ECEN 5623

Blocking and Priority Inversion

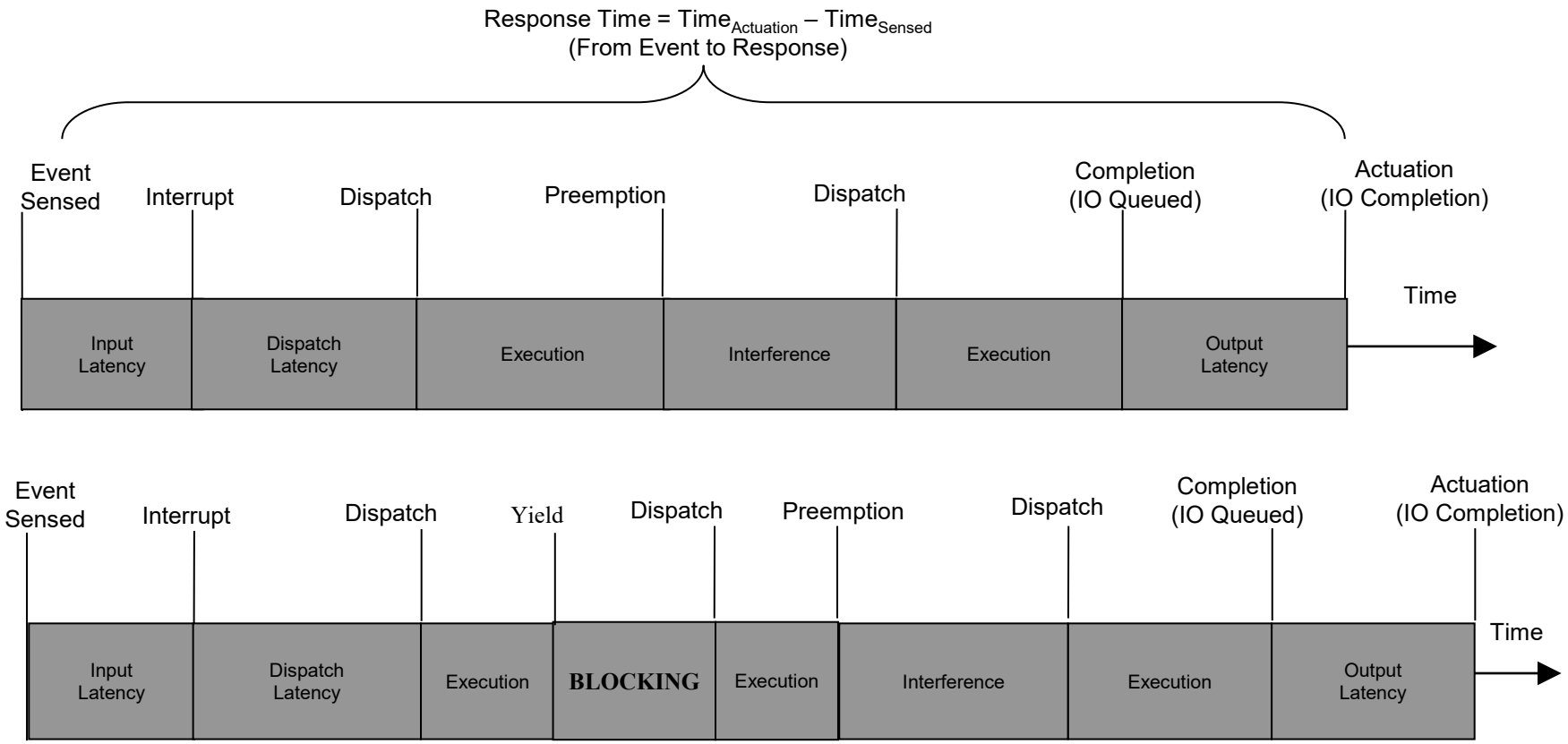
Be Boulder.
University of Colorado Boulder

Blocking

- Blocking is Evil!
- Caused by Need for Shared Resource that is Unavailable Despite Availability of CPU
- Ideally Eliminate Potential for Blocking During Service Execution
- If Elimination Impossible, Bounded Blocking OK
  - Deterministic Upper Bound on Blocking Time
  - Lump Bounded Blocking Time Into Response Timeline
  - Creates Slack
- Unbounded Blocking is Possible and the Real Problem

© Sam Siewert

Service Response Timeline
(With Intermediate Blocking)



How many Services are required to cause a Deadlock?

