Paul E. McKenney, Meta Platforms Kernel Team

Multicore World, February 16, 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 ~~Summer~~ 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

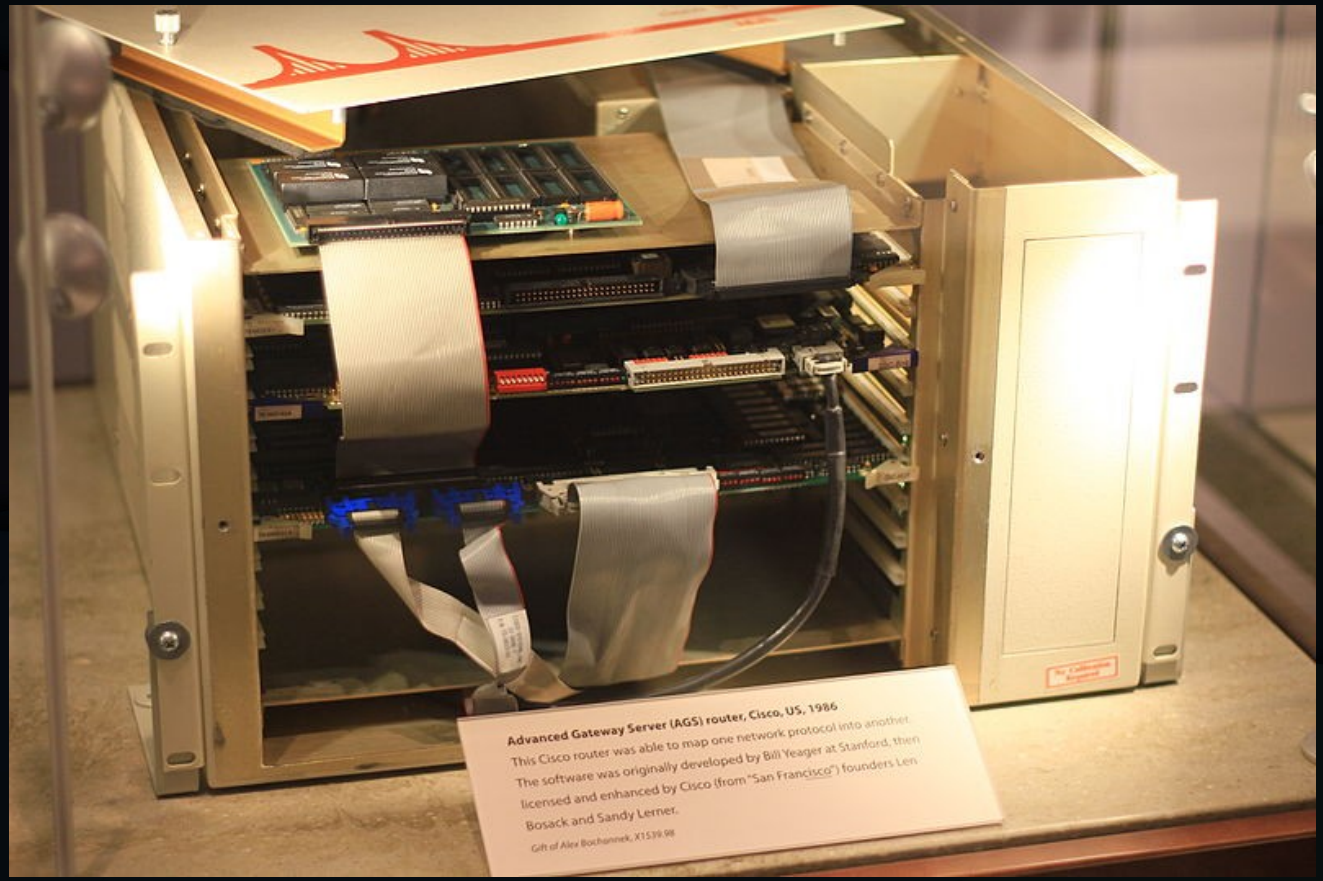
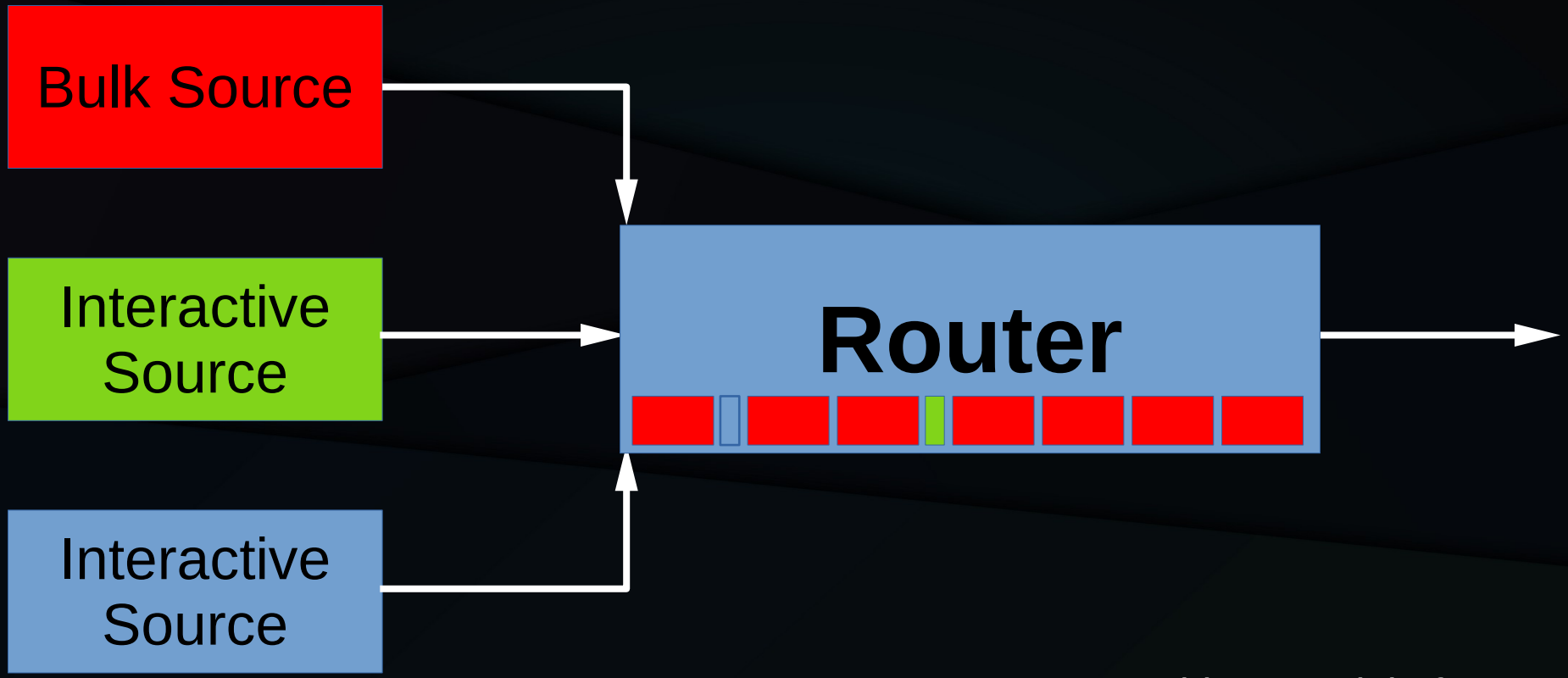


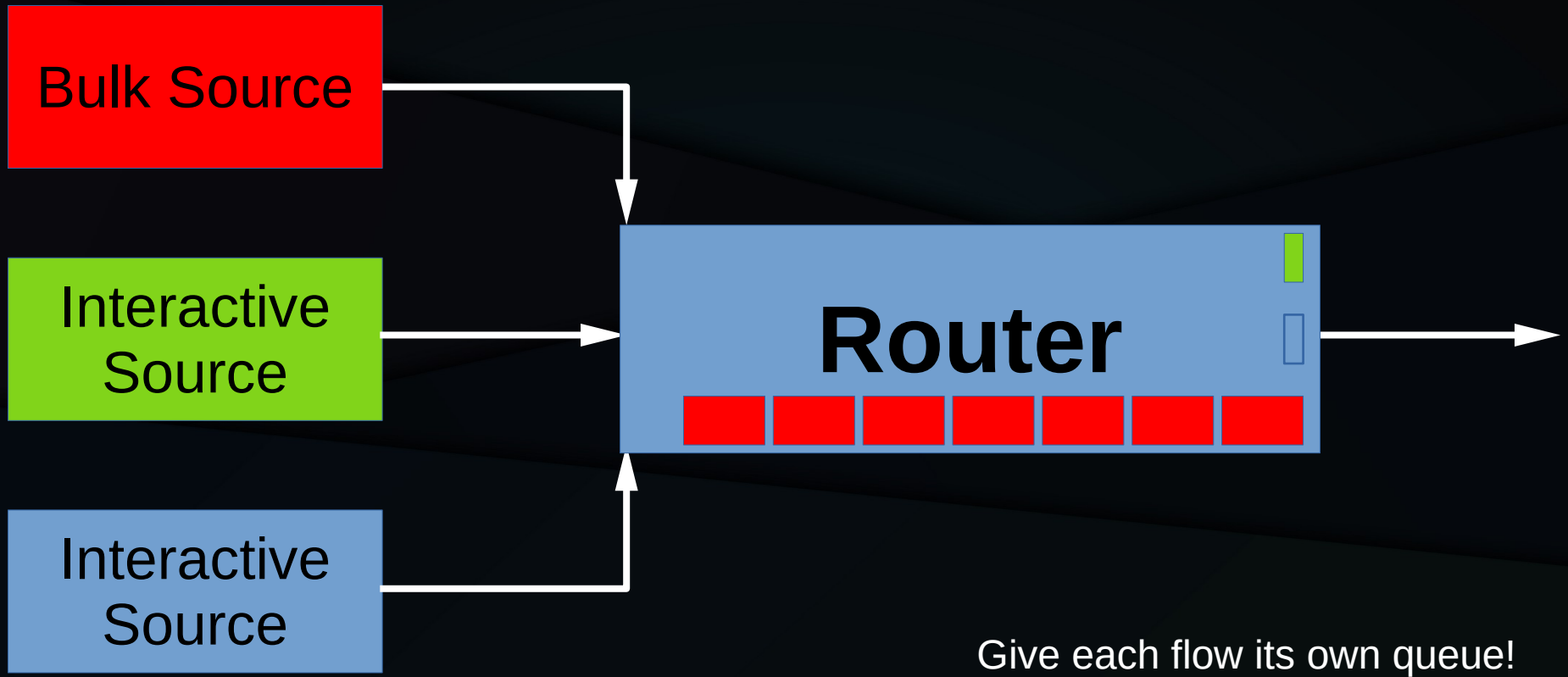
Photo: Michael Hicks, Creative Commons Attribution 2.0 Generic

1990: Queueing Problem



1 Megabit network is *fast*.

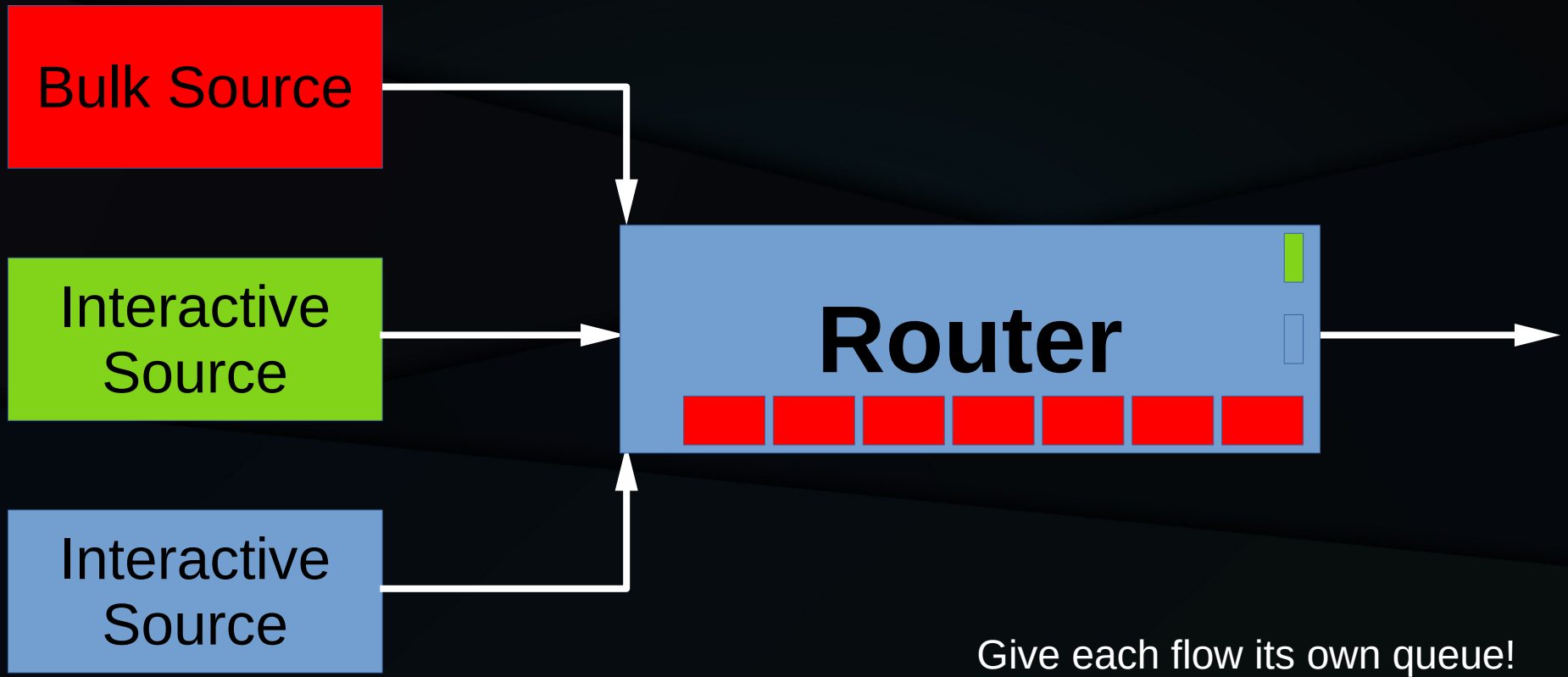
1990: Fair Queueing



Give each flow its own queue!

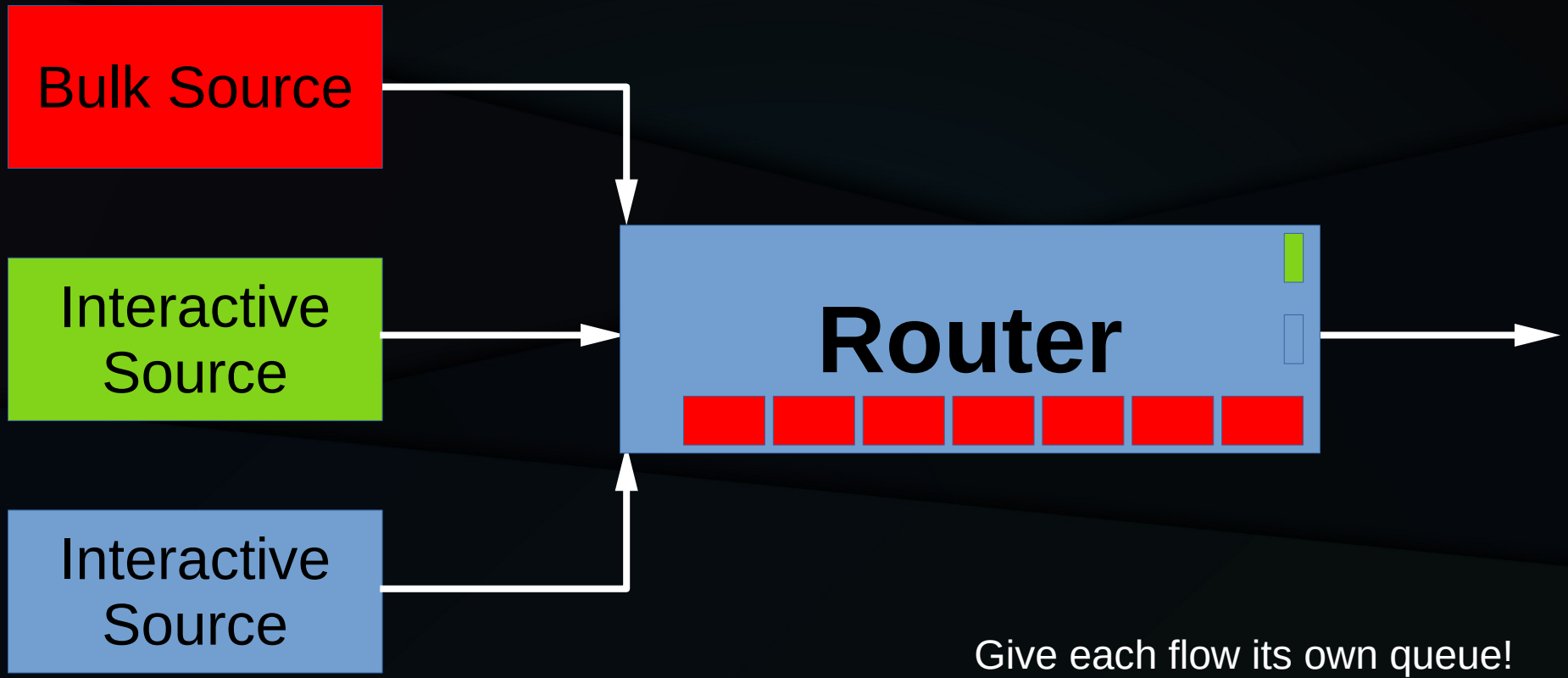
Yeah, you and how many 10MHz CPUs???

1990: Fair Queueing



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

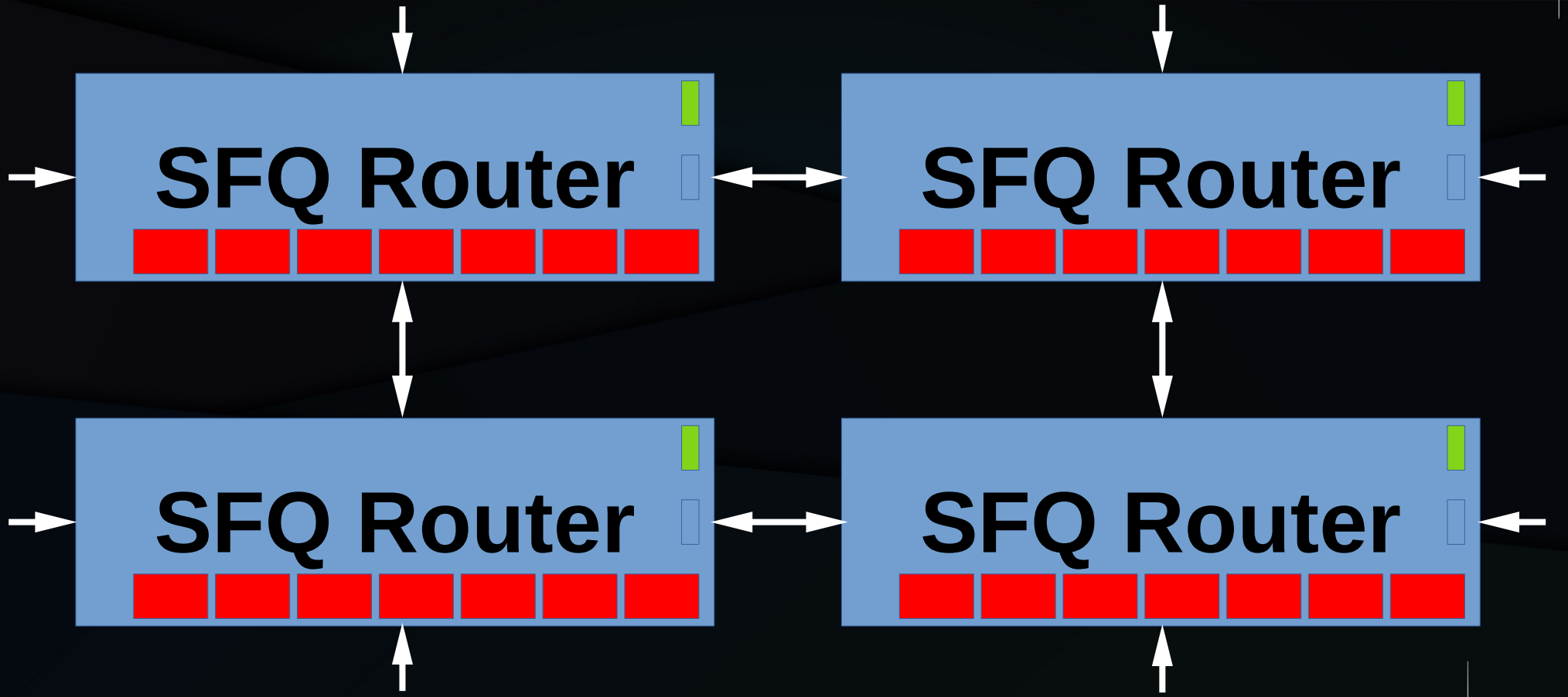
1990: Stochastic Fair Queueing: Hash



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

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

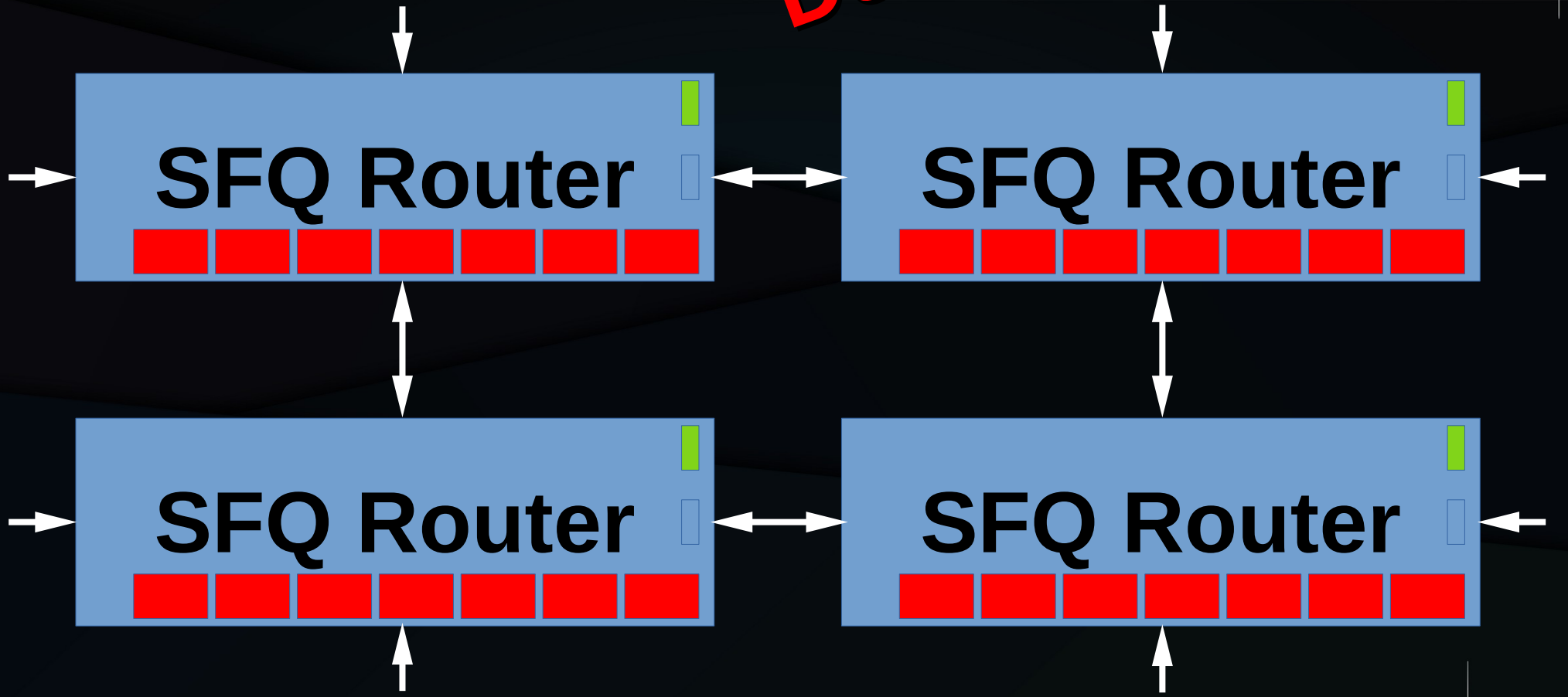
1990: Paul's Internet Vision



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

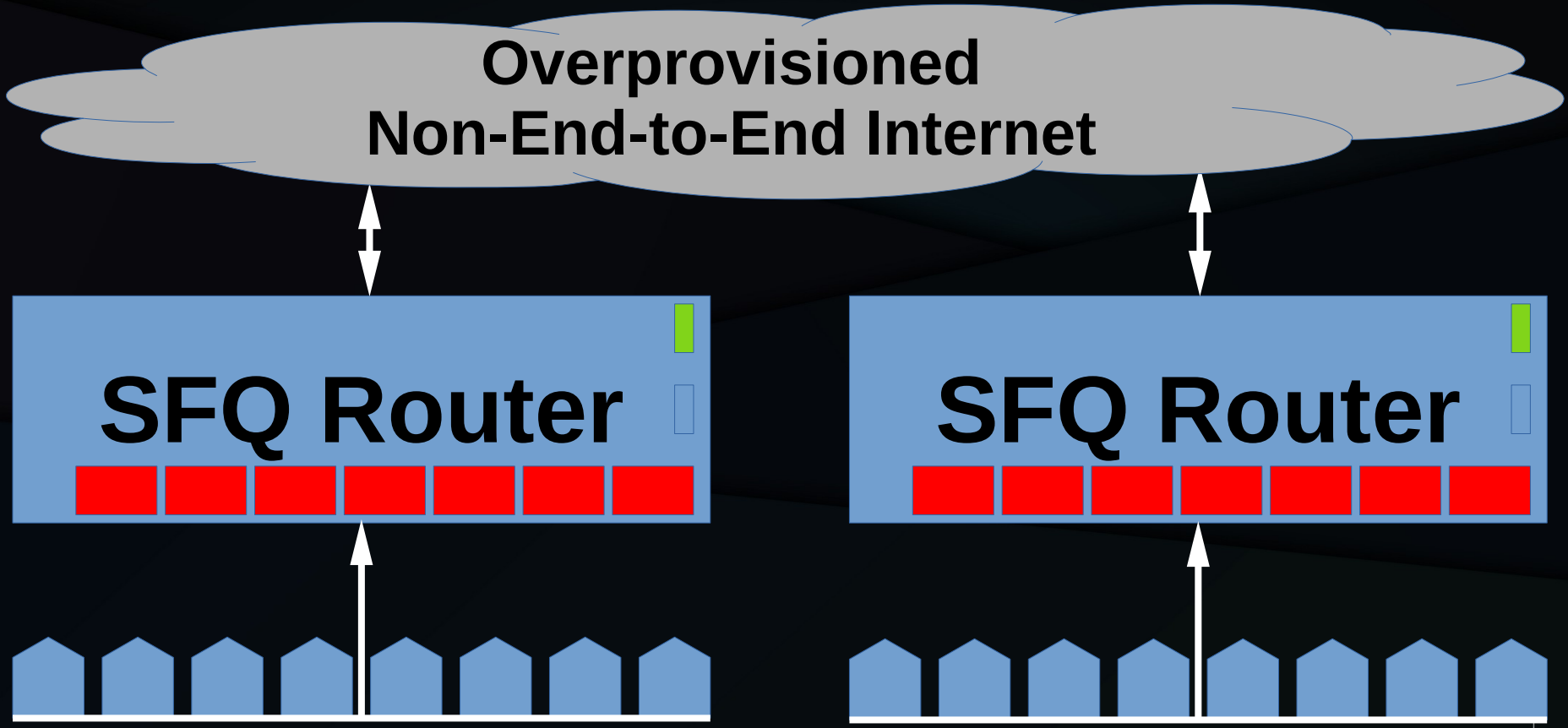
1990: Paul's Internet Vision

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!!!
  - Cisco and Linux-kernel engineers fixed this
- Used heavily until about 2015
  - 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 fairness
  - By sheer dumb luck, my algorithm implemented both
- Research-quality code: Get paper out!!!
  - Engineers at Cisco and Linux kernel fixed this
- Used heavily until about 2015
  - FQ-CODEL and CAKE now address bufferbloat
  - Dave Taht, Eric Dumazet, Toke Høiland-Jørgensen, ...

Bad idea badly implemented

1990 SFQ: What Went Wrong?

- Solved wrong problem: End-to-end fairnes
  - Correct problem: Hop-by-hop & endpoi
  - By sheer dumb luck, my algorithm
- Research-quality code: Ge t!!!
  - Engineers at Cisco a fixed this
- Used heavily u
  - FQ-COD low address bufferbloat
  - Dav mazet, Toke Høiland-Jørgensen, ...

Bad idea badly implemented, resuscitated by dumb luck

1990 SFQ: What Went Wrong?

- Solved w[illegible] problem: End-to-end fairnes[illegible]
  - Correct pr[illegible] Hop-by-hop & endpoint
  - By sheer dumb [illegible] algorithm[illegible]
- Research-quality [illegible]t!!!
  - Engineers at Cisco [illegible] fixed this
- Used heavily u[illegible]
  - FQ-COD[illegible] low addre[illegible]uffer
  - Dave [illegible]mazet, Toke Høiland-Jørgens[illegible]...

Premature abstraction is the root of all evil

Bad idea but implemented, resuscitated by dumb luck

1990 SFQ: What Instead?

- Solved wrong problem: End-to-end fairness
  - Correct problem: Hop-by-hop & endpoint fai
  - By sheer dumb luck, my algorithm ha
- Research-quality code: Get th out!!!
  - Engineers at Cisco and i ernel fixed this
- Used heavily until 015
  - FQ-CODEL 2 now address bufferbloat
  - Dave Tah Dumazet, Toke Høiland-Jørgensen, ...

Live among your users!!!

1980s: Eight-Bit CRM

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: Tha[illegible] me *before* bankruptcy

Bad idea well implemented

1980s: Eight-Bit CRM

- CRM application built to spec und[illegible]ntract
- The company loved it!
- Their prospective cus[illegible]so much
- Dumb luck: Tha[illegible]*before* bankruptcy

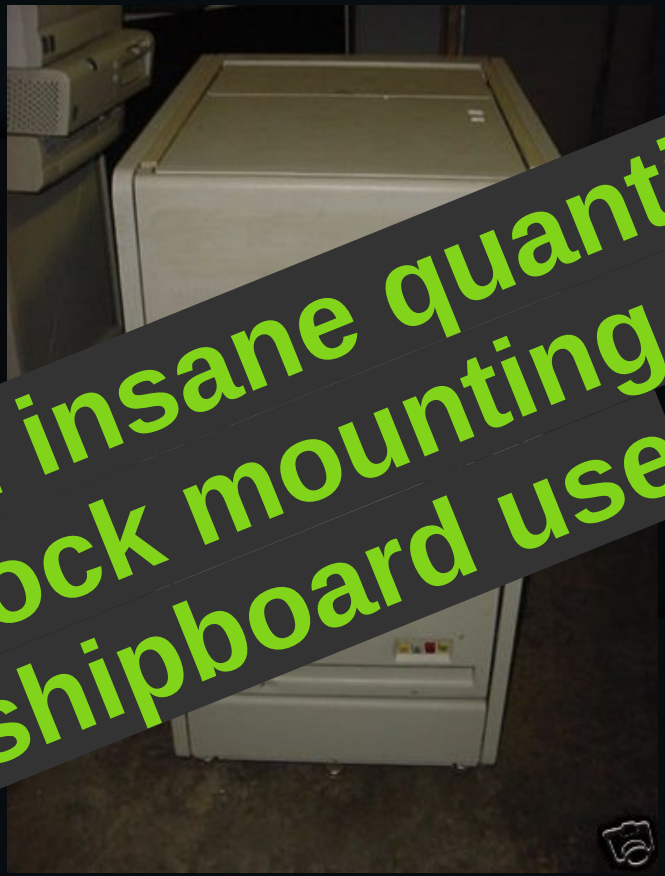
Bad idea well implemented, but hey, I got paid???

1980s: Eight-Bit CRM: What Instead?

