C++ now

The Most Important API Design Guideline

No, It's Not That One

Jody Hagins

2024

C++ now

The Most Important API Design Guideline

No, It's Not That One

Jody Hagins

2024

MAYSTREET

C++Now 2024

The Most Important API Design Guideline

Jody Hagins
jhagins@dev.null

MAYSTREET

C++Now 2024

The Most Important API Design Guideline

Jody Hagins
jhagins@maystreet.com
coachhagins@gmail.com

C++Now 2024

The Most Important API Design Guideline

Where did this title come from?

Remember This?

NDC 2014

4

https://www.youtube.com/watch?v=a76Z__kQJdo&t=83s

The Most Important Design Guideline

NDC 2014

4

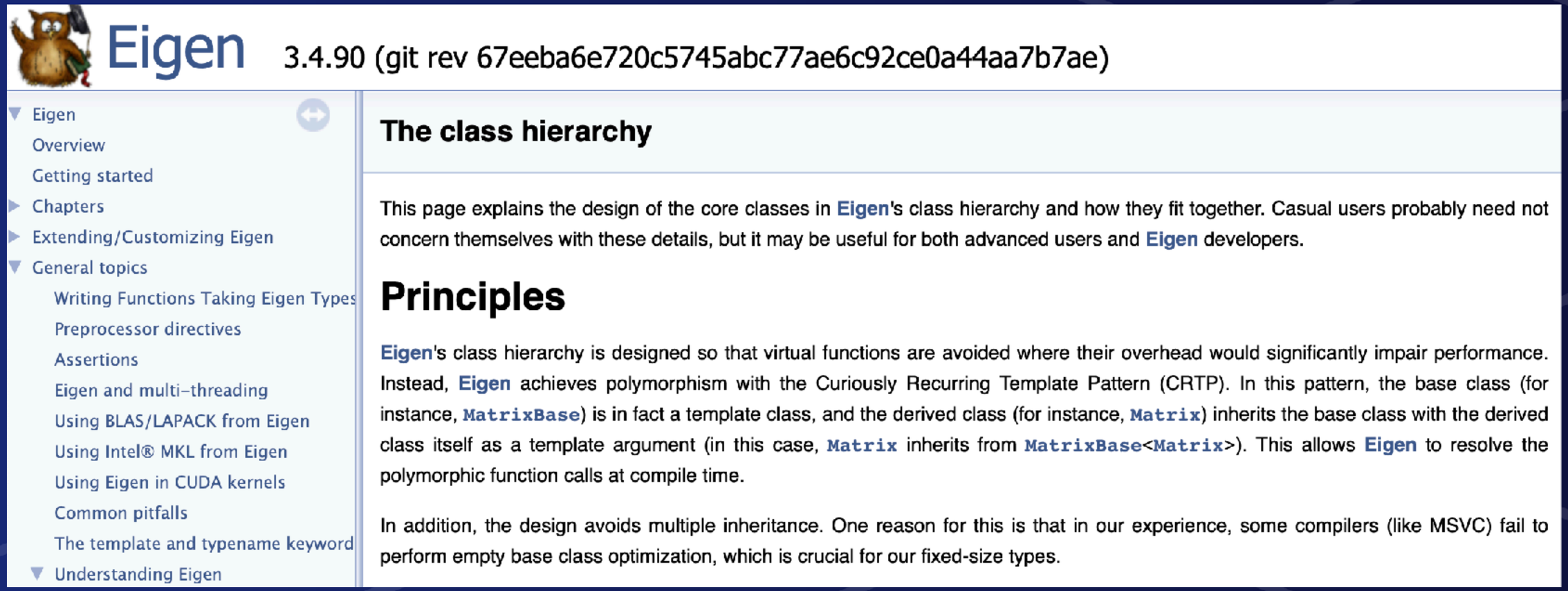
Make Interfaces Easy to Use Correctly and Hard to Use Incorrectly.

Scott Meyers, Software Development Consultant
http://www.aristeia.com/

Slide 2

https://www.youtube.com/watch?v=a76Z__kQJdo&t=83s

Design Guidelines for Eigen



Design Guidelines for Eigen

Eigen's class hierarchy is designed so that virtual functions are avoided where their overhead would significantly impair performance. Instead, **Eigen** achieves polymorphism with the Curiously Recurring Template Pattern (CRTP). In this pattern, the base class (for instance, **`MatrixBase`**) is in fact a template class, and the derived class (for instance, **`Matrix`**) inherits the base class with the derived class itself as a template argument (in this case, **`Matrix`** inherits from **`MatrixBase<Matrix>`**). This allows **Eigen** to resolve the polymorphic function calls at compile time.

In addition, the design avoids multiple inheritance. One reason for this is that in our experience, some compilers (like MSVC) fail to perform empty base class optimization, which is crucial for our fixed-size types.

The C++ Standard Library

- Don't be clever.
- Don't be stupid.
- Naming matters.
- Generic components should be aware of move-only types.
- We are thread-compatible [res.on.data.races].
- Exceptions are used for error conditions (there are some exceptions).
- Do not gratuitously overload operators.
- Classes allocating memory get an allocator (inconsistently applied).
- Containers get allocators unless they don't allocate.
- Containers use allocator through allocator traits.
- Allocators are part of type unless there is already type-erasure for other reasons.

https://github.com/cplusplus/LEWG/blob/archive/library-design-guidelines.md

The C++ Standard Library

- Don't be clever.
- Don't be stupid.
- Naming matters.
- Generic components should be aware of move-only types.
- We are thread-compatible [res.on.data.races].
- Exceptions are used for error conditions (there are some exceptions).
- Do not gratuitously overload operators.
- Classes allocating memory get an allocator (inconsistently applied).
- Containers get allocators unless they don't allocate.
- Containers use allocator through allocator traits.
- Allocators are part of type unless there is already type-erasure for other reasons.

https://github.com/cplusplus/LEWG/blob/archive/library-design-guidelines.md

The C++ Standard Library

- const-correctness is observed and is used as a proxy for thread-safety in the standard library.
- Class signatures want to be near minimal (the obvious counter-example is std::basic_string).
- Destructors shall not throw.
- Things should be constexpr where reasonable.
- Avoid inheritance and virtual functions where possible.
- Prefer function objects (i.e., deduced templates) to function pointers.
- Types should be allowed to be different rather than assuming they are same.
- Generic components should take advantage of parameters with stronger concepts.

https://github.com/cplusplus/LEWG/blob/archive/library-design-guidelines.md

The C++ Standard Library

- const-correctness is observed and is used as a proxy for thread-safety in the standard library.
- Class signatures want to be near minimal (the obvious counter-example is std::basic_string).
- Destructors shall not throw.
- Things should be constexpr where reasonable.
- Avoid inheritance and virtual functions where possible.
- Prefer function objects (i.e., deduced templates) to function pointers.
- Types should be allowed to be different rather than assuming they are same.
- Generic components should take advantage of parameters with stronger concepts.

https://github.com/cplusplus/LEWG/blob/archive/library-design-guidelines.md

The C++ Standard Library

- When designing a class type, where possible it should be a "regular type" (to be defined), e.g., different objects are independent.
- Use std::addressof() to obtain addresses based on generic parameters.
- Prefer to specify nested types as a typedef for an unspecified or implementation-defined type, rather than as a class or enumeration type. This avoids over-constraining implementations.
- Do not use requires clauses or define new language-level concepts until the introduction (section 6) and fundamental concepts (section 7) from the Ranges TS have merged into the IS.

https://github.com/cplusplus/LEWG/blob/archive/library-design-guidelines.md

Boost

Design and Programming

Aim first for clarity and correctness; optimization should be only a secondary concern in most Boost libraries.

Aim for ISO Standard C++. Than means making effective use of the standard features of the language, and avoiding non-standard compiler extensions. It also means using the C++ Standard Library where applicable.

Headers should be good neighbors. See the header policy. See Naming consistency.

Follow quality programming practices. See, for example, "Effective C++" 2nd Edition, and "More Effective C++", both by Scott Meyers, published by Addison Wesley.

Use the C++ Standard Library or other Boost libraries, but only when the benefits outweigh the costs. Do not use libraries other than the C++ Standard Library or Boost. See Library reuse.

Read Implementation Variation to see how to supply performance, platform, or other implementation variations.

https://www.boost.org/development/requirements.html

Boost

Browse through the Best Practices Handbook for ideas and links to source code in existing Boost libraries.

Read the guidelines for libraries with separate source to see how to ensure that compiled link libraries meet user expectations.

Use the naming conventions of the C++ Standard Library (See Naming conventions rationale):

- Names (except as noted below) should be all lowercase, with words separated by underscores.
- Acronyms should be treated as ordinary names (e.g. `xml_parser` instead of `XML_parser`).
- Template parameter names begin with an uppercase letter.
- Macro (gasp!) names all uppercase and begin with BOOST_.

https://www.boost.org/development/requirements.html

Boost

Choose meaningful names - explicit is better than implicit, and readability counts. There is a strong preference for clear and descriptive names, even if lengthy.

Use exceptions to report errors where appropriate, and write code that is safe in the face of exceptions.

Avoid exception-specifications. See exception-specification rationale.

Provide sample programs or confidence tests so potential users can see how to use your library.

Provide a regression test program or programs which follow the Test Policies and Protocols.

https://www.boost.org/development/requirements.html

Boost

Although some boost members use proportional fonts, tabs, and unrestricted line lengths in their own code, boost's widely distributed source code should follow more conservative guidelines:

- Use fixed-width fonts. See fonts rationale.
- Use spaces rather than tabs. See tabs rationale.
- Limit line lengths to 80 characters.

End all documentation files (HTML or otherwise) with a copyright message and a licensing message. See the license information page for the preferred form.

https://www.boost.org/development/requirements.html

Boost

Begin all source files (including programs, headers, scripts, etc.) with:

- A comment line describing the contents of the file.
- Comments describing copyright and licensing: again, the preferred form is indicated i the license information page
- Note that developers are allowed to provide a copy of the license text in `LICENSE_1_0.txt`, `LICENSE.txt` or `LICENSE` file within repositories of their libraries.
- A comment line referencing your library on the Boost web site. For example:

```
// See https://www.boost.org/libs/foo for library home page.
```

Where `foo` is the directory name (see below) for the library. As well as aiding users who come across a Boost file detached from its documentation, some of Boost's automatic tools depend on this comment to identify which library header files belong to.

https://www.boost.org/development/requirements.html

The Most Important API Design Guideline?

NDC 2014

Make Interfaces Easy to Use Correctly and Hard to Use Incorrectly.

Scott Meyers, Software Development Consultant
http://www.aristeia.com/

Slide 2

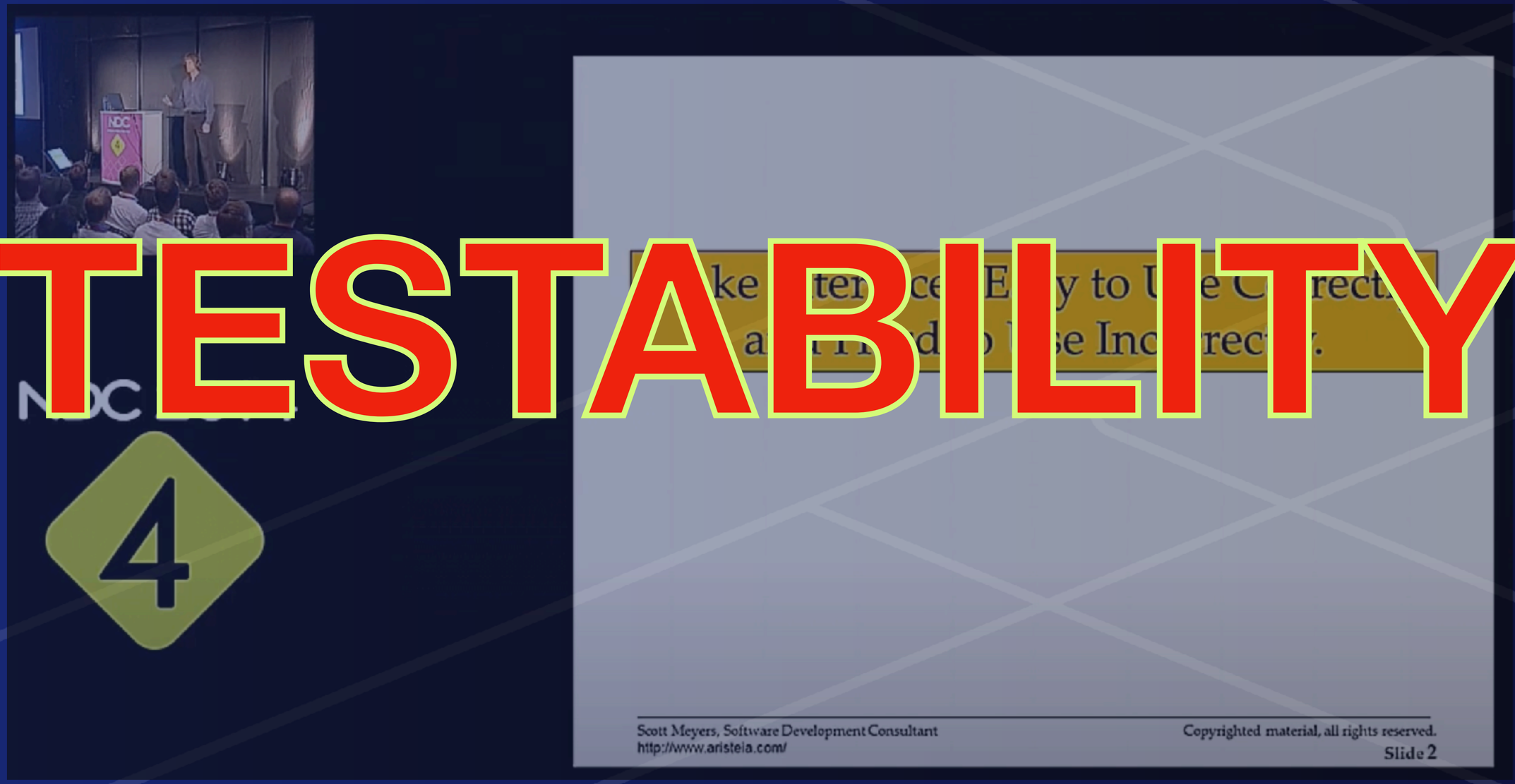
https://www.youtube.com/watch?v=a76Z__kQJdo&t=83s

The Hinnant Rule

compiler implicitly declares

user declares	default constructor	destructor	copy constructor	copy assignment	move constructor	move assignment
Nothing	defaulted	defaulted	defaulted	defaulted	defaulted	defaulted
Any constructor	not declared	defaulted	defaulted	defaulted	defaulted	defaulted
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	defaulted
destructor	defaulted	user declared	defaulted	defaulted	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted	not declared	not declared
copy assignment	defaulted	defaulted	defaulted	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	defaulted	deleted	deleted	not declared	user declared

The Most Important API Design Guideline?