A. 2
B. 3
C. 4
D. 5

When poll is active, respond at **pollev.com/timscherr391**

Text **TIMSCHERR391** to **37607** once to join

How many Services are required to cause a Deadlock?

A 2

B 3

C 4

D 5

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

How many Services are required to cause a Priority Inversion?

A. 2
B. 3
C. 4
D. 5

When poll is active, respond at **pollev.com/timscherr391**

Text **TIMSCHERR391** to **37607** once to join

How many Services are required to cause a Priority Inversion?

A 2

B 3

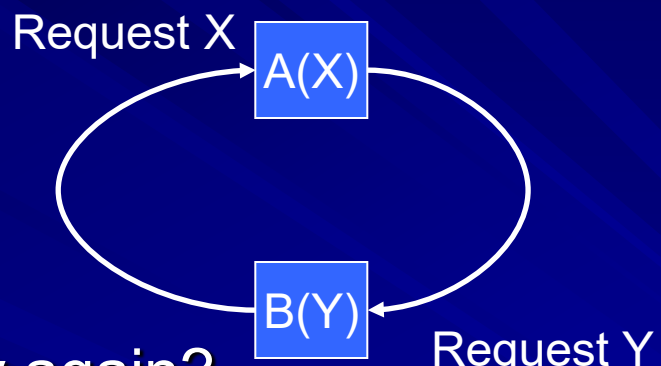
C 4

D 5

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Resource Deadlock (Circular Wait)



- A is holding X and would like Y
- B is holding Y and would like X
- How is this resolved?
- A and B could Block Indefinitely
- Each could release X or Y and try again?
  - Can Result in Livelock
  - They Release, A grabs X, B grabs Y, Deadlock, Detection, Release, A grabs X, B grabs Y …
  - Random Back-Off Times (Possible Solution)
- Circular Wait Can Evolve over Complex Sets of Tasks and Resources (Hard to Detect or Prevent)
- This is Unbounded Blocking
- Detection Most Often with Watch-Dog and Sanity Monitors

© Sam Siewert

Pthread Deadlock

- http://ecee.colorado.edu/~ecen5623/ecen/ex/ecen5623/Linux/example-sync/
- Read and understand example
- Indefinite Blocking

© Sam Siewert

Deadlock NPTL Demo

- Guaranteed Deadlock (default)

[root@localhost example-sync]# ./deadlock
Will set up unsafe deadlock scenario
Creating thread 1
Thread 1 spawned
Creating thread 2
Thread 2 spawned
rsrcACnt=0, rsrcBCnt=0
will try to join CS threads unless they deadlock
THREAD 1 grabbing resources
THREAD 2 grabbing resources
THREAD 1 got A, trying for B
THREAD 2 got B, trying for A
<Ctrl-C>
[root@localhost example-sync]#

Deadlock NPTL Demo

- Guaranteed Safe

```
[root@localhost example-sync]# ./deadlock safe
Creating thread 1
Thread 1 spawned
THREAD 1 grabbing resources
THREAD 1 got A, trying for B
THREAD 1 got A and B
THREAD 1 done
Creating thread 2
Thread 2 spawned
rsrcACnt=1, rsrcBCnt=1
will try to join CS threads unless they deadlock
THREAD 2 grabbing resources
THREAD 1 got B, trying for A
THREAD 2 got B and A
THREAD 2 done
All done
[root@localhost example-sync]#
```

© Sam Siewert

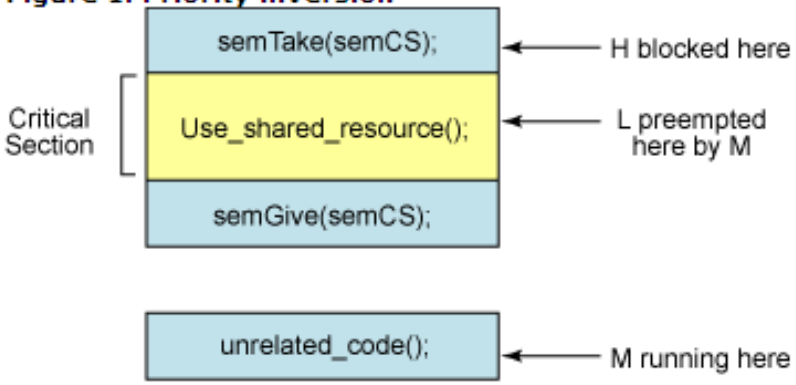
Shared Mutually Exclusive Access Resource

- Shared Mutex Resources Require Protection from Unintentional Non-Mutex Access
- E.g. Shared Memory State that Can't Be Updated Atomically
  - Position[] = {x, y, z}
  - To Update Position Requires More Than One Write Instruction
  - What Happens on Interrupt Between Update of X and Y?
  - What if That Interrupt Releases a Service that Reads Position?
  - Data Corruption!

