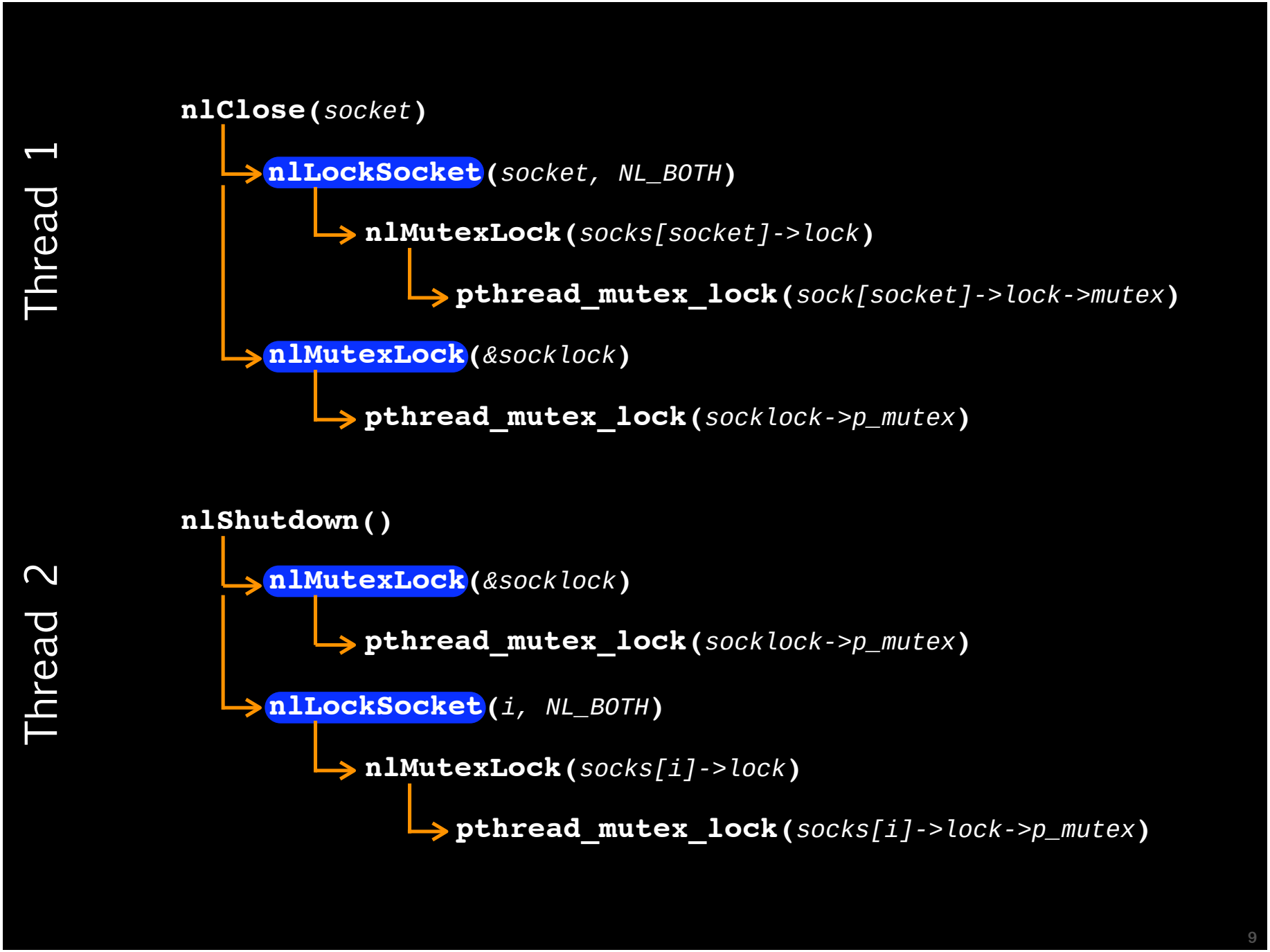
Lock Inversion => Deadlock Bug

```
nlLockSocket(socket, NL_BOTH)
nlMutexLock(&socklock)
```

```
nlMutexLock(&socklock)
nlLockSocket(i, NL_BOTH)
```


Thread 1
nlClose(socket)
nlLockSocket(socket, NL_BOTH)
nlMutexLock(socks[socket]->lock)
pthread_mutex_lock(sock[socket]->lock->mutex)
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
Thread 2
nlShutdown()
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
nlLockSocket(i, NL_BOTH)
nlMutexLock(socks[i]->lock)
pthread_mutex_lock(socks[i]->lock->p_mutex)


Thread 1
nlClose(socket)
nlLockSocket(socket, NL_BOTH)
nlMutexLock(socks[socket]->lock)
pthread_mutex_lock(sock[socket]->lock->mutex)
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
Thread 2
nlShutdown()
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
nlLockSocket(i, NL_BOTH)
nlMutexLock(socks[i]->lock)
pthread_mutex_lock(socks[i]->lock->p_mutex)

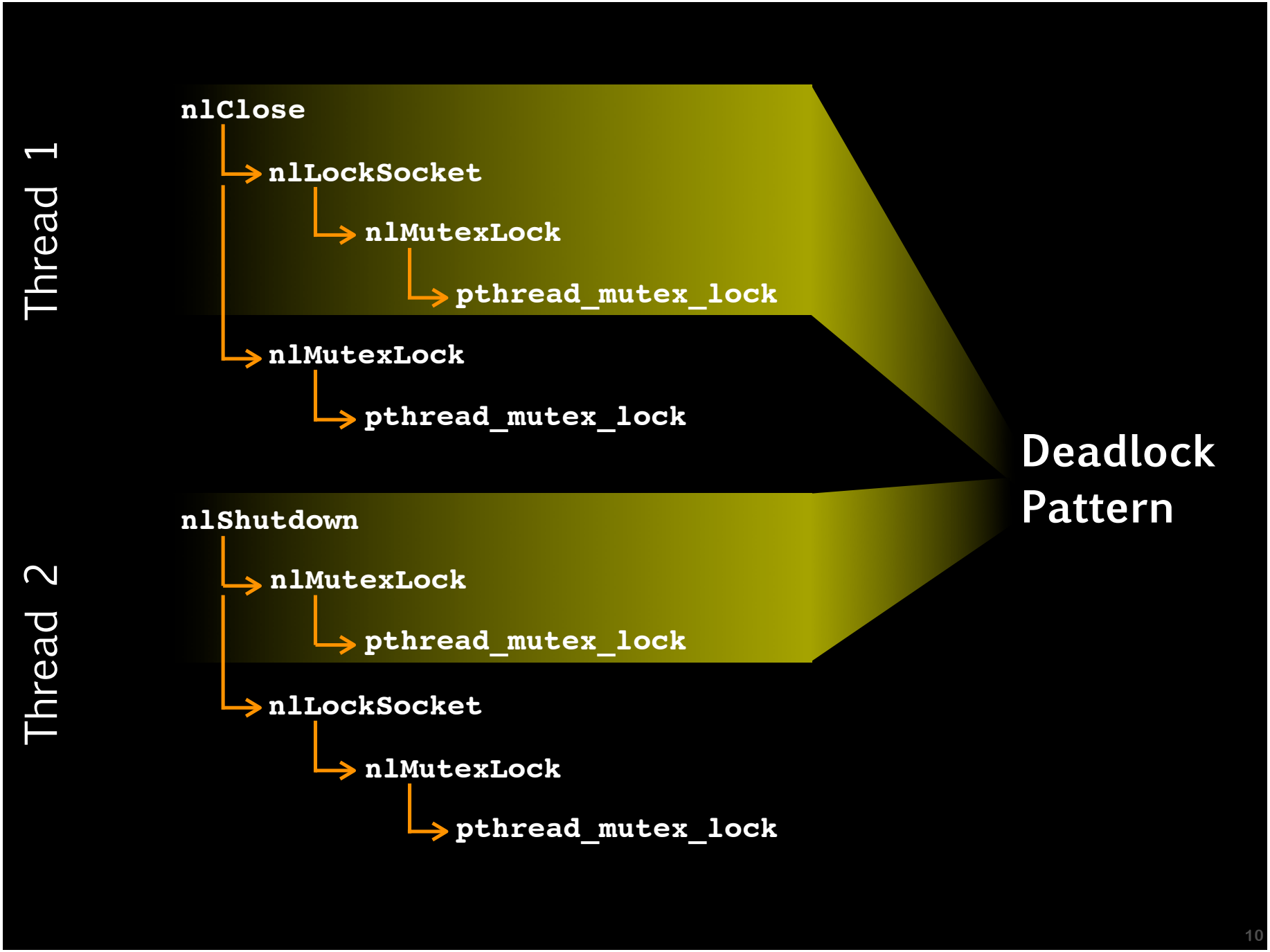

Thread 1
nlClose(socket)
nlLockSocket(socket, NL_BOTH)
nlMutexLock(socks[socket]->lock)
pthread_mutex_lock(sock[socket]->lock->mutex)
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
Thread 2
nlShutdown()
nlMutexLock(&socklock)
pthread_mutex_lock(socklock->p_mutex)
nlLockSocket(i, NL_BOTH)
nlMutexLock(socks[i]->lock)
pthread_mutex_lock(socks[i]->lock->p_mutex)

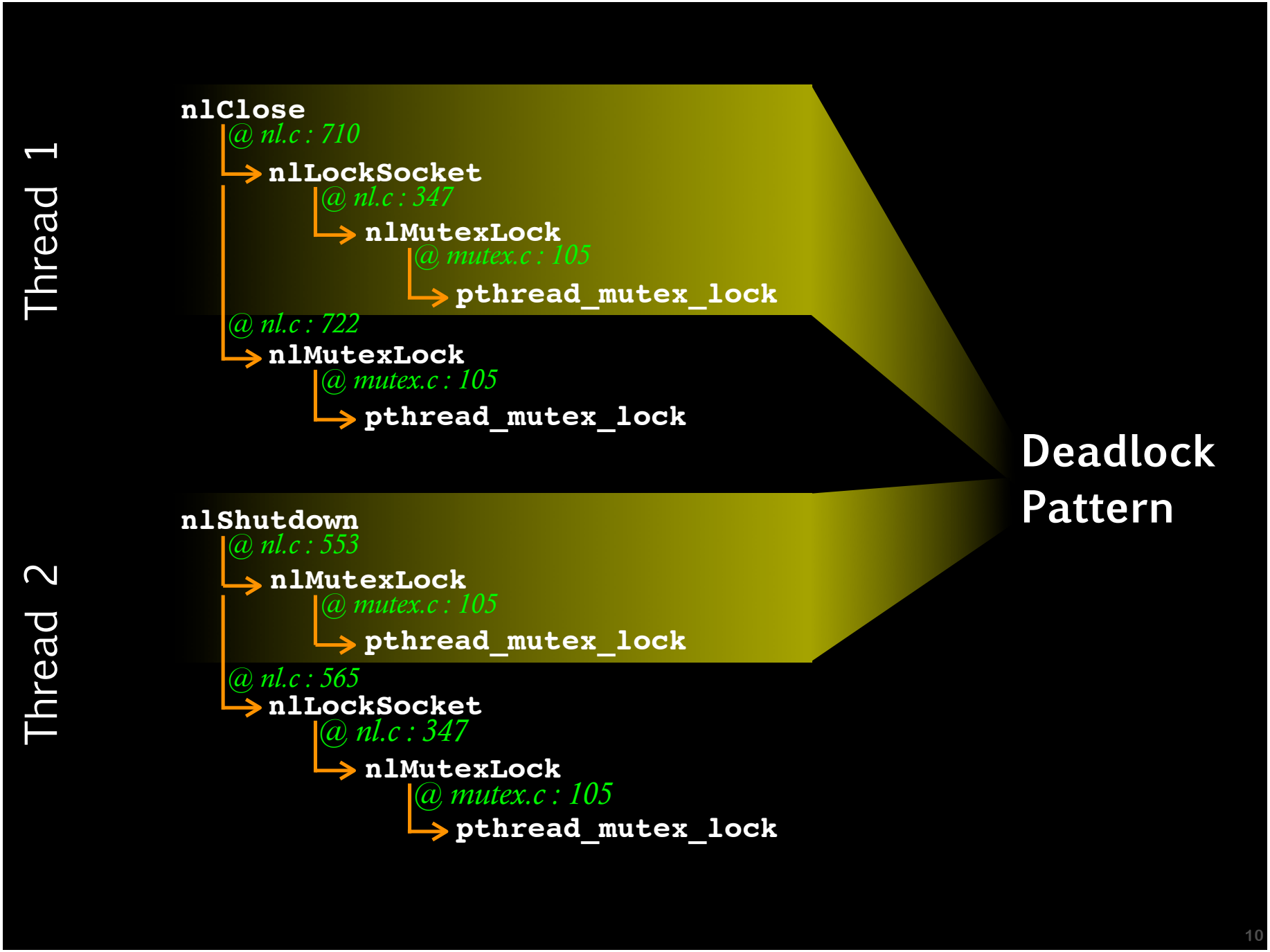
Thread 1

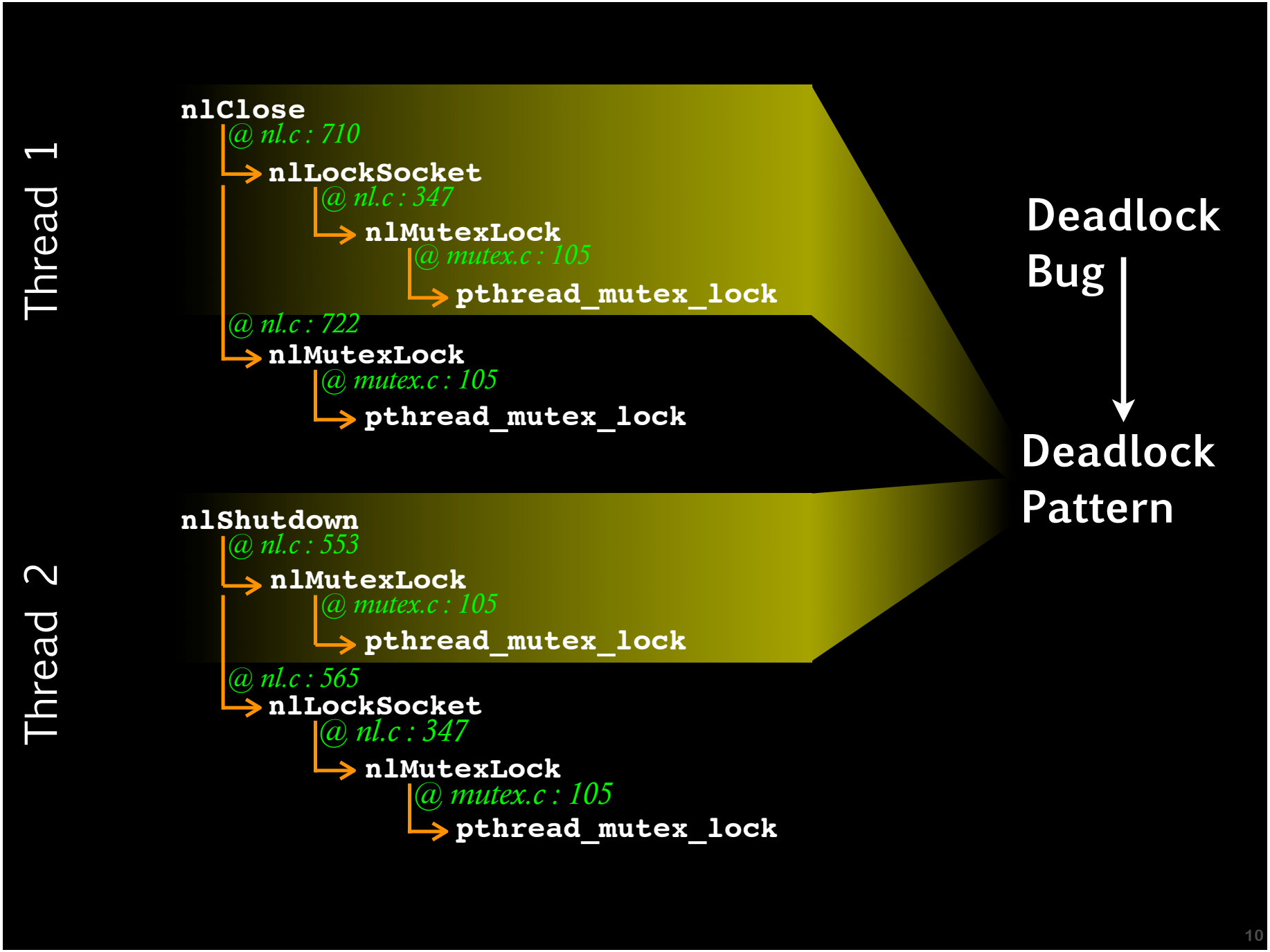
```
nlClose
    nlLockSocket
        nlMutexLock
            pthread_mutex_lock
    nlMutexLock
        pthread_mutex_lock
```

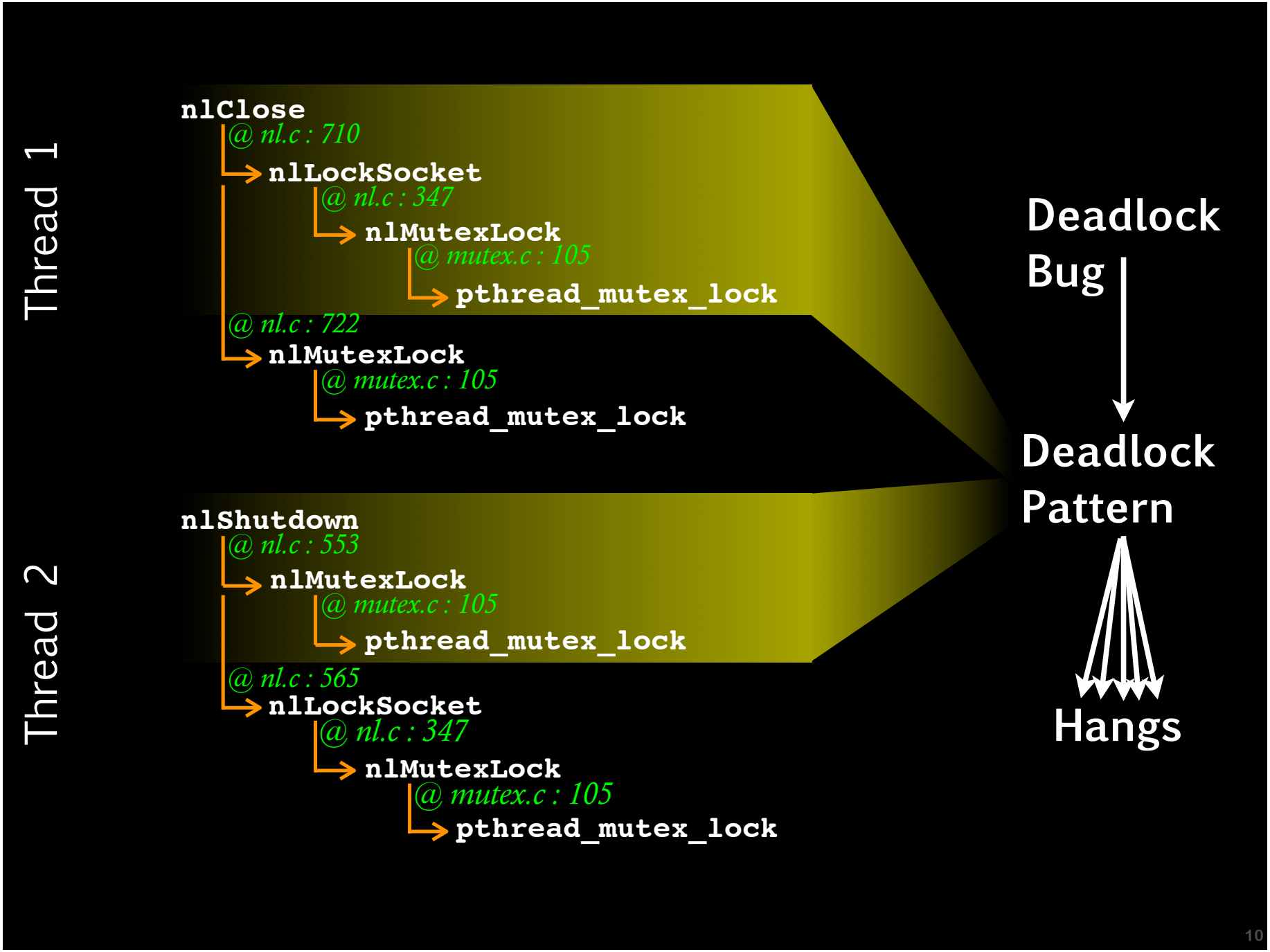
Thread 2

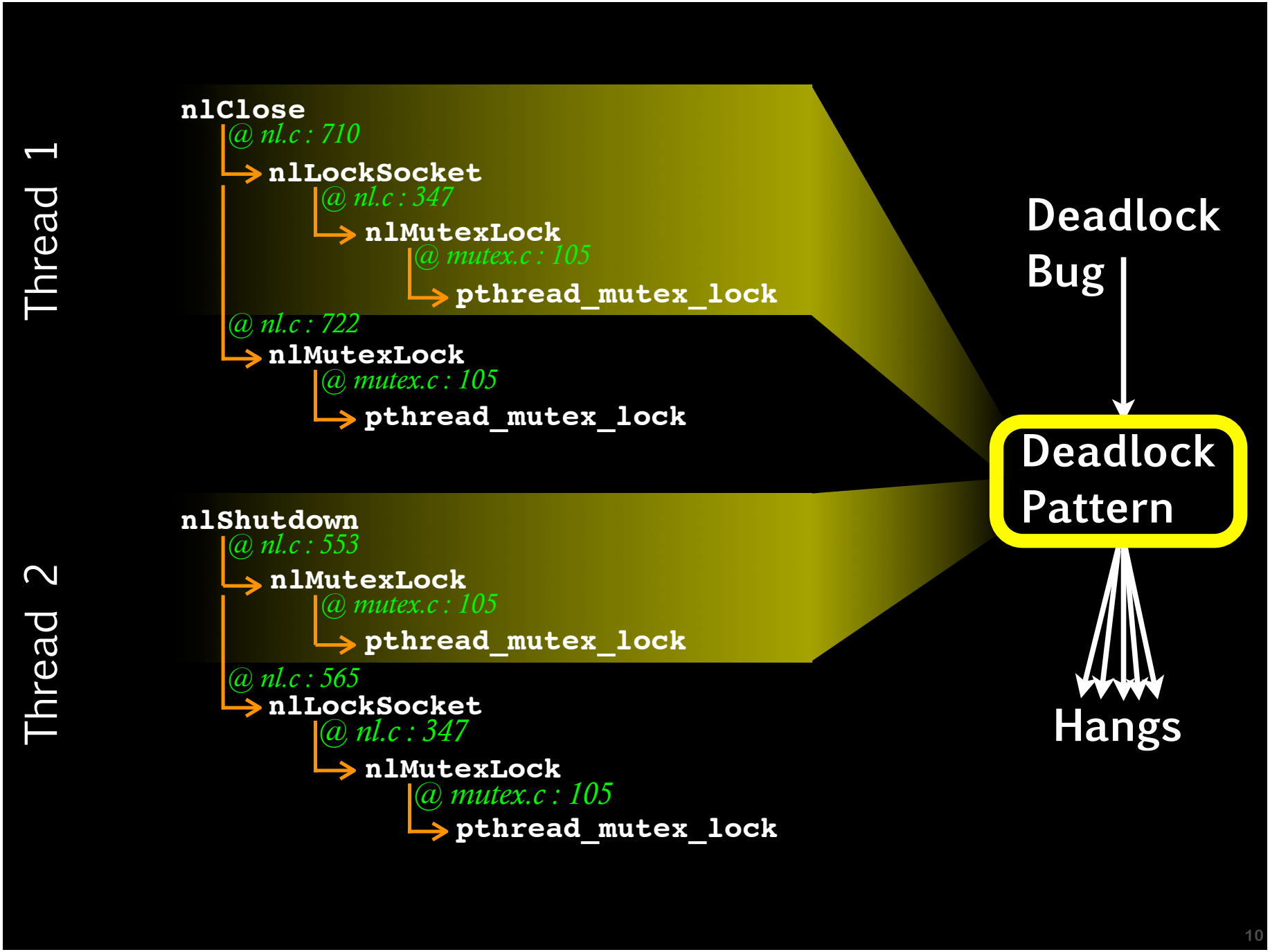
```
nlShutdown
    nlMutexLock
        pthread_mutex_lock
    nlLockSocket
        nlMutexLock
            pthread_mutex_lock
```

