Paul E. McKenney, Meta Platforms Kernel Team

Huawei Global Software Technology Summit, May 31, 2023

Cautionary Tales on Implementing the Software That People Want

Be Careful What You Wish For. You Might Get It!!!

How Did Paul Get This Way?

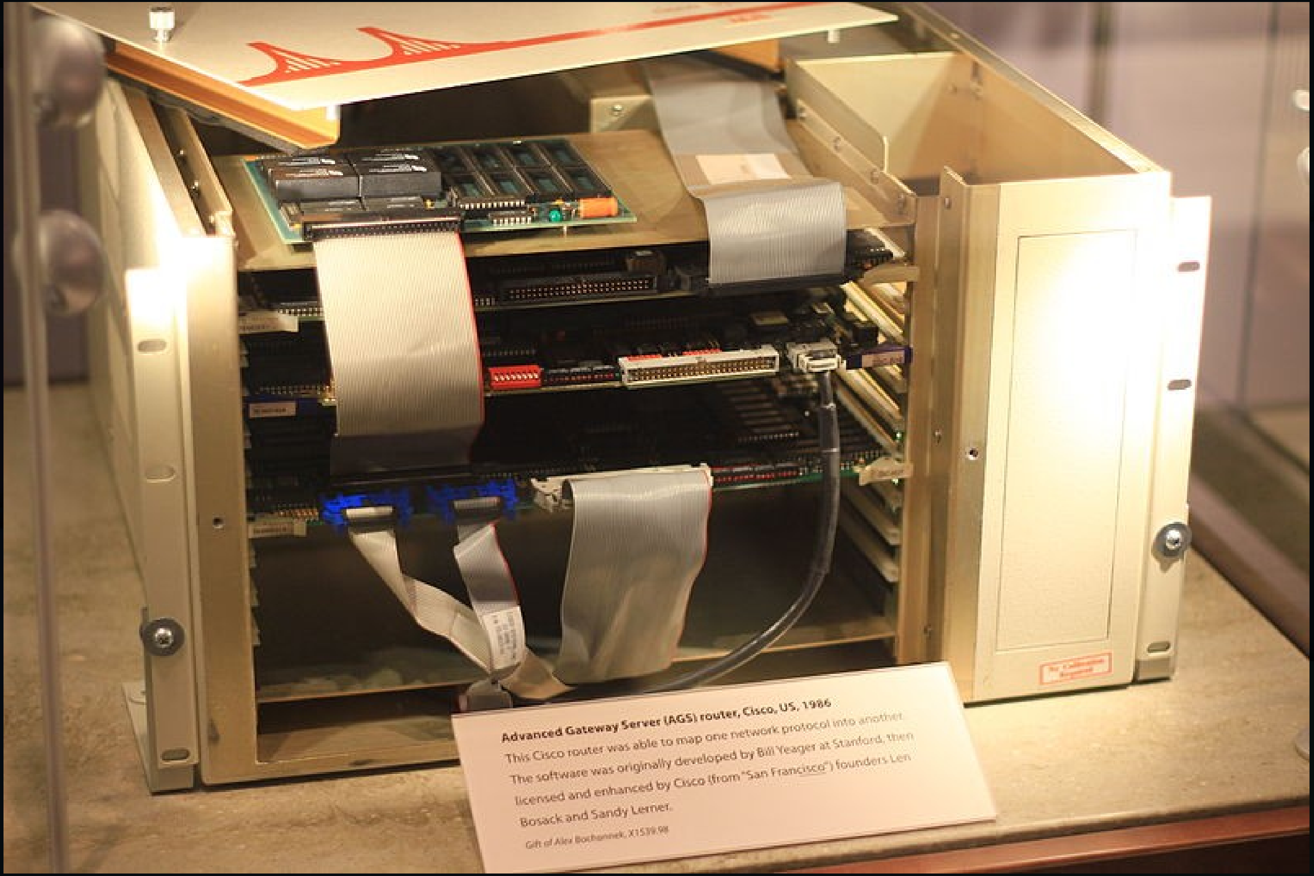
- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
  - Started supporting self by coding in June 1977
- Contract programming (1981-1985)
- Systems administration and Internet research (1986-1990)
- Concurrent proprietary UNIX (1990-2000)
- Linux kernel concurrency and realtime (2001-present)

Cautionary Quotes

- The first secret of getting what you want is knowing what you want. *Arthur D. Hlavaty*
- If you don't know what you want, you will probably never get it. *Oliver Wendell Holmes, Jr.*
- If you don't know what you want, you end up with a lot you don't. *Chuck Palahniuk*

1990: Stochastic Fairness Queueing

1990: Stochastic Fairness Queueing



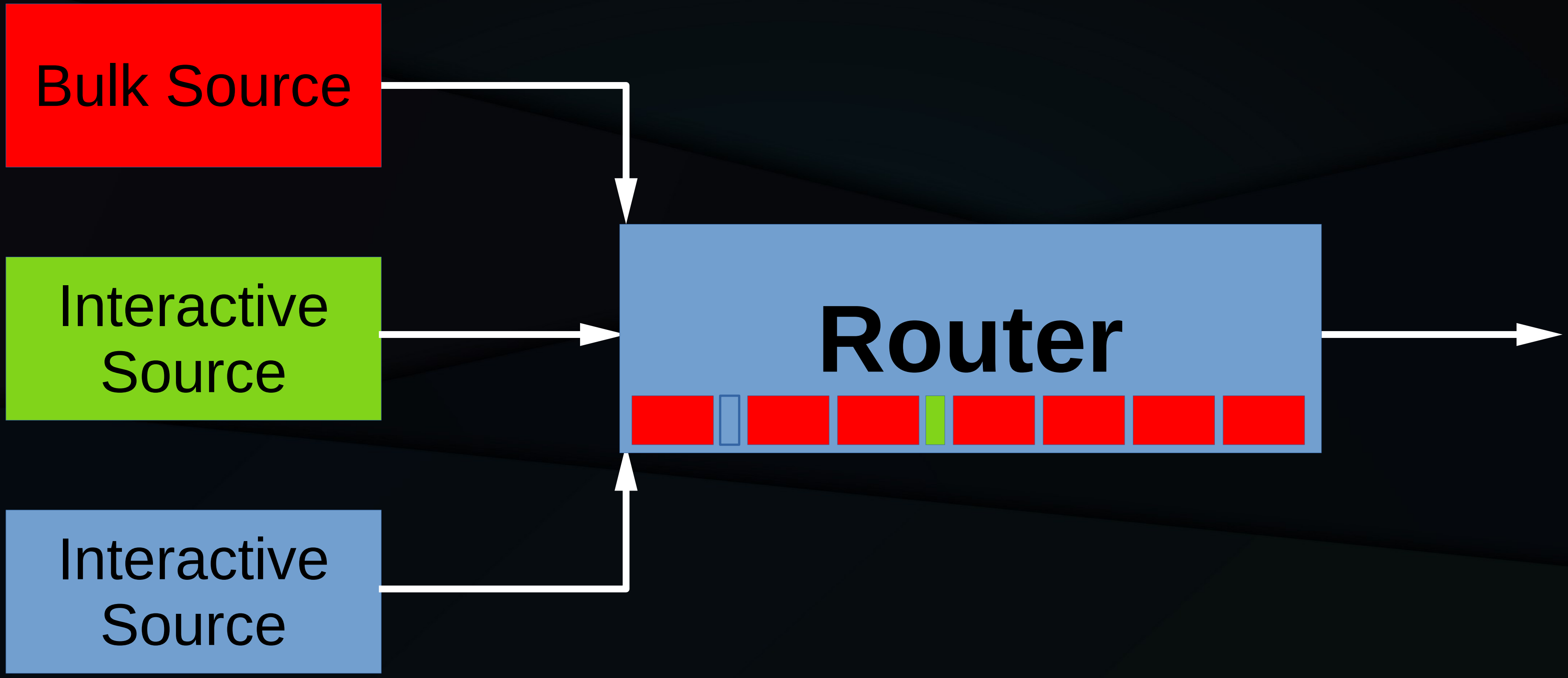
Advanced Gateway Server (AGS) router, Cisco, US, 1986

This Cisco router was able to map one network protocol into another. The software was originally developed by Bill Yeager at Stanford, then licensed and enhanced by Cisco (from "San Francisco") founders Len Bosack and Sandy Lerner.

Gift of Alex Bochannek, X1539.98

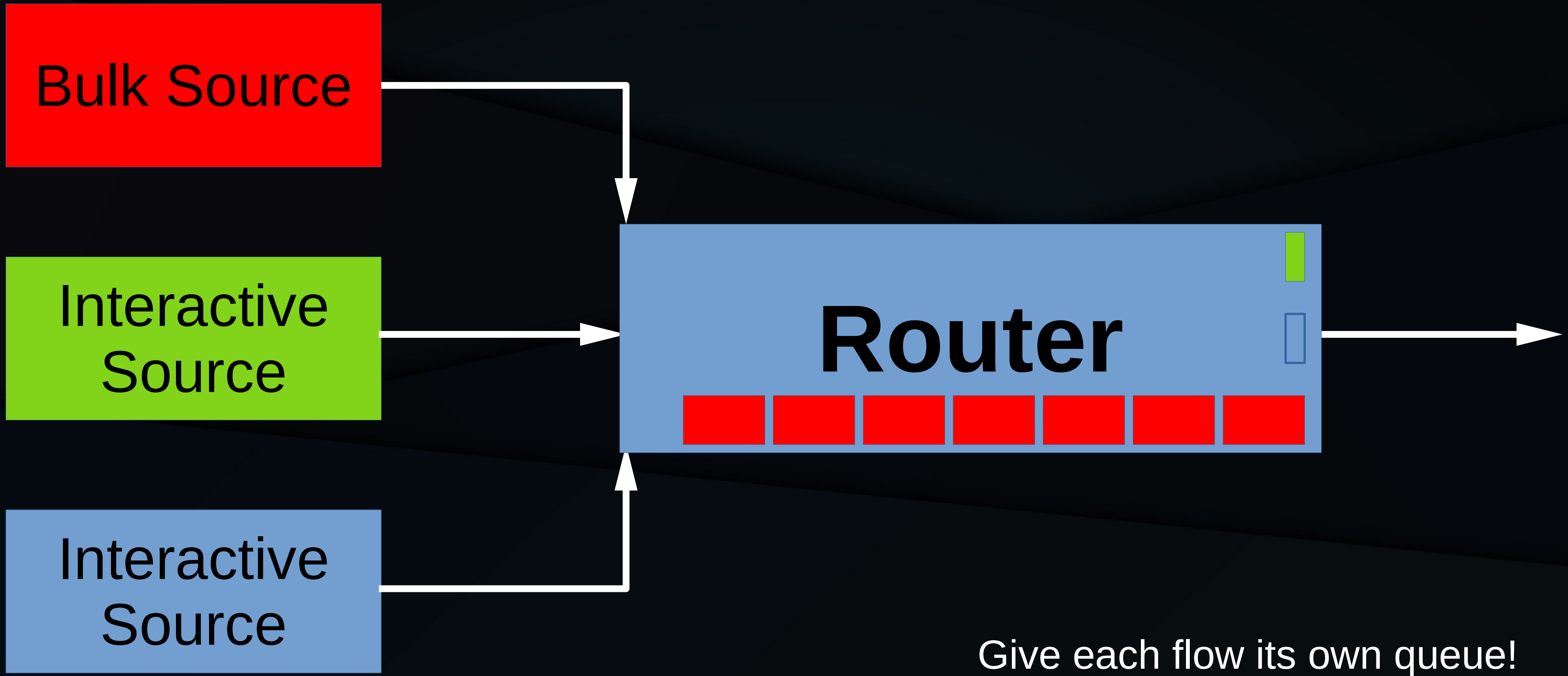
Photo: Michael Hicks, Creative Commons Attribution 2.0 Generic

1990: Queueing Problem



1 Megabit network is *fast*.

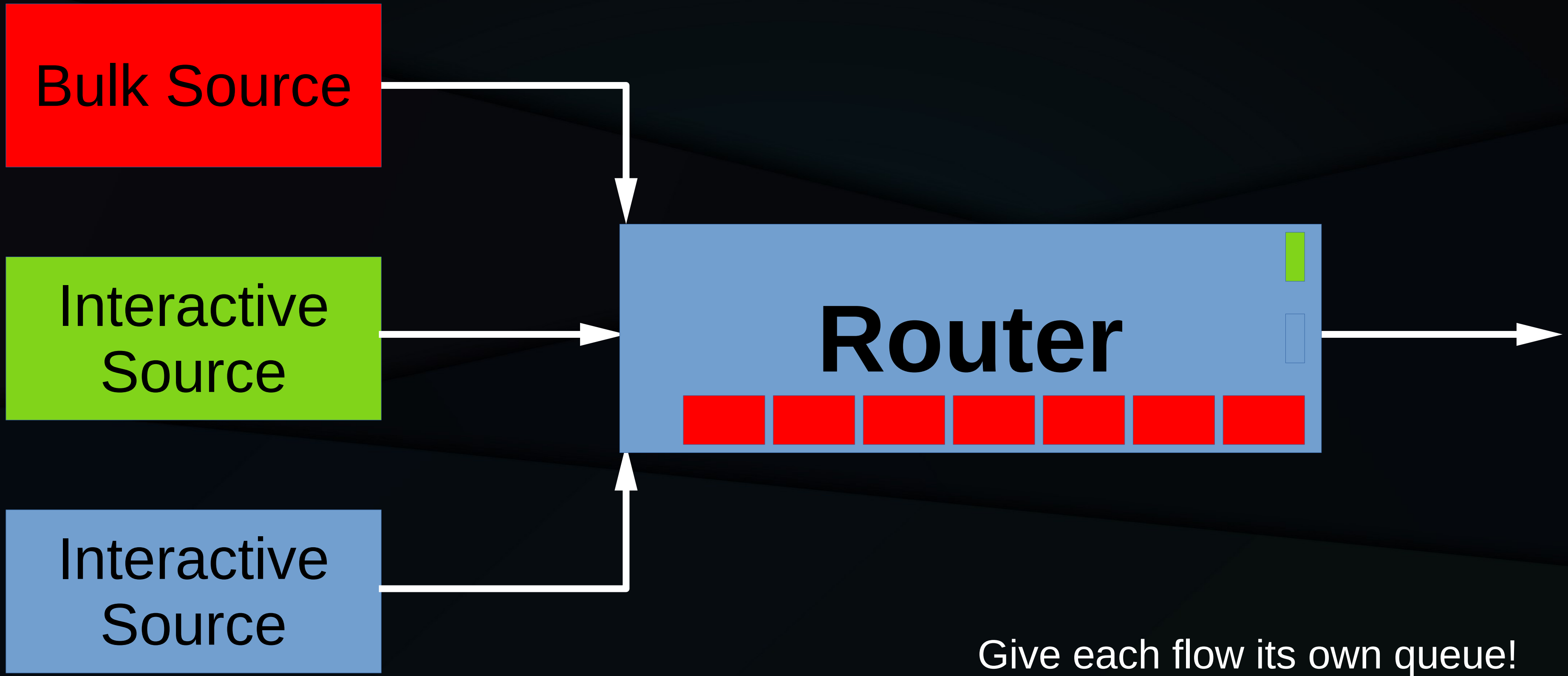
1990: Fair Queueing



Give each flow its own queue!

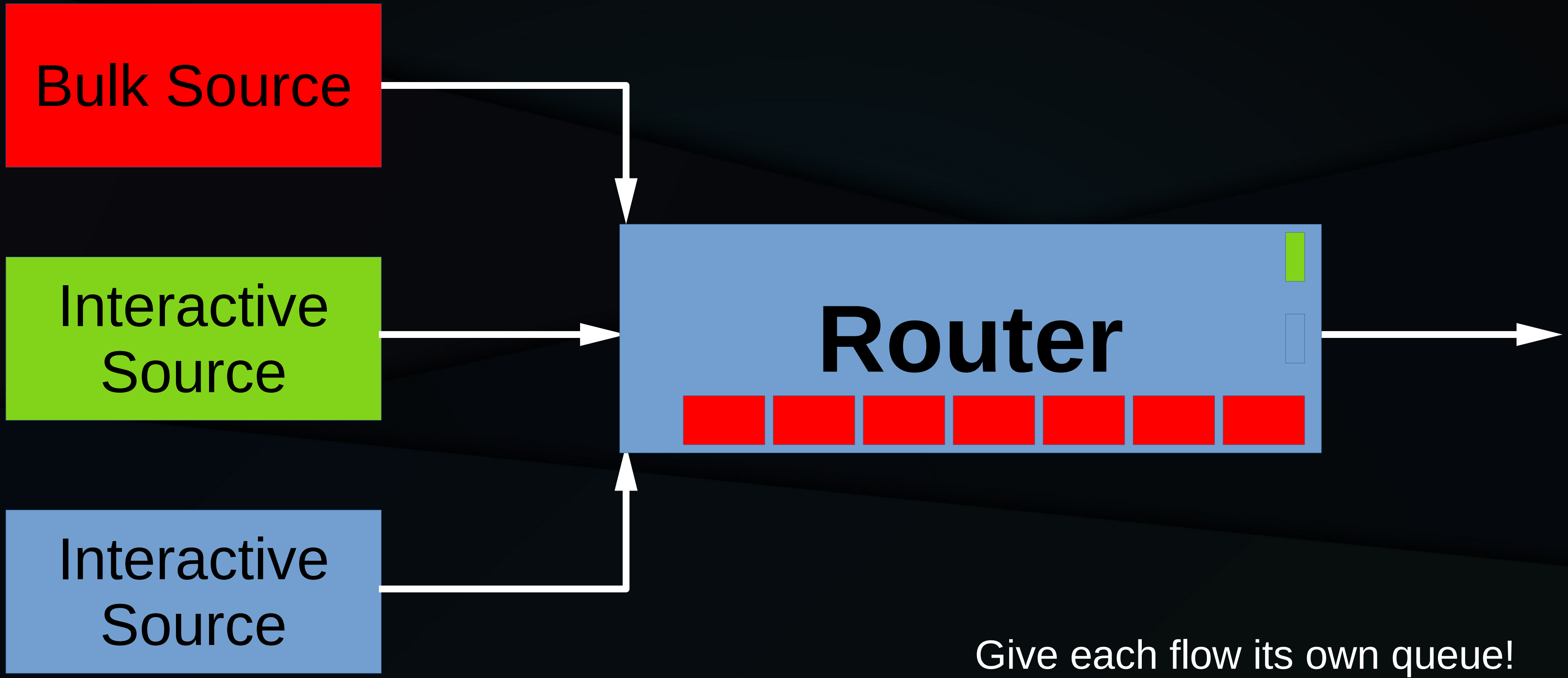
Yeah, you and how many 10MHz CPUs???

1990: Fair Queueing



Give each flow its own queue!
Yeah, you and how many 10MHz CPUs???

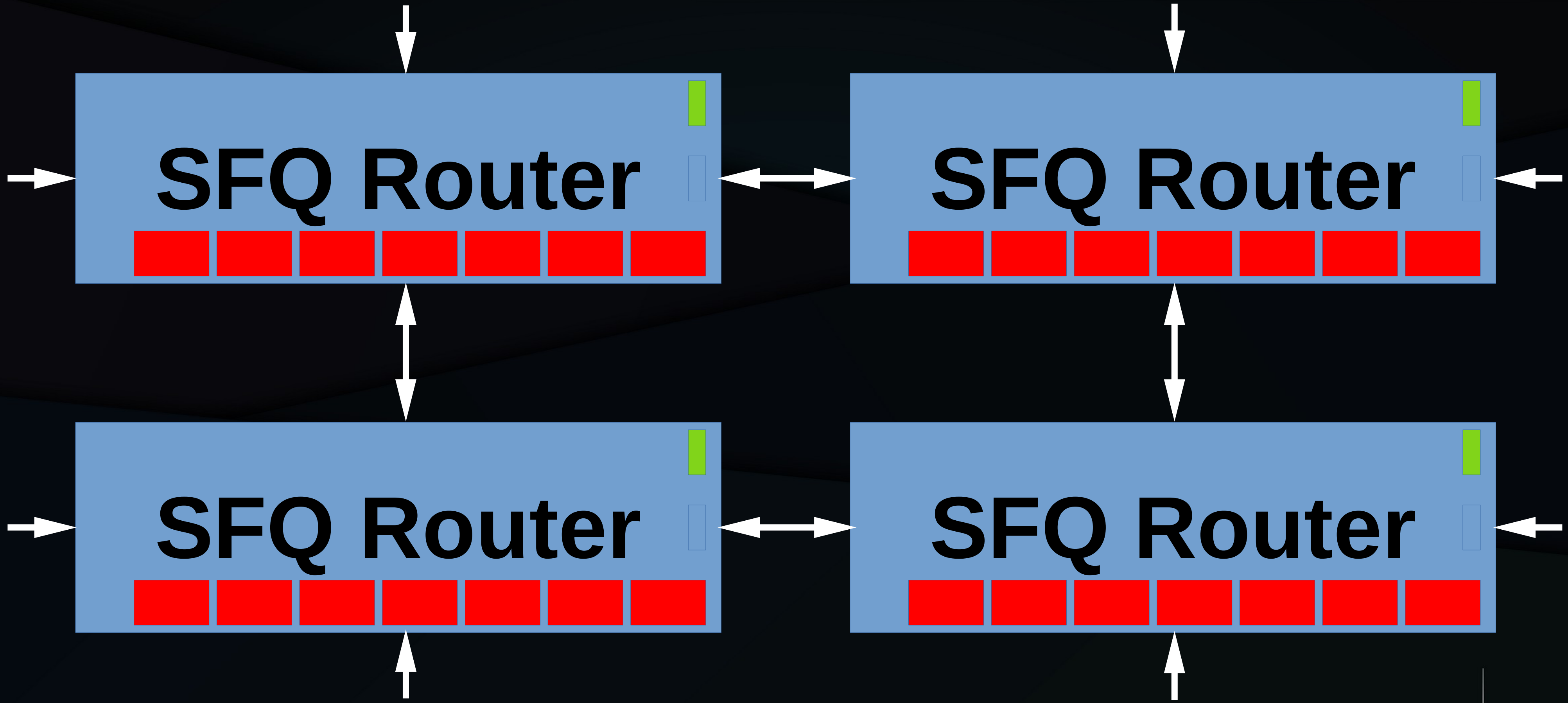
1990: Stochastic Fair Queueing: Hash



Give each flow its own queue!
But only with high probabilty!!!

Hash IP-address/Port quadruple for wonderous end-to-end fairness!!!

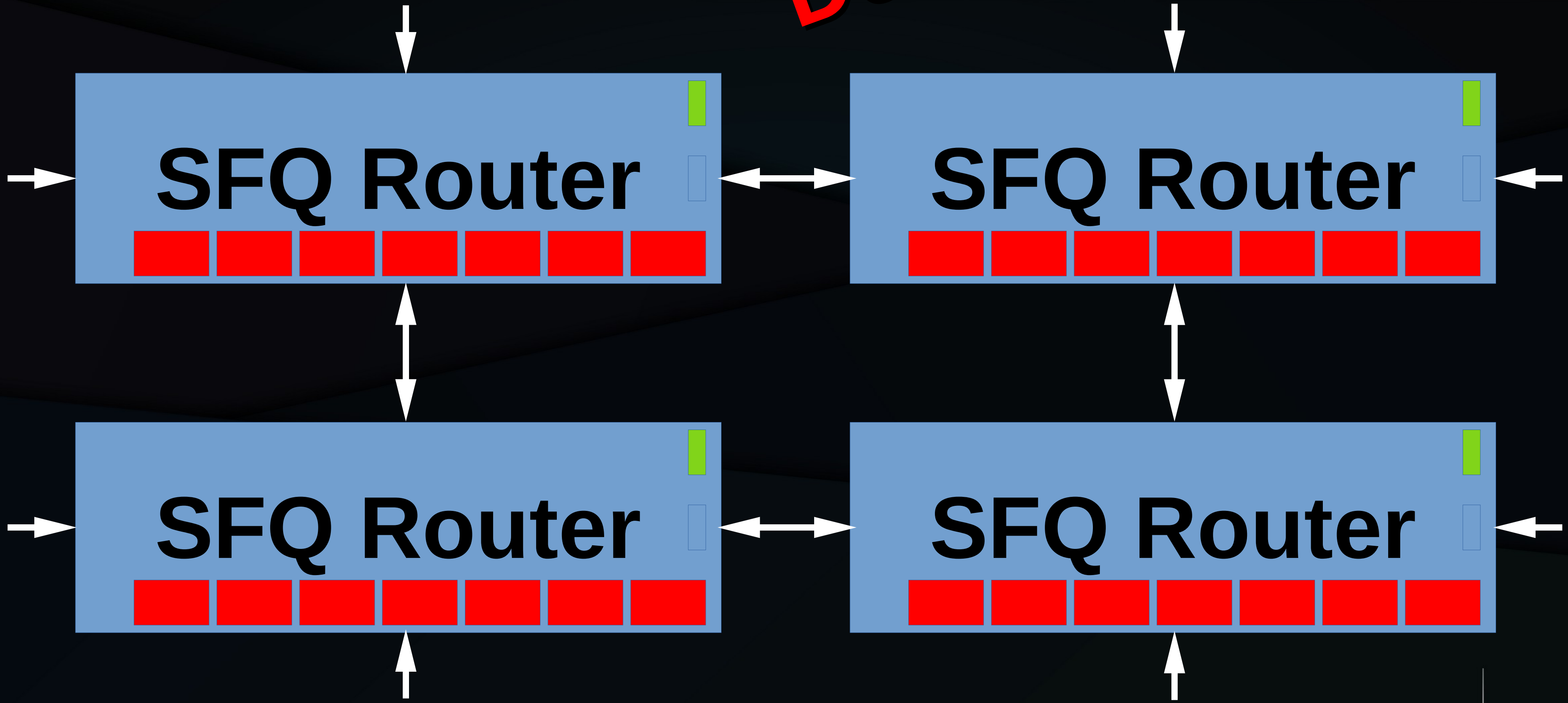
1990: Paul’s Internet Vision



Hash IP-address/Port quadruple for wonderous end-to-end fairness!!!

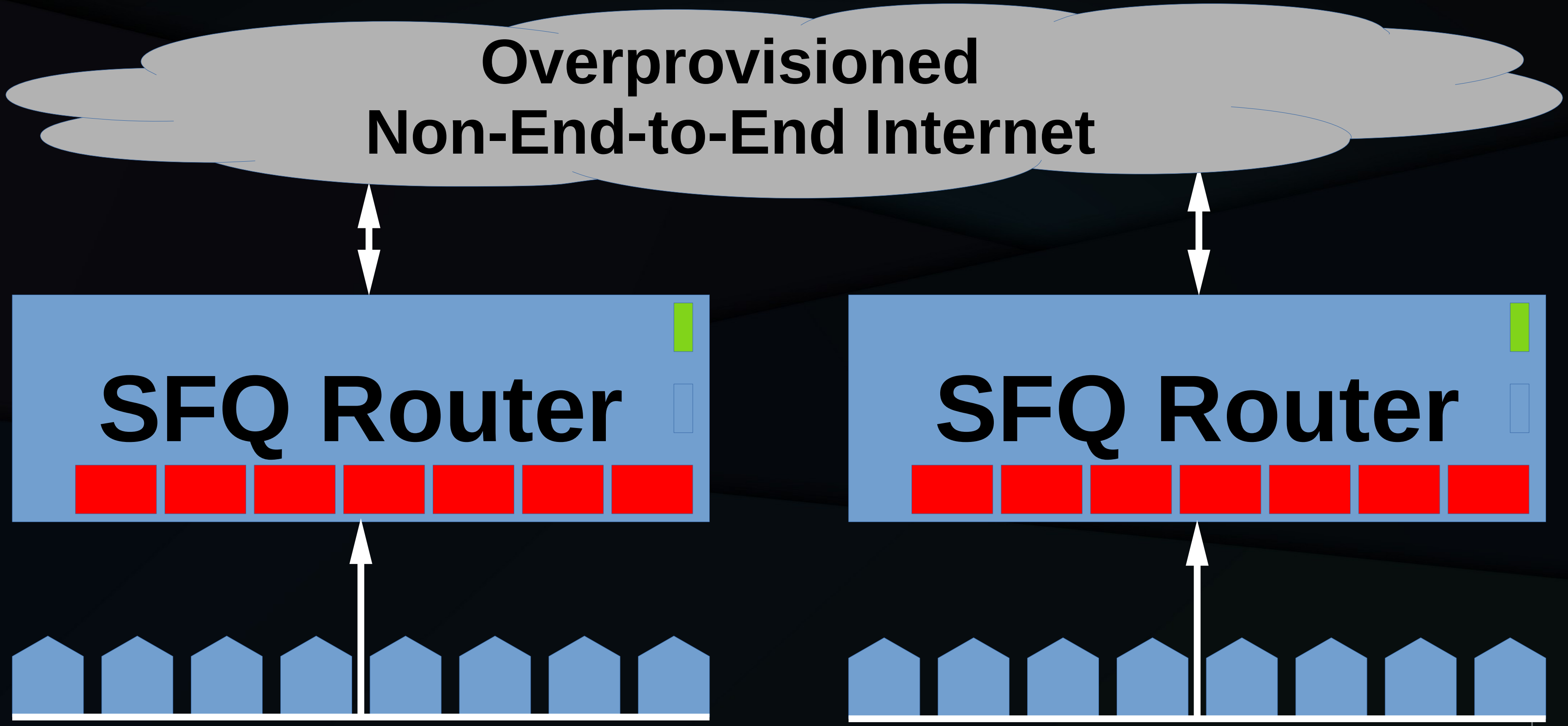
1990: Paul's Internet Vision

Delusion



~~Hash IP-address/Port quadruple for wonderous end-to-end fairness!!!~~

1990: What Internet Did Instead



Internet gateways hash Ethernet MAC addresses for approximate real-world fairness.

1990 SFQ: What Went Wrong?

- Solved wrong problem: End-to-end fairness
  - Correct problem: Hop-by-hop & endpoint fairness
  - By sheer dumb luck, my algorithm handled both
- Research-quality code: Get the paper out!!!
  - Engineers at Cisco and in Linux kernel fixed this
- Used heavily until about 2015 (aside from WISPs)
  - FQ-CODEL and CAKE now address bufferbloat
  - Dave Taht, Eric Dumazet, Toke Høiland-Jørgensen, ...

1990 SFQ: What Went Wrong?

- Solved wrong problem: End-to-end fairness
  - Correct problem: Hop-by-hop & endpoint f s
  - By sheer dumb luck, my algorithm h both
- Research-quality code: Get aper out!!!
  - Engineers at Cisco an ux kernel fixed this
- Used heavily un ut 2015 (aside from WISPs)
  - FQ-COD CAKE now address bufferbloat
  - Dav , Eric Dumazet, Toke Høiland-Jørgensen, ...

Bad idea badly implemented

1990 SFQ: What Went Wrong?

- Solved wrong problem: End-to-end fairness
  - Correct problem: Hop-by-hop & endpoint f
  - By sheer dumb luck, my algorithm h
- Research-quality code: Get t!!!
  - Engineers at Cisco an fixed this
- Used heavily un (aside from WISPs)
  - FQ-CODE w address bufferbloat
  - Dav nazet, Toke Høiland-Jørgensen, ...

Bad idea badly implemented,
resuscitated by dumb luck

1990 SFQ: What Went Wrong?

- Solved w g problem: End-to-end fairness
  - Correct pr Hop-by-hop & endpoint f
  - By sheer dumb l algorithm h
- Research-quality t!!!
  - Engineers at Cisco a fixed this
- Used heavily un m WISPs)
  - FQ-COD ew addre ifer
  - Dav mazet, Toke Høiland-Jørgens .

Premature abstraction is the

root of all evil

Bad idea ba mplemented,

resuscitate dumb luck

1990 SFQ: What Went Wrong?

- Solved w g problem: End-to-end fairness
  - Correct pr Hop-by-hop & endpoint f
  - By sheer dumb l y algorithm h
- Research-quality t!!!
  - Engi
- Used heavily u m WISPs)
  - FQ-COD w addr iffe
  - Dav mazet, Toke Høiland-Jørgens .

Premature abs on is the

roo all evil

Bad idea b mplemented,

resuscitate dumb luck

Live among your users!!!

1980s: Eight-Bit CRM

Permission to use granted February 5, 2023 by Olivier Boisseau of OLD-COMPUTERS.COM
Image source: https://www.old-computers.com/museum/photos/otrona_attache_1s.jpg

1980s: Eight-Bit CRM

- CRM application built to spec under contract
- The company loved it!

1980s: Eight-Bit CRM

- CRM application built to spec under contract
- The company loved it!
- Their prospective customers, not so much

1980s: Eight-Bit CRM

- CRM application built to spec under contract
- The company loved it!
- Their prospective customers, not so much
- Dumb luck: They paid me *before* bankruptcy

1980s: Eight-Bit CRM

- CRM application built to spec und[illegible]ntract
- The company loved it!
- Their prospective cus[illegible]rs, not so much
- Dumb luck: Th[illegible]d me *before* bankruptcy

Bad idea well implemented

1980s: Eight-Bit CRM

- CRM application built to spec und ntract
- The company loved it!
- Their prospective cus so much
- Dumb luck: Th *efore* bankruptcy

Bad idea well implemented,
but hey, I got paid???

1980s: Eight-Bit CRM: What Instead?

Time and Grade: Experience

Cautionary Quote

- "Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?" - *Brian W. Kernighan*, "The Elements of Programming Style", 2nd Edition, Chapter 2.

Cautionary Quote

- "Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?" - *Brian W. Kernighan*, "The Elements of Programming Style", 2nd Edition, Chapter 2.
- While programming, you are living in blissful ignorance of important requirements. These requirements make themselves known during debugging.
- Which is but one cause of Kernighan’s observation.

Cautionary Quote

- "Everyone knows that debugging is twice as har ng a program in the first place. So if you're as cl n be when you write it, how will you ever de *Kernighan*, "The Elements of Pro 2nd Edition, Chapter 2.
- While programming lissful ignorance of important requi equirements make themselves known du
- Whic se of Kernighan's observation.

I failed to understand that I was competing with a file cabinet

Cautionary Quote

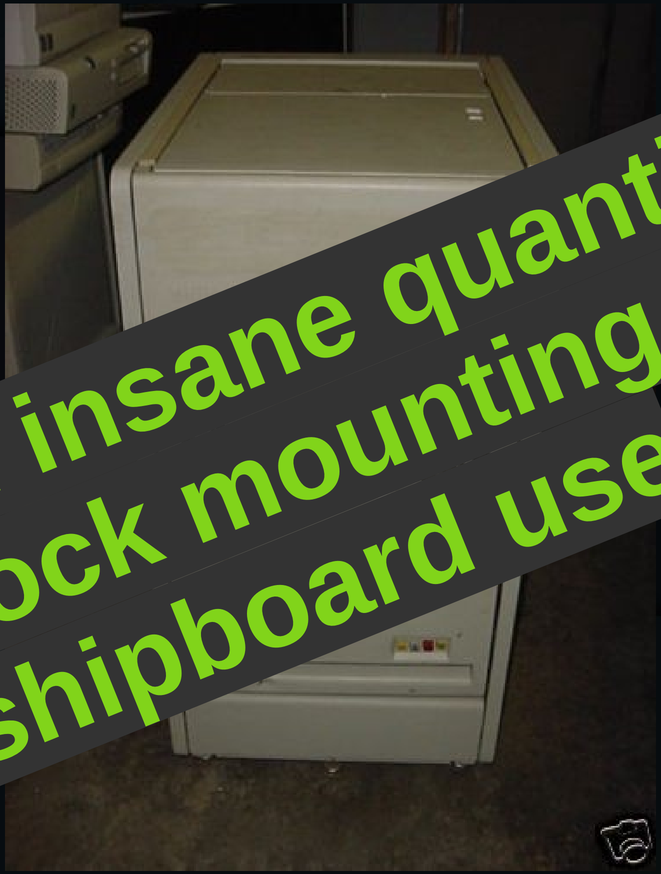
- "Everyone knows that debugging is twice as har ng a program i st place. So if you're as cl n be when you wri will you ever d *Kernighan*, "The E of Pro 2nd Edition, Chapter 2.
- While programming ssful ignorance of important requi equ make themselves known du
- Whic se of Kernighan's observati .