Time and Grade: Experience

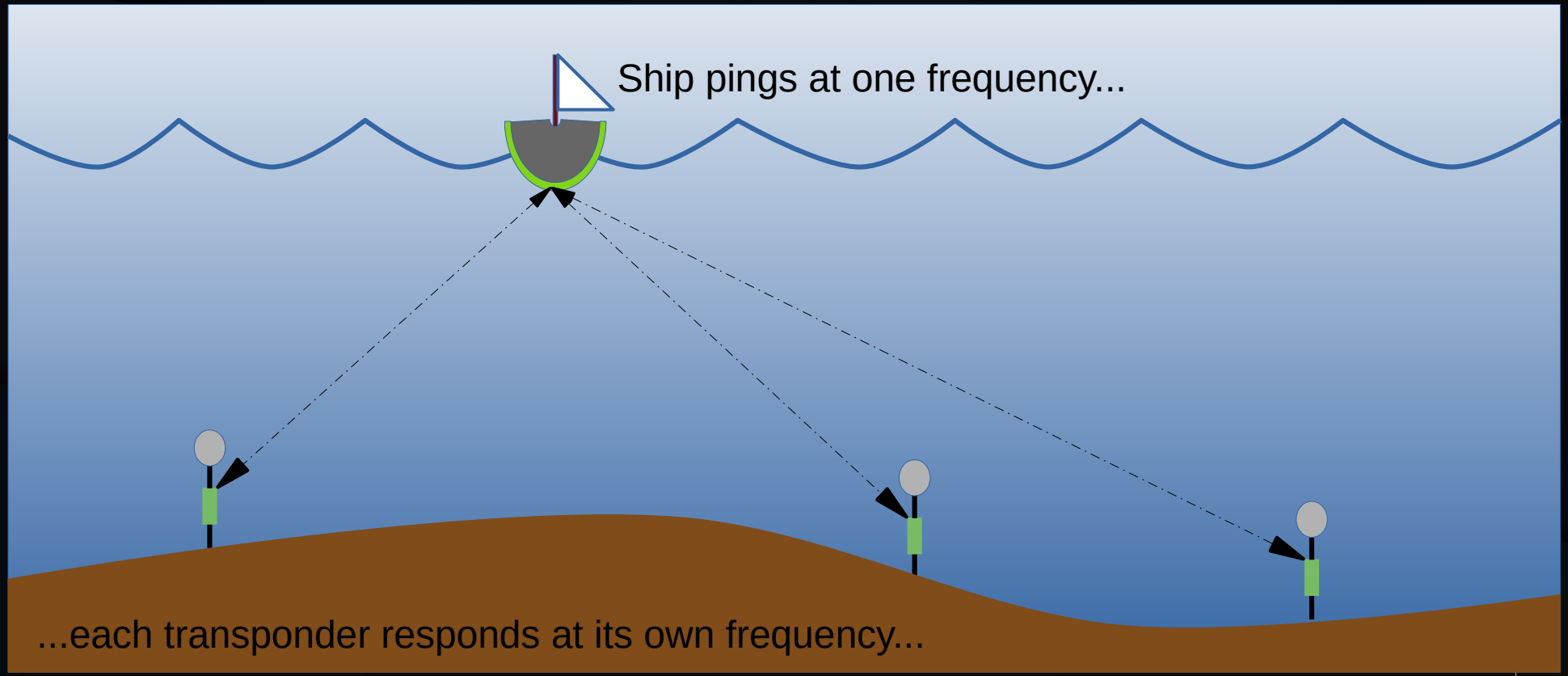
1980s: Acoustic Navigation

1980s: Acoustic Navigation



But with insane quantities of shock mounting for shipboard use

1980s: Acoustic Navigation (Pre-GPS)



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

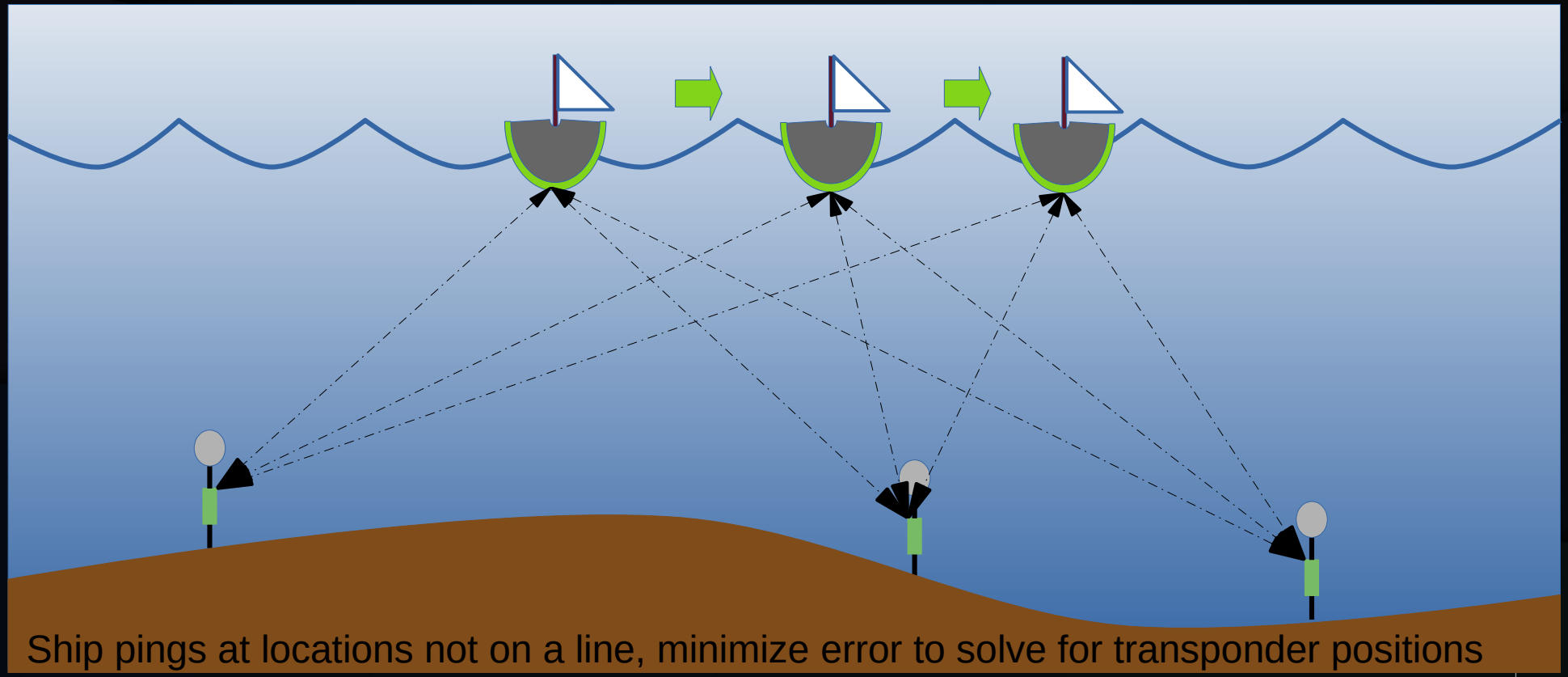
Acoustic Navigation Complications

- 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

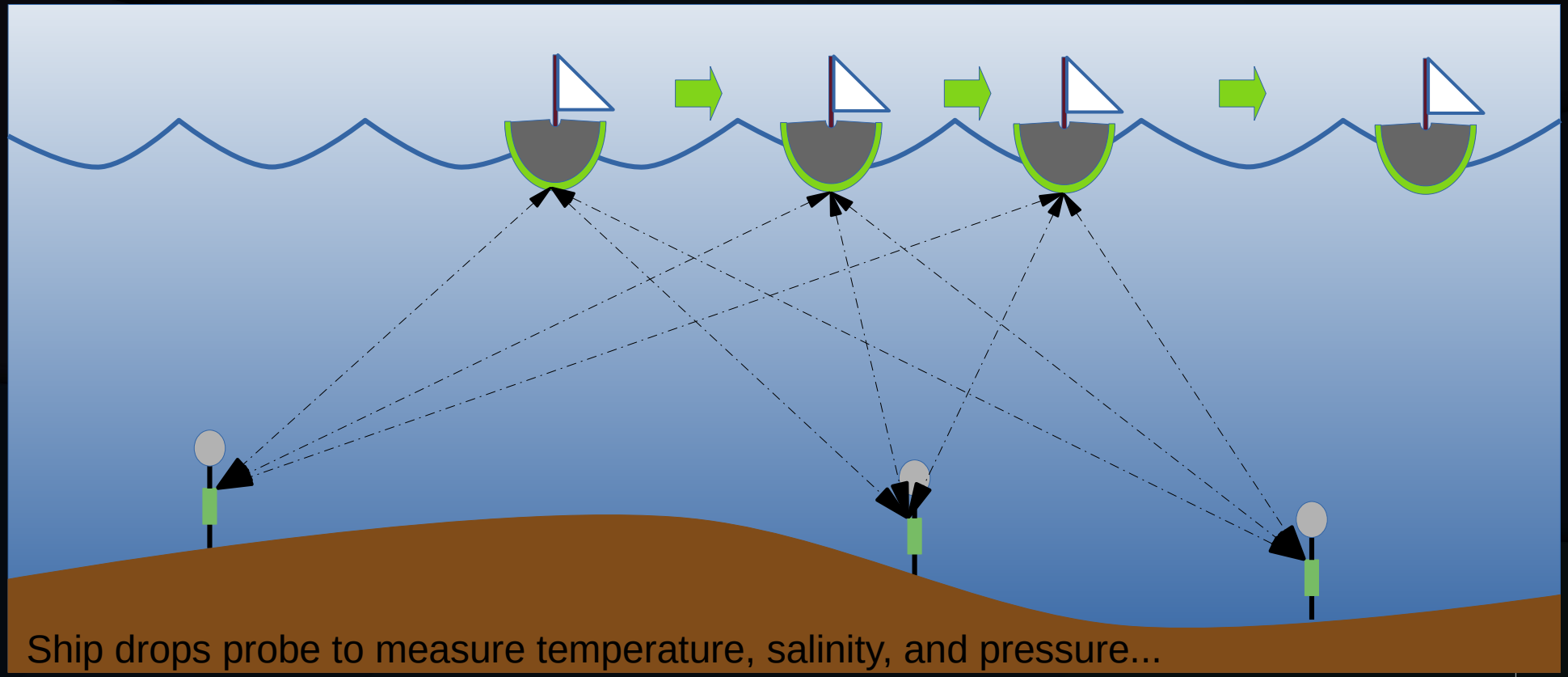
Acoustic Navigation Calibration (1/2)



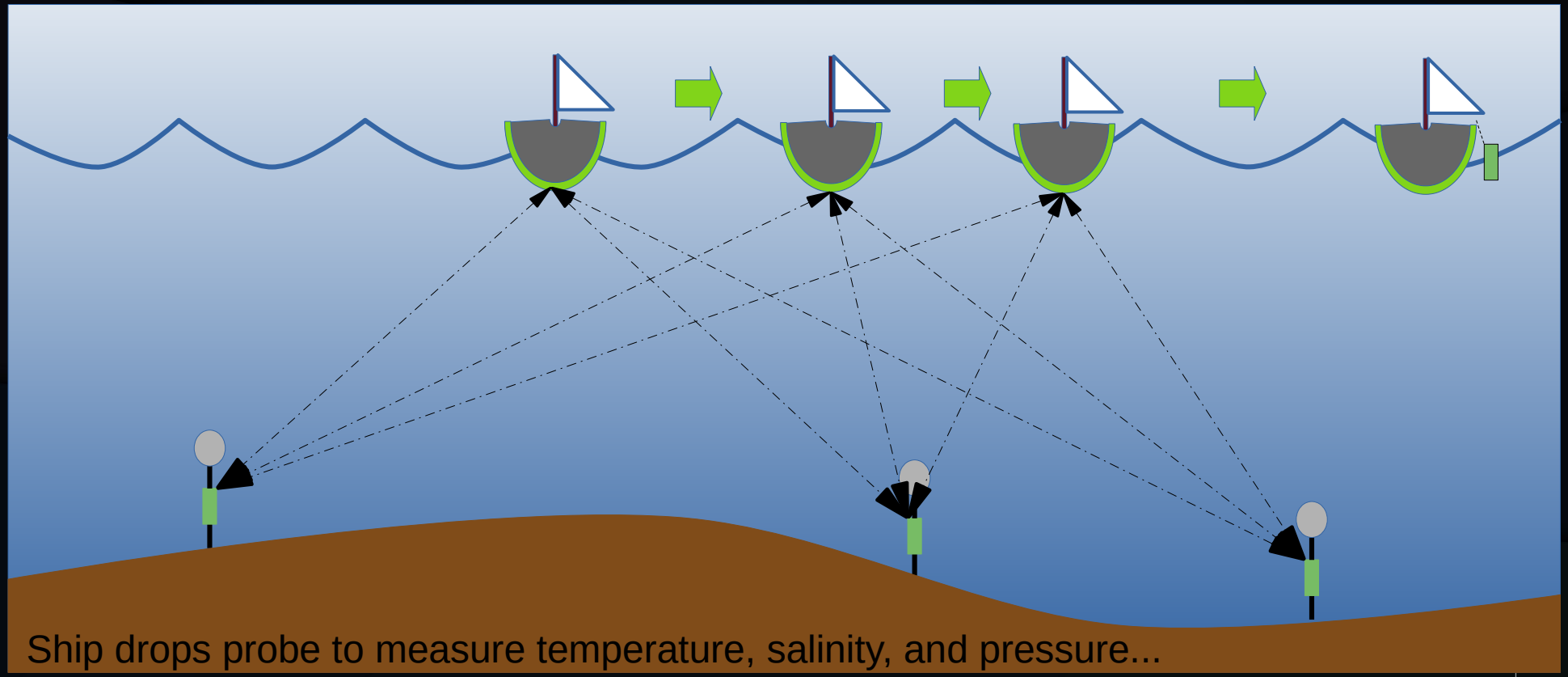
Acoustic Navigation Complications

- 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

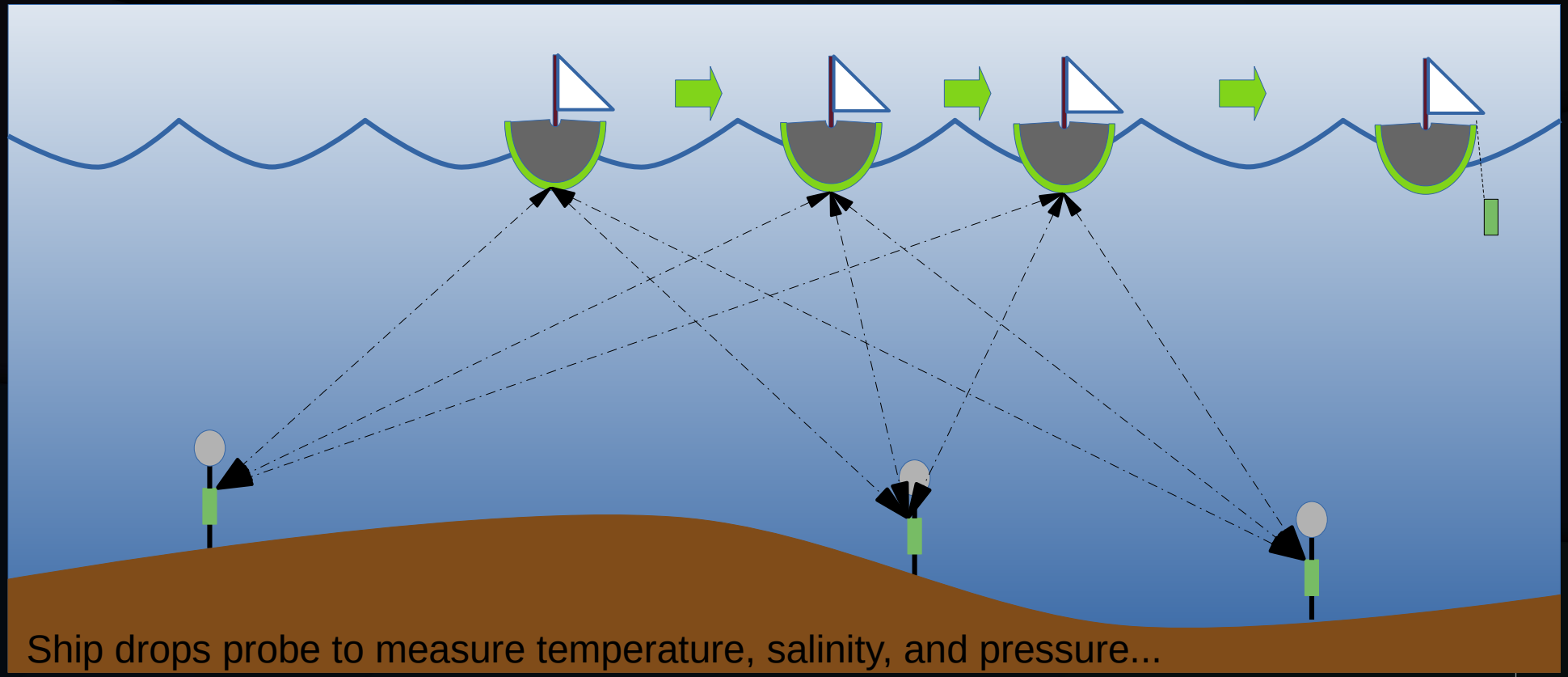
Acoustic Navigation Calibration (2/2)



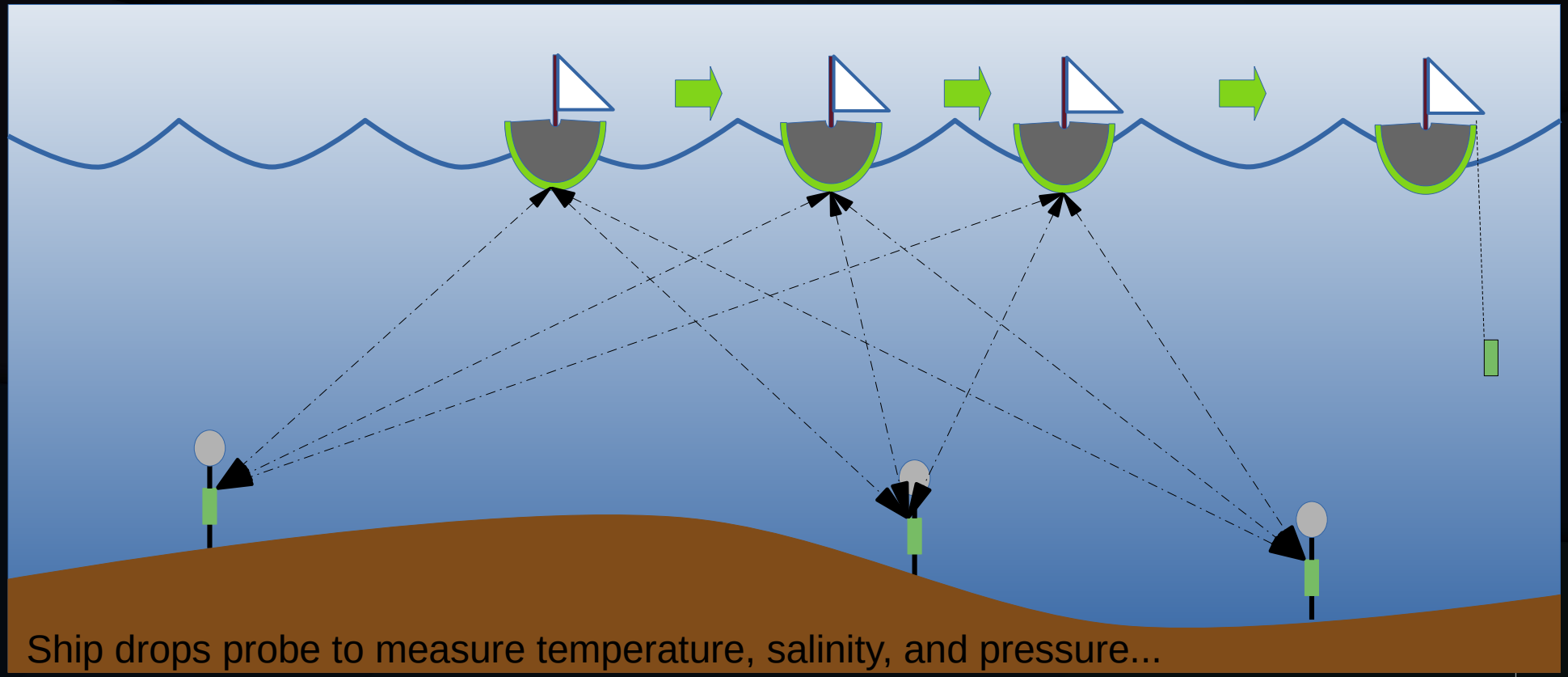
Acoustic Navigation Calibration (2/2)



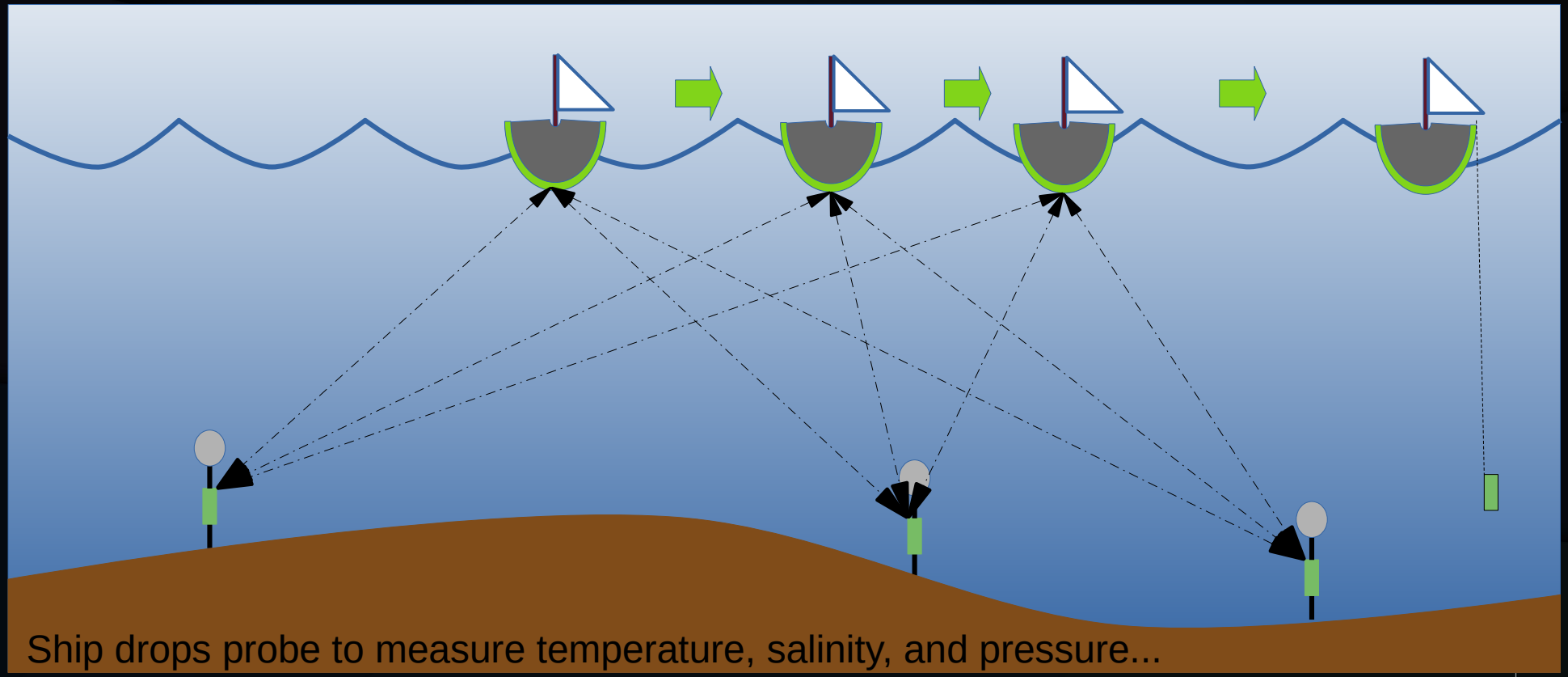
Acoustic Navigation Calibration (2/2)



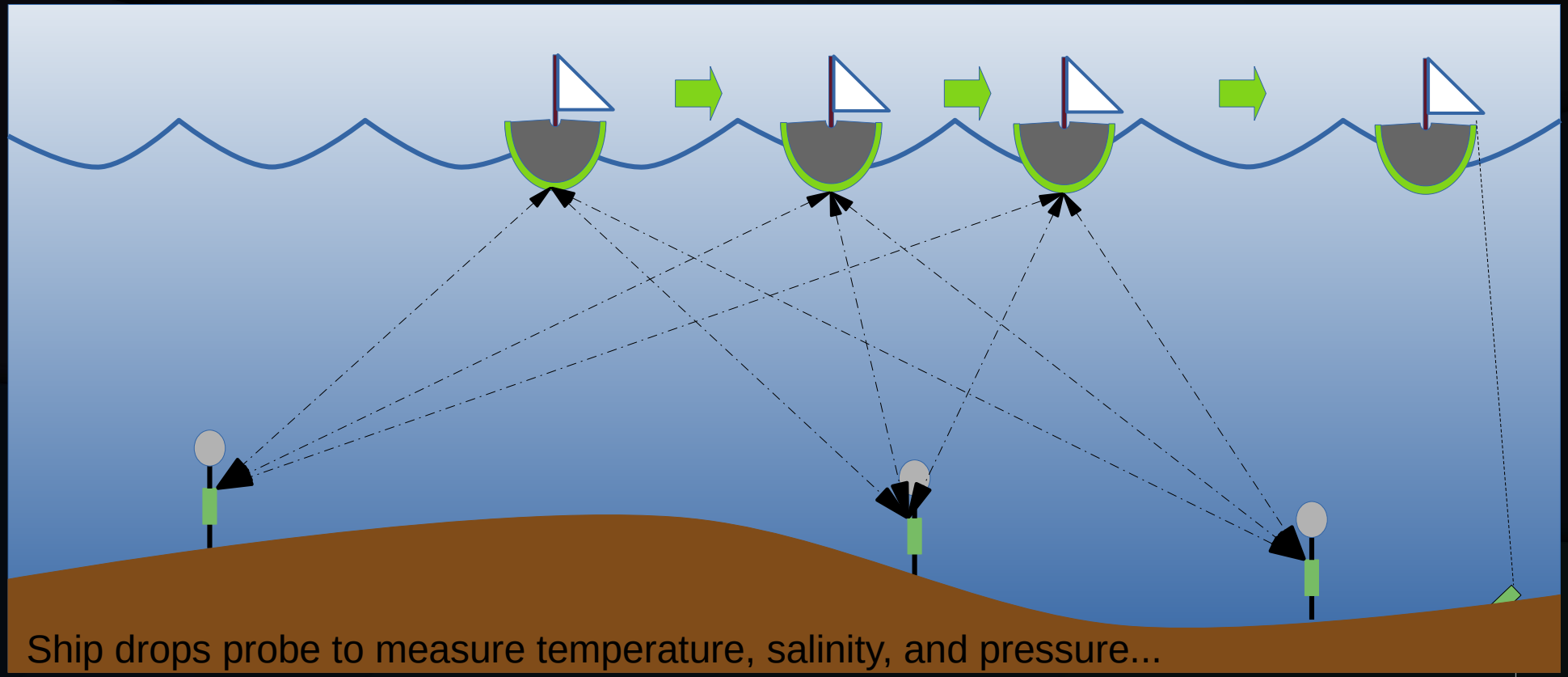
Acoustic Navigation Calibration (2/2)



Acoustic Navigation Calibration (2/2)



Acoustic Navigation Calibration (2/2)