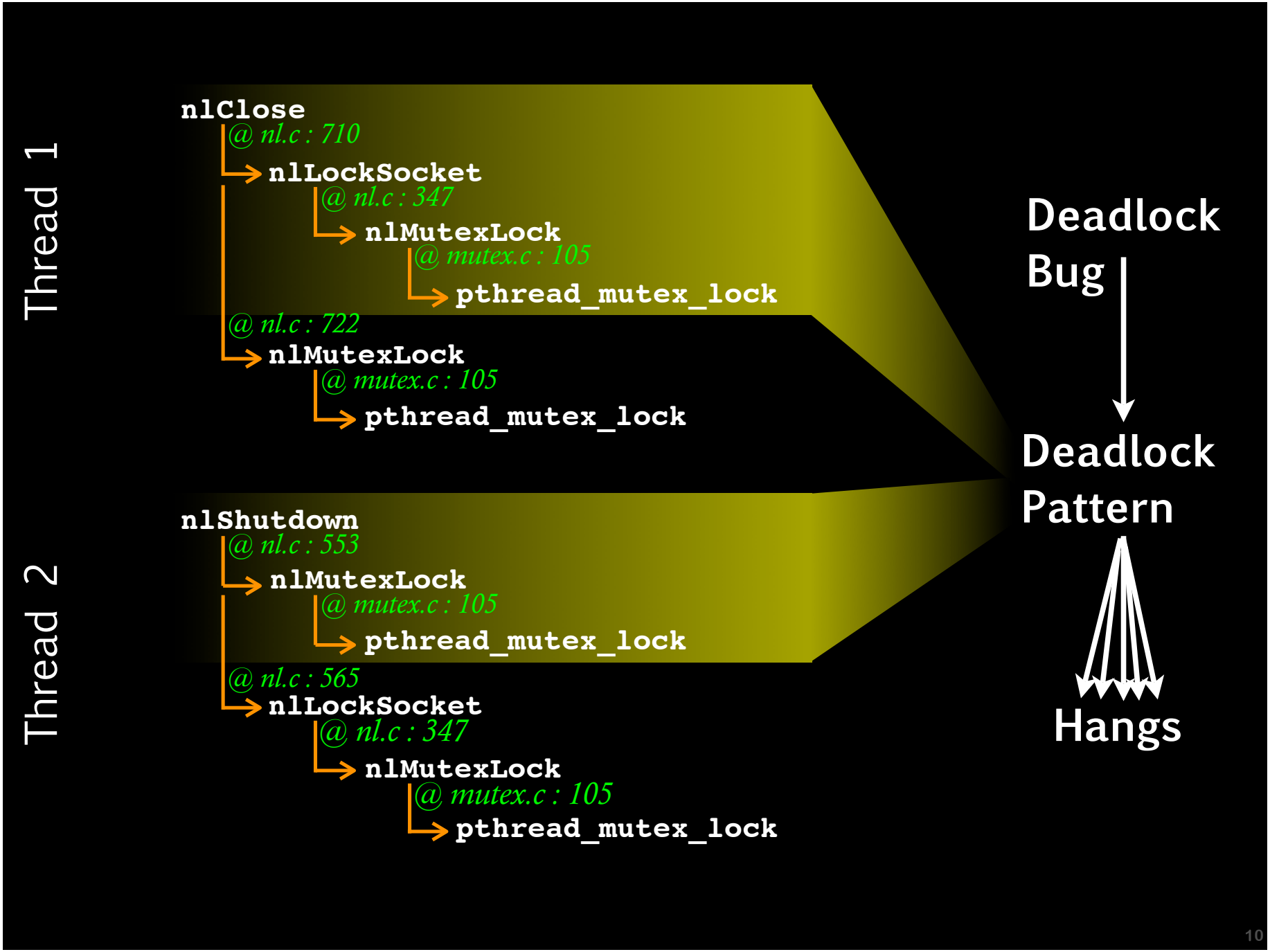

Thread 1
nlClose
nlLockSocket
nlMutexLock
pthread_mutex_lock
nlMutexLock
pthread_mutex_lock
Thread 2
nlShutdown
nlMutexLock
pthread_mutex_lock
nlLockSocket
nlMutexLock
pthread_mutex_lock
Deadlock Pattern

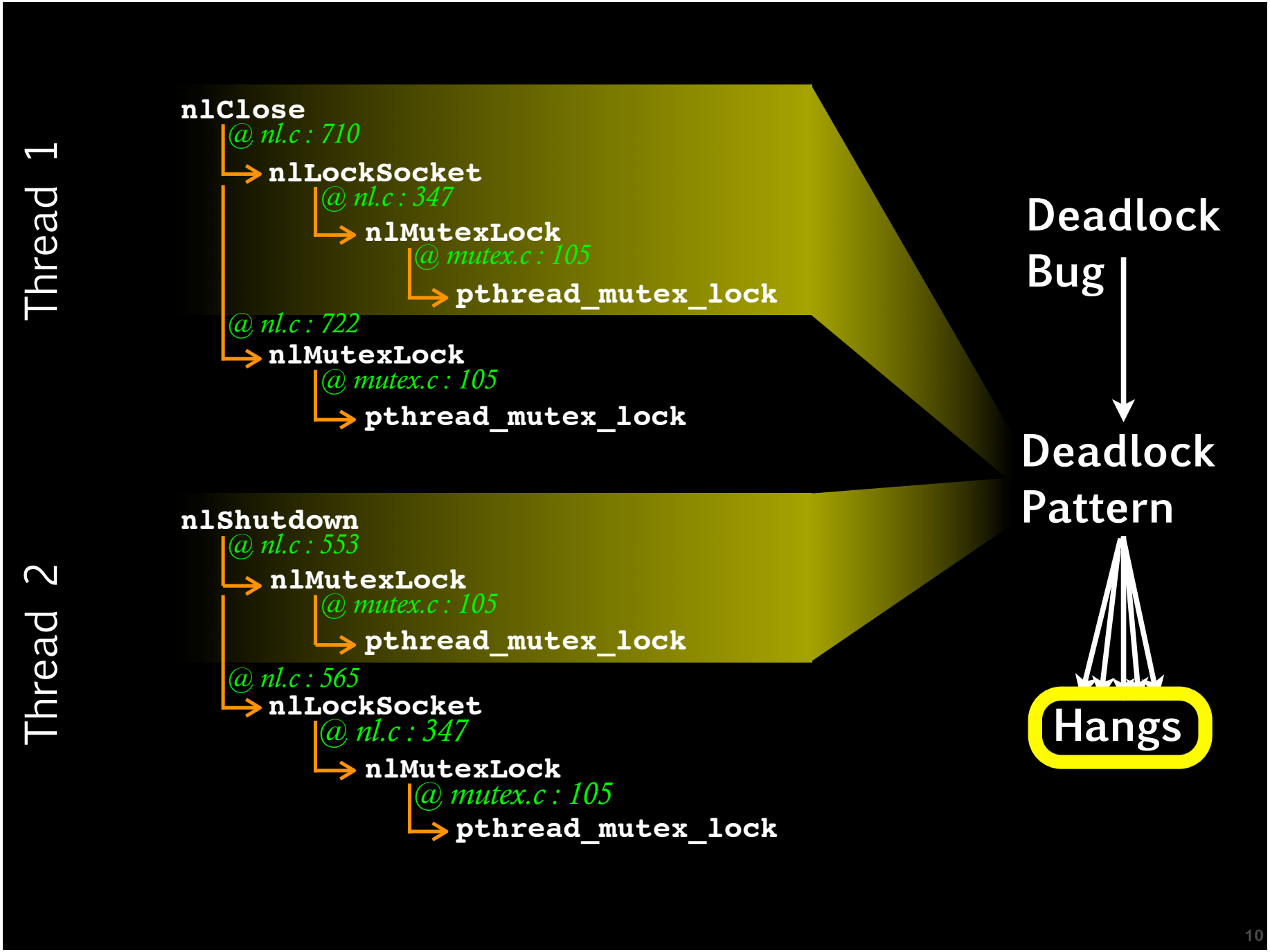

Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Pattern


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Bug
Deadlock Pattern


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Bug
Deadlock Pattern
Hangs


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Bug
Deadlock Pattern
Hangs


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Bug
Deadlock Pattern
Hangs


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
Deadlock Bug
Deadlock Pattern
Hangs

Dimmunix

- Intercepts calls to lock/unlock
- Detects deadlocks automatically
- Saves the observed deadlock pattern
- Avoids executions that match saved patterns

Dimmunix

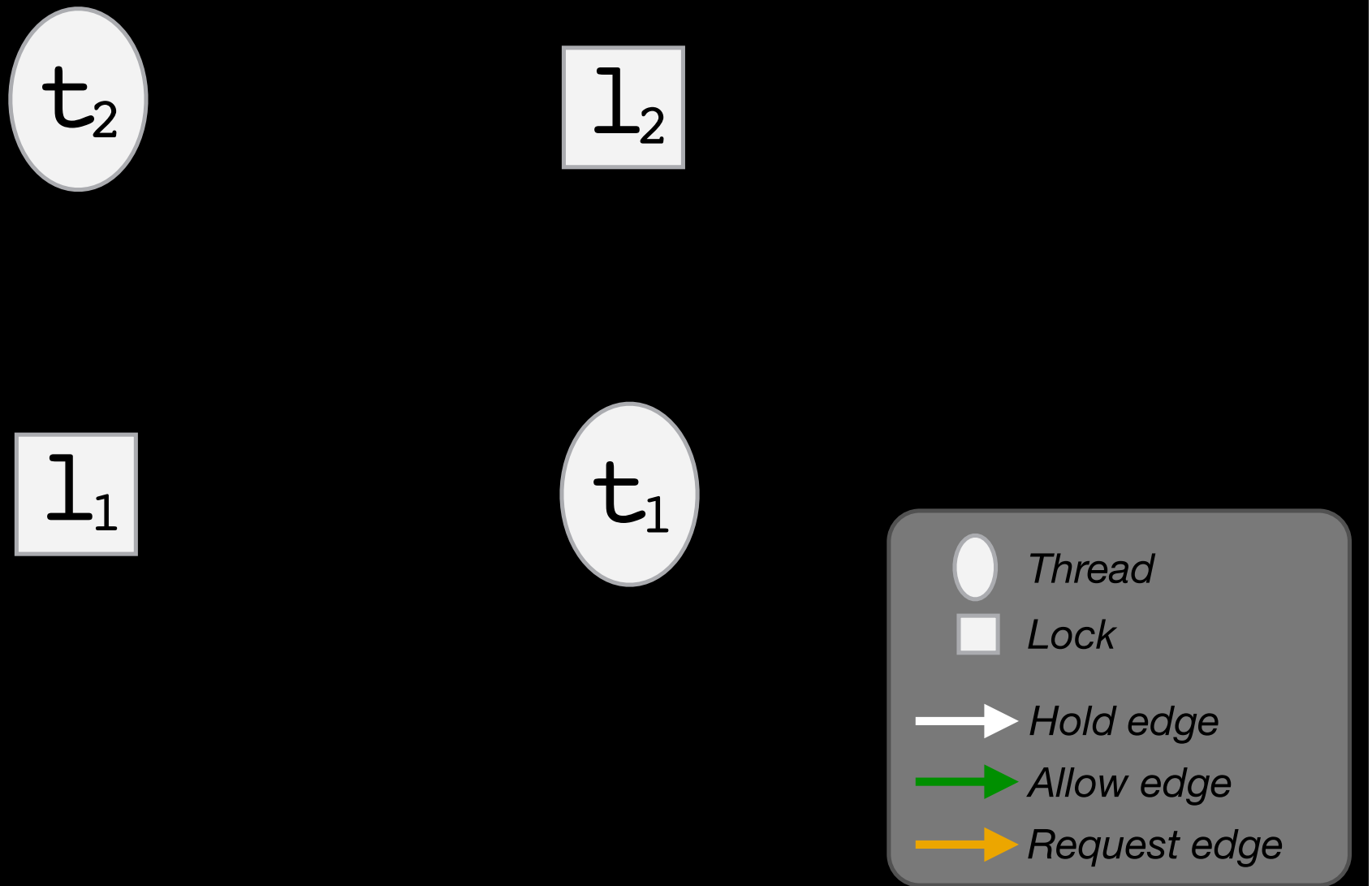
- Intercepts calls to lock/unlock
- Detects deadlocks automatically
- Saves the observed deadlock pattern
- Avoids executions that match saved patterns

History of patterns = immune system

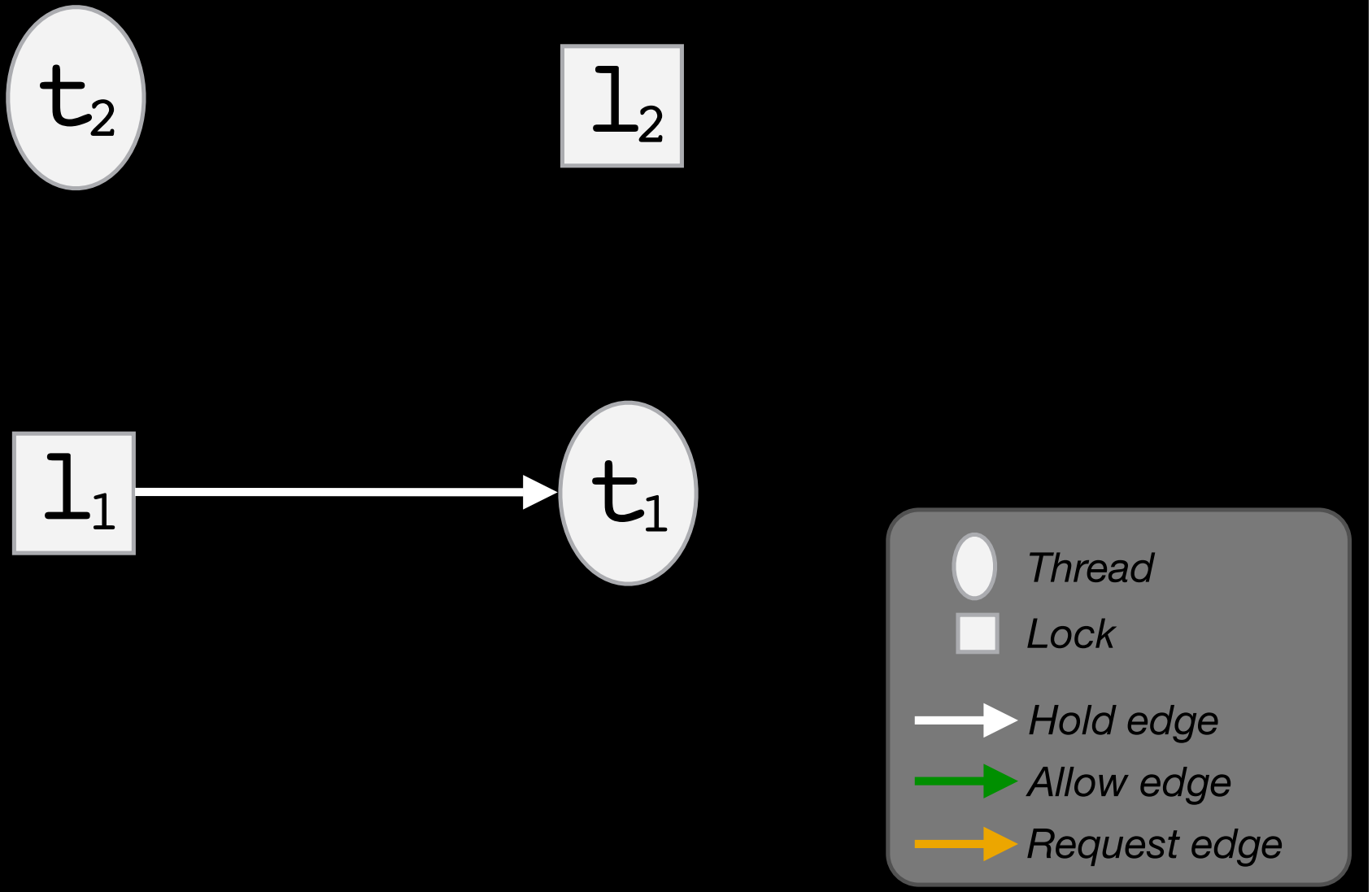
Interception

- Two versions
  - Java: Direct instrumentation
  - POSIX Threads: Modified pthreads shared library
- Benefits
  - Zero assistance from programmers
  - Zero assistance from users
  - Zero need for source code