I failed to und stand that I was competing wit file cabinet

And the file cabinet won

1980s: Acoustic Navigation

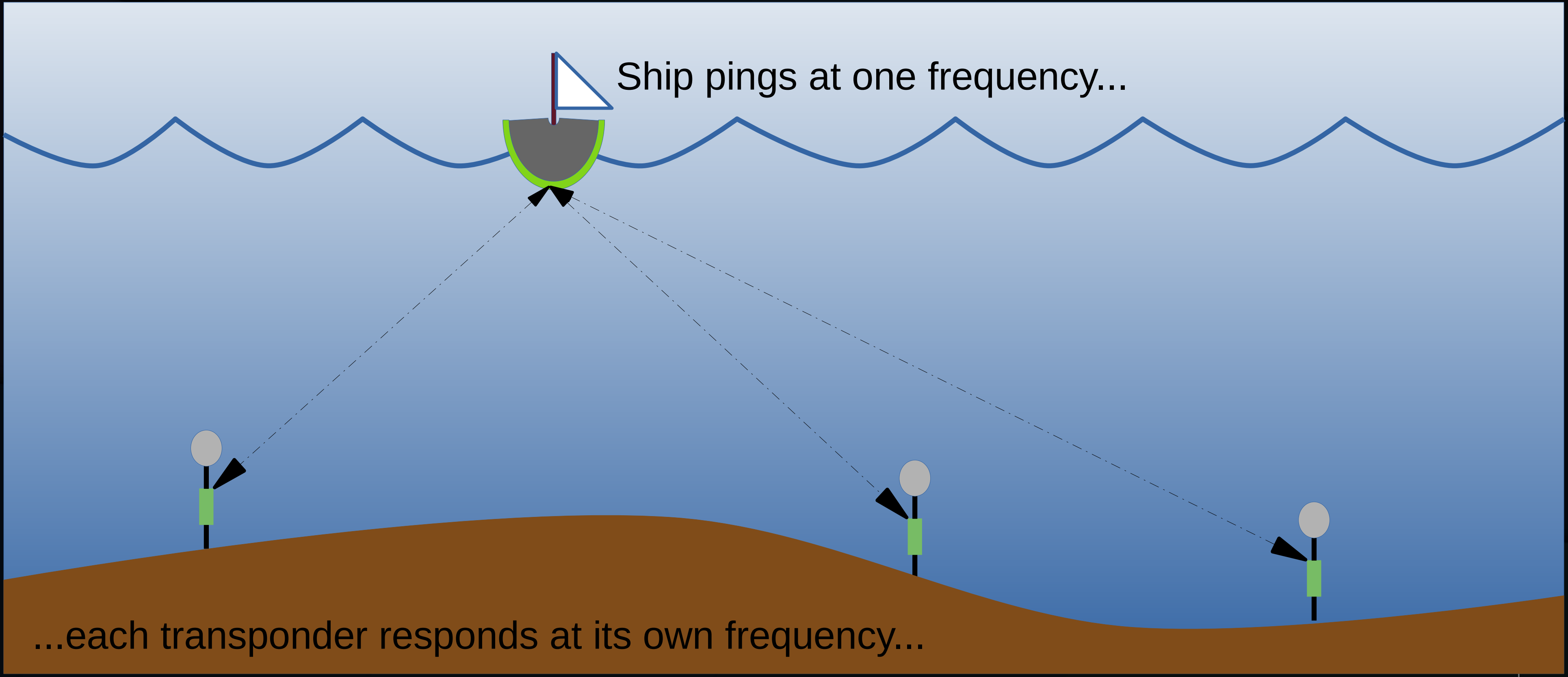
1980s: Acoustic Navigation



But with insane quantities of shock mounting for shipboard use

Photo by user Neozeed https://gunkies.org/wiki/File:Pdp11-23.jpg GNU Free Documentation License 1.2

1980s: Acoustic Navigation (Pre-GPS)


Ship pings at one frequency...
...each transponder responds at its own frequency...

...then convert time to distance and triangulate!!!

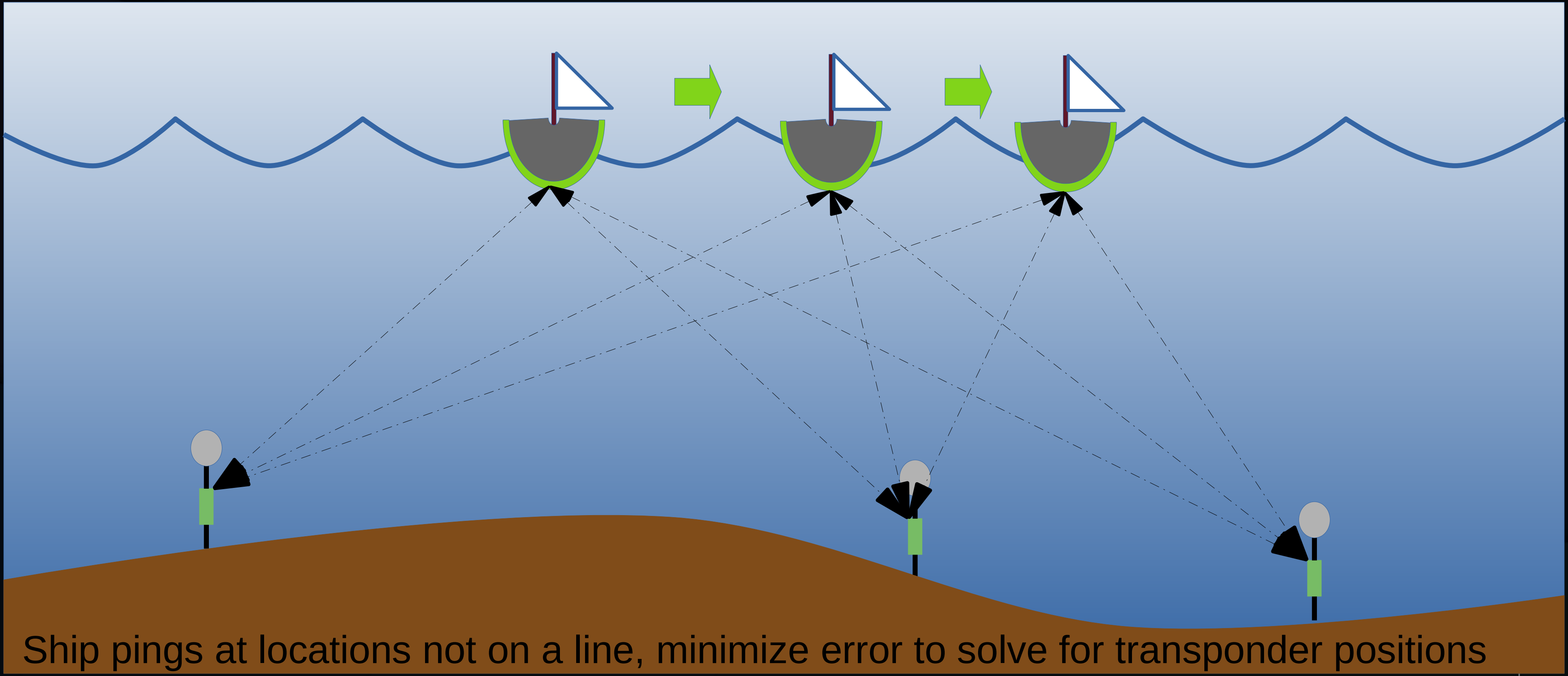
Acoustic Navigation Complications

- If the ship’s position was known when deploying the transponder, there would be no need for acoustic navigation
- Transponders do not fall exactly straight down through four miles of water
- Ocean surface is not perfectly level
- Sound does not travel in a straight line through ocean water
- Sound does not travel at a uniform speed through ocean water
- Dolphins like to play with transponders

Acoustic Navigation Complications

- If the ship’s position was known when deploying the transponder, there would be no need for acoustic navigation
- Transponders do not fall exactly straight down through four miles of water
- Ocean surface is not perfectly level
- Sound does not travel in a straight line through ocean water
- Sound does not travel at a uniform speed through ocean water
- Dolphins like to play with transponders

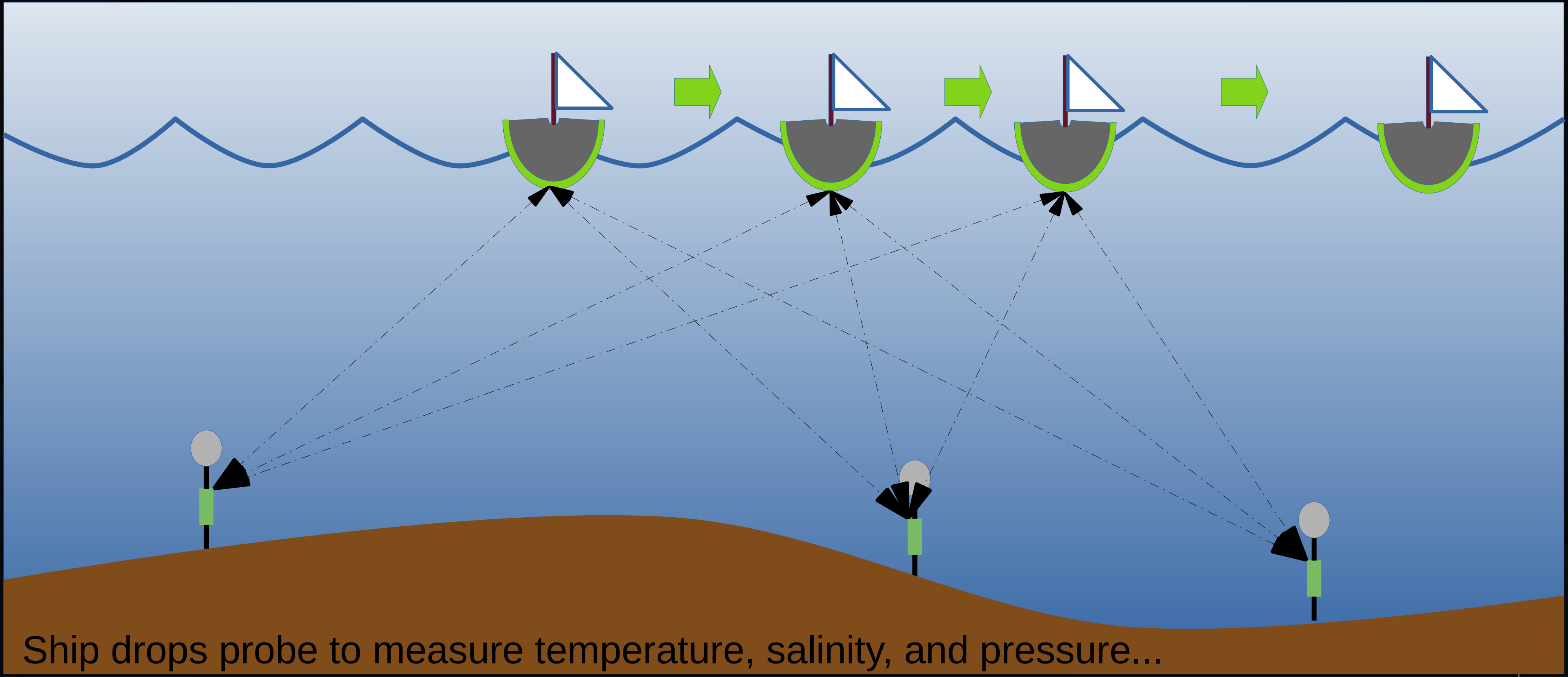
Acoustic Navigation Calibration (1/2)


Ship pings at locations not on a line, minimize error to solve for transponder positions

Acoustic Navigation Complications

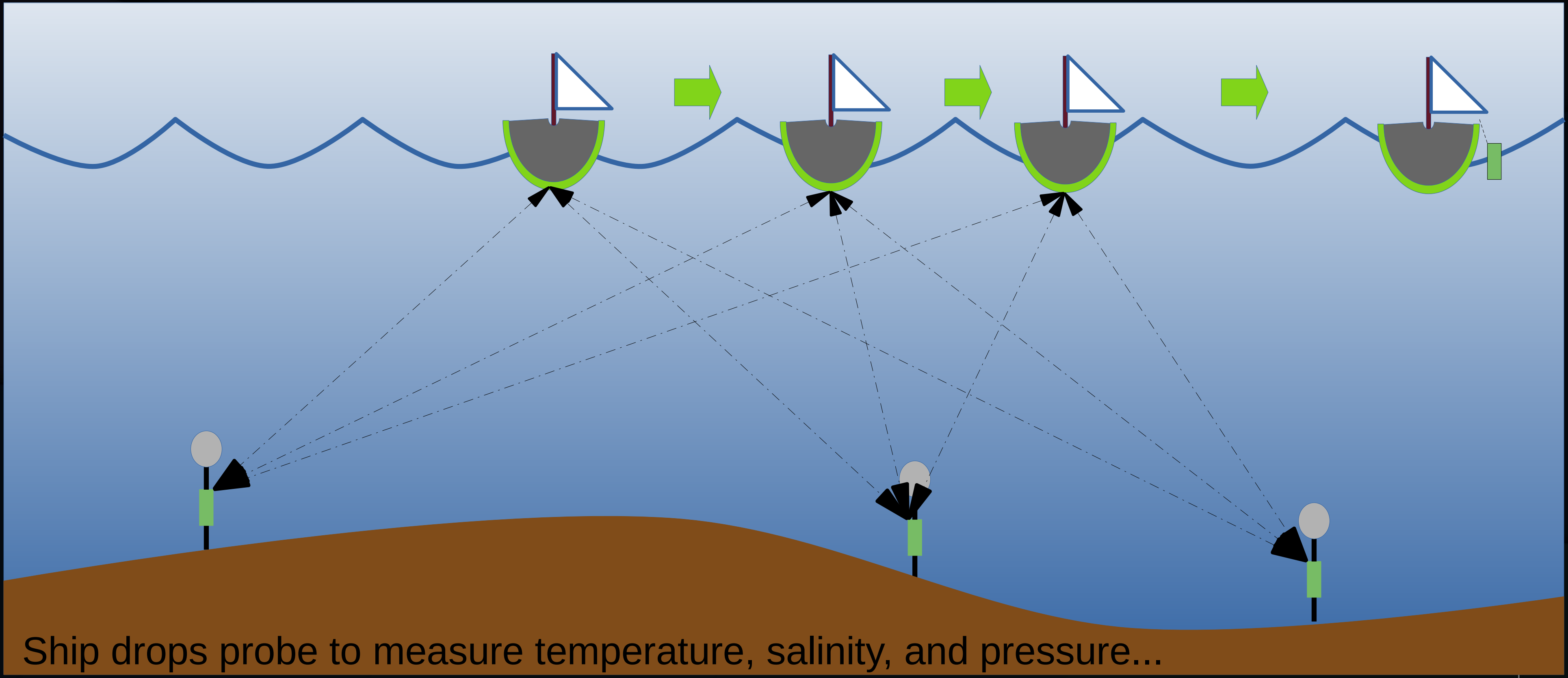
- If the ship’s position was known when deploying the transponder, there would be no need for acoustic navigation
- Transponders do not fall exactly straight down through four miles of water
- Ocean surface is not perfectly level
- Sound does not travel in a straight line through ocean water
- Sound does not travel at a uniform speed through ocean water
- Dolphins like to play with transponders

Acoustic Navigation Calibration (2/2)

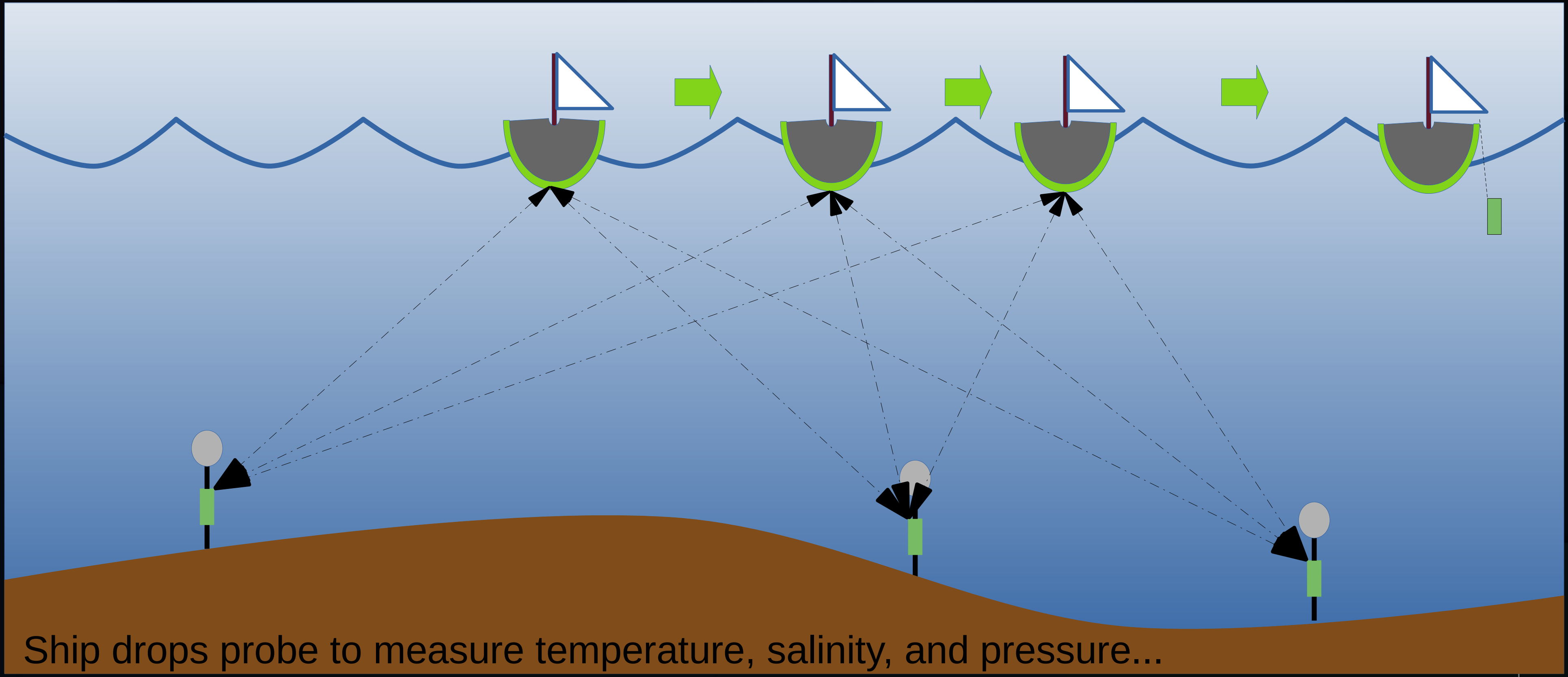


Ship drops probe to measure temperature, salinity, and pressure...

Acoustic Navigation Calibration (2/2)

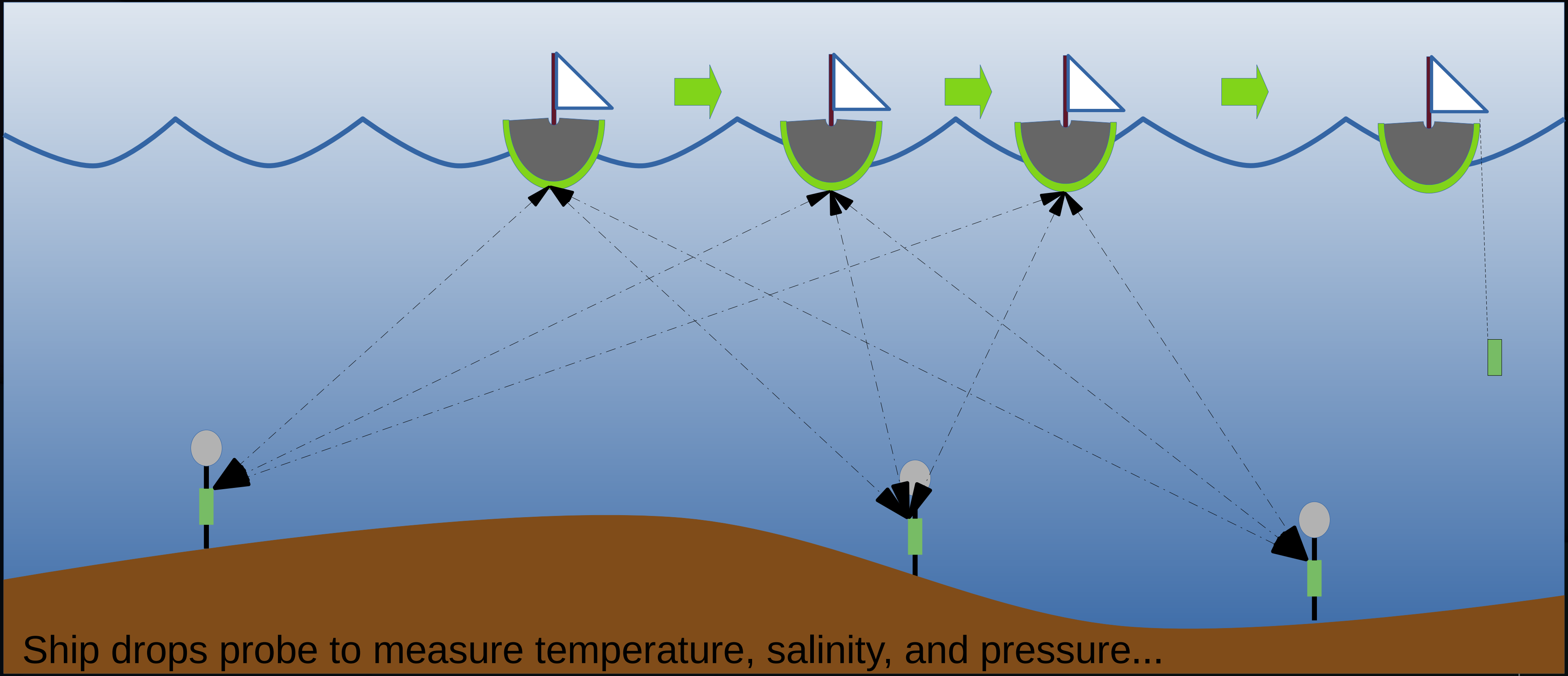

Ship drops probe to measure temperature, salinity, and pressure...

Acoustic Navigation Calibration (2/2)

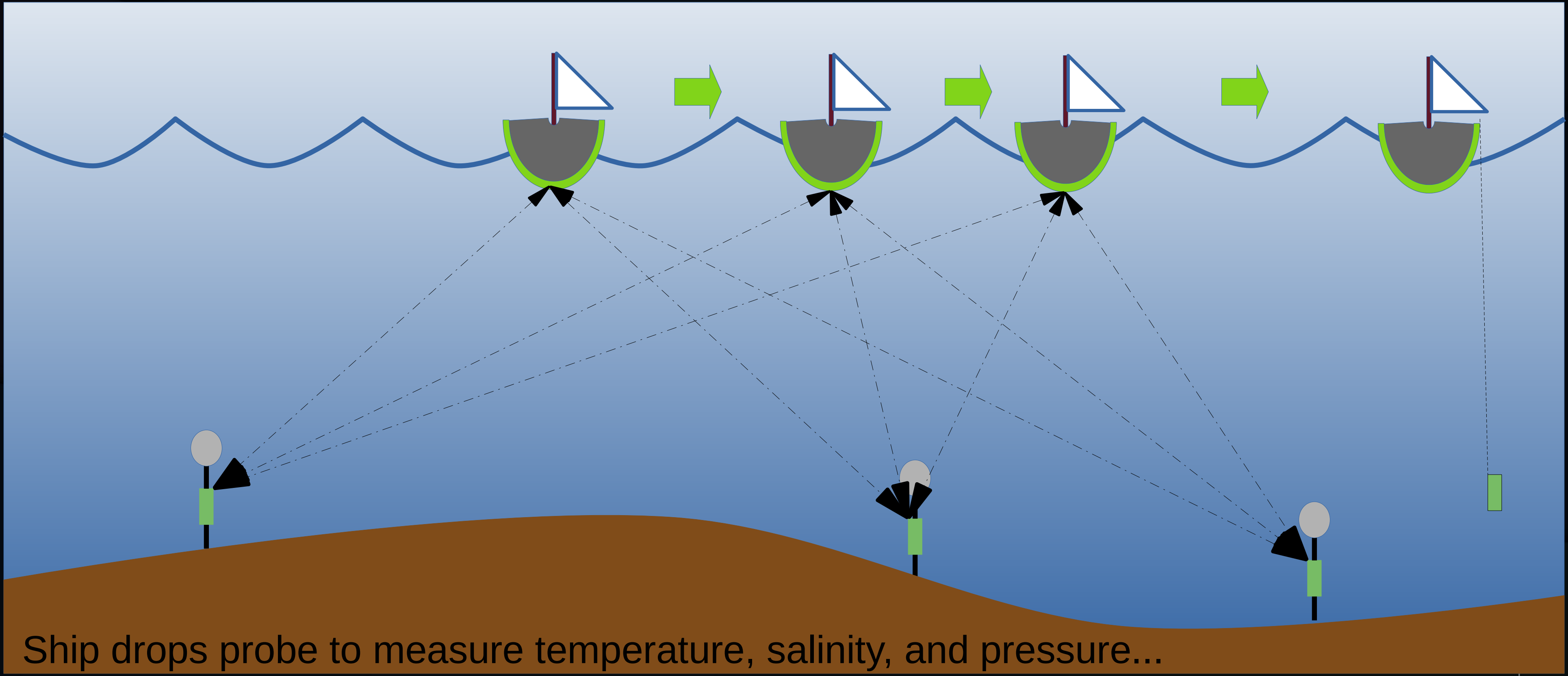


Ship drops probe to measure temperature, salinity, and pressure...

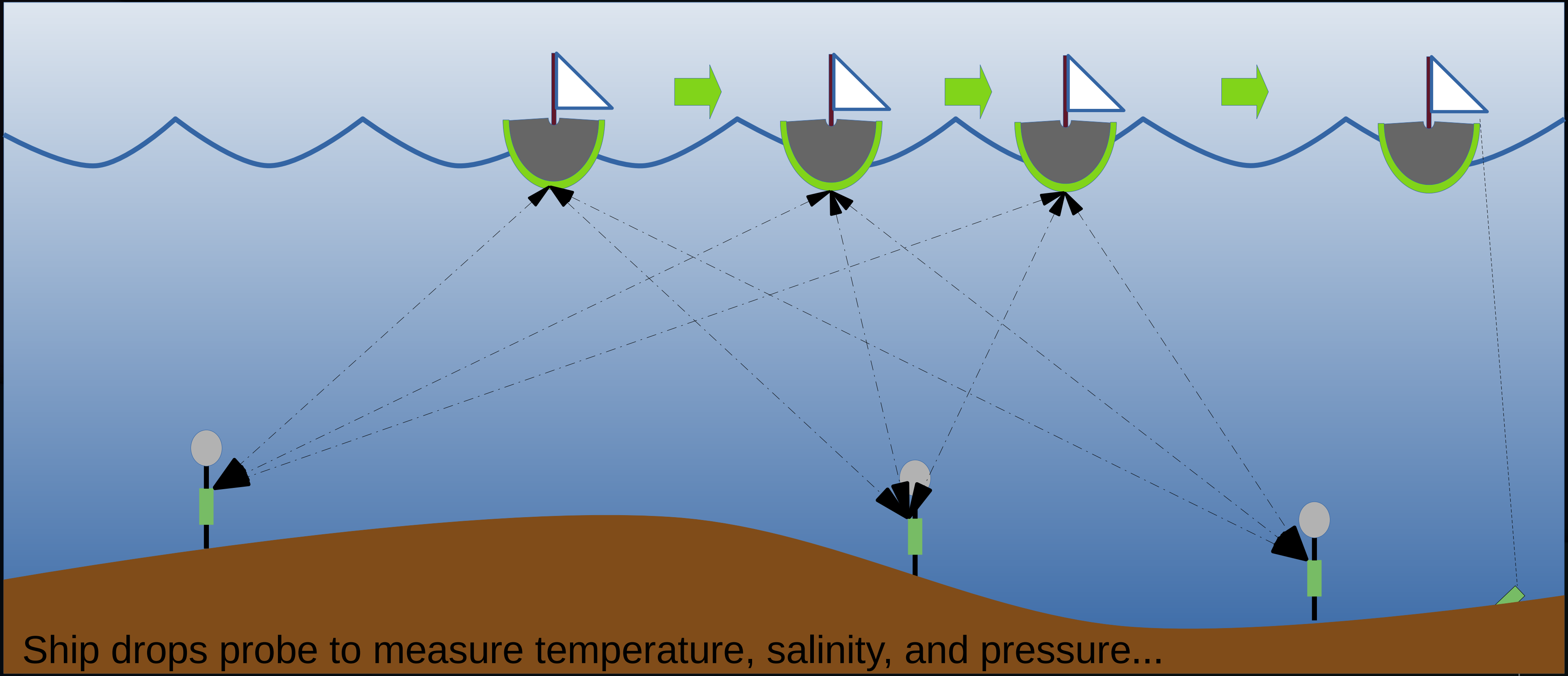
Acoustic Navigation Calibration (2/2)


Ship drops probe to measure temperature, salinity, and pressure...

Acoustic Navigation Calibration (2/2)


Ship drops probe to measure temperature, salinity, and pressure...

Acoustic Navigation Calibration (2/2)


Ship drops probe to measure temperature, salinity, and pressure...

Then calculate sound velocity as a function of depth, and finally do ray-tracing.

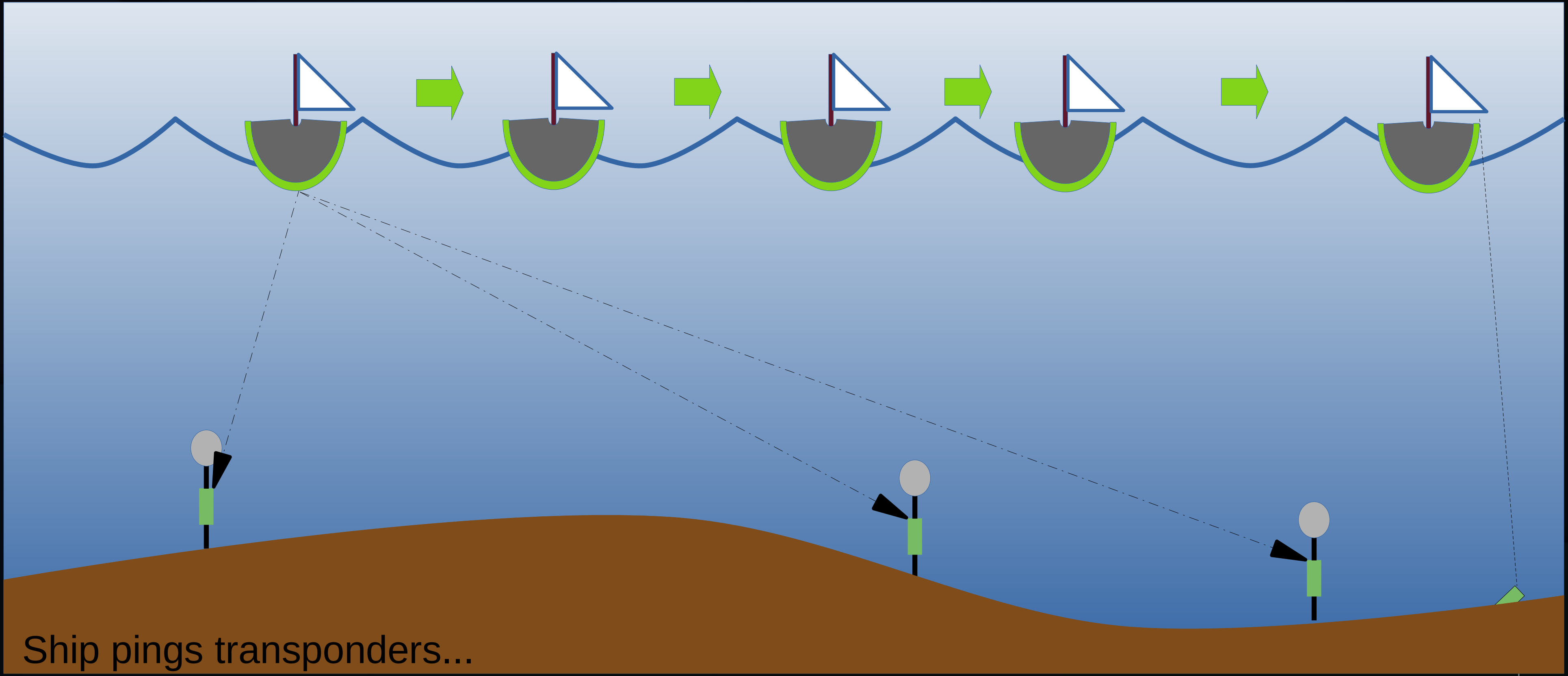
Acoustic Navigation Complications

- If ship’s position known when deploying transponder, no need for system
- Transponders do not fall exactly straight down through four miles of water
- Ocean surface is not perfectly level
- Sound does not travel in a straight line through ocean water
- Sound does not travel at a uniform speed through ocean water
- Dolphins like to play with transponders
- Error minimization has difficulty with three unknowns per transponder