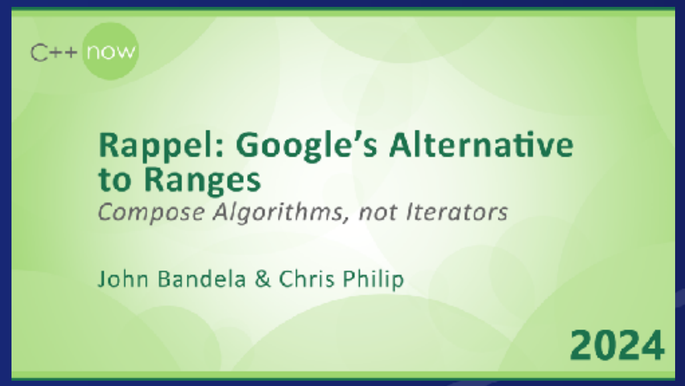


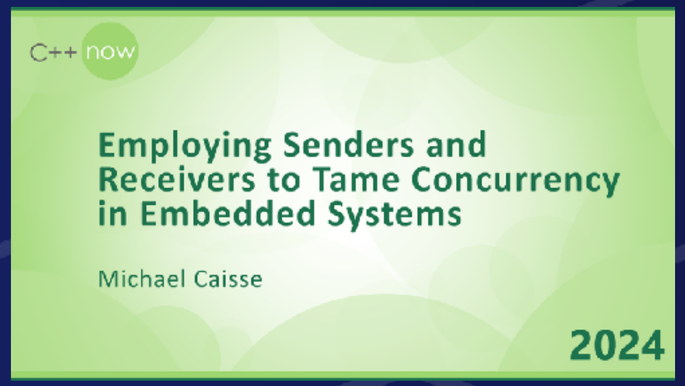
https://www.youtube.com/watch?v=a76Z__kQJdo&t=83s

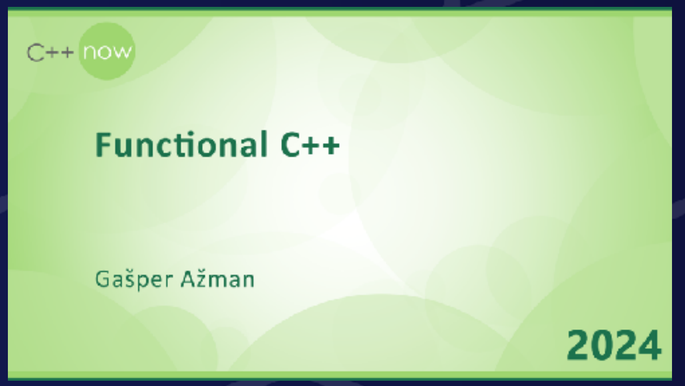
The Most Important API Design Guideline is Testability

The Most Important API Design Guideline is Testability

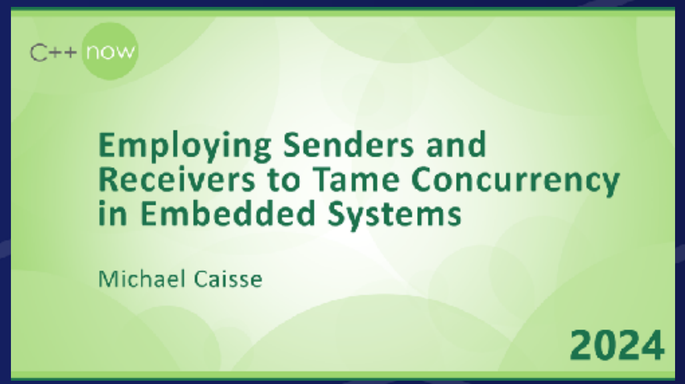
The Most Important API Design Guideline is Testability

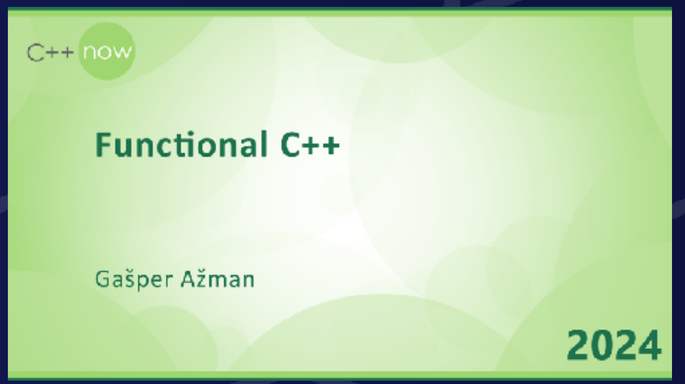


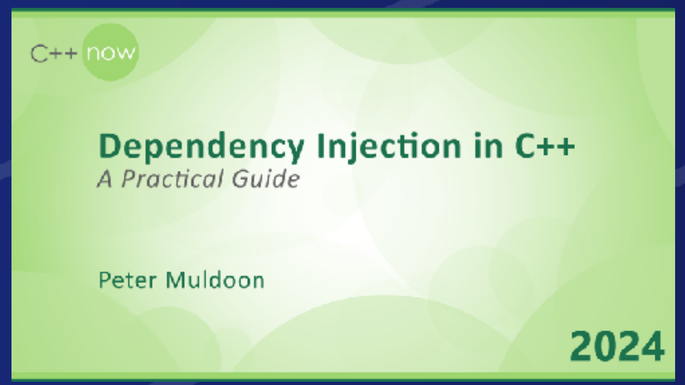


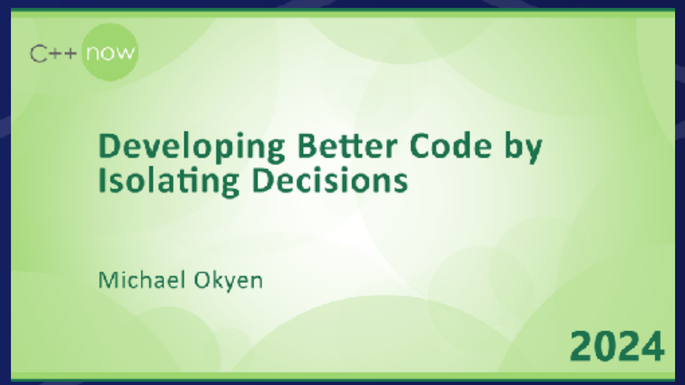


The Most Important API Design Guideline is Testability









The Most Important API Design Guideline is Testability

Why is Testing Important?

- We are human!

Testability

- Unit Tests

Testability

- Unit Tests

Interlude

Testability

- Unit Tests

Testability

```
procedure insert2 (integer x, ℓ)
begin B[ℓ] ← B[ℓ] ∨ (2 ↑ (x mod 16));
    size[ℓ] ← size[ℓ]+1;
    if x < least[ℓ] then least[ℓ] ← x
    else if x > greatest[ℓ] then greatest[ℓ] ← x;
    end;
```

The implementation of deletion would be similar. It is safe to use 0 and $2^{16}-1$ for $-\infty$ and $+\infty$.

Beware of bugs in the above code; I have only proved it correct, not tried it.

https://staff.fnwi.uva.nl/p.vanemdeboas/knuthnote.pdf

Testability or Provability?

```
procedure insert2 (integer x, ℓ)
begin B[ℓ] ← B[ℓ] ∨ (2 ↑ (x mod 16));
    size[ℓ] ← size[ℓ]+1;
    if x < least[ℓ] then least[ℓ] ← x
    else if x > greatest[ℓ] then greatest[ℓ] ← x;
    end;
```

The implementation of deletion would be similar. It is safe to use 0 and $2^{16}-1$ for $-\infty$ and $+\infty$.

Beware of bugs in the above code; I have only proved it correct, not tried it.

https://staff.fnwi.uva.nl/p.vanemdeboas/knuthnote.pdf

Testability

- Unit Tests

Knuth on Unit Tests

In that vein, today’s developers frequently build programs writing small code increments followed by immediate compilation and the creation and running of unit tests. What are your thoughts on this approach to software development?

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on Unit Tests

In that vein, today’s developers frequently build programs writing small code increments followed by immediate compilation and the creation and running of unit tests. What are your thoughts on this approach to software development?

As to your real question, the idea of immediate compilation and "unit tests" appeals to me as the ultimate silver bullet. I've been waiting my entire career for such an advancement in computer programming methodology.

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on Unit Tests

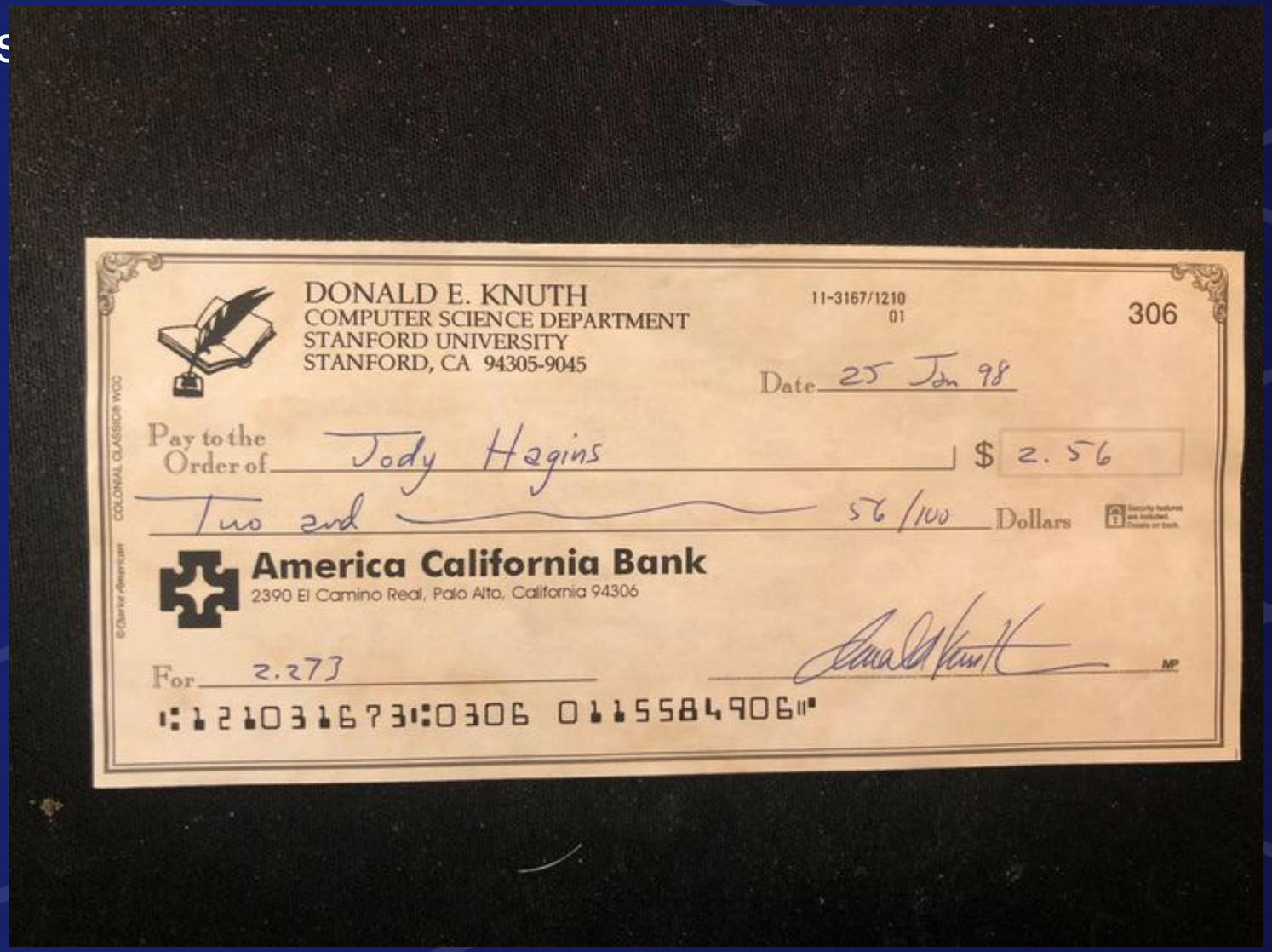
In that vein, today’s developers frequently build programs writing small code increments followed by immediate compilation and the creation and running of unit tests. What are your thoughts on this approach to software development?

As to your real question, the idea of immediate compilation and "unit tests" appeals to me only rarely, when I’m feeling my way in a totally unknown environment and need feedback about what works and what doesn’t. Otherwise, lots of time is wasted on activities that I simply never need to perform or even think about. Nothing needs to be "mocked up."

https://www.informit.com/articles/article.aspx?p=1193856

Testability

- Unit Tests



Knuth on Unit Tests

In that vein, today’s developers frequently build programs writing small code increments followed by immediate compilation and the creation and running of unit tests. What are your thoughts on this approach to software development?

As to your real question, the idea of immediate compilation and "unit tests" appeals to me only rarely, when I’m feeling my way in a totally unknown environment and need feedback about what works and what doesn’t. Otherwise, lots of time is wasted on activities that I simply never need to perform or even think about. Nothing needs to be "mocked up."

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on ...

Can you give some examples that are currently in vogue, which developers shouldn’t adopt simply because they’re currently popular or because that’s the way they’re currently done? Would you care to identify important examples of this outside of software development?

https://www.informit.com/articles/article.aspx?p=1193856

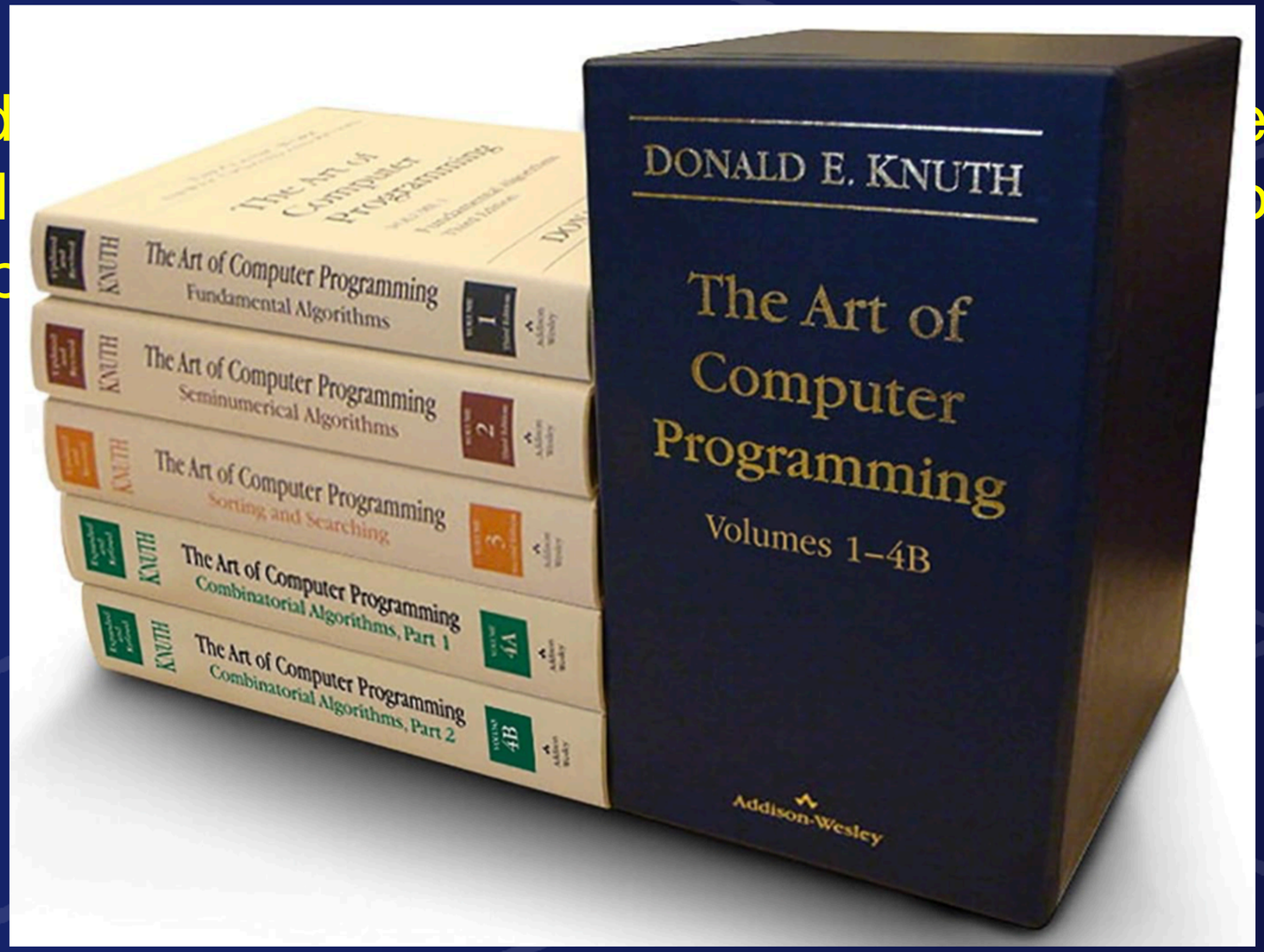
Knuth on ...

I hate to duck your questions even though I also hate to offend other people's sensibilities—given that software methodology has always been akin to religion.

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on ...

I hate to d e to offend other peopl odology has always b



https://www.informit.com/articles/article.aspx?p=1193856

Knuth on ...