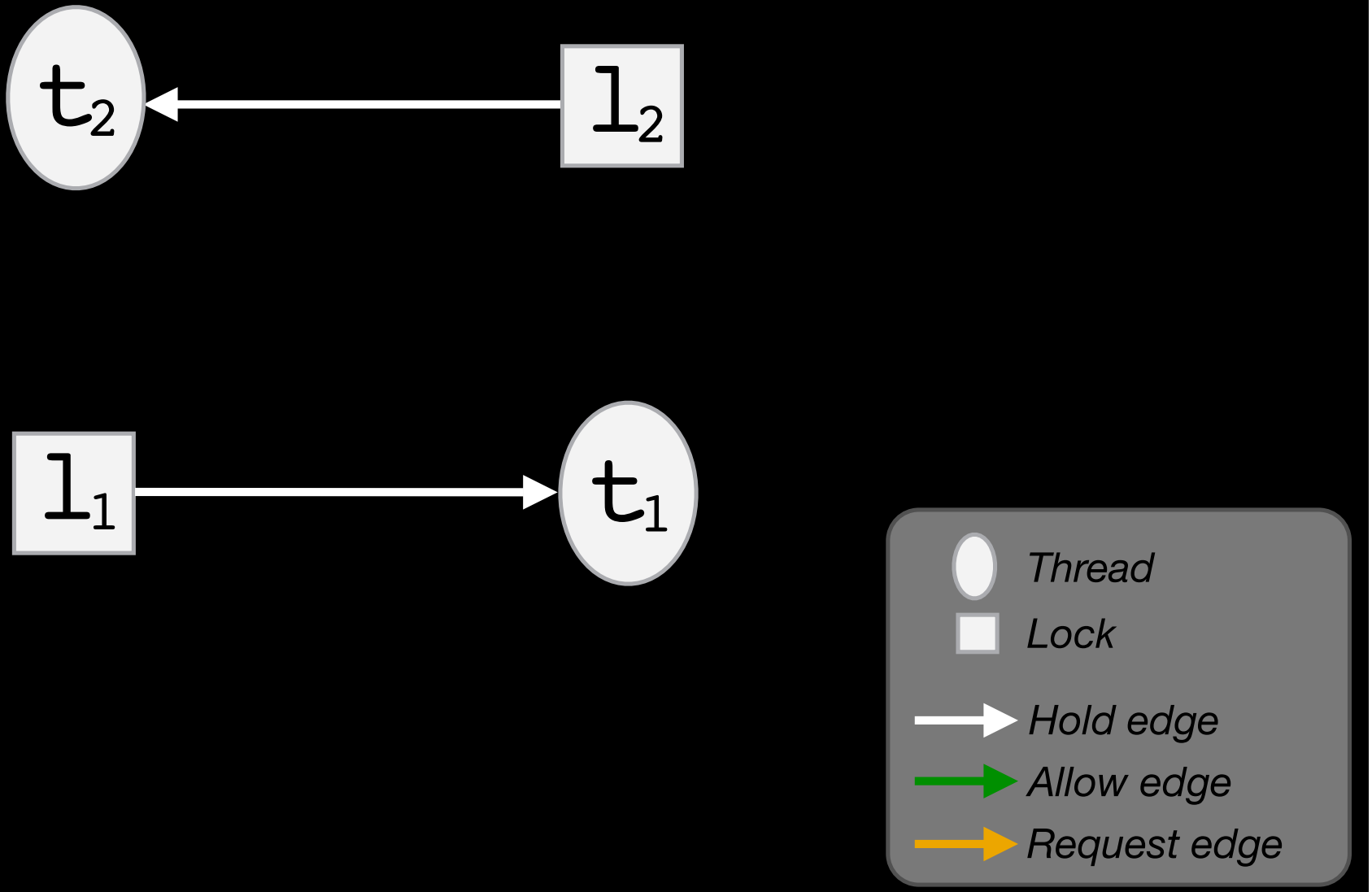
Deadlock Detection



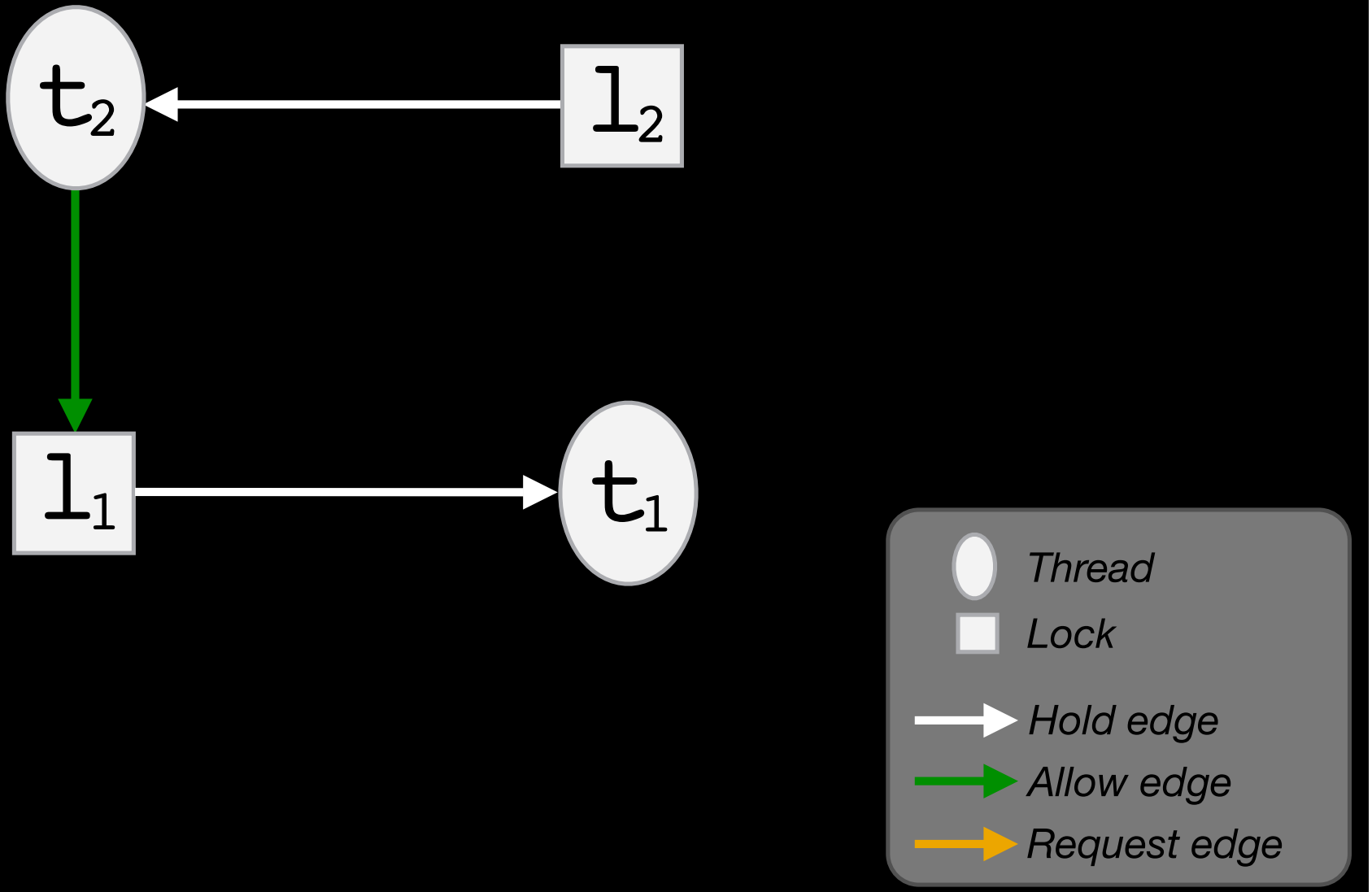
Deadlock Detection



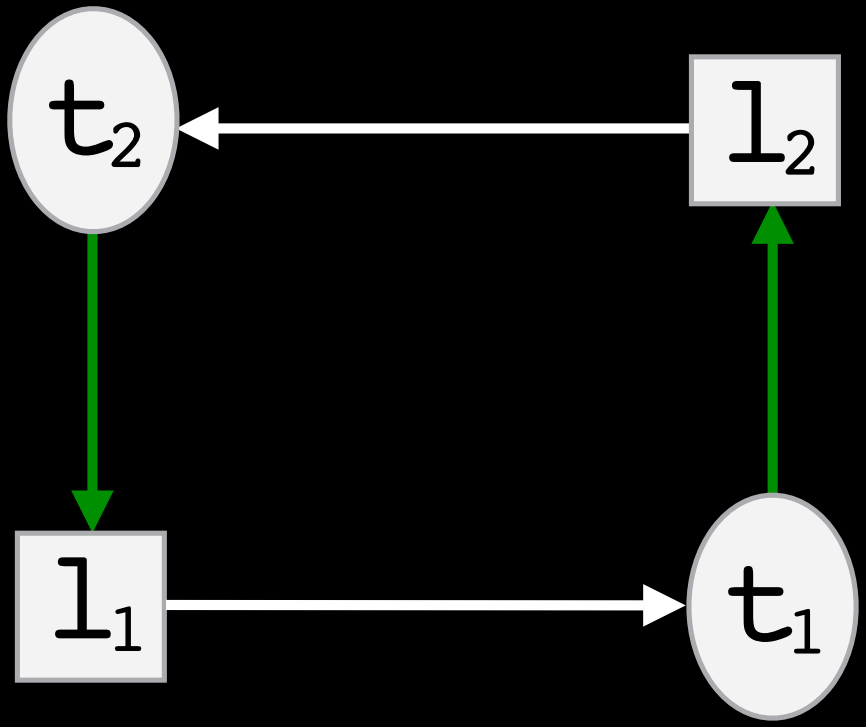
Deadlock Detection

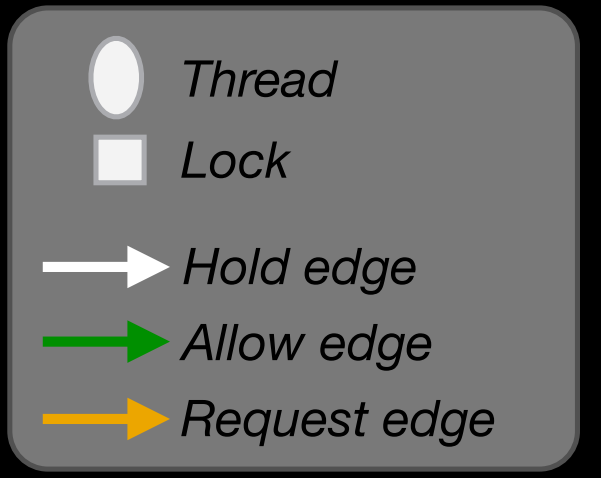


Deadlock Detection

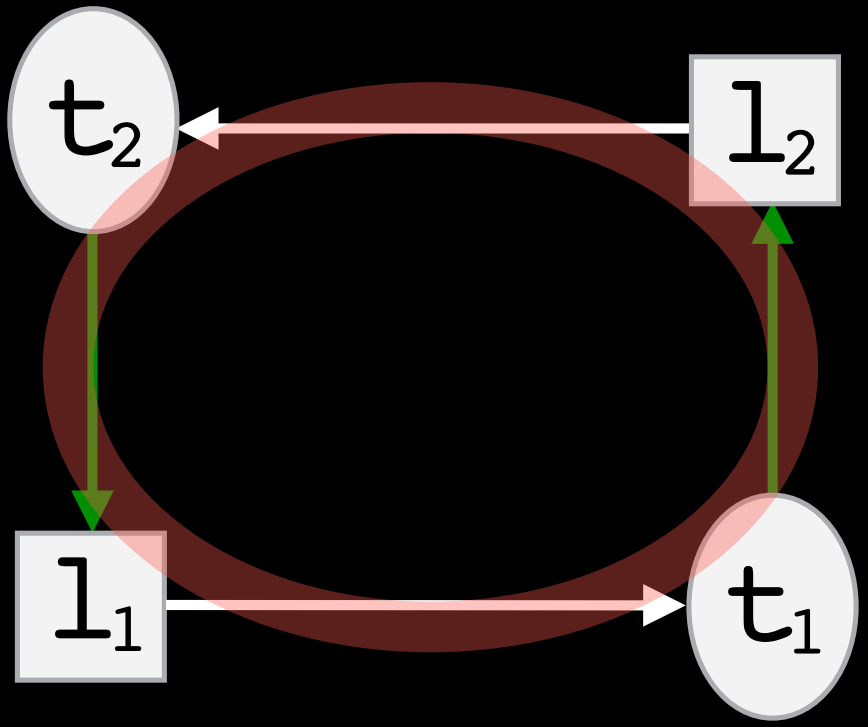


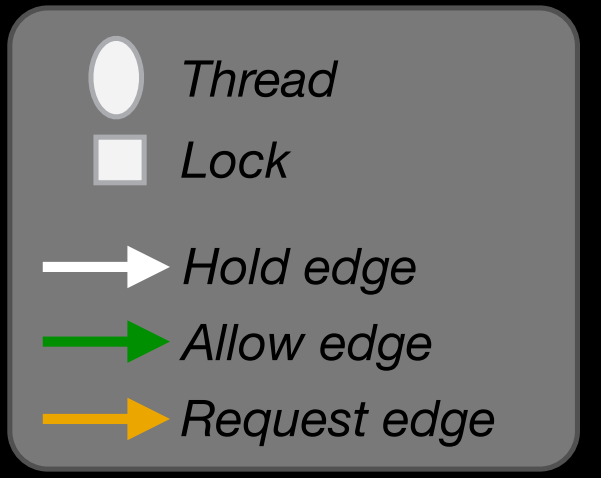
Deadlock Detection



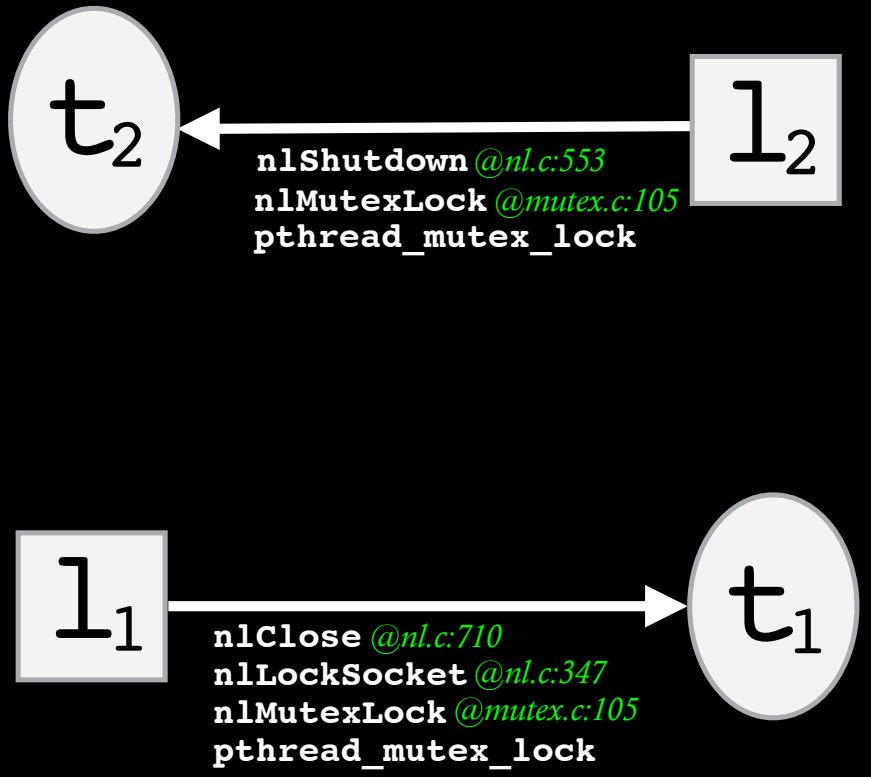


Deadlock Detection

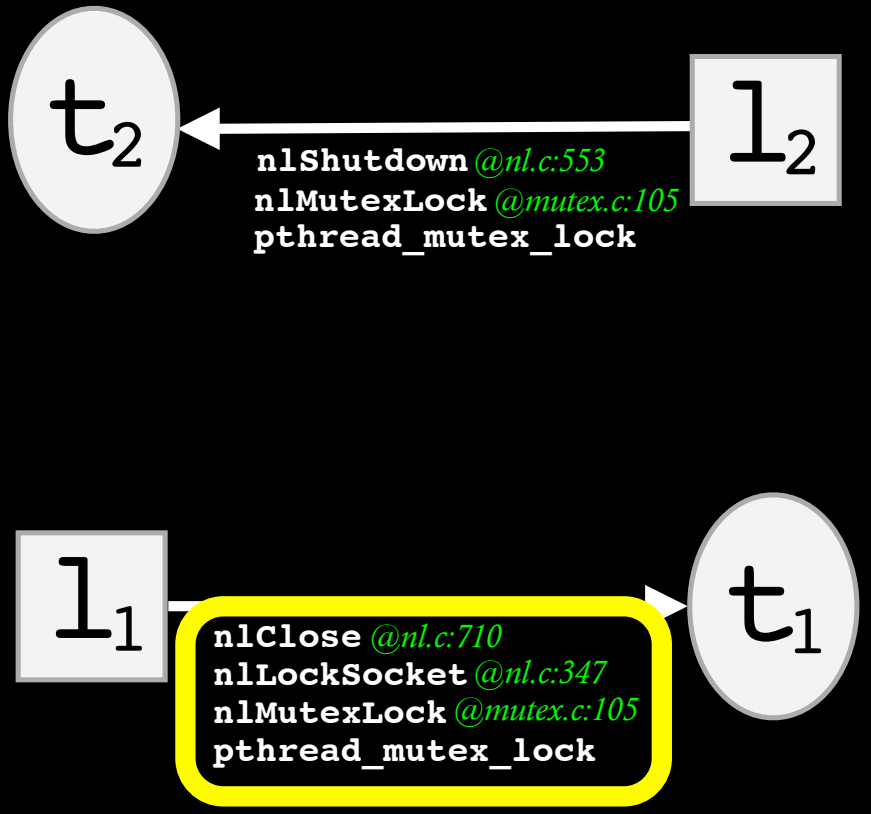




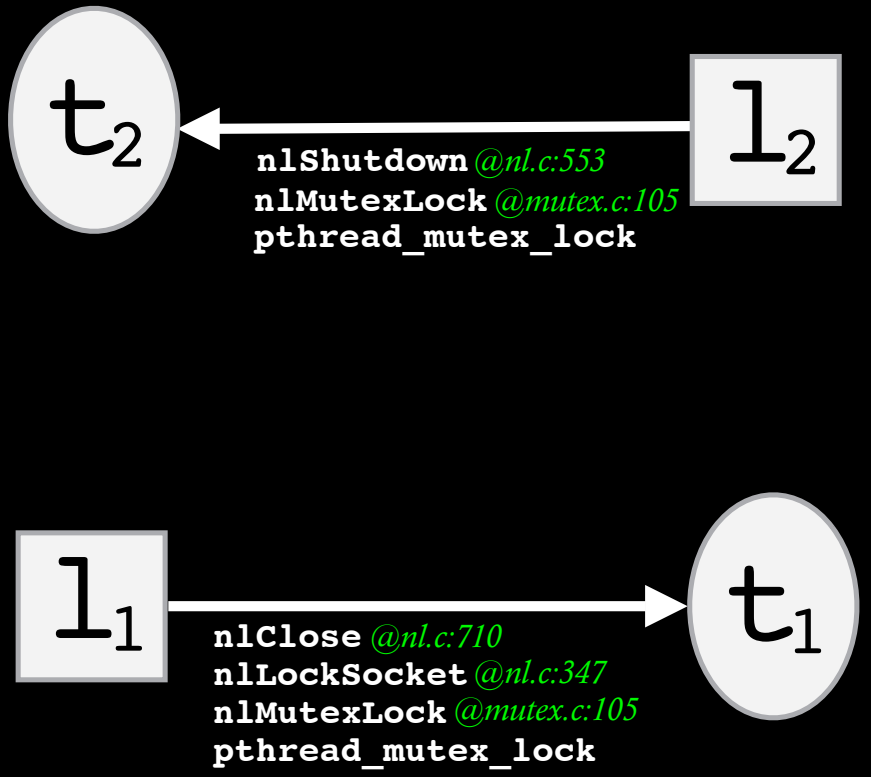
Fingerprinting Deadlocks



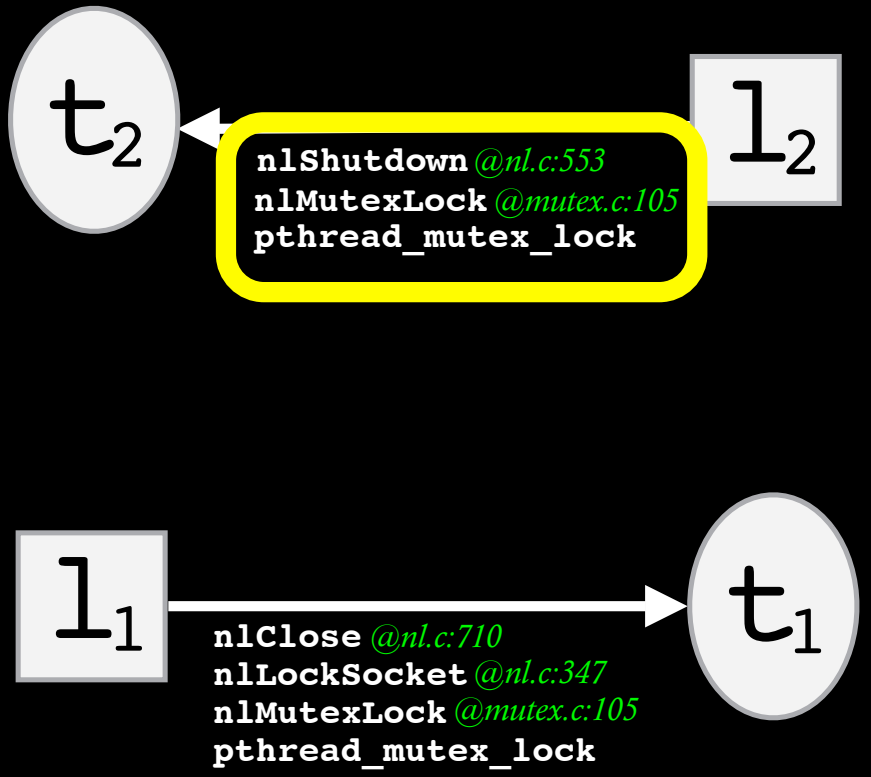
Fingerprinting Deadlocks



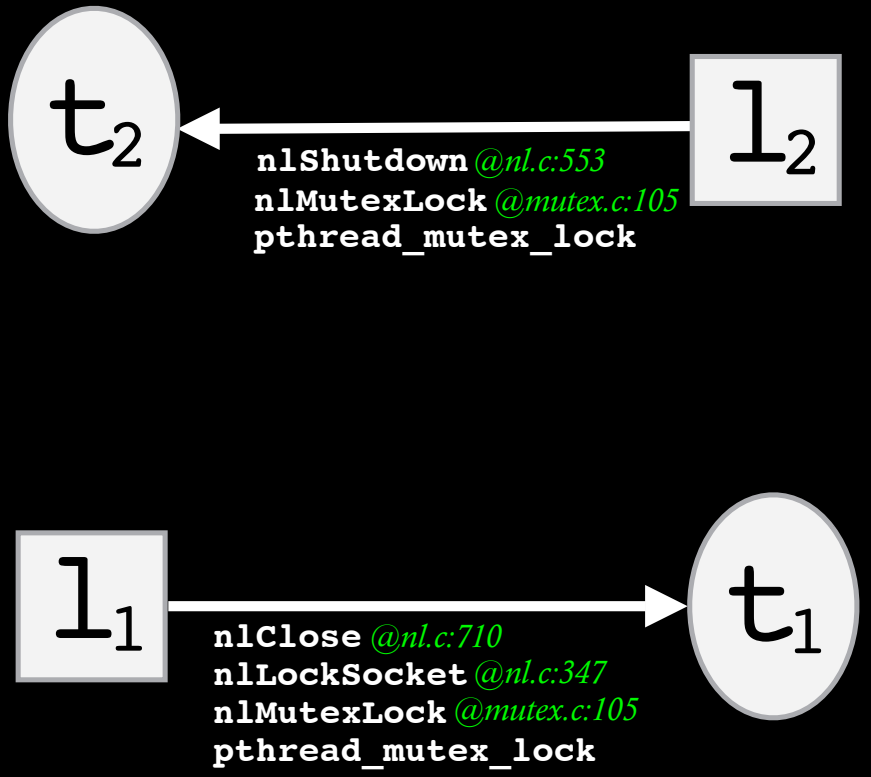
Fingerprinting Deadlocks



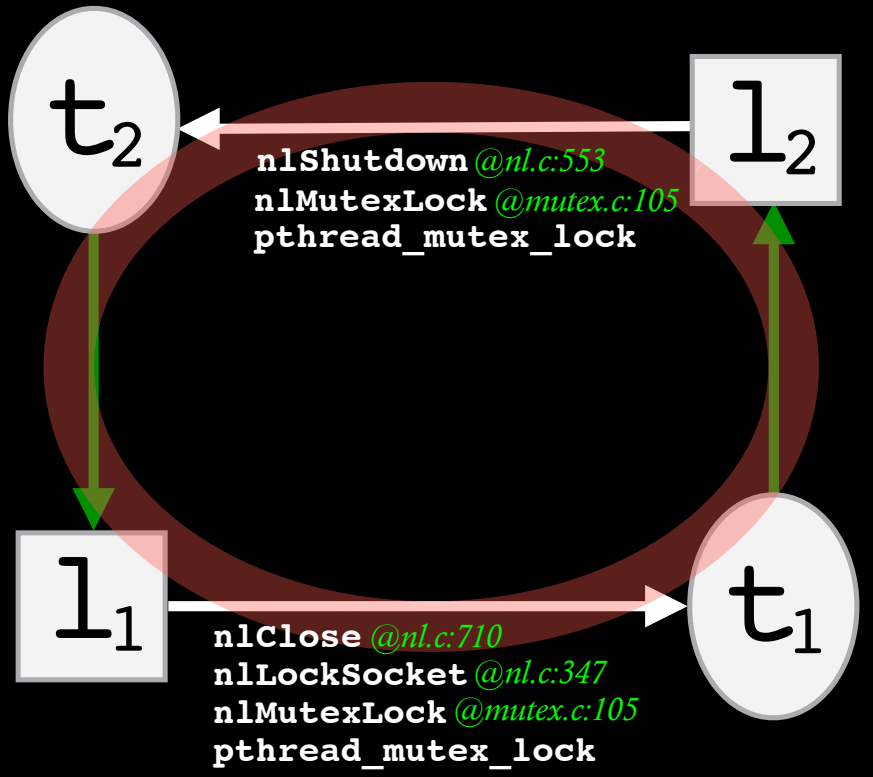
Fingerprinting Deadlocks



Fingerprinting Deadlocks



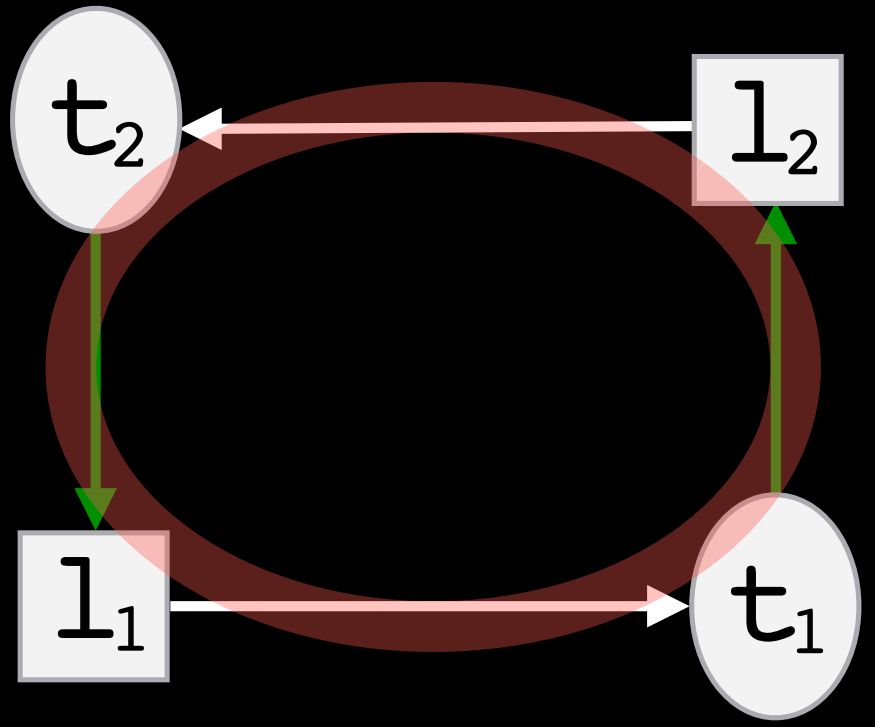
Fingerprinting Deadlocks



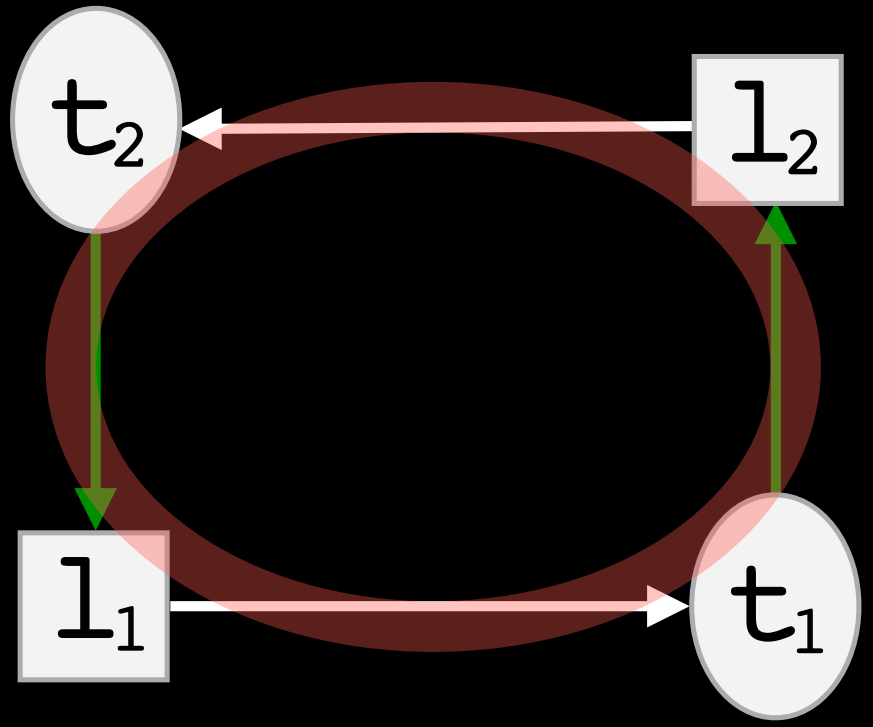
Fingerprinting Deadlocks

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```



Fingerprinting Deadlocks



Avoidance

Thread 1

nlClose

Thread 2

nlShutdown

nlShutdown *@nl.c:553*
nlMutexLock *@mutex.c:105*
pthread_mutex_lock

nlClose *@nl.c:710*
nlLockSocket *@nl.c:347*
nlMutexLock *@mutex.c:105*
pthread_mutex_lock

Avoidance

Thread 1

```
nlClose
    @ nl.c : 710
    → nlLockSocket
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Thread 2

```
nlShutdown
```

Avoidance

Thread 1

```
nlClose
    @ nl.c : 710
    → nlLockSocket
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Thread 2

```
nlShutdown
    @ nl.c : 553
    → nlMutexLock
```

Avoidance

Thread 1

```
nlClose
    @ nl.c : 710
    → nlLockSocket
        @ nl.c : 347
        → nlMutexLock
```