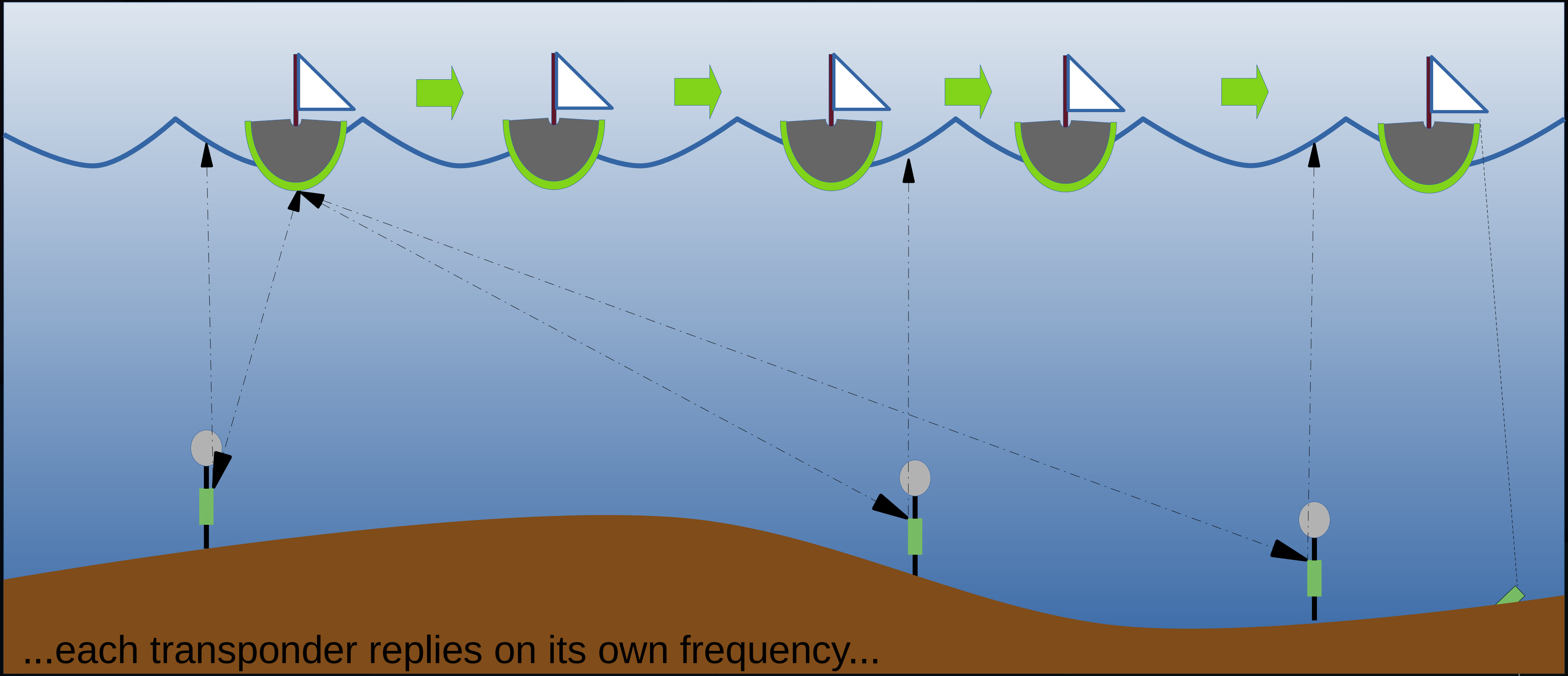
Acoustic Navigation Complications

- If ship's position known when deploying transponder, no need for system
- Transponders do not fall exactly straight down through four miles of water
- Ocean surface is not perfectly level
- Sound does not travel in a straight line through ocean water
- Sound does not travel at a uniform speed through ocean water
- Dolphins like to play with transponders
- Error minimization has difficulty with three unknowns per transponder

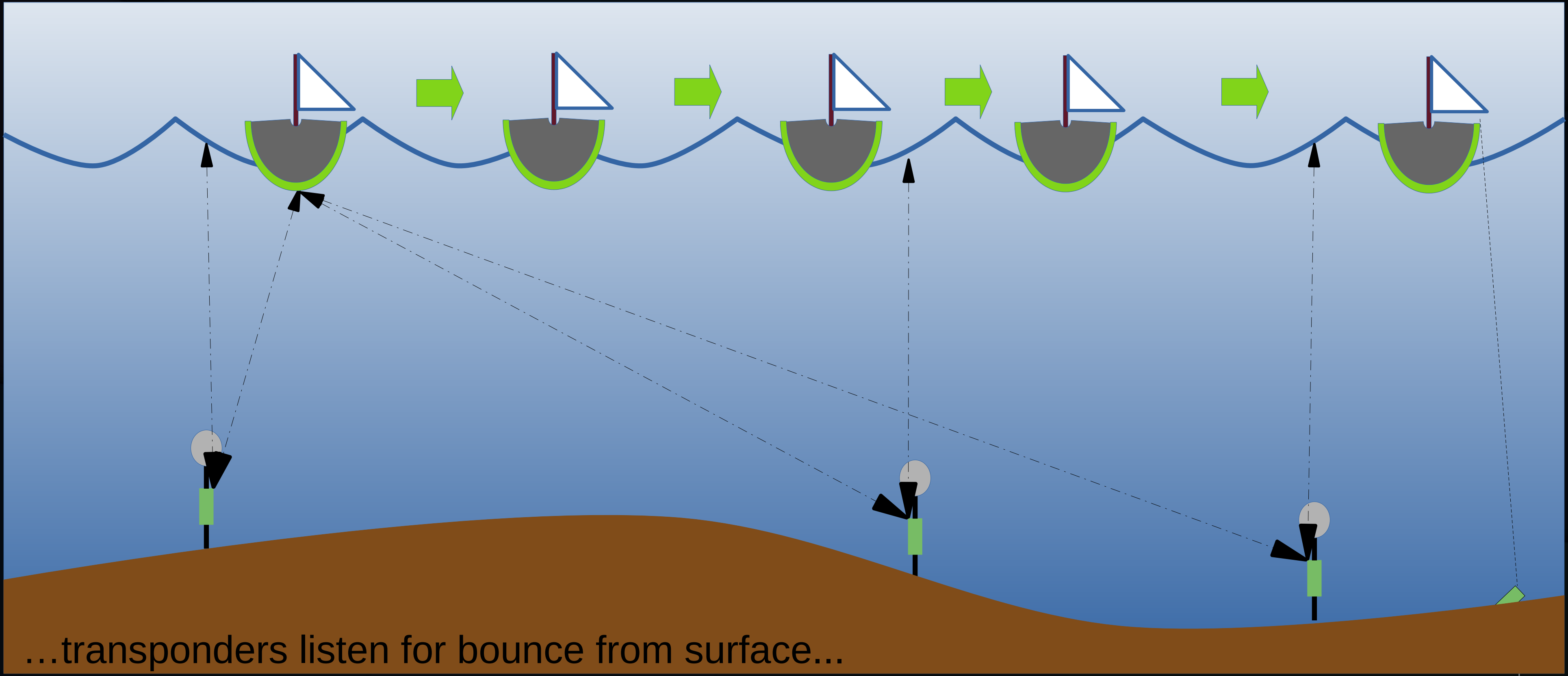
Acoustic Navigation: Measure Depth



Acoustic Navigation: Measure Depth

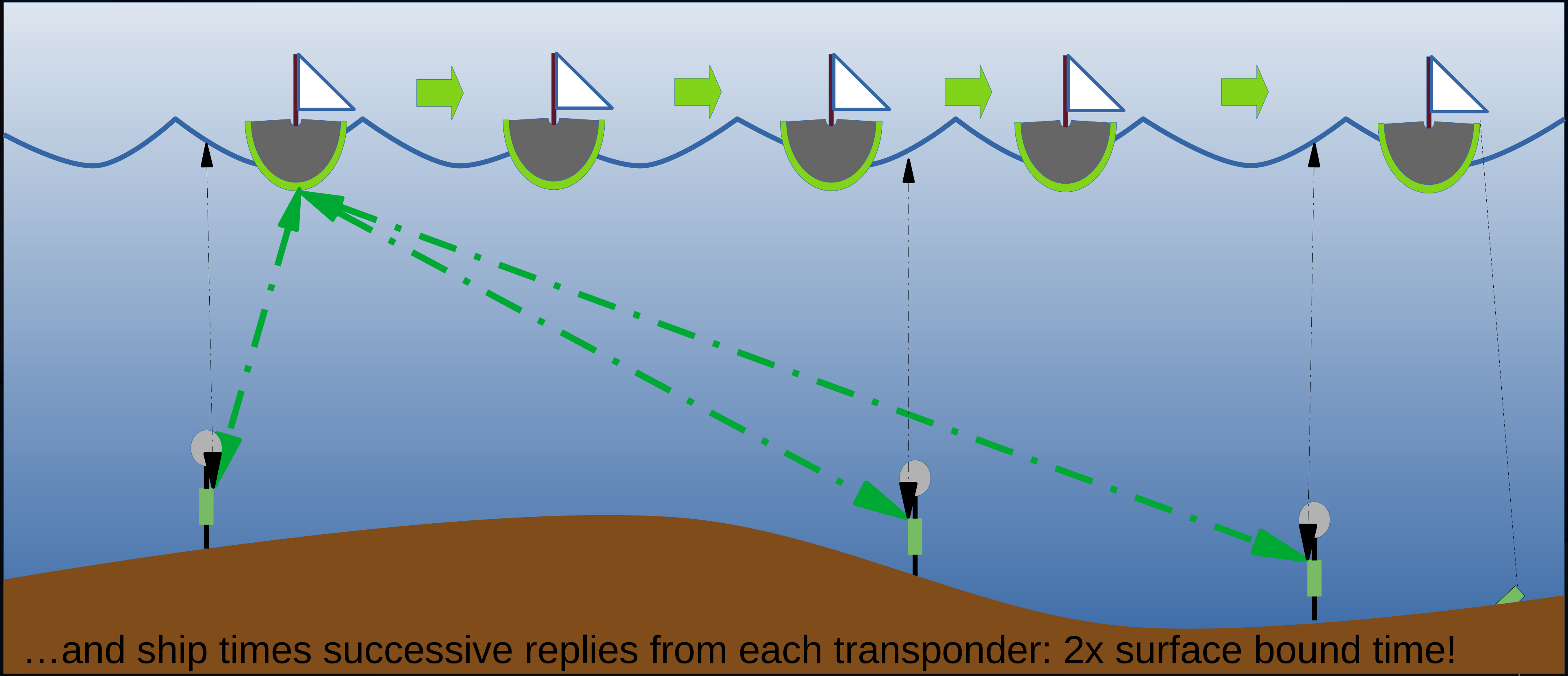

...each transponder replies on its own frequency...

Acoustic Navigation: Measure Depth



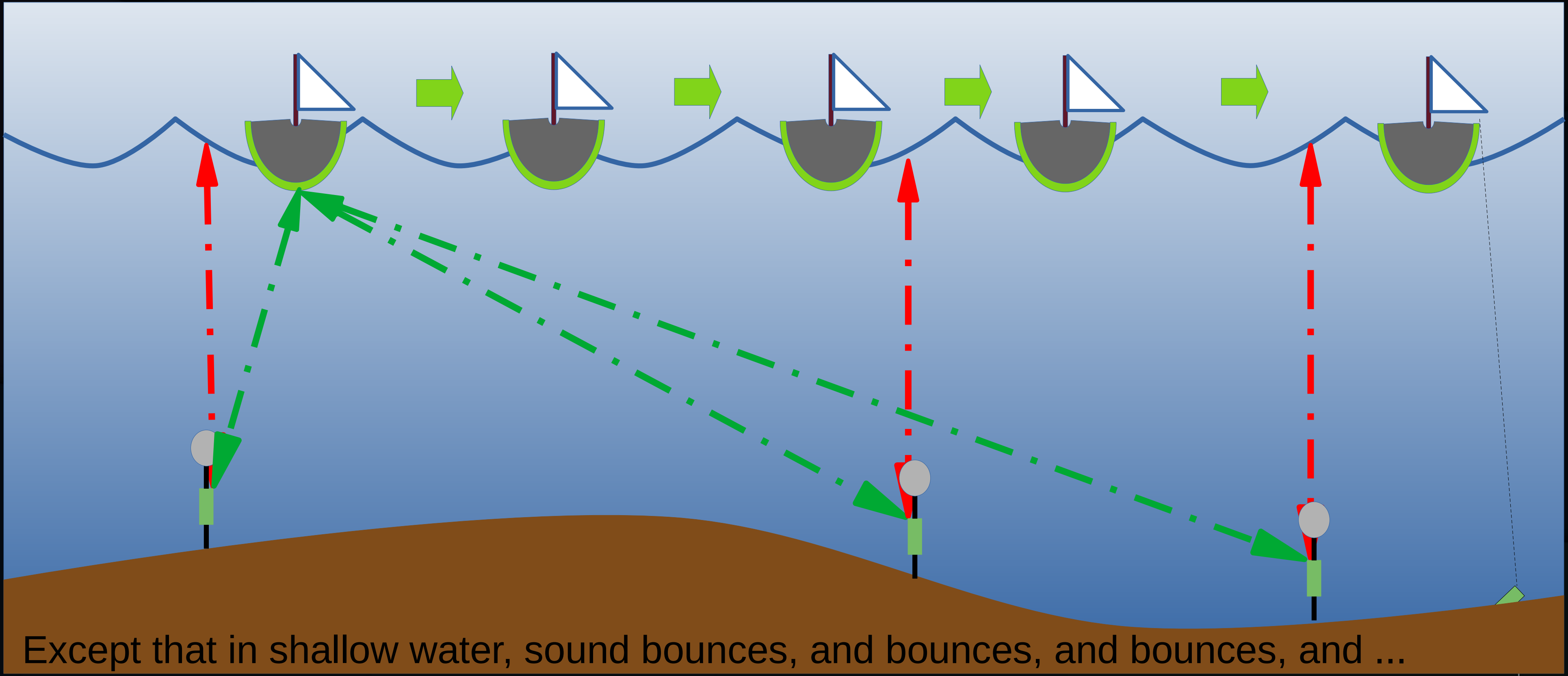
…transponders listen for bounce from surface...

Acoustic Navigation: Measure Depth



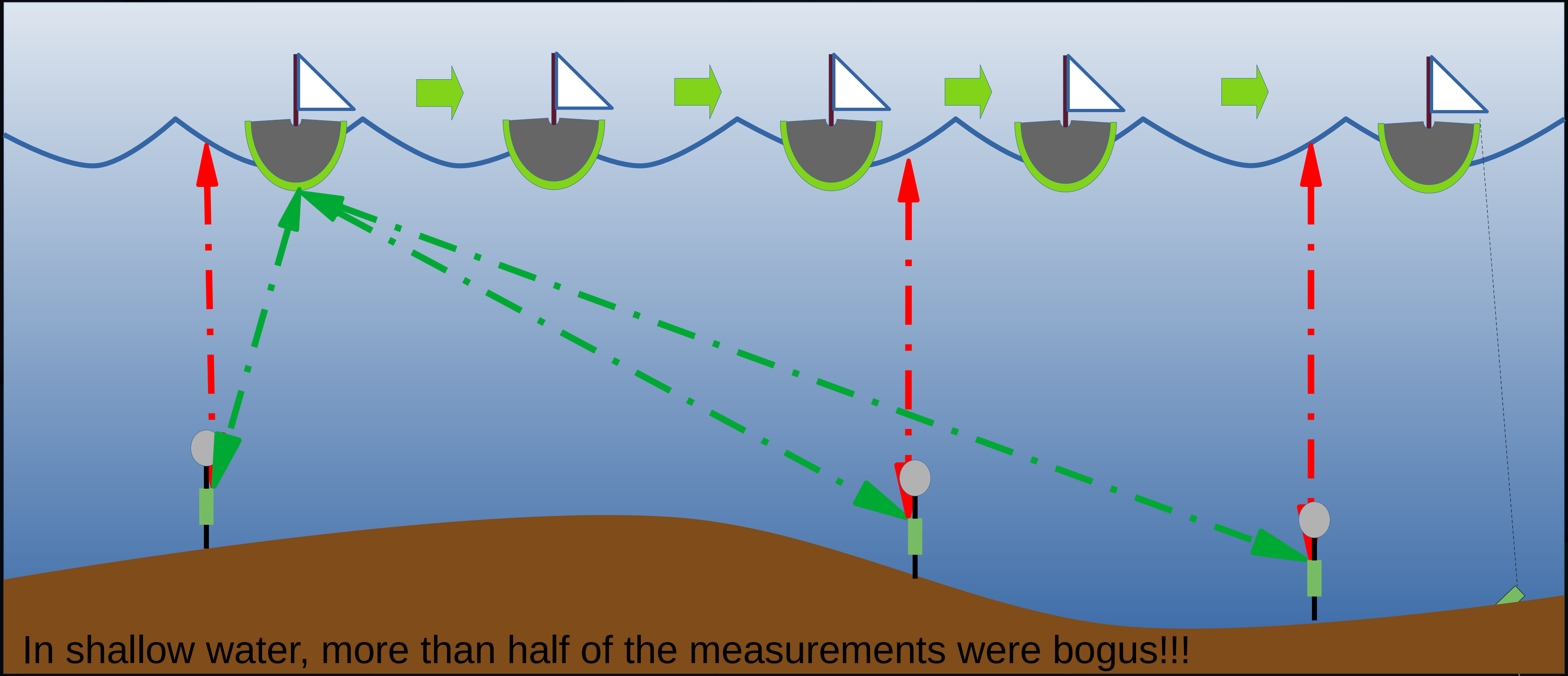
…and ship times successive replies from each transponder: 2x surface bound time!

Acoustic Navigation: Measure Depth



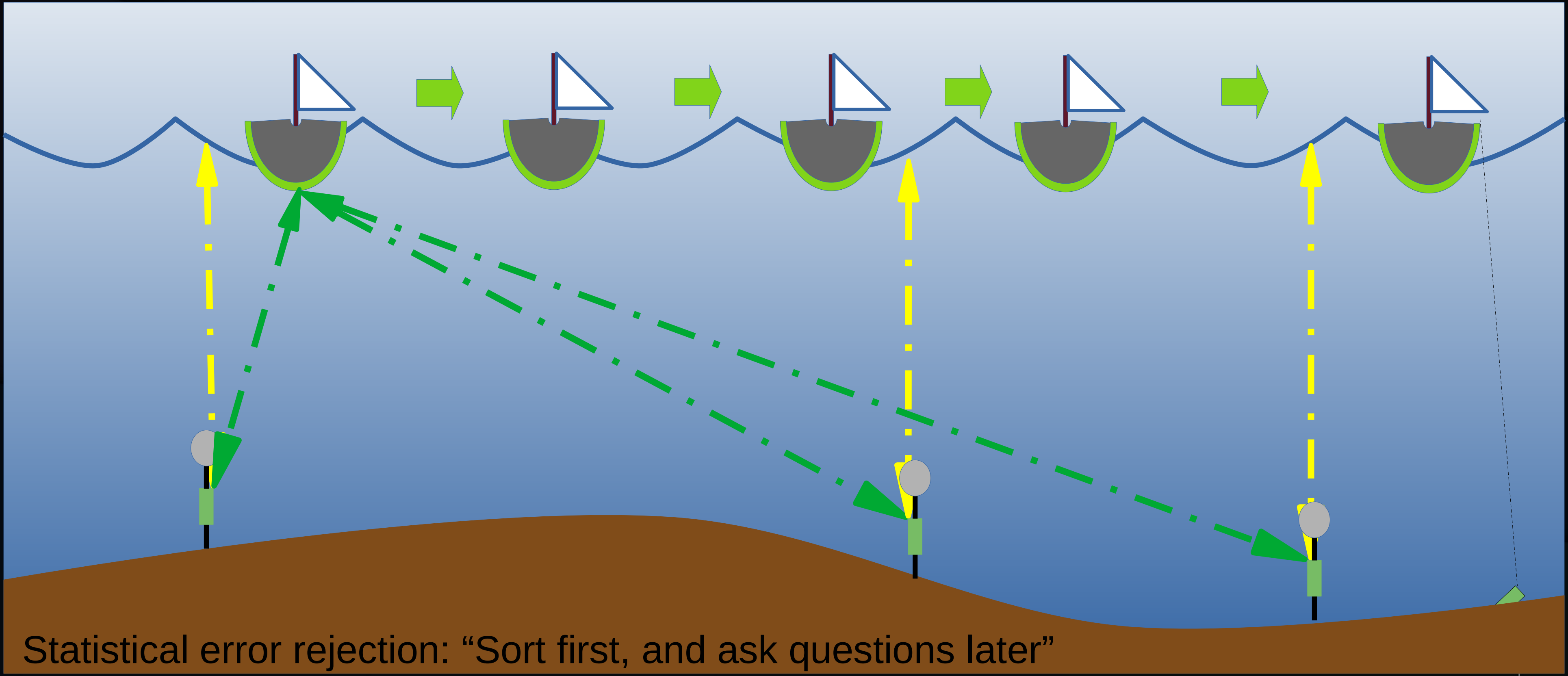
Except that in shallow water, sound bounces, and bounces, and bounces, and ...

Acoustic Navigation: Measure Depth

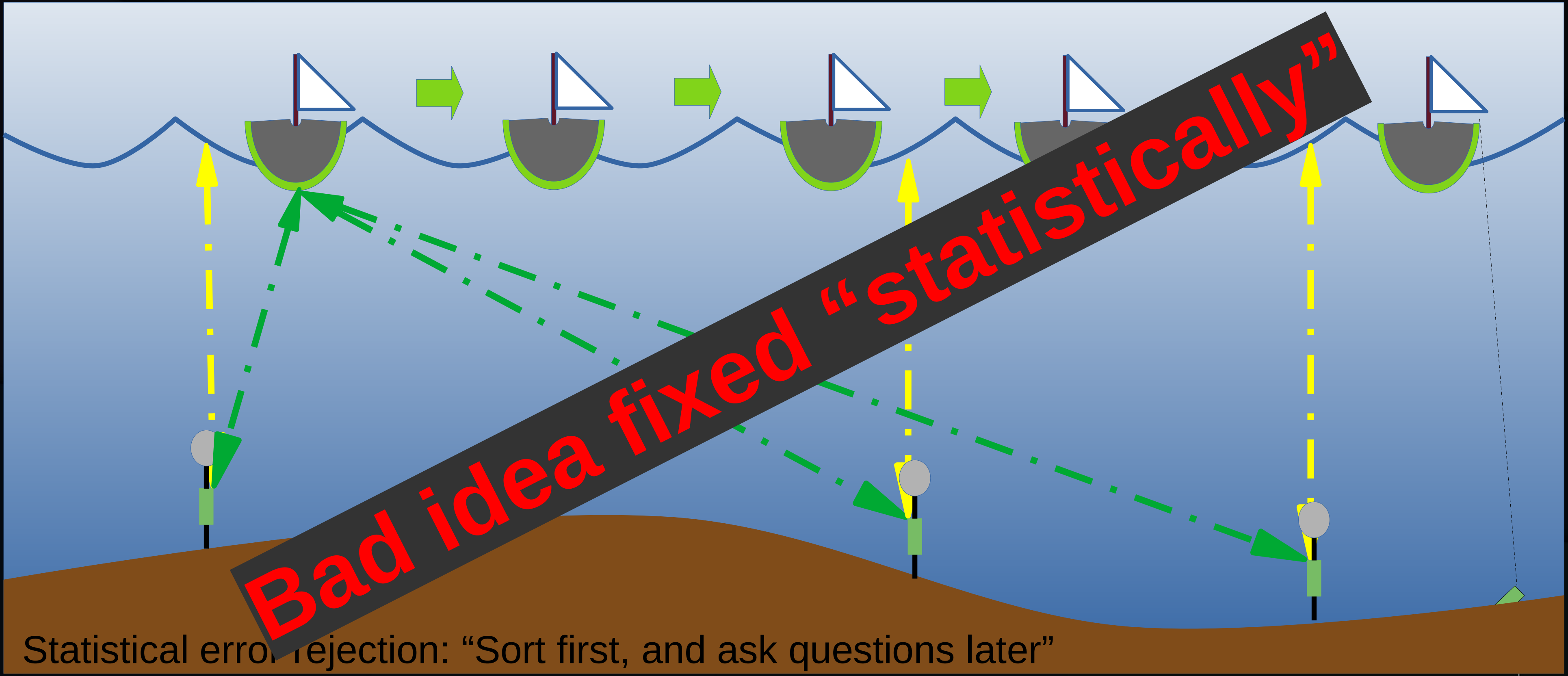


In shallow water, more than half of the measurements were bogus!!!

Acoustic Navigation: Measure Depth

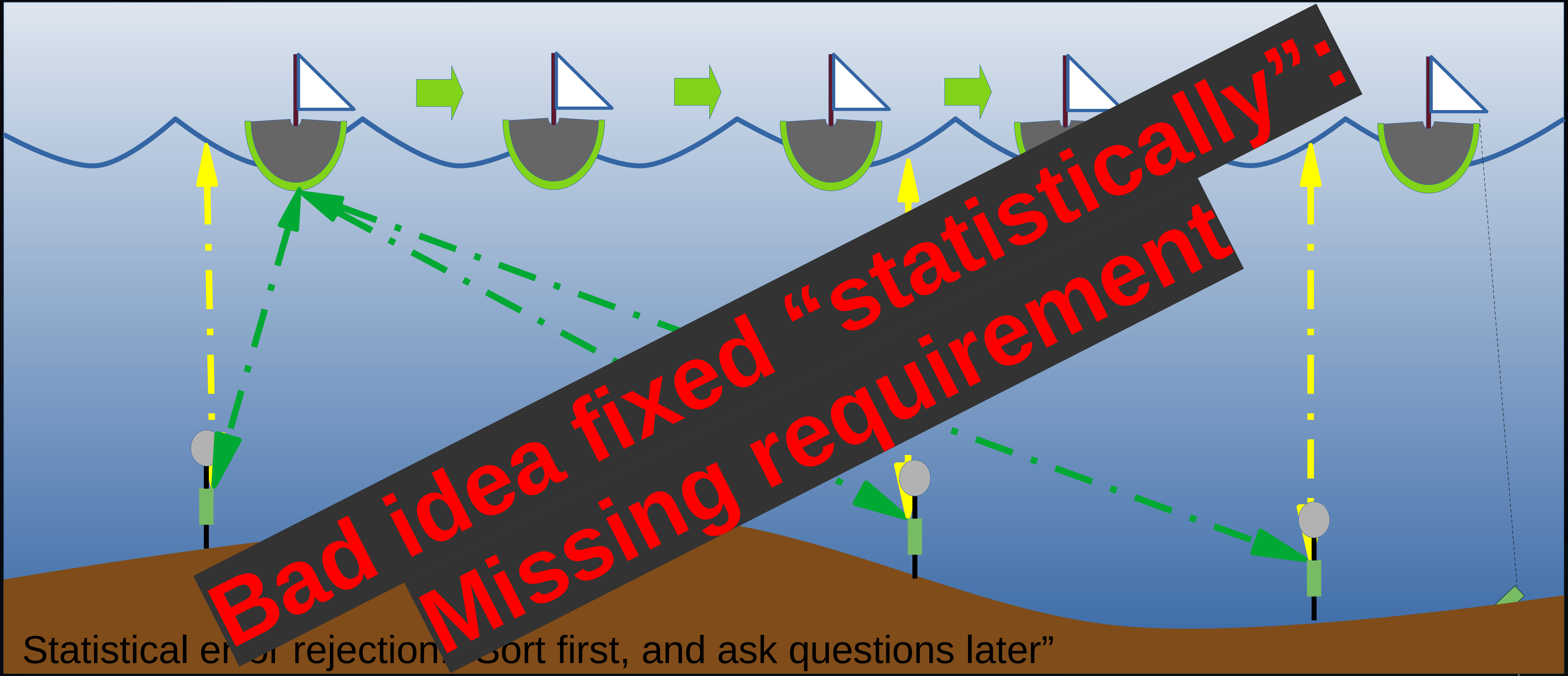

Statistical error rejection: “Sort first, and ask questions later”

Acoustic Navigation: Measure Depth



Statistical error rejection: "Sort first, and ask questions later"

Acoustic Navigation: Measure Depth


Bad idea fixed "statistically":
Missing requirement
Statistical error rejection: "Sort first, and ask questions later"

1970s: Student Housing System

Photo courtesy of Fundacio´n ICA: CDC 3300

1970s: Student Housing System

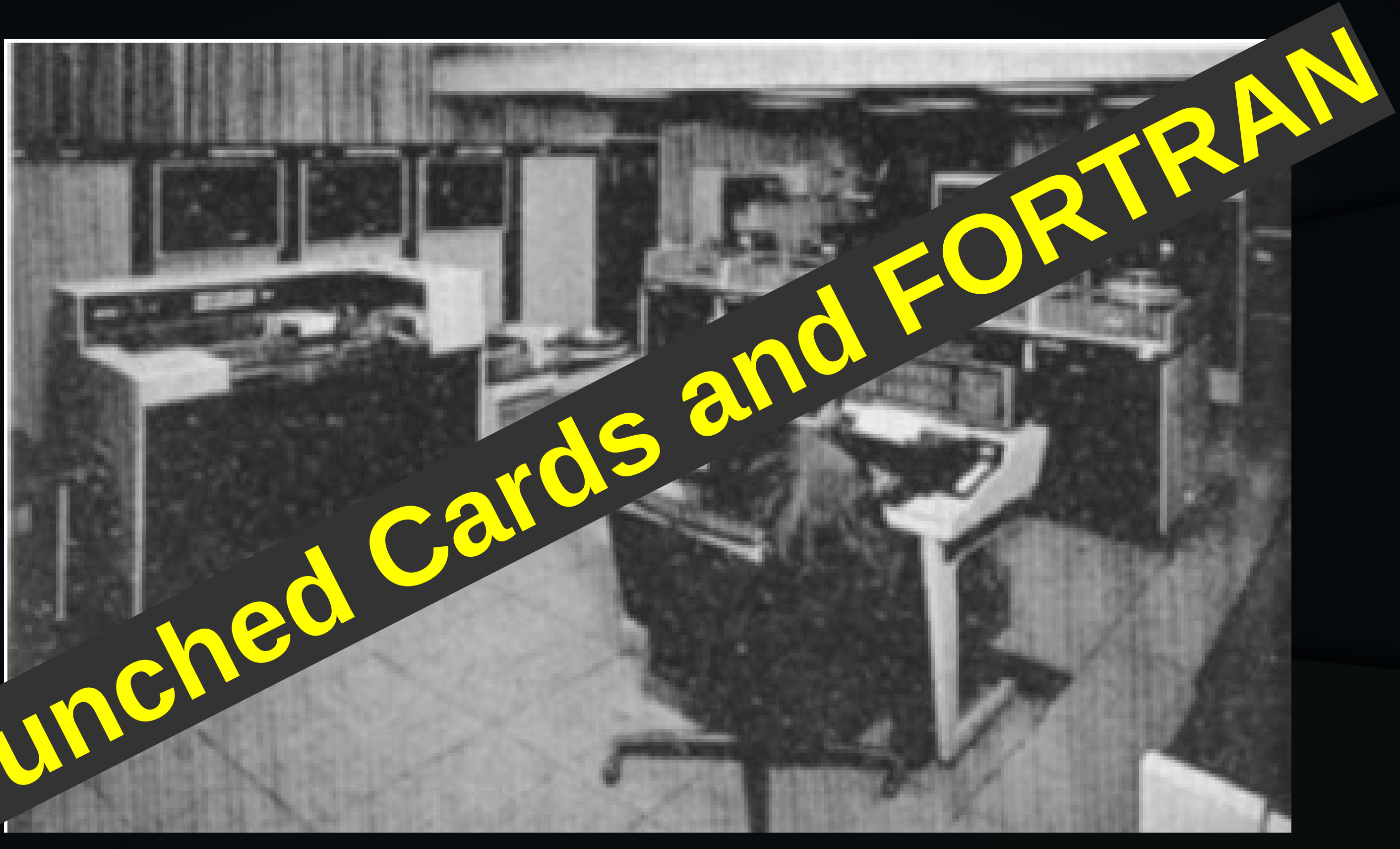


Photo courtesy of Fundacio´ n ICA: CDC 3300

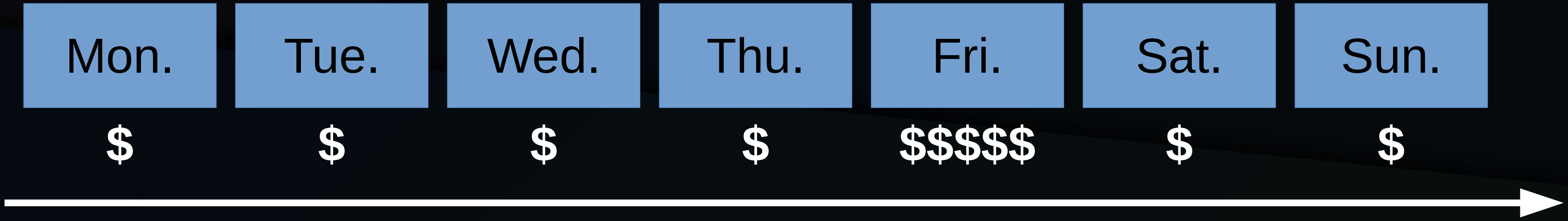
1970s: Student Housing System

Photo courtesy Wikipedia user Bubba CC BY-SA 4.0: CDC Cyber 73 console (rest of computer fills room)

1970s: Student Housing System

Photo courtesy Wikipedia user Bubba CC BY-SA 4.0: CDC Cyber 73 console (rest of computer fills room)

Student Housing Temporal Confusion



Student Housing Temporal Confusion

Student started on Friday and was not amused by the bill.

Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
$	$	$	$	$$$$$	$	$

Student Housing Temporal Confusion

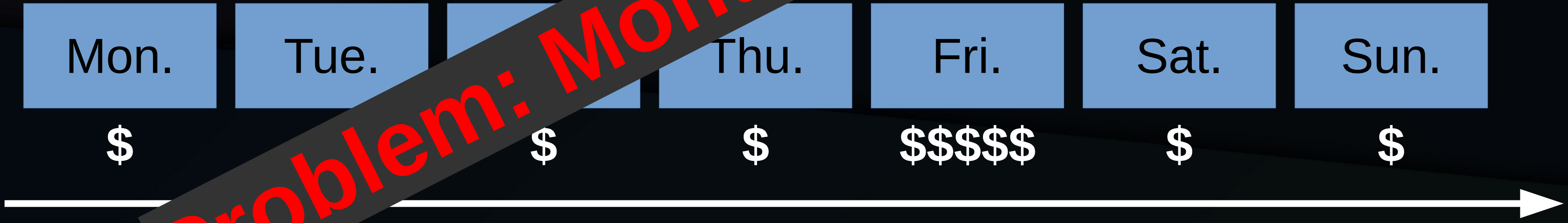
Student started on Friday and was not amused by the bill. My manager had the usual 1970s earthy suggestion for alternative uses of the money.

Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
$	**$**	**$**	**$**	**$$$$$**	**$**	**$**

Student Housing Temporal Confusion

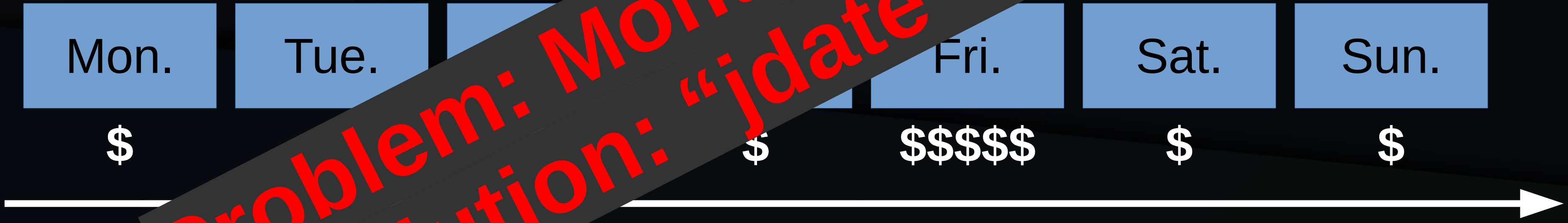
Student start day and was not by the bill. My had the usual earthy suggestion for ernative uses of the money.

Problem: Months vary in length



Student Housing Temporal Confusion

Student start day and
was not the bill.
My e usual
gestion for
es of the money.



Problem: Months vary in length

Solution: “jdate” algorithm

“jdate” algorithm: https://aa.usno.navy.mil/faq/JD_formula

Student Housing Temporal Confusion

Student started o [illegible] day and was not amu [illegible] e bill. My ma [illegible] d the usual 19 [illegible] y suggestion for [illegible] ative uses of the money.