© Sam Siewert

Shared Mutually Exclusive Access Resource

Figure 1. Priority inversion



H has priority > M and M has priority > L
H and L are blocked as long as M continues to run
By priority policy H should run except it can't obtain mutual exclusion resource

Priority inversion and deadlock can kill "five nines"

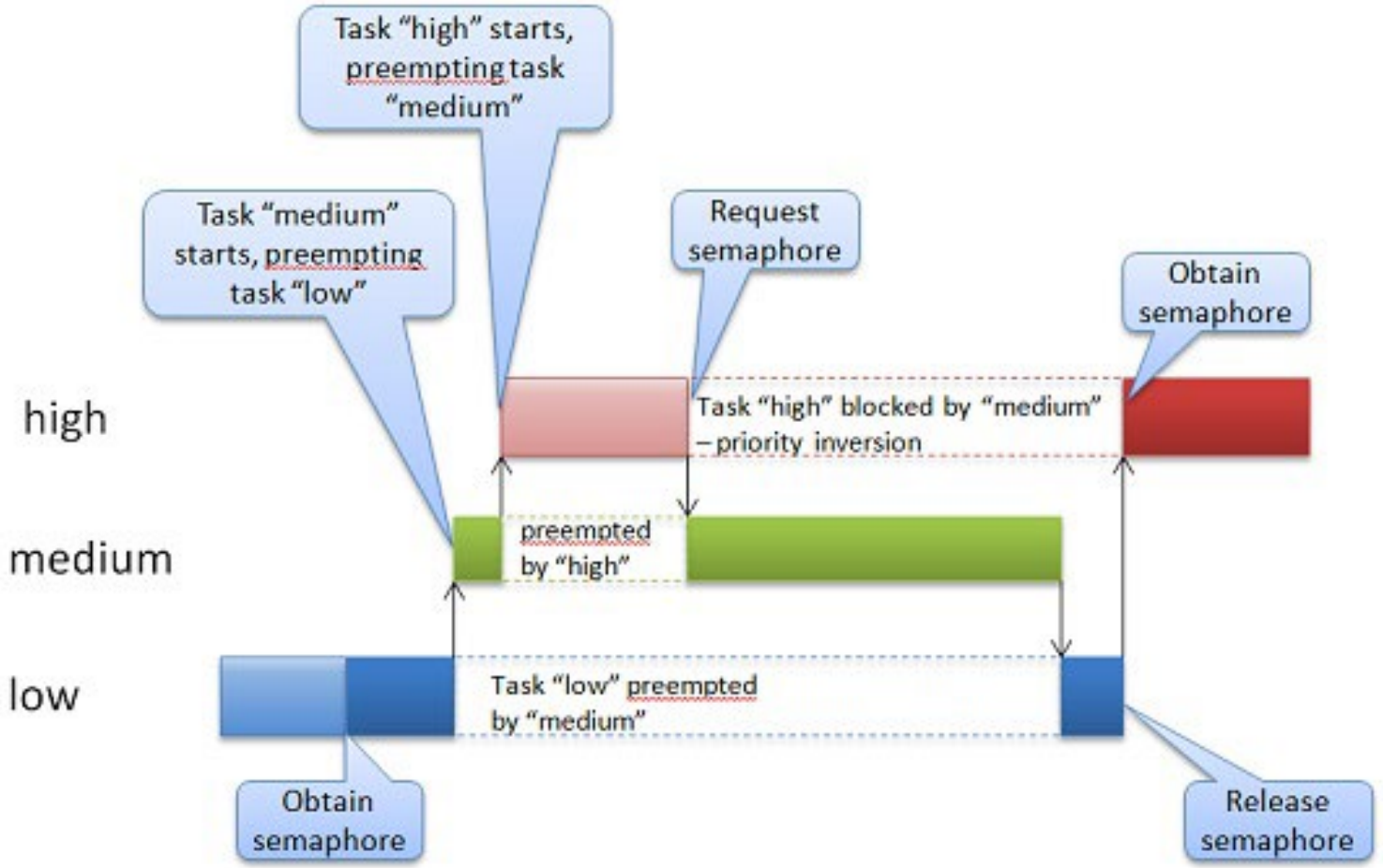
Both priority inversion and deadlock in software often require recovery through a hardware watchdog timer and a reboot. Rebooting means loss of service time. A "five nines" system (one with 99.999% uptime) can't be out of service for more than five minutes per year, so rebooting to recover directly affects a system's ability to meet this availability measure. Quicker recovery methods can help, but avoiding the necessary conditions for deadlock and priority inversion is a much better design approach.