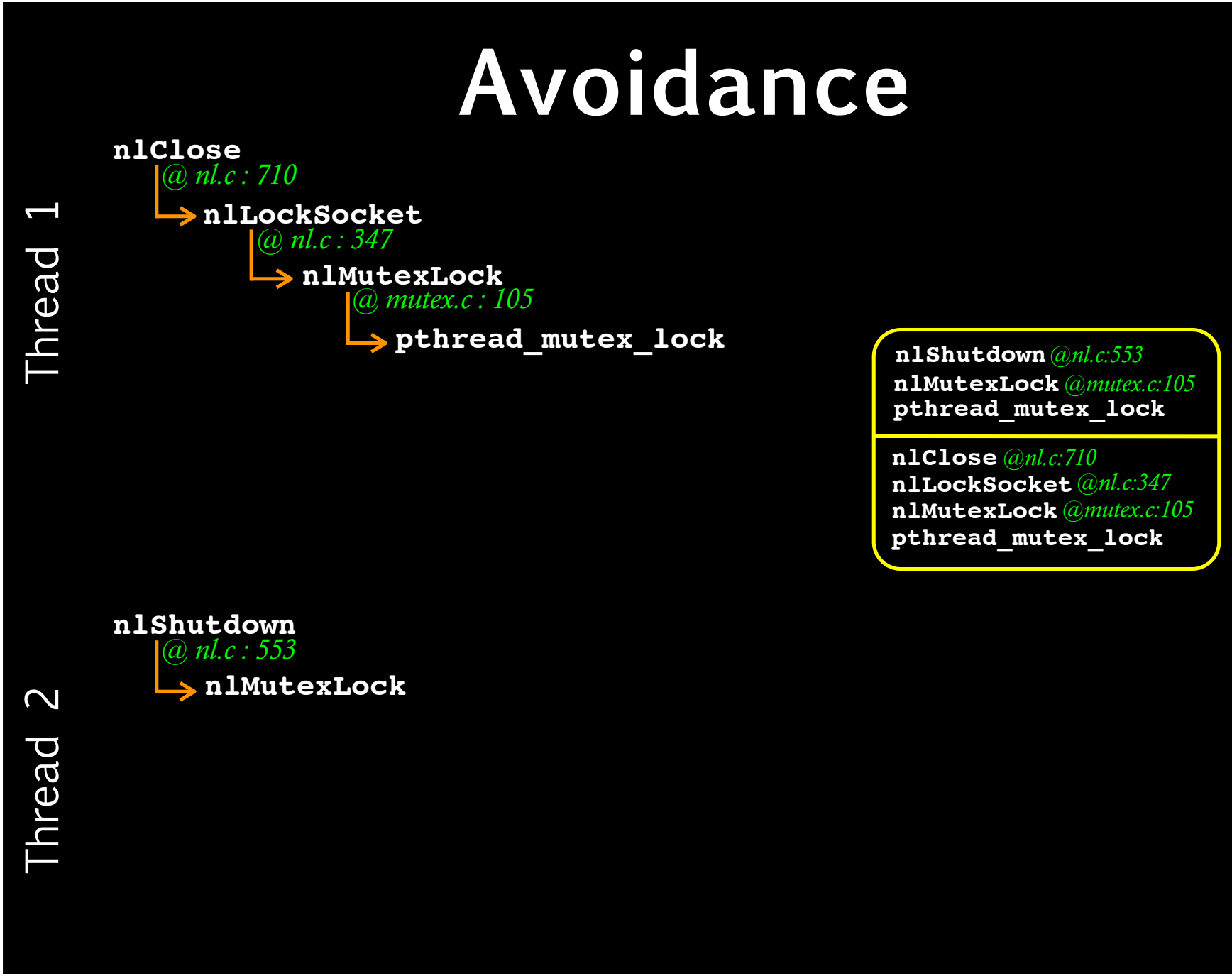
Thread 2

```
nlShutdown
    @ nl.c : 553
    → nlMutexLock
```

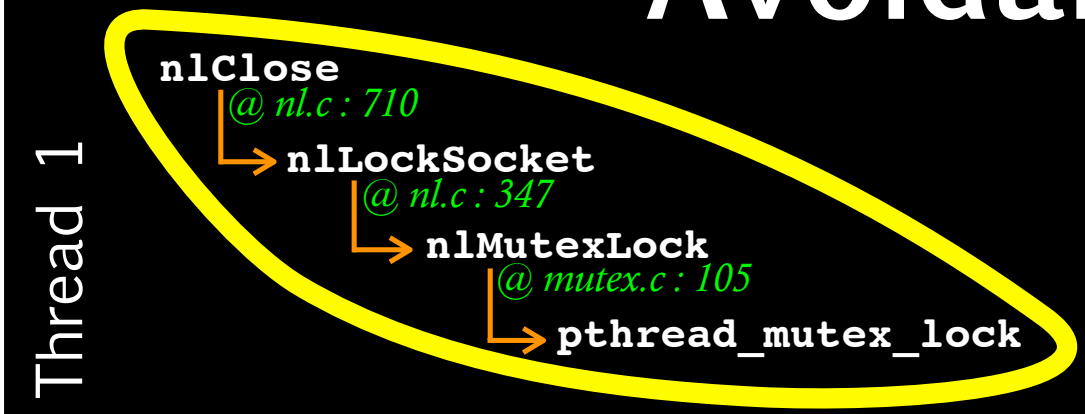
```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

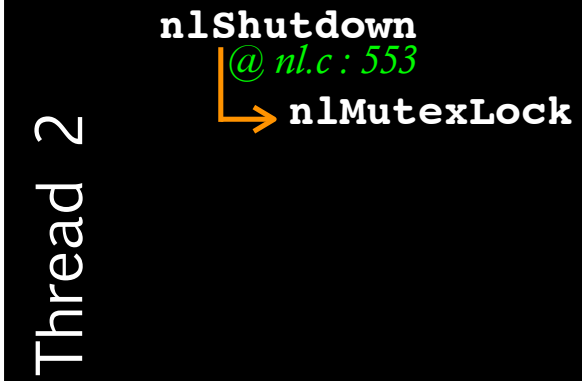
Avoidance


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock

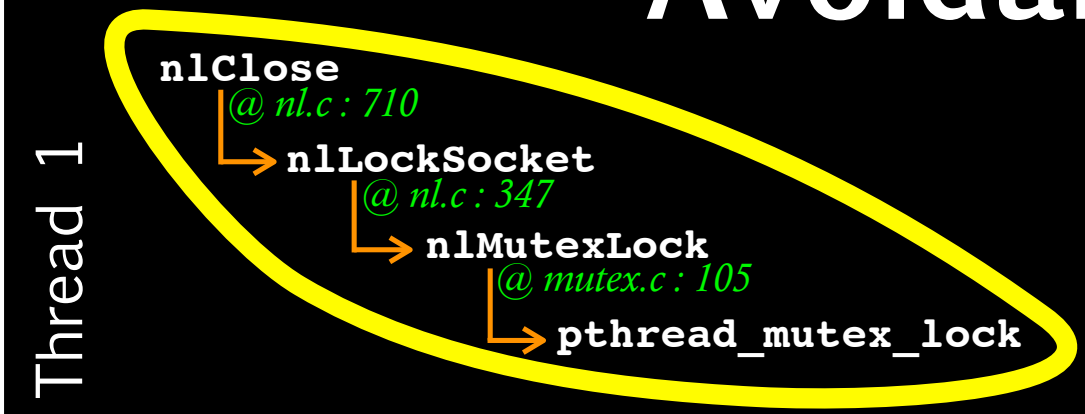
Avoidance


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock

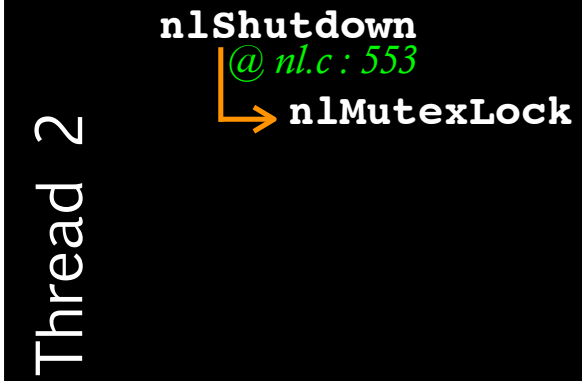

nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock


Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock

Avoidance


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock


nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock


Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
    nlLockSocket
      @ nl.c : 347
        nlMutexLock
          @ mutex.c : 105
            pthread_mutex_lock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
    nlMutexLock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
    @ nl.c : 710
    → nlLockSocket
        @ nl.c : 347
        → nlMutexLock
            @ mutex.c : 105
            → pthread_mutex_lock
    @ nl.c : 722
    → nlMutexLock
```

Thread 2

```
nlShutdown
    @ nl.c : 553
    → nlMutexLock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

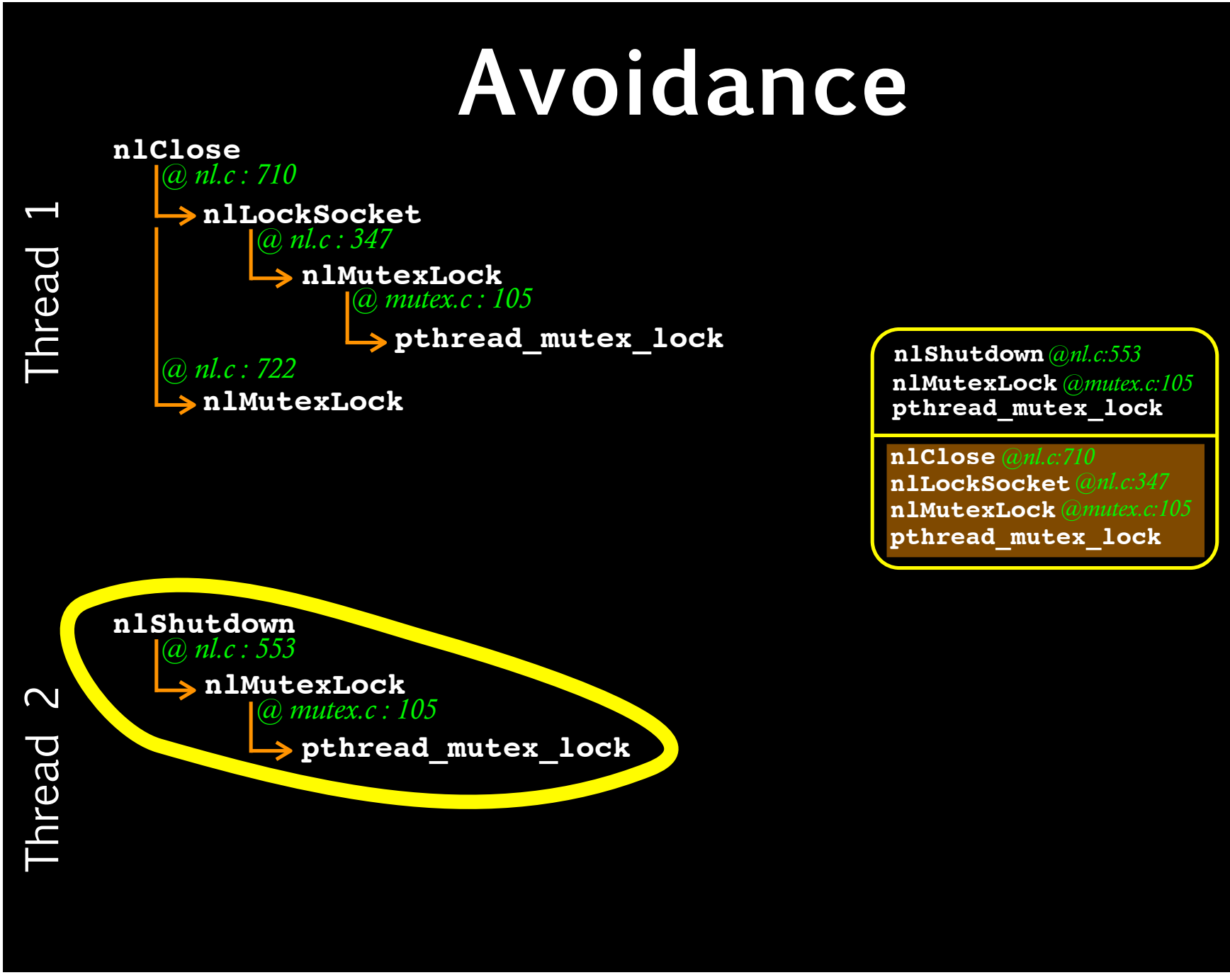
Thread 1

```
nlClose
    @ nl.c : 710
    → nlLockSocket
        @ nl.c : 347
        → nlMutexLock
            @ mutex.c : 105
            → pthread_mutex_lock
    @ nl.c : 722
    → nlMutexLock
```

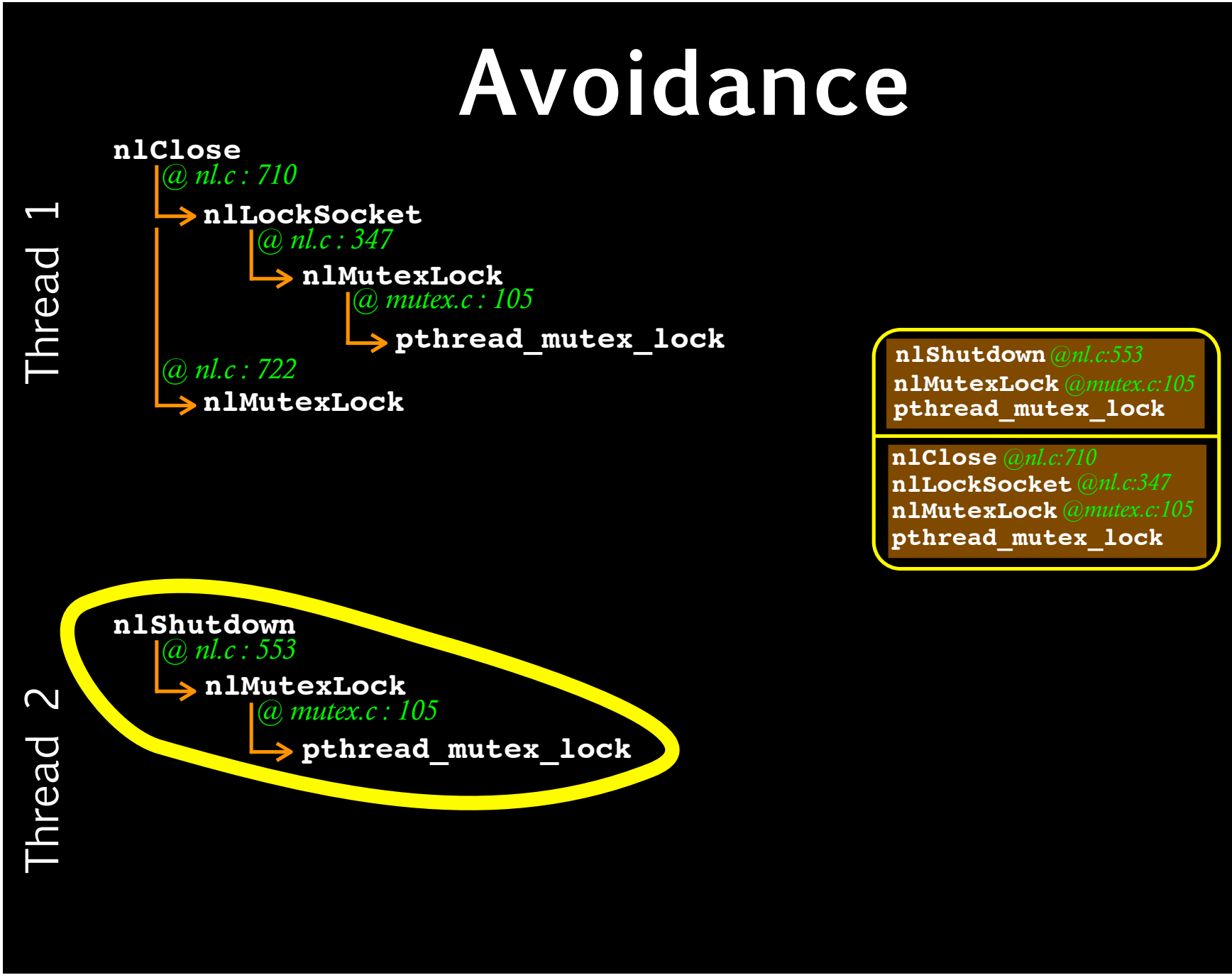
Thread 2

```
nlShutdown
    @ nl.c : 553
    → nlMutexLock
        @ mutex.c : 105
        → pthread_mutex_lock
```

Avoidance


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock

Avoidance


Thread 1
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
Thread 2
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  → nlLockSocket
      @ nl.c : 347
      → nlMutexLock
          @ mutex.c : 105
          → pthread_mutex_lock
  @ nl.c : 722
  → nlMutexLock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  → nlMutexLock
      @ mutex.c : 105
      → pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
  pthread_mutex_unlock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  → nlLockSocket
      @ nl.c : 347
      → nlMutexLock
          @ mutex.c : 105
          → pthread_mutex_lock
  @ nl.c : 722
  → nlMutexLock
      @ mutex.c : 105
      → pthread_mutex_lock
  pthread_mutex_unlock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  → nlMutexLock
      @ mutex.c : 105
      → pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
  pthread_mutex_unlock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock

  pthread_mutex_unlock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
  @ nl.c : 565
  nlLockSocket
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

```
nlClose
  @ nl.c : 710
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
      @ mutex.c : 105
      pthread_mutex_lock
  @ nl.c : 722
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
  pthread_mutex_unlock
```

Thread 2

```
nlShutdown
  @ nl.c : 553
  nlMutexLock
    @ mutex.c : 105
    pthread_mutex_lock
  @ nl.c : 565
  nlLockSocket
    @ nl.c : 347
    nlMutexLock
```

```
nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

```
nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
```

Avoidance

Thread 1

nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
pthread_mutex_unlock

nlShutdown @nl.c:553
nlMutexLock @mutex.c:105
pthread_mutex_lock

nlClose @nl.c:710
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock

Thread 2

nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock

Does It Work ?

- C/C++
  - MySQL
  - SQLite
  - HawkNL
- Java
  - Apache ActiveMQ
  - Limewire
  - MySQL JDBC
  - Sun JDK

System	Bug#	Deadlock Bug Description
MySQL 6.0.4	37080	INSERT and TRUNCATE in two different threads
SQLite 3.3.0	1672	Race condition in custom recursive lock implementation
HawkNL 1.6b3	n/a	nlShutdown() called concurrently with nlClose()
Limewire 4.17.9	1449	Cancel HsqlDB TaskQueue concurrently with program shutdown()
ActiveMQ 3.1	336	Concurrently create listener and actively dispatch messages to consumer
ActiveMQ 4.0	575	Concurrent invocation of Queue.dropEvent() and PrefetchSubscription.add()
MySQL 5.0 JDBC	2147	Connection.close() called concurrently with PreparedStatement.getWarnings()
	14972	Connection.prepareStatement() called concurrently with Statement.close()
	31136	PreparedStatement.executeQuery() called concurrently w/ Connection.close()
	17709	Statement.executeQuery() concurrent with Connection.prepareStatement()

System	Bug#	Deadlock Bug Description
MySQL 6.0.4	37080	INSERT and TRUNCATE in two different threads
SQLite 3.3.0	1672	Race condition in custom recursive lock implementation
HawkNL 1.6b3	n/a	nlShutdown() called concurrently with nlClose()
Limewire 4.17.9	1449	Cancel HsqlDB TaskQueue concurrently with program shutdown()
ActiveMQ 3.1	336	Concurrently create listener and actively dispatch messages to consumer
ActiveMQ 4.0	575	Concurrent invocation of Queue.dropEvent() and PrefetchSubscription.add()
MySQL 5.0 JDBC	2147	Connection.close() called concurrently with PreparedStatement.getWarnings()
	14972	Connection.prepareStatement() called concurrently with Statement.close()
	31136	PreparedStatement.executeQuery() called concurrently w/ Connection.close()
	17709	Statement.executeQuery() concurrent with Connection.prepareStatement()

Avoids real deadlocks in real systems

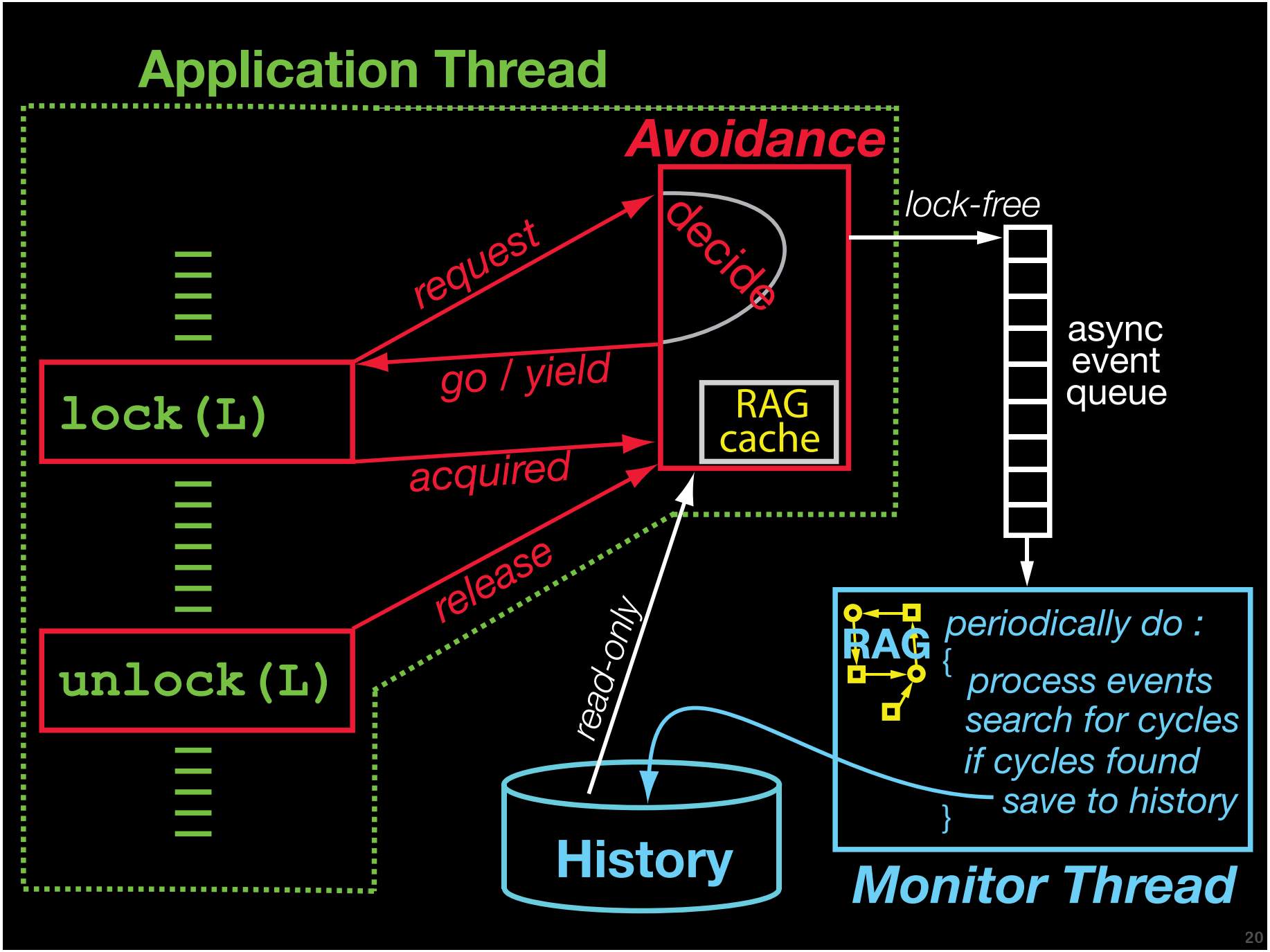
Challenges

1. Performance
2. Induced Starvation
3. Precision Calibration

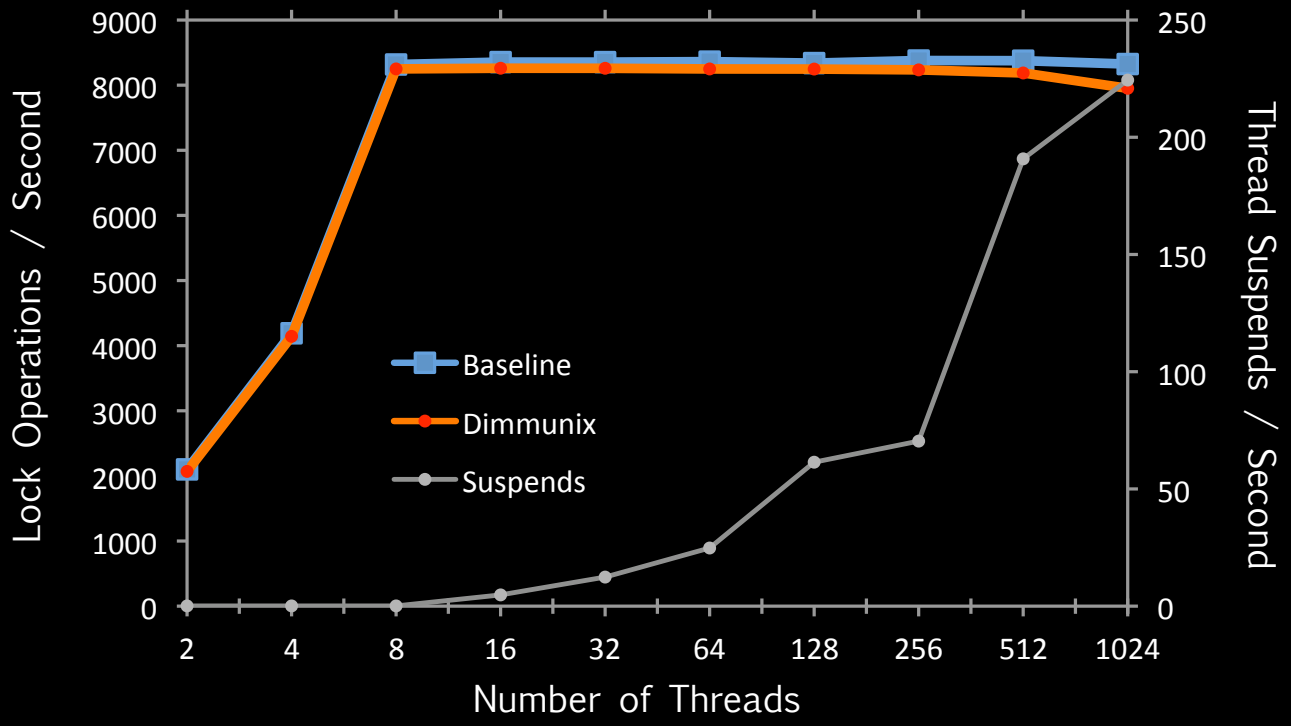
- *See paper for...*
  - Weak vs. strong immunity
  - Software upgrades
  - Evaluation of false positives