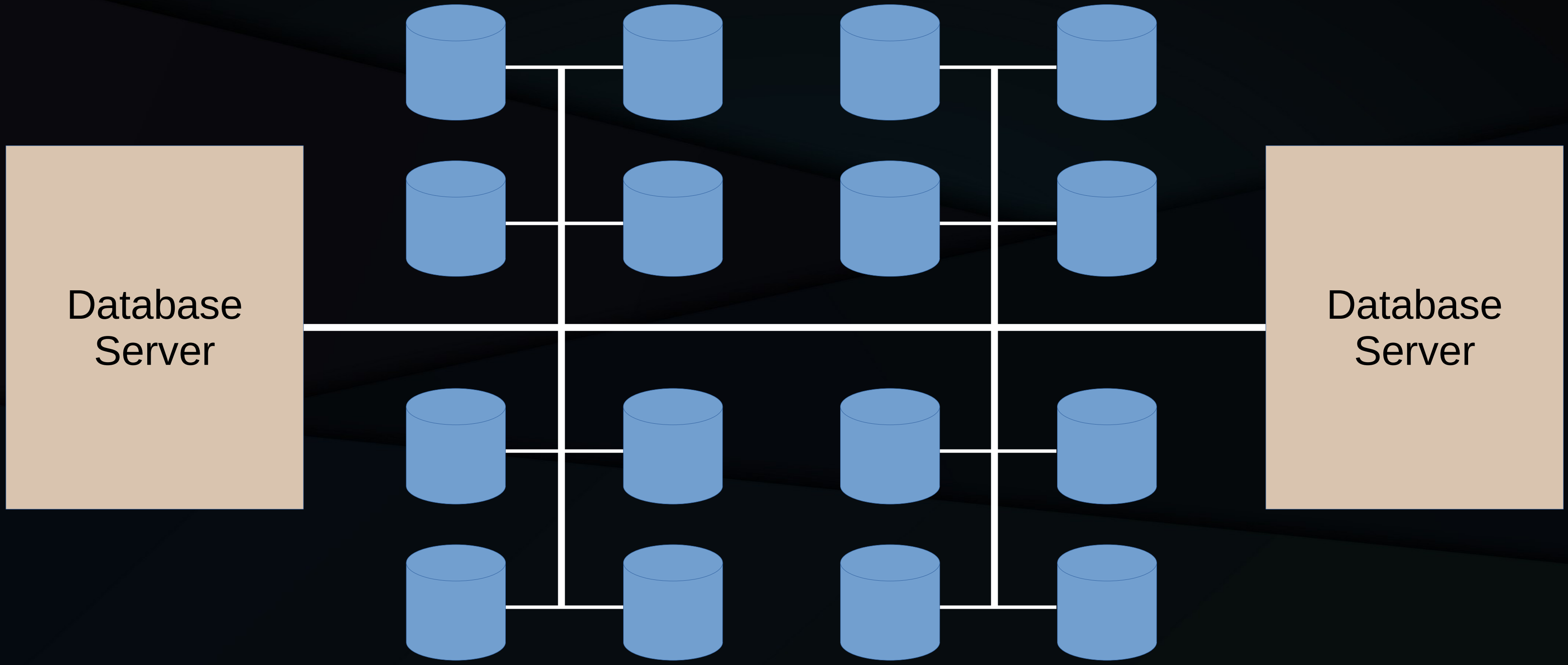
Mon.	Tue.	We [illegible]	[illegible] u.	Fri.	Sat.	Sun.
$	$	[illegible]	$	$$$$$	$	$

Good idea implemented poorly

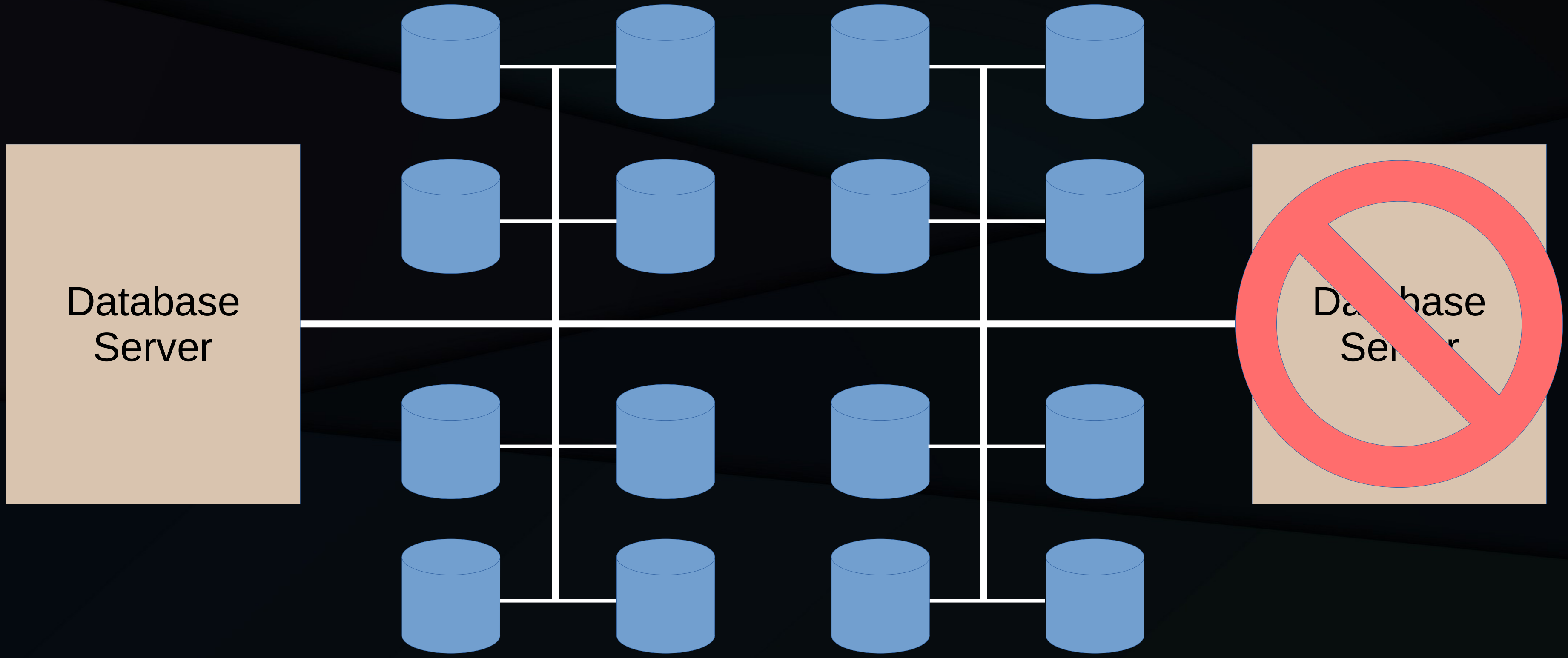
1990s: Clustered Database Servers

Photo Wikipedia user OpcomWikipedia user Opcom, CC BY-SA 3.0CC BY-SA 3.0

Shared Disks For Availability Win!!!

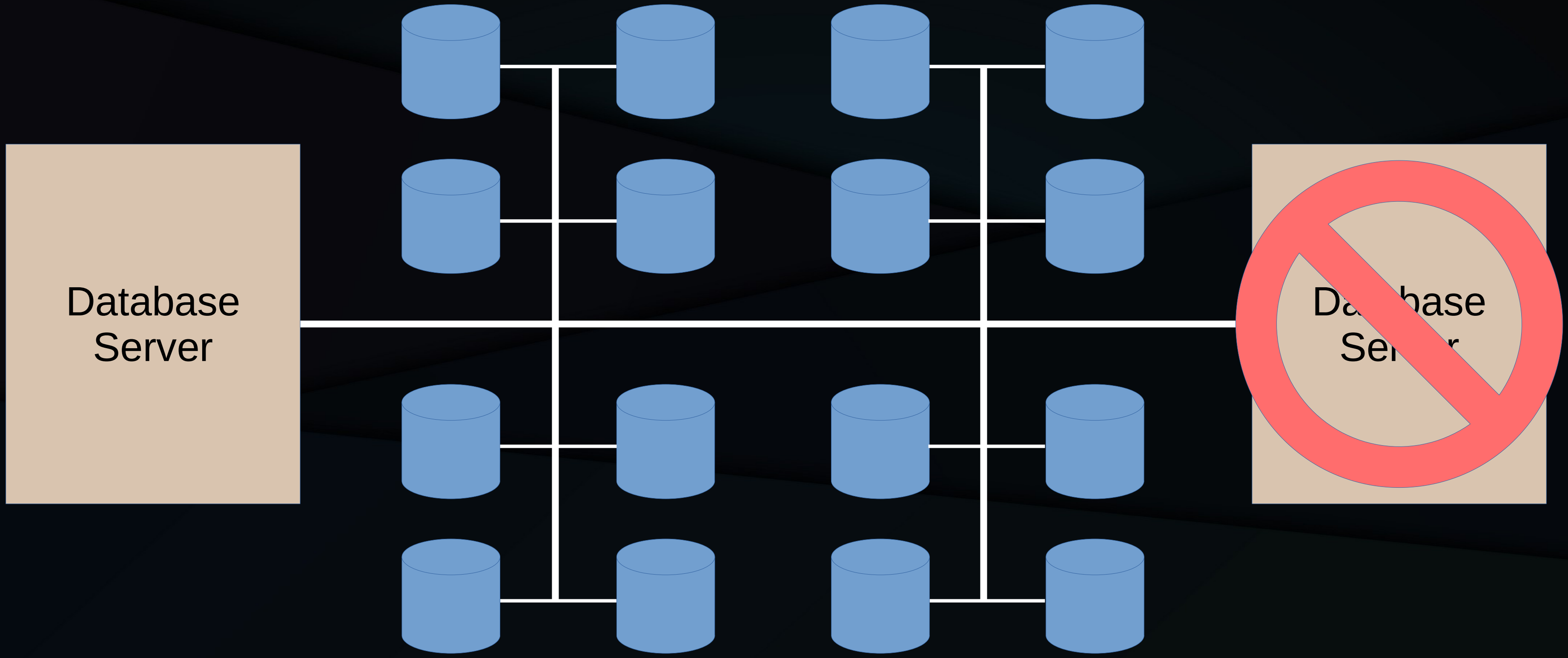


Shared Disks For Availability Win!!!



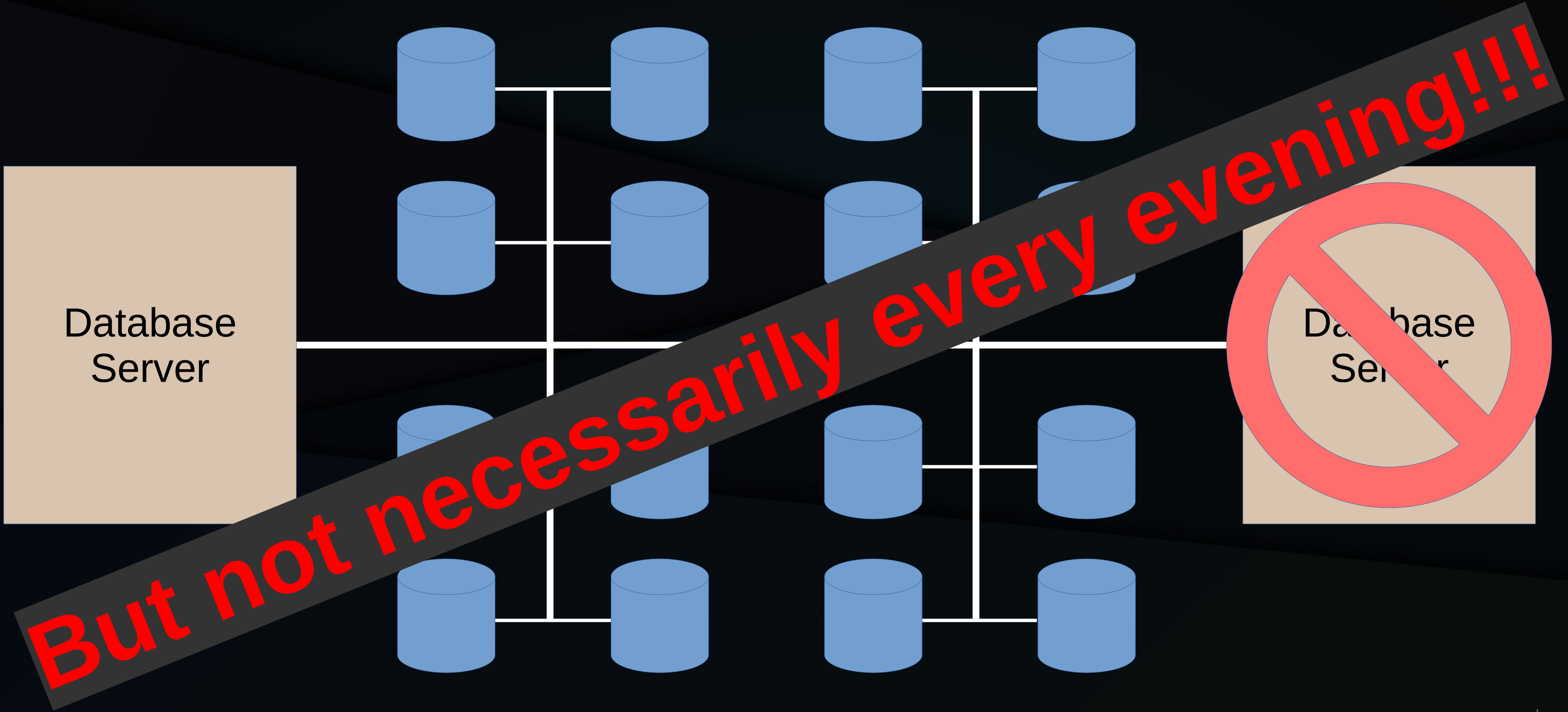
All data is still accessible!!!

Shared Disks For Availability Win!!!



All data is still accessible!!! Of course, sites should test this frequently...

Shared Disks For Availability Win!!!



All data is still accessible!!! Of course, sites should test this frequently...

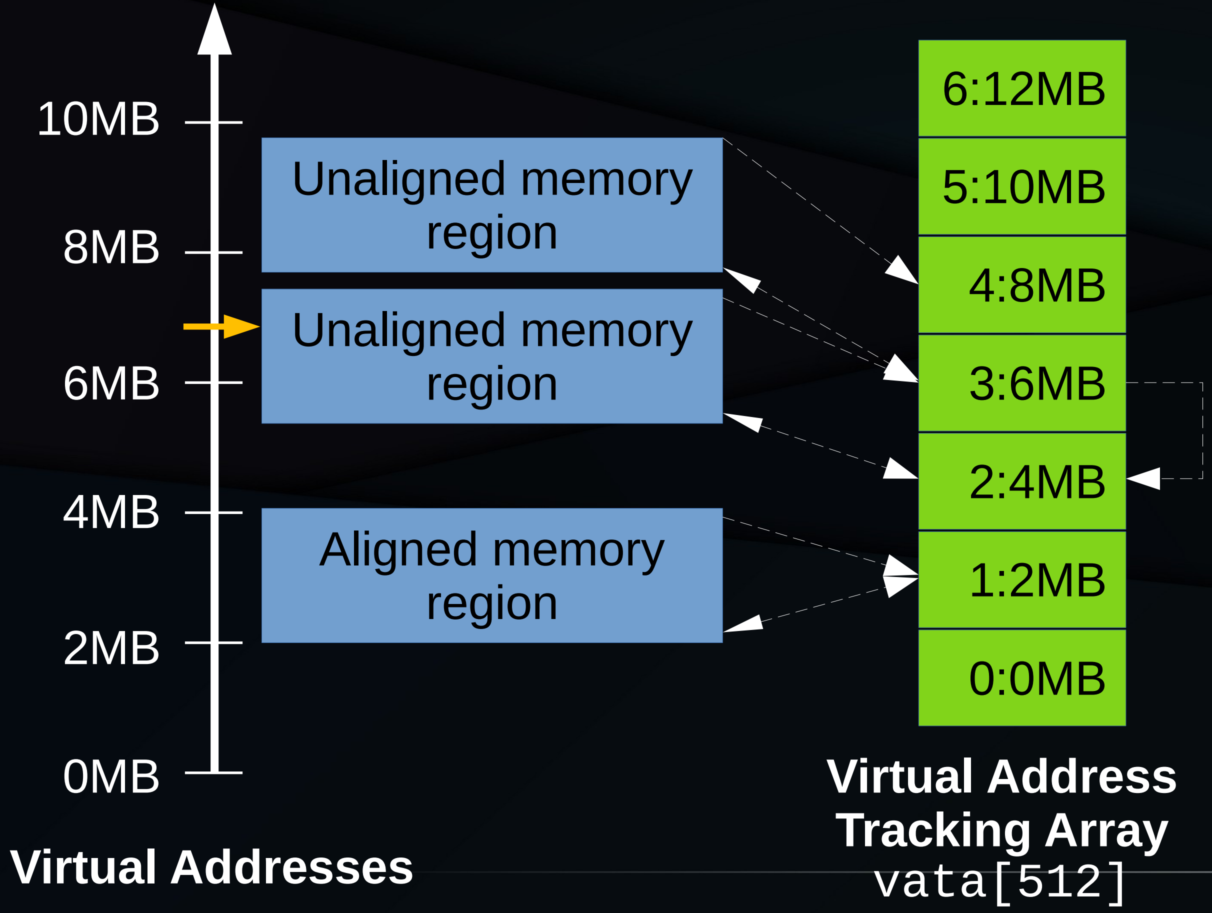
Chaos-Monkey Challenges

- Crash dump was a complete disaster area
  - No hints for on-site debugging instrumentation
- Unable to reproduce in the lab

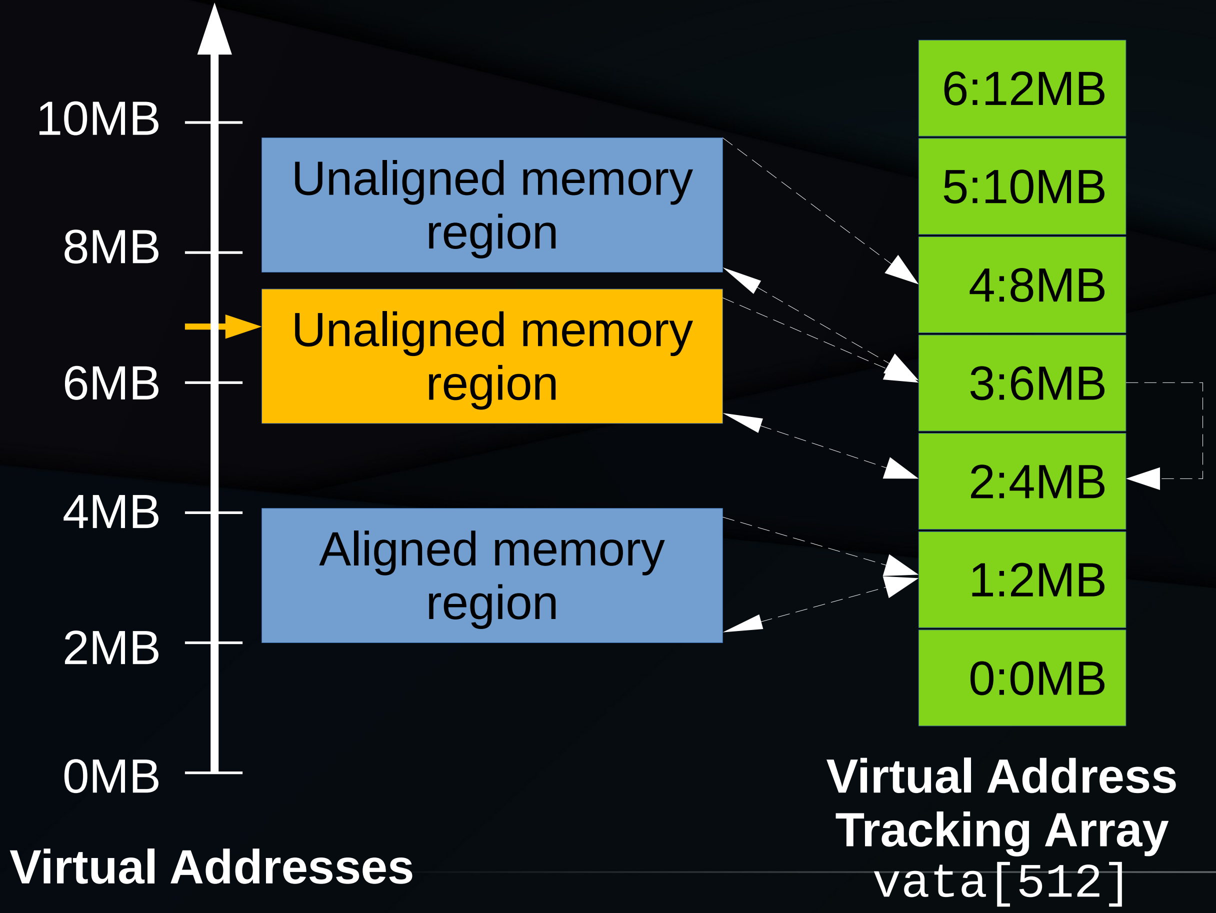
Chaos-Monkey Challenges

- Crash dump was a complete disaster area
  - No hints for on-site debugging instrumentation
- Unable to reproduce in the lab
- Eventually, found test case: 5-27-hour MTBF
  - But need week-long test for any alleged fix!!!
  - And it was now Memorial Day weekend...

Hint From Stack Trace



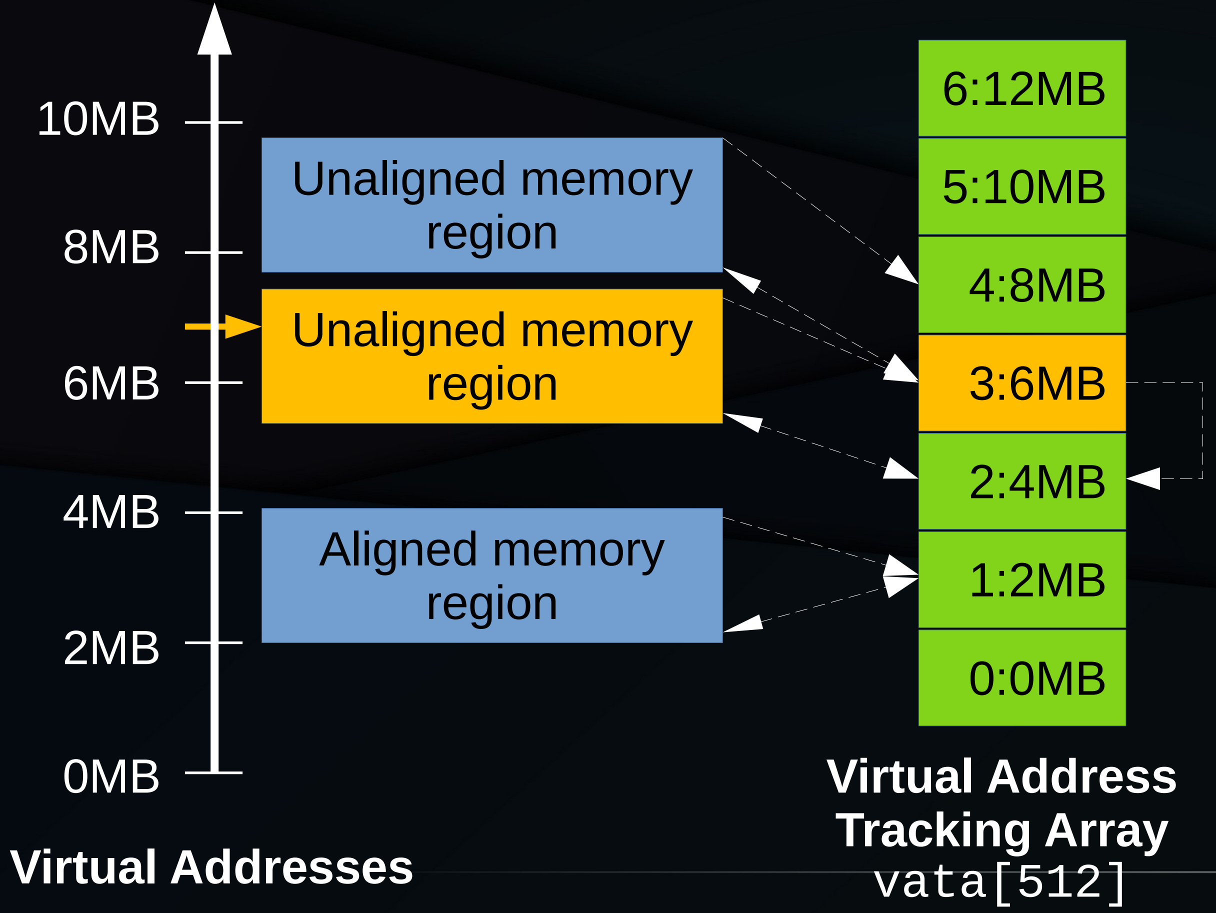
Hint From Stack Trace



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

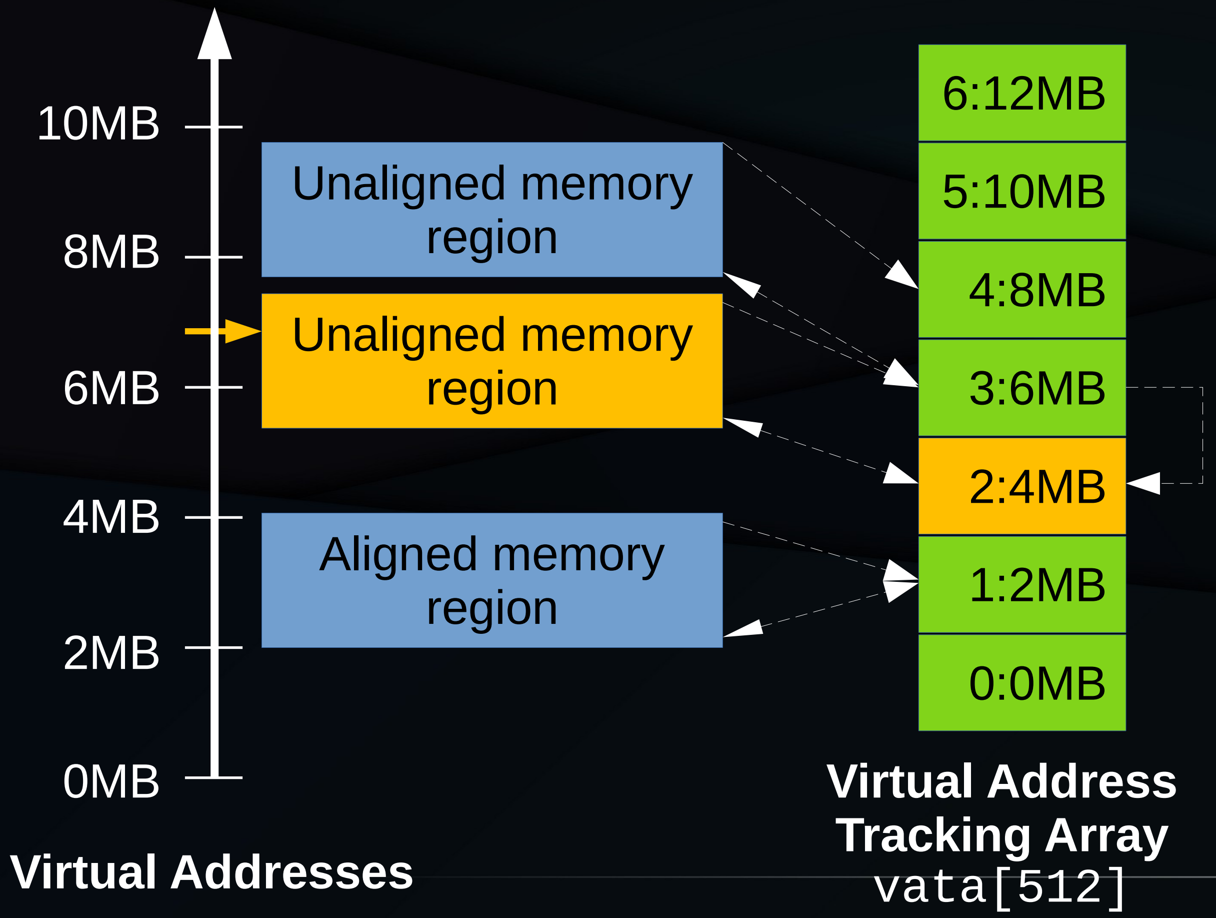
Hint From Stack Trace



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vta || vadr < vta)
    vta = vta[idx - 1];
```

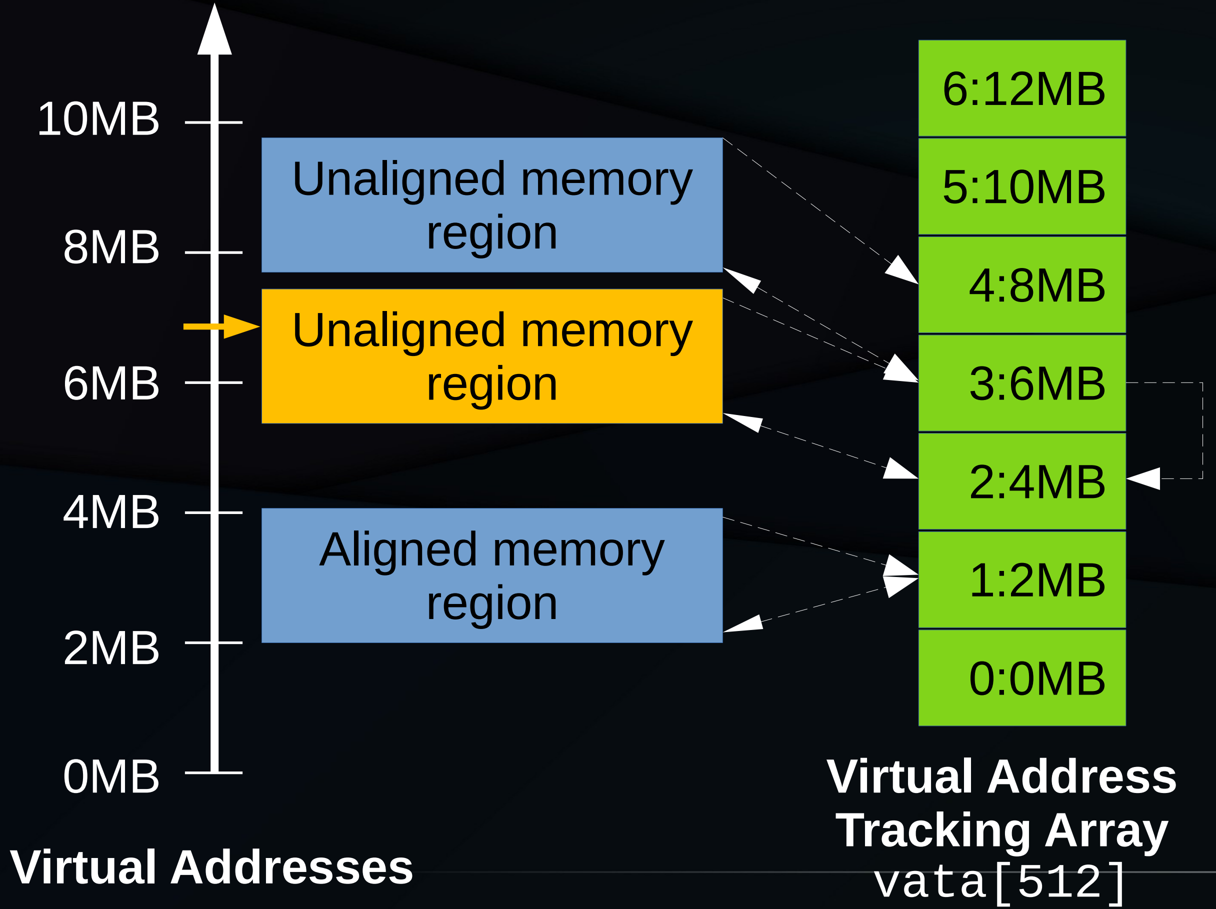
Hint From Stack Trace



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

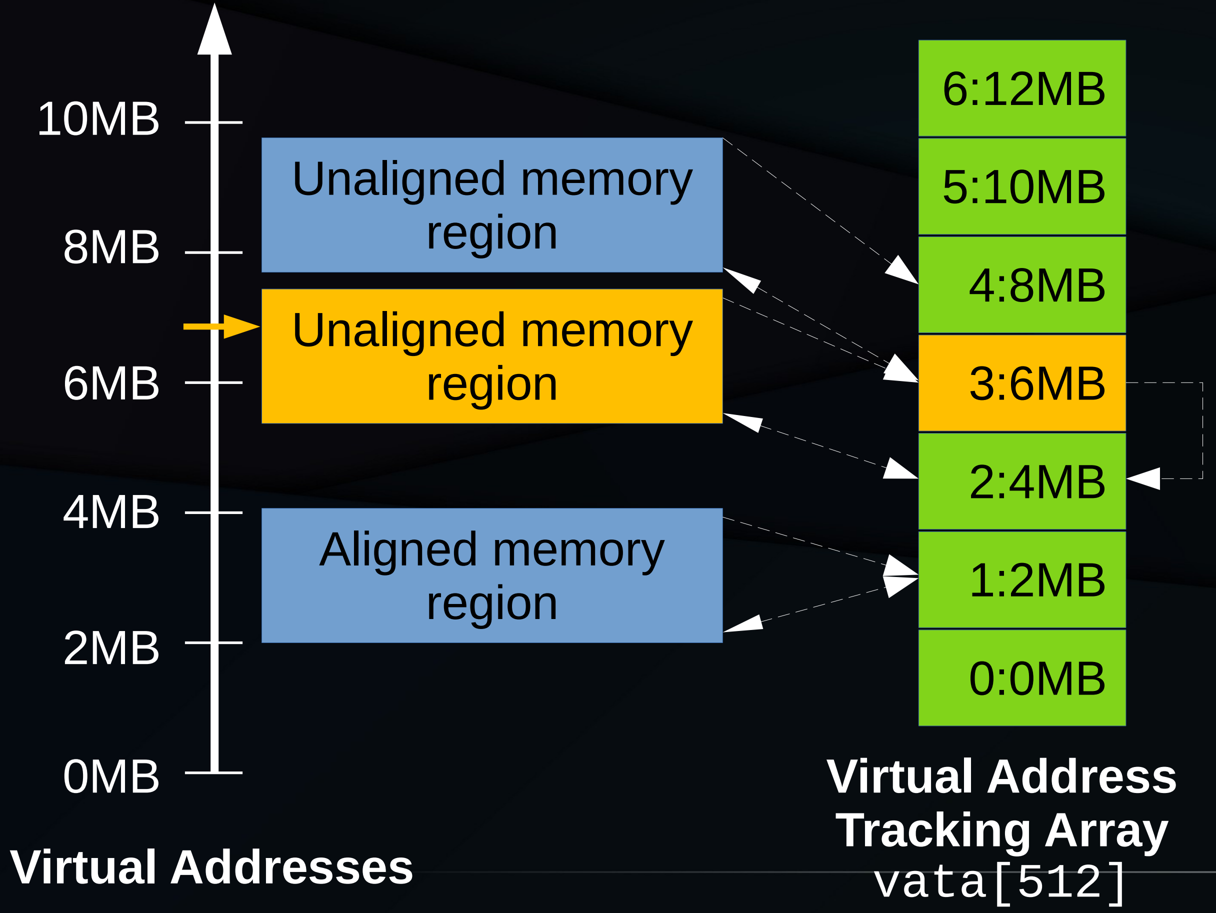
Hint From Stack Trace: Compiler Fun



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vata[idx] ||
     vadr < vata[idx])
    vta = vta[idx – 1];
```