I hate to duck your questions even though I also hate to offend other people's sensibilities—given that software methodology has always been akin to religion. With the caveat that there's no reason anybody should care about the opinions of a computer scientist/mathematician like me regarding software development,

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on Extreme Programming

I hate to duck your questions even though I also hate to offend other people's sensibilities—given that software methodology has always been akin to religion. With the caveat that there's no reason anybody should care about the opinions of a computer scientist/mathematician like me regarding software development, let me just say that almost everything I've ever heard associated with the term "extreme programming" sounds like exactly the wrong way to go...with one exception. The exception is the idea of working in teams and reading each other's code. That idea is crucial, and it might even mask out all the terrible aspects of extreme programming that alarm me.

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on Extreme Programming and Code Reuse

I also must confess to a strong bias against the fashion for reusable code. To me, "re-editable code" is much, much better than an untouchable black box or toolkit. I could go on and on about this. If you're totally convinced that reusable code is wonderful, I probably won't be able to sway you anyway, but you'll never convince me that reusable code isn't mostly a menace.

https://www.informit.com/articles/article.aspx?p=1193856

Knuth on His Testing Methodology

I generally get best results by writing a test program that no sane user would ever think of writing. My test programs are intended to break the system, to push it to its extreme limits, to pile complication on complication, in ways that the system programmer never consciously anticipated. To prepare such test data, I get into the meanest, nastiest frame of mind that I can manage, and I write the cruelest code I can think of; then I turn around and embed that in even nastier constructions that are almost obscene.

Donald Knuth, *The Errors of TEX* (1989)

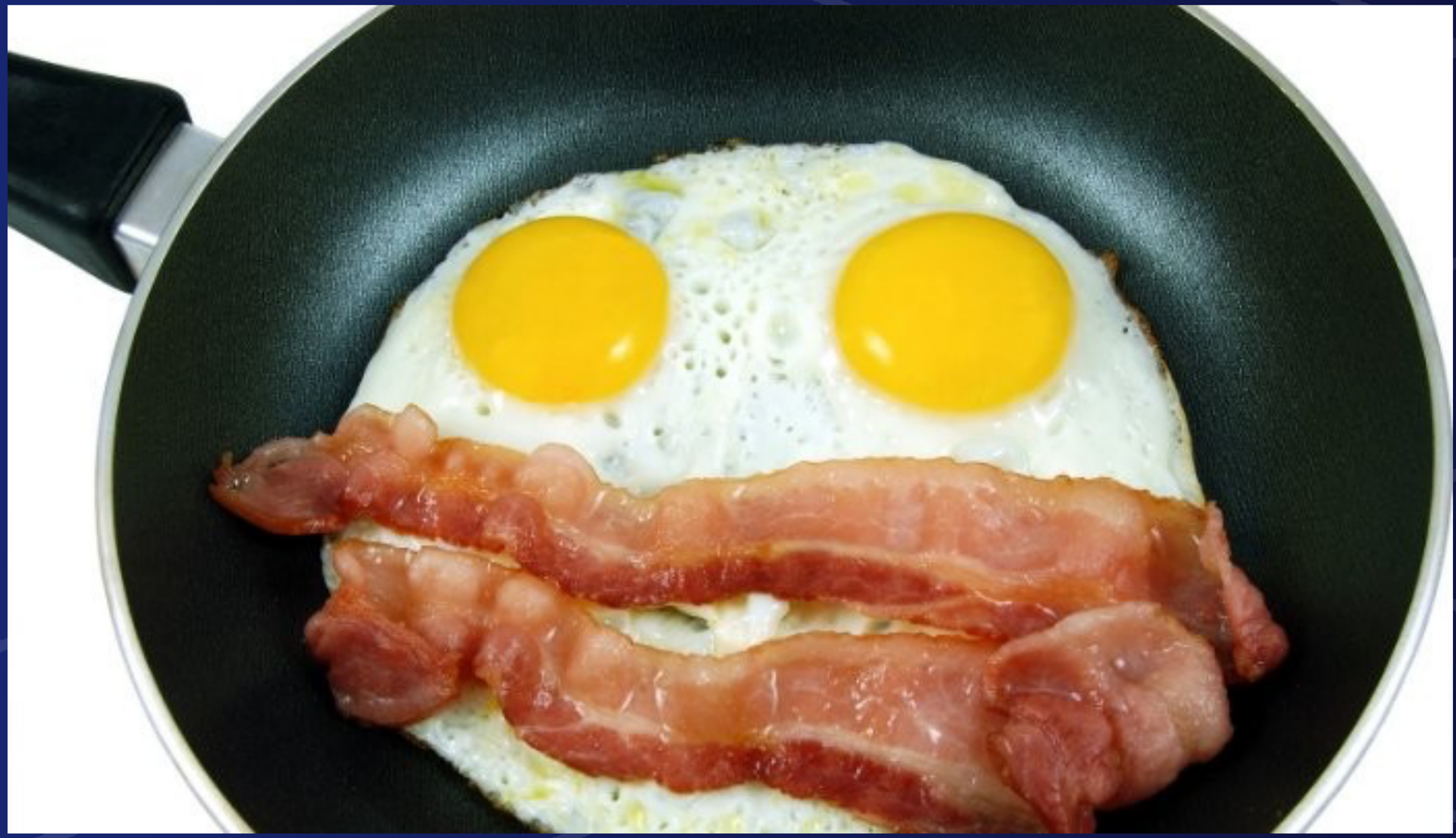
What is the purpose of testing software?

How do you test your software?

How do you test your software?

https://en.wikipedia.org/wiki/Cruise_missile

Commitment to testing your software



https://www.linkedin.com/pulse/involved-vs-committed-you-pig-forrest

How do you test your software?

```
Blarg do_something(std::int64_t price, std::uint64_t quantity);
```

How do you test your software?

```
Blarg do_something(std::int64_t price, std::uint64_t quantity);
```

How do you test your software?

```
Blarg do_something(std::int64_t price, std::uint64_t quantity);
```

Use the Compiler

```
Blarg do_something(Price price, Quantity quantity);
```

Use the Compiler

```
Blarg do_something(Price price, Quantity quantity);
```

No Naked Types

Use the Compiler

```
Blarg do_something(Price price, Quantity quantity);

Blarg do_something(Price, Quantity);
```

Use the Compiler

```
Blarg do_something(
    Price bid_price,
    Quantity bid_quantity,
    Price ask_price,
    Quantity ask_quantity);
```

Use the Compiler

```
Blarg do_something(
    BidPrice bid_price,
    BidQuantity bid_quantity,
    AskPrice ask_price,
    AskQuantity ask_quantity);
```

```
Blarg do_something(
    BidPrice,
    BidQuantity,
    AskPrice,
    AskQuantity);
```

Easy Strong Types???

```
struct [[clang::annotate("strong(int)")]] Flip;
struct [[clang::annotate("strong(int,+,-,->")]] Flop;
```

What do you test? What is an API?

https://www.youtube.com/watch?v=Y9clBHENy4Q

goto;

Knight Capital

Unrestricted - Public

SUBSCRIBE;

Programming's Greatest Mistakes • Mark Rendle • GOTO 2023

Knight Capital Cease and Desist

SECURITIES EXCHANGE ACT OF 1934
Release No. 70694 / October 16, 2013

ADMINISTRATIVE PROCEEDING
File No. 3-15570

In the Matter of

Knight Capital Americas LLC

Respondent.

ORDER INSTITUTING ADMINISTRATIVE AND CEASE-AND-DESIST PROCEEDINGS, PURSUANT TO SECTIONS 15(b) AND 21C OF THE SECURITIES EXCHANGE ACT OF 1934, MAKING FINDINGS, AND IMPOSING REMEDIAL SANCTIONS AND A CEASE-AND-DESIST ORDER

Summary

1. On August 1, 2012, Knight Capital Americas LLC ("Knight") experienced a significant error in the operation of its automated routing system for equity orders, known as SMARS. While processing 212 small retail orders that Knight had received from its customers, SMARS routed millions of orders into the market over a 45-minute period, and obtained over 4 million executions in 154 stocks for more than 397 million shares. By the time that Knight stopped sending the orders, Knight had assumed a net long position in 80 stocks of approximately $3.5 billion and a net short position in 74 stocks of approximately $3.15 billion. Ultimately, Knight lost over $460 million from these unwanted positions. The subject of these proceedings is Knight's violation of a Commission rule that requires brokers or dealers to have controls and procedures in place reasonably designed to limit the risks associated with their access to the markets, including the risks associated with automated systems and the possibility of these types of errors.

Retail Liquidity Program

12. To enable its customers' participation in the Retail Liquidity Program ("RLP") at the New York Stock Exchange,[5] which was scheduled to commence on August 1, 2012, Knight made a number of changes to its systems and software code related to its order handling processes. These changes included developing and deploying new software code in SMARS. SMARS is an automated, high speed, algorithmic router that sends orders into the market for execution. A core function of SMARS is to receive orders passed from other components of Knight's trading platform ("parent" orders) and then, as needed based on the available liquidity, send one or more representative (or "child") orders to external venues for execution.

General Idea: Replace Power Peg with RLP

13. Upon deployment, the new RLP code in SMARS was intended to replace unused code in the relevant portion of the order router. This unused code previously had been used for functionality called "Power Peg," which Knight had discontinued using many years earlier. Despite the lack of use, the Power Peg functionality remained present and callable at the time of the RLP deployment. The new RLP code also repurposed a flag that was formerly used to activate the Power Peg code. Knight intended to delete the Power Peg code so that when this flag was set to "yes," the new RLP functionality—rather than Power Peg—would be engaged.

General Idea: Replace Power Peg with RLP

13. Upon deployment, the new RLP code in SMARS was intended to replace unused code in the relevant portion of the order router. This unused code previously had been used for functionality called "Power Peg," which Knight had discontinued using many years earlier. Despite the lack of use, the Power Peg functionality remained present and callable at the time of the RLP deployment. The new RLP code also repurposed a flag that was formerly used to activate the Power Peg code. Knight intended to delete the Power Peg code so that when this flag was set to "yes," the new RLP functionality—rather than Power Peg—would be engaged.

General Idea: Replace Power Peg with RLP

13. Upon deployment, the new RLP code in SMARS was intended to replace unused code in the relevant portion of the order router. This unused code previously had been used for functionality called "Power Peg," which Knight had discontinued using many years earlier. Despite the lack of use, the Power Peg functionality remained present and callable at the time of the RLP deployment. The new RLP code also repurposed a flag that was formerly used to activate the Power Peg code. Knight intended to delete the Power Peg code so that when this flag was set to "yes," the new RLP functionality—rather than Power Peg—would be engaged.

Power Peg Routes Orders Until Filled

14. When Knight used the Power Peg code previously, as child orders were executed, a cumulative quantity function counted the number of shares of the parent order that had been executed. This feature instructed the code to stop routing child orders after the parent order had been filled completely. In 2003, Knight ceased using the Power Peg functionality. In 2005, Knight moved the tracking of cumulative shares function in the Power Peg code to an earlier point in the SMARS code sequence. Knight did not retest the Power Peg code after moving the cumulative quantity function to determine whether Power Peg would still function correctly if called.

Power Peg Value No Longer Used
Source Not Removed

14. When Knight used the Power Peg code previously, as child orders were executed, a cumulative quantity function counted the number of shares of the parent order that had been executed. This feature instructed the code to stop routing child orders after the parent order had been filled completely. In 2003, Knight ceased using the Power Peg functionality. In 2005, Knight moved the tracking of cumulative shares function in the Power Peg code to an earlier point in the SMARS code sequence. Knight did not retest the Power Peg code after moving the cumulative quantity function to determine whether Power Peg would still function correctly if called.

Rewrite; Power Peg Moved, Not Removed

14. When Knight used the Power Peg code previously, as child orders were executed, a cumulative quantity function counted the number of shares of the parent order that had been executed. This feature instructed the code to stop routing child orders after the parent order had been filled completely. In 2003, Knight ceased using the Power Peg functionality. In 2005, Knight moved the tracking of cumulative shares function in the Power Peg code to an earlier point in the SMARS code sequence. Knight did not retest the Power Peg code after moving the cumulative quantity function to determine whether Power Peg would still function correctly if called.

Did Not Retest Power Peg

14. When Knight used the Power Peg code previously, as child orders were executed, a cumulative quantity function counted the number of shares of the parent order that had been executed. This feature instructed the code to stop routing child orders after the parent order had been filled completely. In 2003, Knight ceased using the Power Peg functionality. In 2005, Knight moved the tracking of cumulative shares function in the Power Peg code to an earlier point in the SMARS code sequence. Knight did not retest the Power Peg code after moving the cumulative quantity function to determine whether Power Peg would still function correctly if called.

Deploy Version to Support RLP

15. Beginning on July 27, 2012, Knight deployed the new RLP code in SMARS in stages by placing it on a limited number of servers in SMARS on successive days. During the deployment of the new code, however, one of Knight's technicians did not copy the new code to one of the eight SMARS computer servers. Knight did not have a second technician review this deployment and no one at Knight realized that the Power Peg code had not been removed from the eighth server, nor the new RLP code added. Knight had no written procedures that required such a review.

Manual Deployment Mistake

15. Beginning on July 27, 2012, Knight deployed the new RLP code in SMARS in stages by placing it on a limited number of servers in SMARS on successive days. During the deployment of the new code, however, one of Knight's technicians did not copy the new code to one of the eight SMARS computer servers. Knight did not have a second technician review this deployment and no one at Knight realized that the Power Peg code had not been removed from the eighth server, nor the new RLP code added. Knight had no written procedures that required such a review.

No Review or Test of Deployment

15. Beginning on July 27, 2012, Knight deployed the new RLP code in SMARS in stages by placing it on a limited number of servers in SMARS on successive days. During the deployment of the new code, however, one of Knight's technicians did not copy the new code to one of the eight SMARS computer servers. Knight did not have a second technician review this deployment and no one at Knight realized that the Power Peg code had not been removed from the eighth server, nor the new RLP code added. Knight had no written procedures that required such a review.

RLP Goes Live on 7 Servers

16. On August 1, Knight received orders from broker-dealers whose customers were eligible to participate in the RLP. The seven servers that received the new code processed these orders correctly. However, orders sent with the repurposed flag to the eighth server triggered the defective Power Peg code still present on that server. As a result, this server began sending child orders to certain trading centers for execution. Because the cumulative quantity function had been moved, this server continuously sent child orders, in rapid sequence, for each incoming parent order without regard to the number of share executions Knight had already received from trading centers. Although one part of Knight's order handling system recognized that the parent orders had been filled, this information was not communicated to SMARS.

Power Peg Runs on 1 Server

16. On August 1, Knight received orders from broker-dealers whose customers were eligible to participate in the RLP. The seven servers that received the new code processed these orders correctly. However, orders sent with the repurposed flag to the eighth server triggered the defective Power Peg code still present on that server. As a result, this server began sending child orders to certain trading centers for execution. Because the cumulative quantity function had been moved, this server continuously sent child orders, in rapid sequence, for each incoming parent order without regard to the number of share executions Knight had already received from trading centers. Although one part of Knight's order handling system recognized that the parent orders had been filled, this information was not communicated to SMARS.

2005 Bites Back

16. On August 1, Knight received orders from broker-dealers whose customers were eligible to participate in the RLP. The seven servers that received the new code processed these orders correctly. However, orders sent with the repurposed flag to the eighth server triggered the defective Power Peg code still present on that server. As a result, this server began sending child orders to certain trading centers for execution. Because the cumulative quantity function had been moved, this server continuously sent child orders, in rapid sequence, for each incoming parent order without regard to the number of share executions Knight had already received from trading centers. Although one part of Knight's order handling system recognized that the parent orders had been filled, this information was not communicated to SMARS.

Why Should I Tell You?

16. On August 1, Knight received orders from broker-dealers whose customers were eligible to participate in the RLP. The seven servers that received the new code processed these orders correctly. However, orders sent with the repurposed flag to the eighth server triggered the defective Power Peg code still present on that server. As a result, this server began sending child orders to certain trading centers for execution. Because the cumulative quantity function had been moved, this server continuously sent child orders, in rapid sequence, for each incoming parent order without regard to the number of share executions Knight had already received from trading centers. Although one part of Knight's order handling system recognized that the parent orders had been filled, this information was not communicated to SMARS.