Figure 1 shows a scenario that can cause an unbounded inversion. Here we have a high-priority service (H), a middle-priority service (M), and a low-priority service (L). The necessary condition for priority inversion are:

- **Three or more services in the system sharing a CPU**
- **Two services with priority H and L accessing a shared resource.** The resource must have mutually exclusive access to avoid data corruption or contention.
- **prio(H) > prio(M1) > prio(M2) > ... > prio(Mn) > prio(L).** Thus, all M services with priorities between H and L are not involved in the shared resource, but can preempt L.

© Sam Siewert

Priority Inversion



Priority Inversion

- Problem: Service Using Shared Resource May Suffer Unbounded Priority Inversion
  - Mutex Protection of a Resource May Result in Unbounded Inversion
  - ***3 Necessary Conditions for Unbounded Inversion***
    - Three or More Services With Unique Priority in the System - High, Medium, Low Priority Sets of Services
    - At Least Two Services of Different Priority Share a Resource with Mutex Protection - One or More High and One or More Low Involved
    - One or More Services Not Involved in the Mutex Has Priority Between the Two Involved in the Mutex
  - What Happens?
    - Low Priority Service Enters Mutex and High Priority Blocks on Mutex
    - The Medium Priority Services Not Involved in the Mutex Can Interfere with the Low Priority Service for An Indeterminate Amount of Time
  - ***Possible Solution***: Priority Inheritance or Priority Ceiling

© Sam Siewert

Priority Inversion

- Problem: Service Using Shared Resource May Suffer Unbounded Priority Inversion
  - Mutex Protection of a Resource May Result in Unbounded Inversion
  - ***3 Necessary Conditions for Unbounded Inversion***
    - Three or More Services With Unique Priority in the System - High, Medium, Low Priority Sets of Services
    - At Least Two Services of Different Priority Share a Resource with Mutex Protection - One or More High and One or More Low Involved
    - One or More Services Not Involved in the Mutex Has Priority Between the Two Involved in the Mutex
  - What Happens?
    - Low Priority Service Enters Mutex and High Priority Blocks on Mutex
    - The Medium Priority Services Not Involved in the Mutex Can Interfere with the Low Priority Service for An Indeterminate Amount of Time
  - ***Possible Solution***: Priority Inheritance or Priority Ceiling

© Sam Siewert

Priority Inversion

- Problem: Service Using Shared Resource May Suffer Unbounded Priority Inversion
  - Mutex Protection of a Resource May Result in Unbounded Inversion
  - ***3 Necessary Conditions for Unbounded Inversion***
    - Three or More Services With Unique Priority in the System - High, Medium, Low Priority Sets of Services
    - At Least Two Services of Different Priority Share a Resource with Mutex Protection - One or More High and One or More Low Involved
    - One or More Services Not Involved in the Mutex Has Priority Between the Two Involved in the Mutex
  - What Happens?
    - Low Priority Service Enters Mutex and High Priority Blocks on Mutex
    - The Medium Priority Services Not Involved in the Mutex Can Interfere with the Low Priority Service for An Indeterminate Amount of Time
  - ***Possible Solution***: Priority Inheritance or Priority Ceiling

© Sam Siewert

Priority Inversion

- Problem: Service Using Shared Resource May Suffer Unbounded Priority Inversion
  - Mutex Protection of a Resource May Result in Unbounded Inversion
  - ***3 Necessary Conditions for Unbounded Inversion***
    - Three or More Services With Unique Priority in the System - High, Medium, Low Priority Sets of Services
    - At Least Two Services of Different Priority Share a Resource with Mutex Protection - One or More High and One or More Low Involved
    - One or More Services Not Involved in the Mutex Has Priority Between the Two Involved in the Mutex
  - What Happens?
    - Low Priority Service Enters Mutex and High Priority Blocks on Mutex
    - The Medium Priority Services Not Involved in the Mutex Can Interfere with the Low Priority Service for An Indeterminate Amount of Time
  - ***Possible Solution***: Priority Inheritance or Priority Ceiling