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

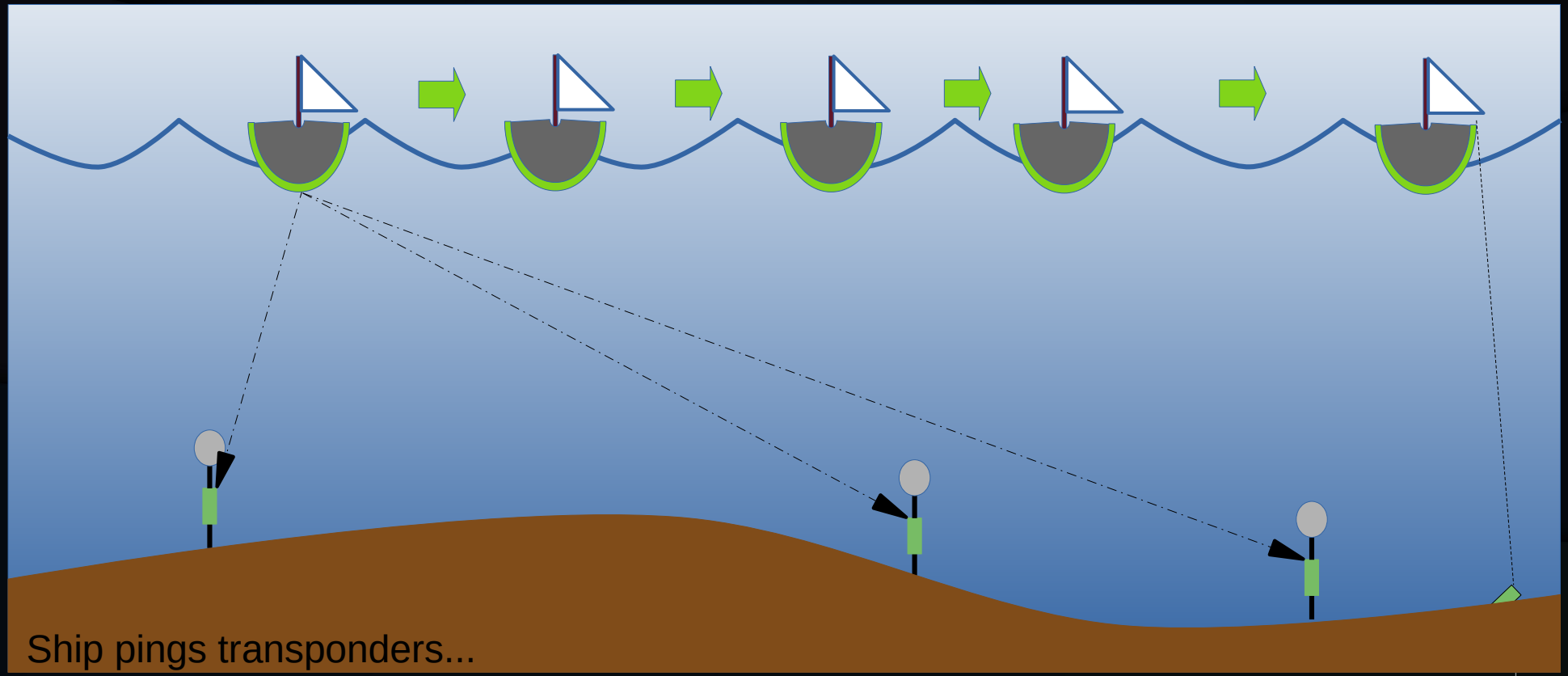
Acoustic Navigation Complications

- 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

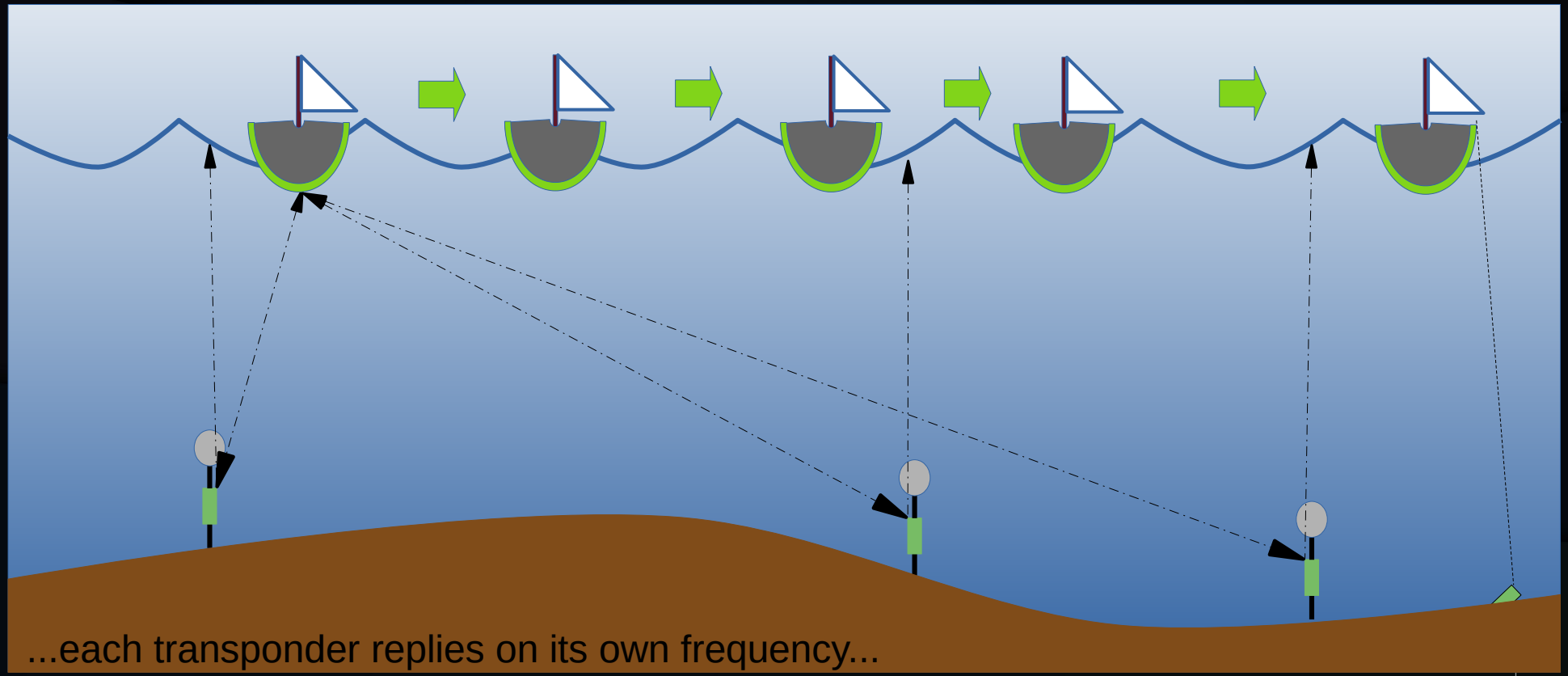
Acoustic Navigation Complications

- 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

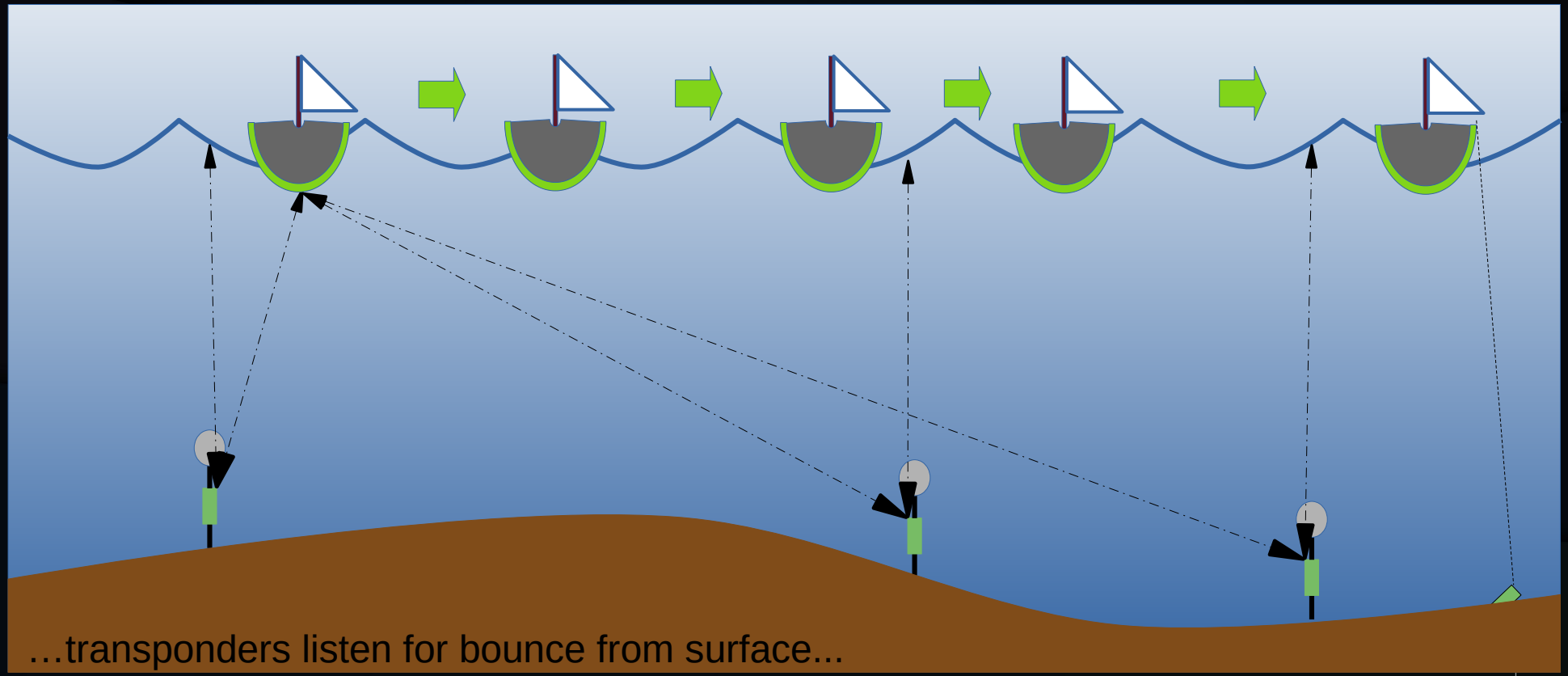
Acoustic Navigation: Measure Depth



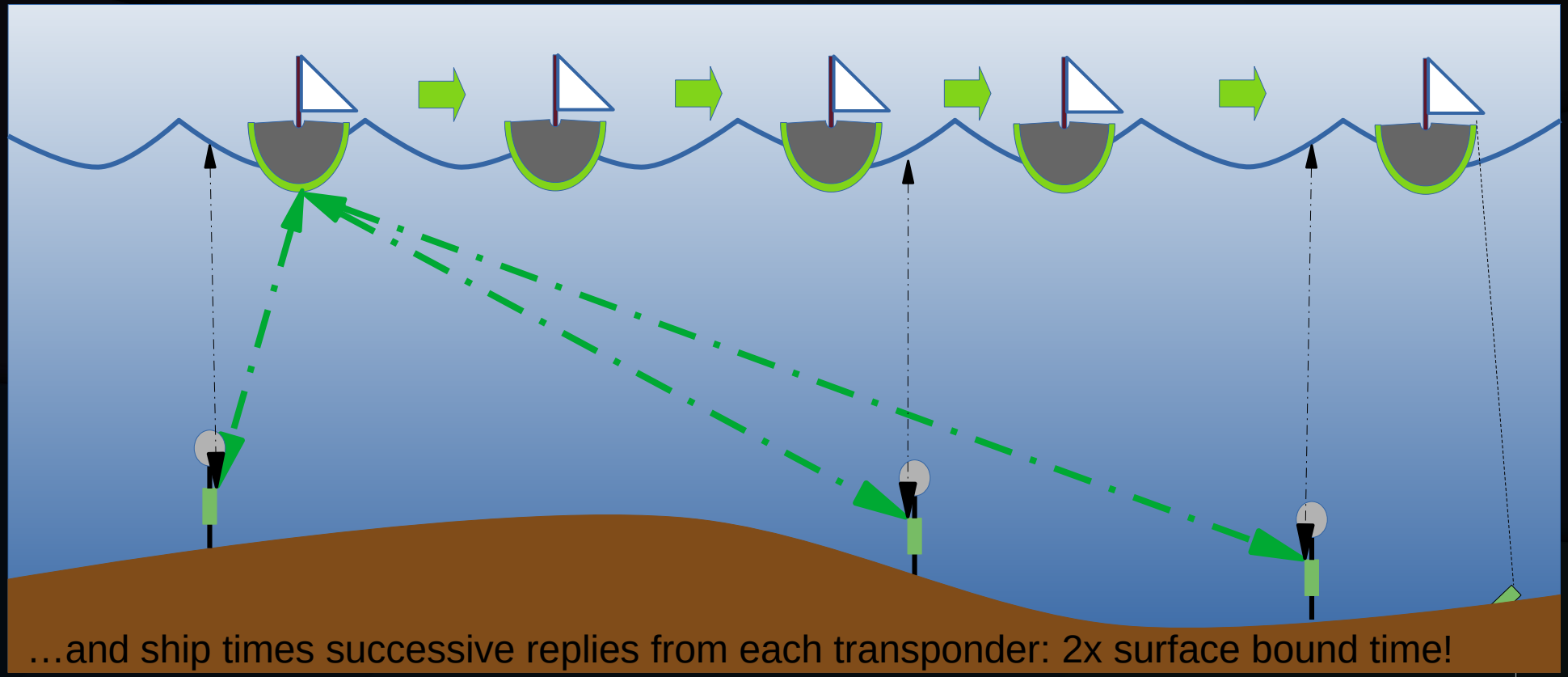
Acoustic Navigation: Measure Depth



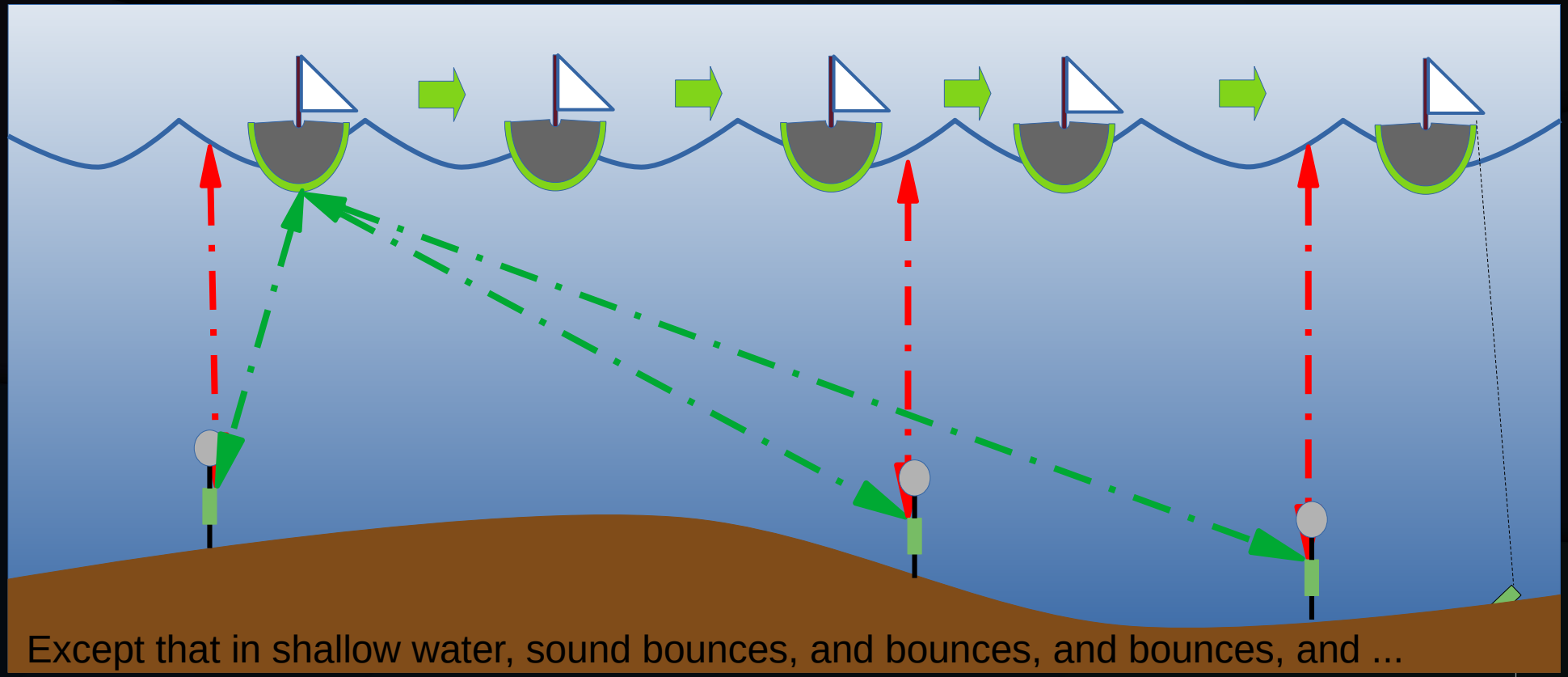
Acoustic Navigation: Measure Depth



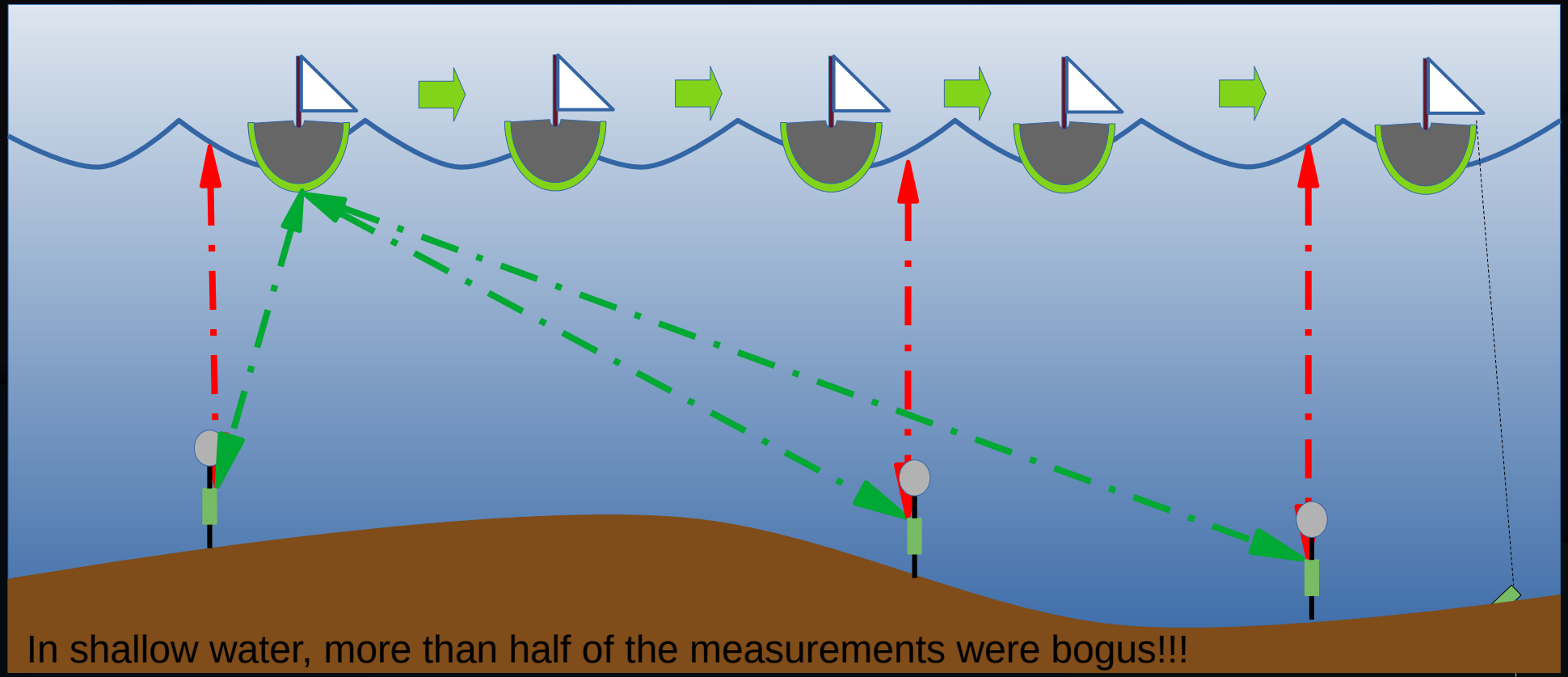
Acoustic Navigation: Measure Depth



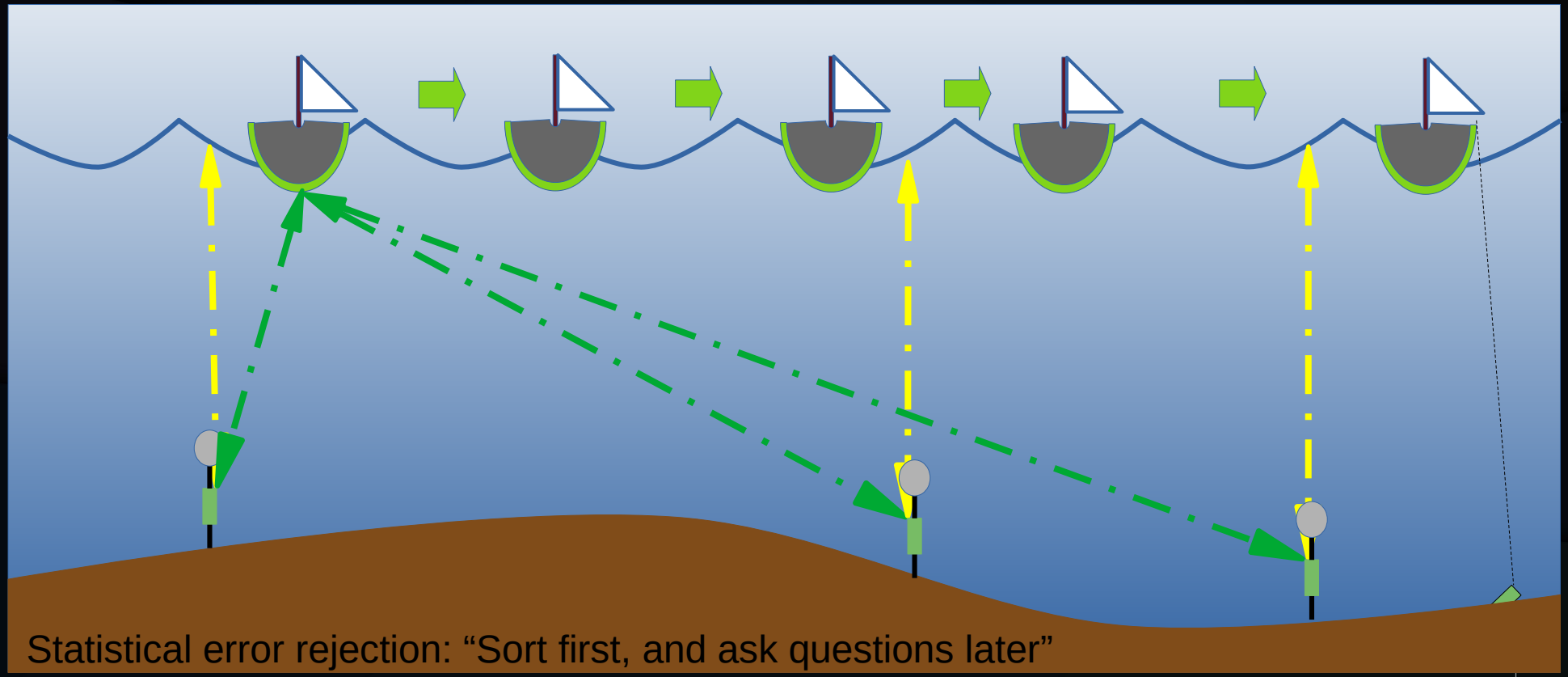
Acoustic Navigation: Measure Depth



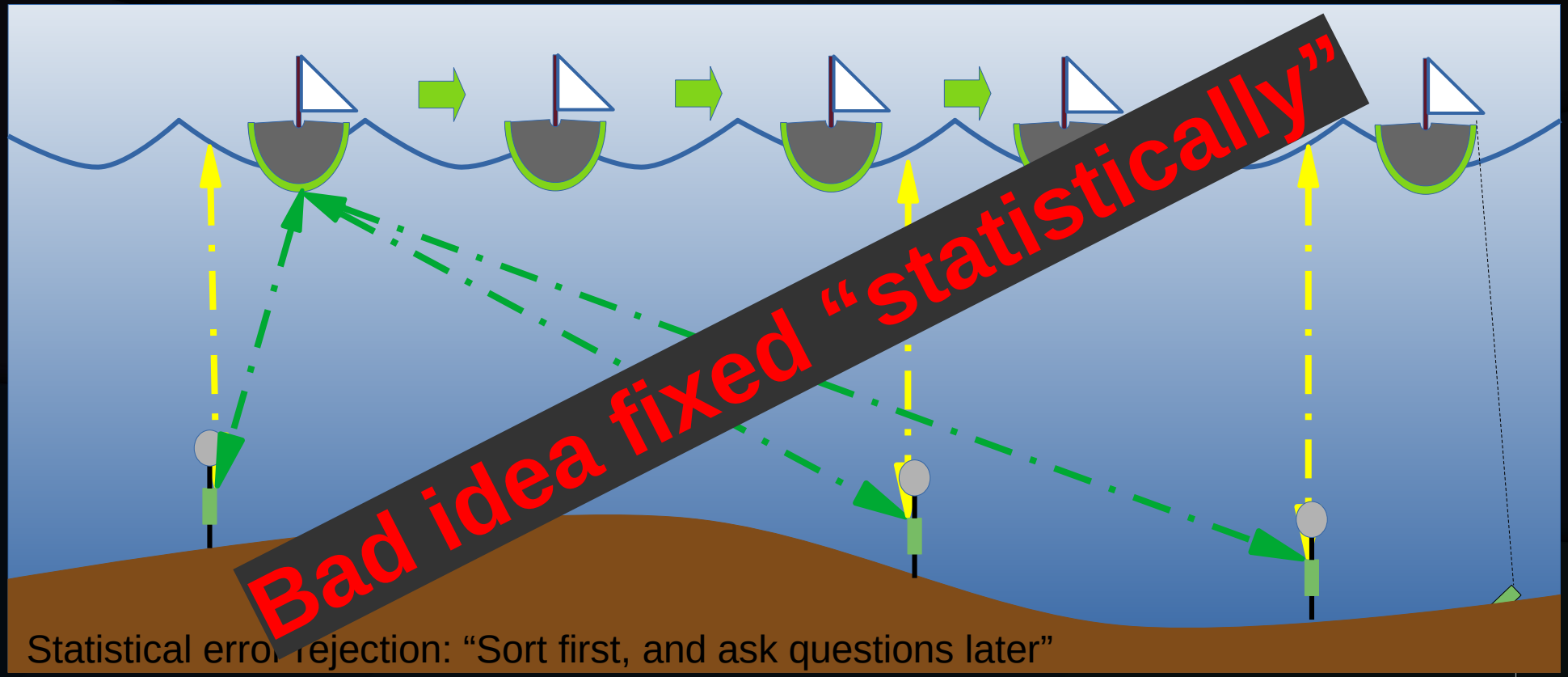
Acoustic Navigation: Measure Depth



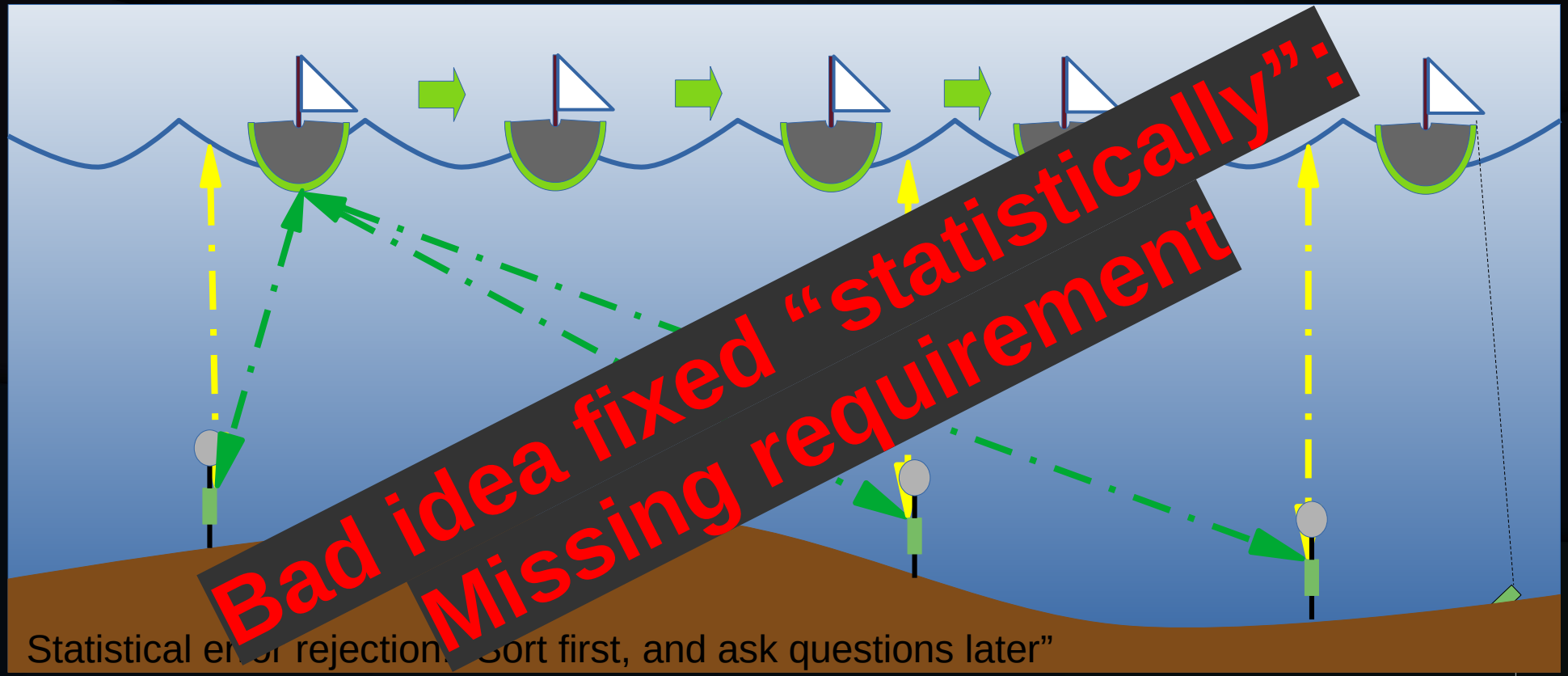
Acoustic Navigation: Measure Depth



Acoustic Navigation: Measure Depth



Acoustic Navigation: Measure Depth



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 ha ng a program in the first place. So if you're as c n be when you write it, how will you ever d ." *Kernighan*, "The Elements of Pr 2nd Edition, Chapter 2.
- While programming issful ignorance of important requ 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 ha… …ng a program i… …st place. So if you're as c… …n be when you wr… …will you ever d… …*Kernighan*, "The E… …f Pr… …2nd Edition, Chapter 2.
- While programming … …sful ignorance of important requ… …equ… …make themselves known du…
- Whic… …se of Kernighan's observation.

And the file cabinet won

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

1970s: Student Housing System

Photo courtesy of Fundacio´n ICA: CDC 3300

1970s: Student Housing System

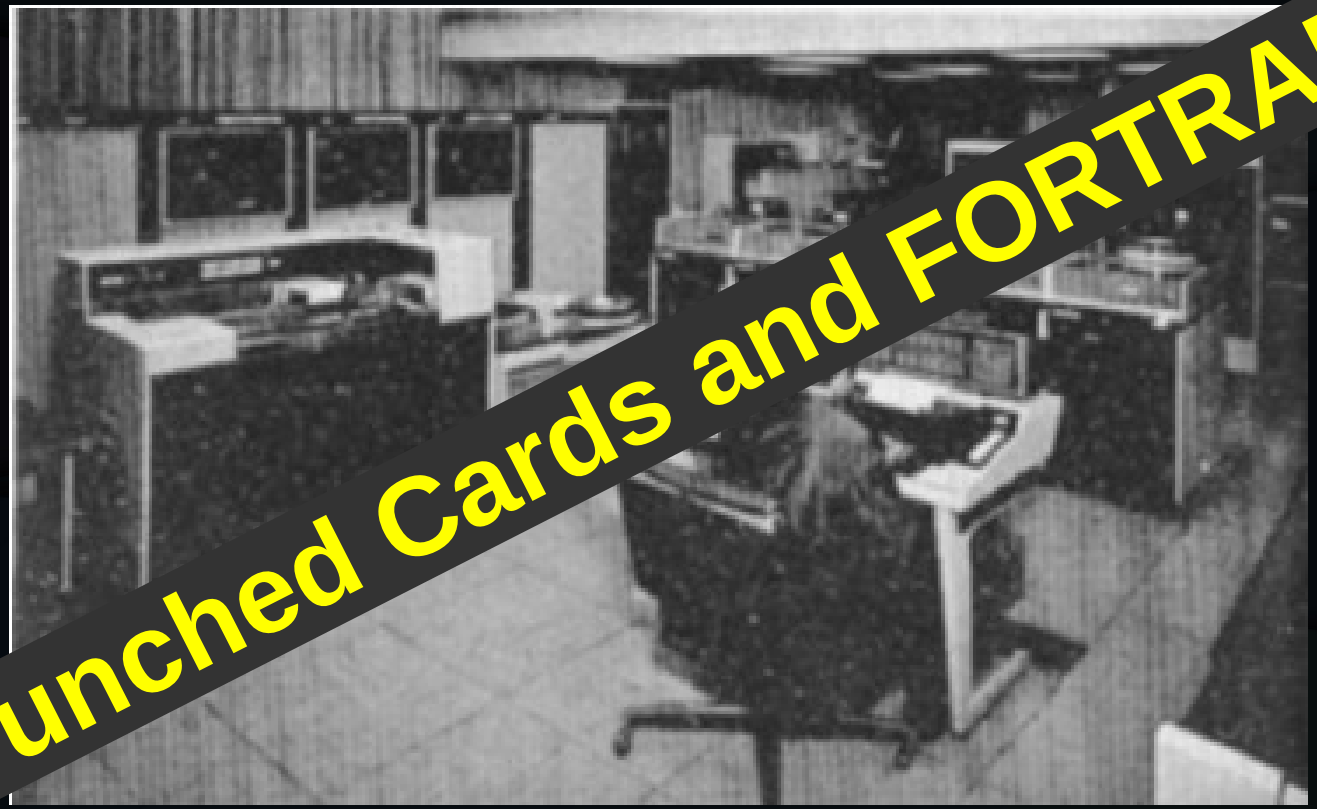


Photo courtesy of Fundacio´n ICA: CDC 3300

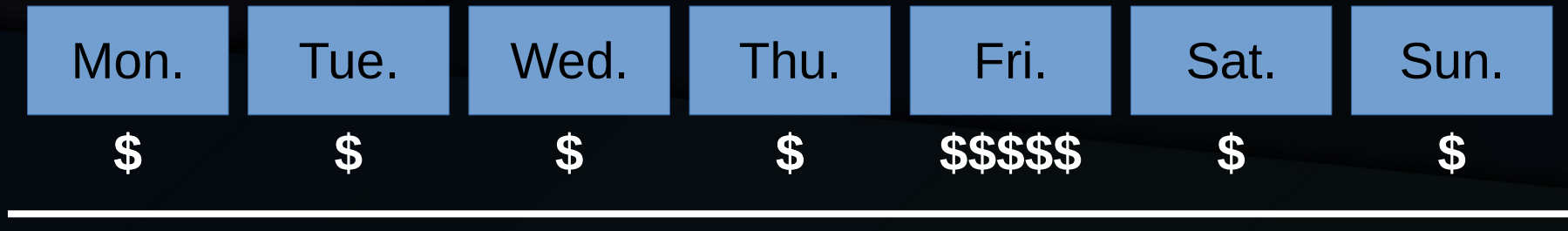
1970s: Student Housing System

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

1970s: Student Housing System

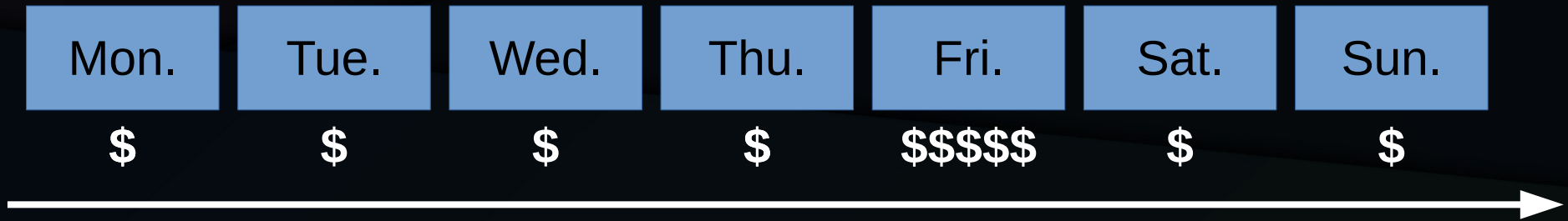
Photo courtesy Wikipedia user Bubba CC BY-SA 4.0: CDC Cyber 73 console (rest of computer fills room)

Student Housing Temporal Confusion



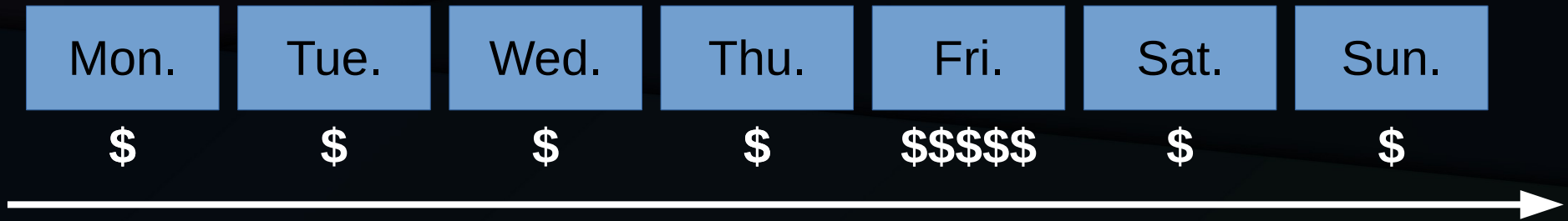
Student Housing Temporal Confusion

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



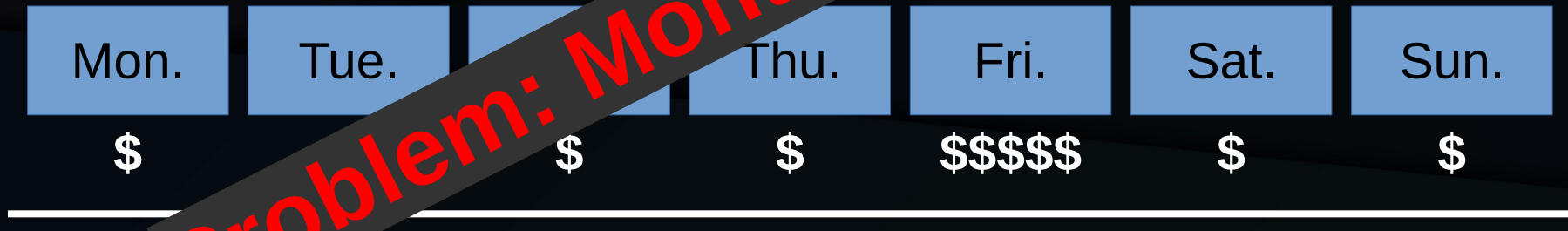
Student Housing Temporal Confusion

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.