$460,000,000 in 45 Minutes

17. The consequences of the failures were substantial. For the 212 incoming parent orders that were processed by the defective Power Peg code, SMARS sent millions of child orders, resulting in 4 million executions in 154 stocks for more than 397 million shares in approximately 45 minutes. Knight inadvertently assumed an approximately $3.5 billion net long position in 80 stocks and an approximately $3.15 billion net short position in 74 stocks. Ultimately, Knight realized a $460 million loss on these positions.

What is an Error?

19. On August 1, Knight also received orders eligible for the RLP but that were designated for pre-market trading.[6] SMARS processed these orders and, beginning at approximately 8:01 a.m. ET, an internal system at Knight generated automated e-mail messages (called "BNET rejects") that referenced SMARS and identified an error described as "Power Peg disabled." Knight's system sent 97 of these e-mail messages to a group of Knight personnel before the 9:30 a.m. market open. Knight did not design these types of messages to be system alerts, and Knight personnel generally did not review them when they were received. However,

Just a Notification: Nothing to See

19. On August 1, Knight also received orders eligible for the RLP but that were designated for pre-market trading.[6] SMARS processed these orders and, beginning at approximately 8:01 a.m. ET, an internal system at Knight generated automated e-mail messages (called "BNET rejects") that referenced SMARS and identified an error described as "Power Peg disabled." Knight's system sent 97 of these e-mail messages to a group of Knight personnel before the 9:30 a.m. market open. Knight did not design these types of messages to be system alerts, and Knight personnel generally did not review them when they were received. However,

If Only Someone Told Me

these messages were sent in real time, were caused by the code deployment failure, and provided Knight with a potential opportunity to identify and fix the coding issue prior to the market open. These notifications were not acted upon before the market opened and were not used to diagnose the problem after the open.

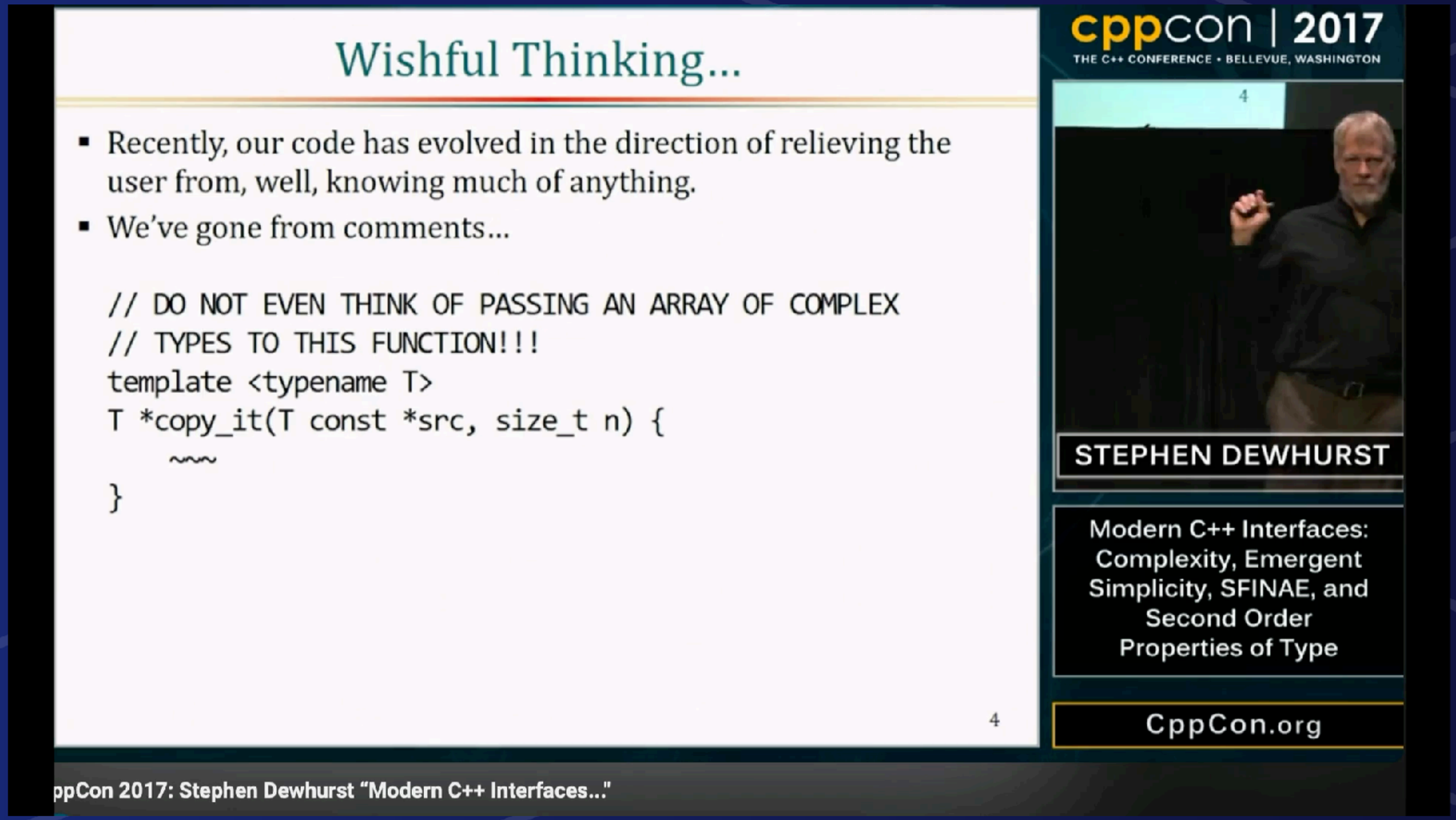
No Automated Monitoring

24. On the morning of August 1, the 33 Account began accumulating an unusually large position resulting from the millions of executions of the child orders that SMARS was sending to the market. Because Knight did not link the 33 Account to pre-set, firm-wide capital thresholds that would prevent the entry of orders, on an automated basis, that exceeded those thresholds, SMARS continued to send millions of child orders to the market despite the fact that the parent orders already had been completely filled.[7] Moreover, because the 33 Account held

Rollback???

27. On August 1, Knight did not have supervisory procedures concerning incident response. More specifically, Knight did not have supervisory procedures to guide its relevant personnel when significant issues developed. On August 1, Knight relied primarily on its technology team to attempt to identify and address the SMARS problem in a live trading environment. Knight's system continued to send millions of child orders while its personnel attempted to identify the source of the problem. In one of its attempts to address the problem, Knight uninstalled the new RLP code from the seven servers where it had been deployed correctly. This action worsened the problem, causing additional incoming parent orders to activate the Power Peg code that was present on those servers, similar to what had already occurred on the eighth server.

Really Good Bugs



Discussion and Questions
How can we test it?

- Remove old code and add new code at the same time
- Reuse and existing "value" for completely different functionality
- Stopped using Power Peg in 2003, dead code still around
- Refactored code in 2005 ; moved dead code ; test still pass
- Manual deployment ; no review ; no tests
- Different components had different view of same thing
- email for important log messages
- Rollback made matters worse

Generated Code

- Got your own code generator?
  - How do you test it?
    - How do you test the generated code?
    - How do you code review the generated code?
  - Always commit code generated from your own generator into the source repository
    - I know everyone else says not to do that
      - I used to be part of everyone...
      - Everyone else is wrong
    - I learned the hard way - not as hard as Knight, but hard enough to never repeat that mistake again.

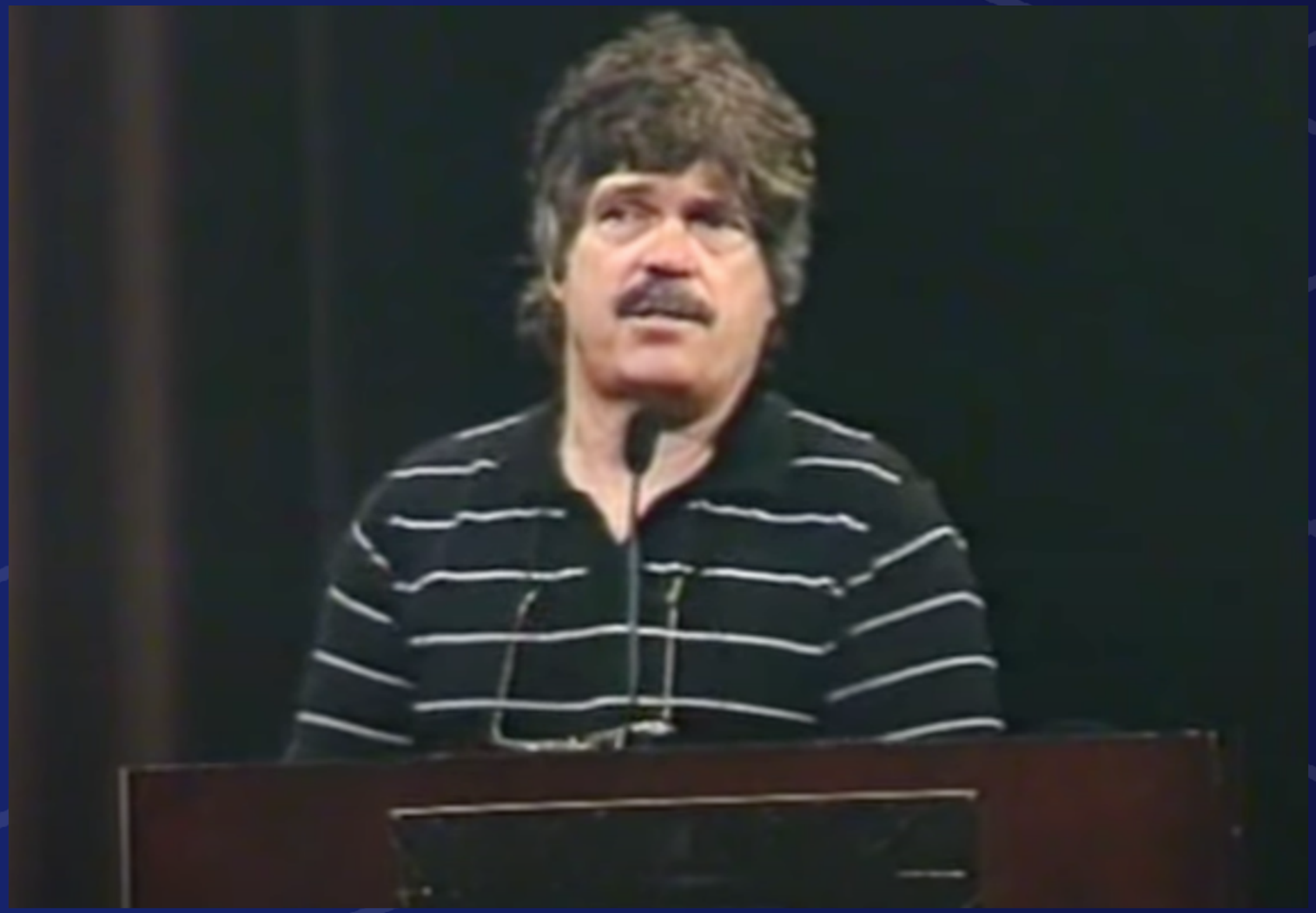
Can We Talk?

Object Oriented Programming

- Name the top object oriented programming languages

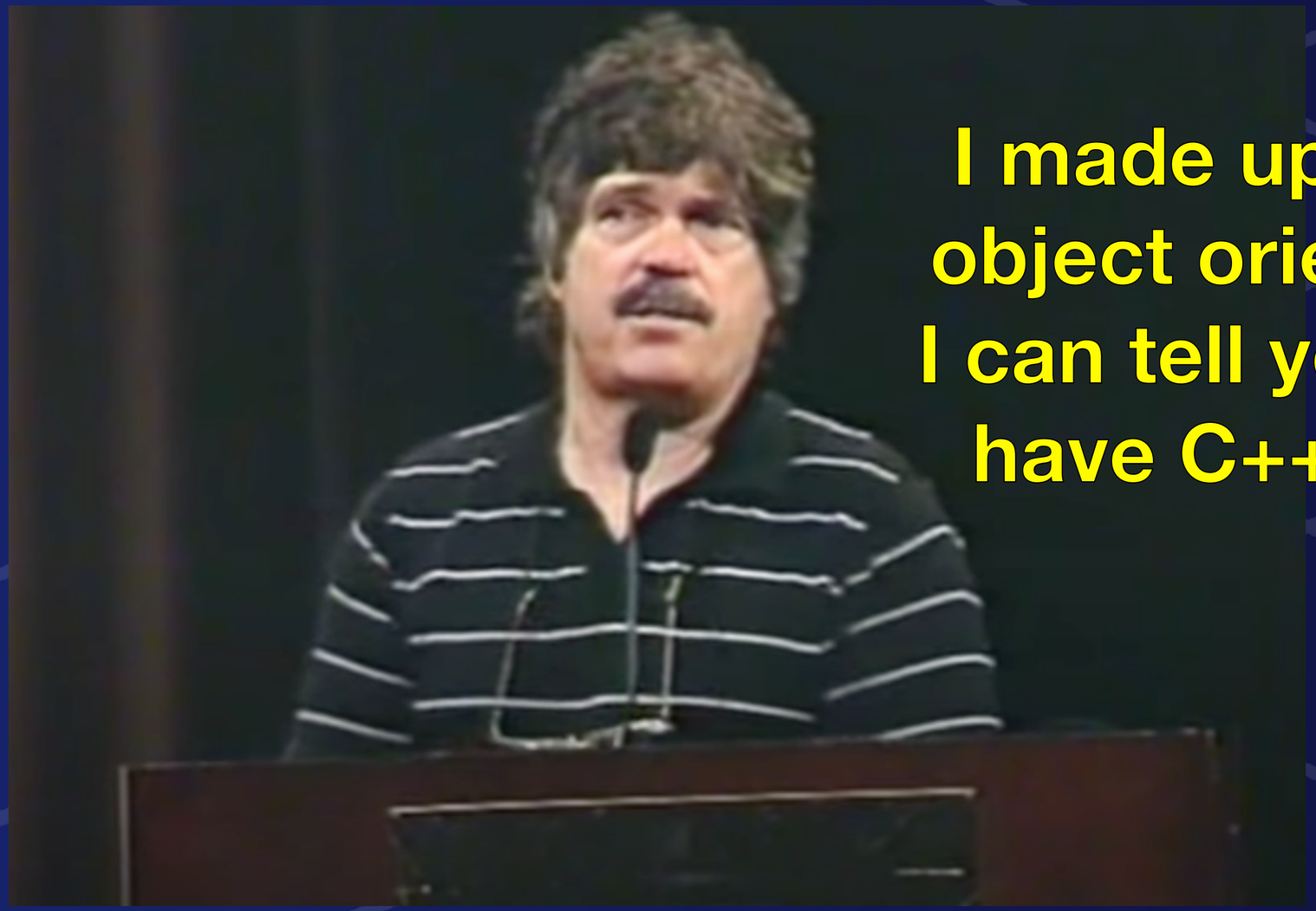
Object Oriented Programming

- Name the top object oriented programming languages



Object Oriented Programming

- Name the top object oriented programming languages



Object Oriented Programming

- Name the top object oriented programming languages
- My Background
  - Grad School: AI; LisP and Smalltalk
- What Defines Object Oriented Programming?
  - Encapsulation
    - Local retention and protection and hiding of state-process
  - Message Passing
    - Cannot be Commands
    - Most important concept
  - Very Late Binding
    - Even behavior

So, What is the Stuff Most of the Industry Calls OOP?

- Alan Kay is nicer than me ; he uses "Real OOP" for the OG stuff
- But why should the OG have to change names?
- I propose Pseudo Object Oriented Programming
  - It even comes with its own acronym: POOP
  - And emoji: 

So, What is the Stuff Most of the Industry Calls OOP?

- Alan Kay is nicer than me ; he uses "Real OOP" for the OG stuff
- But why should the OG have to change names?
- I propose Pseudo Object Oriented Programming
  - It even comes with its own acronym: POOP
  - And emoji: 

So, What is the Stuff Most of the Industry Calls OOP?