© Sam Siewert

Priority Inversion NPTL Demo

- First, 3 Threads with No CS

```
[root@localhost example-sync]# ./pthread3ok 10000
interference time = 10000 secs
unsafe mutex will be created
Pthread Policy is SCHED_OTHER
Pthread Policy is SCHED_OTHER
min prio = 1, max prio = 99
PTHREAD SCOPE SYSTEM
Creating thread 0
Creating thread 1
High prio 1 thread spawned at 1203339106 sec, 128498 nsec
Creating thread 2
Middle prio 2 thread spawned at 1203339106 sec, 128549 nsec
Creating thread 3
Low prio 3 thread spawned at 1203339106 sec, 128604 nsec
**** 1 idle stopping at 1203339106 sec, 168299 nsec
**** 2 idle stopping at 1203339106 sec, 172718 nsec
Start services thread spawned
will join service threads
**** 3 idle stopping at 1203339106 sec, 188757 nsec
LOW PRIO done
MID PRIO done
HIGH PRIO done
START SERVICE done
All done
[root@localhost example-sync]#
```

Priority Inversion NPTL Demo

- 3 Threads Prio H,L in CS and Prio M Not

```
[root@localhost example-sync]# ./pthread3 10000
interference time = 10000 secs
unsafe mutex will be created
Pthread Policy is SCHED_OTHER
Pthread Policy is SCHED_OTHER
min prio = 1, max prio = 99
PTHREAD SCOPE SYSTEM
Creating thread 0
Creating thread 3
Low prio 3 thread spawned at 1203339142 sec, 688239 nsec
Start services thread spawned
will join service threads
Creating thread 2
Middle prio 2 thread spawned at 1203339143 sec, 688496 nsec
Creating thread 1, CScnt=1
High prio 1 thread spawned at 1203339143 sec, 688567 nsec
**** 2 idle NO SEM stopping at 1203339143 sec, 726988 nsec
**** 3 idle stopping at 1203339144 sec, 856797 nsec
LOW PRIO done
MID PRIO done
**** 1 idle stopping at 1203339146 sec, 968860 nsec
HIGH PRIO done
START SERVICE done
All done
[root@localhost example-sync]#
```

Priority Inversion

- Well Understood, Appreciated, and Many Solutions for Unbounded Inversion in RTOS Community
- Linux Was a Different Story at First…
  - Early Claims Made that This Could Never Happen in Linux (Perhaps true before NPTL and POSIX SCHED_FIFO)
  - Debates On the Web
  - Summary of Linux Status
  - Linux Robust Mutex / FUSYN Project
  - Carrier Grade Linux
- For CFS, it is TRUE that Unbounded Priority Inversion Can't Happen
- For SCHED_FIFO, it Can!

© Sam Siewert

Priority Inheritance

- When Higher Priority Task is Blocked on Mutex and Lower Priority Task is in Mutex, Higher Prio Loans Its Prio to the Lower for Scope of Mutex
- Can Chain
  - Even Higher Prio Task Also Blocks and Again Loans Even Higher Prio
  - As More Block More Temporary Prio Transfers Occur
  - All Prios Must Ultimately Be Restored
  - What is the Limit of Chaining?
- What Happens if Mutexes are Nested?