Student Housing Temporal Confusion

Student start[illegible]day and was not [illegible]by the bill. My [illegible]had the usual [illegible]arthy suggestion for [illegible]ernative uses of the money.



Problem: Months vary in length

Student Housing Temporal Confusion

Student start… day and
was not… the bill.
My… 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

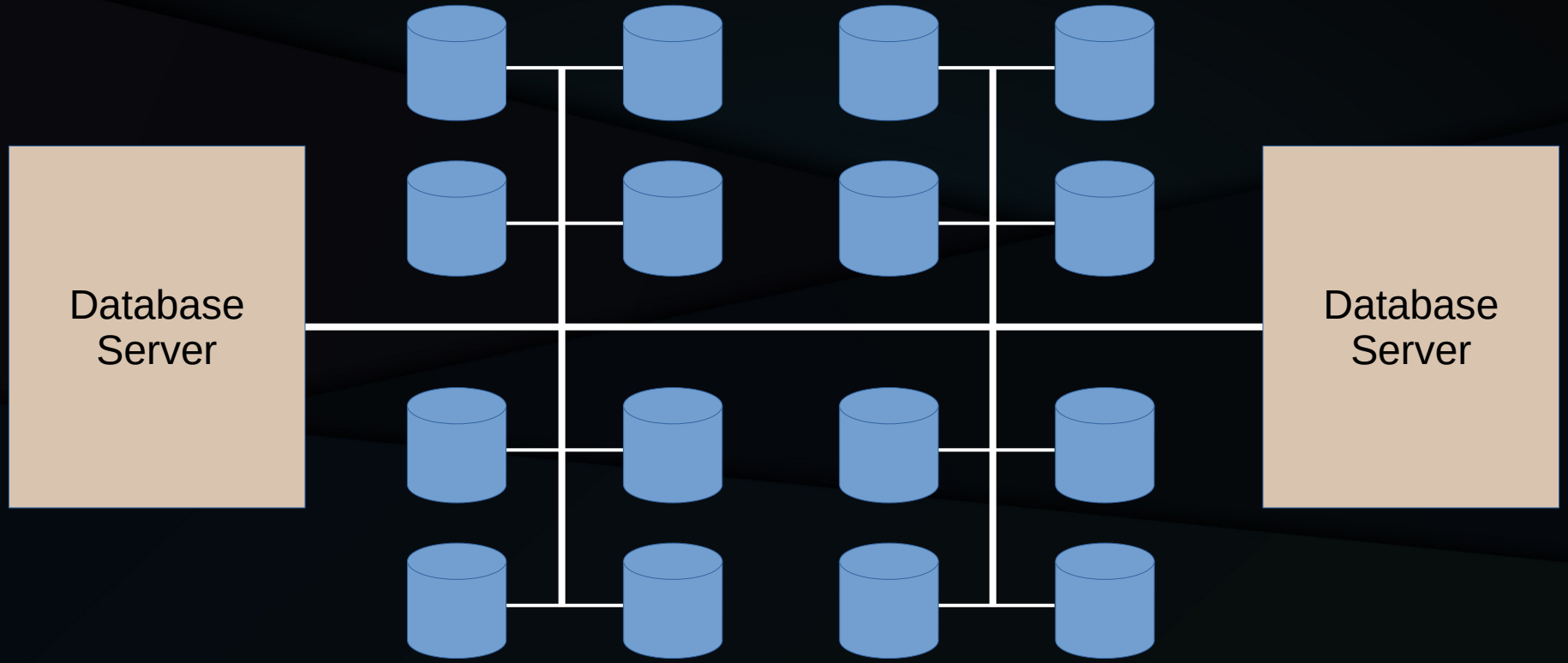
Student Housing Temporal Confusion

Student started o… day and
was not am… e bill.
My ma… the usual
19… suggestion for
…tive uses of the money.

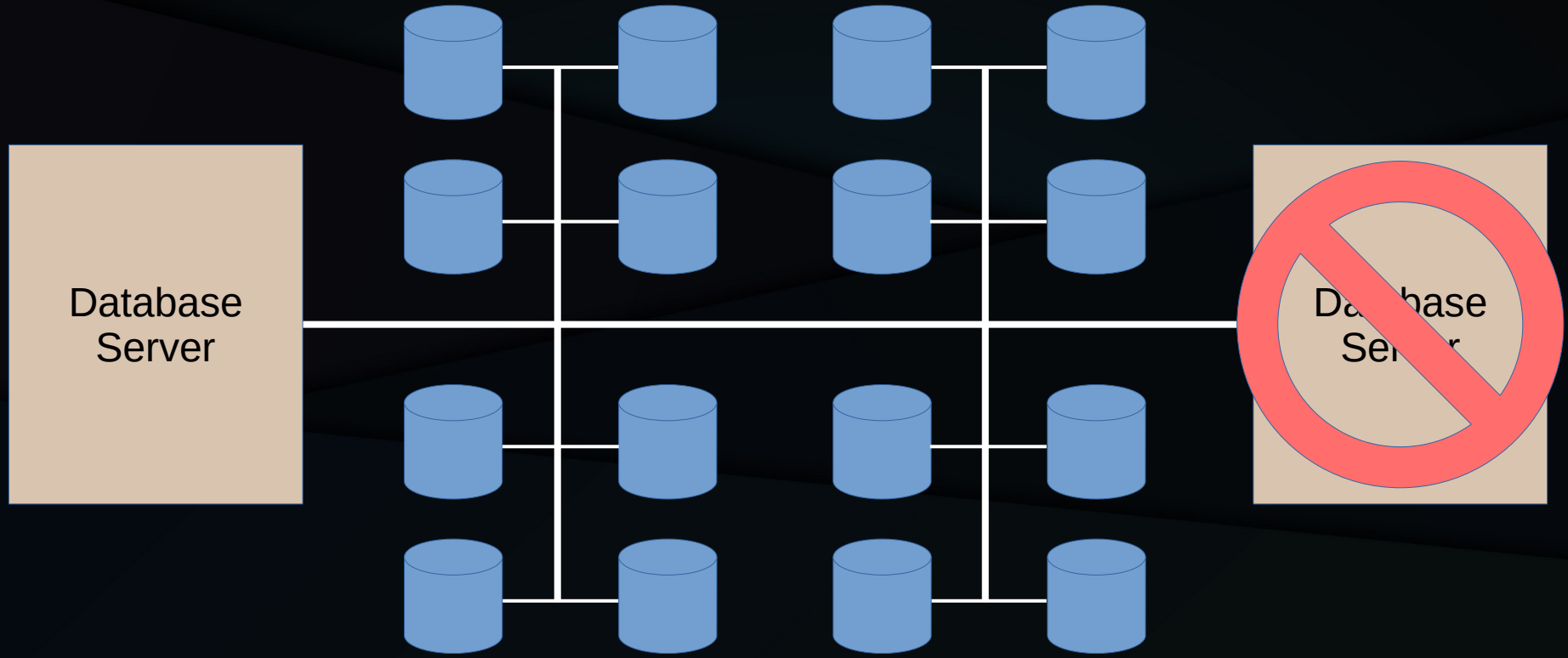
Good idea implemented poorly

1990s: Clustered Database Servers

Shared Disks For Availability Win!!!

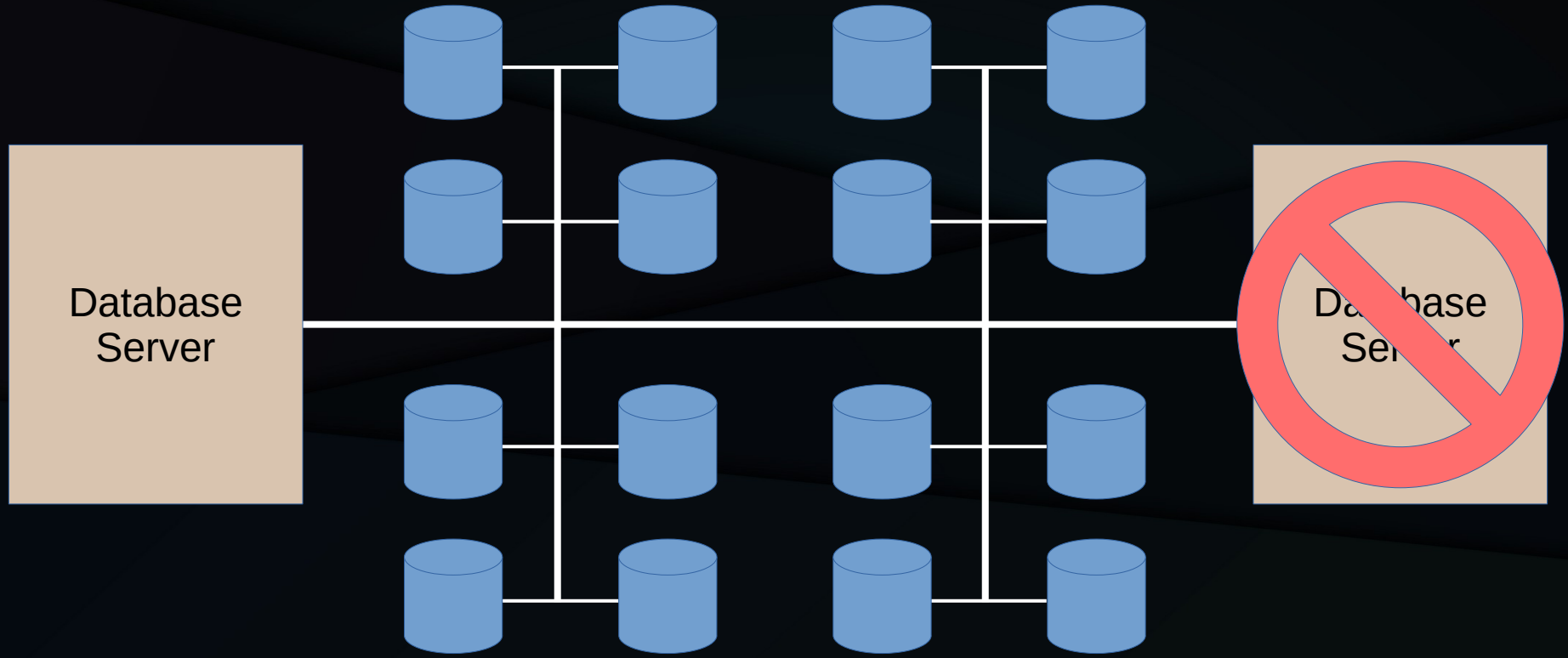


Shared Disks For Availability Win!!!



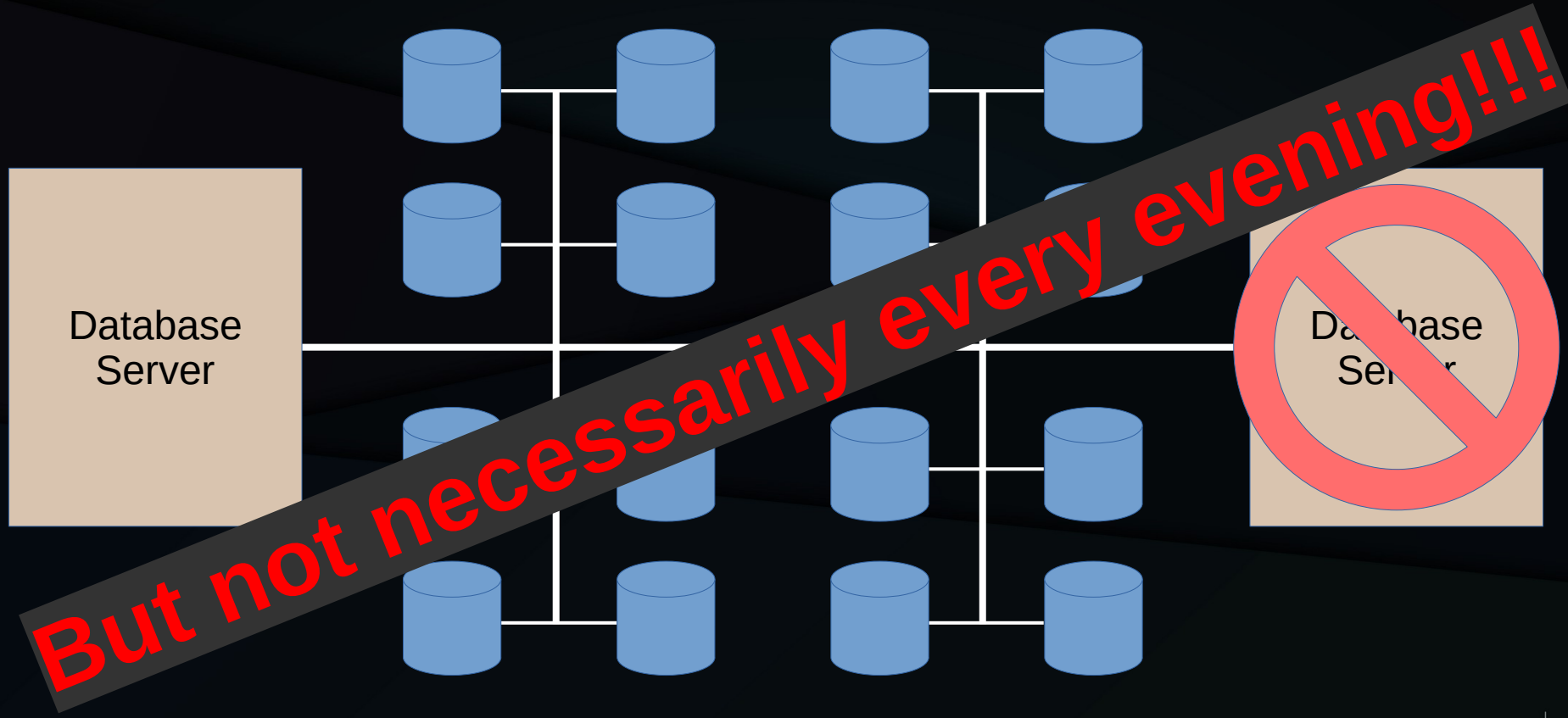
All data is still accessible!!!

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

Chaos-Monkey Challenges

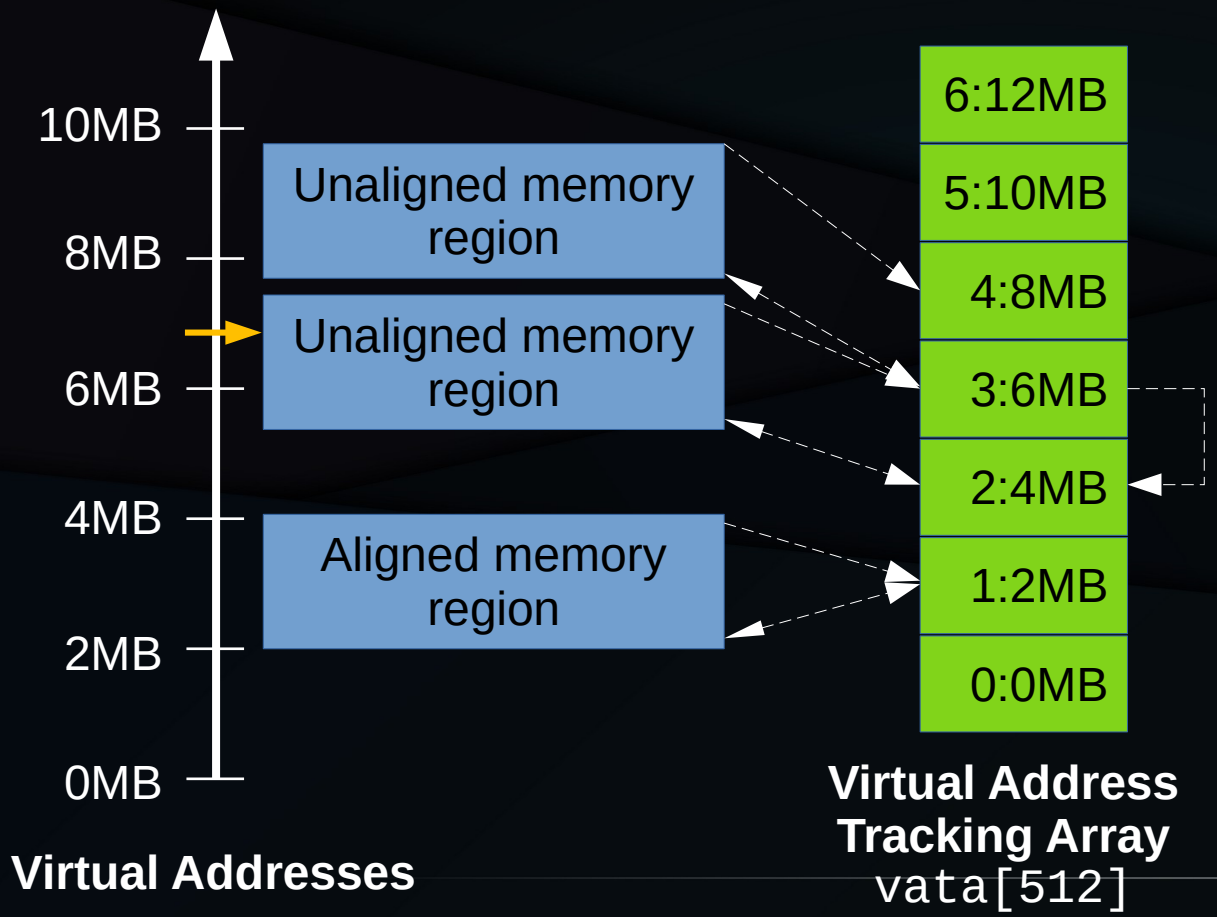
- Crash dump was a complete disast[illegible]
  - No hints for on-site debugging i[illegible]entation
- Unable to reproduce in [illegible]ab
- Eventually, four[illegible] case: 5-27-hour MTBF

Created rudimentary digital twin?

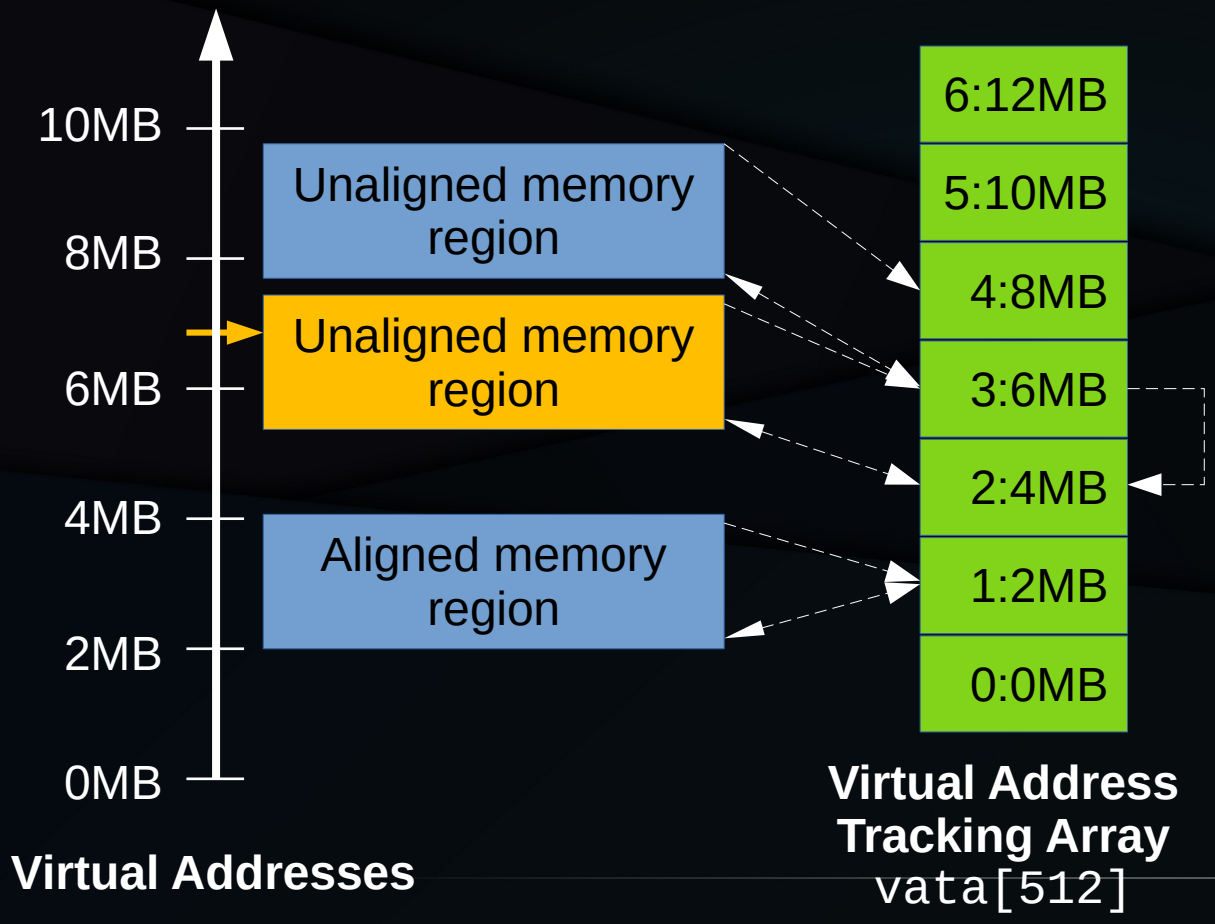
Chaos-Monkey Challenges

- 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

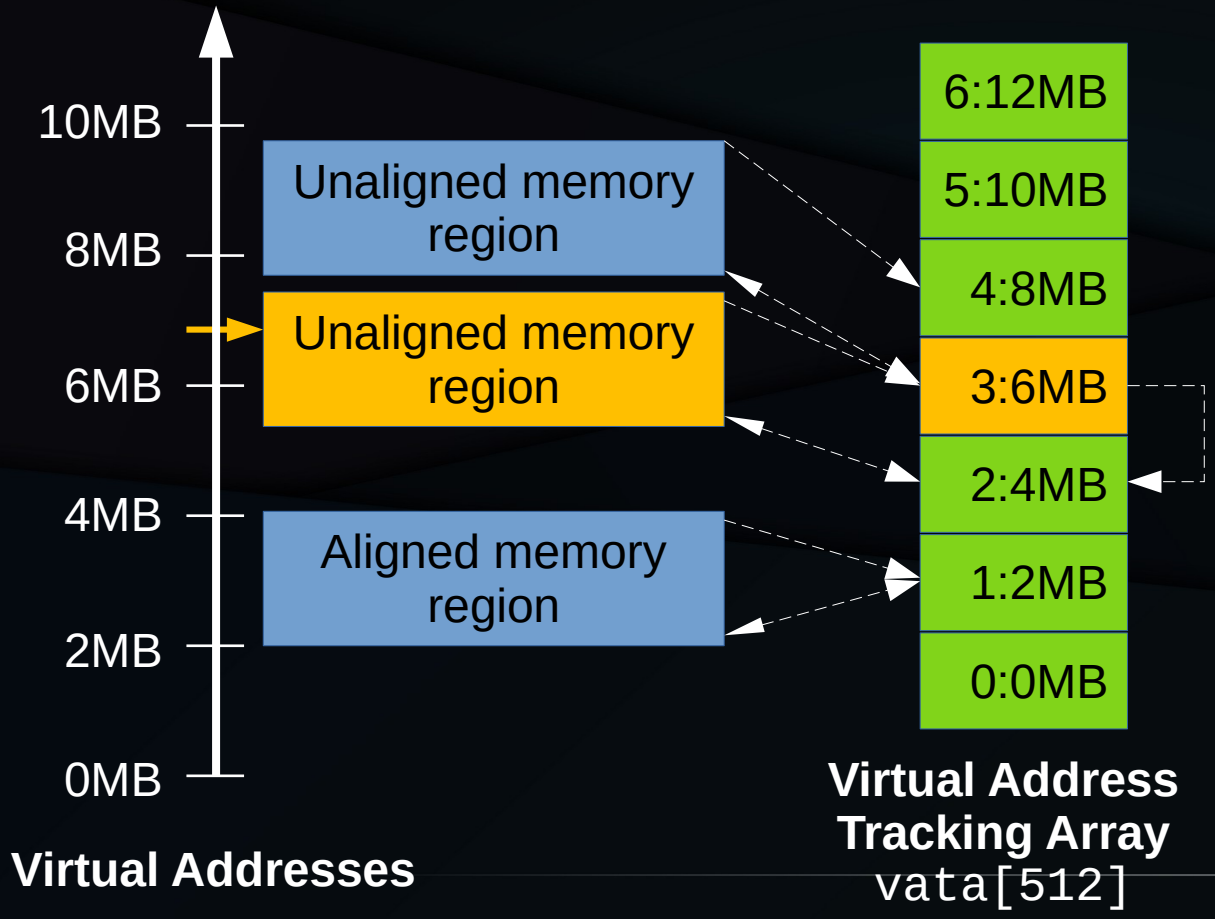


Hint From Stack Trace



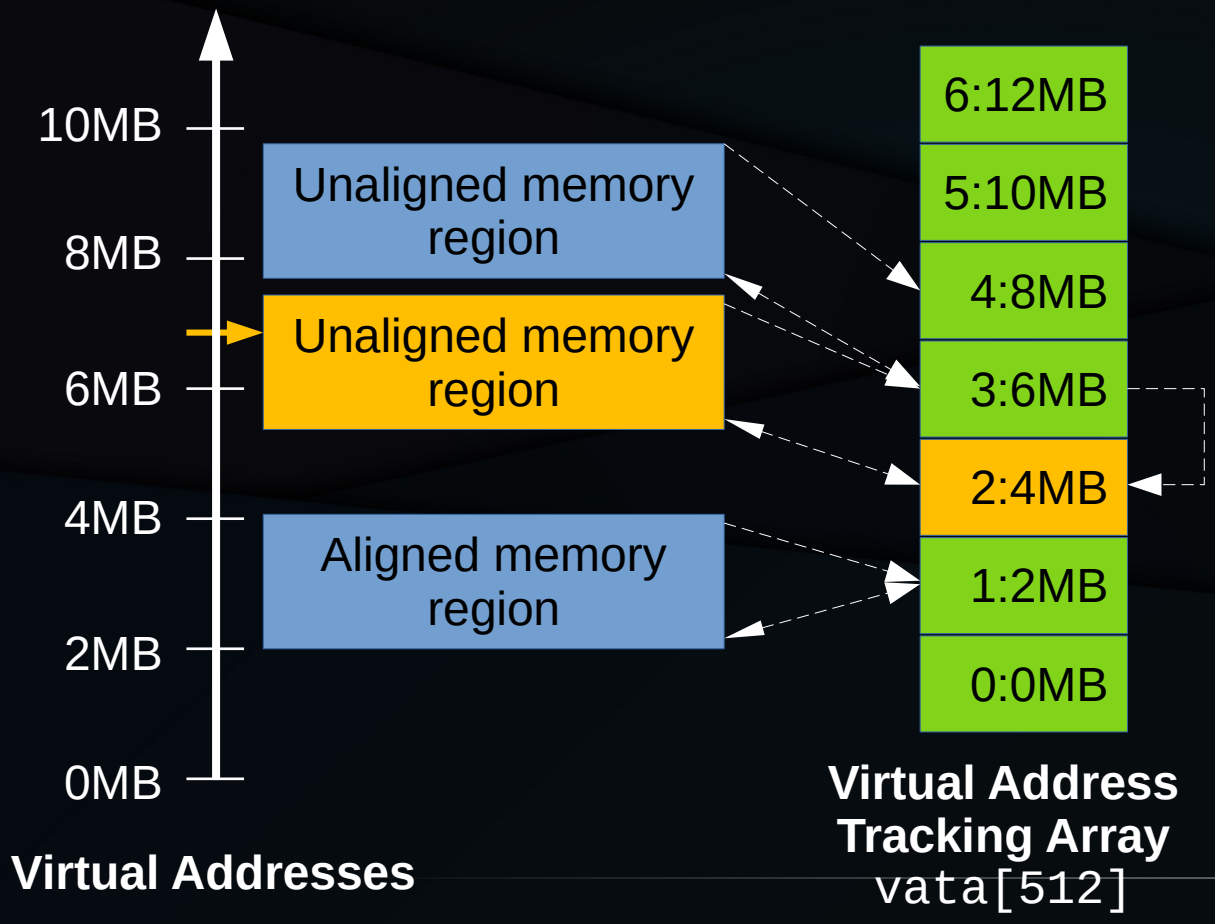

```
int idx = vadr / (2 * MB);
void *vta;

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

Hint From Stack Trace



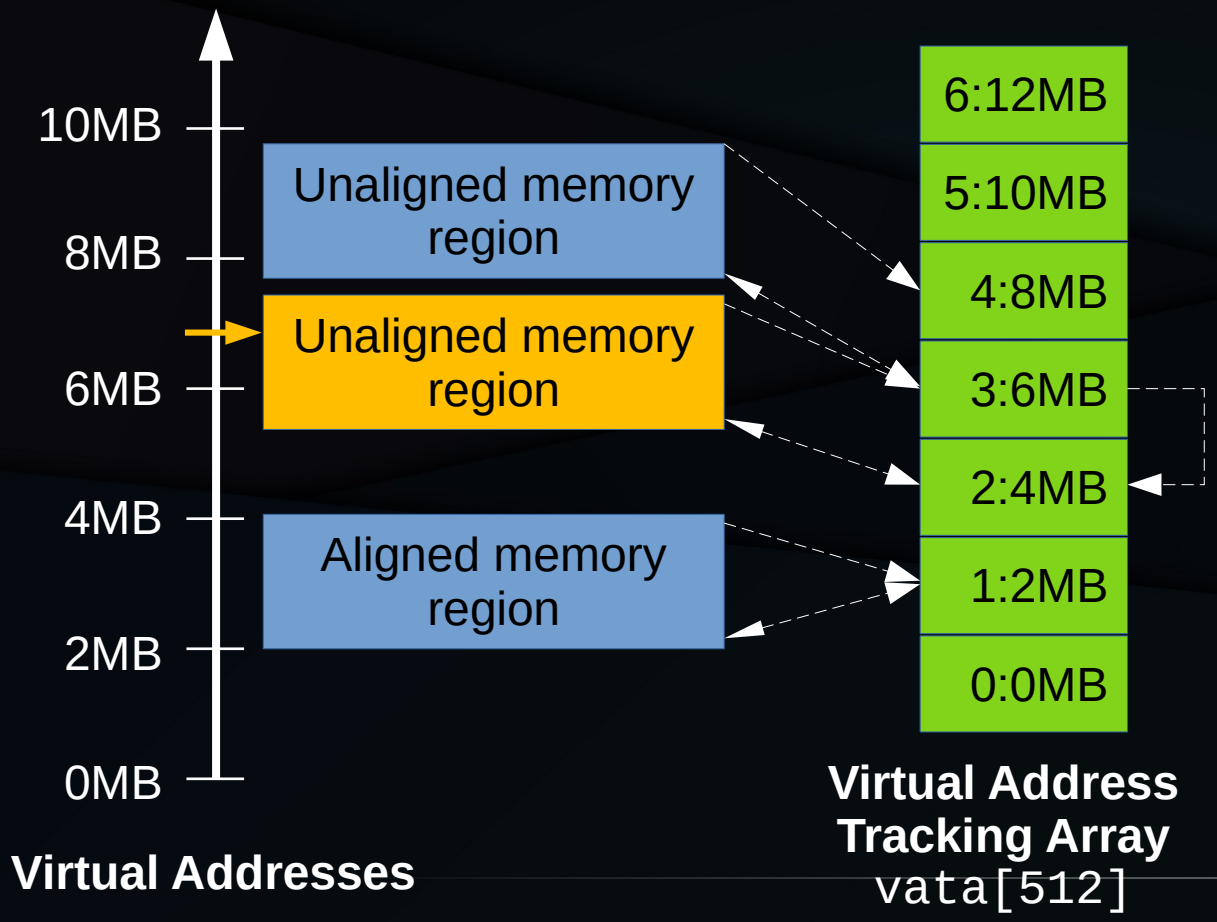