- Alan Kay is nicer than me ; he uses "Real OOP" for the OG stuff
- But why should the OG have to change names?
- I propose Pseudo Object Oriented Programming
  - It even comes with its own acronym: POOP
  - And emoji: 

Object Oriented Programming

Let's just talk

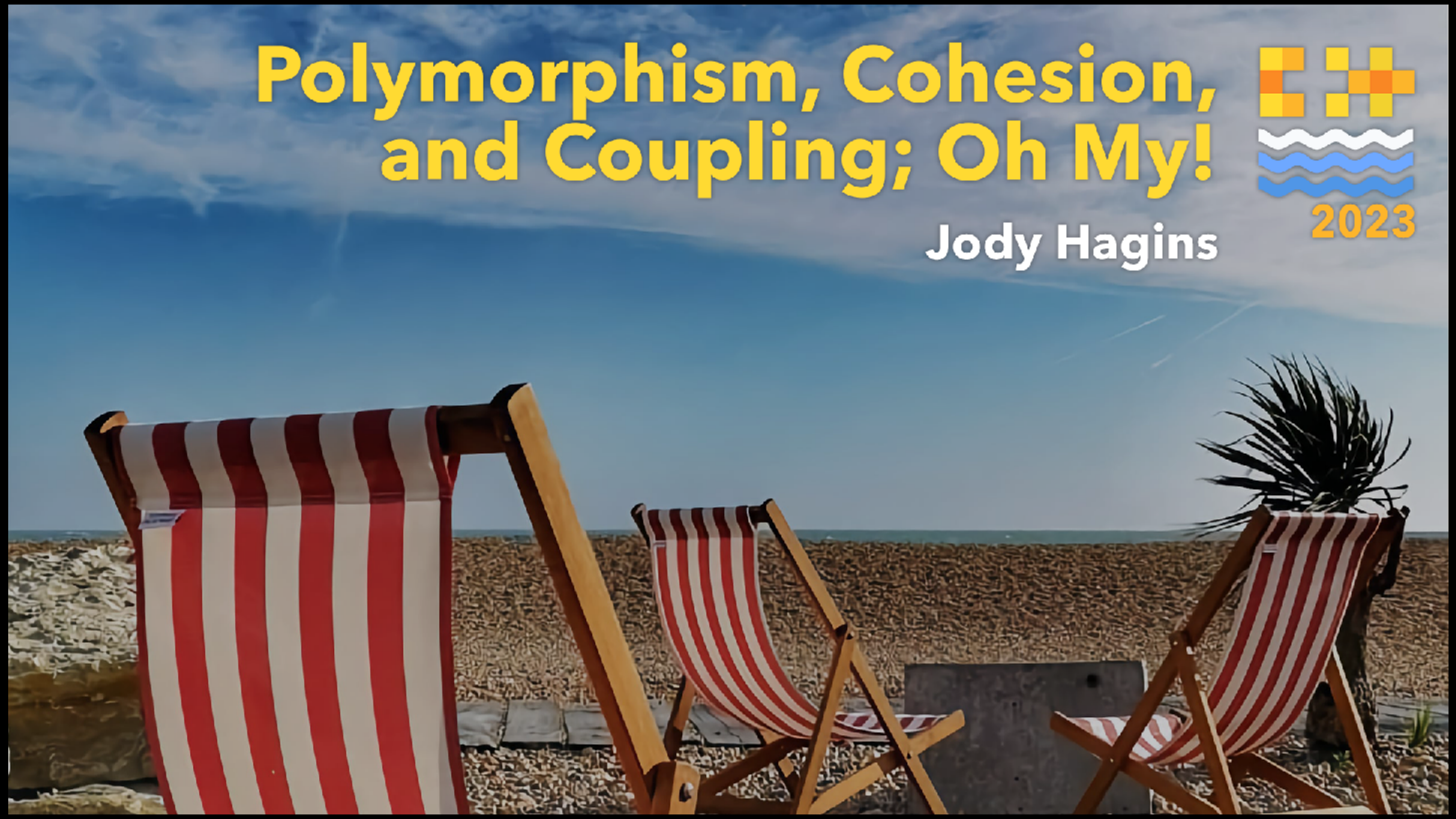
Using Modern C++ to Revive an Old Design

JODY HAGINS

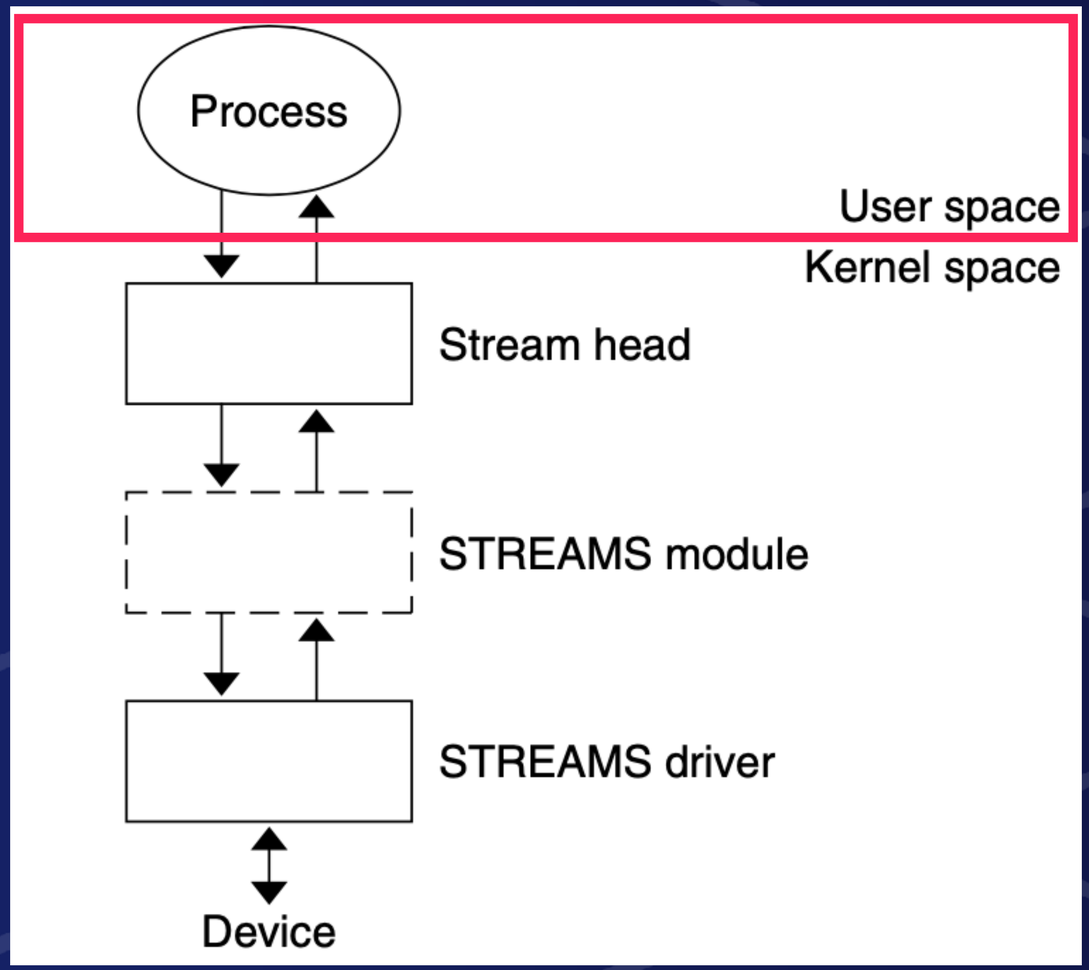
Polymorphism, Cohesion, and Coupling; Oh My!