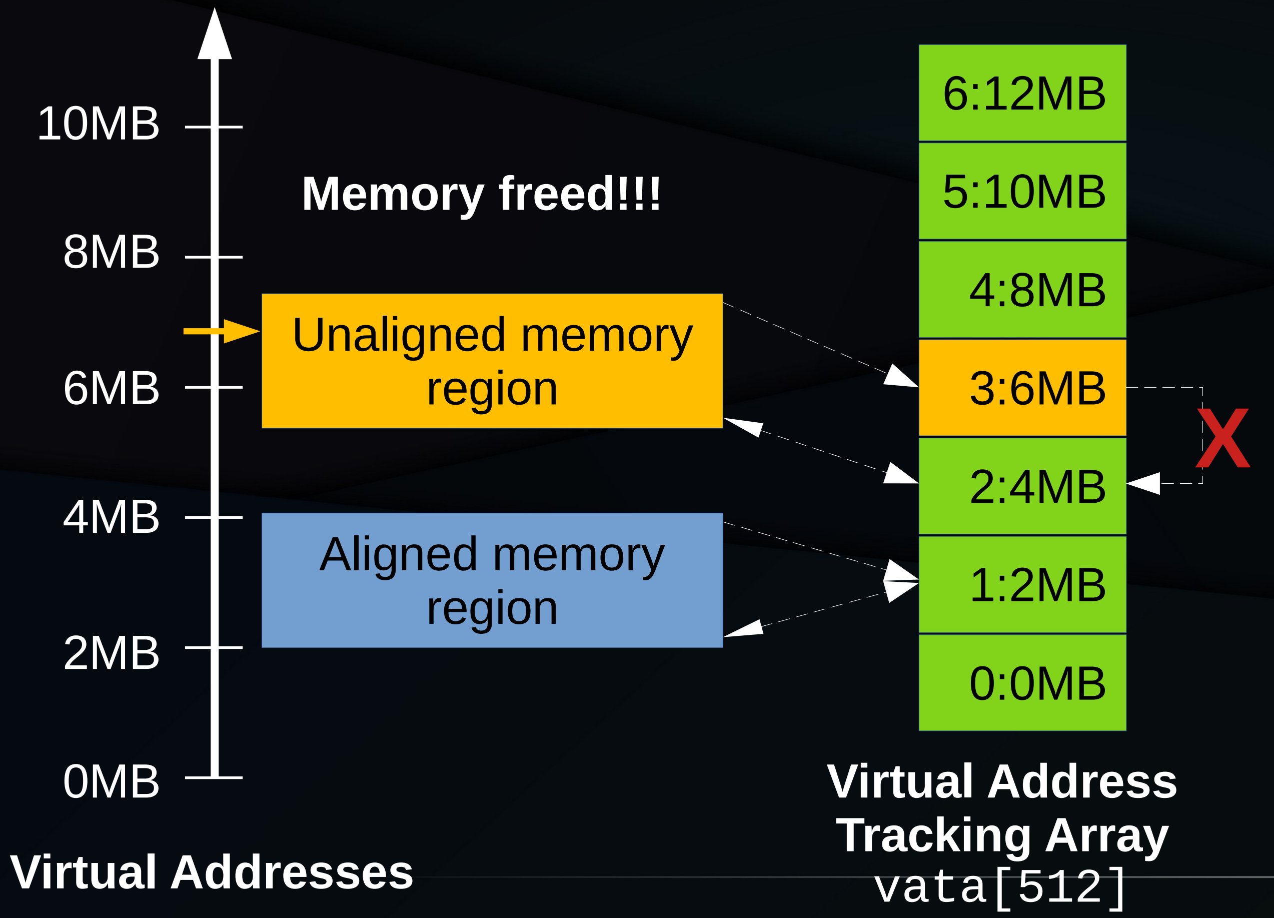
Compiler Fun In Failure Case



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vata[idx] ||
     vadr < vata[idx])
    vta = vta[idx – 1];
```

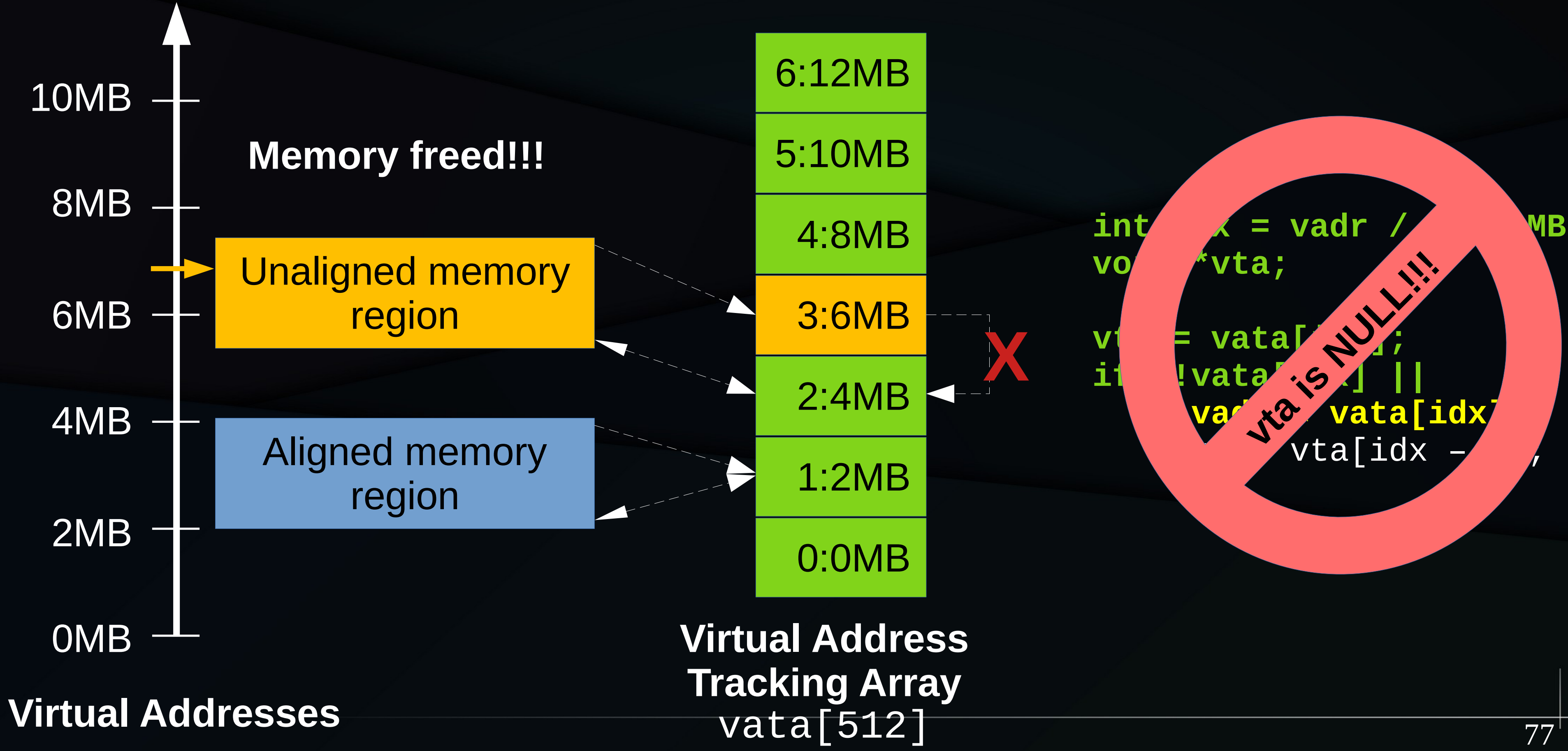
Compiler Fun In Failure Case: Update



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vata[idx] ||
      vadr < vata[idx])
    vta = vta[idx - 1];
```

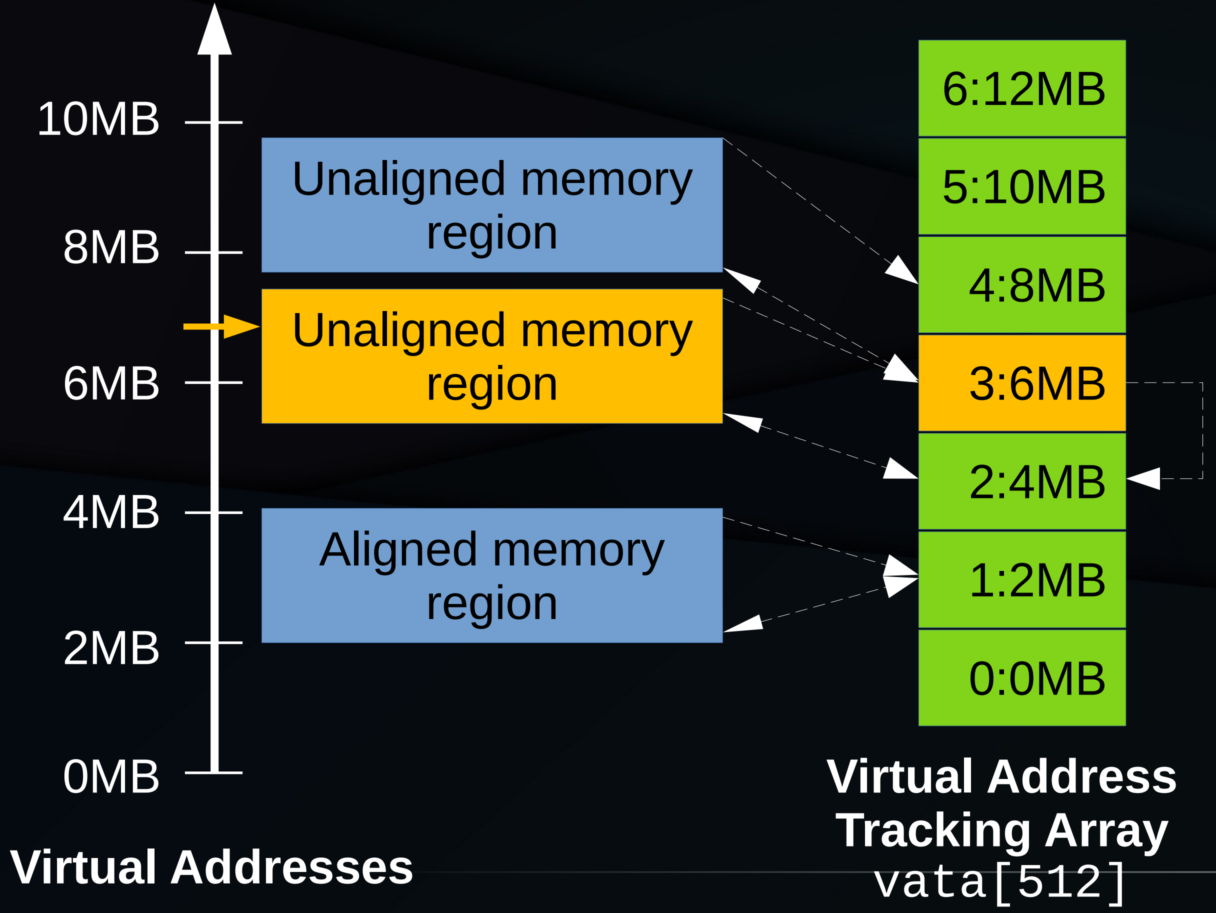
Compiler Fun In Failure Case: Update



```
int   x = vadr /      MB);
vo    *vta;

vt    = vata[   ];
if   !vata[   x] ||
      va      vata[idx]
          vta[idx -    ,
```

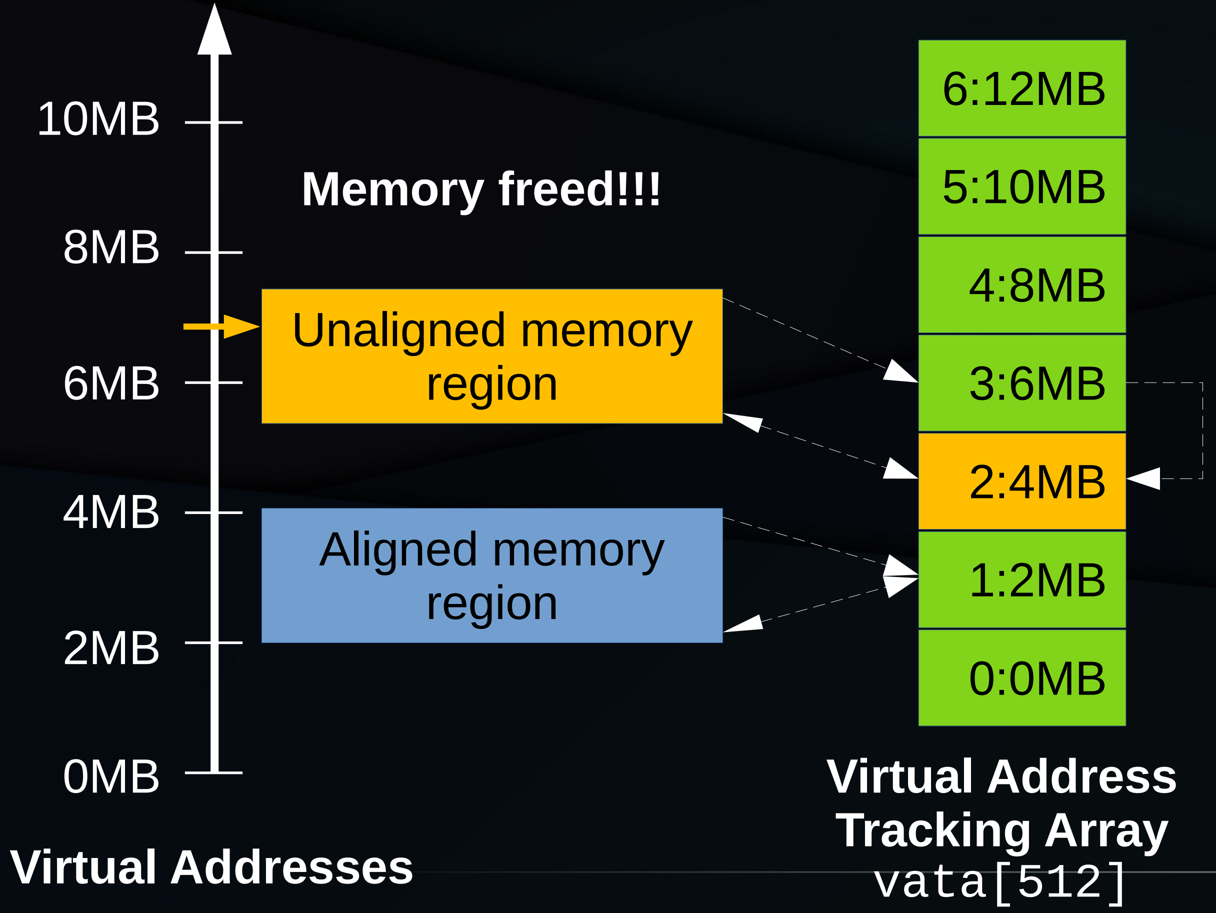
Thwarting Compiler Fun



```
int idx = vadr / (2 * MB);
void *vta;

vta = READ_ONCE(vata[idx]);
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

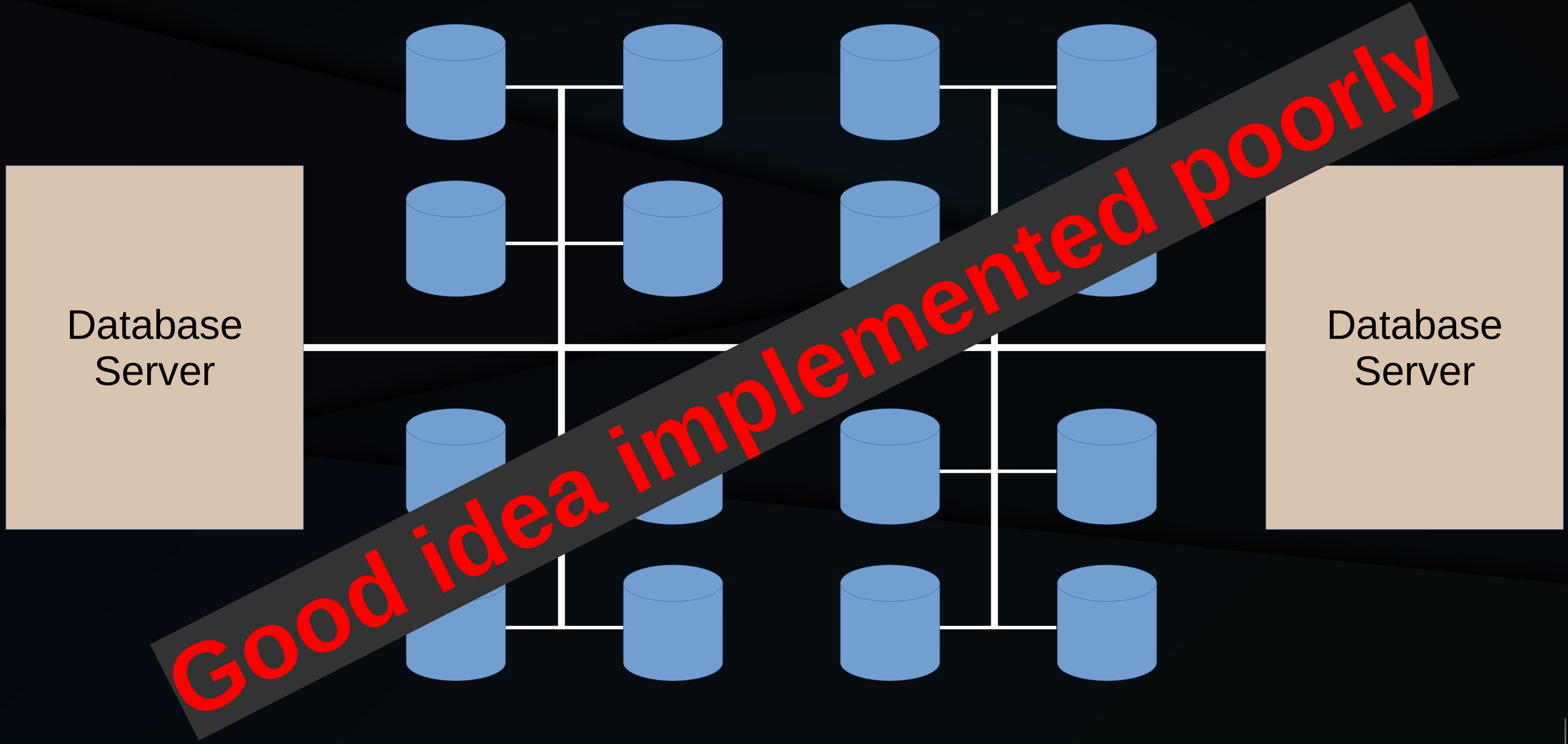
Thwarting Compiler Fun: Update OK



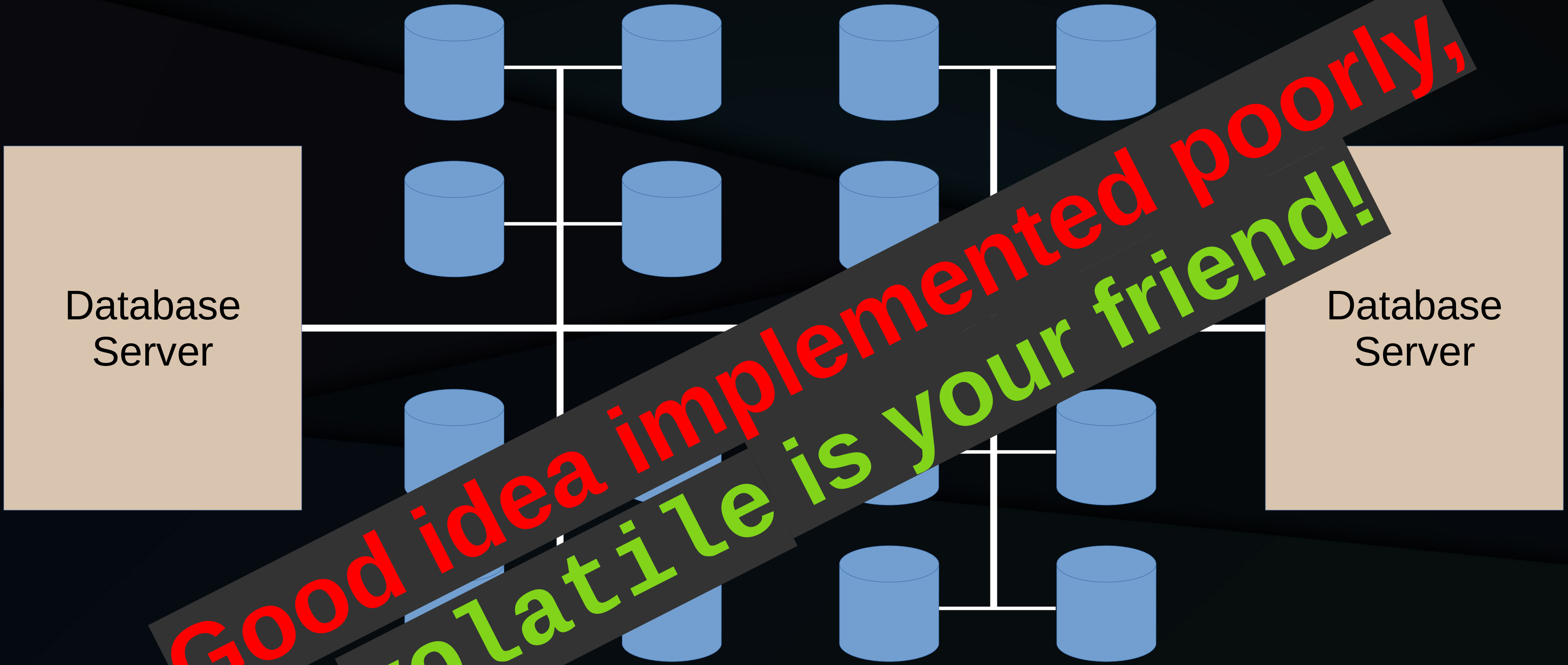
```
int idx = vadr / (2 * MB);
void *vta;

vta = READ_ONCE(vata[idx]);
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

Shared Disks For Availability Win!!!



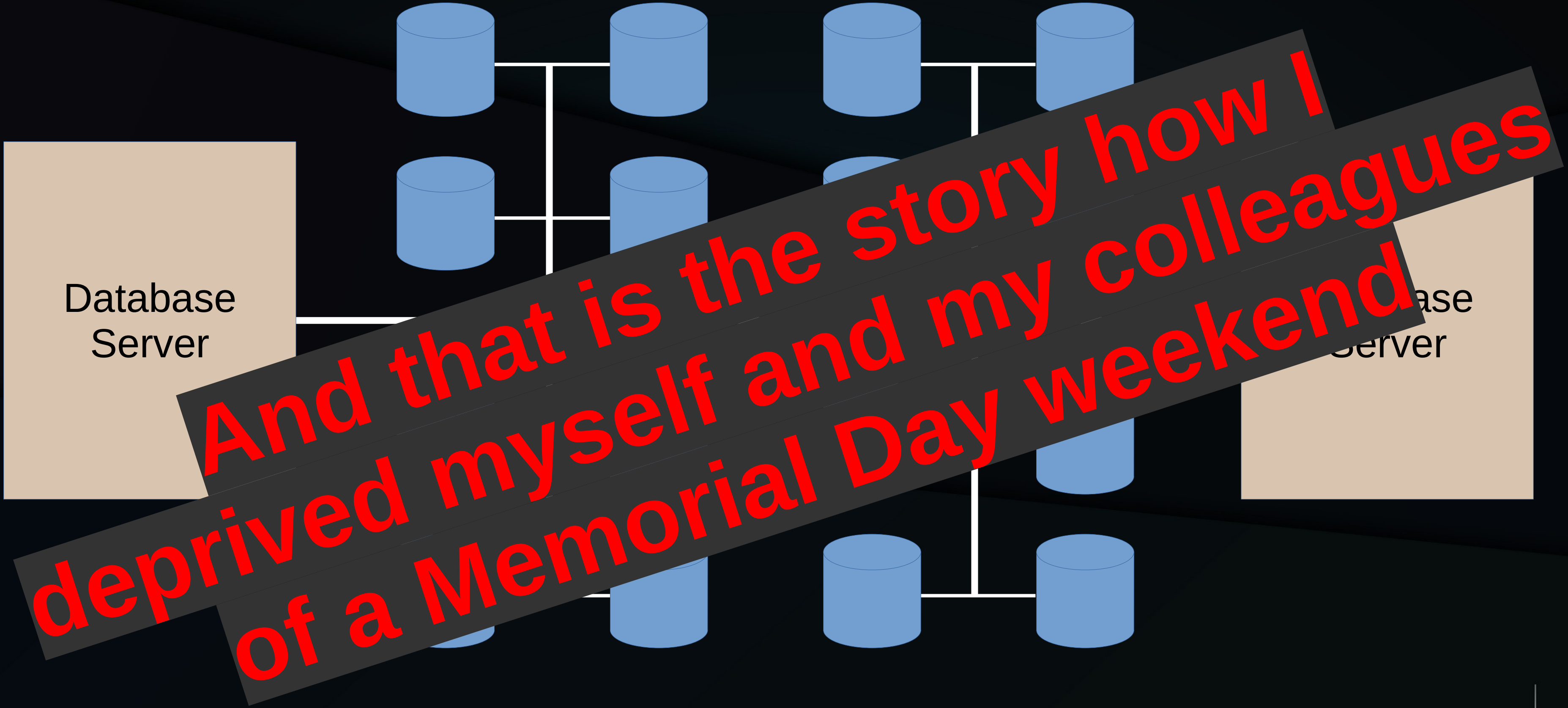
Shared Disks For Availability Win!!!



Good idea implemented poorly,

`volatile` is your friend!

Shared Disks For Availability Win!!!



And that is the story how I deprived myself and my colleagues of a Memorial Day weekend

1970s: My First Professional Project

Photo courtesy of Rama & Musée Bolo, CC BY-SA 2.0 fr ASR-33 Teletype

1970s: My First Professional Project

- Pro-bono computer dating program for National Honor Society fundraiser during my senior year in high school
- Questions from Home Economics teacher
- Simple Hamming-distance matching with expected 1970s constraints on matches
- Students' paper questionnaires transcribed to paper tape, then read into program
- Simple, effective, worked great!!!

One Dissatisfied Customer

- Senior girl matched only with freshmen boys
  - And she really did check the seniors-only box
- Program looked to be correct
- Turned out to be data-entry error
- Correct program is not enough
  - Environment and processes matter!!!

One Dissatisfied Customer

- Senior girl matched only with fres[illegible]
  - And she really did check the [illegible]-only box
- Program looked to be [illegible]ect
- Turned out to [illegible]a-entry error
- Correct [illegible]am is not enough
  - [illegible]onment and processes matter!!!

Good idea implemented properly,

One Dissatisfied Customer

- Senior girl matched only with fresh
  - And she really did check the
- Program looked to be
- Turned out to b   error
- Correct   ot enough
  - nd processes matter!!!

Good idea implemented properly,
but I was also overall project lead!

Cautionary Quote

- A lot of success in life and business comes from knowing what you want to avoid. - *Charlie Munger*

2004: Real-Time Linux

2004: Real-Time Linux

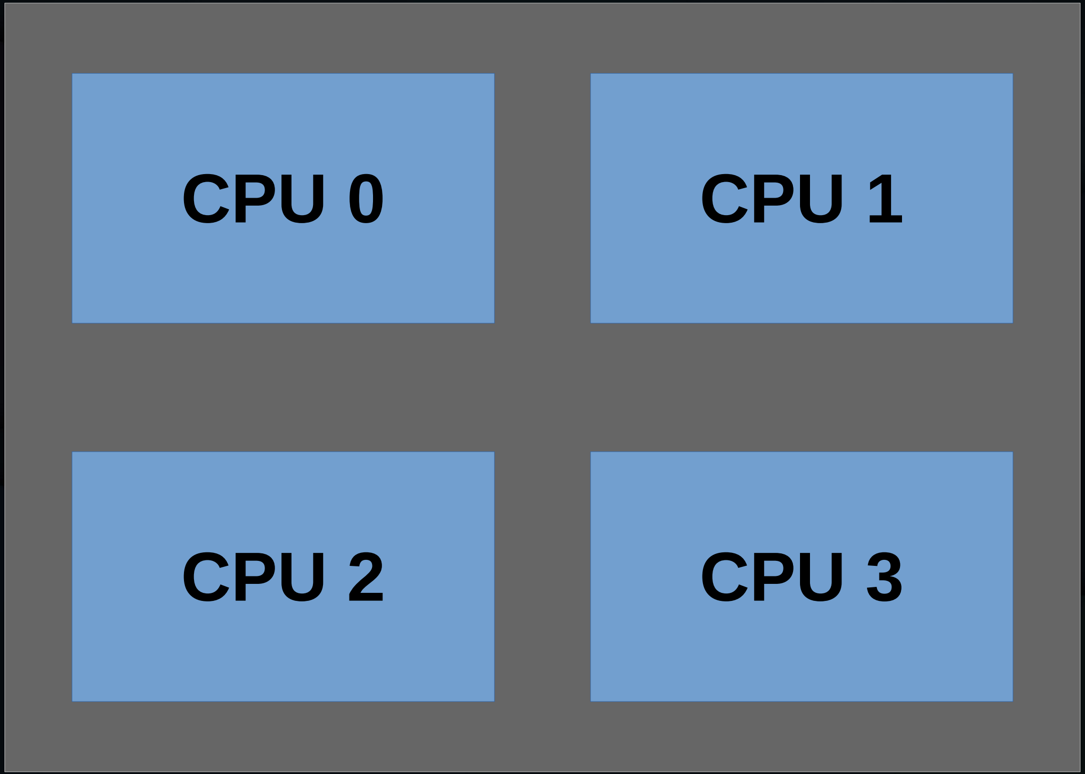
- Early 2000s: Many requests for real-time Linux
  - But “enterprise-grade real-time Linux”
- Except that no such thing existed
- And my employer had strict rules for contracts calling for mythical creatures

2004: Real-Time Linux

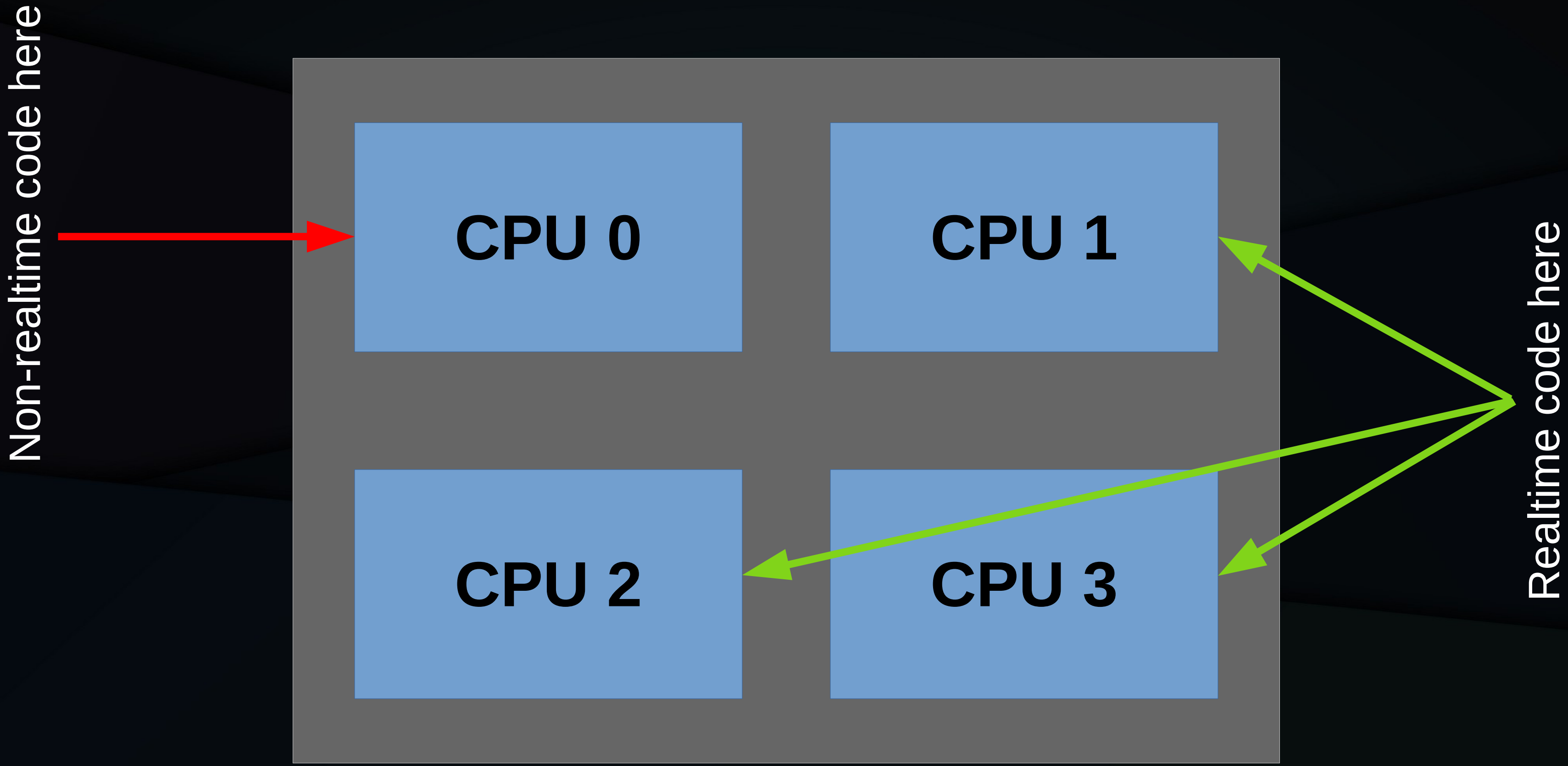
- Early 2000s: Many requests real-time Linux
  - But “enterprise-grade r ux”
- Except that no s ted
- And my emp trict rules for contracts calling for myt creatures

No Bid

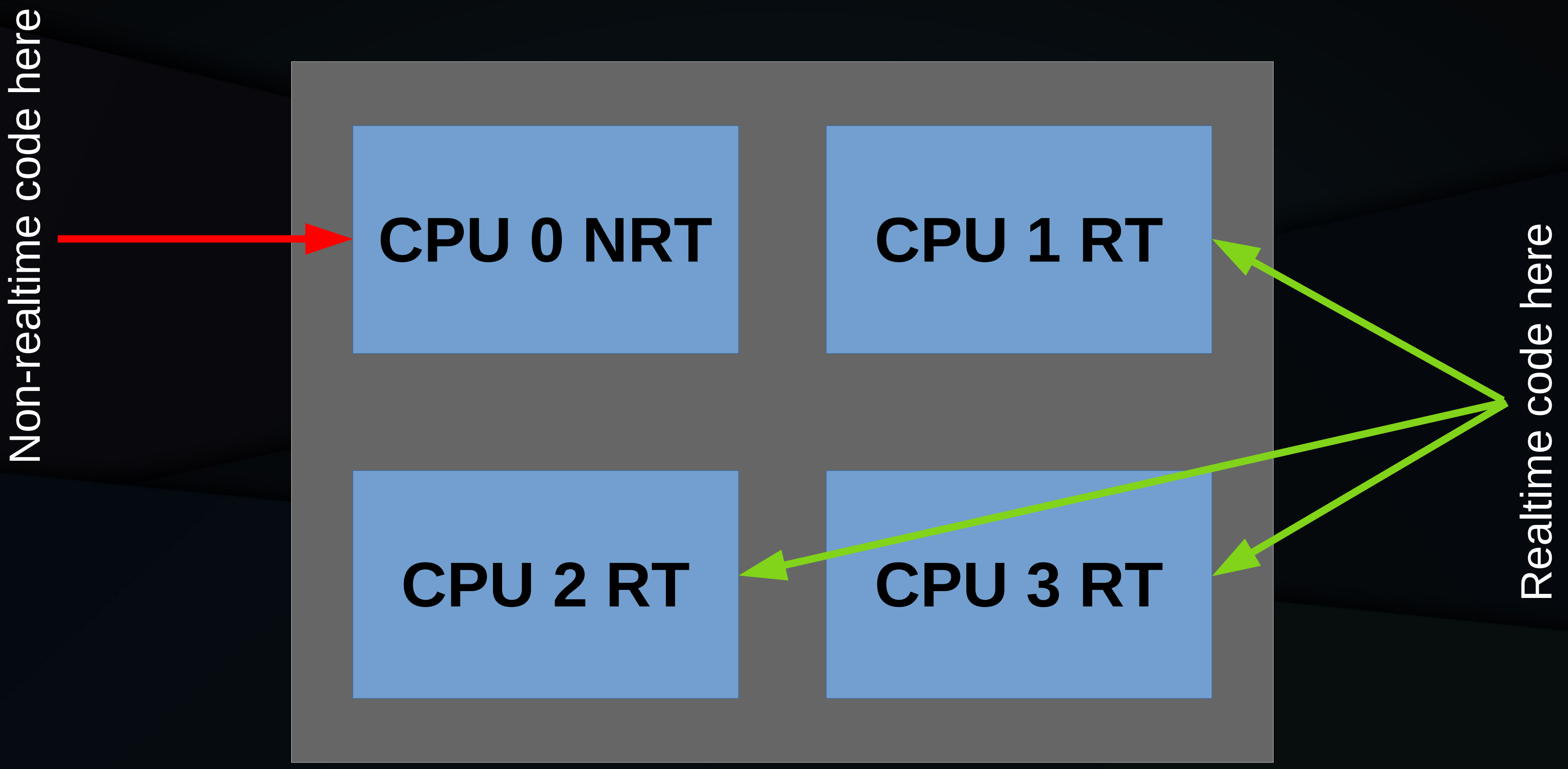
2004: Dawn of Multicore Embedded



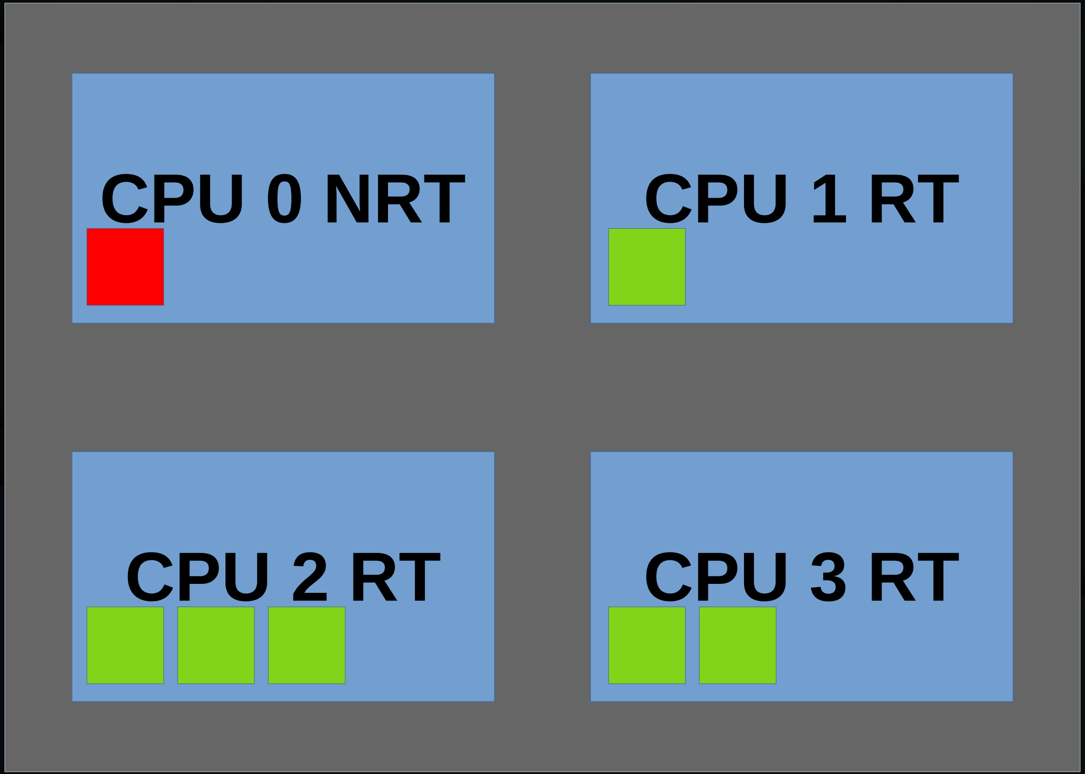
Multicore Embedded for Real Time!!!