```
int idx = vadr / (2 * MB);
void *vta;

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

Hint From Stack Trace



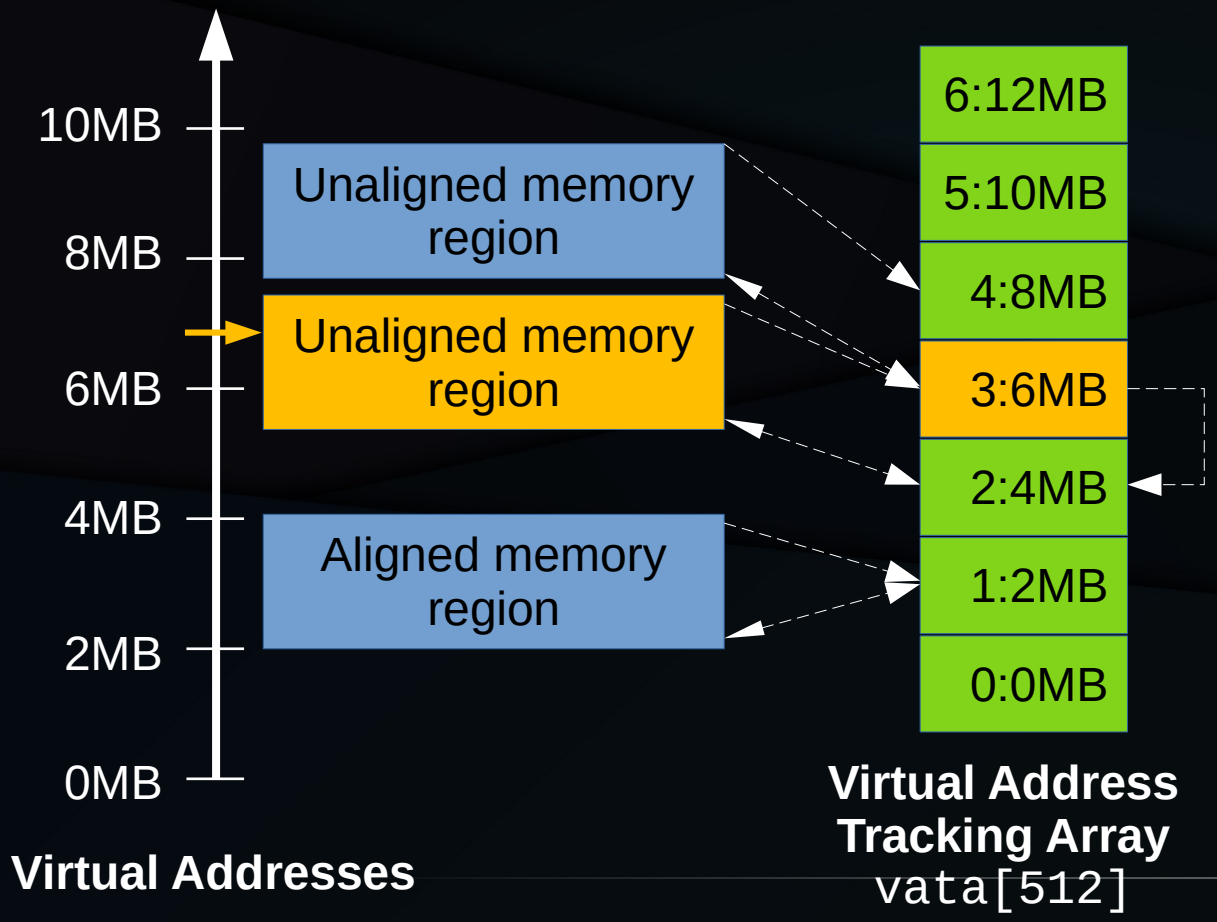

```
int idx = vadr / (2 * MB);
void *vta;

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

Hint From Stack Trace: Compiler Fun



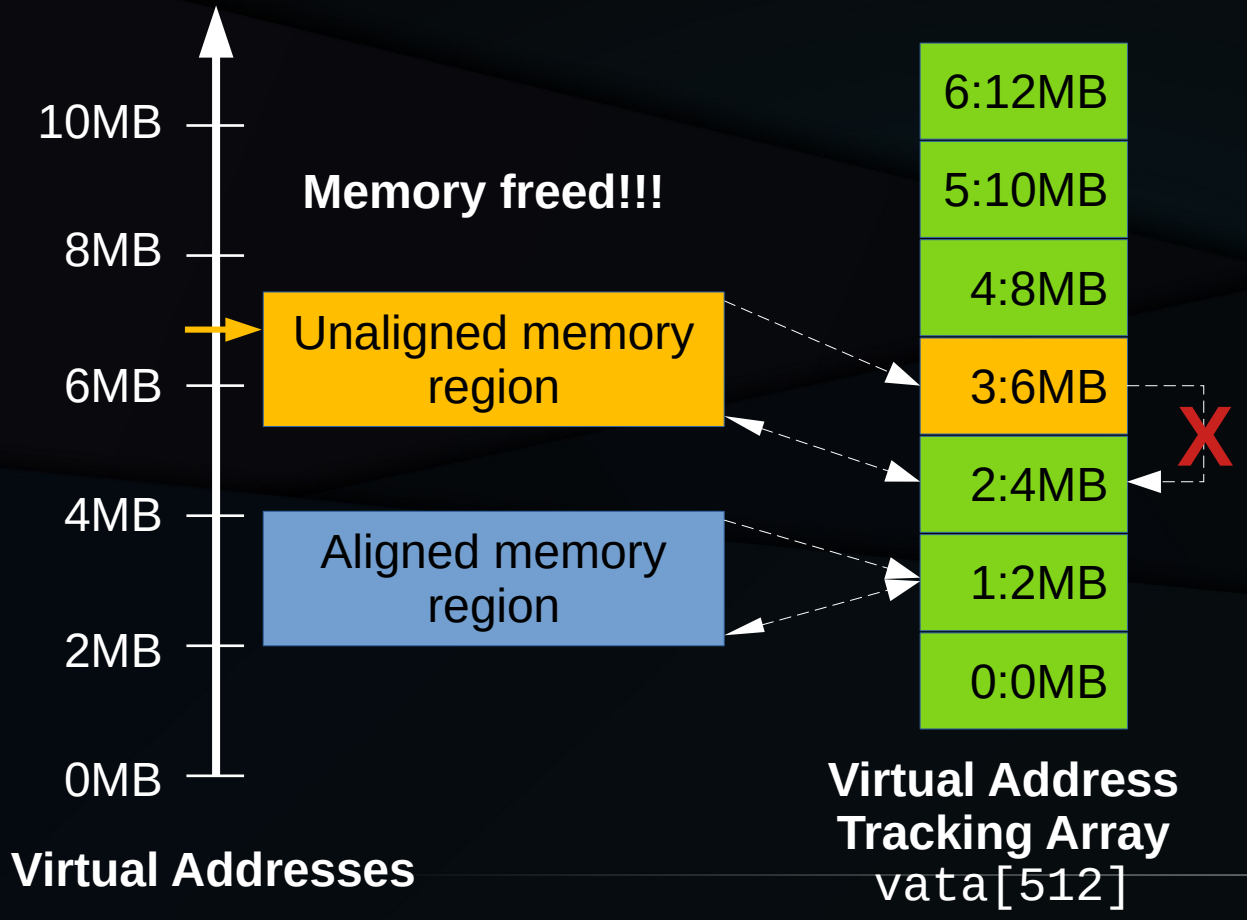

```
int idx = vadr / (2 * MB);
void *vta;

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

Compiler Fun In Failure Case



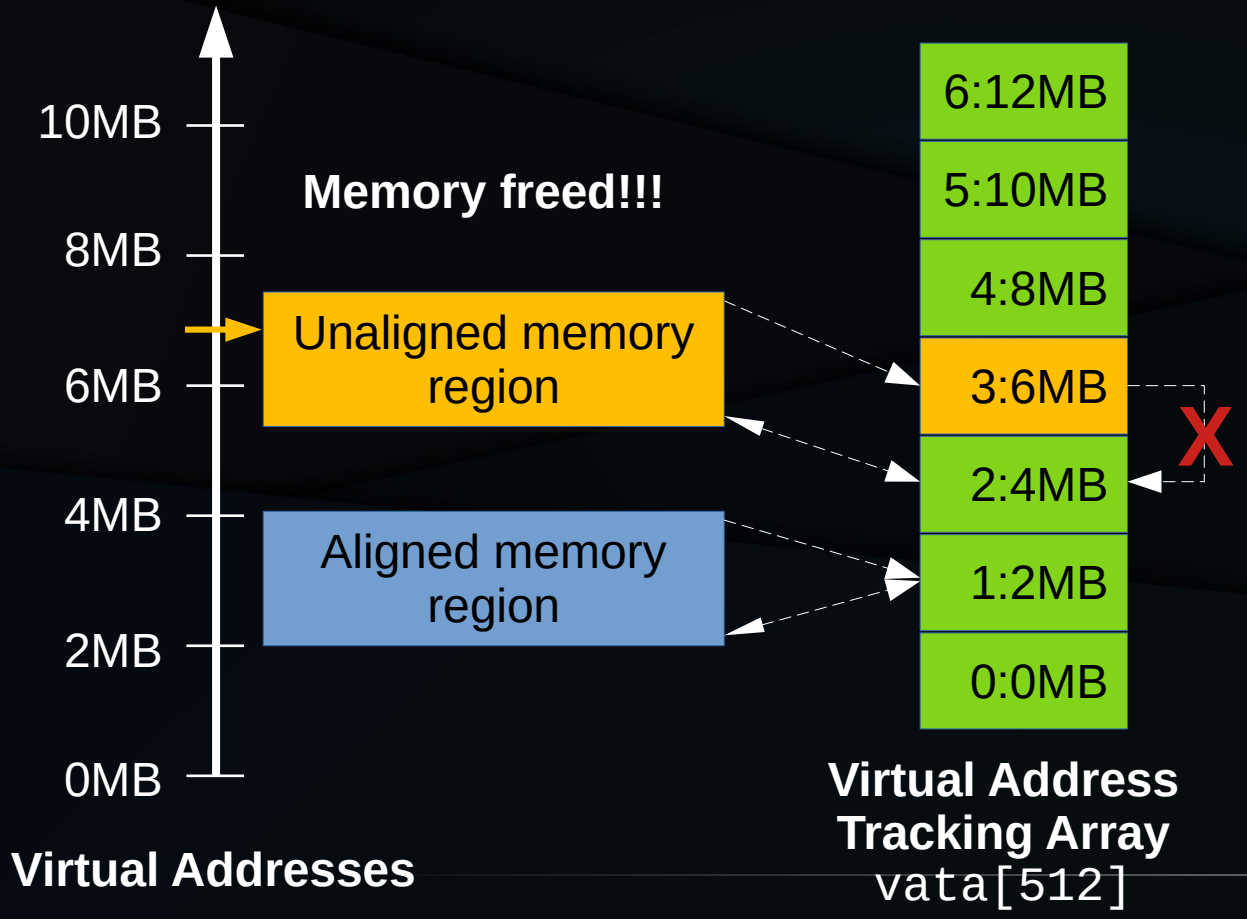

```
int idx = vadr / (2 * MB);
void *vta;

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

Compiler Fun In Failure Case: Update

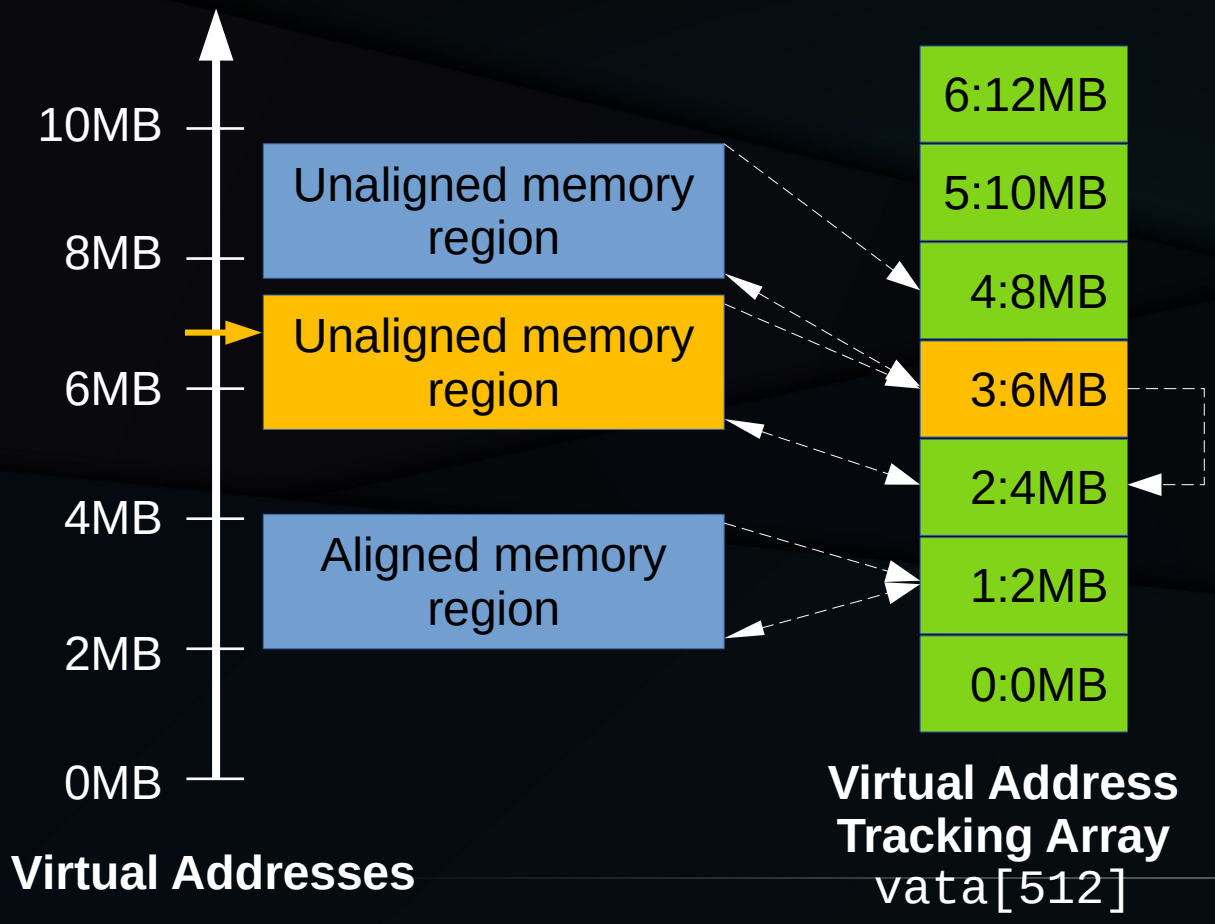



```
int idx = vadr / (2 * MB);
void *vta;

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

Compiler Fun In Failure Case: Update

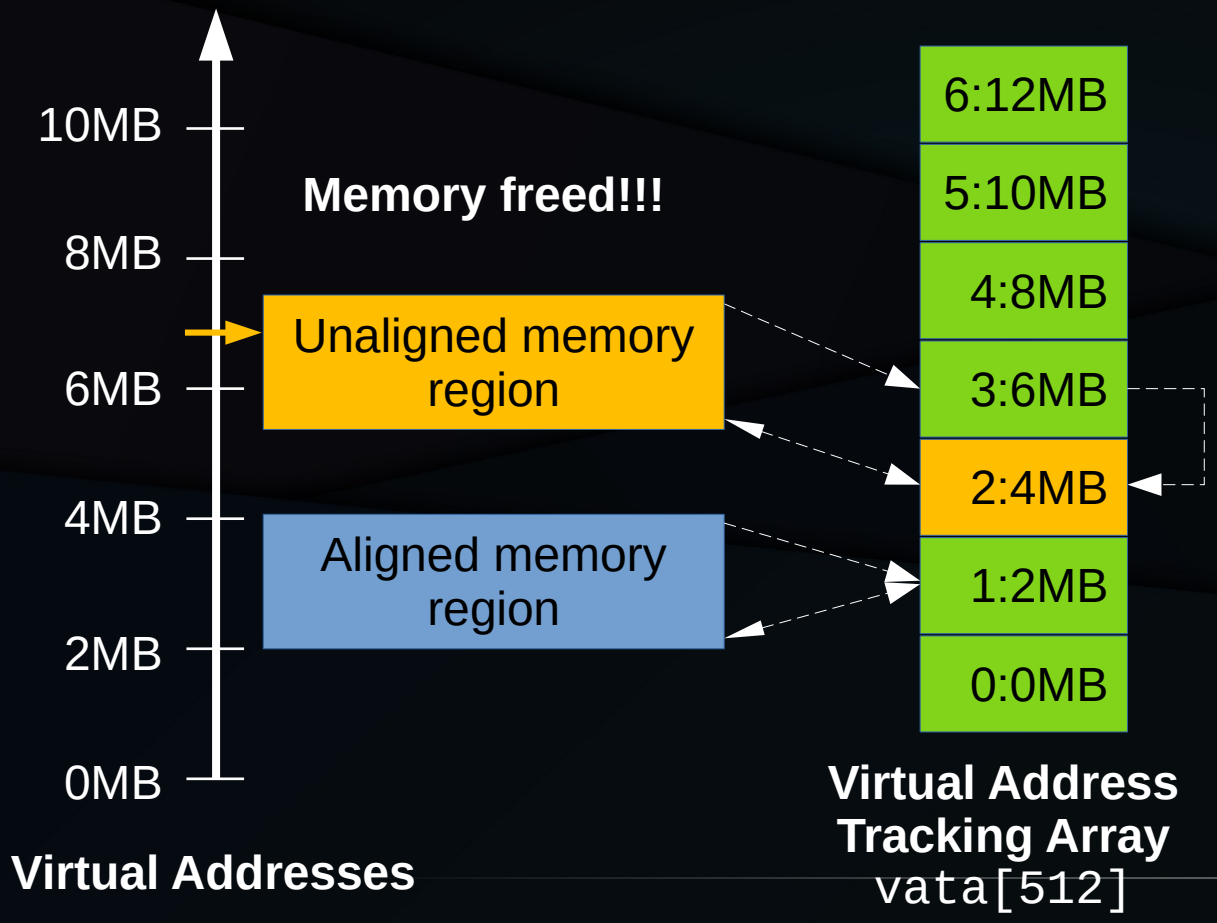


Thwarting Compiler Fun



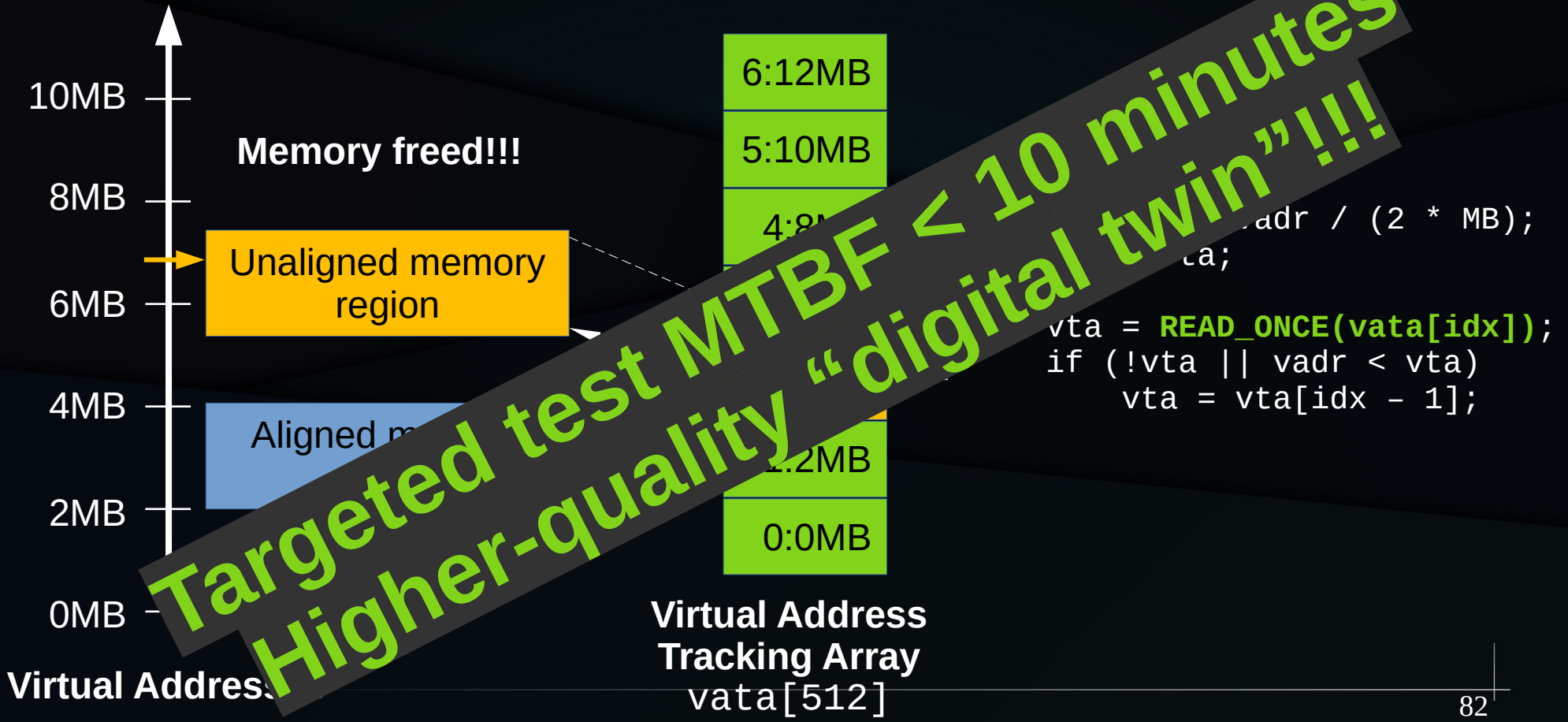

```
int idx = vadr / (2 * MB);
void *vta;

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];
```