© Sam Siewert

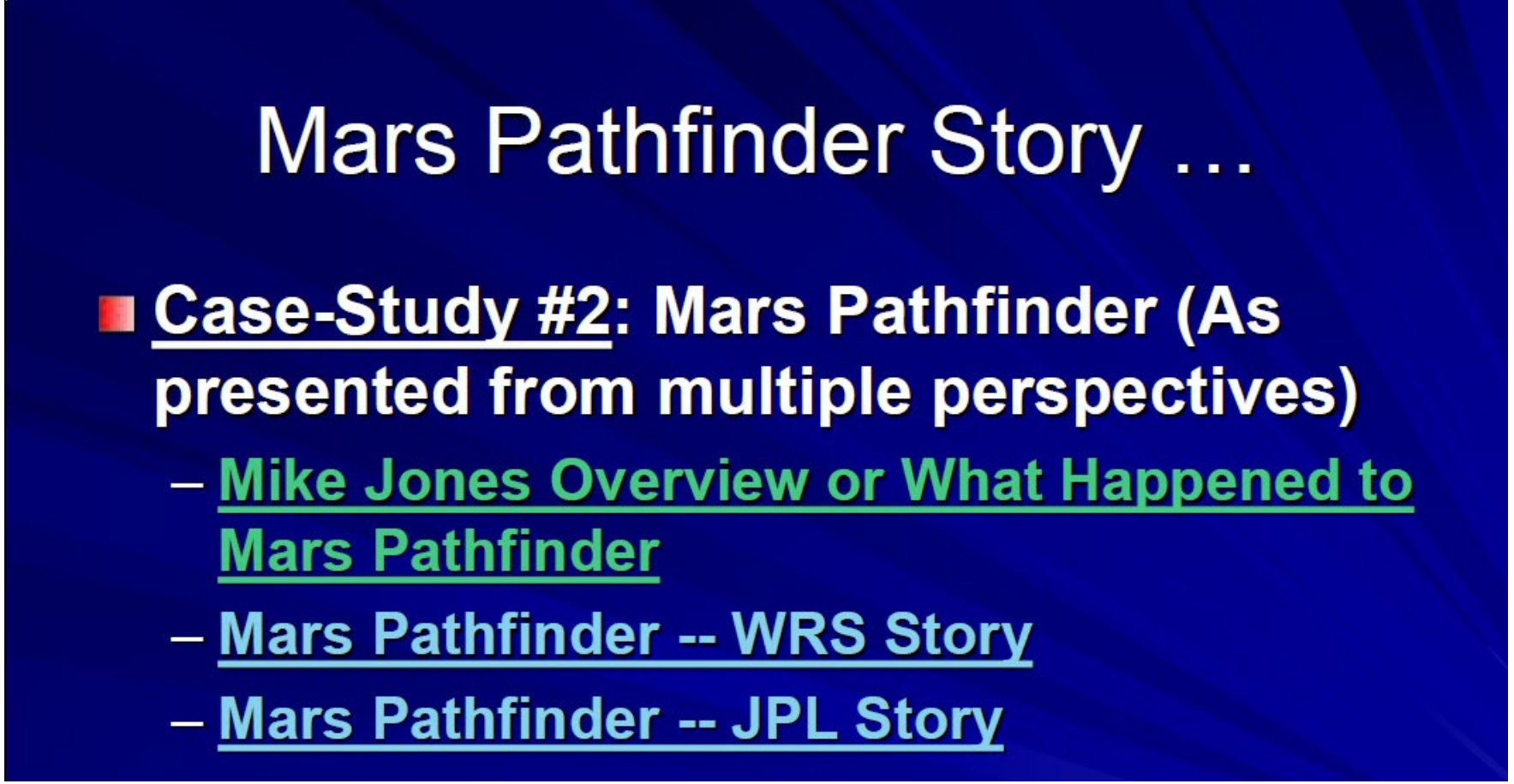
Priority Ceiling Protocol

- Precise Version of Priority Inheritance
- System Ceiling for ALL Lockers Known by OS
- CS Entry Precondition: Holders of System Ceiling or Higher Can Access CS (Safe From Unbounded Inversion)
- Thread in CS will Receive Blocked Task Priority - Inheritance

Priority Ceiling Emulation Protocol

- “Highest Locker”
- Instead of Chaining, Simply Set Prio of Task in Mutex to Highest Immediately When There is an Inversion
- Could be highest Prio in the System
  - May Over-amplify
  - Simple to Implement
- More Precisely Can Be highest Prio of Those Tasks Actually Involved in Mutex