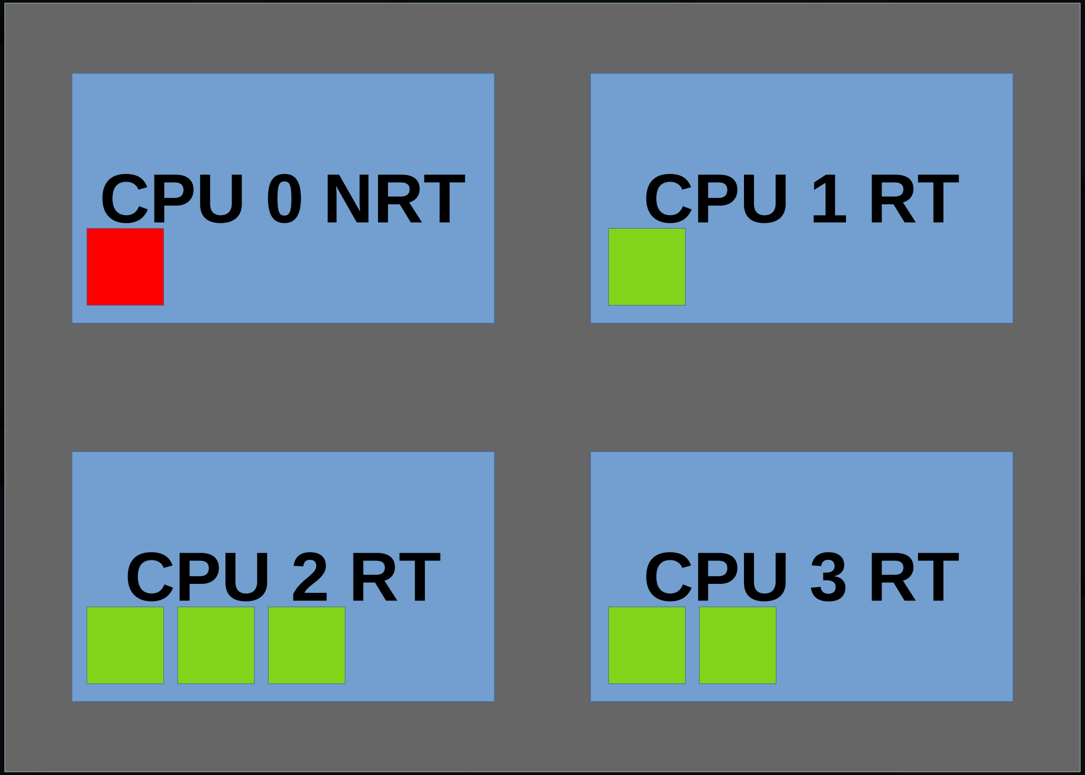
Multicore Embedded for Real Time!!!



Multicore Embedded for Real Time!!!

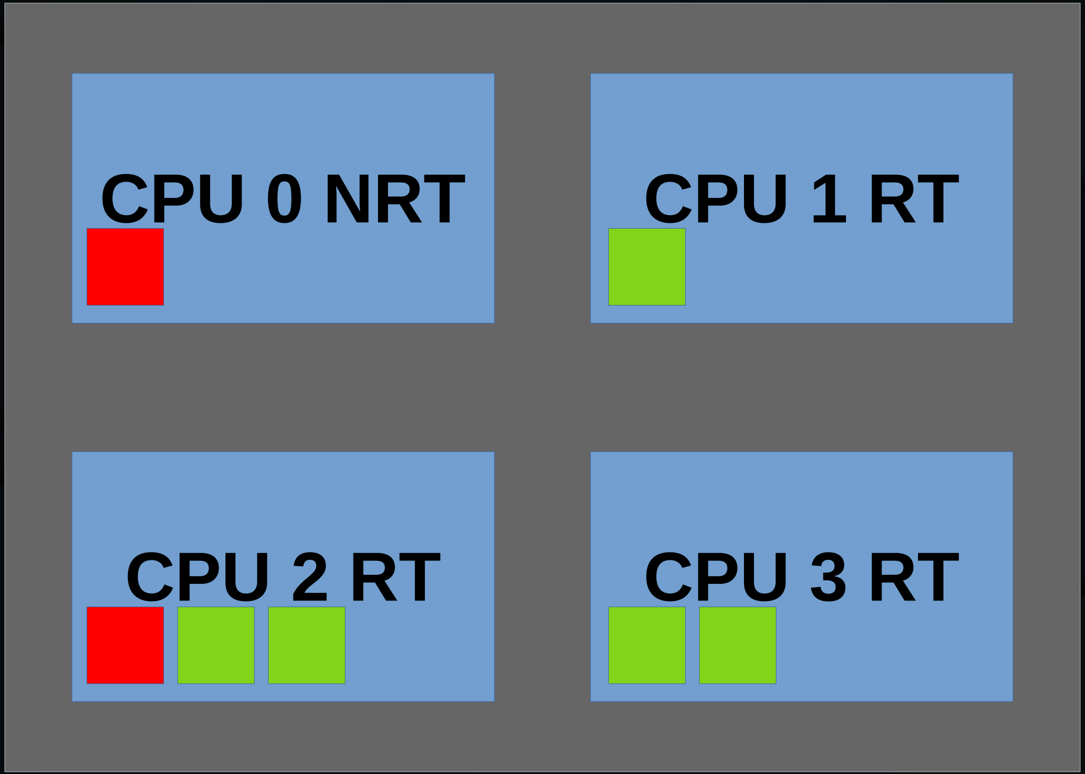


Multicore Embedded for Real Time!!!



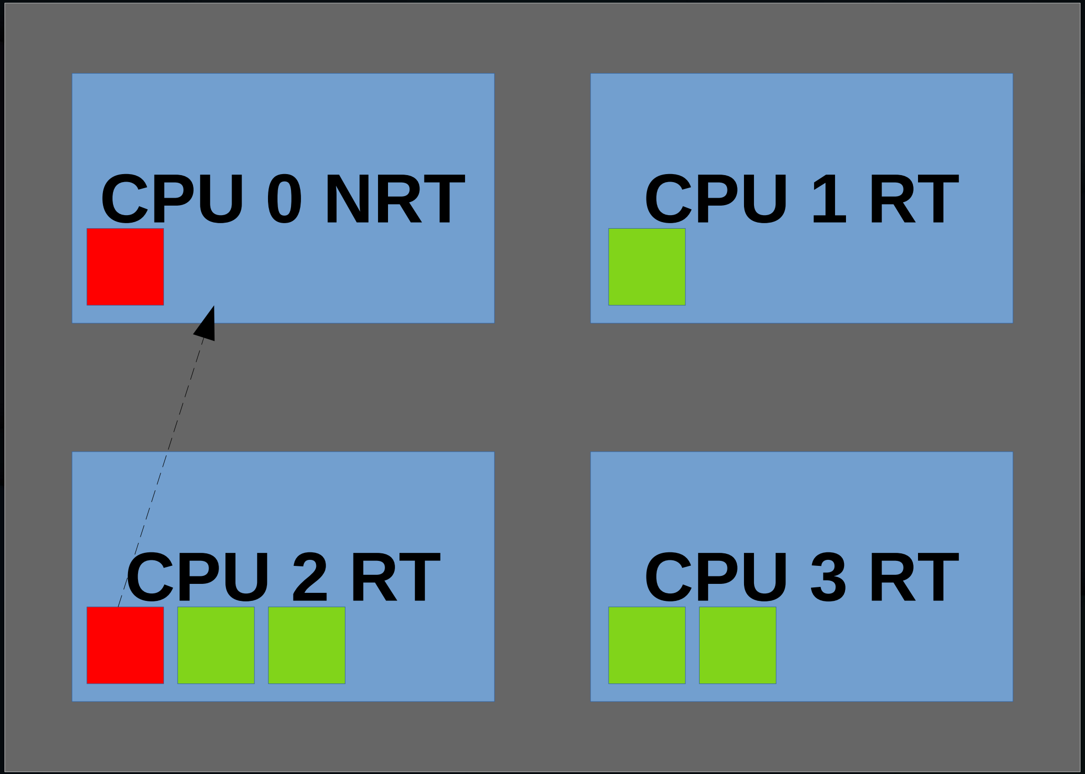
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



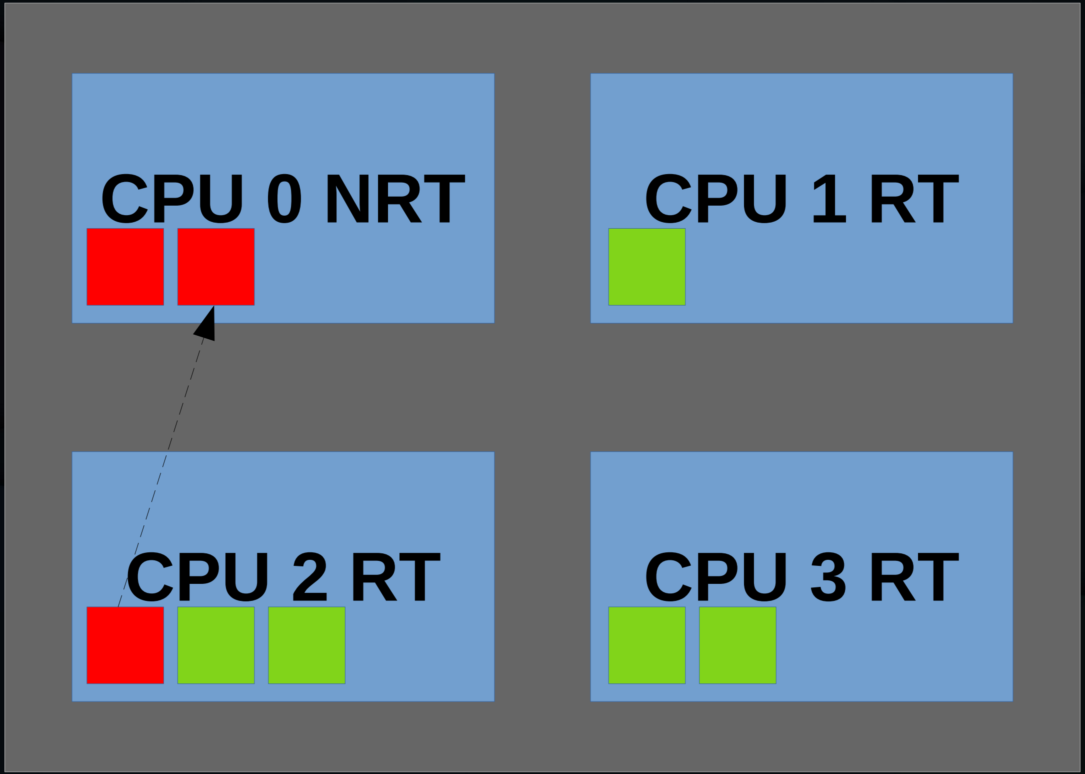
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



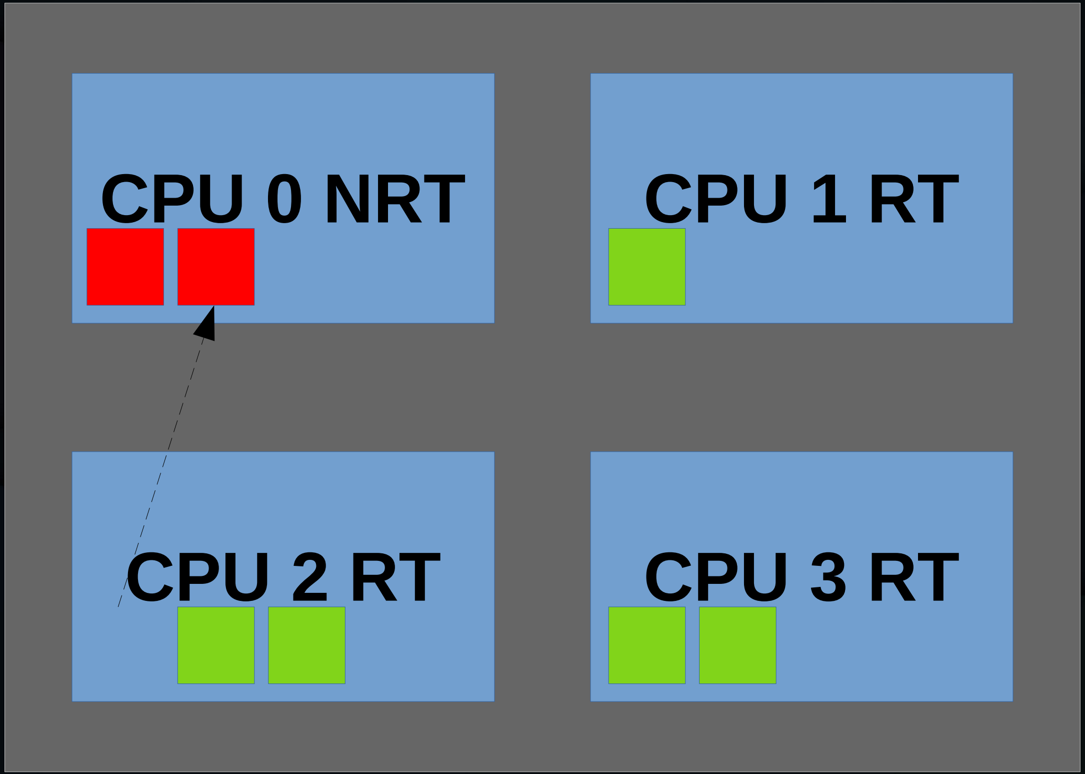
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



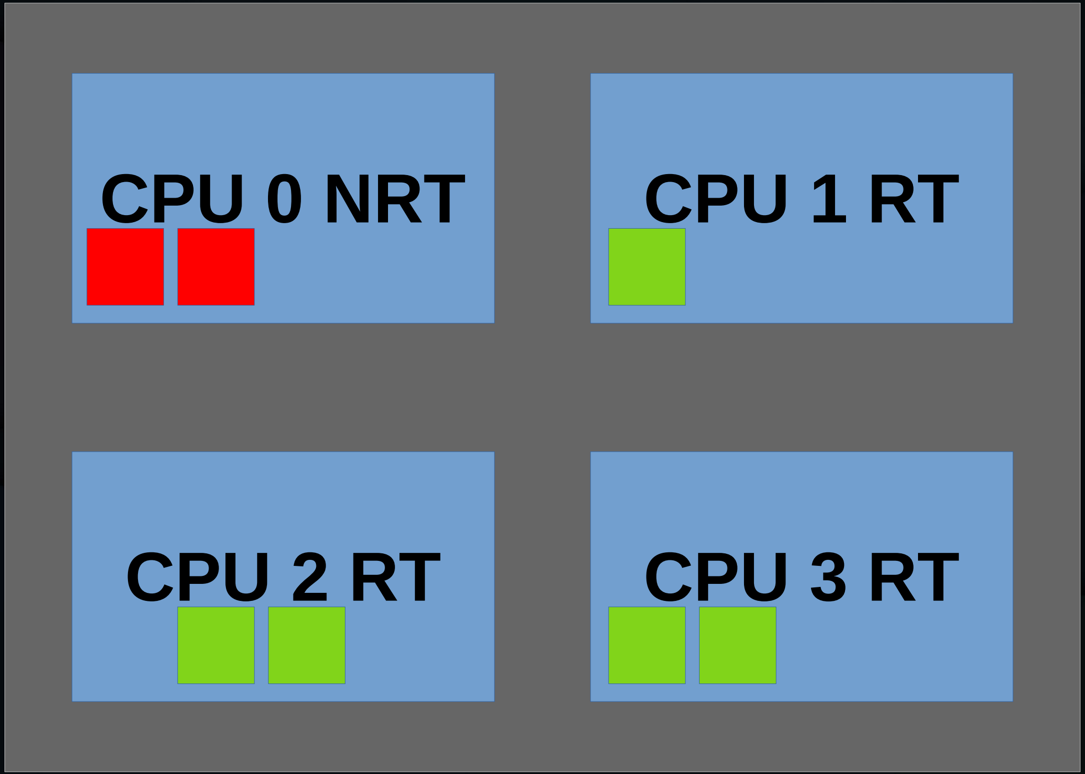
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



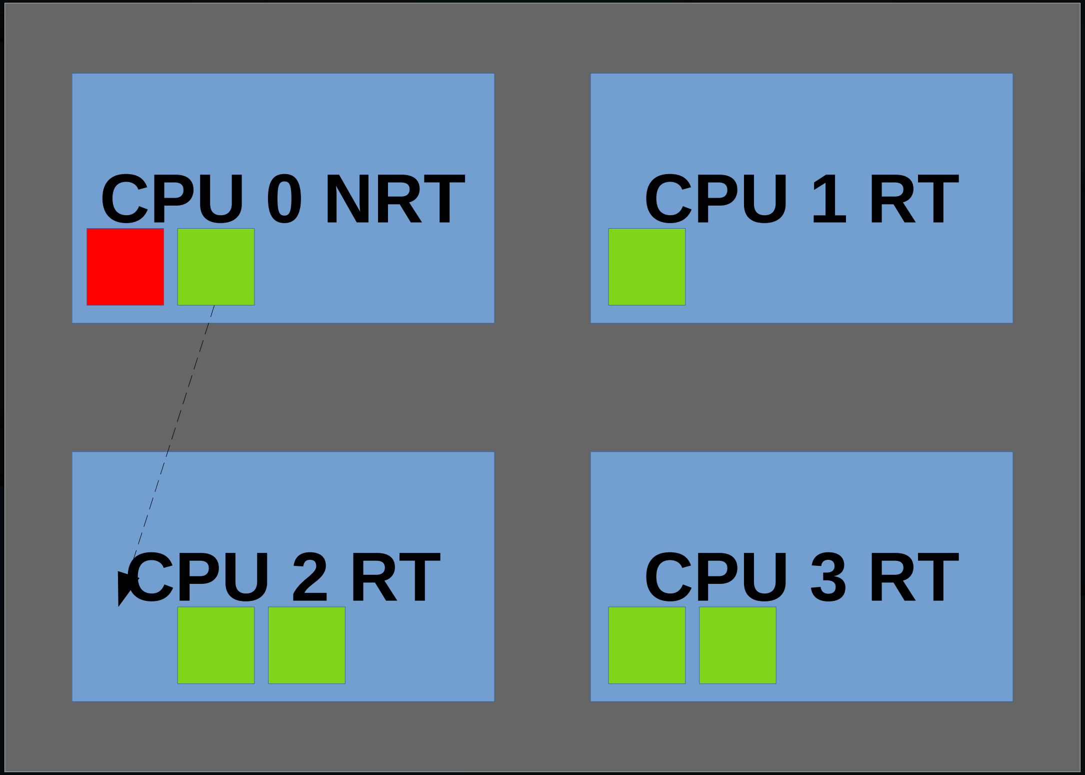
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



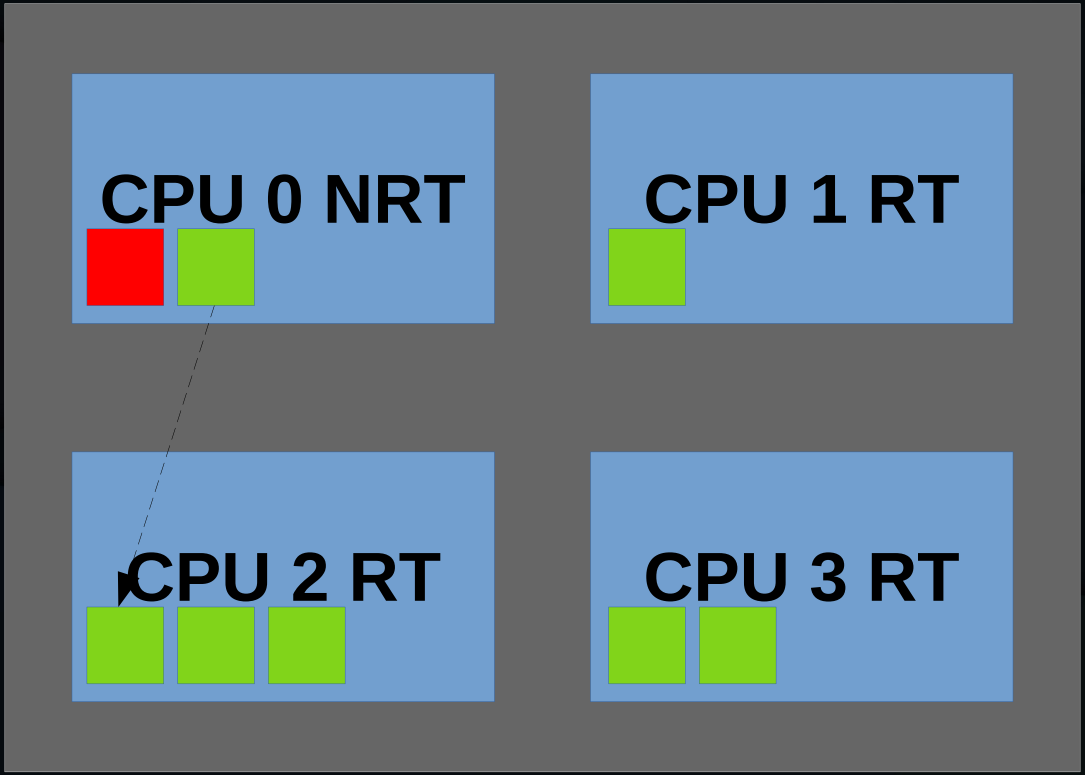
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



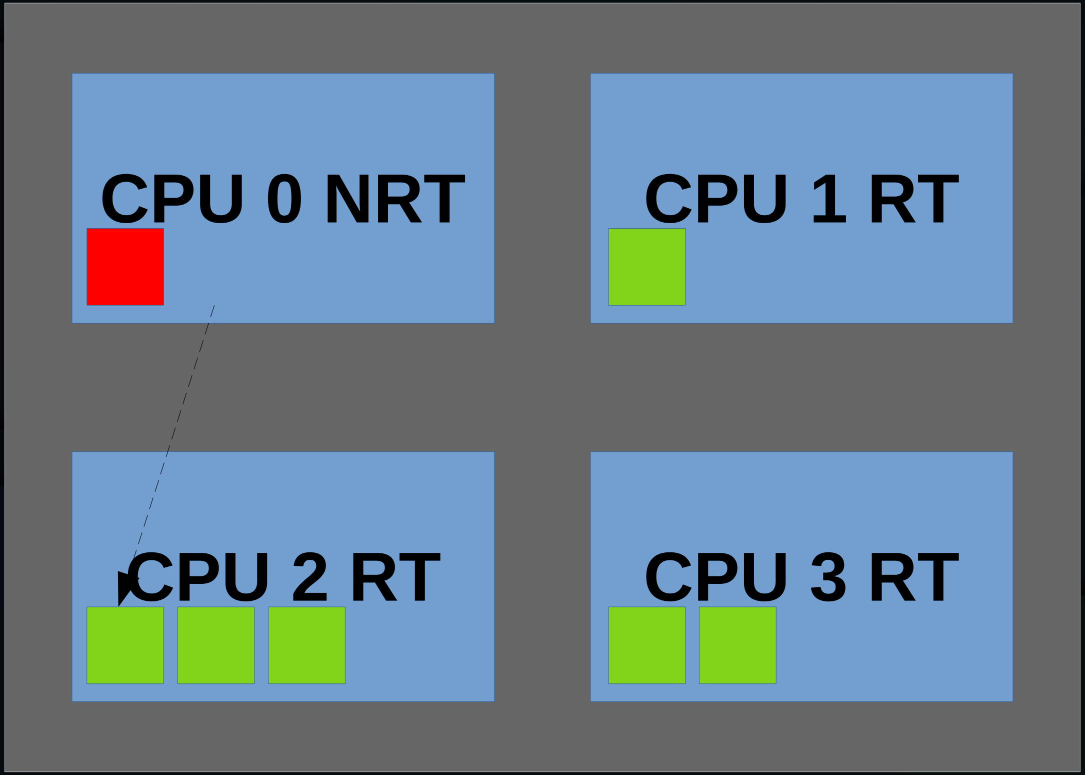
Respond to non-real-time activity by migrating.and back when done (system call).

Multicore Embedded for Real Time!!!



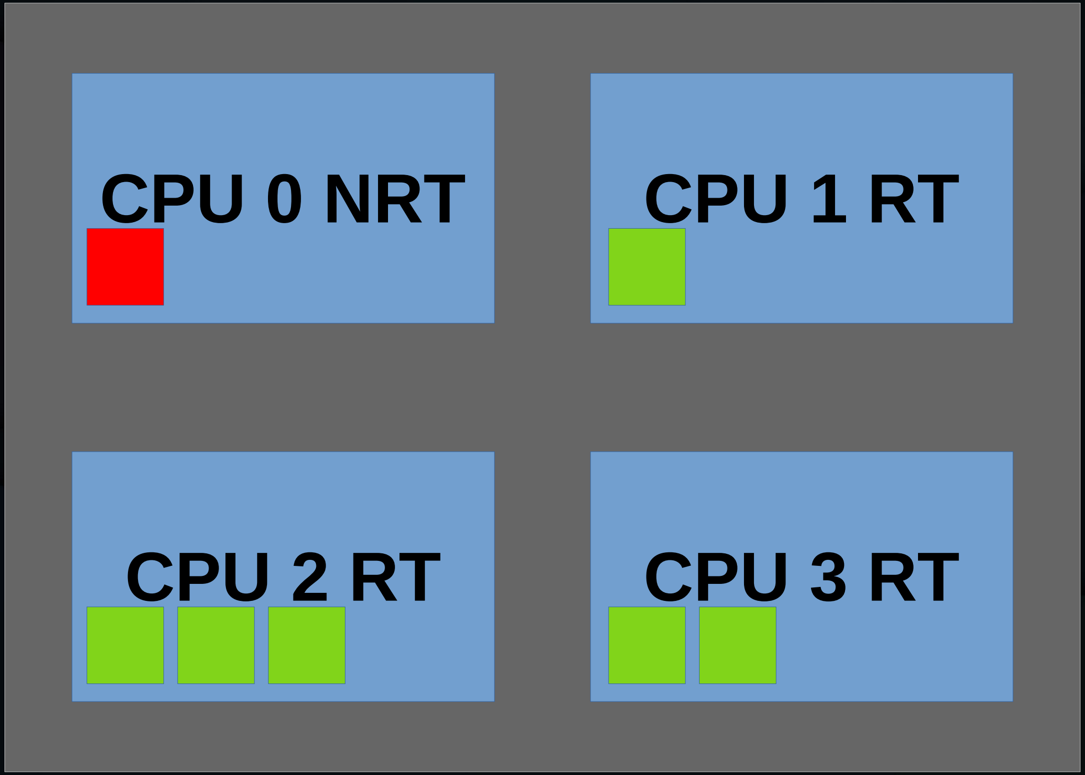
Respond to non-real-time activity by migrating.and back when done (system call).

Multicore Embedded for Real Time!!!



Respond to non-real-time activity by migrating.and back when done (system call).

Multicore Embedded for Real Time!!!



Respond to non-real-time activity by migrating.and back when done (system call).

Multicore Real Time Linux Actions

- Produce patch implementing syscall migration

Multicore Real Time Linux Actions

- Produce patch implementing syscall migration
- Test it out, works great!

Multicore Real Time Linux Actions

- Produce patch implementing sysc… …ion
- Test it out, works great!

There is a real-time effort spinning up.
But they are rewriting the kernel.
Pragmatism for the win!!!

Multicore Real Time Linux Actions

- Produce patch implementing syscall migration
- Test it out, works great!
- Inform executives real-time Linux is real!!!
- No more need for no-bid!!!

Multicore Real Time Linux Actions

- Produce patch implementing syscall migration
- Test it out, works great!
- Inform executives real-time Linux is real!!!
- No more need for no-bid!!!
- And we win a large contract!!!

Multicore Real Time Linux Actions

- Produce patch implementing sys ll migration
- Test it out, works great!
- Inform executives re e Linux is real!!!
- No more need o-bid!!!
- And we wi large contract!!!

My idea is rejected!!!

Multicore Real Time Linux Actions

- Why was my brilliant idea rejected?

Multicore Real Time Linux Actions

- Why was my brilliant idea rejected?

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

Multicore Real Time Linux Actions

- Produce patch impleme ng ea ll igration
- Test it out, works
- Inform executi nux is real!!!
- No more n
- And we wi ge tract!!!

Rejected!!!
Except that we have contractual commitments to meet...

Multicore Real Time Linux Actions

- e patch implem g all igration
- Test it out,
- Inform executi ux is real!!!
- No more n
- And we wi ge tract!!!

Remember that rewrite-the-kernel effort?

R cted!!!

Except t we have

contractual c mitments

to mee

Multicore Real Time Linux Actions

- patch implem ng all igration
- Tes
- Inform exe nux is real!!!
- No more n
- And we wi ge ntract!!!

Remember that rewrite-the-kernel effort?

Well, I helped them with RCU

cted!!!

we have

mitments

Exce

contractu

to r

Multicore Real Time Linux Actions

- ... patch implem... ...g ... all ... igration

  Tes

- I... ...nux is real!!!
- No more ...
- And we wi... ...ge ...

Remember that rewrite-the-kernel effort?

Well, I helped them with RCU

Three from-scratch implementations

...cted!!!

...t we have

...mitments

Ex...

contra...

Multicore Real Time Linux Actions

- patch impleme g all igration

  Tes

- No more

- And we wi ge

Remember that rew e-kernel effort?

cted!!!

t we have

mitments

Well,

Thr

-scratch implementations

them with RCU

Ex

contra

One of the highlights of my career

2004: Real-Time Linux

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

2004: Real-Time Linux

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

2004: Real-Time Linux

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

Formal Verification

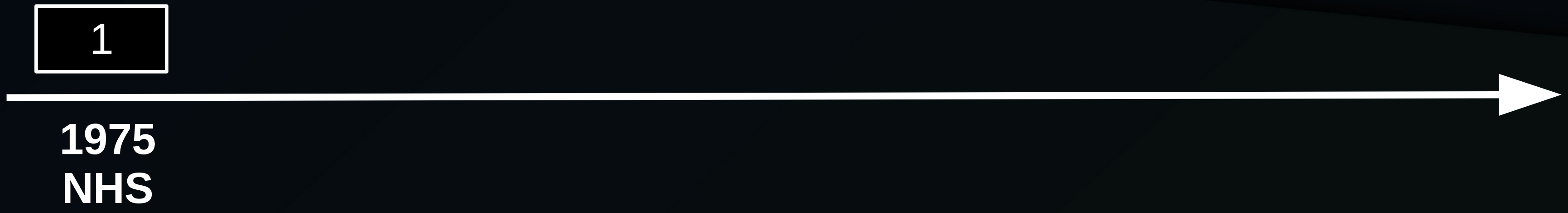
Formal Verification: Why Bother?

Installed Base

Million-Year Bug? Once In a Million Years!!!