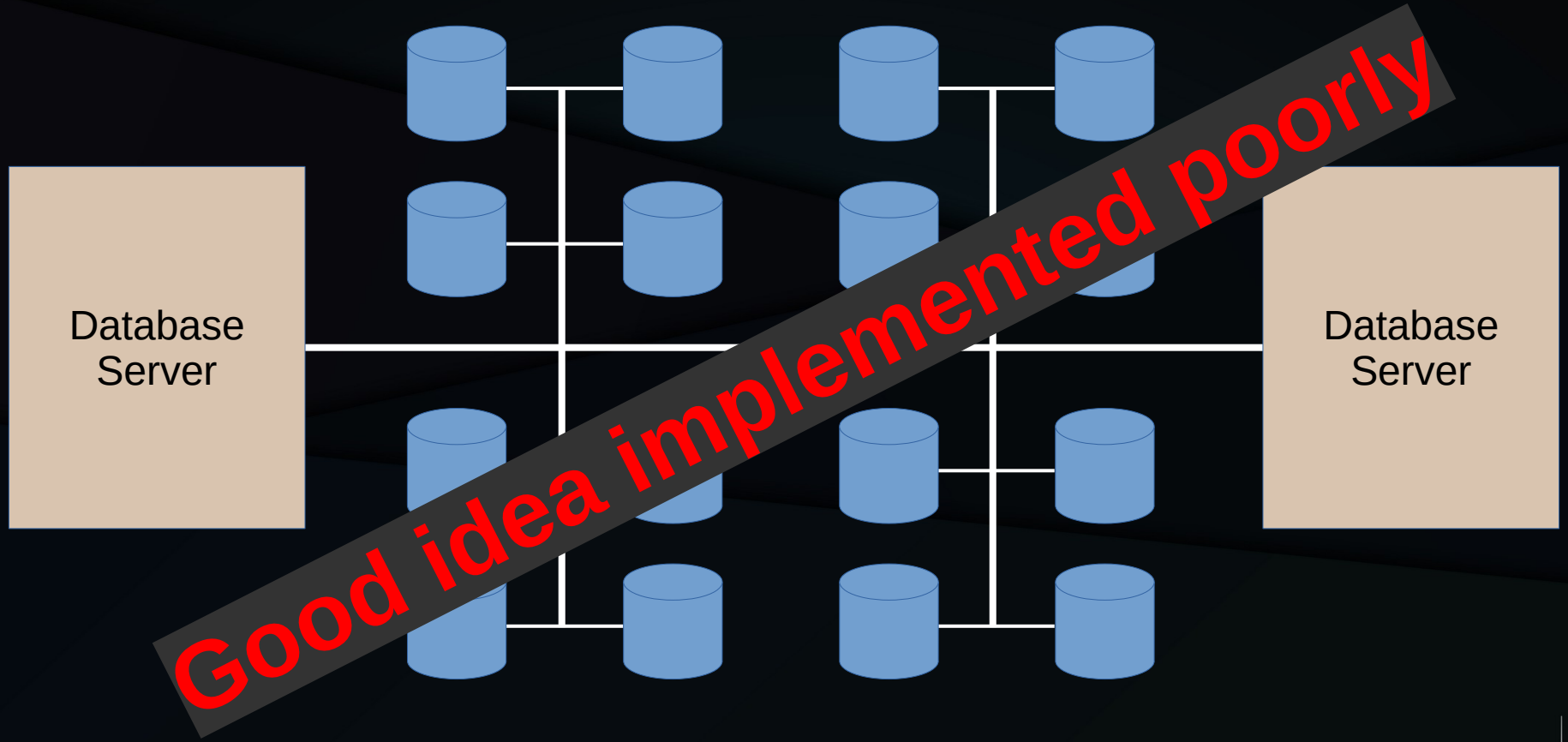
Thwarting Compiler Fun: Update OK



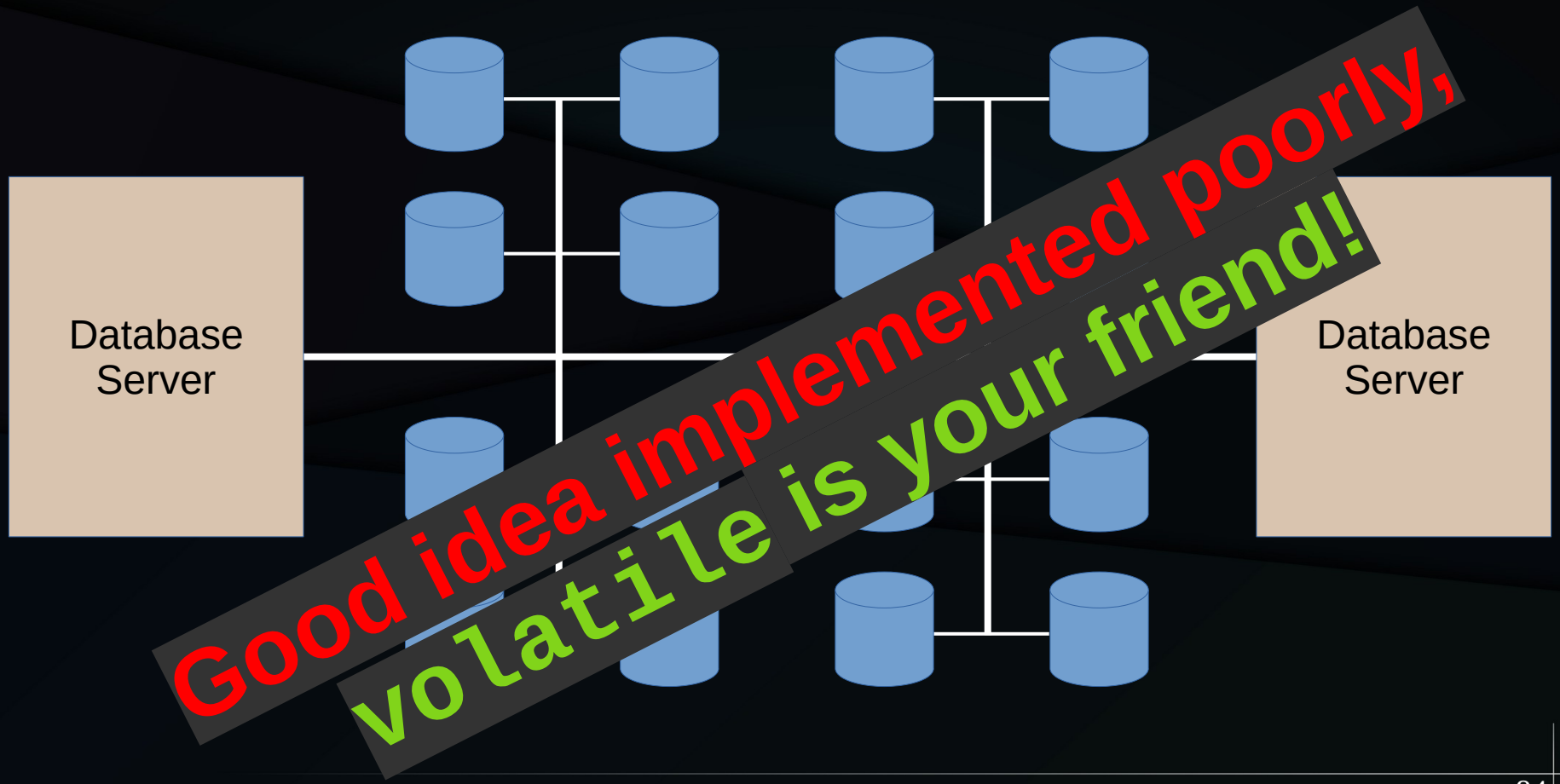

```
adr / (2 * MB);
ta;

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

Targeted test MTBF < 10 minutes
Higher-quality "digital twin"!!!

Shared Disks For Availability Win!!!



Shared Disks For Availability Win!!!



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 Unsatisfied 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 Unsatisfied Customer

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

Good idea implemented properly,

One Unsatisfied Customer

- Senior girl matched only with fres
  - And she really did check the
- Program looked to b
- Turned out to l error
- Correct t 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

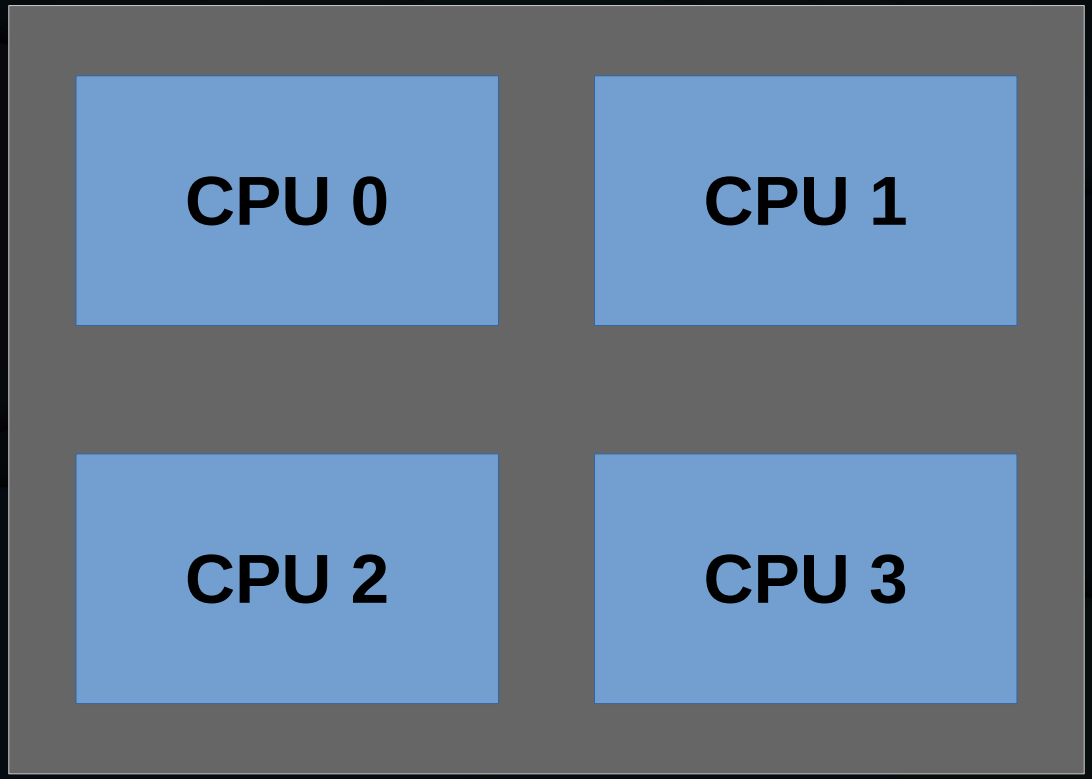
2004: Real-Time Linux

- 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

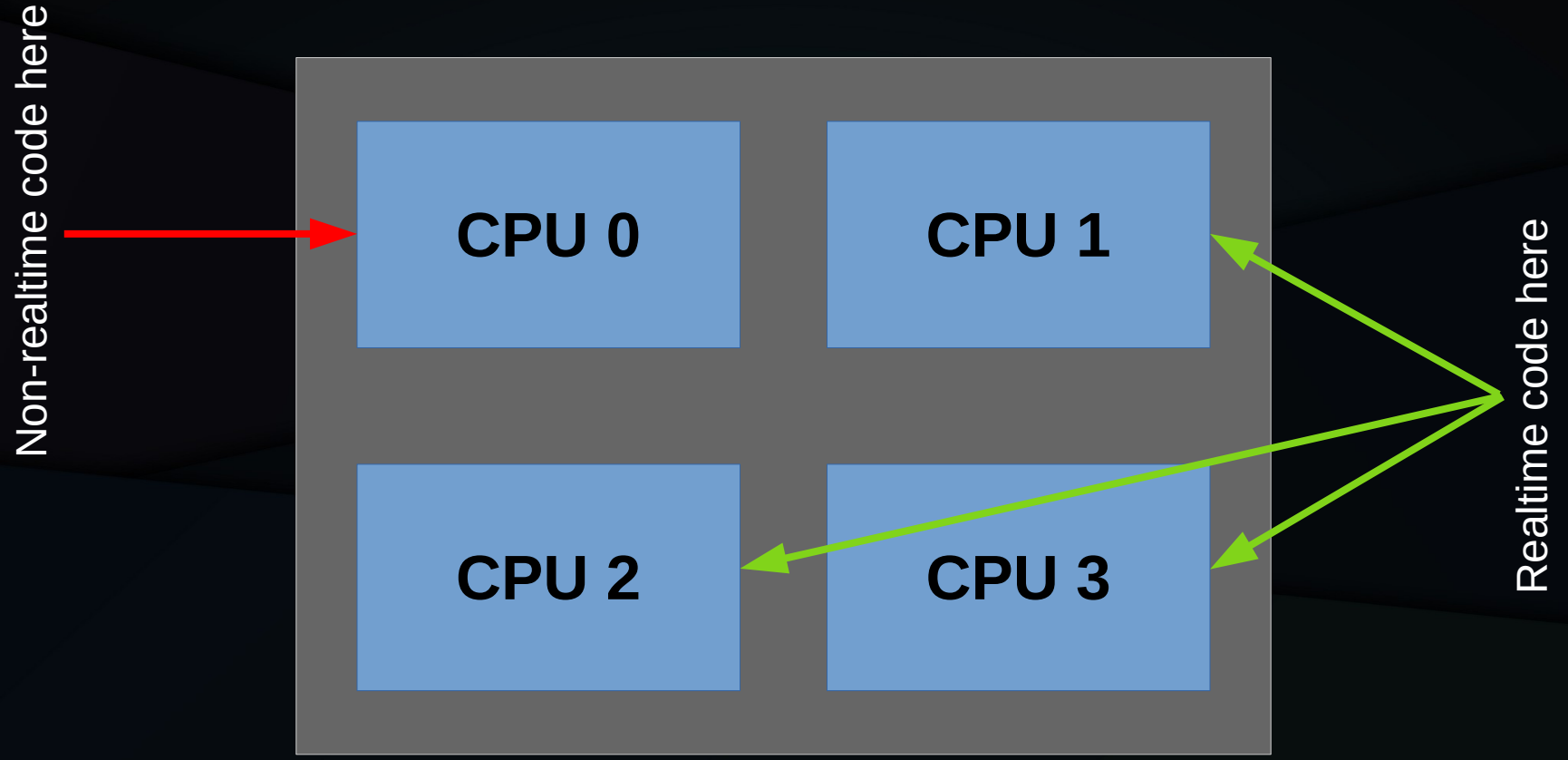
2004: Real-Time Linux

- Early 2000s: Many requests 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

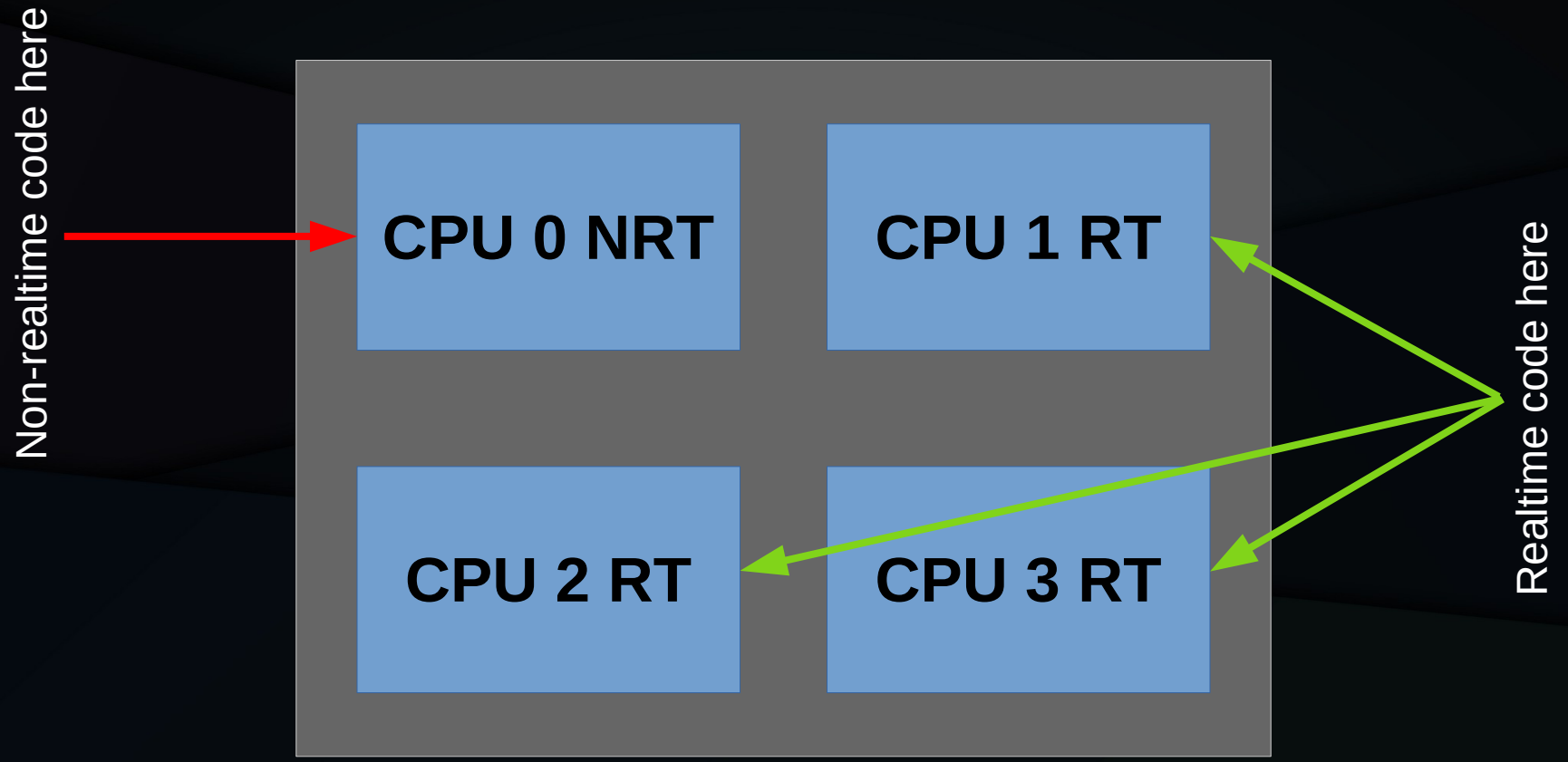
2004: Dawn of Multicore Embedded



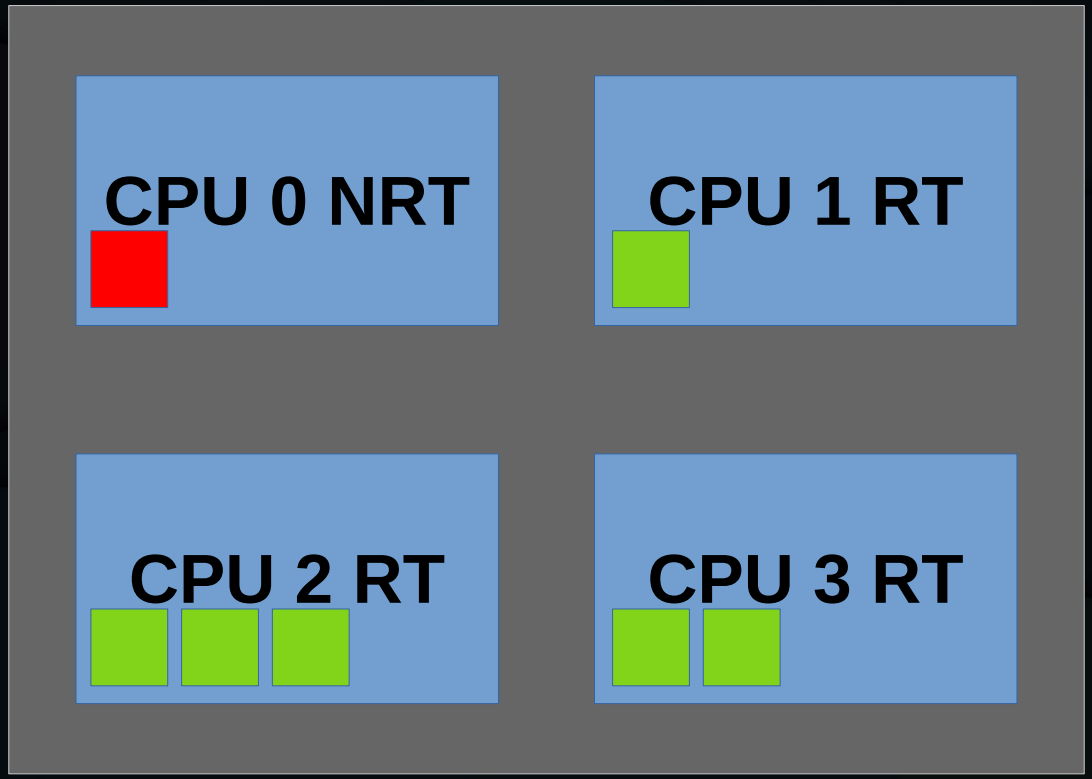
Multicore Embedded for Real Time!!!



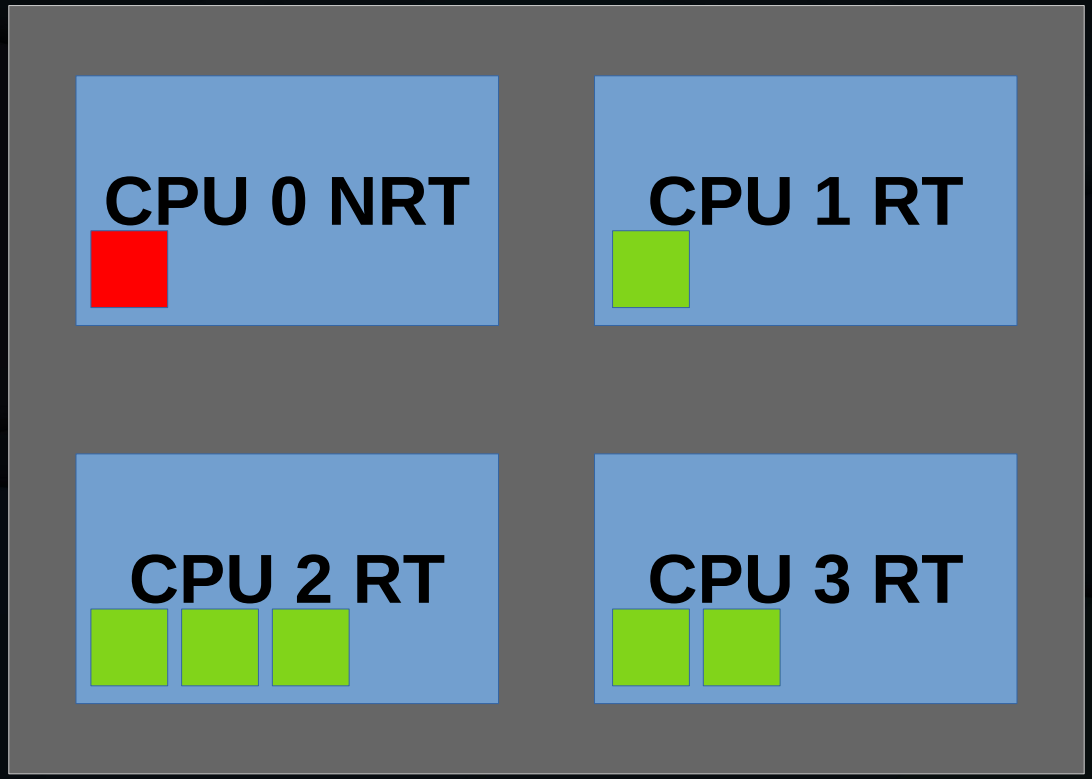
Multicore Embedded for Real Time!!!



Multicore Embedded for Real Time!!!

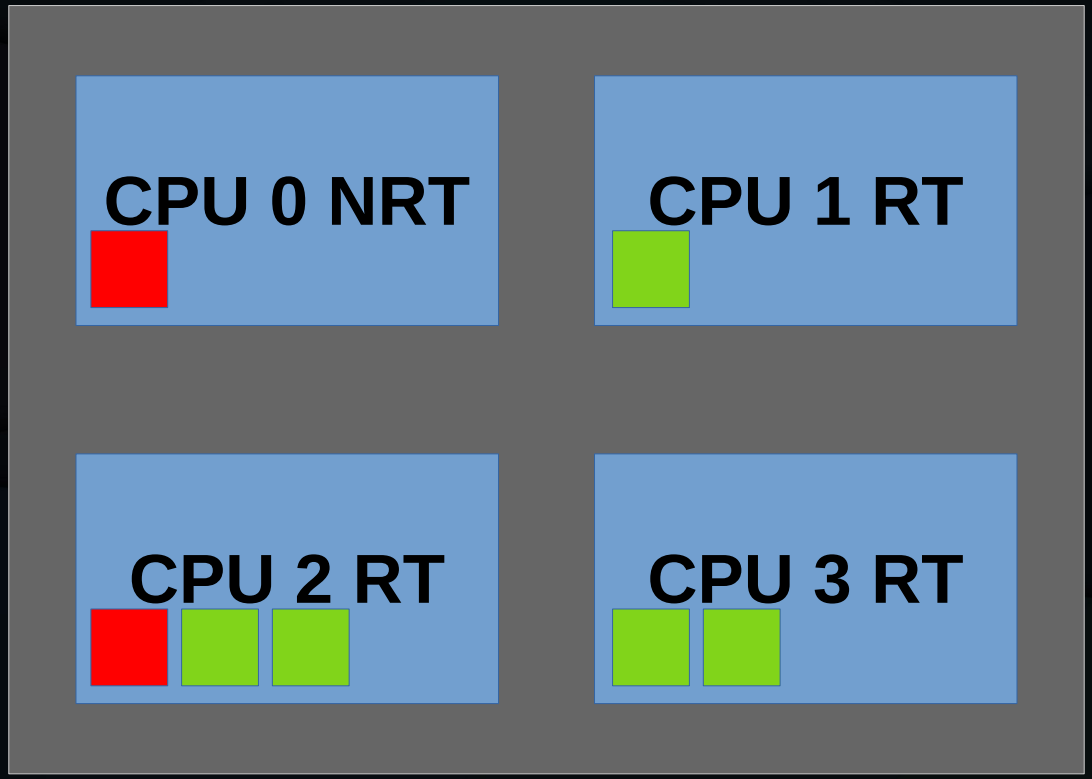


Multicore Embedded for Real Time!!!



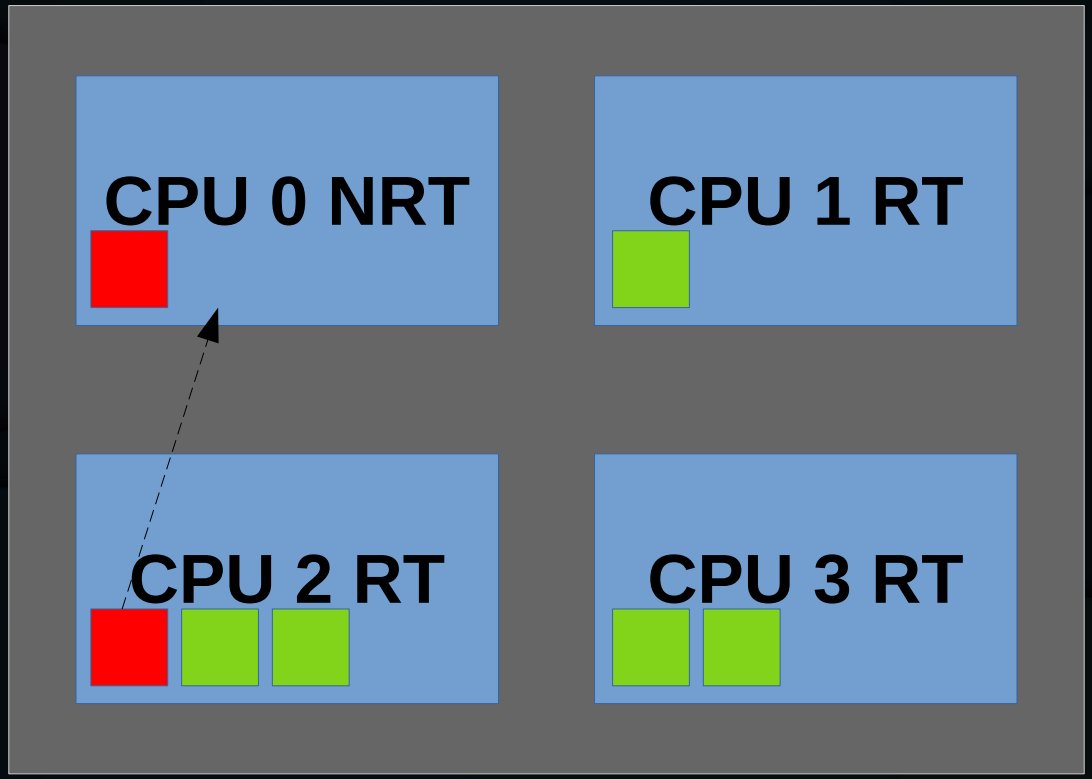
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



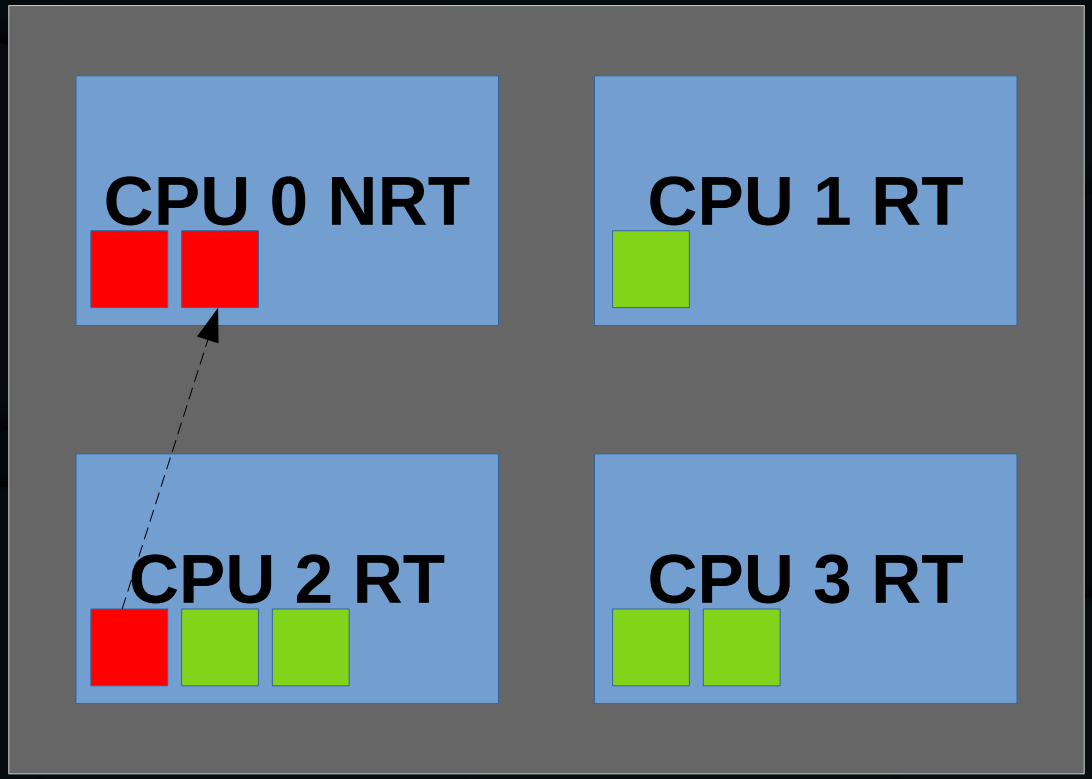
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



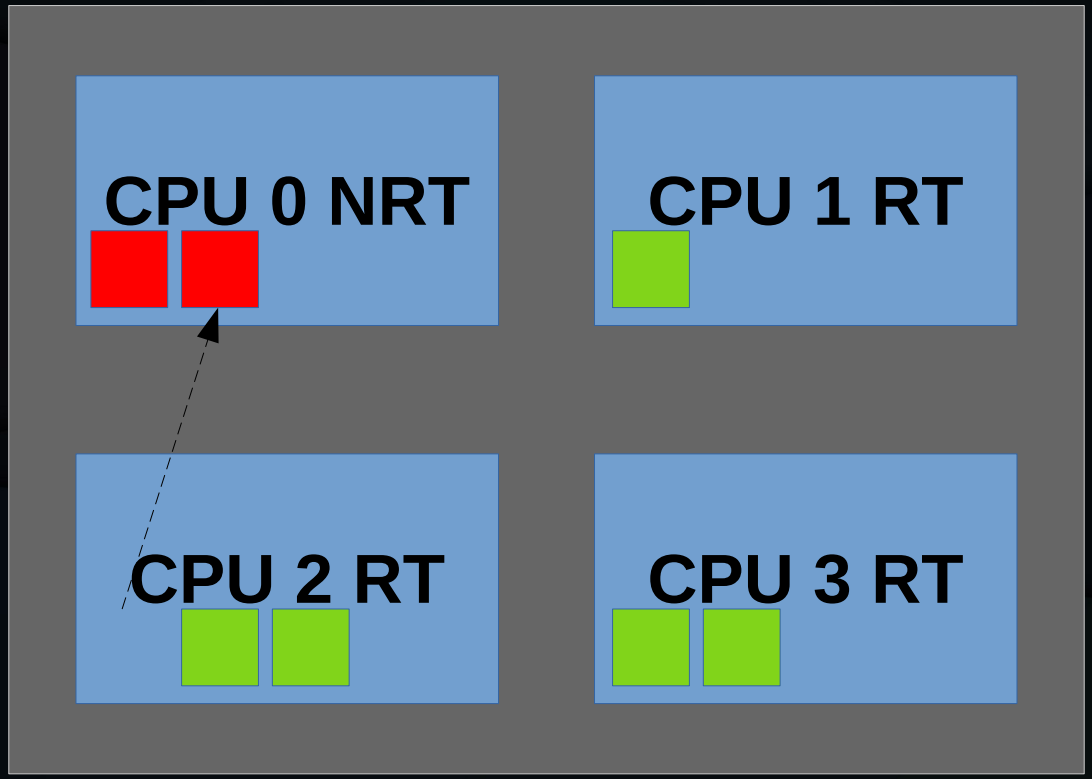
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



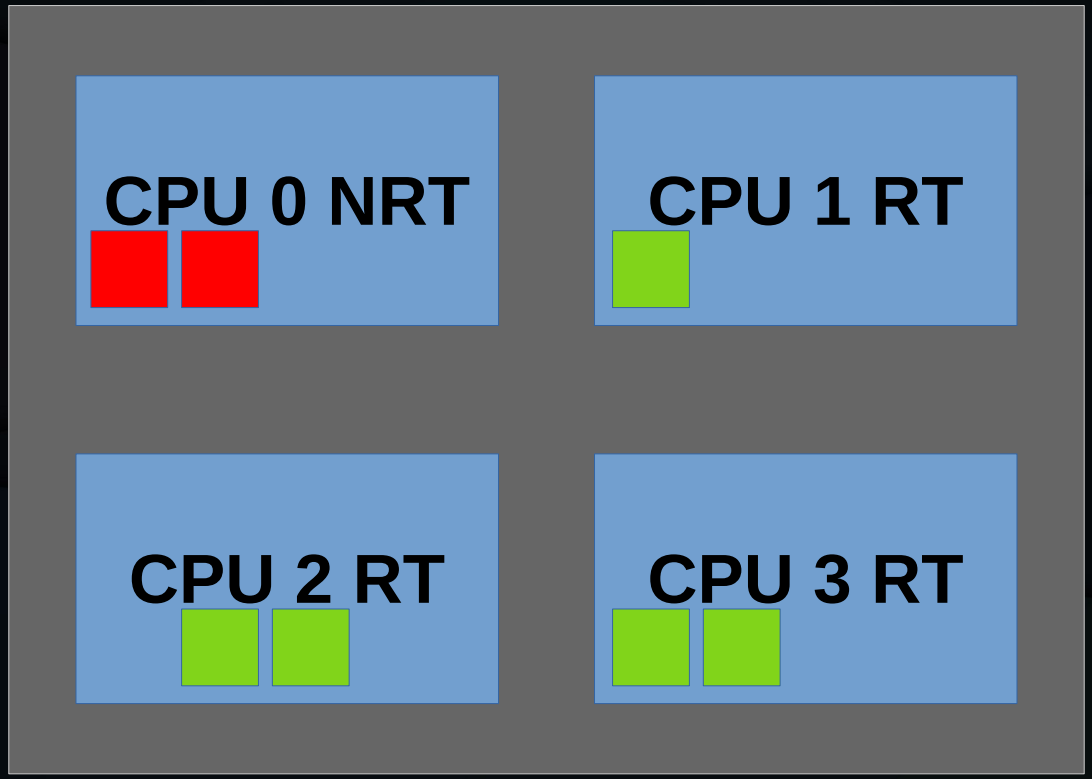
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



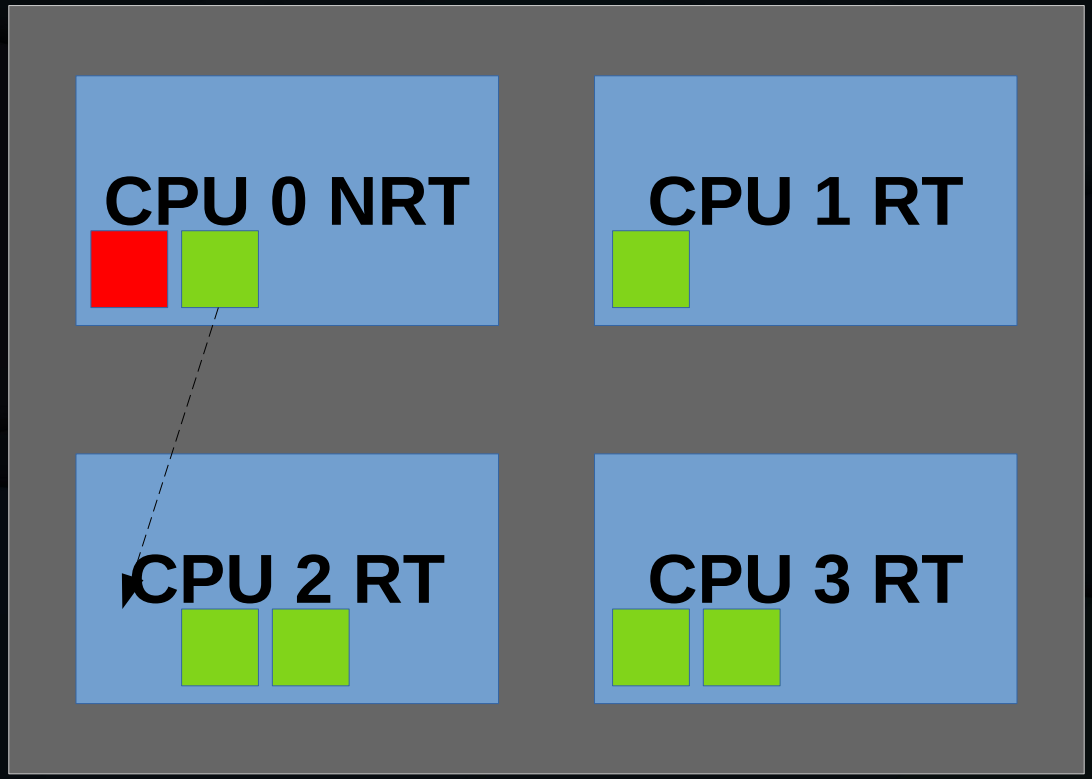
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



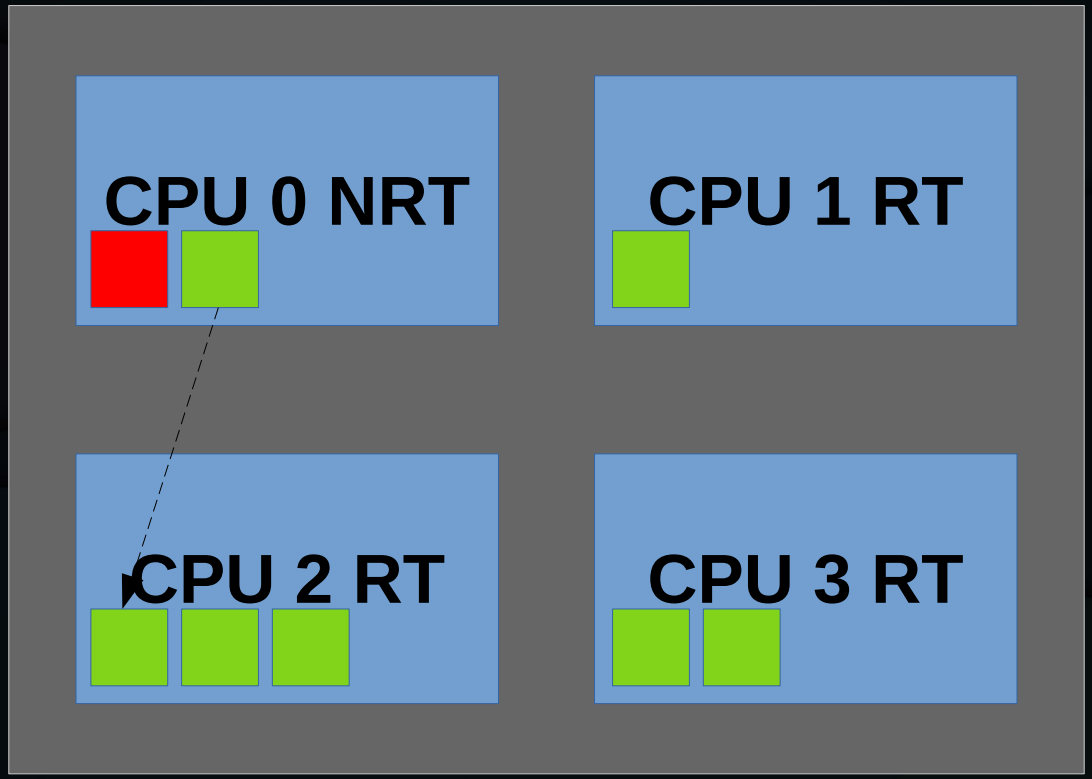
Respond to non-real-time activity by migrating.

Multicore Embedded for Real Time!!!



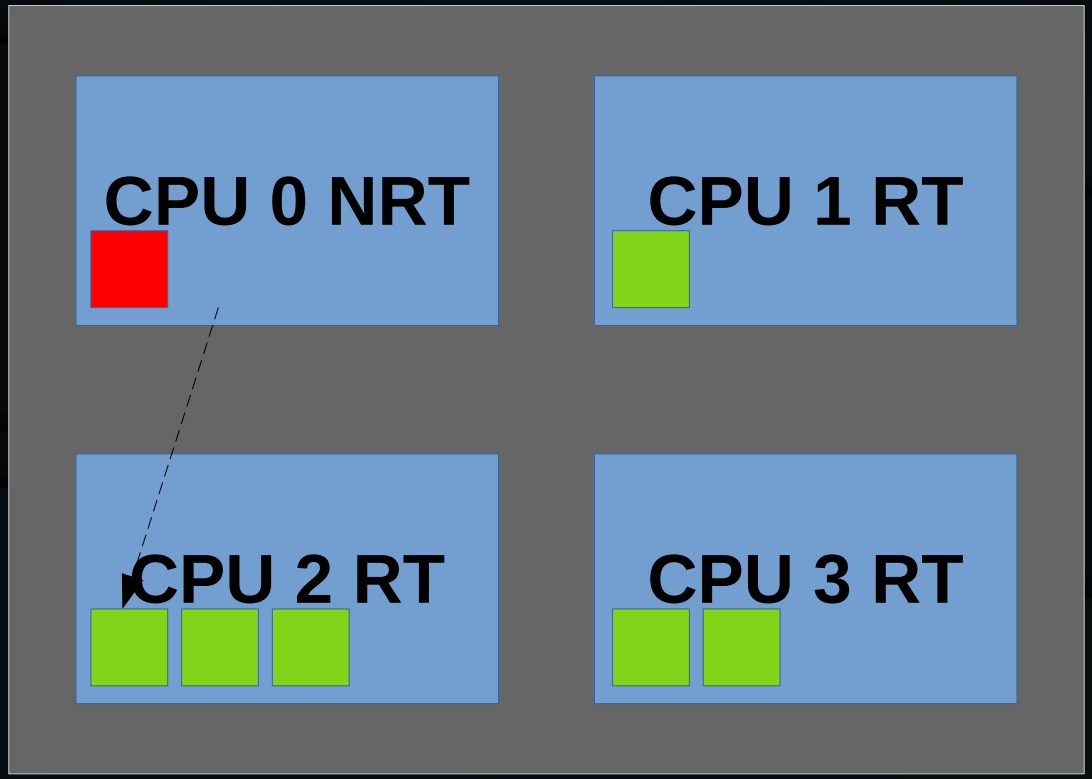
Respond to non-real-time activity by migrating.and back when done (system call).

Multicore Embedded for Real Time!!!



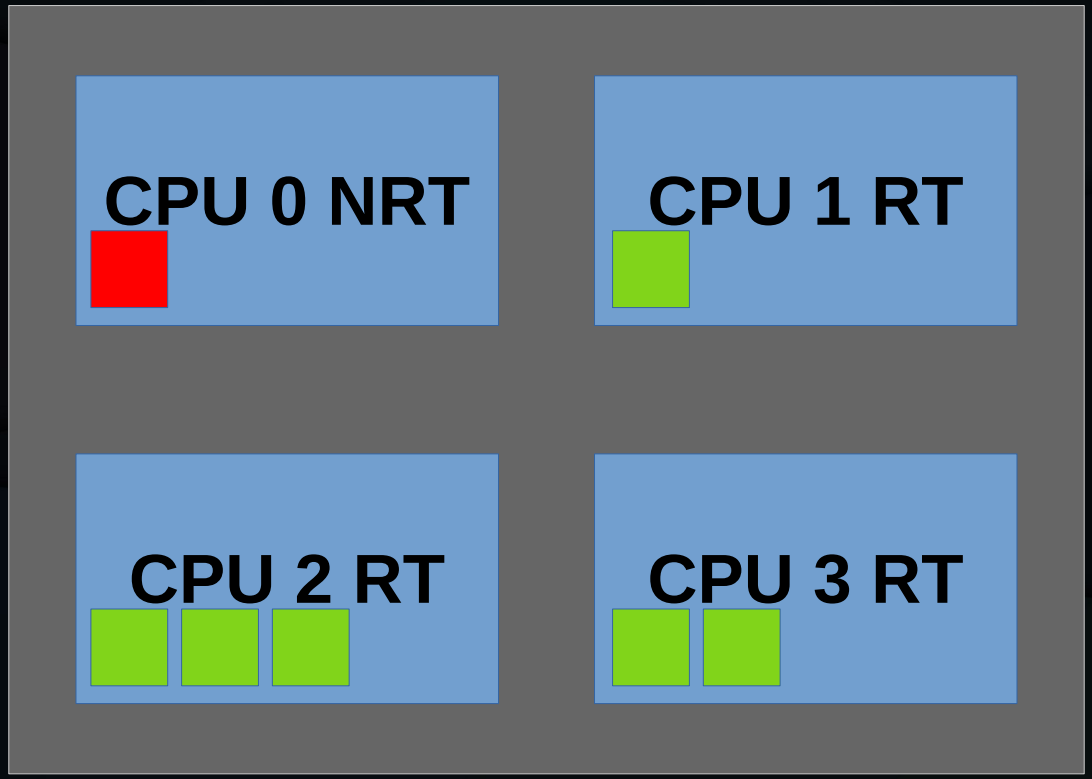
Respond to non-real-time activity by migrating.and back when done (system call).

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 sys[illegible]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_call migration
- Test it out, works great!
- Inform executives re[illegible]e Linux is real!!!
- No more need [illegible]o-bid!!!
- And we win a 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 implem[illegible]g [illegible]al[illegible]igration
- Test it out, works
- Inform executi[illegible]nux is real!!!
- No more n[illegible]
- And we wi[illegible]ge [illegible]ntract!!!

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