© Sam Siewert



http://ecee.colorado.edu/~siewerts/marspath/mars.html

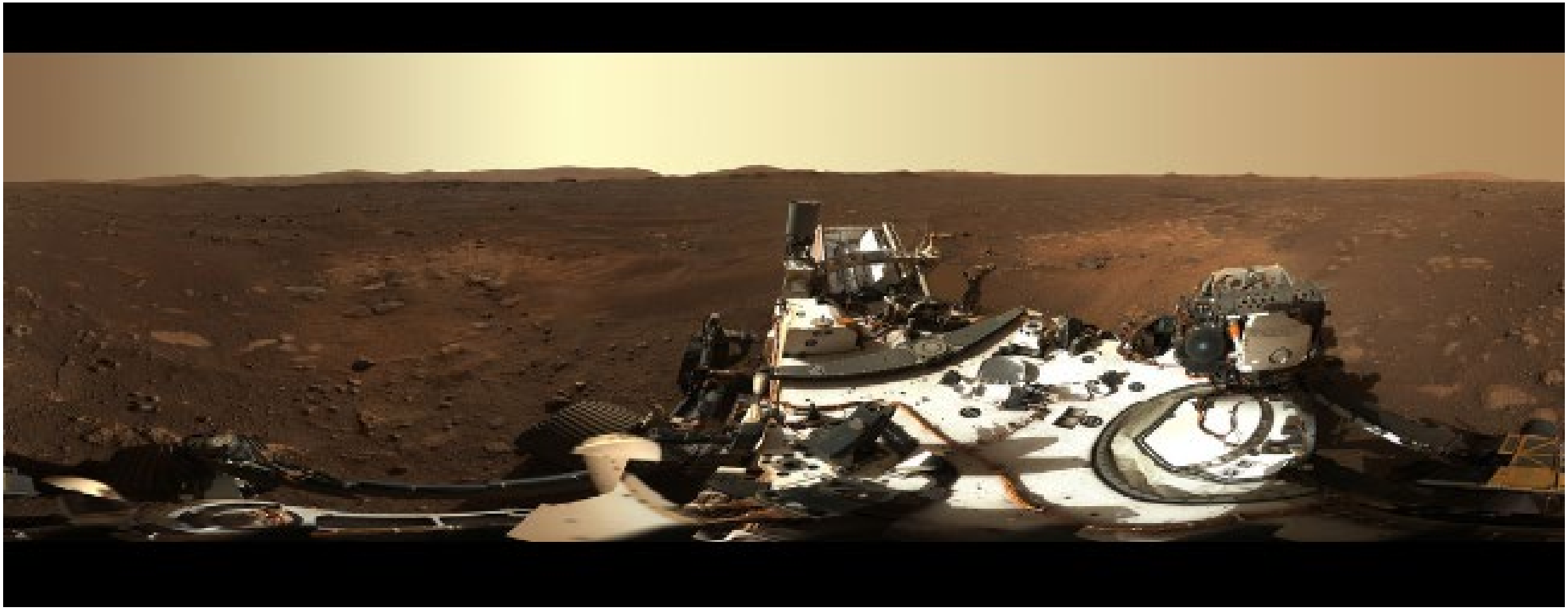
http://ecee.colorado.edu/~siewerts/marspath/wrs/index.htm

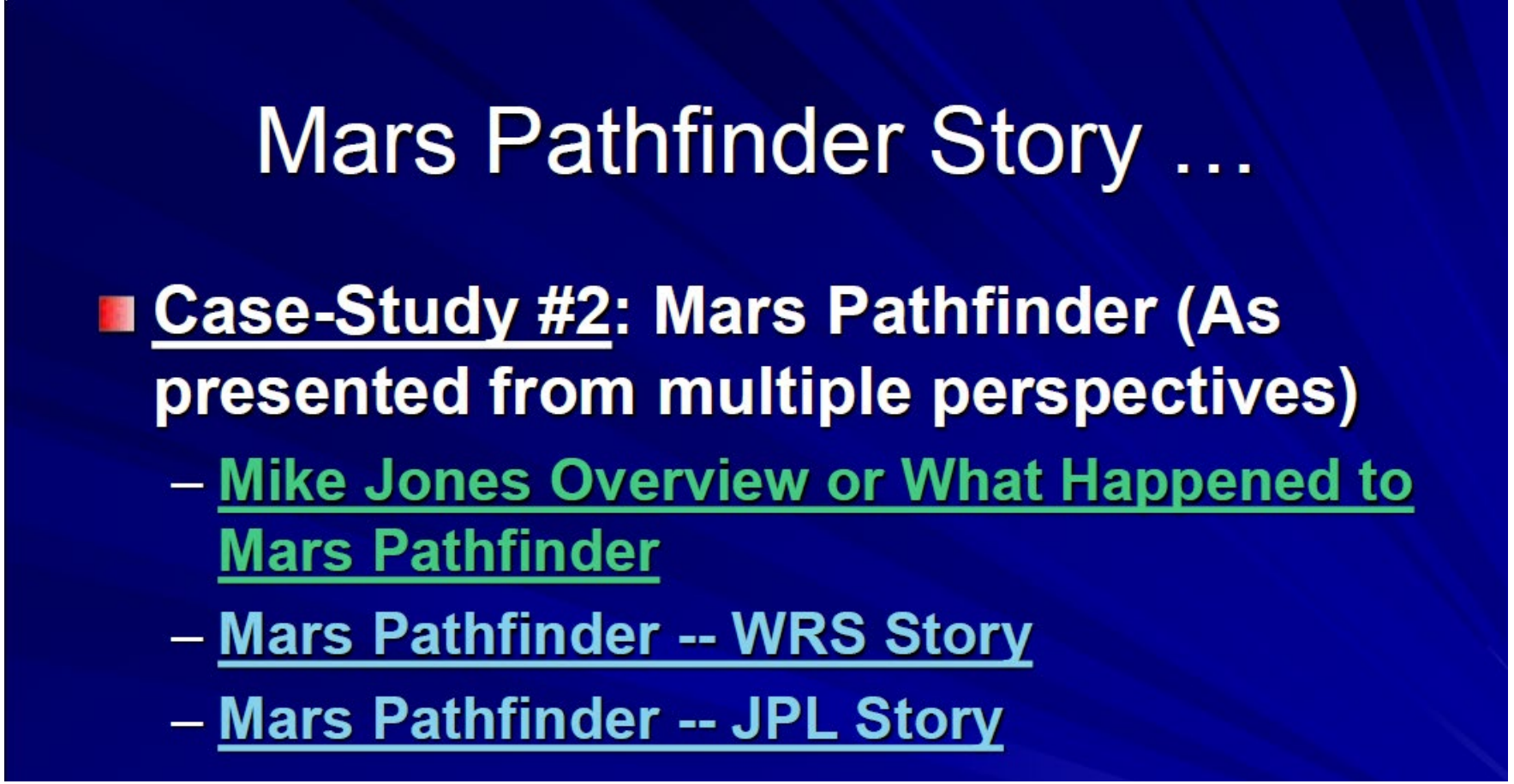
http://ecee.colorado.edu/~siewerts/marspath/jpl/index.htm