1. Performance

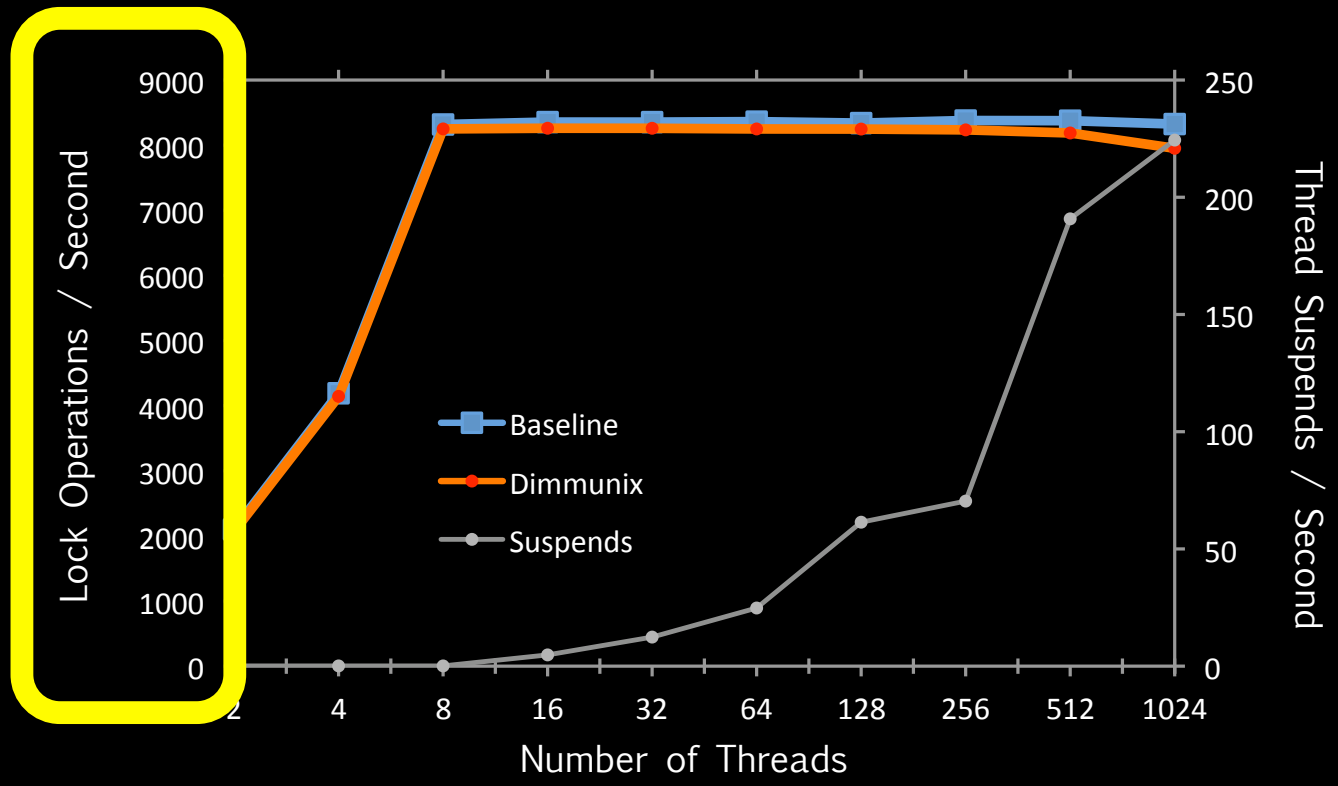
- Potential overhead
  - Intercept every call to lock/unlock
  - Update resource allocation graph
  - Find cycles + make suspend/resume decisions
- Solution: asynchronous architecture
  - Do expensive work in a separate thread
  - Take advantage of multi-core CPU
  - Lock-free data structures


Application Thread
Avoidance
request
decide
go / yield
lock(L)
RAG
cache
acquired
lock-free
async
event
queue
release
unlock(L)
read-only
RAG
periodically do :
{
process events
search for cycles
if cycles found
save to history
}
History
Monitor Thread

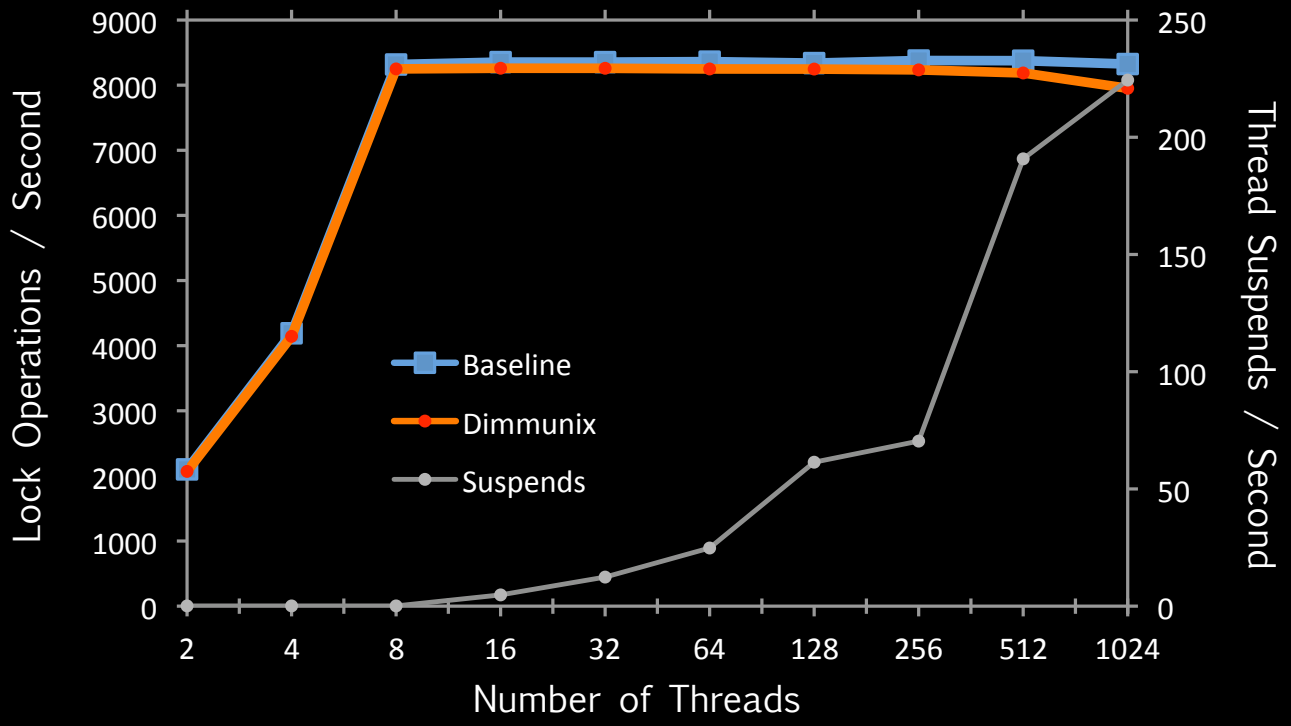
1. Performance



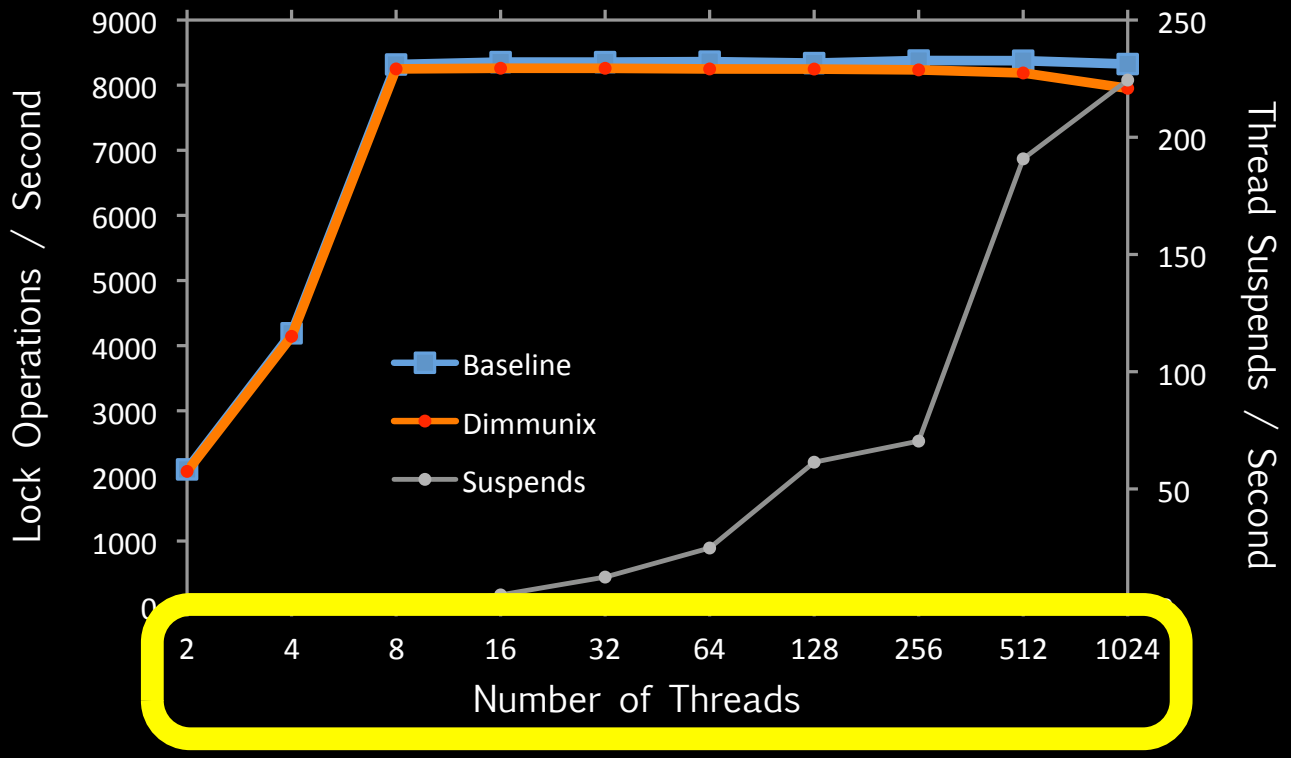
1. Performance



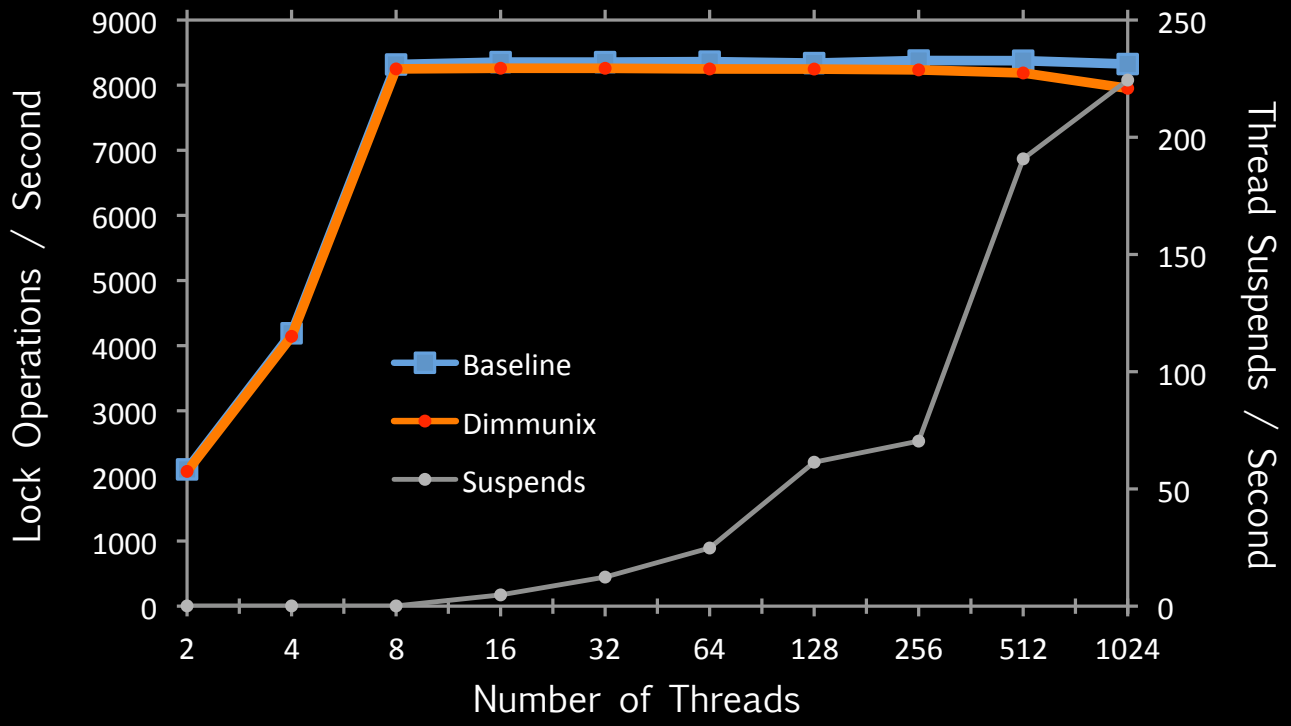
1. Performance



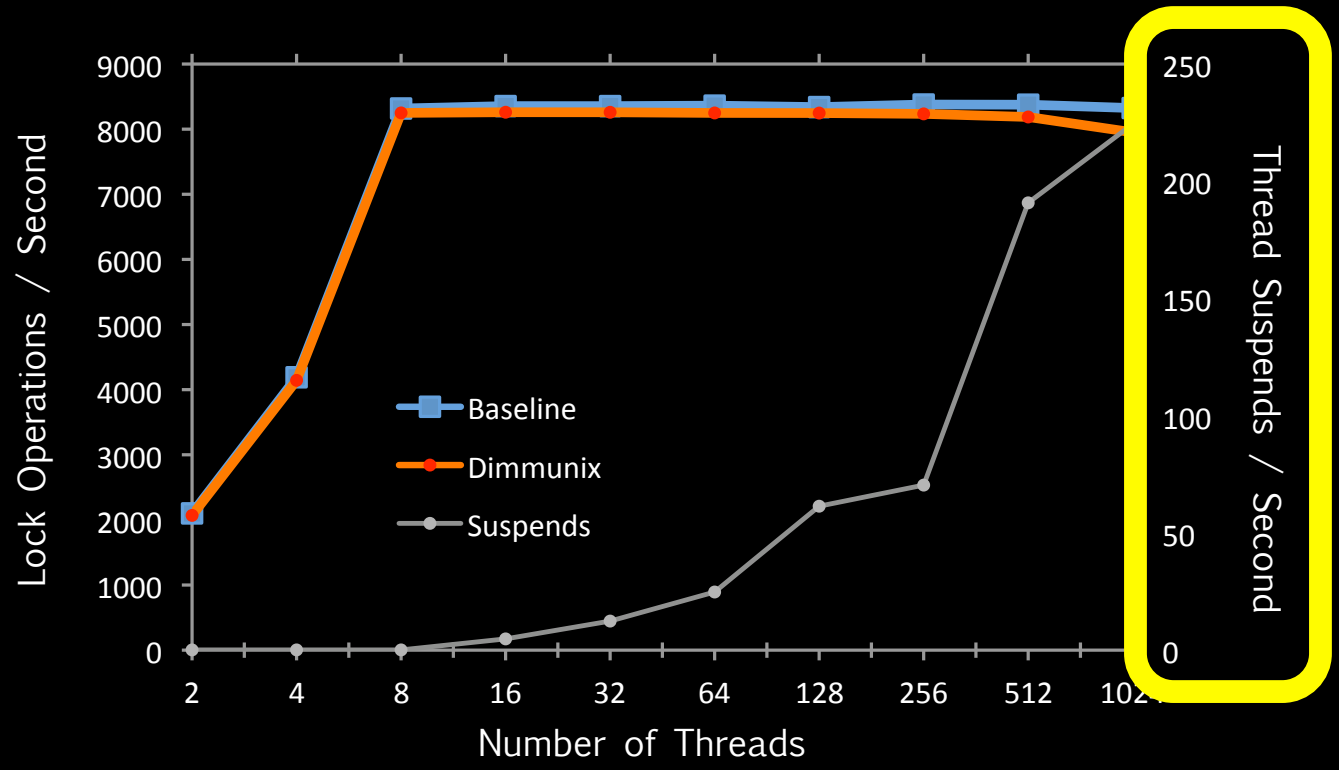
1. Performance



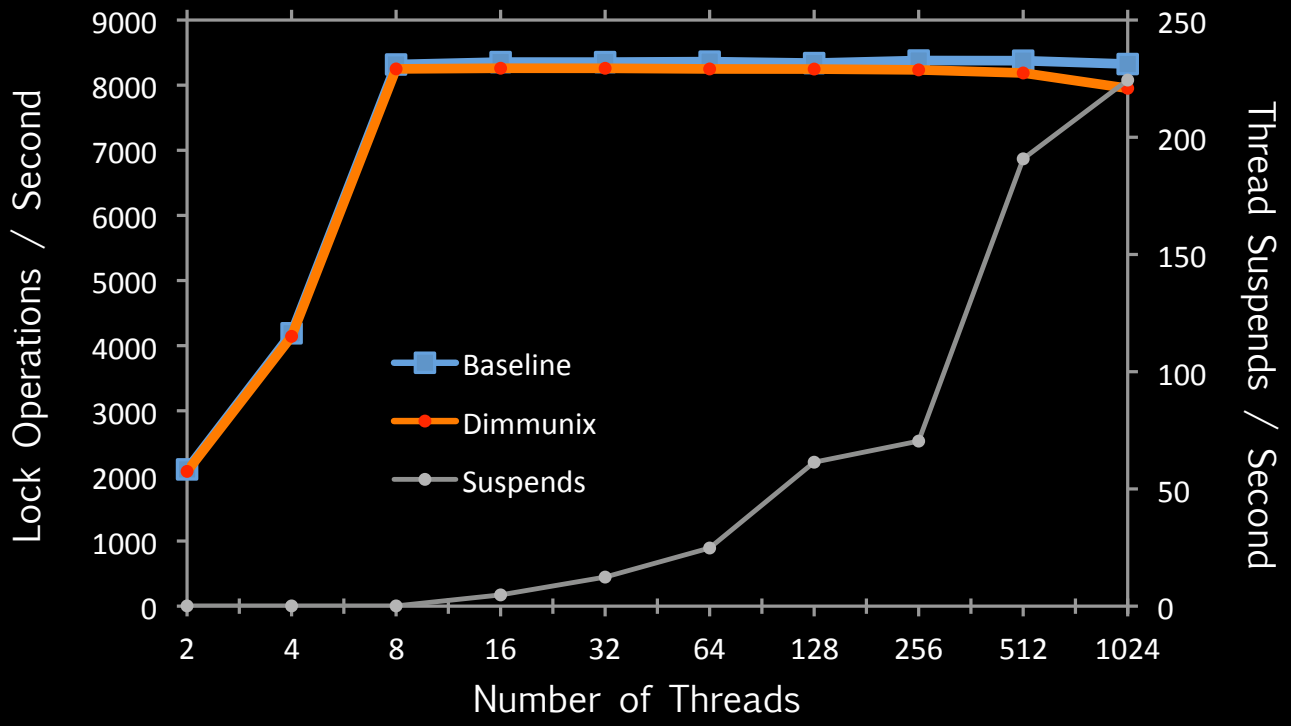
1. Performance



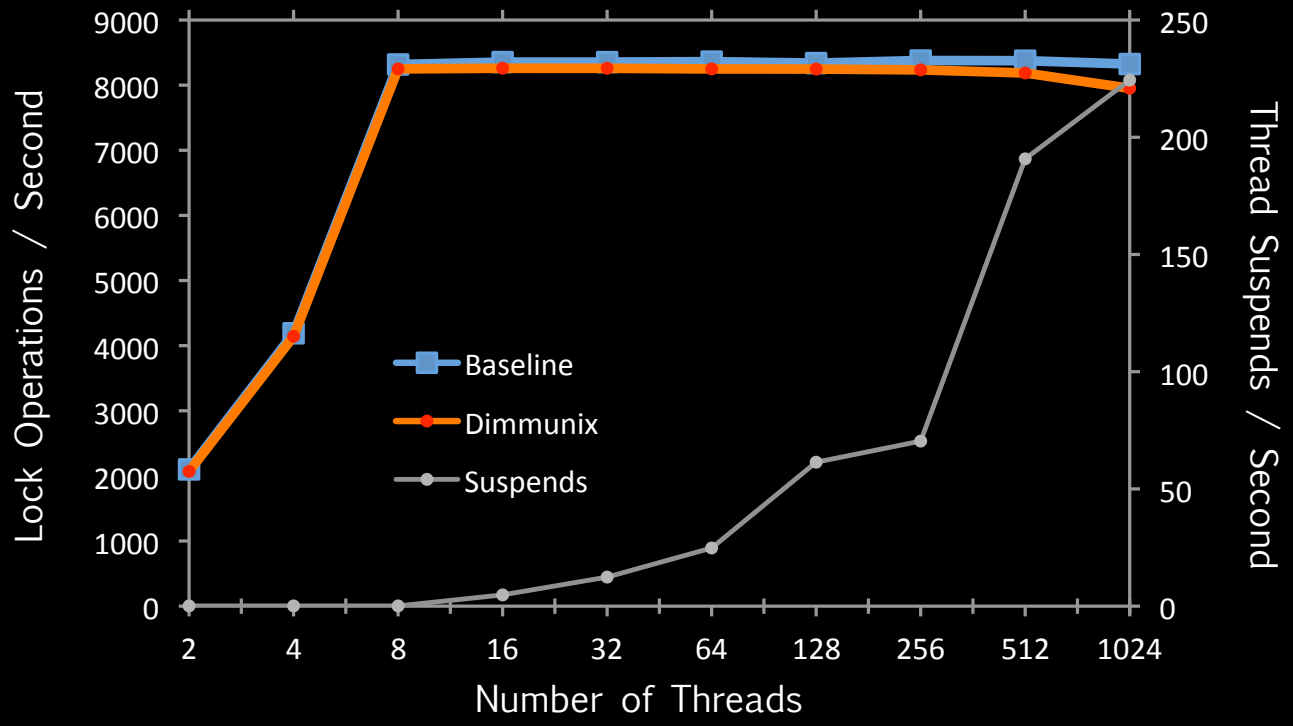
1. Performance



1. Performance



1. Performance



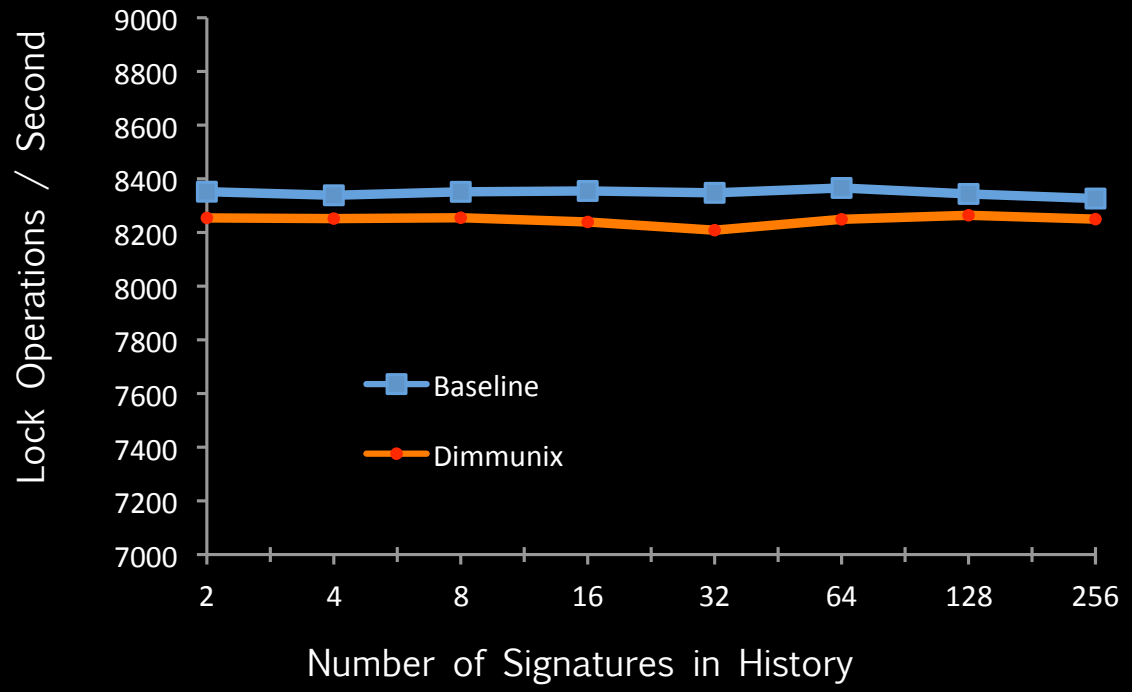
Overhead = **0.6% - 4.5%**

1. Performance

- Matching signatures from history
  - Hash call stacks and index into precomputed tables
  - Thread-local cache of data structures