Multicore Real Time Linux Actions

- patch implem ng a all igration
- Test it out,
- Inform executi nux is real!!!
- No more n
- And we wi ge ntract!!!

Remember that rewrite-the-kernel effort?

Except that we have contractual commitments to meet

Multicore Real Time Linux Actions

- patch implem ng a 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

Excep cted!!! we have

contractu mitments

to k

Multicore Real Time Linux Actions

- patch implem… …igration
  - Tes…
- I… …nux is real!!!
- No more i…
- And we wi… …ge …

Remember that rewrite-the-kernel effort?

…cted!!! …we have …mitments

Well, I helped them with RCU

Three from-scratch implementations

E…

contra…

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

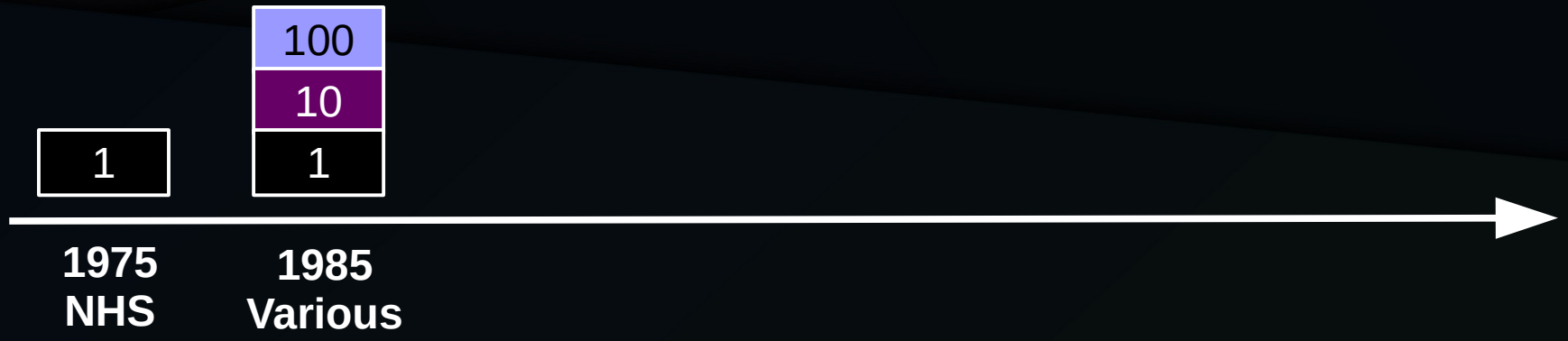
Million-Year Bug? Once In a Million Years!!!
Murphy is a nice guy: Everything that can happen, will...

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

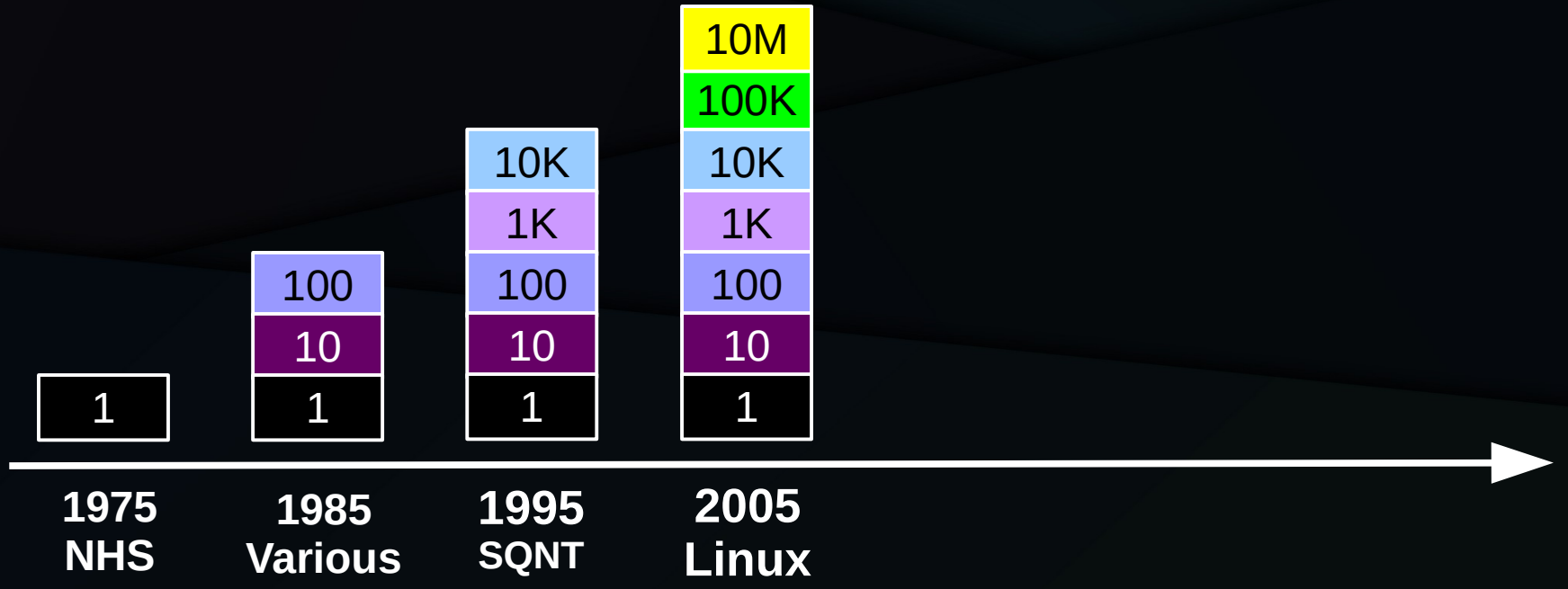


Installed Base

Million-Year Bug? Once per Century

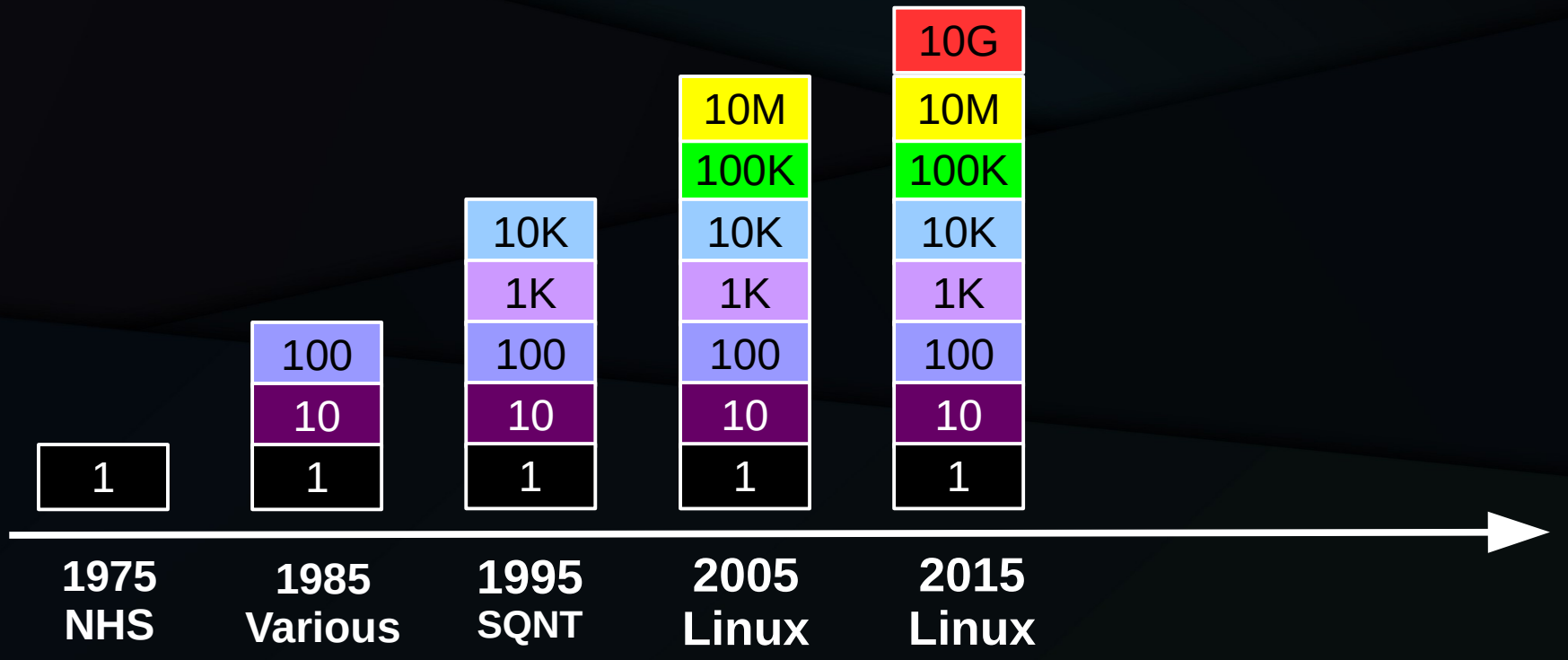
Installed Base

Million-Year Bug? Once a Month



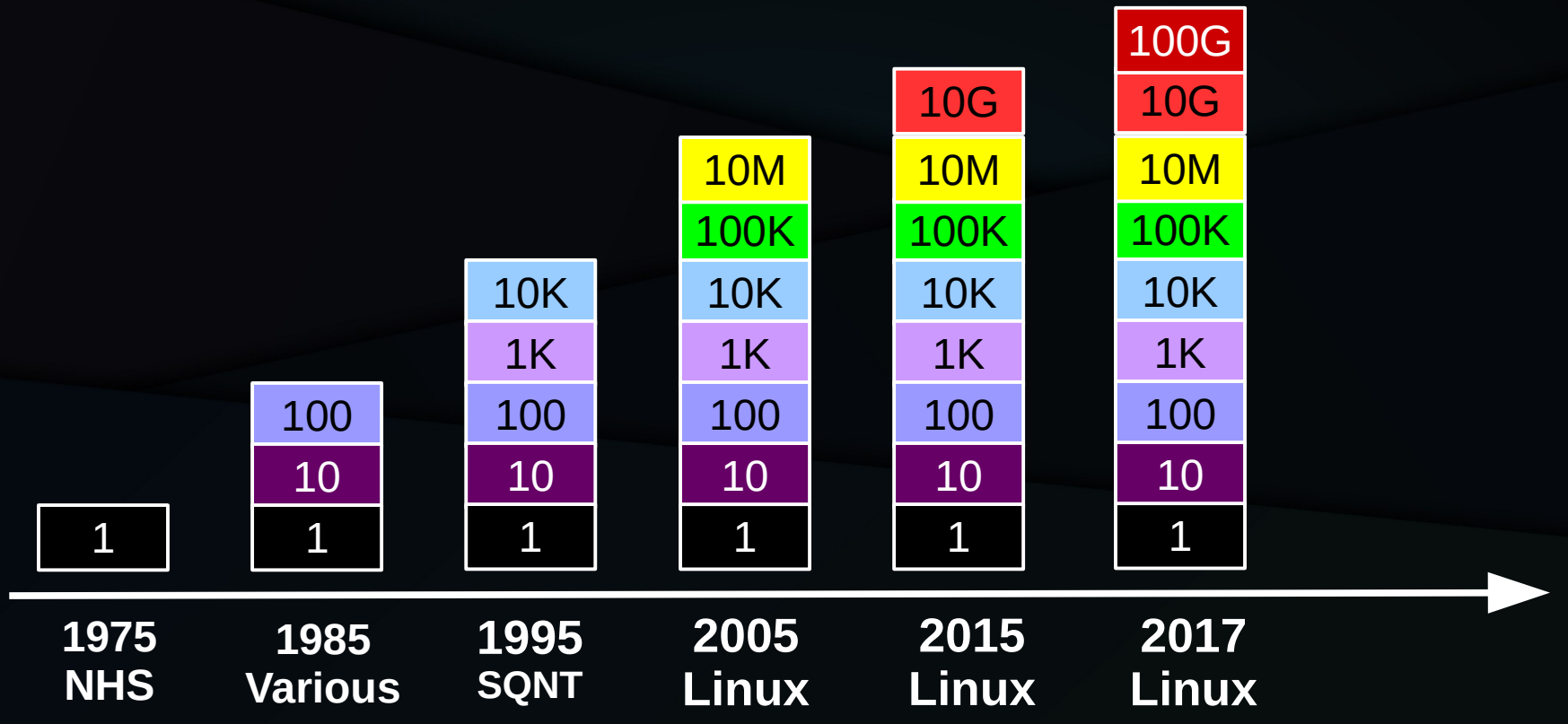
Installed Base

Million-Year Bug? Several Times per *Day*



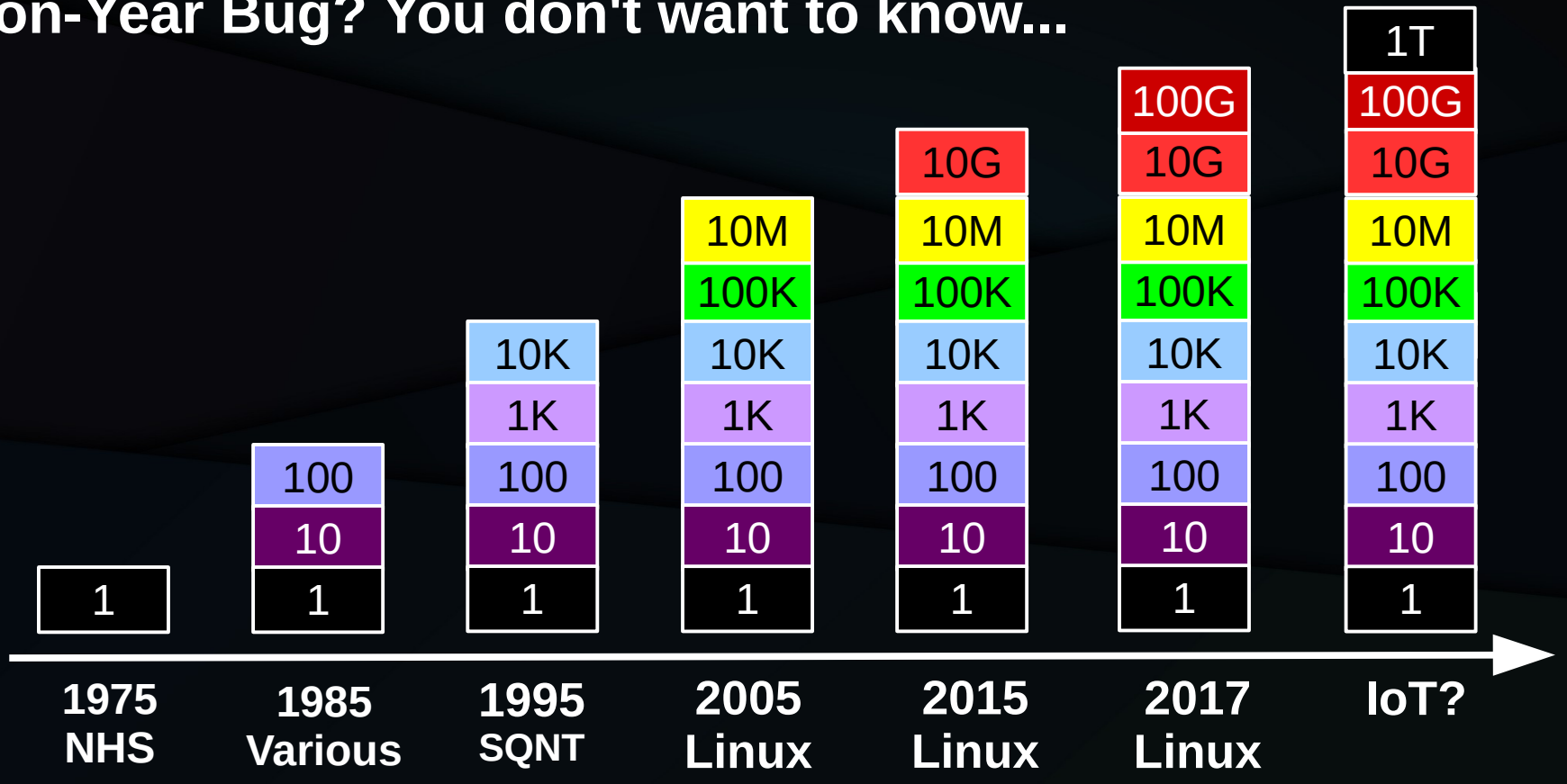
Installed Base

Million-Year Bug? Several Times per *Hour*



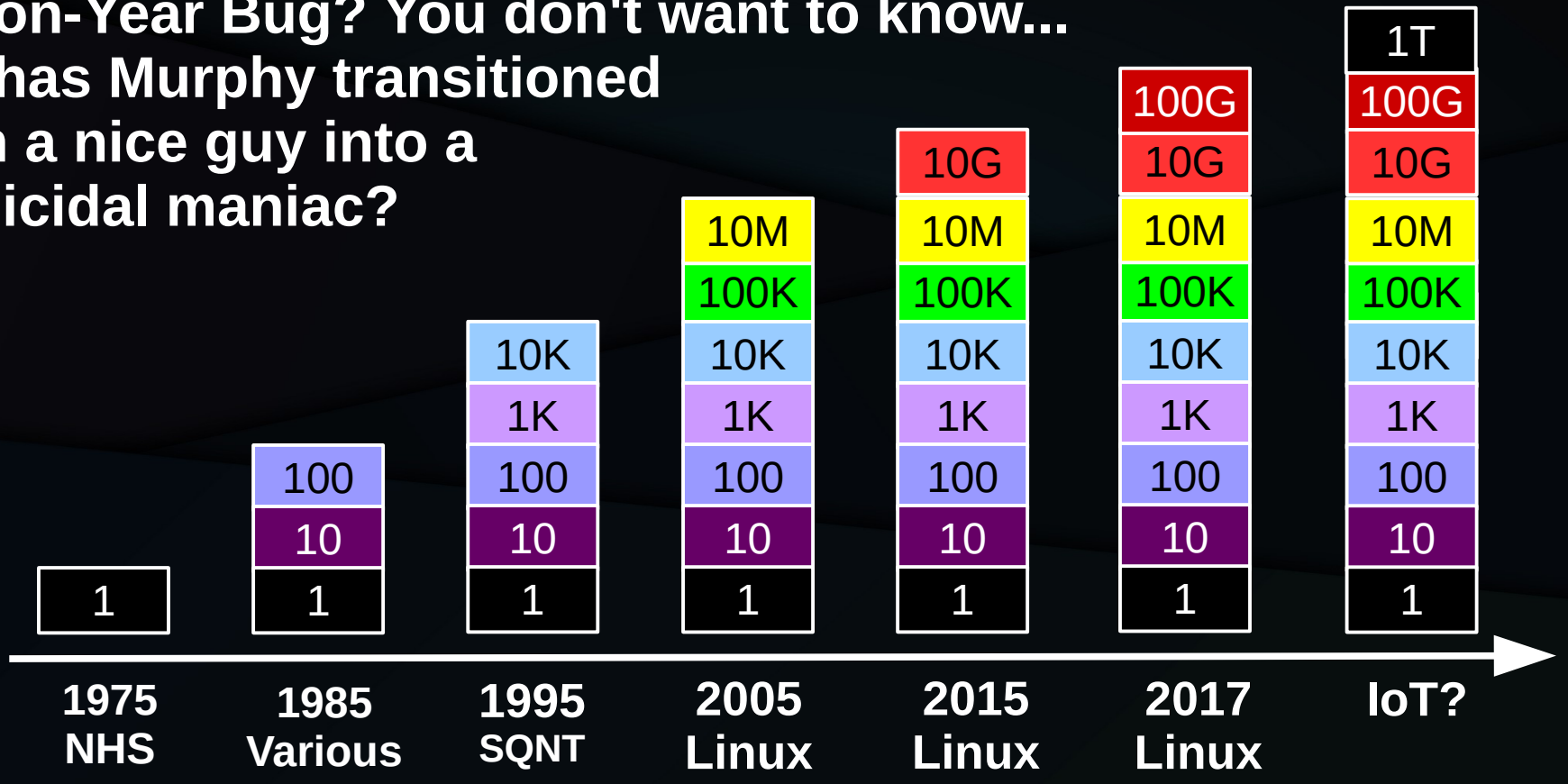
Installed Base

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 algorithm
- 2007: “Quick” RCU (QRCU) v[illegible]tion
- 2008: RCU idle-detection [illegible]nergy efficiency
- 2012: Verify users[illegible]RCU
- 2014: Verify [illegible]idle detection for NMIs
- 2018: [illegible]eavy use of herd7 and LKMM

Verifying design, not regression testing

Formal Verification Experience

- 1993: Promela/spin election algorith
- 2007: “Quick” RCU (QRCU) v
- 2008: RCU idle-detectio ciency
- 2012: Verify user
- 2014: Verif 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
- “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, [illegible]cidable
- "Macho" verification requir[illegible] specification
  - Which is large, thus [illegible]ning lots of bugs!
- Successful form[illegible]erification highly restricted:
  - Small p[illegible]ns, simple properties of large pro[illegible], or execution-guided verification

Powerful when used properly

Formal Verification is *Expensive*

- At best, exponential; in general, [illegible]ble
- "Macho" verification requir[illegible]ification
  - Which is large, thus [illegible] of bugs!
- Successful form[illegible]ion highly restricted:
  - Small p[illegible]ple properties of large pro[illegible]ecution-guided verification

Powerful when used properly
How to verify the verification?

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 verificati… e small
  - Check for … rable properties