Jody Hagins

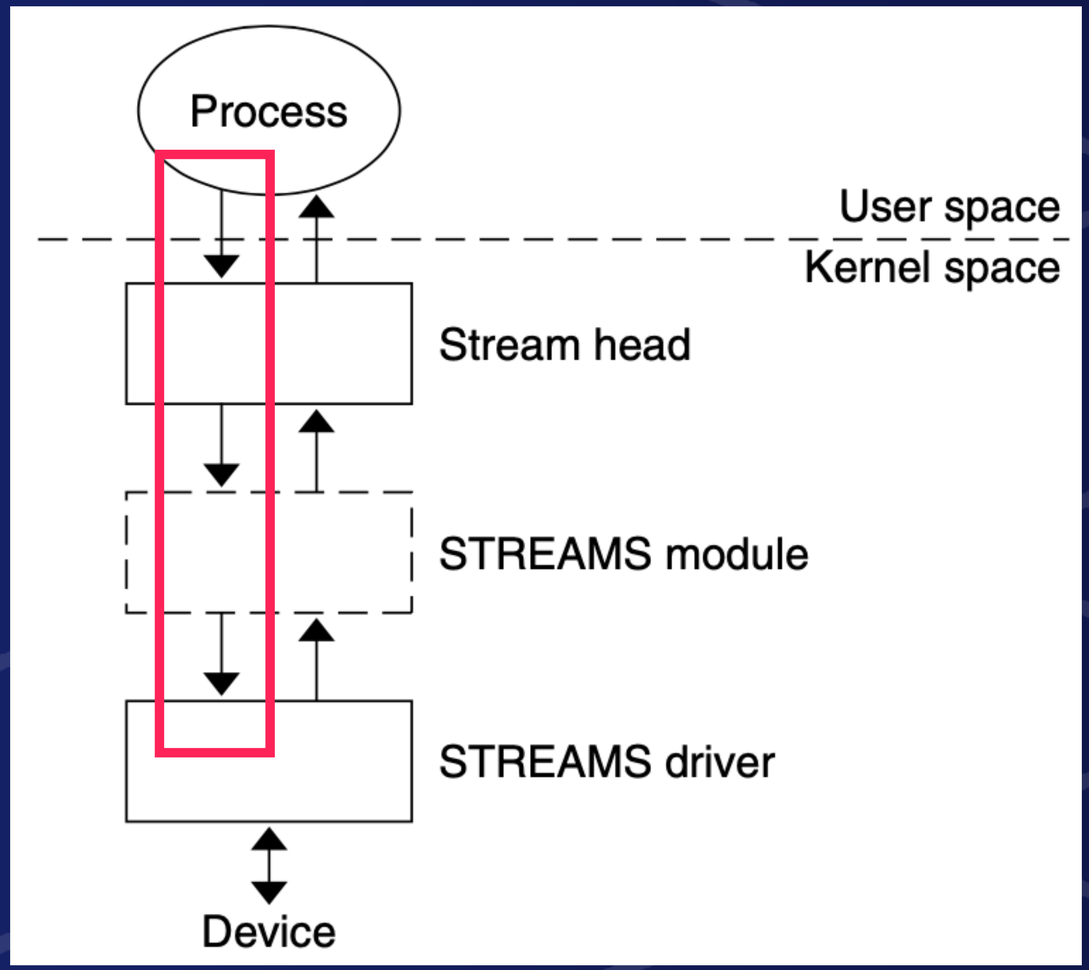
2023



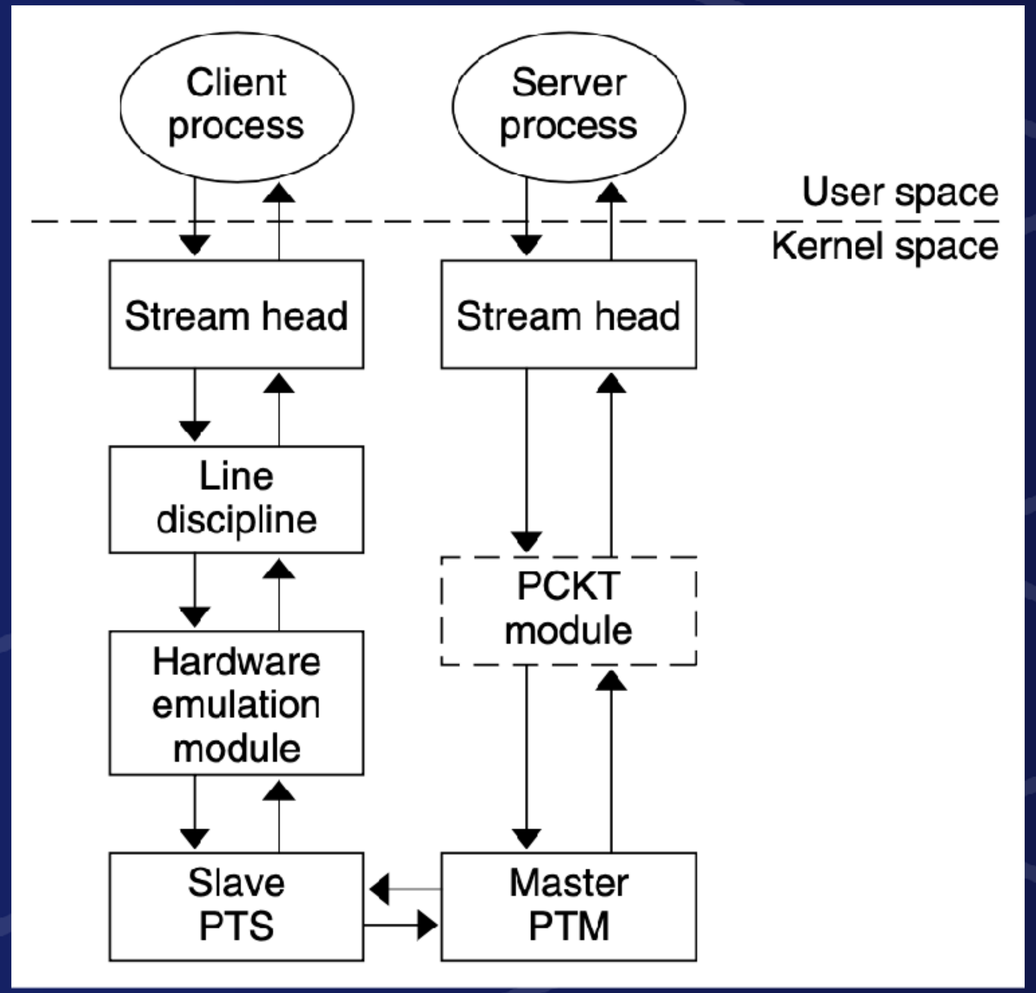
Love at First Sight



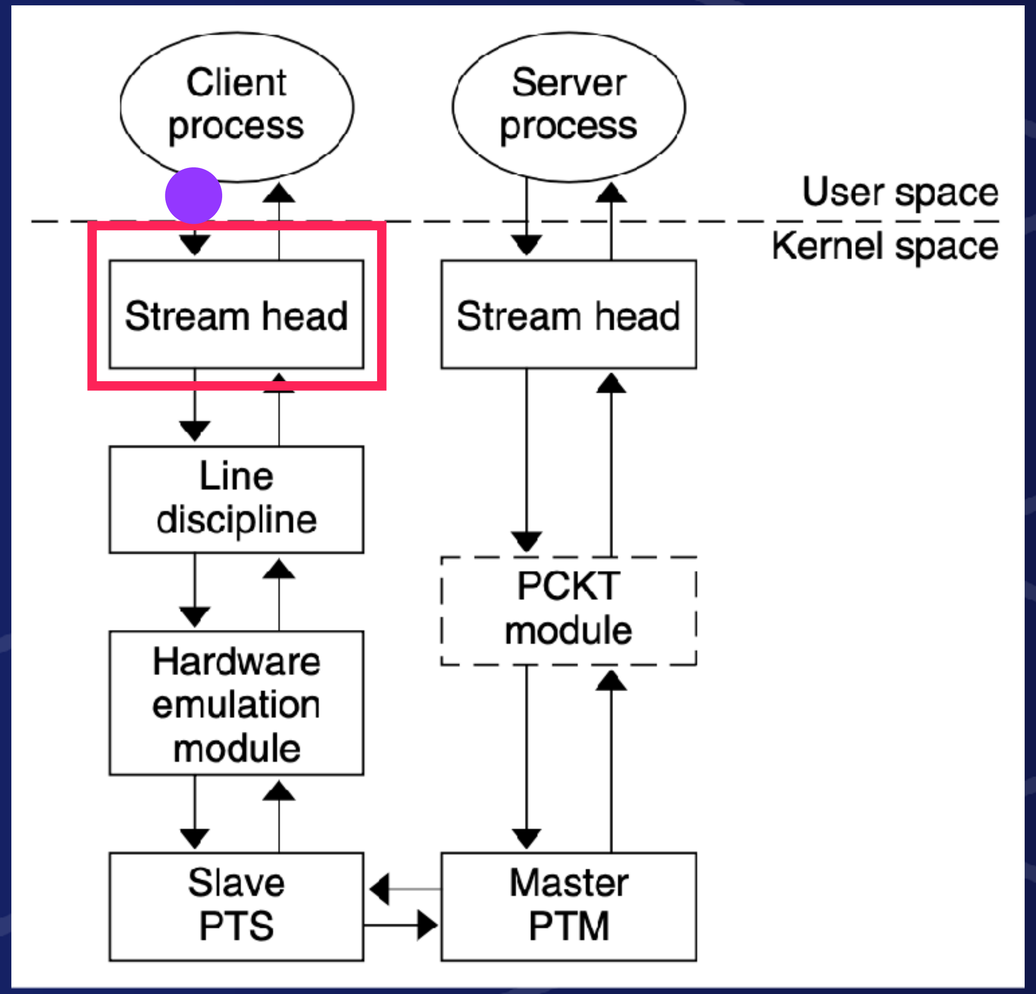
Love at First Sight



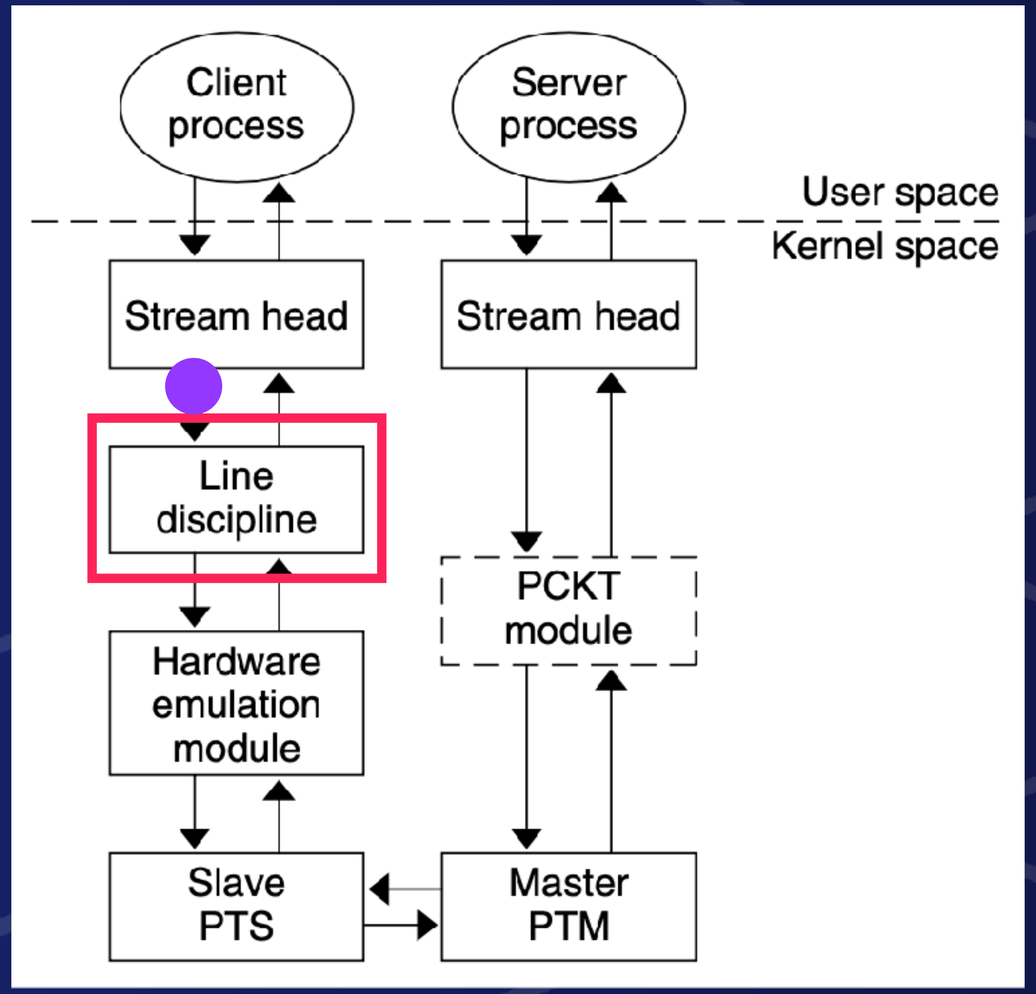
Pseudo-TTY in STREAMS



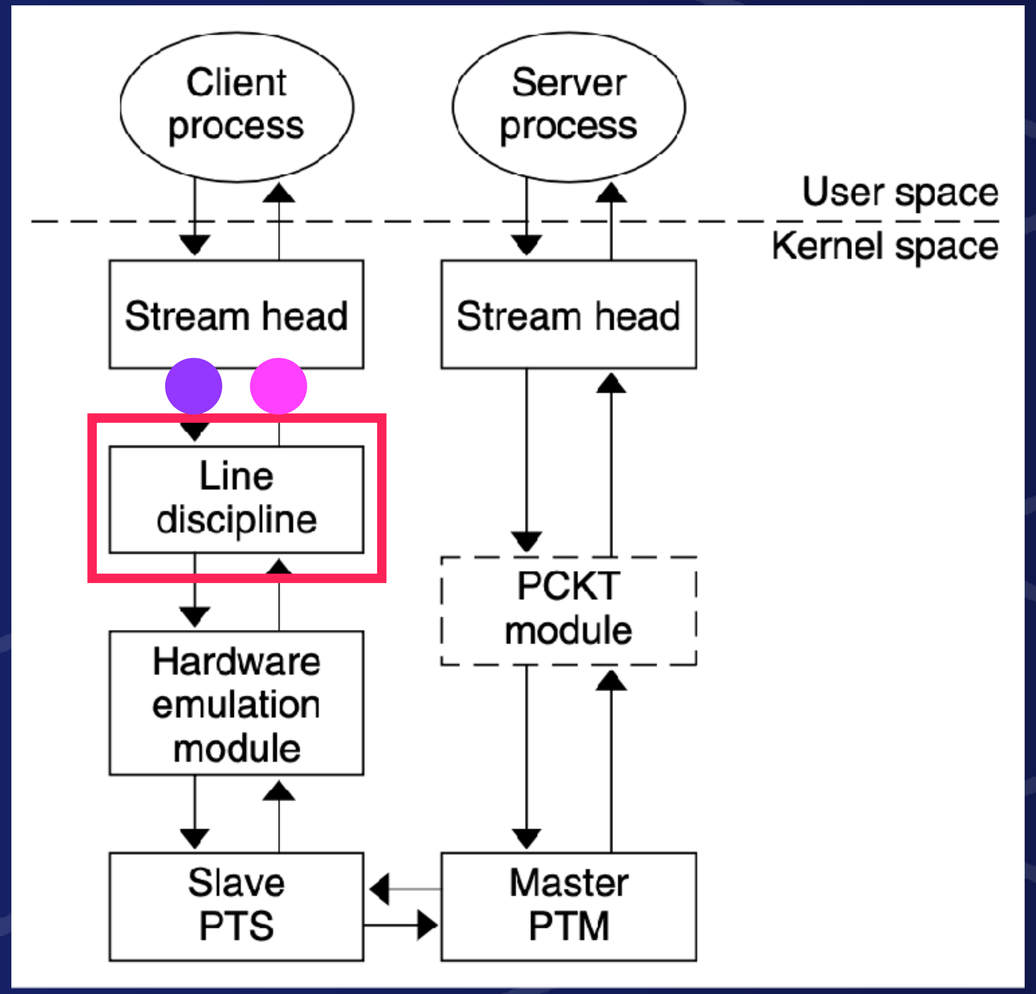
Pseudo-TTY in STREAMS



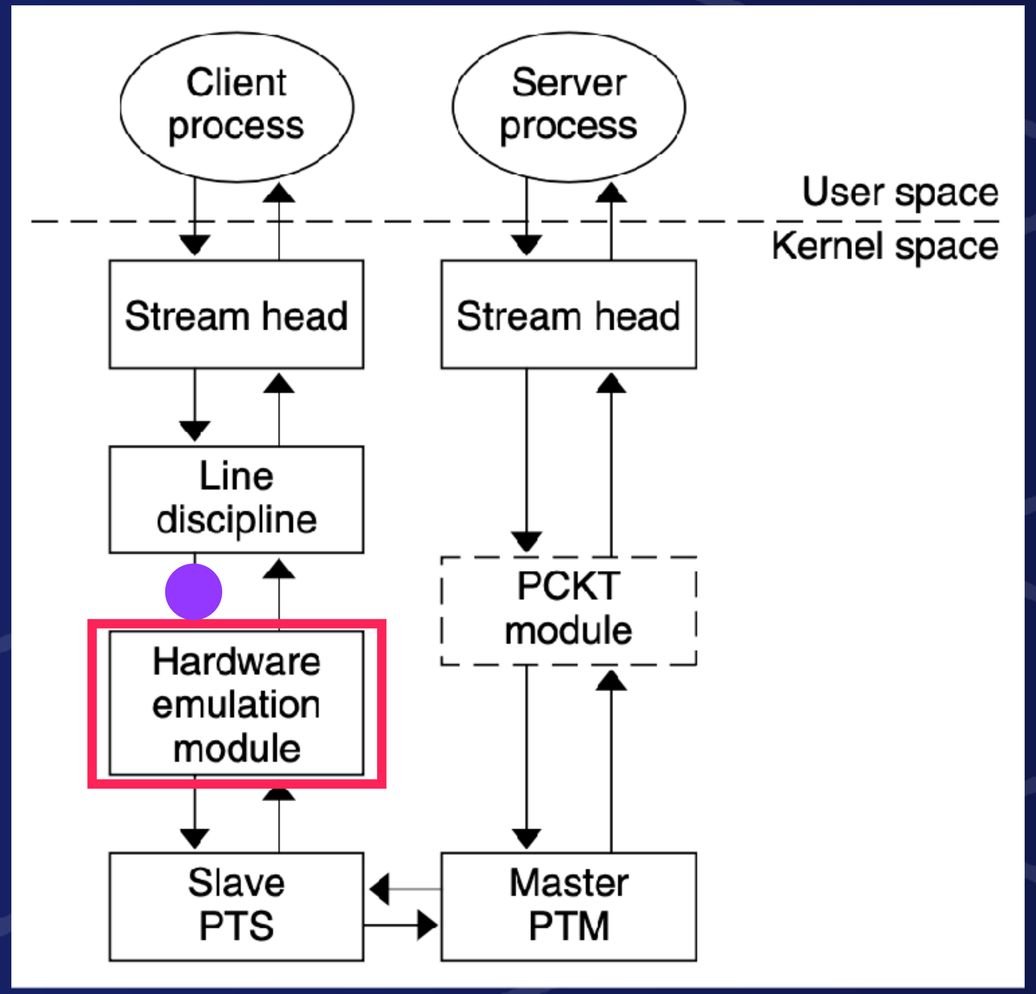
Pseudo-TTY in STREAMS



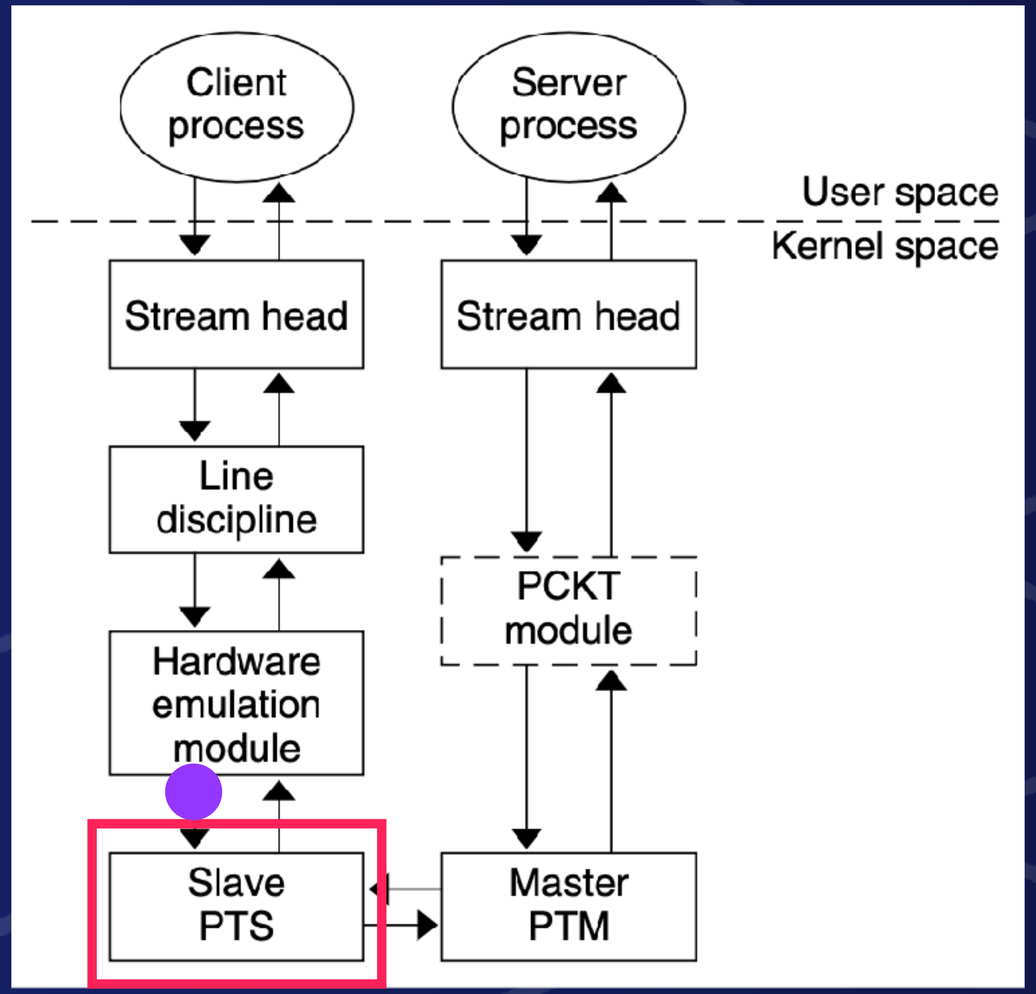
Pseudo-TTY in STREAMS



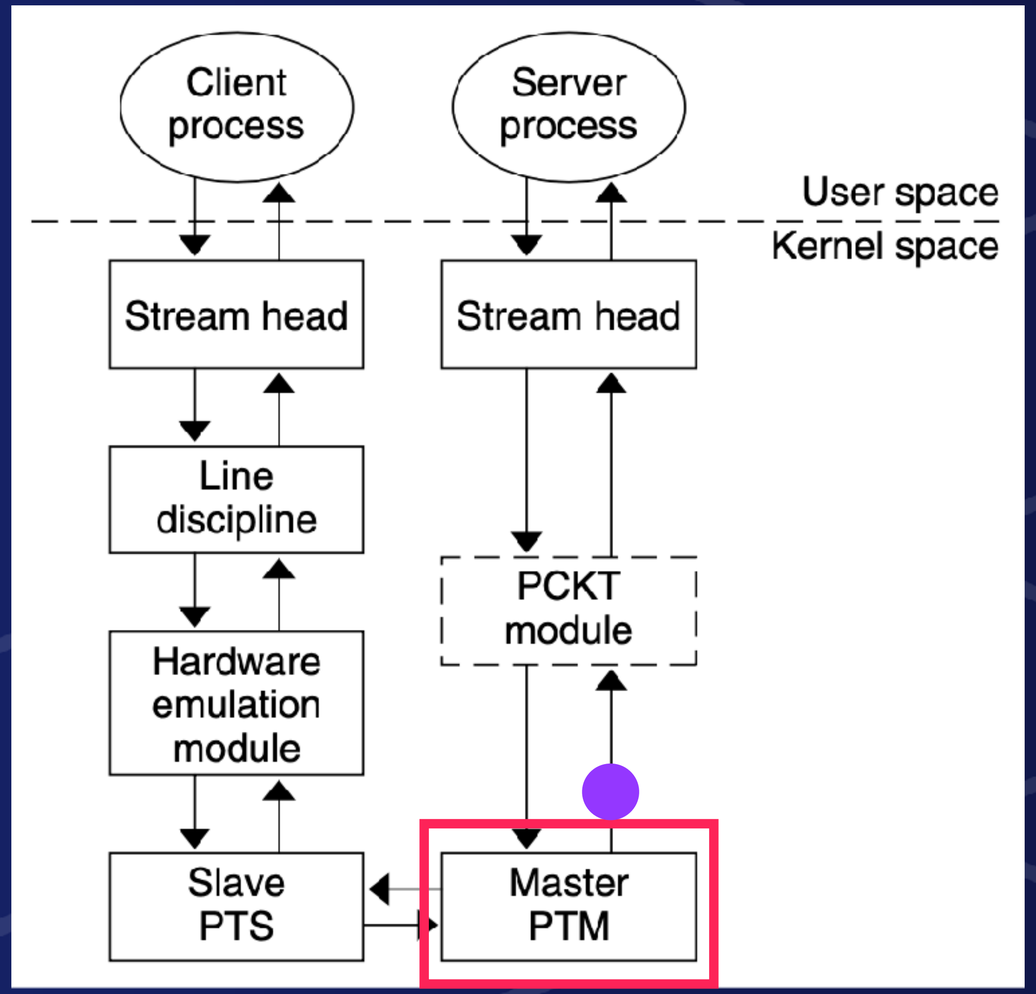