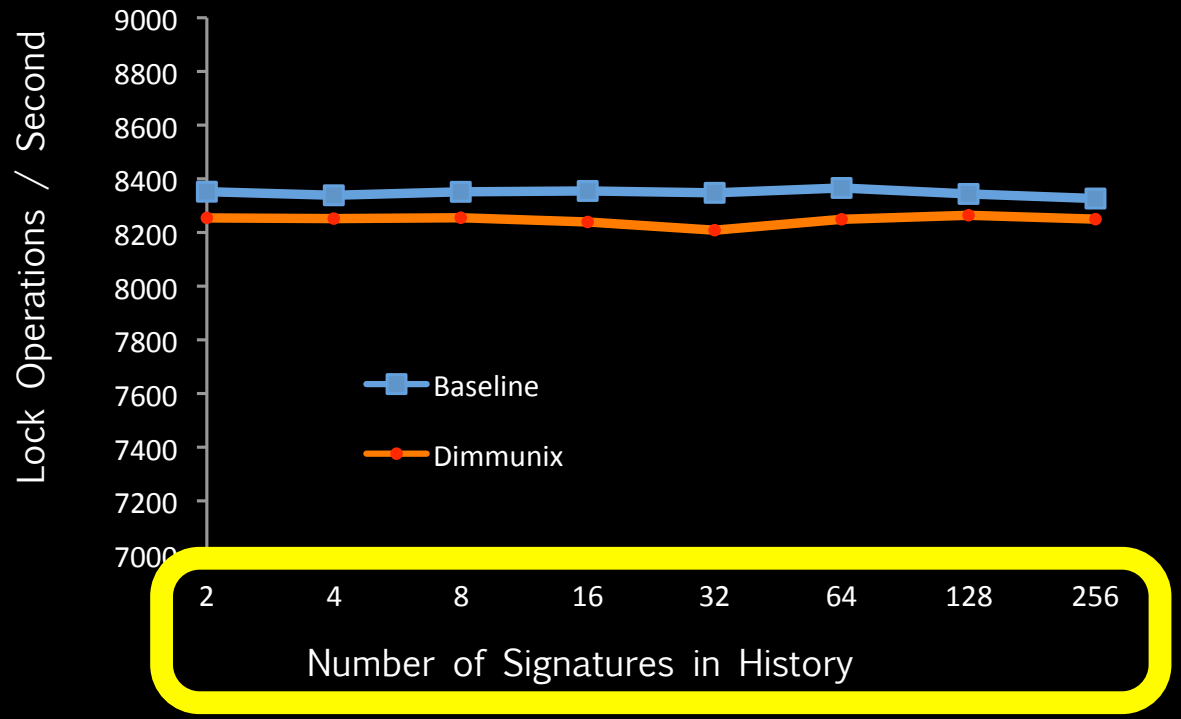
1. Performance

- Matching signatures from history
  - Hash call stacks and index into precomputed tables
  - Thread-local cache of data structures



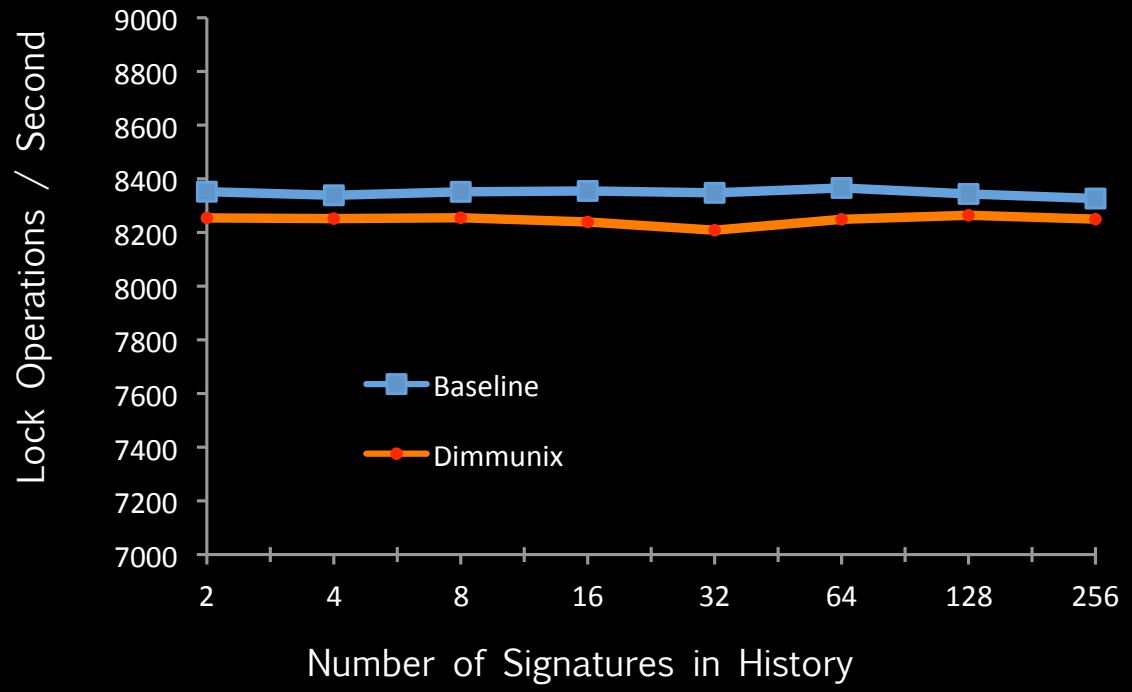
1. Performance

- Matching signatures from history
  - Hash call stacks and index into precomputed tables
  - Thread-local cache of data structures



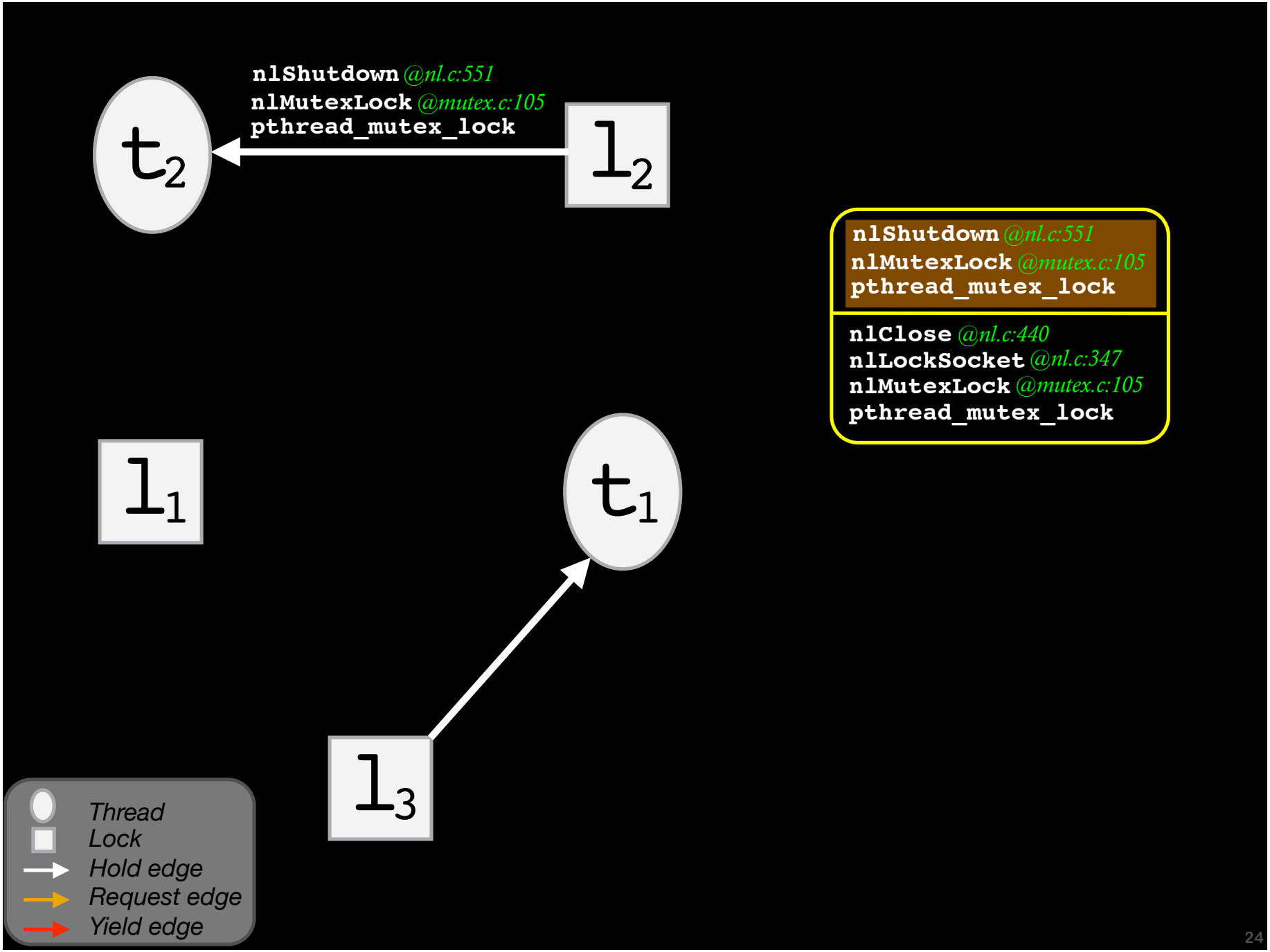
1. Performance

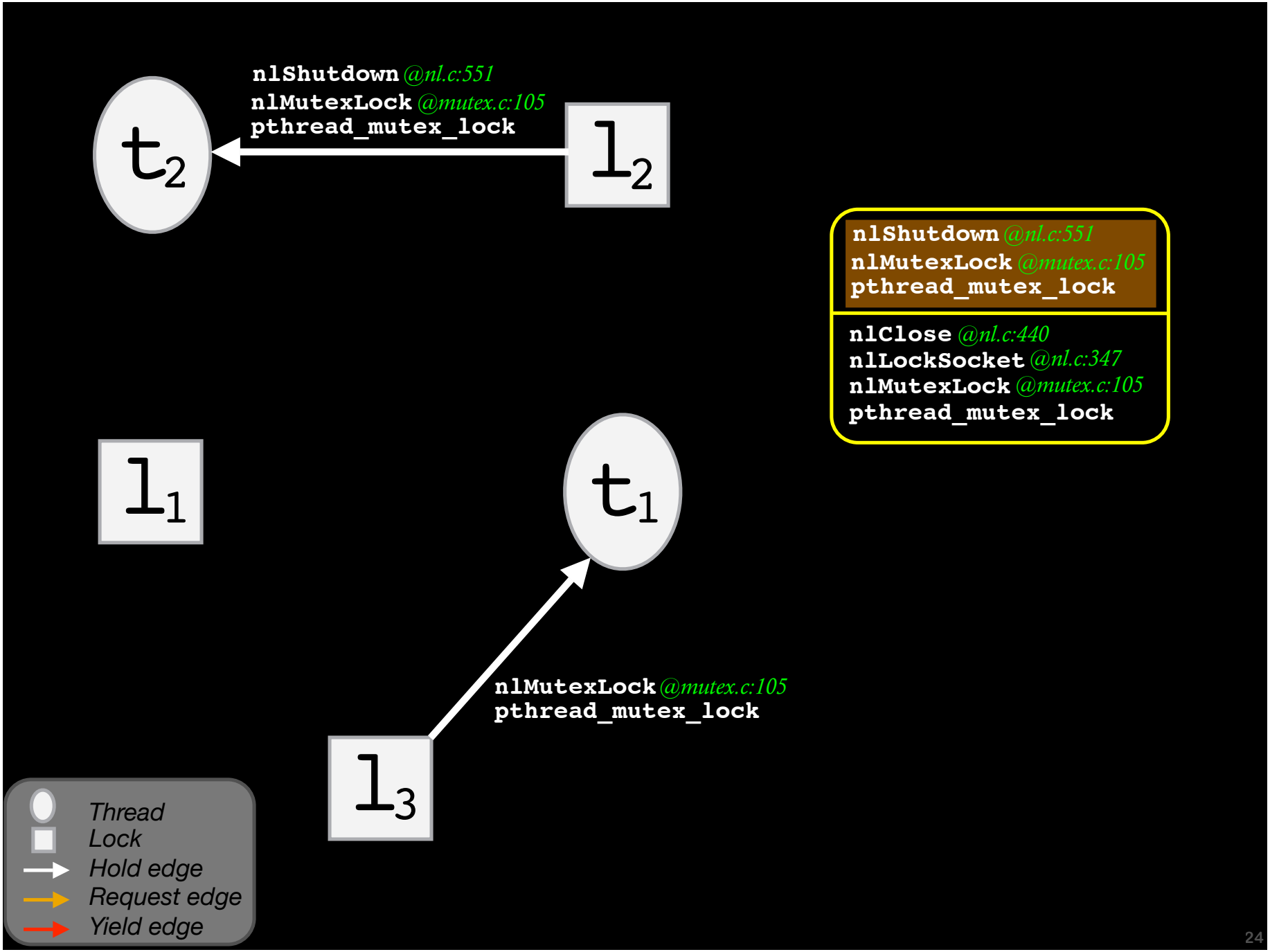
- Matching signatures from history
  - Hash call stacks and index into precomputed tables
  - Thread-local cache of data structures

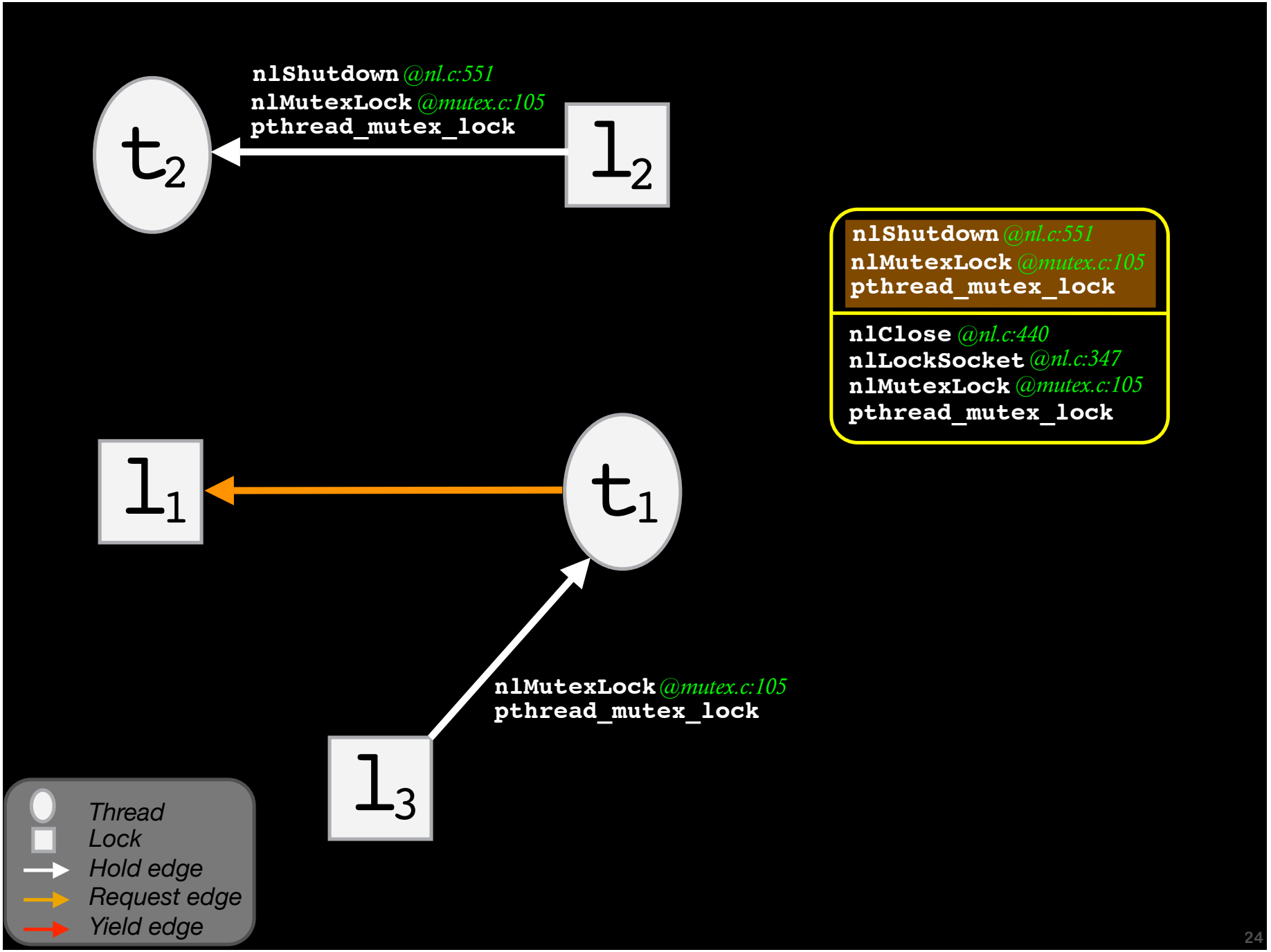


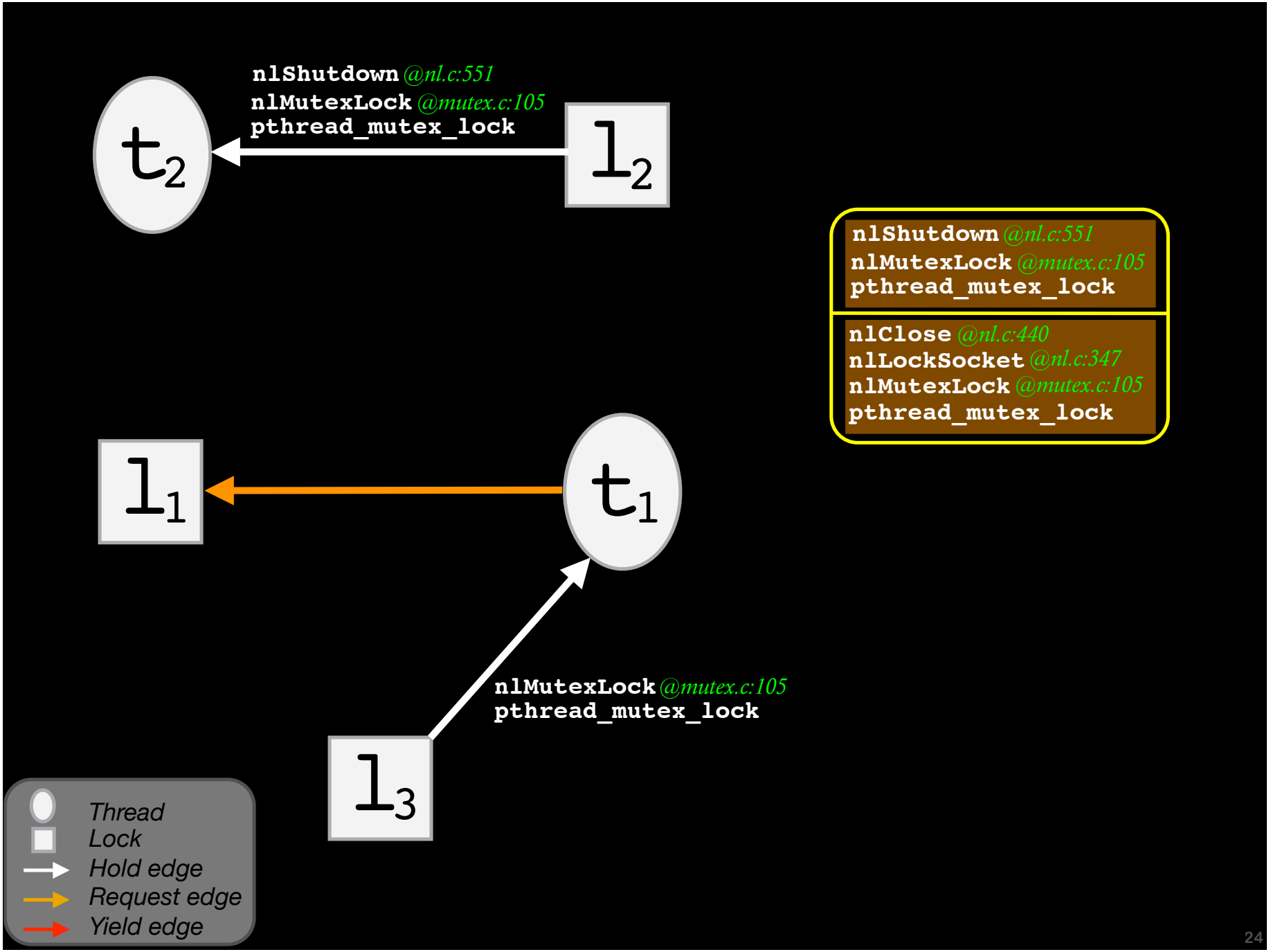
2. Induced Starvation

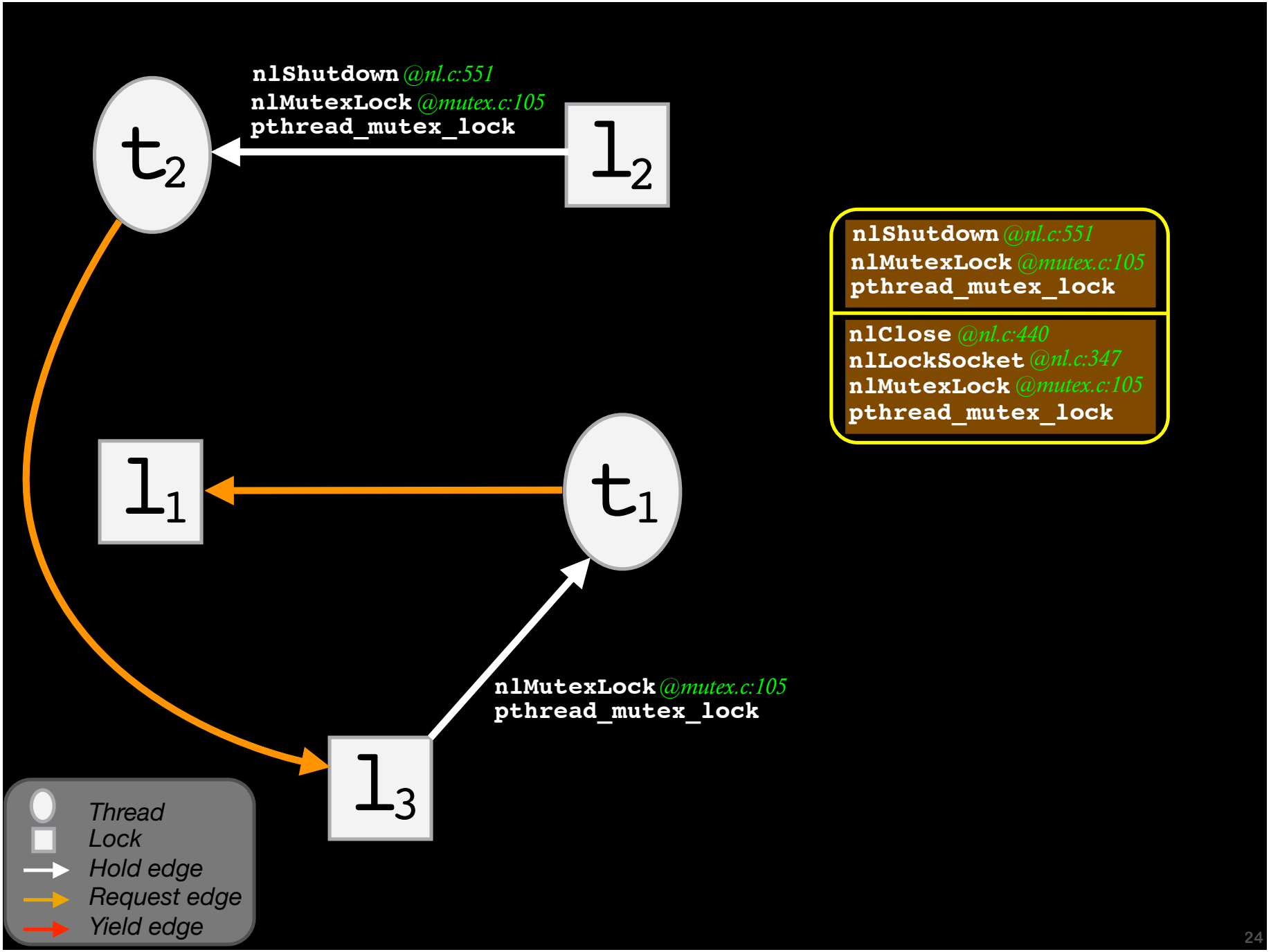
- All approaches based on yield/suspend can induce starvation
  - i.e., a thread actively yields, waiting for a blocked thread to make progress
- Never encountered in our tests