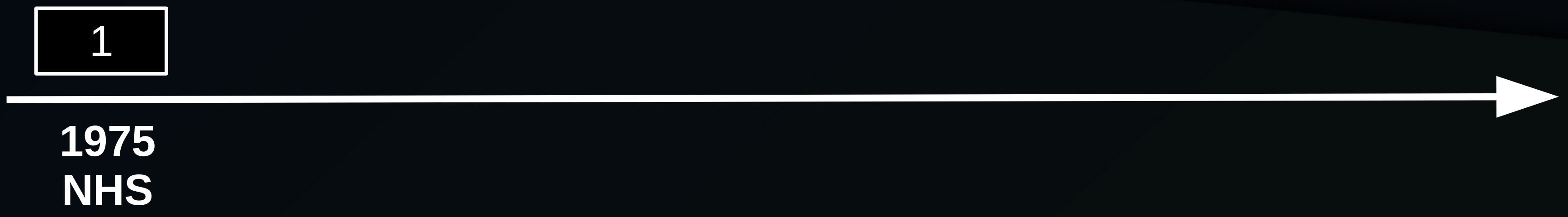
Installed Base

Million-Year Bug? Once In a Million Years!!!
Murphy is a nice guy: Everything that can happen, will...



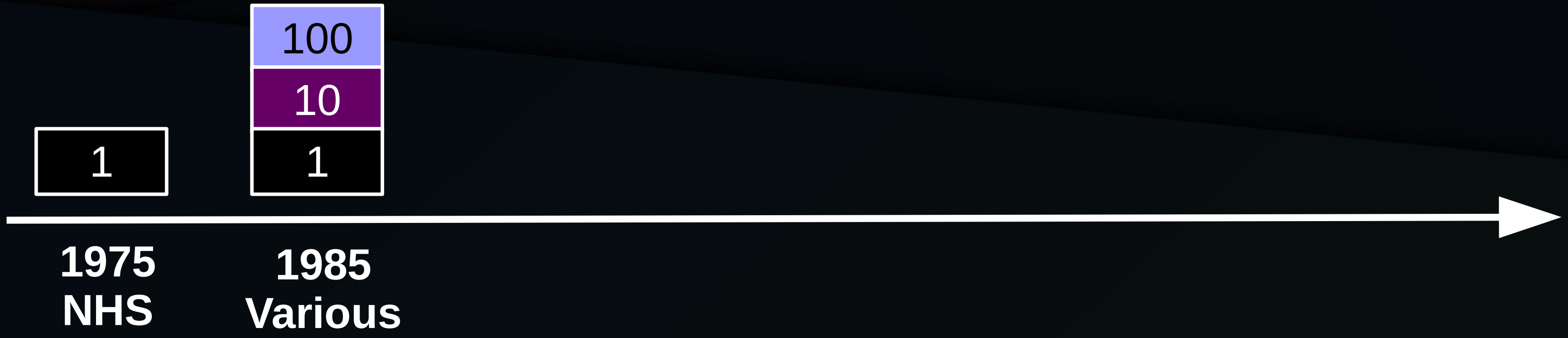
Installed Base

Million-Year Bug? Once In a Million Years!!!
Murphy is a nice guy: Everything that can happen, will...
...maybe in geologic time



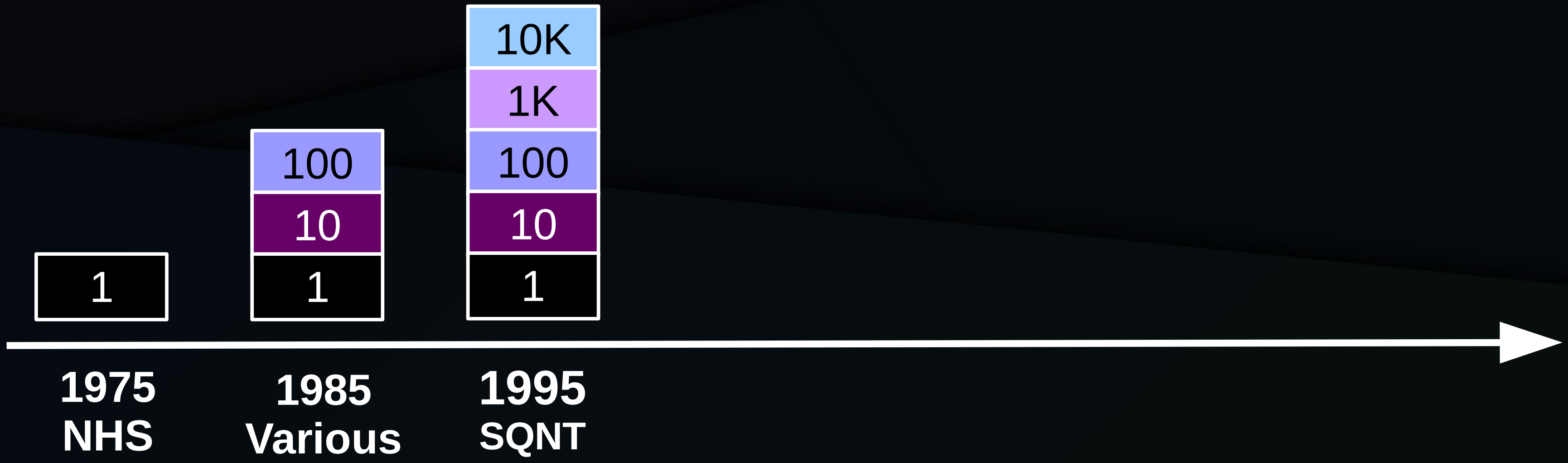
Installed Base

Million-Year Bug? Once in Ten Millennia



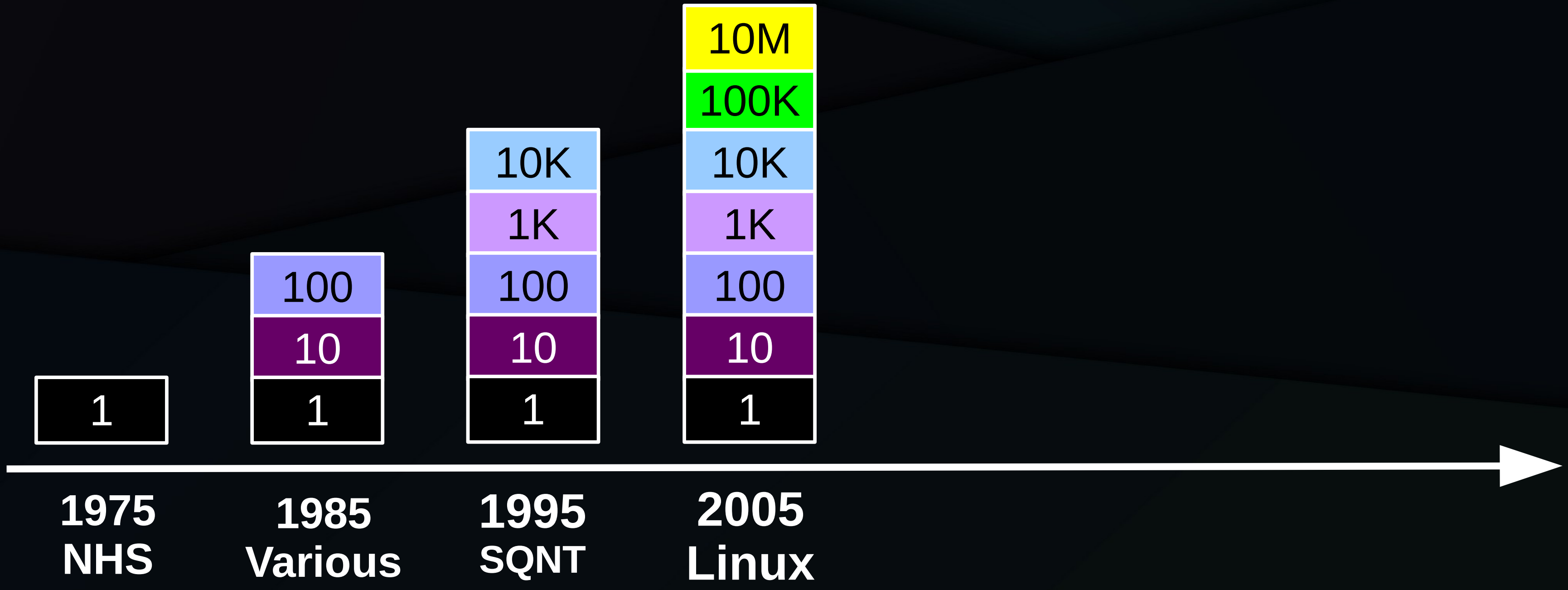
Installed Base

Million-Year Bug? Once per Century



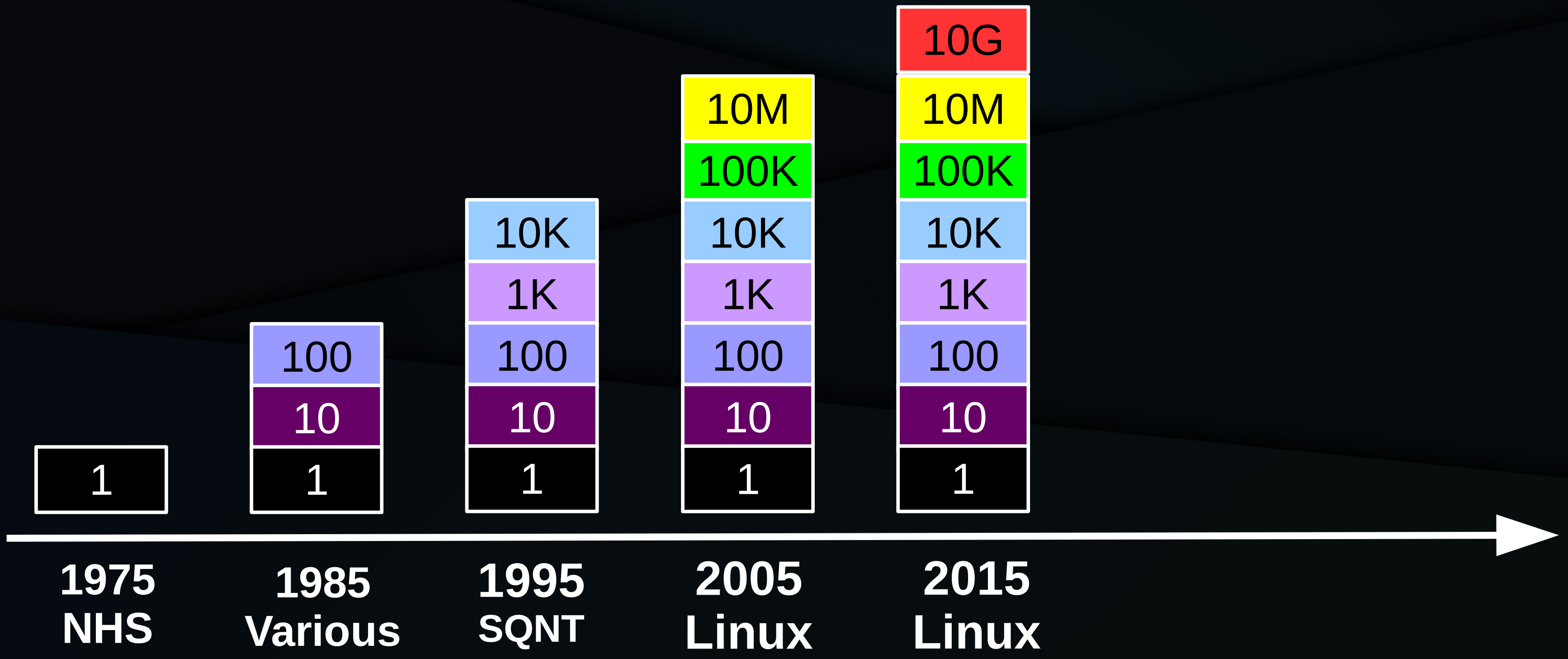
Installed Base

Million-Year Bug? Once a Month



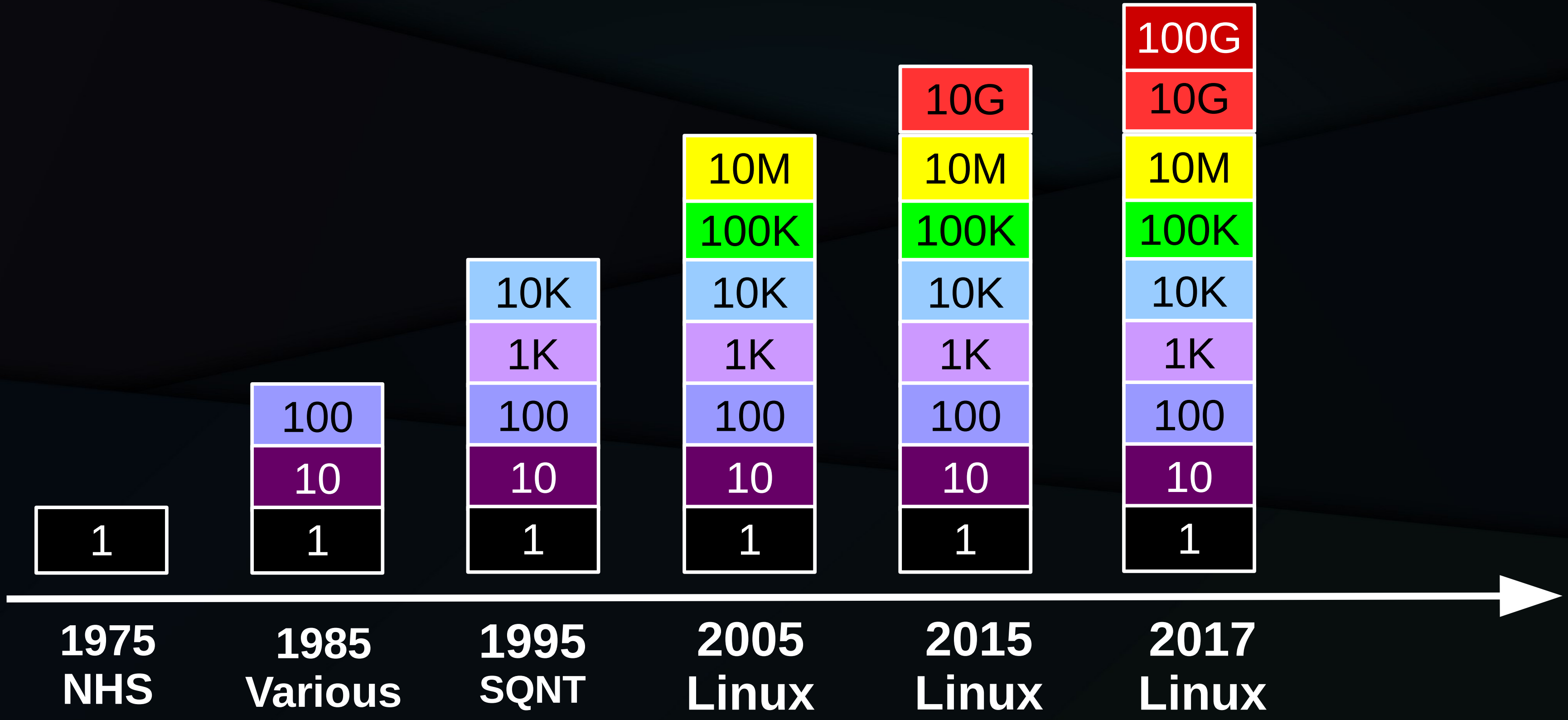
Installed Base

Million-Year Bug? Several Times per *Day*



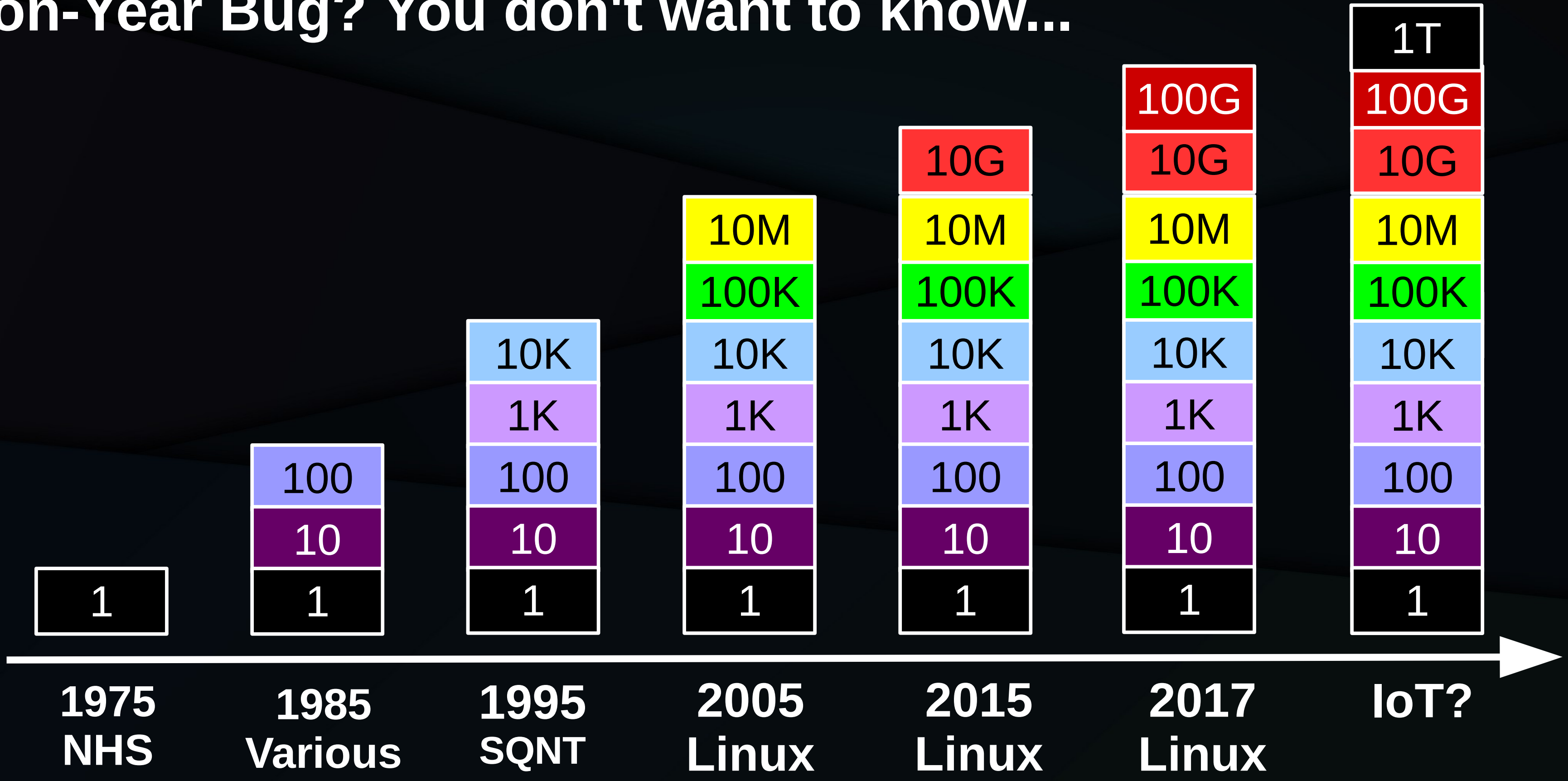
Installed Base

Million-Year Bug? Several Times per *Hour*



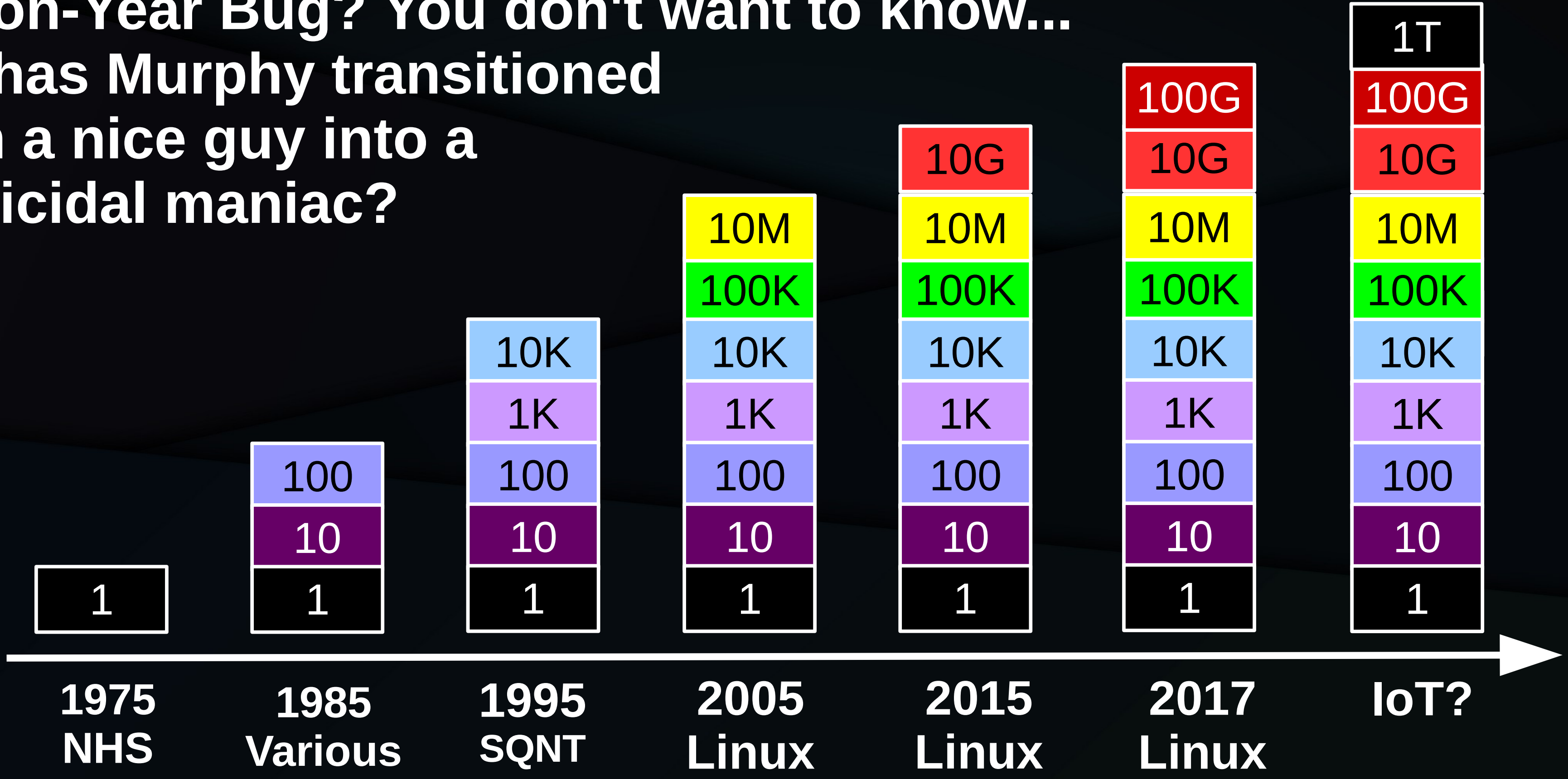
Installed Base

Million-Year Bug? You don't want to know...



Installed Base

Million-Year Bug? You don't want to know...
But has Murphy transitioned from a nice guy into a homicidal maniac?



Formal Verification: Why Bother?

- 2017: 20 billion instances of the Linux kernel
  - Million-year MTBF bug fails >50 times per day
  - New kernel version every 2-3 months
- Testing really is feasible for low-duty-cycle devices
  - But not for the ~80 million servers!!!
- Plus Linux is used in safety-critical applications!!!
- Full state-space search is quite attractive

Formal Verification Experience

- 1993: Promela/spin election algorithm
- 2007: “Quick” RCU (QRCU) verification
- 2008: RCU idle-detection for energy efficiency
- 2012: Verify userspace RCU
- 2014: Verify RCU idle detection for NMIs
- 2018-on: Heavy use of herd7 and LKMM

Formal Verification Experience

- 1993: Promela/spin election algorith
- 2007: “Quick” RCU (QRCU) v tion
- 2008: RCU idle-detectio nergy efficiency
- 2012: Verify users RCU
- 2014: Verify idle detection for NMIs
- 2018- eavy use of herd7 and LKMM

Verifying design, not regression testing

Formal Verification Experience

- 1993: Promela/spin election algorith
- 2007: "Quick" RCU (QRCU) ve
- 2008: RCU idle-detectio efficiency
- 2012: Verify users
- 2014: Verify tection for NMIs
- 2018: e of herd7 and LKMM

Verifying design, not regression testing

Verification valid after bug fix???

Formal Verification is *Expensive*

- At best, exponential; in general, undecidable
  - Partitioning for combinatorial implosion?
- “Macho” verification requires full specification
  - Which is large, thus containing lots of bugs!
- Successful formal verification highly restricted:
  - Small programs, simple properties of large programs, or execution-guided verification

Formal Verification is *Expensive*

- At best, exponential; in general, unde
  - Partitioning for combinatorial impl
- “Macho” verification requir cation
  - Which is large, thus bugs!
- Successful for highly restricted:
  - Small properties of large programs, or exc verification

Powerful when used properly, static analysis can be fast

Formal Verification is *Expensive*

- At best, nential; in general, unde
  - Partitioning binatorial impl
- "Macho" verificatio i cation
  - Which is large, thus bugs!
- Successful for on restricted:
  - Small p properties of grams, or exc erification

How to verify the verification?

Powerful when used properly, static analysis can be fast

Remember That File Cabinet?

Formal Verification’s Scope Is Limited

- "Everyone nows that debugging is twice as har ng a program i st place. So if you're as cl n be when you wri will you ever d . *Kernighan*, "The E of Pro 2nd Edition, Chapter 2.
- While programming ssful ignorance of important requi equ make themselves known dur
- Whic se of Kernighan’s observati n.

I failed to understand that I was competing with a file cabinet

And the file cabinet won

Formal Verification’s Scope Is Limited

- Does anyone really want the software?
- Is the software economically valuable?
  - Enough to pay the software’s developers? Validation personnel? Service personnel? Sales? Documentation? Maintenance?
- Are any supply chains robust?
- Are the requirements correct? Complete?
- Are the requirements met?
  - Functional requirements? Performance requirements? Non-real-time latency requirements? Real-time latency requirements? Energy-efficiency requirements? Human-factors requirements? Legal requirements? Human-language requirements?

Formal Verification's Scope Is Limited

- Does anyone really want the software?
- **Is the software economically valuable?**
  - **Enough to pay the software's developers? Validation personnel? Service personnel? Sales? Documentation? Maintenance?**
- Are any supply chains robust?
- Are the requirements correct? Complete?
- Are the requirements met?
  - Functional requirements? Performance requirements? Non-real-time latency requirements? Real-time latency requirements? Energy-efficiency requirements? Human-factors requirements? Legal requirements? Human-language requirements?

Real-Time Linux System Options

1) Special system for this bid
2) New real-time product line
3) Put real-time capabilities into standard product

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

Real-Time Linux System Option 1

- Special system for this bid
  - Low development cost for group producing server
  - High development cost for real-time Linux group
  - High likelihood of firmware issues
  - High service costs for real-time Linux group
  - So-so customer experience

Real-Time Linux System Option 2

- New real-time product line
  - High development cost for group producing server
  - Low development cost for real-time Linux group
  - Lower likelihood of firmware issues
  - Low service costs for real-time Linux group
  - Good customer experience

Real-Time Linux System Option 3

- Put real-time capabilities into standard product
  - Negative costs (!) for group producing server
  - Low development cost for real-time Linux group
  - Lower likelihood of firmware issues
  - Low service costs for real-time Linux group
  - Good customer experience for many customers

Real-Time Linux System Options

1) ~~Special system for this bid~~
2) ~~New real-time product line~~
3) Put real-time capabilities into standard product

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

Real-Time Linux System Options

1) ~~Special system for this bid~~
2) ~~New real-time produc~~
3) Put real-tim standard product

Great things can happen if techies and business people work together!!!

Photo credit: National Museum of the U.S. Navy, 2016, Public Domain Zumwalt-class DDG destroyer

Formal Verification is Heavily Used

- Several test projects on the Linux kernel
- Many proprietary projects verify each commit
- But…
  - Formal verification in the small
  - Check for undesirable properties
    - File bug reports as appropriate

Formal Verification is Heavily Used

- Several test projects on the Linux kernel
- Many proprietary projects ve ach commit
- But…
  - Formal verificatio ne small
  - Check for u irable properties
    - File b eports as appropriate

De-risk via one-way bet

Cautionary Quote (Redux)

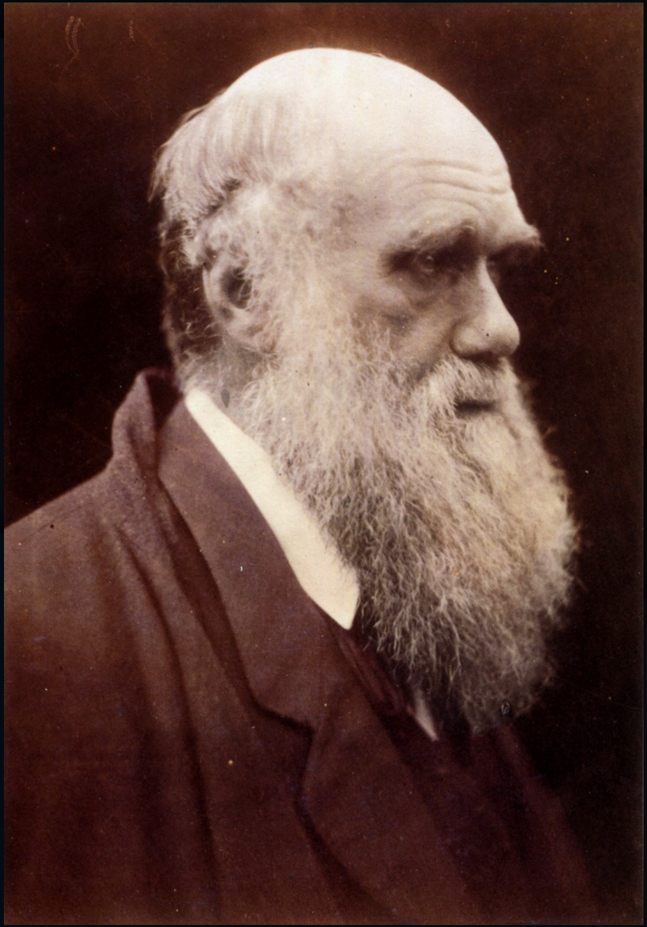
- A lot of success in life and business comes from knowing what you want to avoid. - *Charlie Munger*

Cautionary Quotes

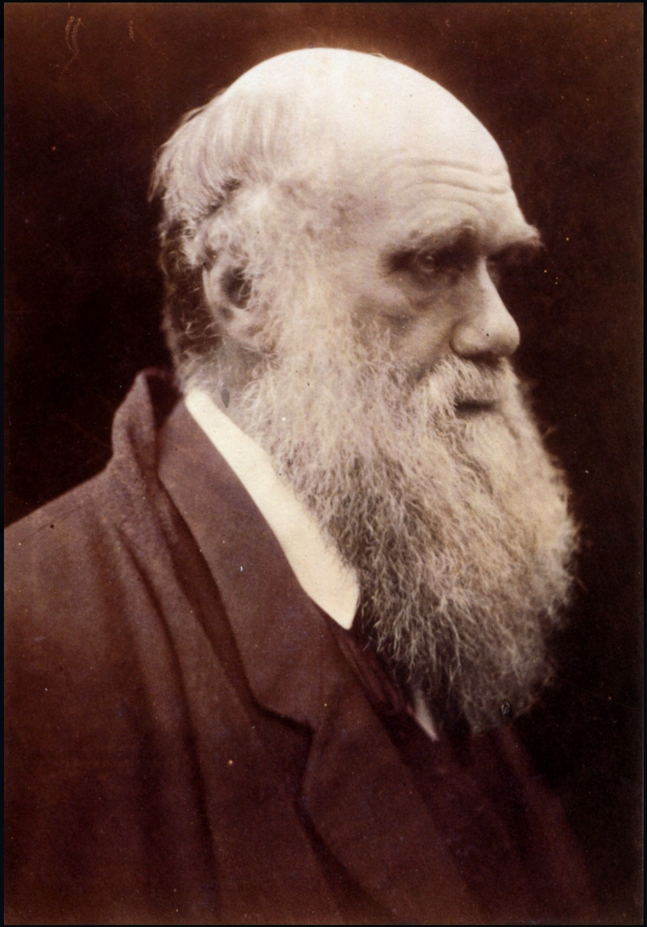
- Sometimes you don't even know what you want until you find out you can't have it. - *Meghan O'Rourke*
- Sometimes we don't know what we want until we don't get it. - *Sloane Crosley*
- We don't know what we want, but we are ready to bite somebody to get it - *Will Rogers*