NASA's Perseverance Rover Successfully Lands on Mars

NASA's Perseverance Rover

NASA's Perseverance Rover





http://ecee.colorado.edu/~siewerts/marspath/mars.html

http://ecee.colorado.edu/~siewerts/marspath/wrs/index.htm

http://ecee.colorado.edu/~siewerts/marspath/jpl/index.htm

Volunteers Needed!

Organization	Position	Who?
NASA	Ground Control Engineer	Shweta
	Program Manager	Hardik
JPL	CEO	Suraj
	Project Manager	Ayswariya
	Hardware Engineer	Parth
	Software Engineer	Satish
Wind River	CTO	Akshith
	Software Engineer	Ritika
	Firmware Engineer	Krish

Mars Pathfinder Story …

- **Case-Study #2: Mars Pathfinder (As presented from multiple perspectives)**
  - **Mike Jones Overview or What Happened to Mars Pathfinder**
  - **Mars Pathfinder -- WRS Story**
  - **Mars Pathfinder -- JPL Story**

© Sam Siewert

Mars Pathfinder Story …

- **Case-Study #2: Mars Pathfinder (As presented from multiple perspectives)**
  - **Mike Jones Overview or What Happened to Mars Pathfinder**
  - **Mars Pathfinder -- WRS Story**
  - **Mars Pathfinder -- JPL Story**

© Sam Siewert

Mars Pathfinder Story …

- **Case-Study #2: Mars Pathfinder (As presented from multiple perspectives)**
  - **Mike Jones Overview or What Happened to Mars Pathfinder**
  - **Mars Pathfinder -- WRS Story**
  - **Mars Pathfinder -- JPL Story**

© Sam Siewert

Mars Pathfinder Story …

- **Case-Study #2: Mars Pathfinder (As presented from multiple perspectives)**
  - **Mike Jones Overview or What Happened to Mars Pathfinder**
  - **Mars Pathfinder -- WRS Story**
  - **Mars Pathfinder -- JPL Story**

© Sam Siewert

Lessons Learned - Pathfinder

- How you respond to errors is just as important as trying to create designs without them.
- Remote update can be very helpful and needs to be planned for in advance.
- Difficult problems with tight time constraints require teams to work together for a successful conclusion. Egos must be set aside.
- Producing good documentation is important, as is reading it.
- Plan for the unexpected, as it usually happens in complex product design.
- Embedded systems should have watchdog timers as software sanity is difficult to guarantee.
- It's good to have a test bench complete with an exact replica of the product.
- Test your system with real data if you have it.