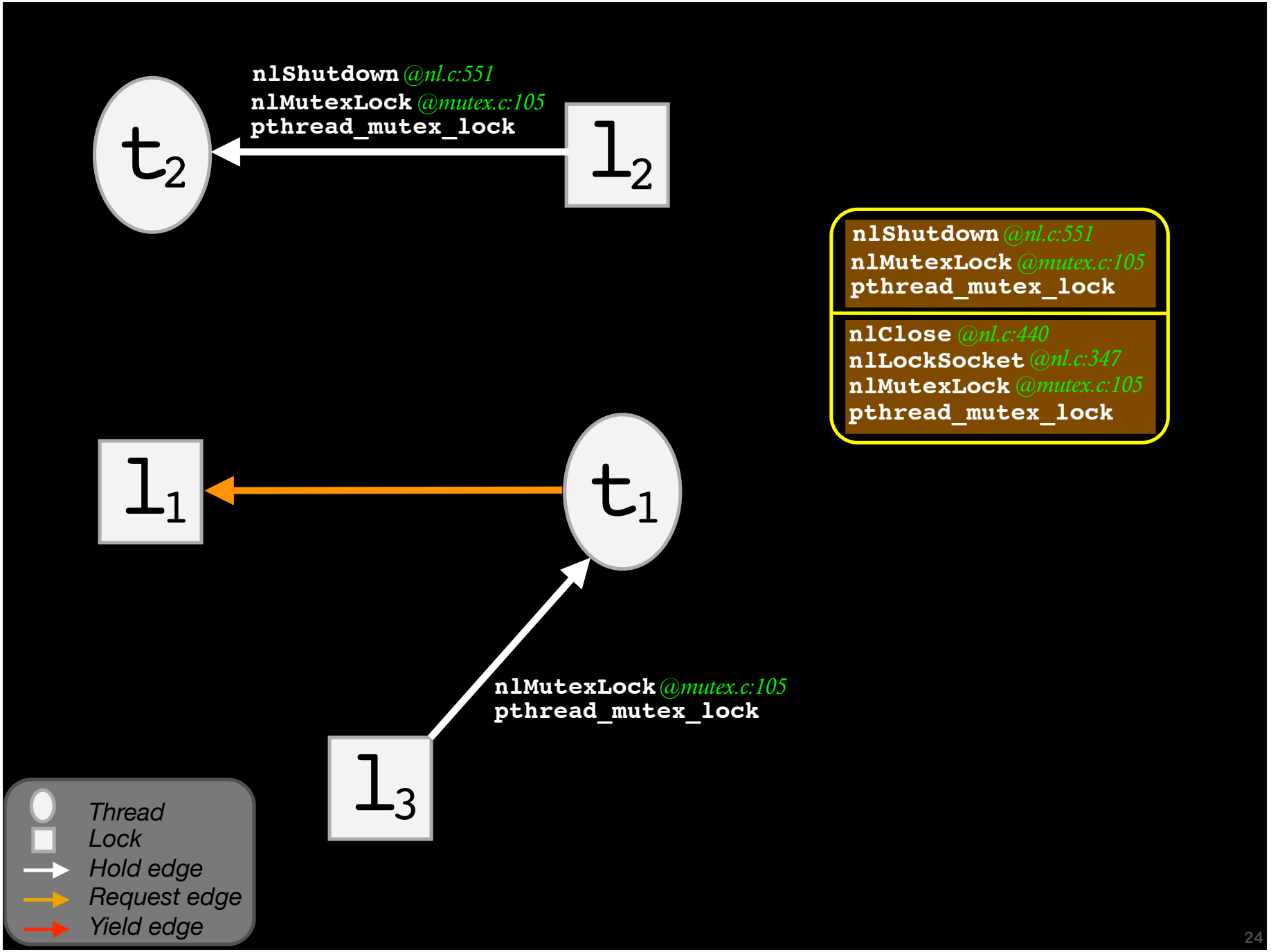

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
l3
Thread
Lock
Hold edge
Request edge
Yield edge