Pseudo-TTY in STREAMS



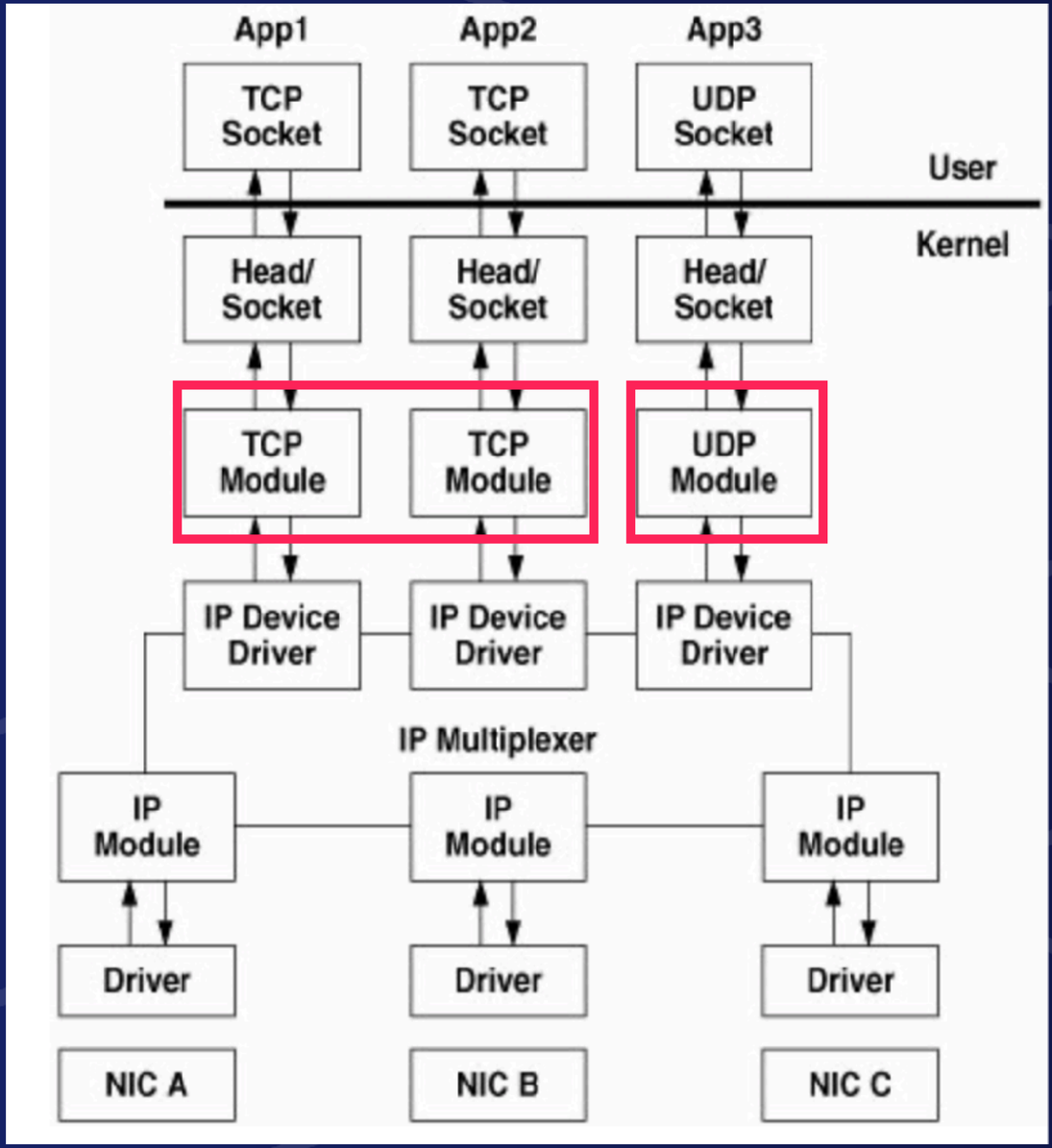
Pseudo-TTY in STREAMS



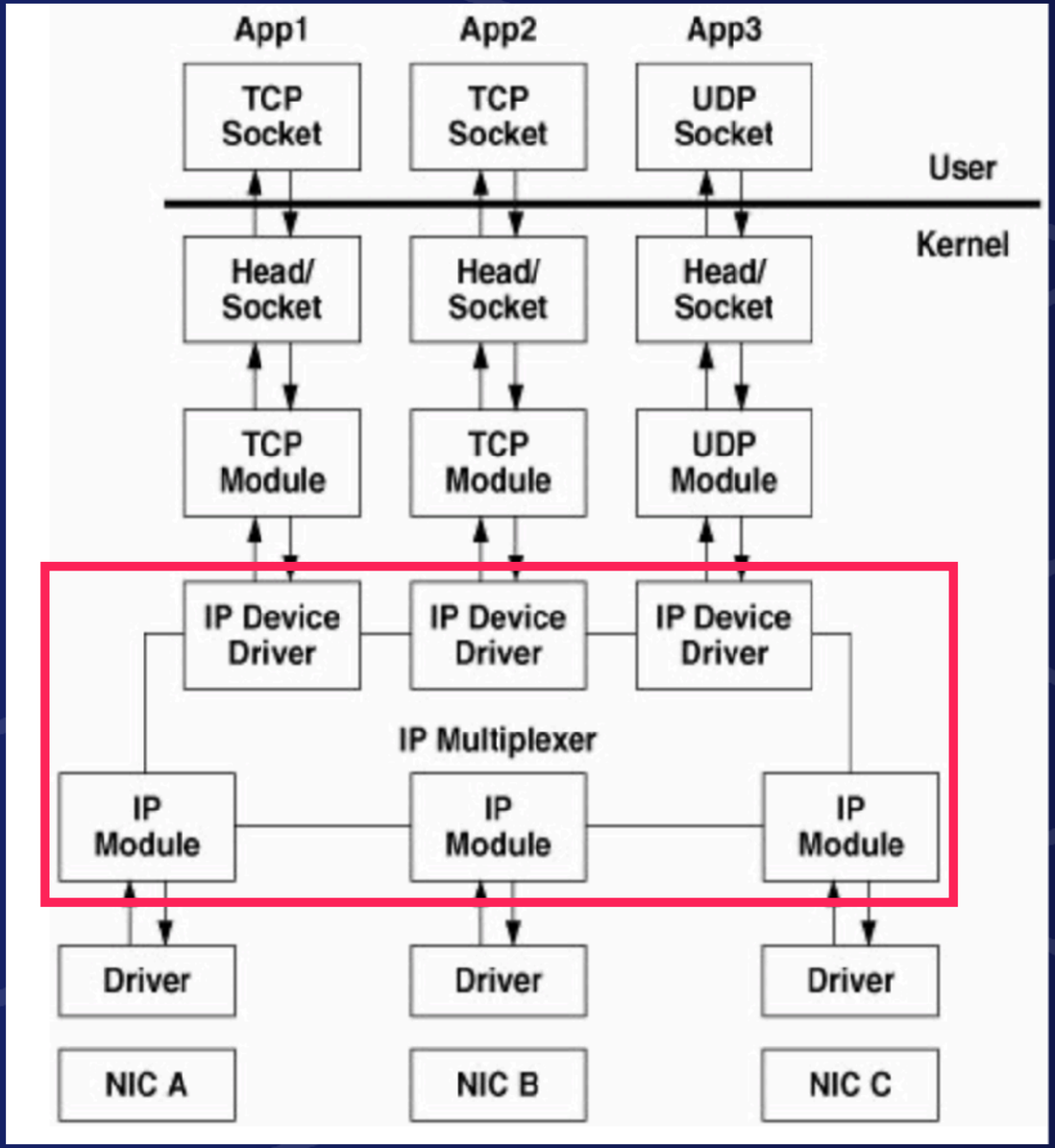
Pseudo-TTY in STREAMS



TCP/UDP in STREAMS



TCP/UDP in STREAMS



Reliability

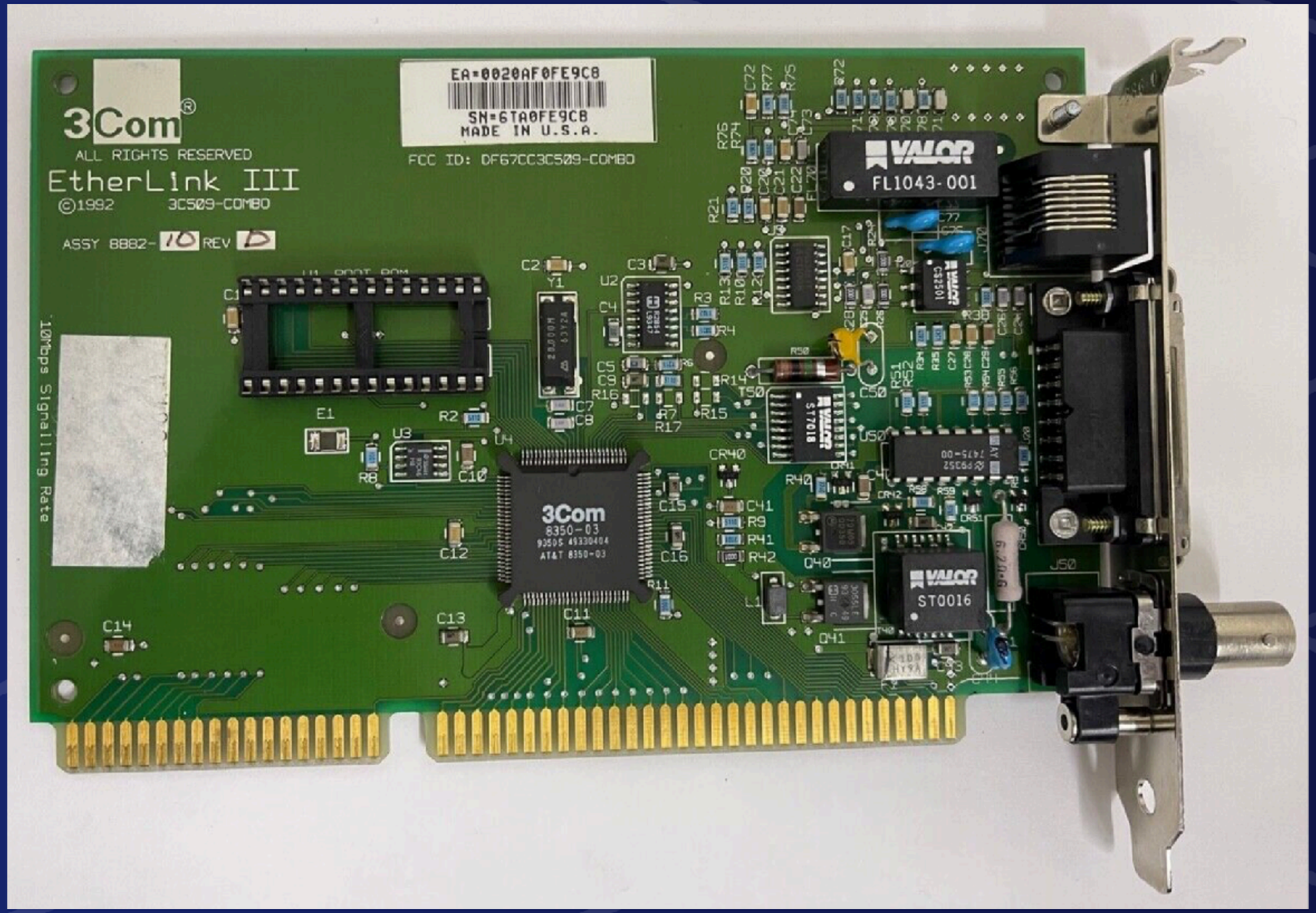
Remember the library to intercept SYSV IPC?

New requirement - reliability

Multiple network cards

Use available network cards to recovery from failures

Reliability



3COM ETHERLINK III

Reliability

Remember the library to intercept SYSV IPC?