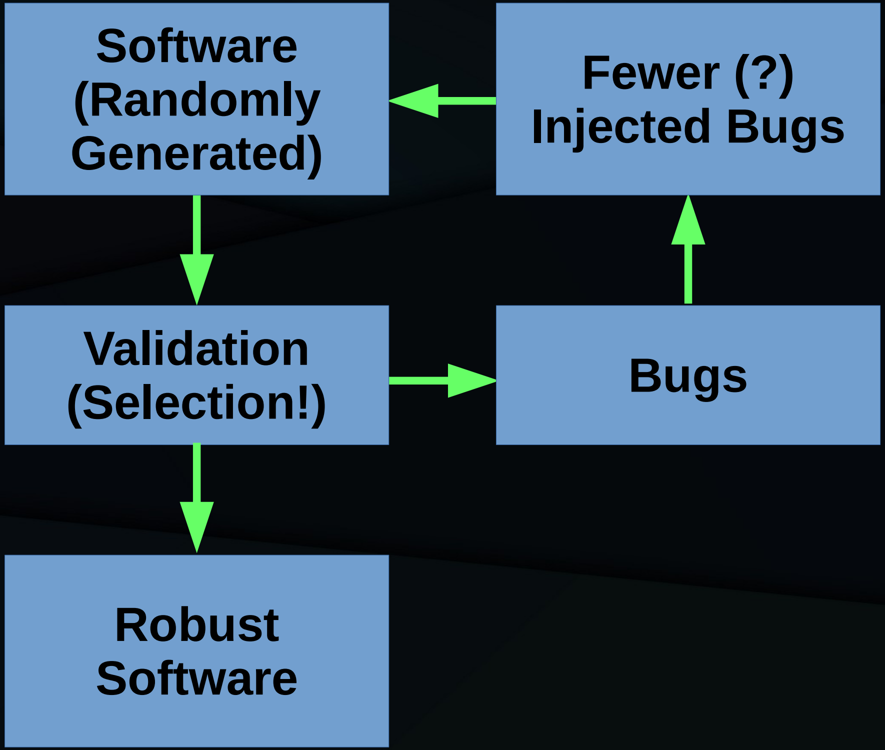
Natural Selection

Natural Selection



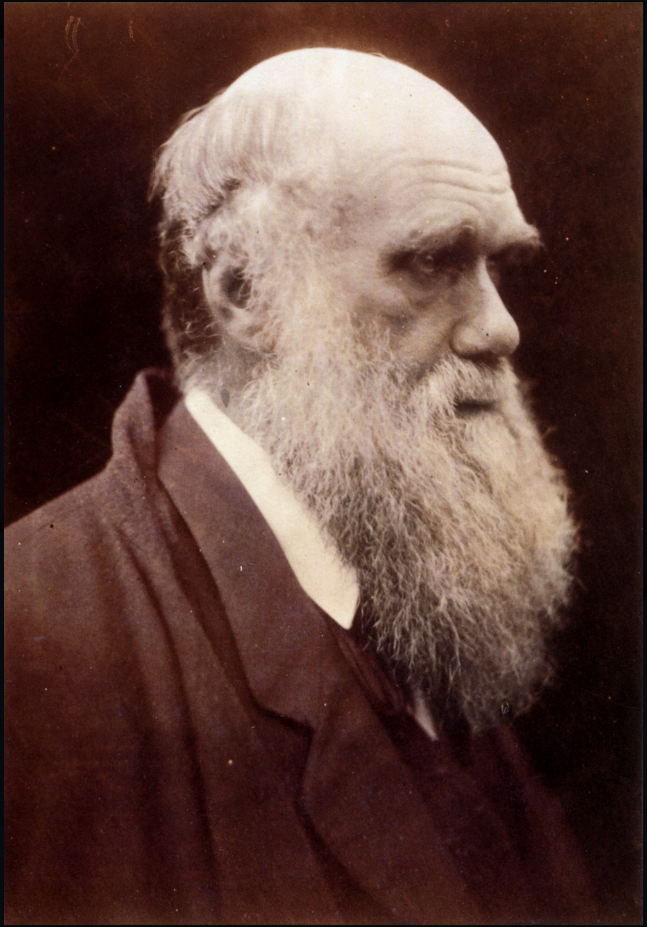
Natural Selection: Not Just Lifeforms

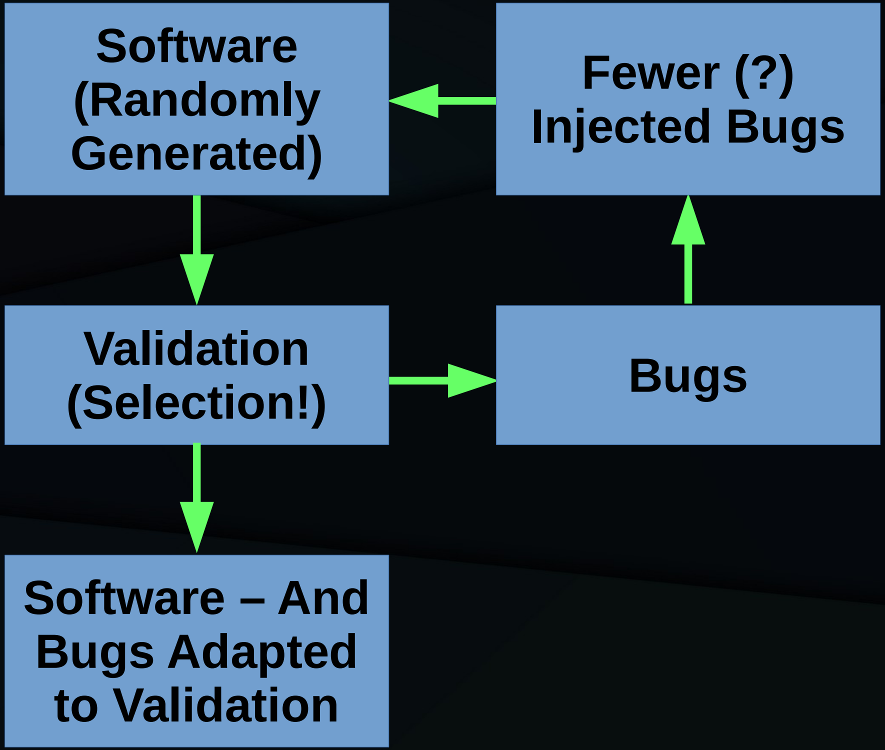




Agile methods attempt to push this methodology back to the specification

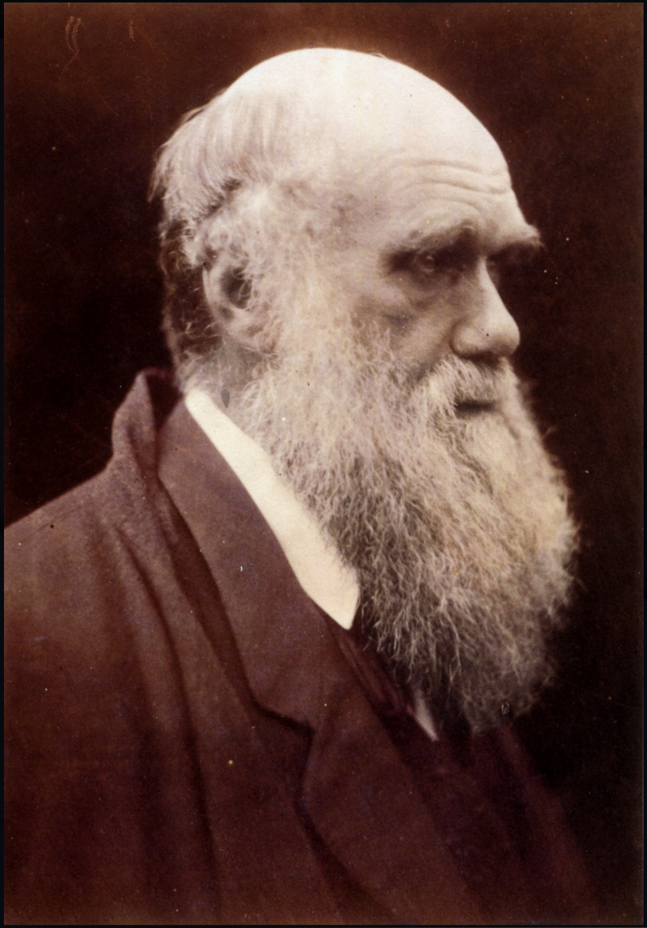
Natural Selection: Bugs are Software!

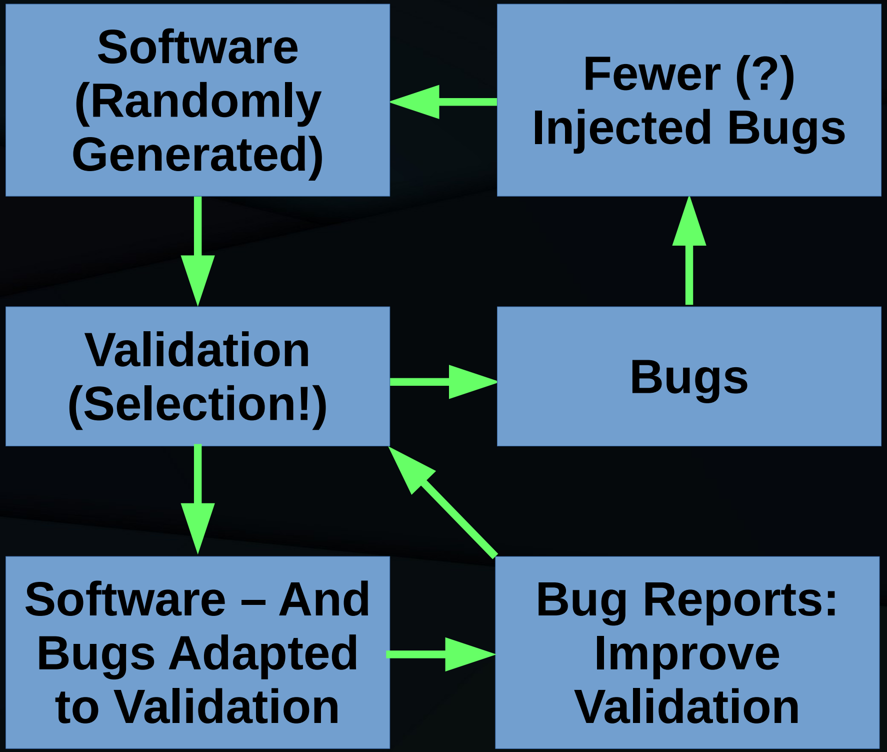




Software (Randomly Generated)
Fewer (?) Injected Bugs
Validation (Selection!)
Bugs
Software – And Bugs Adapted to Validation

Natural Selection: Bugs are Software!



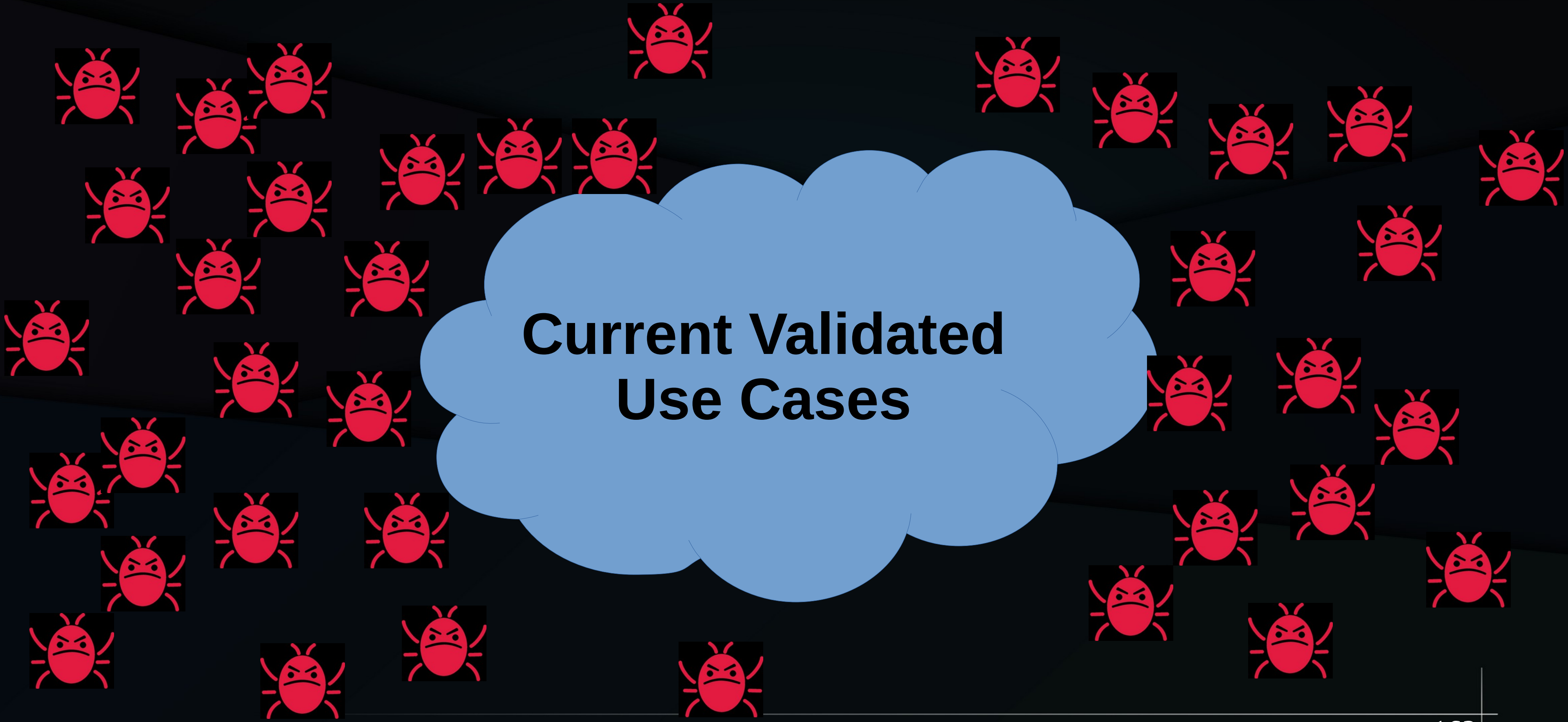

Software (Randomly Generated)
Fewer (?) Injected Bugs
Validation (Selection!)
Bugs
Software – And Bugs Adapted to Validation
Bug Reports: Improve Validation

Validate Only Intended Use Cases

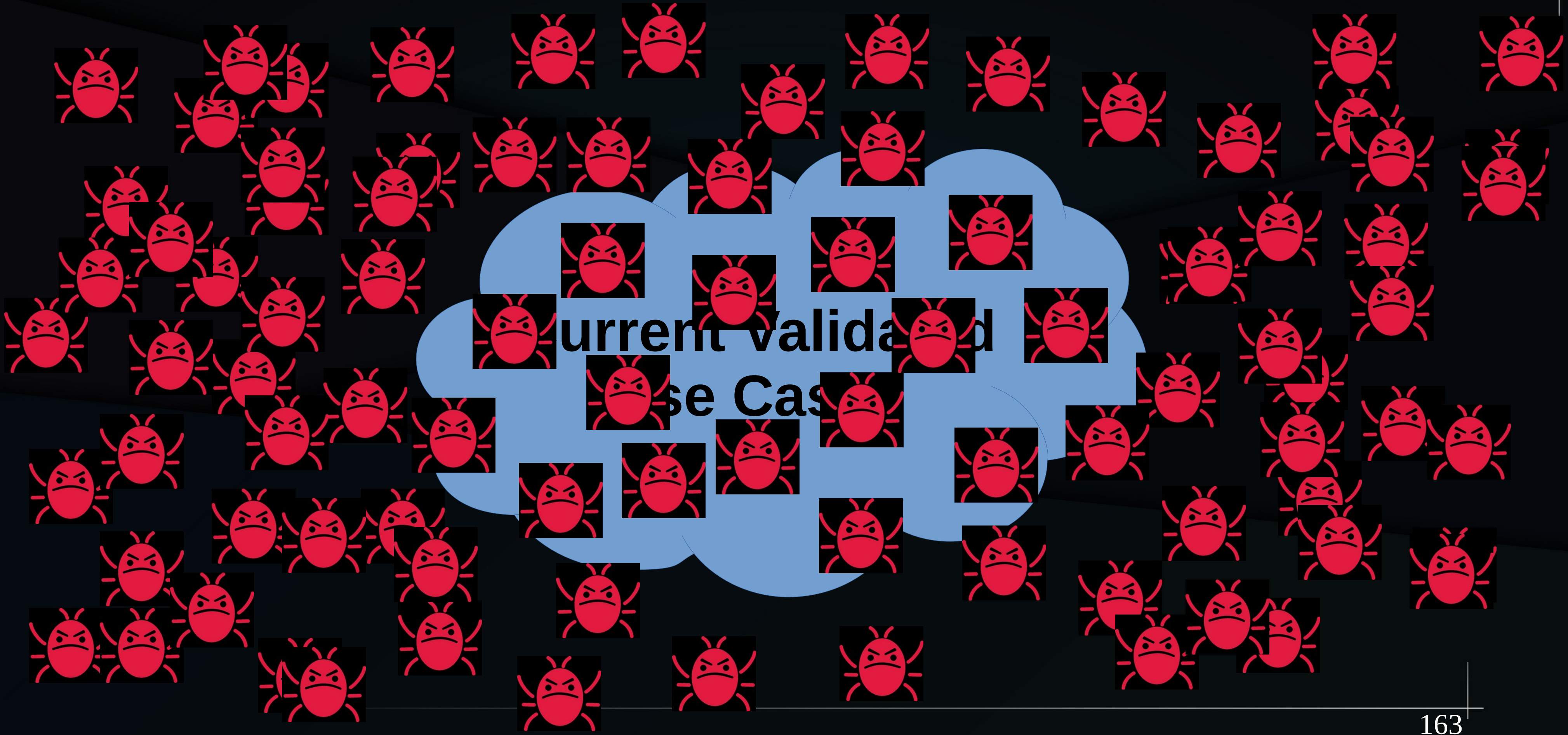
Current Validated Use Cases

Major Development Generates Bug

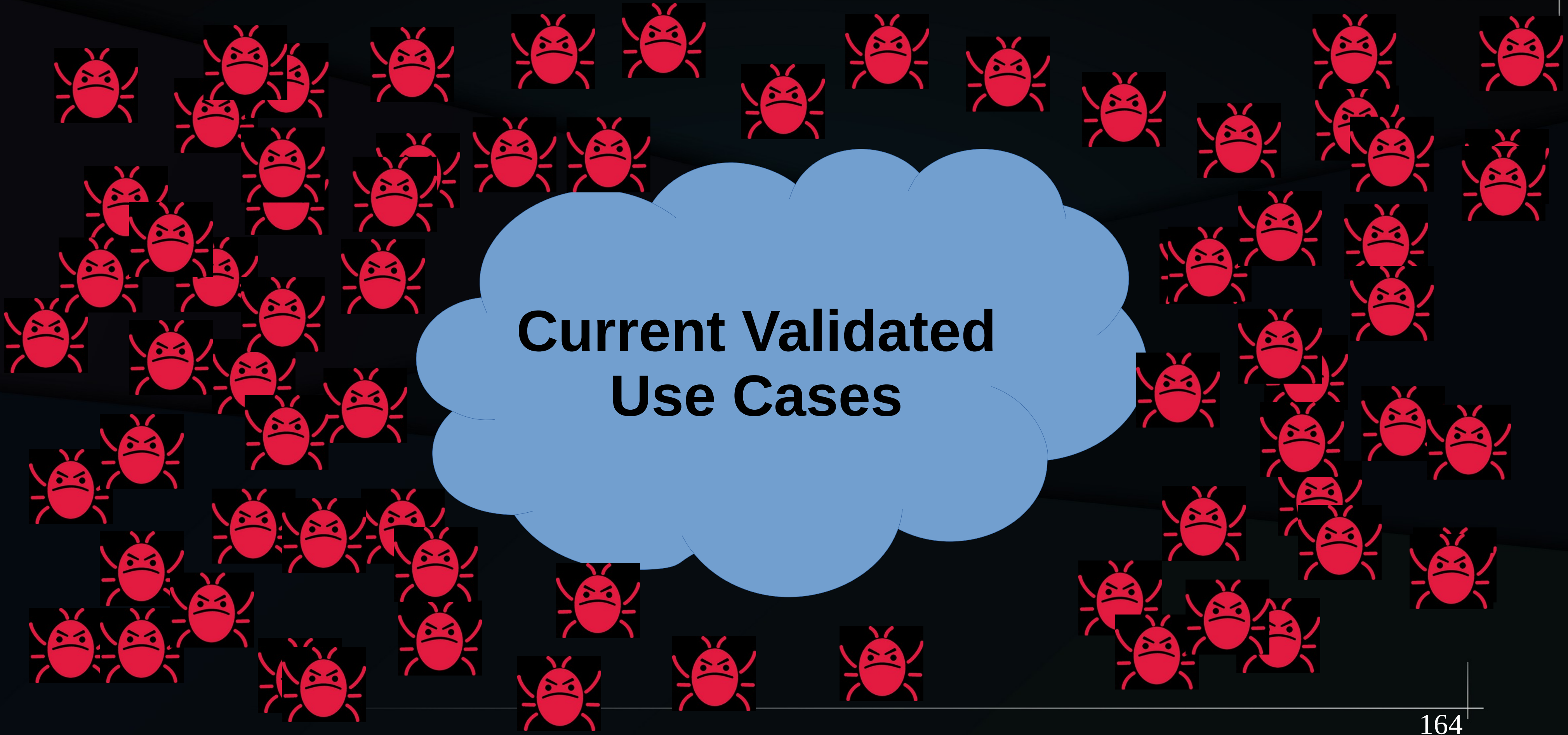
After Validation and Bug Fixing



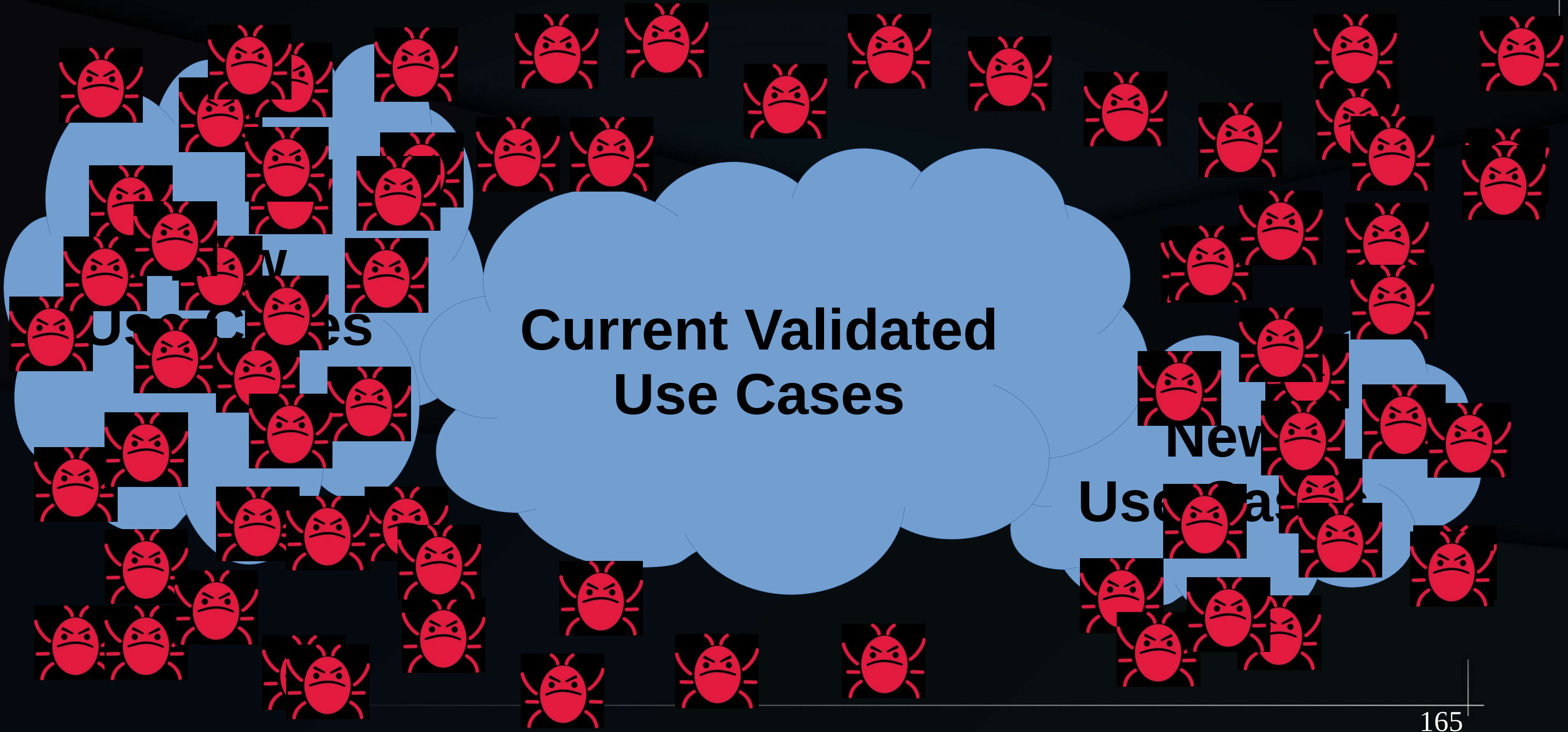
After Another Round of Development



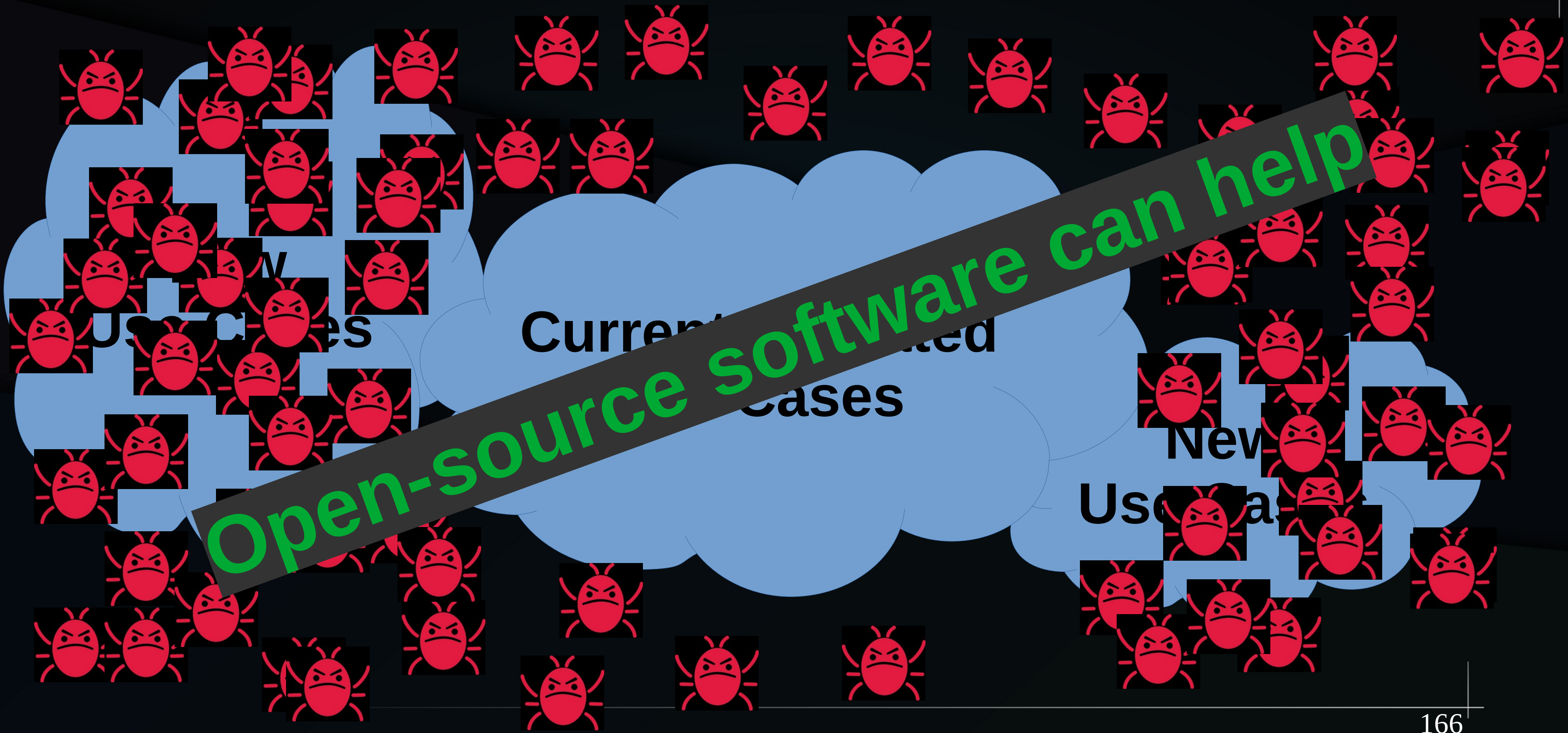
More Validation and Bug Fixing



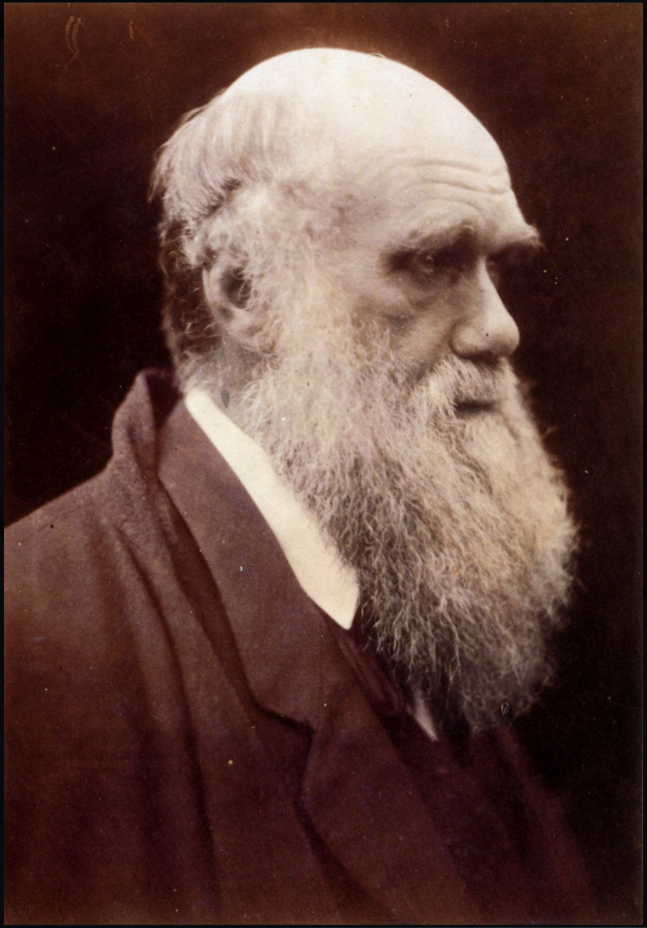
New Use Cases: Walls of Bugs!!!

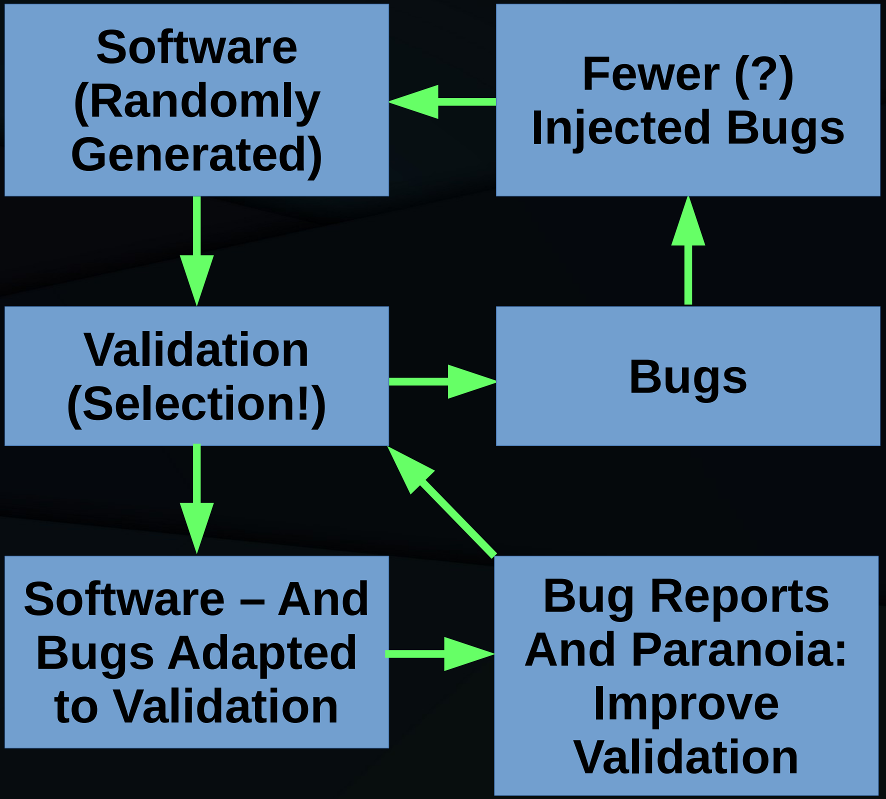


New Use Cases: Walls of Bugs!!!



Natural Selection: Bugs are Software!





$$\begin{tikzcd}
\text{Software (Randomly Generated)} \arrow[d] & \text{Fewer (?) Injected Bugs} \arrow[l] \\
\text{Validation (Selection!)} \arrow[r] \arrow[d] & \text{Bugs} \arrow[u] \\
\text{Software – And Bugs Adapted to Validation} \arrow[r] & \text{Bug Reports And Paranoia: Improve Validation} \arrow[ul]
\end{tikzcd}$$

“Natural Selection” is a Euphemism

“Natural Selection” is a Euphemism

If your tests are not failing, they are not improving your software

“Natural Selection” is a Euphemism

If your tests are not failing, they are not improving your software
If your users are not complaining, they are not improving your software

Why Would Users Fail to Complain?

- They are not actually using your software (common case)
- They do not know who to complain to
- The last N times they complained:
  - Nothing useful happened
  - They were yelled at or otherwise belittled
- Your software is technically successful
  - And has thus “faded into the woodwork”

Cautionary Quotes

- Customers don’t know what they want until we’ve shown them. - *Steve Jobs*

Cautionary Quotes

- Customers don’t know what they want until we’ve shown them. - *Steve Jobs*
- If there is any one secret of success, it lies in the ability to get the other person’s point of view and see things from that person’s angle as well as from your own. - *Henry Ford*

Cautionary Quotes

- Customers don’t know what they want
  we’ve shown them. - *Steve Job*
- If there is any one sec uccess, it lies in
  the ability to get ther person’s point of view
  and see from that person’s angle as well
  your own. - *Henry Ford*

You must live among your users

Cautionary Quotes

- Customers don't know what they want u
  we've shown them. - *Steve Job*
- If there is any one sec   t lies in
  the ability to get   on's point of view
  and see t   person's angle as well
  a   . - *Henry Ford*

You must live among your users

You must complain on their behalf

Summary

Summary

- People don’t know what they want

Summary

- People don’t know what they want
- But for software developers, this is no excuse

Summary

- People don’t know what they want
- But for software developers, this is no excuse
  - You have only failed if you have given up...until then it's called learning. - *Unknown*

Summary

- People don’t know what they want
- But for software developers, this is no excuse
  - You have only failed if you have given up...until then it's called learning. - *Unknown*
  - You are not a failure until you start blaming others for your mistakes. - *John Wooden*

But I Had It Easy!

- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
  - Do the assigned work
- Contract programming (1981-1985)
- Systems administration and Internet research (1986-1990)
- Concurrent proprietary UNIX (1990-2000)
- Linux kernel concurrency and realtime (2001-present)

But I Had It Easy!

- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
- Contract programming (1981-1985)
  - Keep them out of court (or even out of jail)
- Systems administration and Internet research (1986-1990)
- Concurrent proprietary UNIX (1990-2000)
- Linux kernel concurrency and realtime (2001-present)

But I Had It Easy!

- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
- Contract programming (1981-1985)
- Systems administration and Internet research (1986-1990)
  - Keep users happy and fulfill terms of research contracts
- Concurrent proprietary UNIX (1990-2000)
- Linux kernel concurrency and realtime (2001-present)

But I Had It Easy!

- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
- Contract programming (1981-1985)
- Systems administration and Internet research (1986-1990)
- Concurrent proprietary UNIX (1990-2000)
  - Make it fast and scalable (up to 64 CPUs)
- Linux kernel concurrency and realtime (2001-present)

But I Had It Easy!

- High school class: IBM mainframe & HP Basic (1973-1976)
- University: Computer science & mechanical engineering, business applications (1976-1981)
- Contract programming (1981-1985)
- Systems administration and Internet research (1986-1990)
- Concurrent proprietary UNIX (1990-2000)
- Linux kernel concurrency and realtime (2001-present)
  - Make it many things...

But I Had It Easy!

- Linux kernel concurrency and realtime (2001-present):
  - Fast and scalable (up to 4096 CPUs)
  - Real-time response (sub-20-microsecond latencies)
  - Energy efficiency
  - Near-bare-metal efficiency to usermode applications
  - Robustness (20 billion instances)
  - Ease of use driven by security
  - Ease of administration (large data centers)

But I Had It Easy!

- Linux kernel concurrency and realtime (2001):
  - Fast and scalable (up to 4096 CPUs)
  - Real-time response (sub-20
  - Energy efficiency
  - Near-bar ions
  - ase y
  - Ease o ion (large data centers)

Yes, I am proud of my accomplishments, but modern systems are far more complex and user-centric

But I Had It Easy!

- Linux ke[illegible] concurrency and realtime (200[illegible]):
  - Fast an[illegible] to 4096 CPUs)
  - Real-time respons[illegible]
  - Energy efficiency
  - Near-bar[illegible]ns
  - [illegible]
  - [illegible]ase [illegible]y
  - Ease o[illegible]ion (large data centers)

Yes, I am prou[illegible] accomplishments,
but moder[illegible]ms are far
more complex [illegible]r-centric

My job is to provide reliable infrastructure

Questions?

Photo NASA Public Domain April 7, 2021