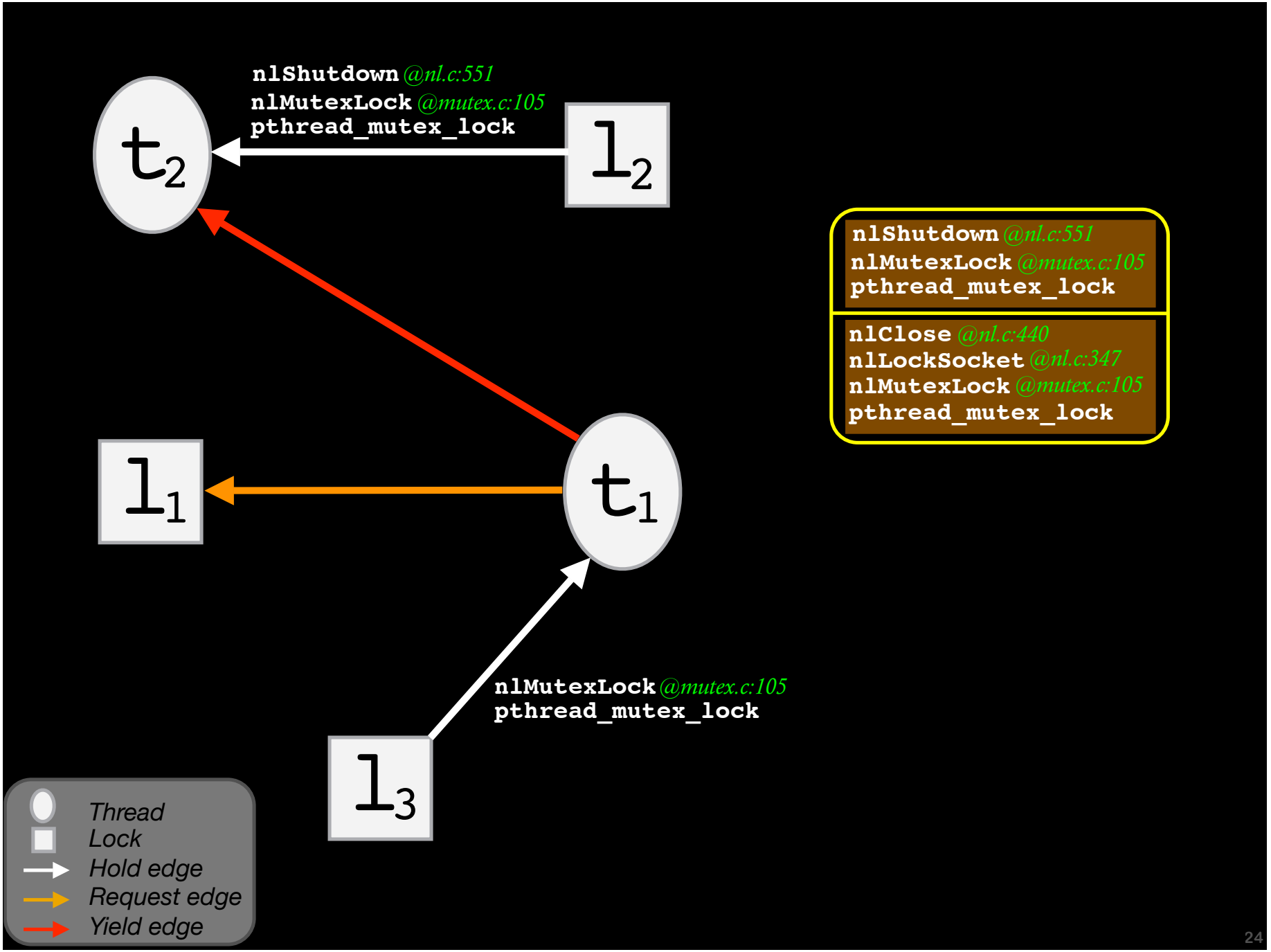

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge

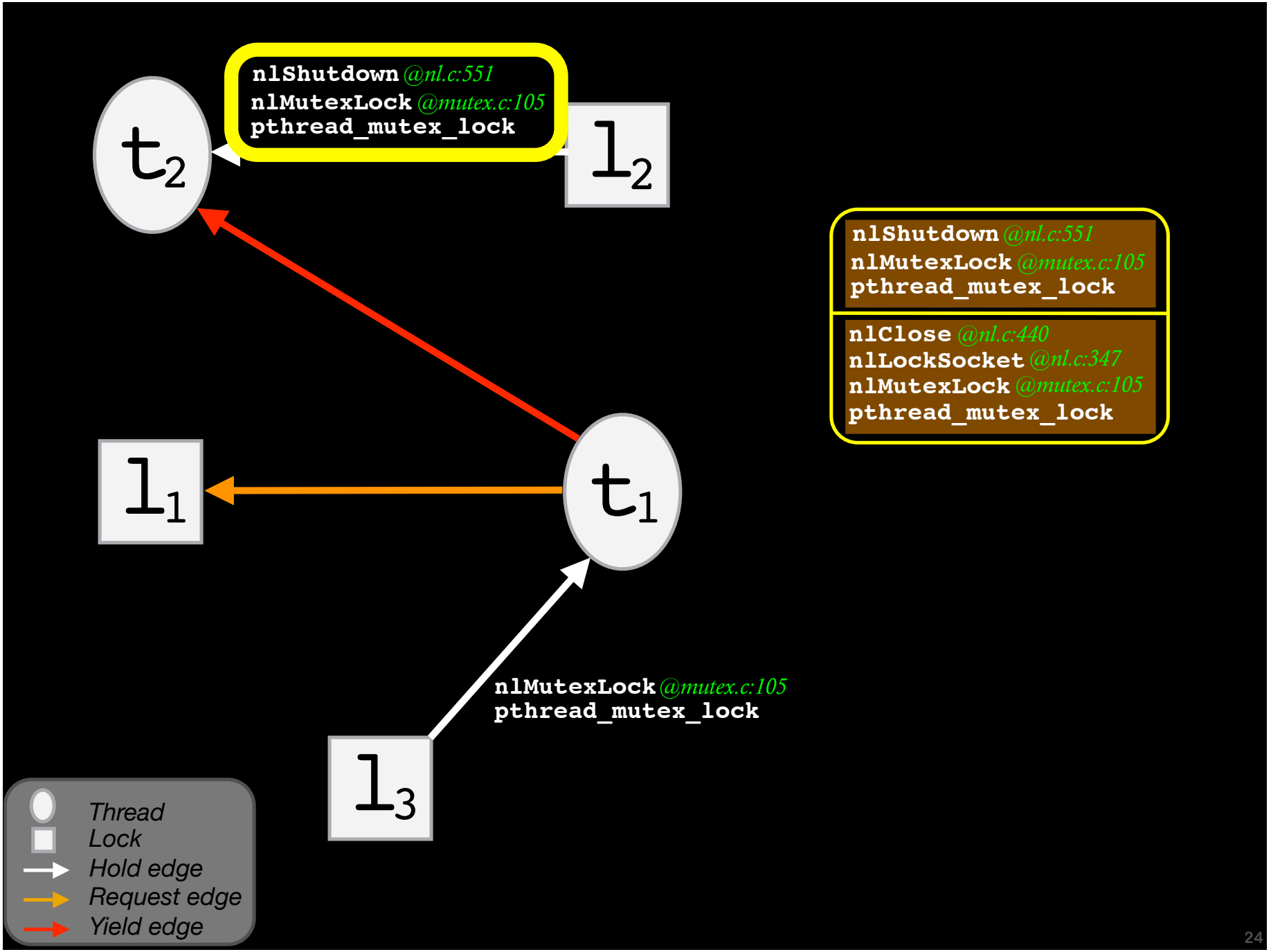

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge

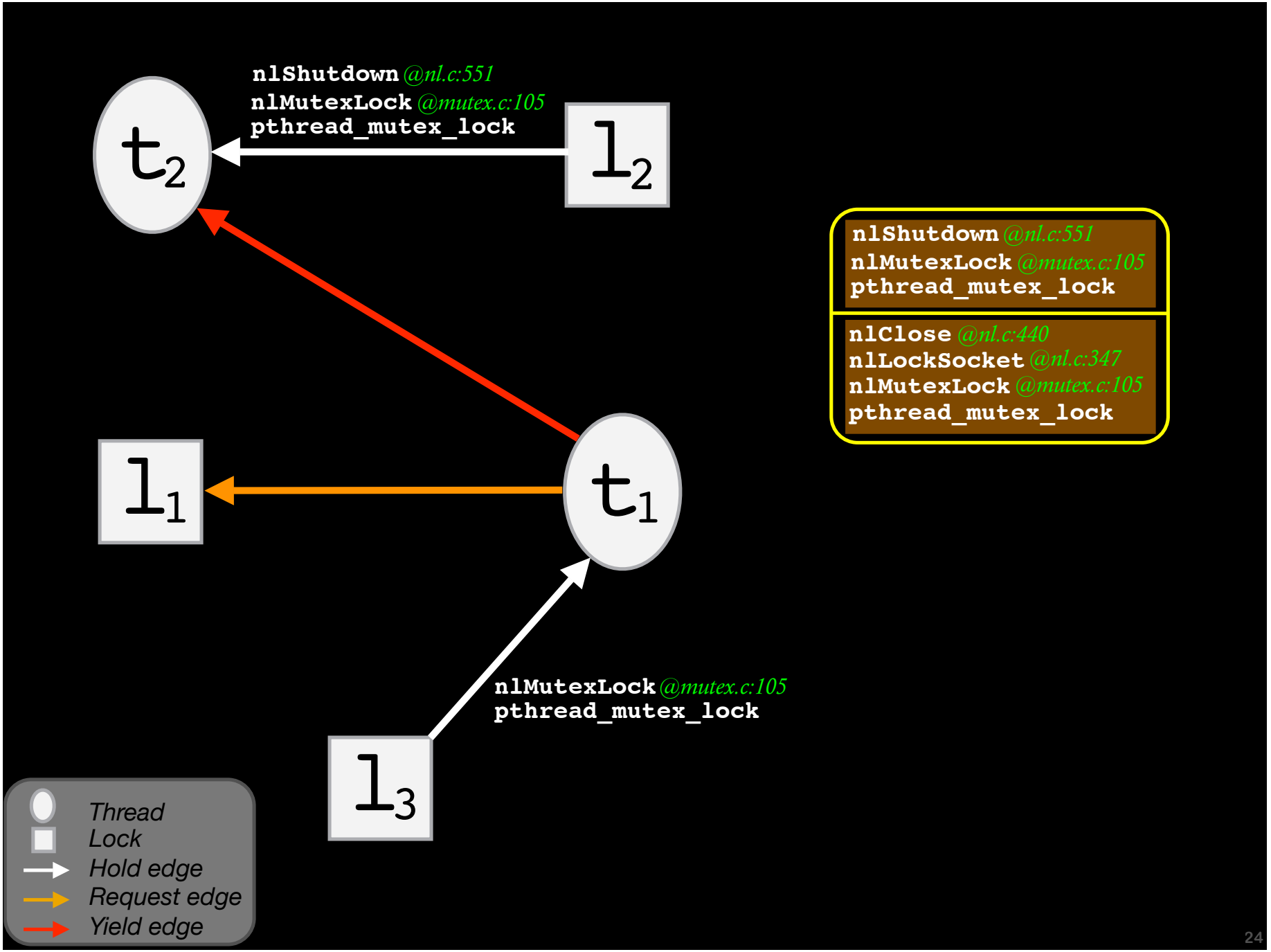

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge

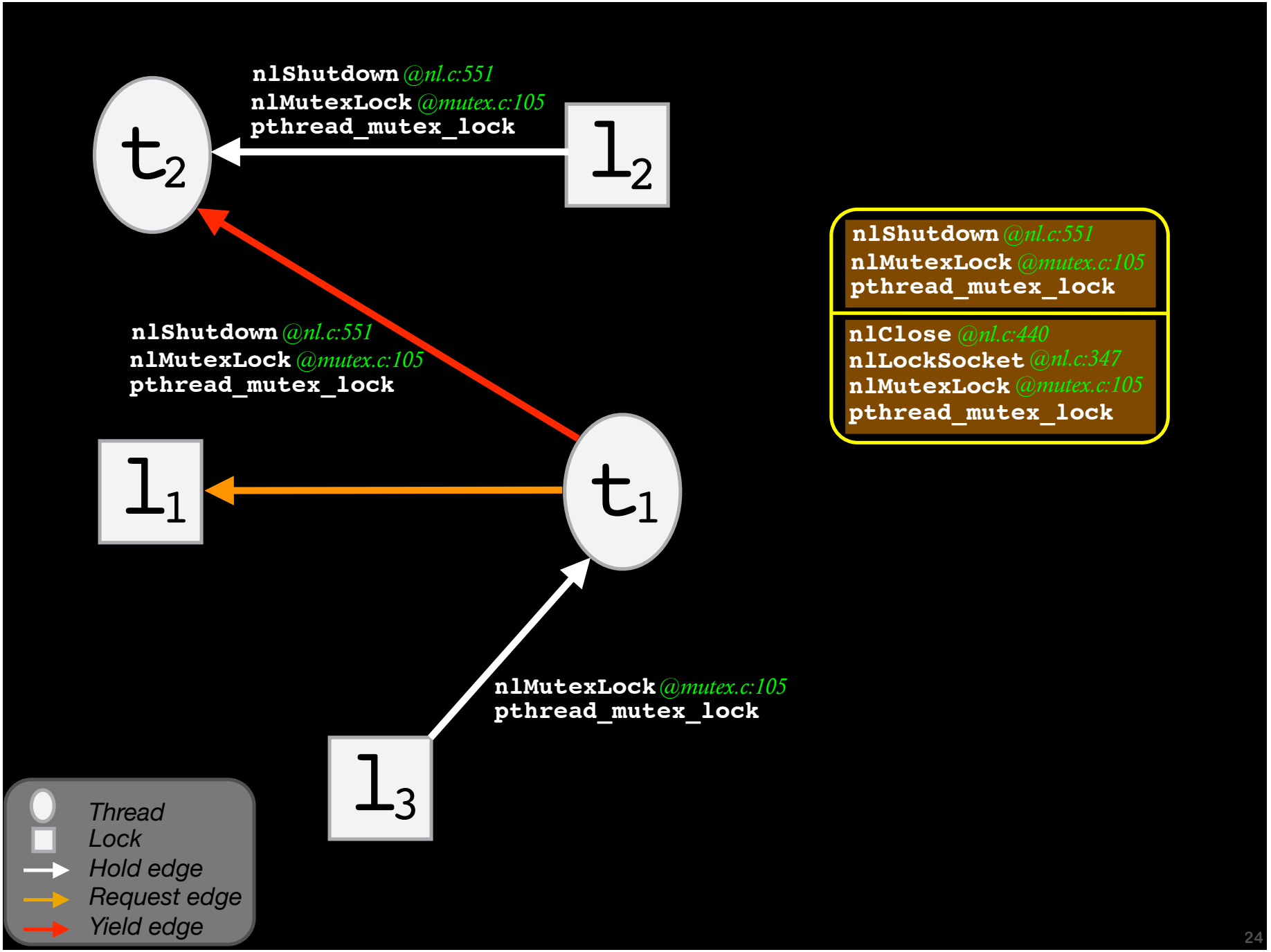

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge

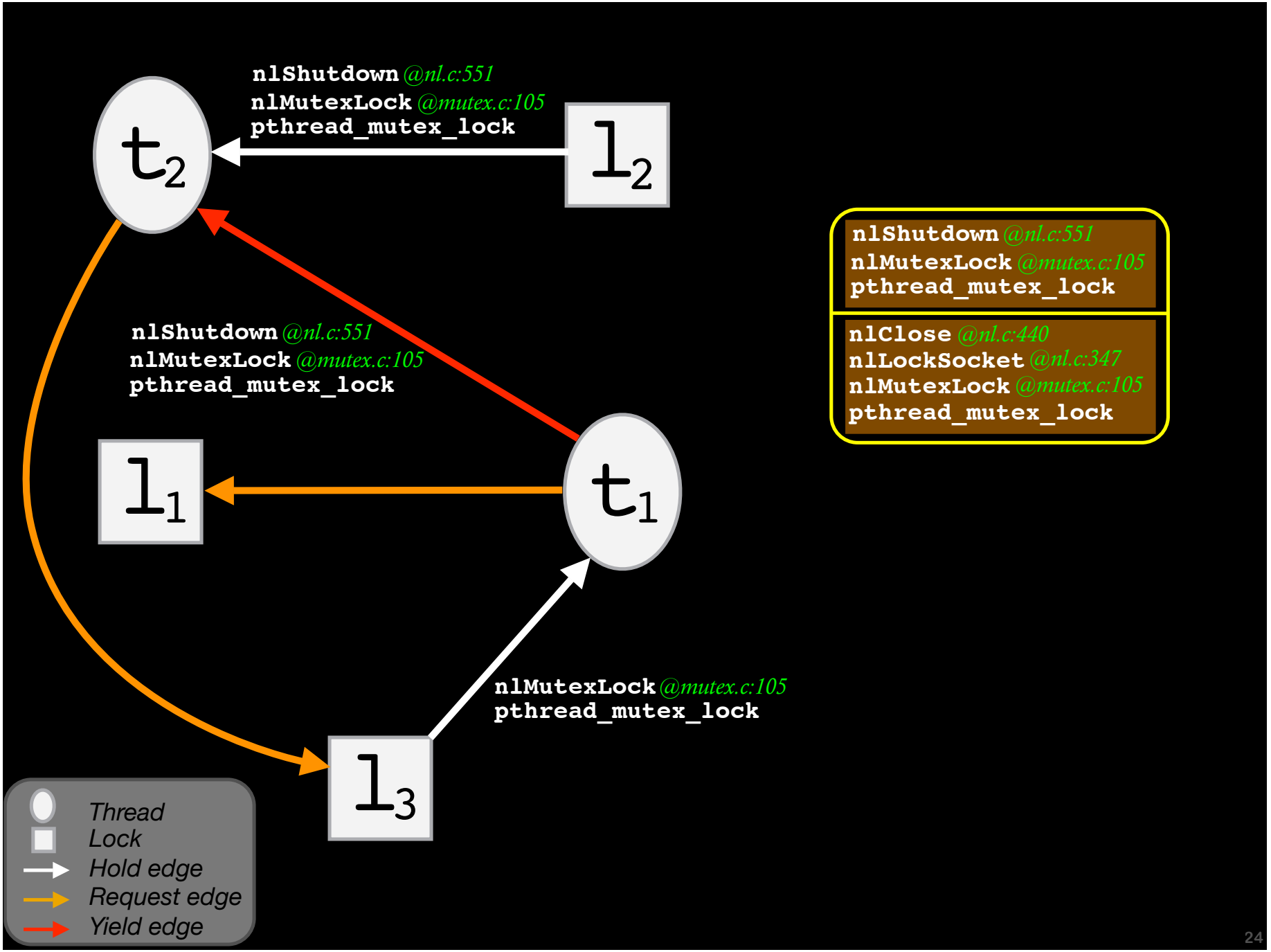

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge

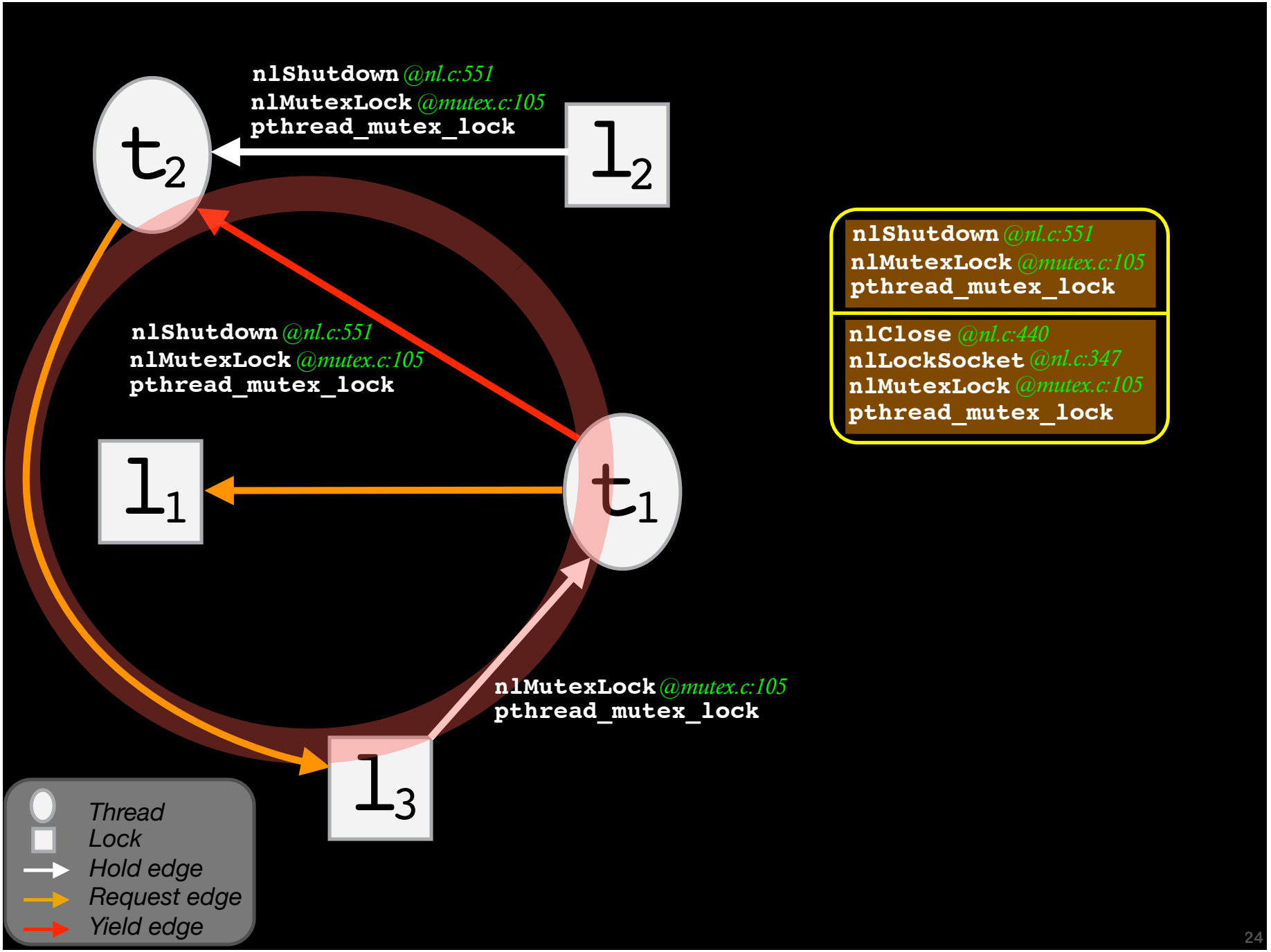

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge

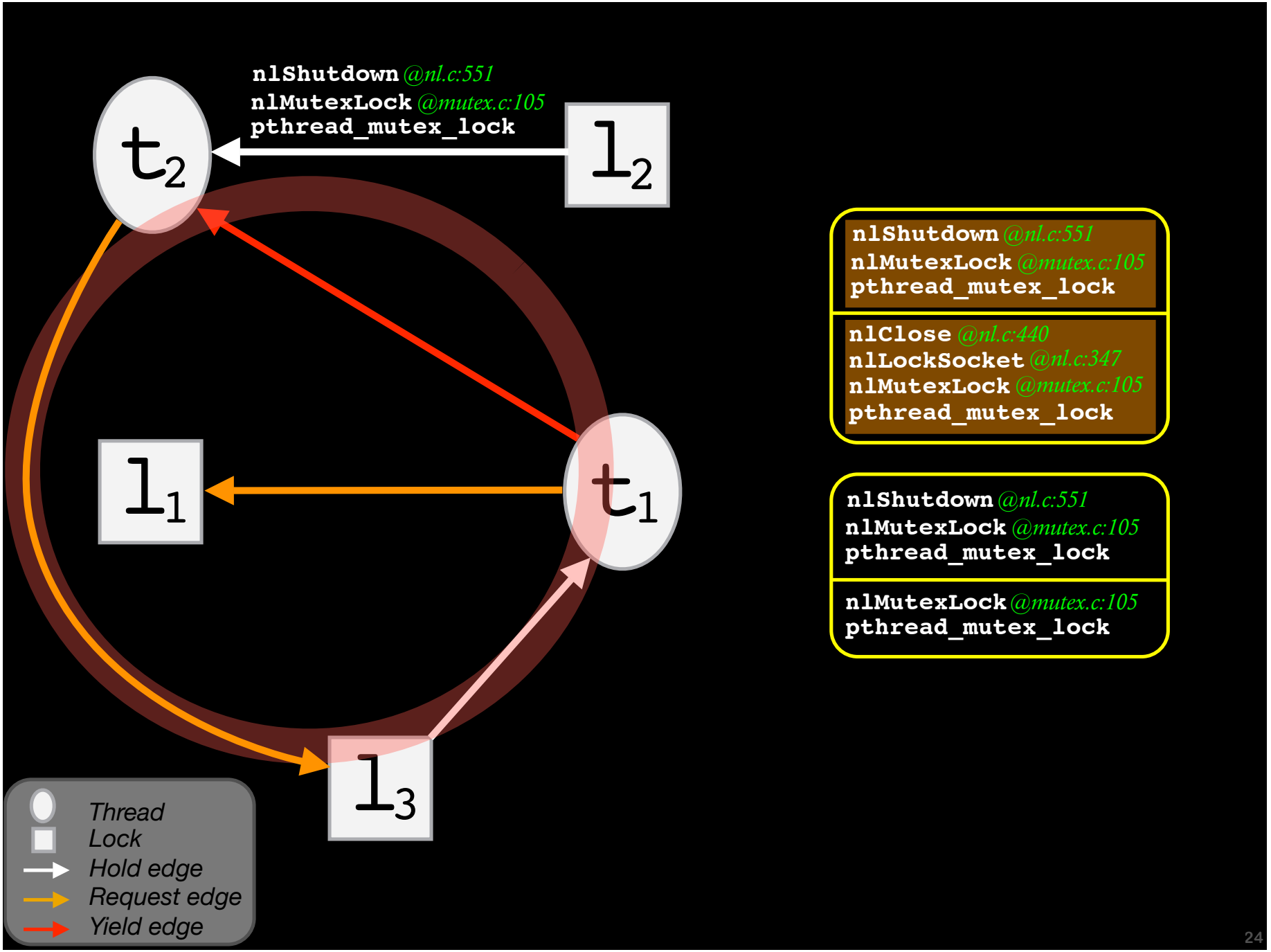

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge

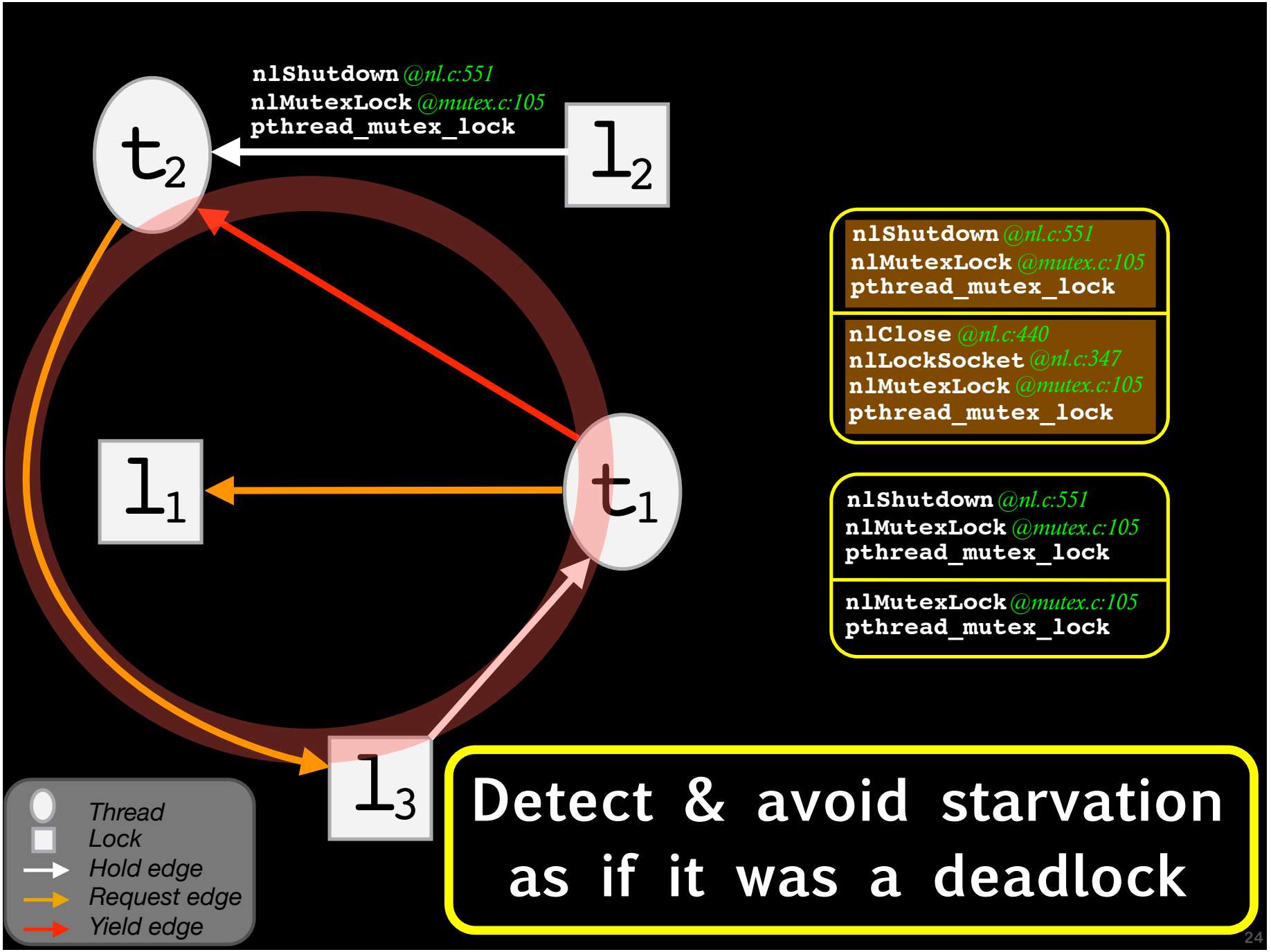

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge


nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge


nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge

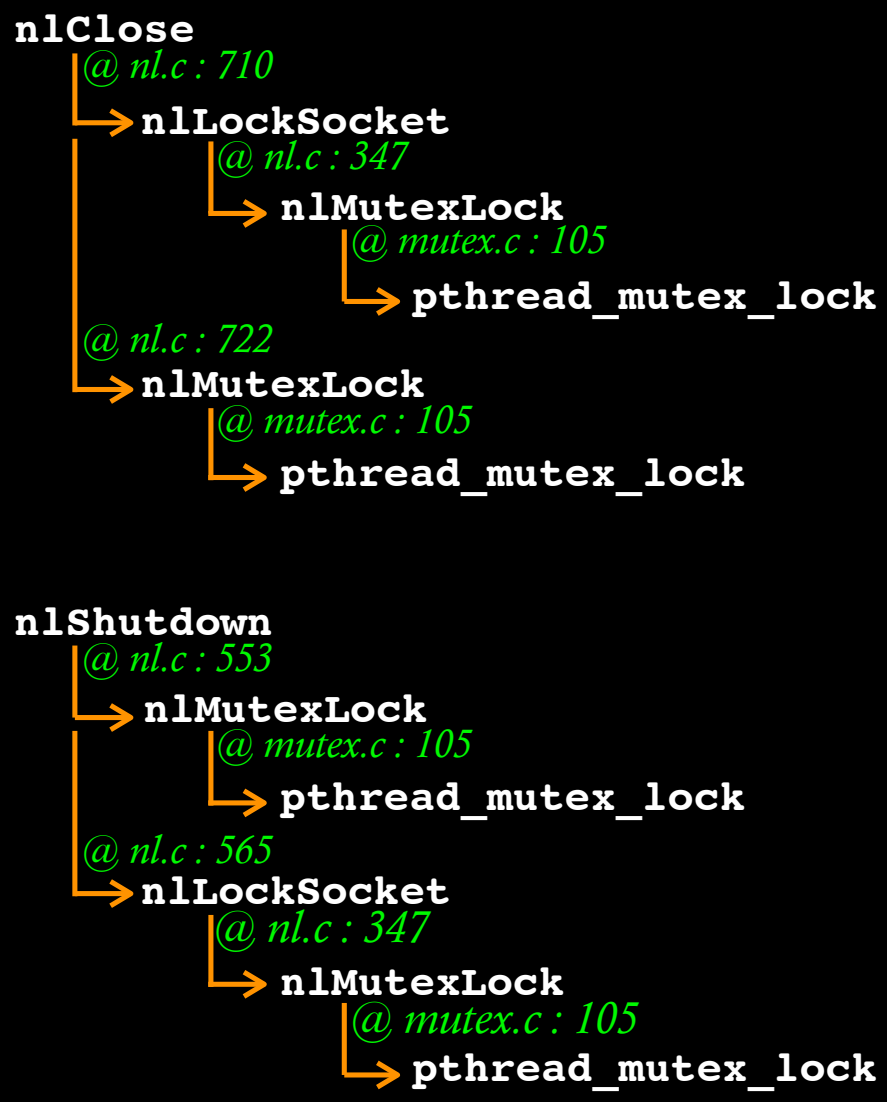

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
l1
t1
nlMutexLock @mutex.c:105
pthread_mutex_lock
l3
Thread
Lock
Hold edge
Request edge
Yield edge
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock

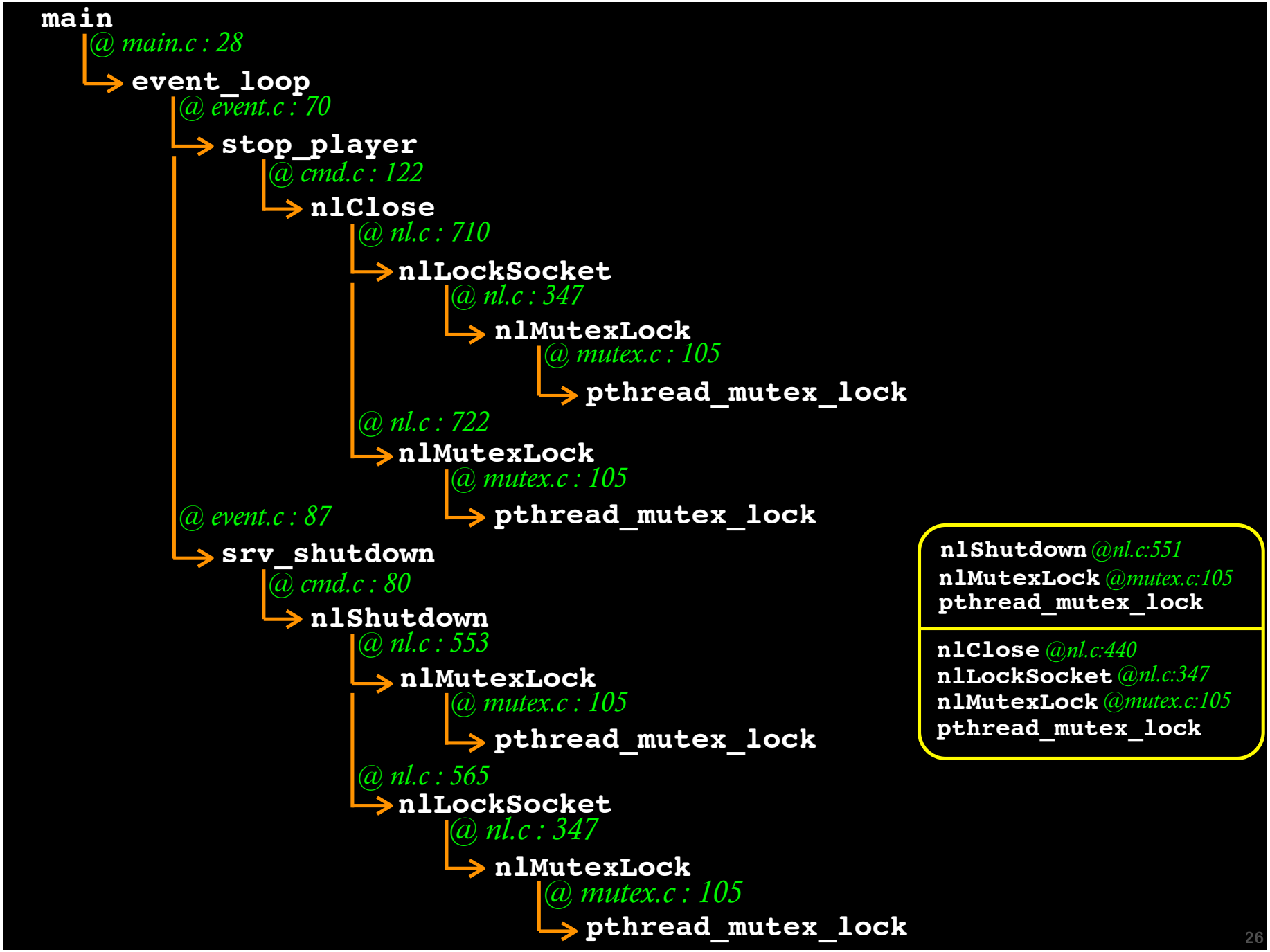

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
l1
t1
l3
Thread
Lock
Hold edge
Request edge
Yield edge
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlMutexLock @mutex.c:105
pthread_mutex_lock

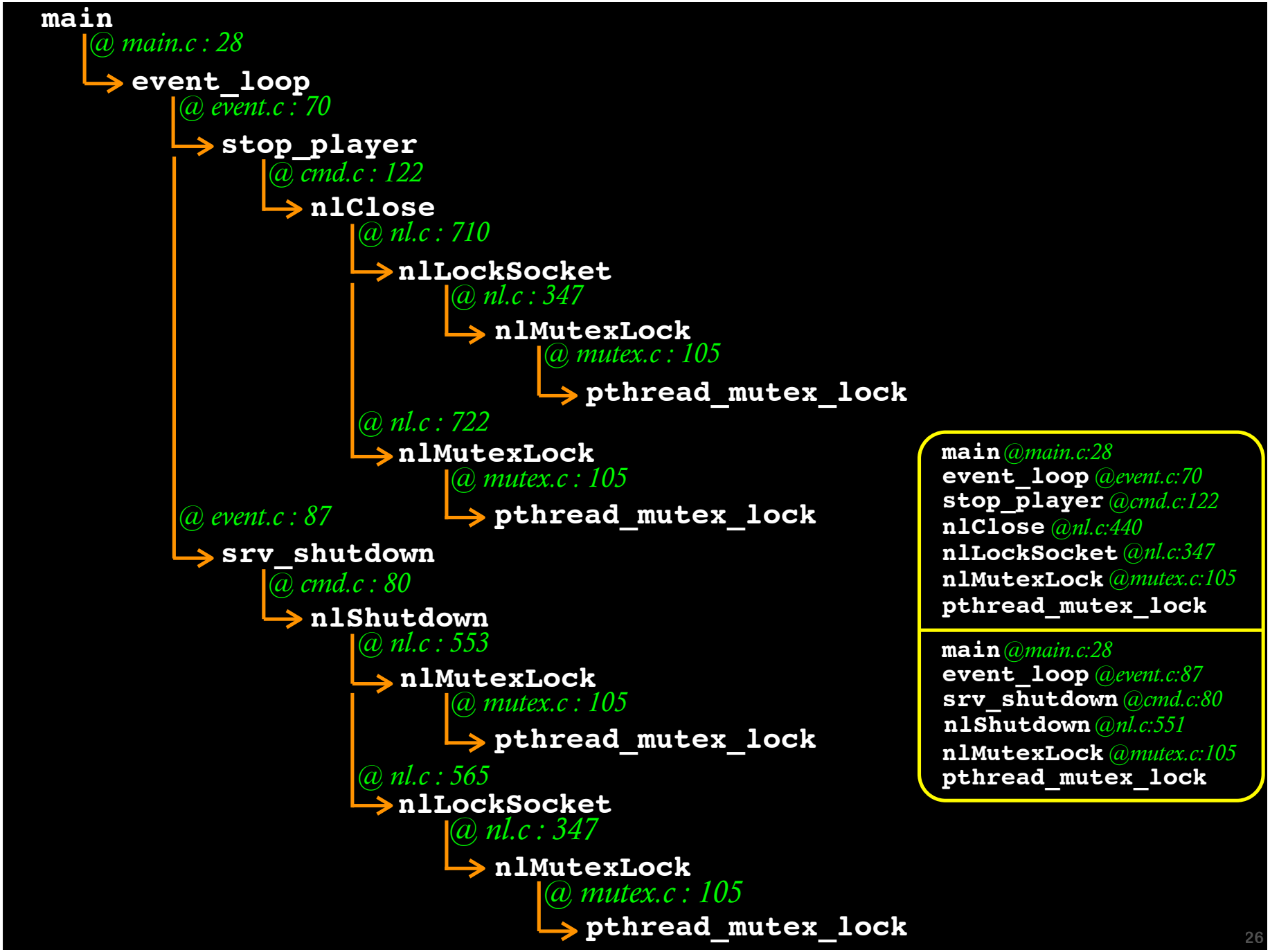

nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
t2
l2
l1
t1
l3
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlMutexLock @mutex.c:105
pthread_mutex_lock
Thread
Lock
Hold edge
Request edge
Yield edge
Detect & avoid starvation as if it was a deadlock

3. Precision Calibration

- False positive = avoiding a deadlock that would not have occurred
- Two sources:
  - Matching signatures too generally
  - Input dependencies


nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock


main
@ main.c : 28
event_loop
@ event.c : 70
stop_player
@ cmd.c : 122
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ event.c : 87
srv_shutdown
@ cmd.c : 80
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock


main
@ main.c : 28
event_loop
@ event.c : 70
stop_player
@ cmd.c : 122
nlClose
@ nl.c : 710
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 722
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ event.c : 87
srv_shutdown
@ cmd.c : 80
nlShutdown
@ nl.c : 553
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
@ nl.c : 565
nlLockSocket
@ nl.c : 347
nlMutexLock
@ mutex.c : 105
pthread_mutex_lock
main @main.c:28
event_loop @event.c:70
stop_player @cmd.c:122
nlClose @nl.c:440
nlLockSocket @nl.c:347
nlMutexLock @mutex.c:105
pthread_mutex_lock
main @main.c:28
event_loop @event.c:87
srv_shutdown @cmd.c:80
nlShutdown @nl.c:551
nlMutexLock @mutex.c:105
pthread_mutex_lock

3. Precision Calibration

- Solution: Automatic precision adjustment
  - Post-factum what-if analysis
  - Heuristic: look for potential lock inversions
- Input dependencies not captured
  - Dimmunix focused 100% on control flow
  - This is a requisite for portability of signatures, i.e., for immunity

Outline

- Immunity Against Deadlocks
- Dimmunix Overview
- Challenges & Solutions
- Discussion & Conclusions

Dimmunix Properties

1. Someone will experience first occurrence
   - Afterward can vaccinate all others
2. Cannot affect deadlock-free programs
3. Cannot induce incorrect outputs (non-RT)
4. Must be aware of all synch mechanisms

Related Work

- **Complements other reliability techniques**
  - **Static analysis** [*Flanagan & Leino*] [*Engler & Ashcraft*]
  - **Modelchecking** [*Henzinger et al.*] [*Havelund et al.*]
  - **Transactional Memory** [*Herlihy & Moss*]
- **Improves upon previous approaches by increasing avoidance precision**
  - **Program transformations** [*Boronat & Cholvi*]
  - **Ghostlocks** [*Zeng & Martin*]
  - **Gatelocks** [*Buchbinder et al.*]

Dimmunix

- Failure immunity applied to deadlocks
- Effective for both Java and C/C++
- Low overhead (4.5% for 1024 threads)
- Needs no assistance & no source code
- Can use as band-aid and vaccine

http://dimmunix.epfl.ch/