    - File b… eports as appropriate

De-risk via one-way bet

Cautionary Quote (Redux)

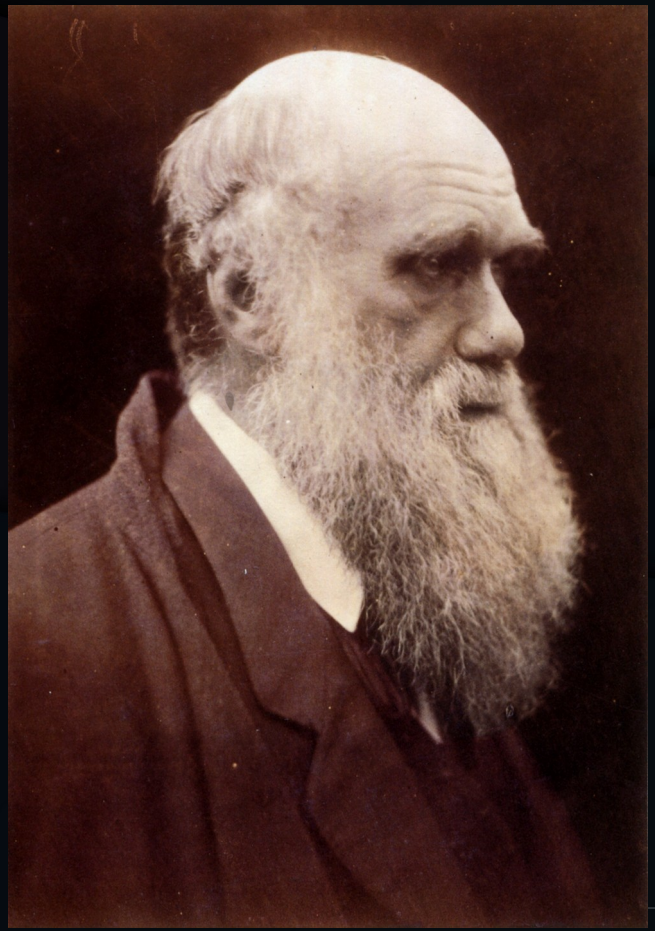
- A lot of success in life and business comes from knowing what you want to avoid. - *Charlie Munger*

Cautionary Quotes

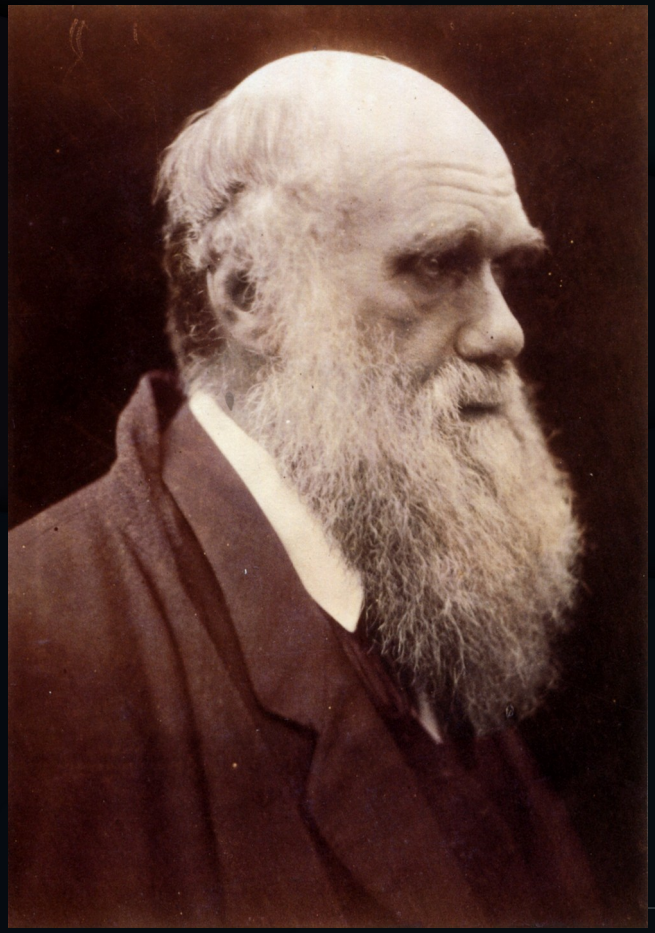
- Sometimes you don't even know what you want until you find out you can't have it. - *Meghan O'Rourke*
- Sometimes we don't know what we want until we don't get it. - *Sloane Crosley*
- We don't know what we want, but we are ready to bite somebody to get it - *Will Rogers*

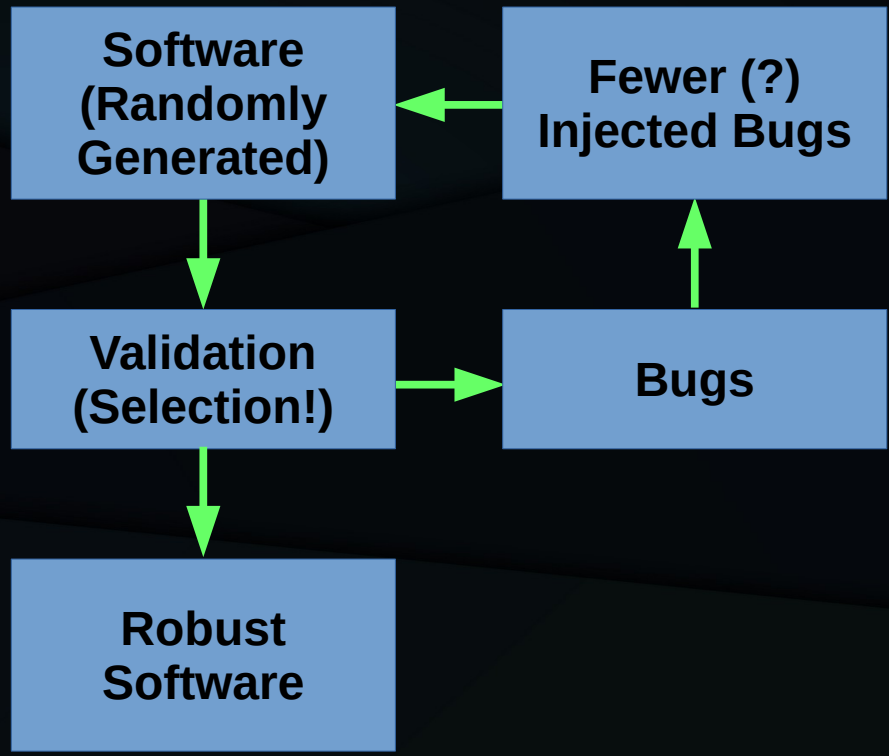
Natural Selection

Natural Selection



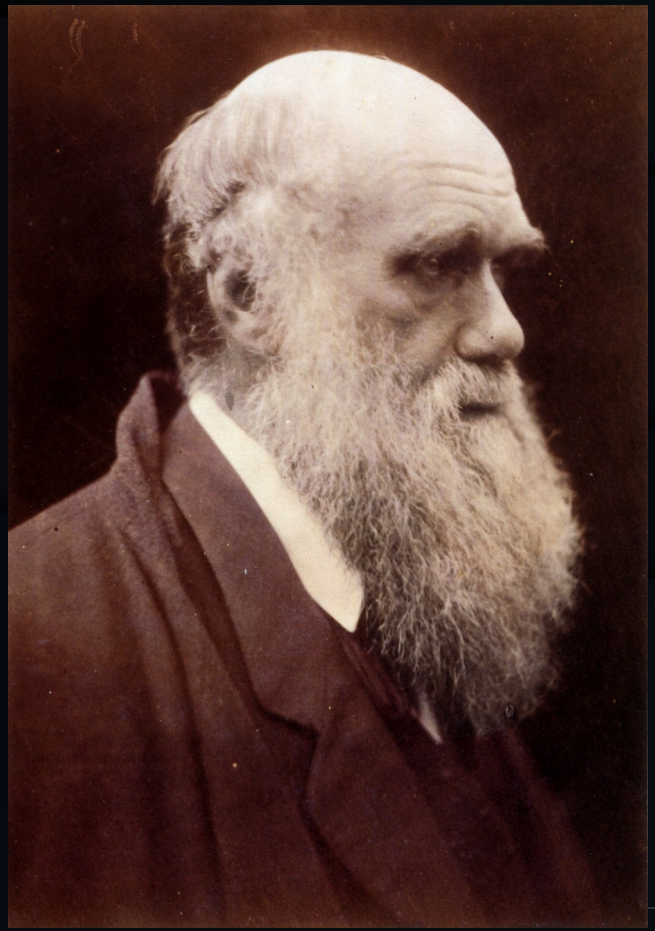
Natural Selection: Not Just Lifeforms

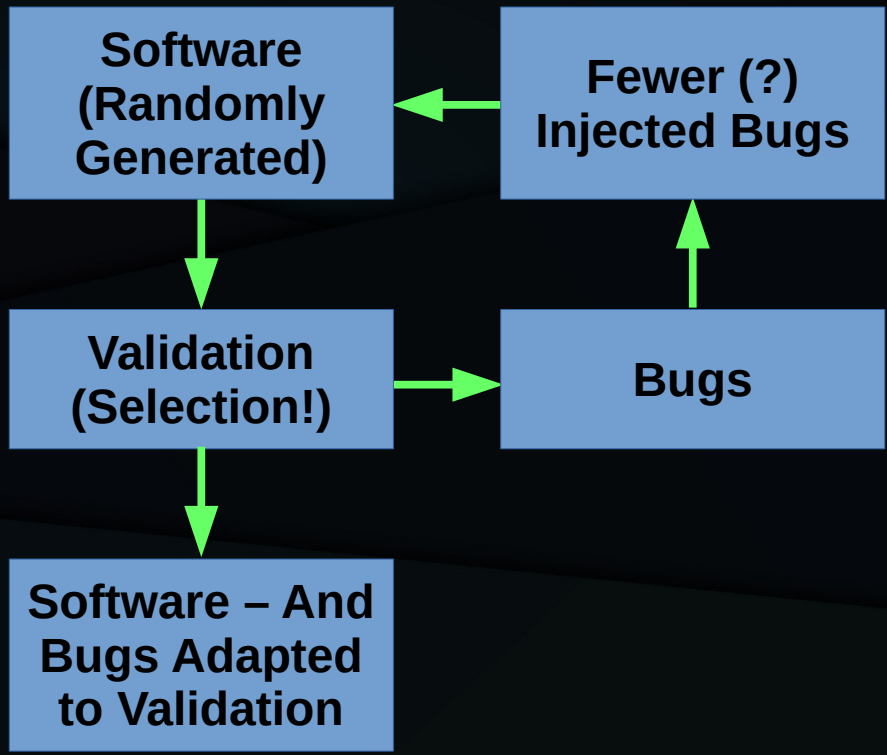




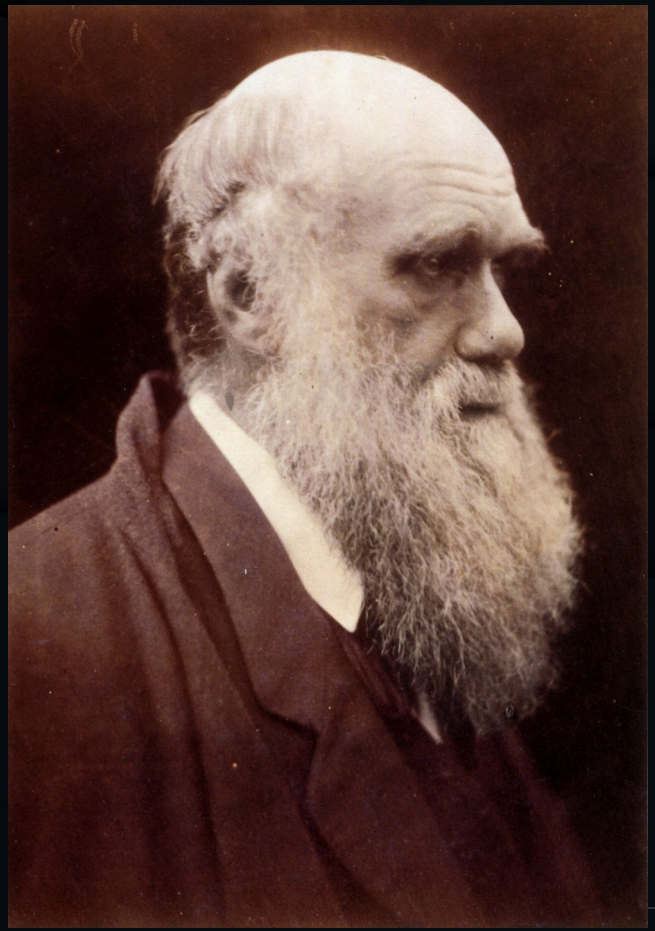
Agile methods attempt to push this methodology back to the specification

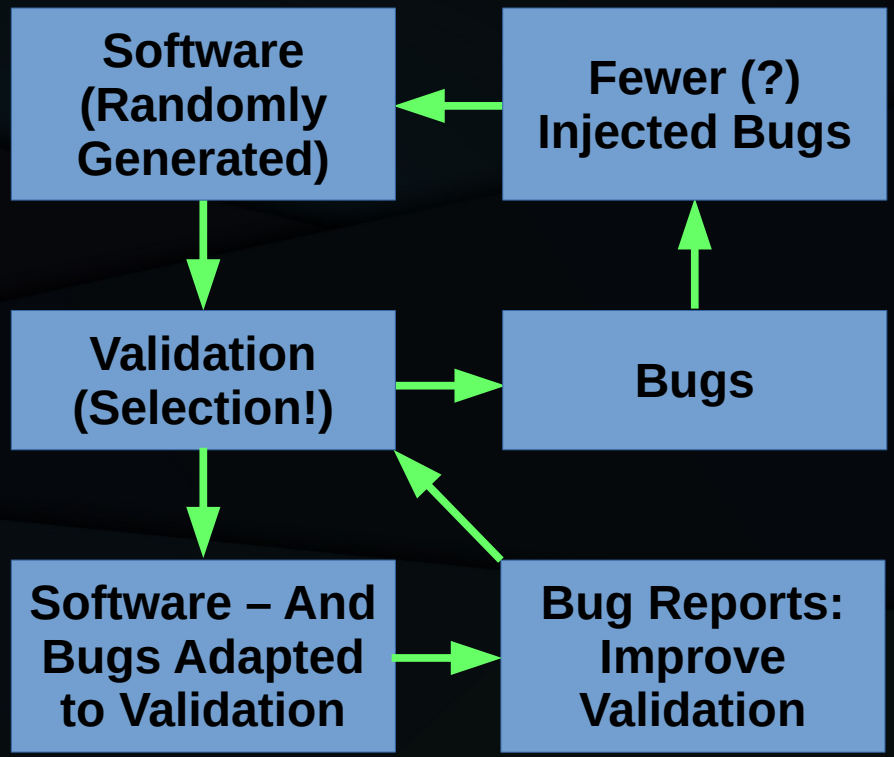
Natural Selection: Bugs are Software!



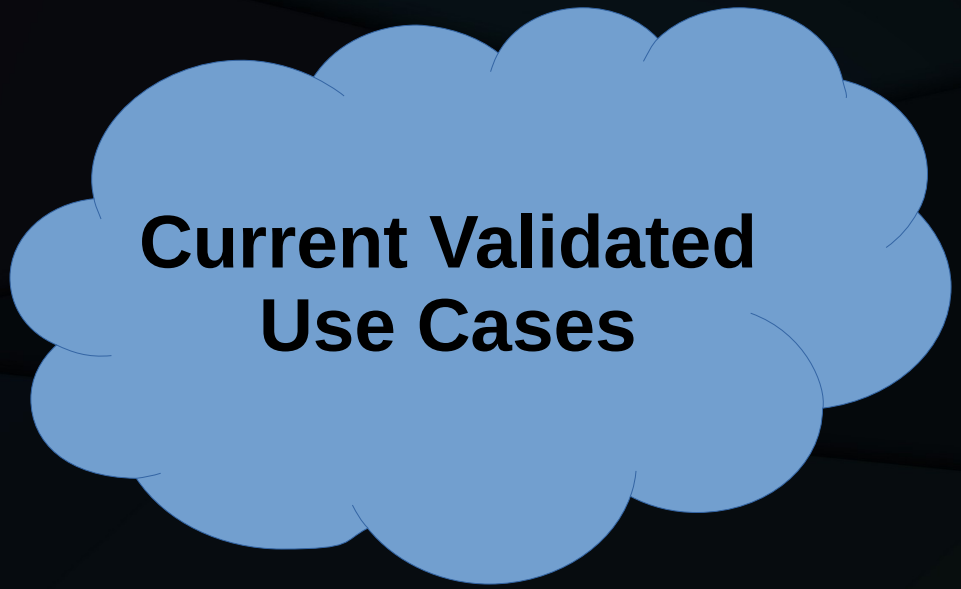


Natural Selection: Bugs are Software!



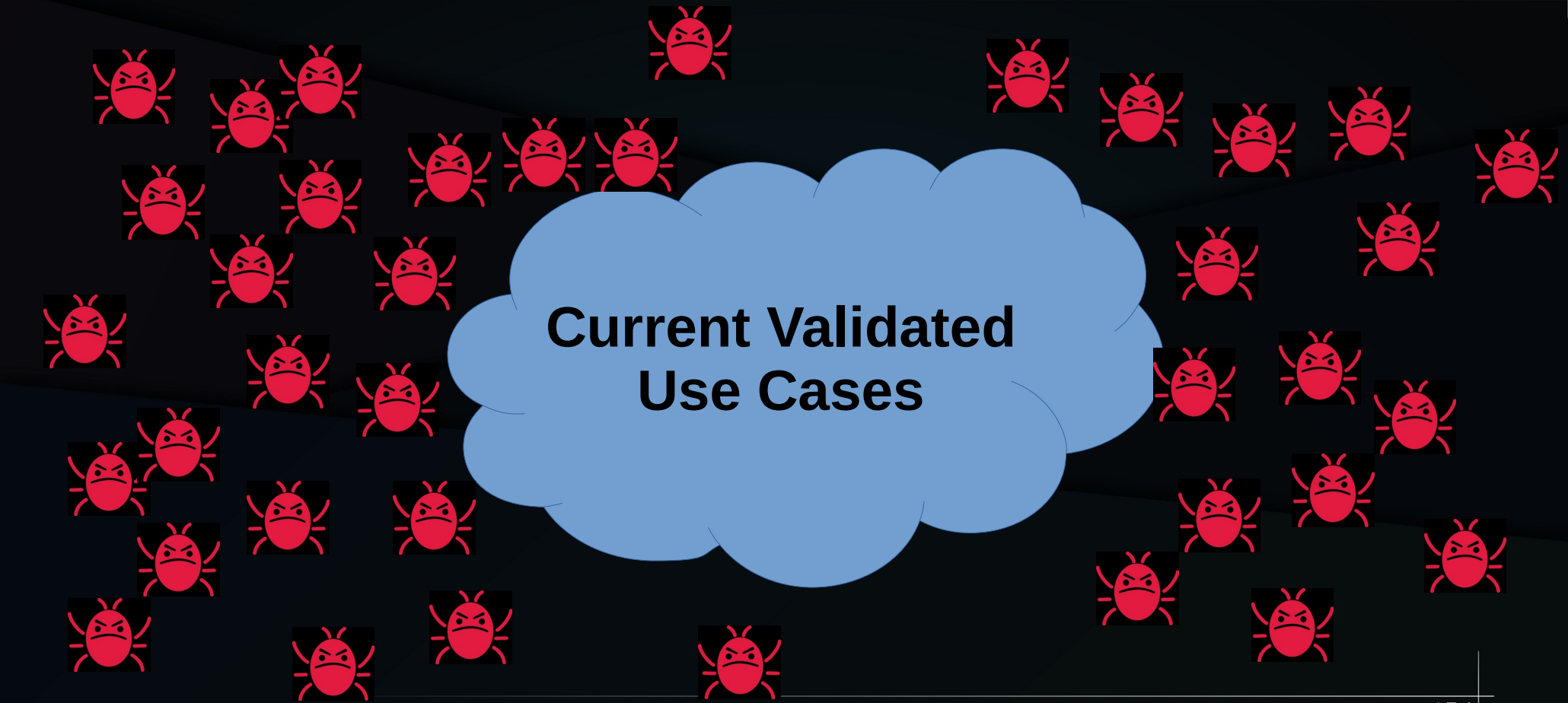


Validate Only Intended Use Cases

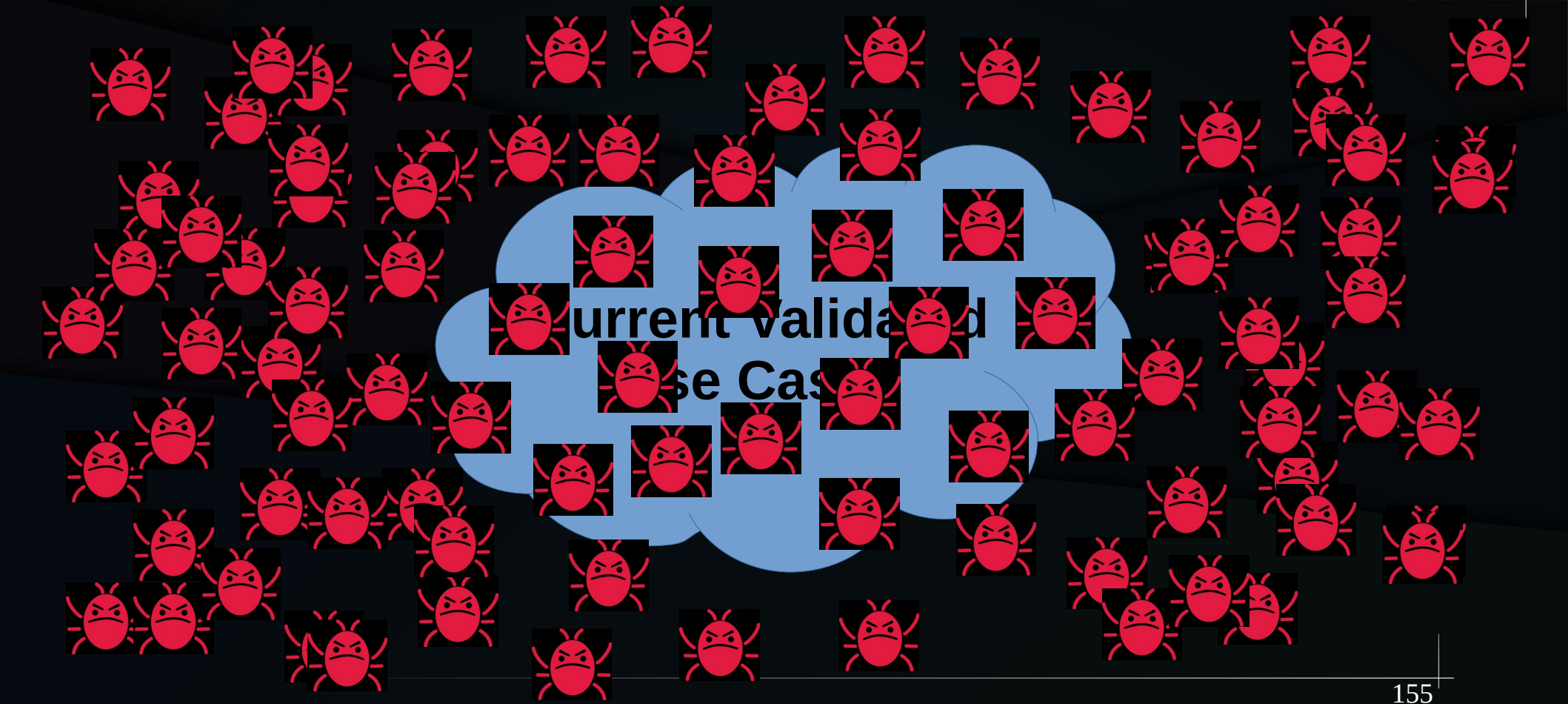


Major Development Generates Bug

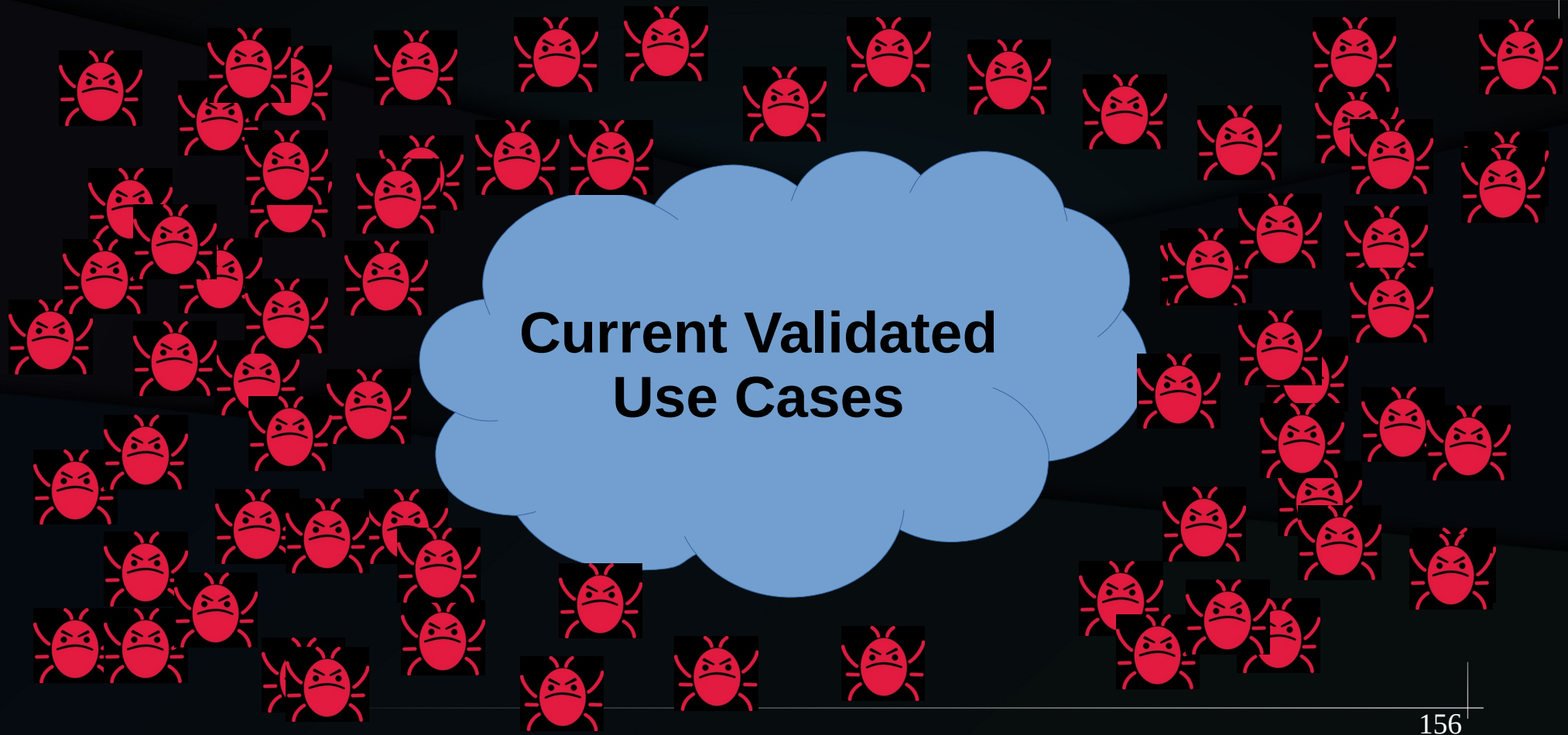
After Validation and Bug Fixing



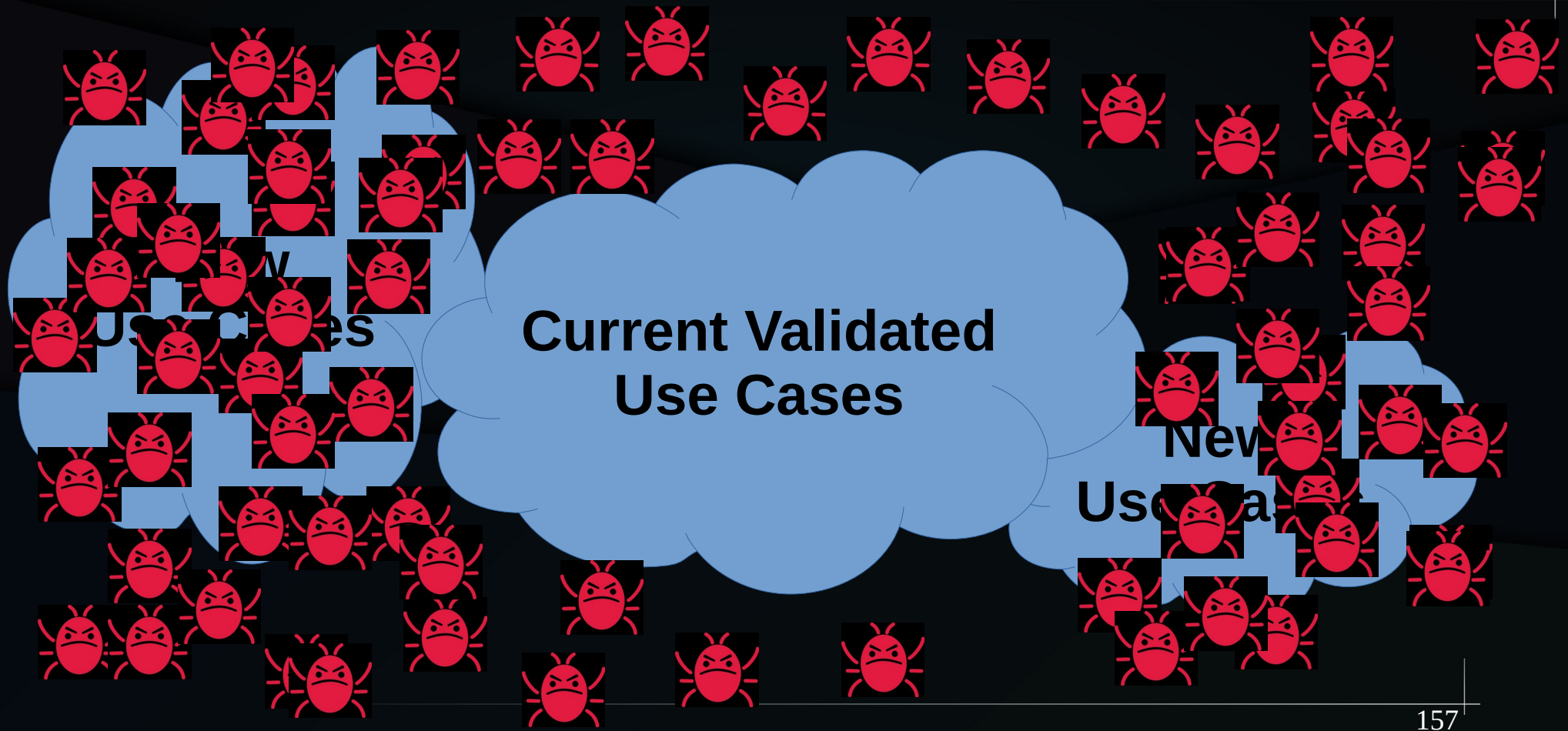
After Another Round of Development



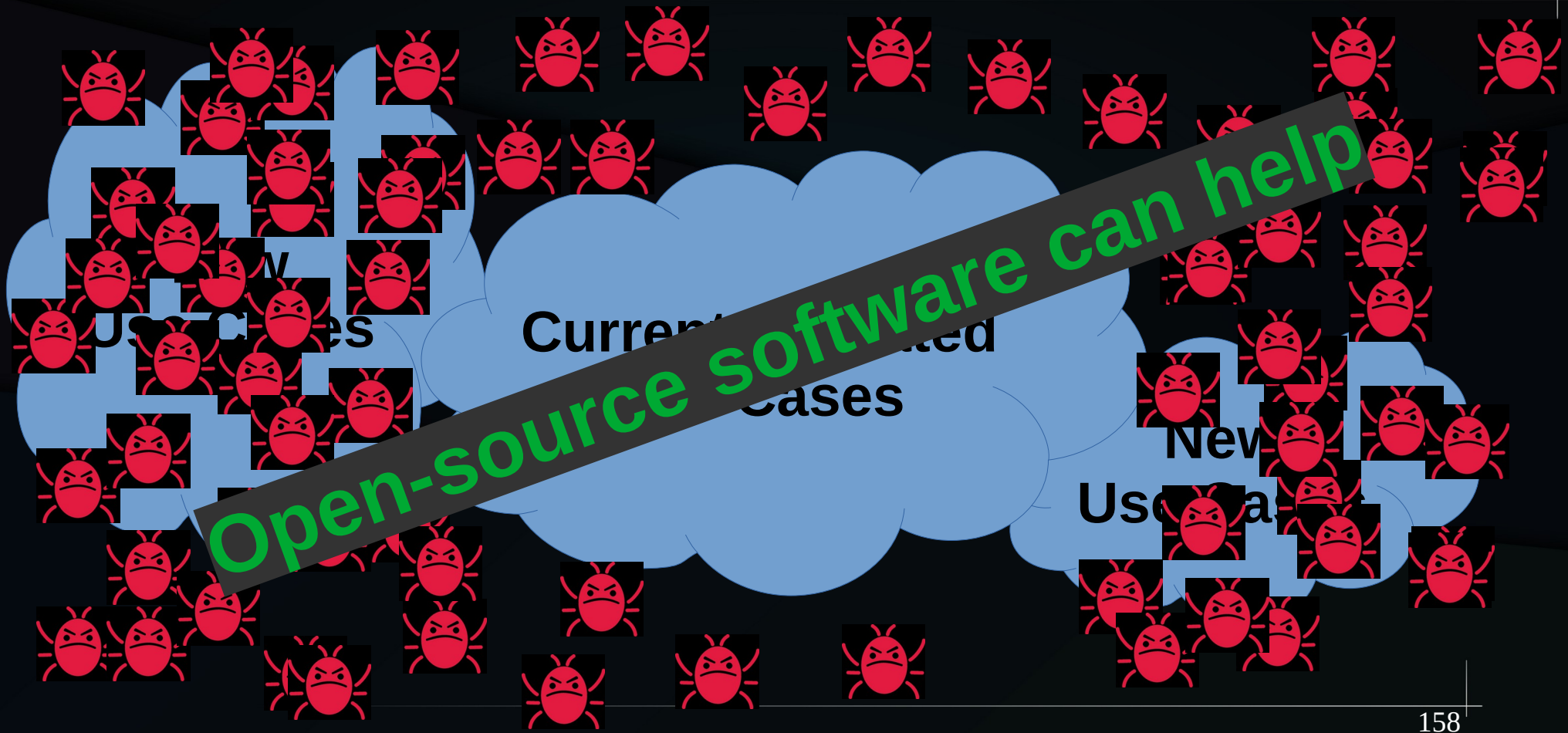
More Validation and Bug Fixing



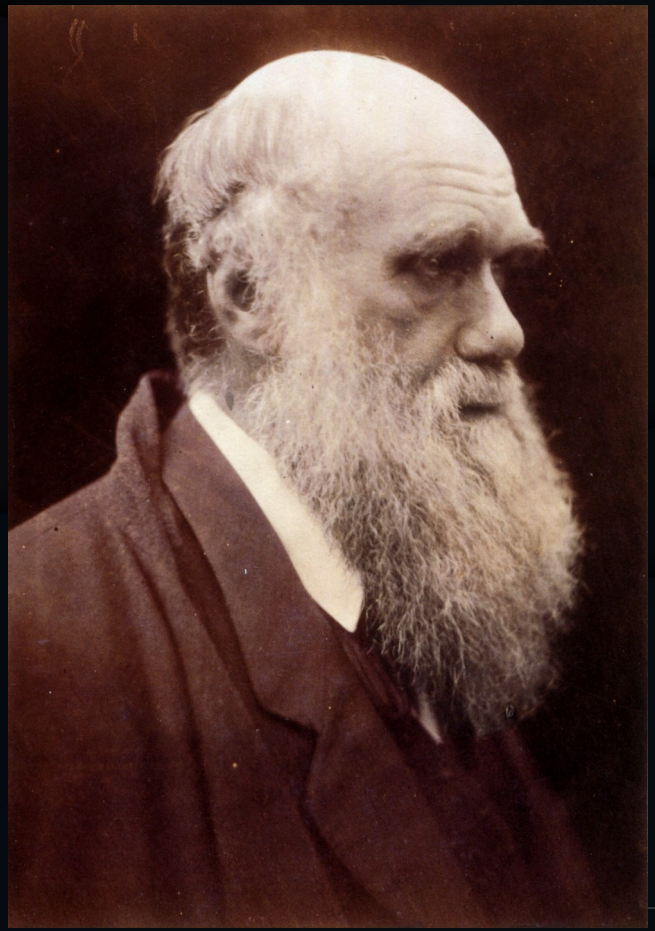
New Use Cases: Walls of Bugs!!!

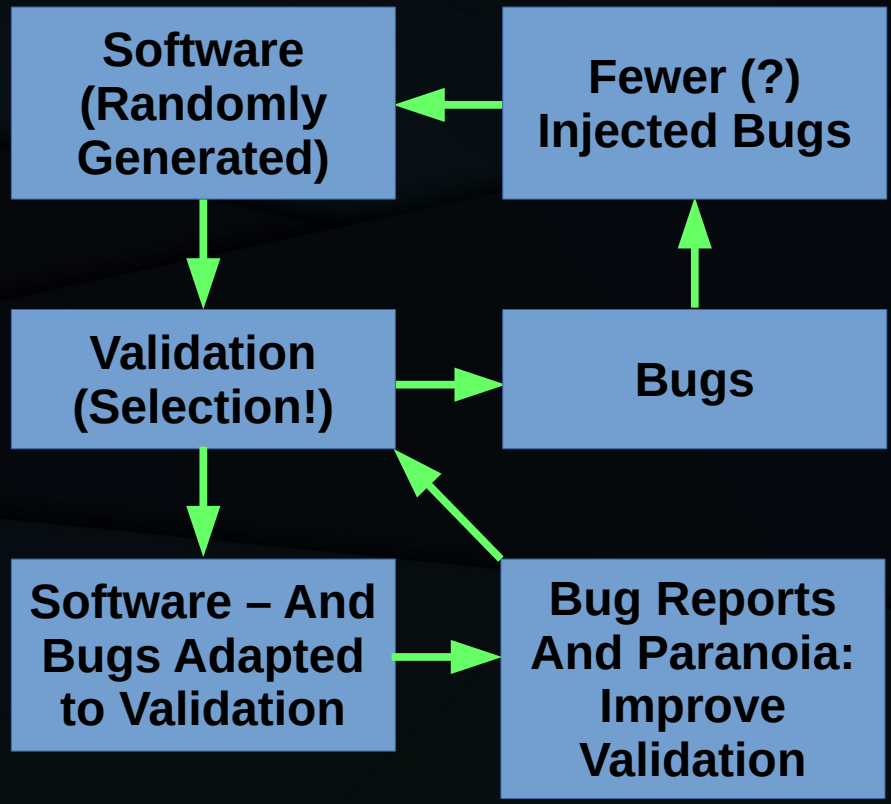


New Use Cases: Walls of Bugs!!!



Natural Selection: Bugs are Software!





“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 don’t know who to complain to
- The last N times they complained:
  - Nothing useful happened
  - They were yelled at or otherwise belittled
- Your software is 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 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*

You must live among your users

Cautionary Quotes

- Customers don’t know what they want [illegible] we’ve shown them. - *Steve Jobs*
- If there is any one secret [illegible] lies in the ability to get [illegible]’s point of view and see things from the other person’s angle as well as [illegible]. - *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 (200
  - Fast and scalable (up to 4096 CPUs)
  - Real-time response (sub-20
  - Energy efficiency
  - Near-bar
  -
  - Ease o (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 kernel concurrency and realtime (2001–)
  - Fast and scalable (up to 4096 CPUs)
  - Real-time response
  - Energy efficiency
  - Near-bare-metal
  - Ease of use
  - Ease of administration (large data centers)

My job is to provide reliable infrastructure

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

Questions?

Questions?

Photo NASA Public Domain April 7, 2021