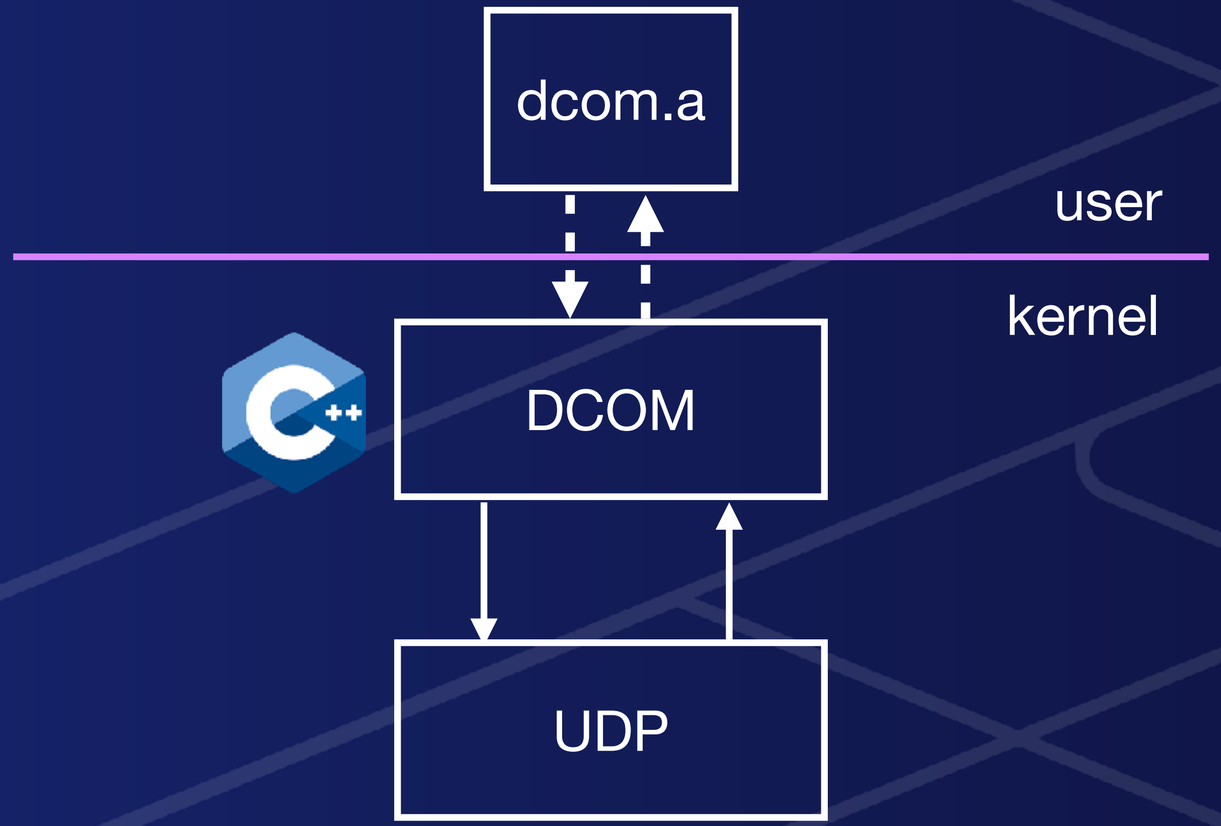
New requirement - reliability

Multiple network cards

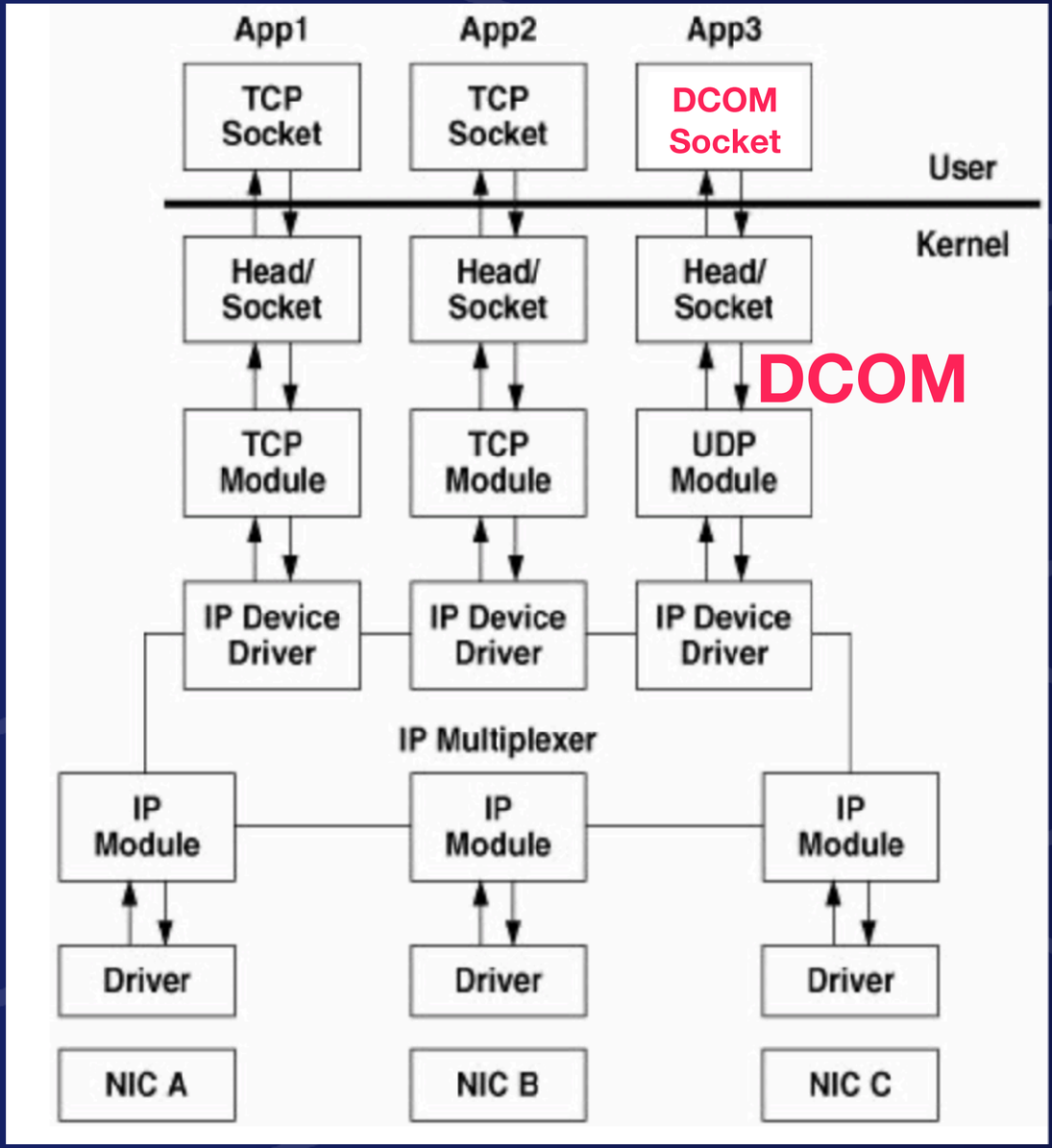
Use available network cards to recovery from failures

Reliability

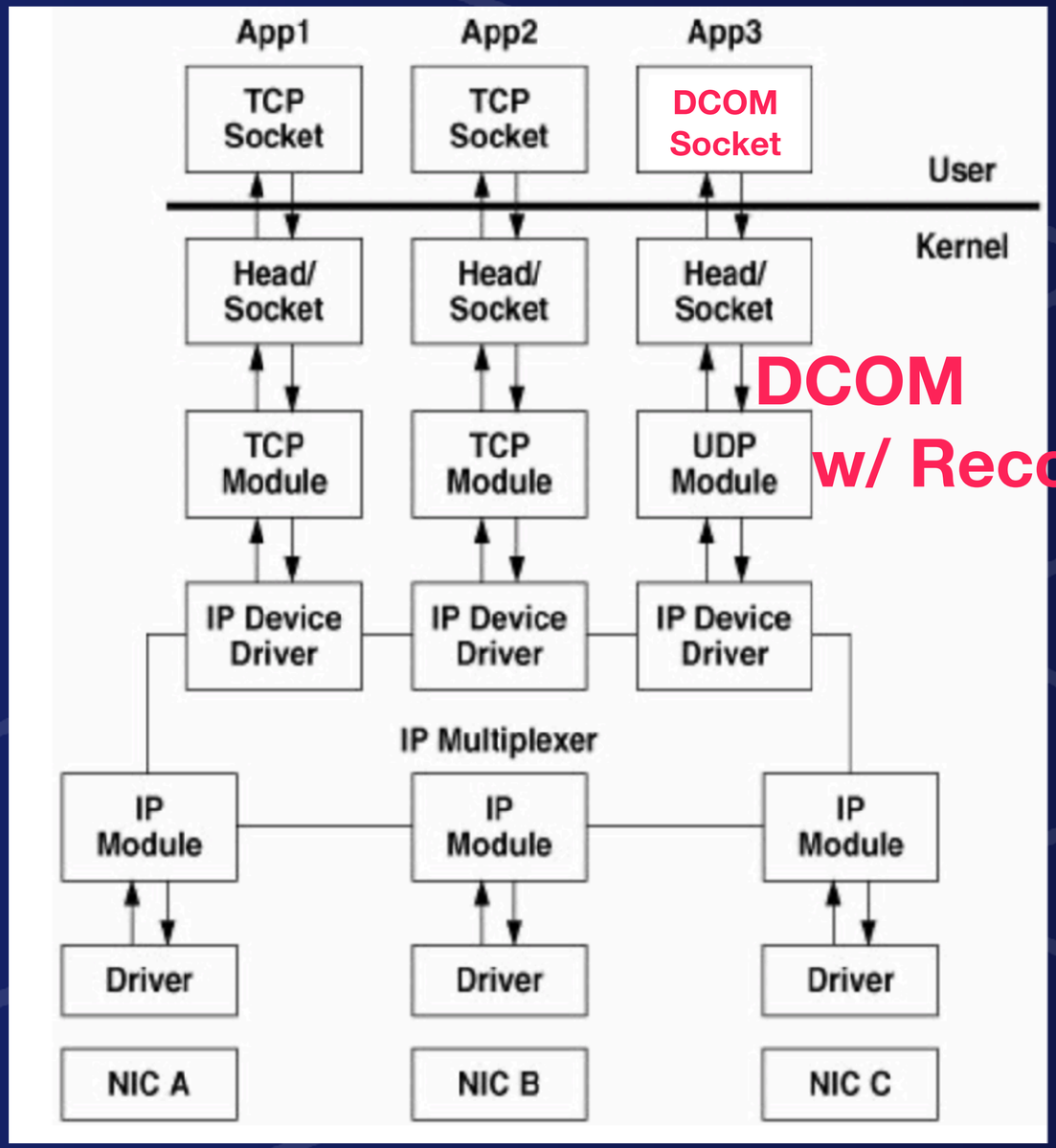
Remember the library to intercept SYSV IPC?



Reliability

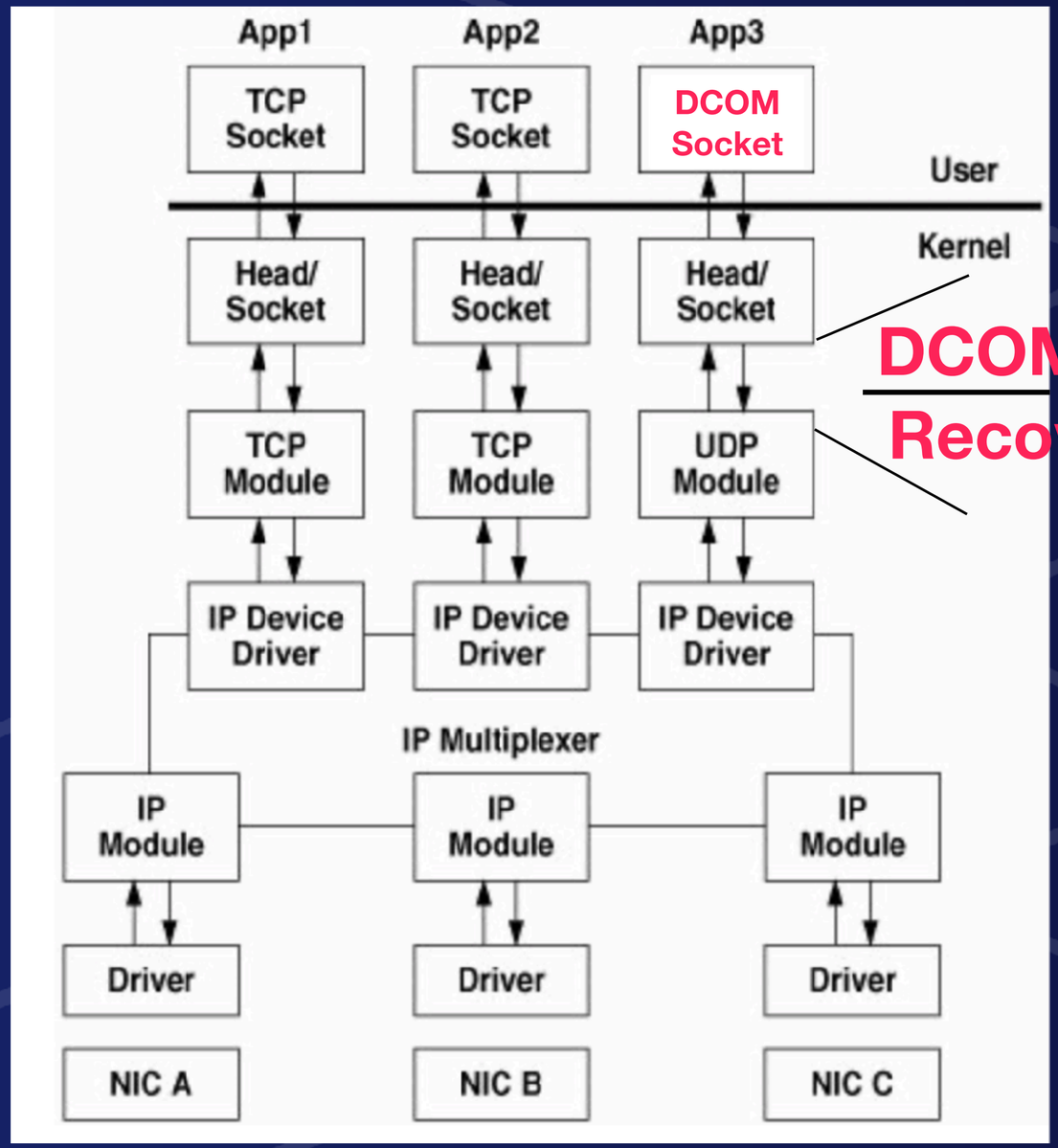


Reliability

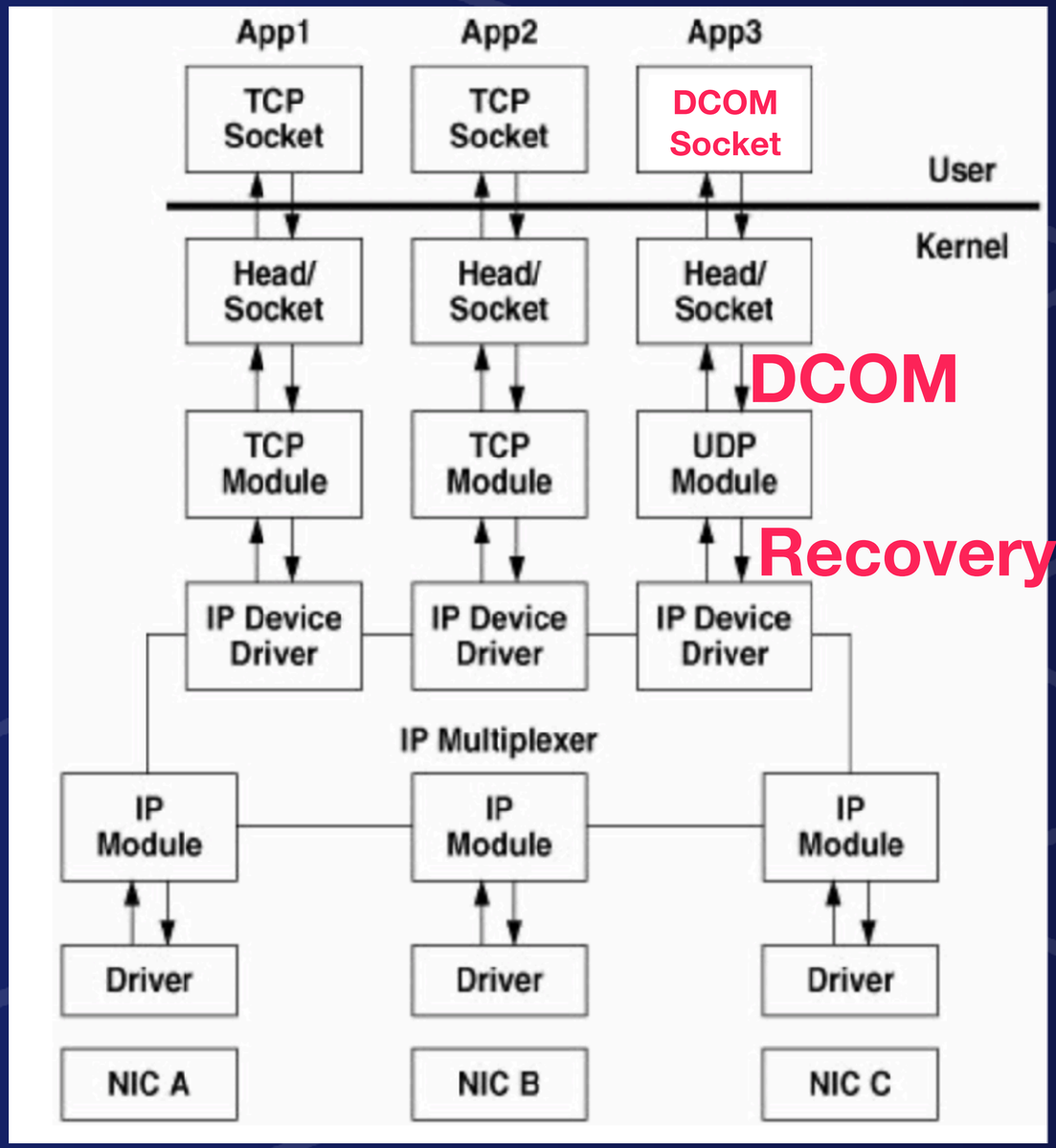


DCOM w/ Recovery

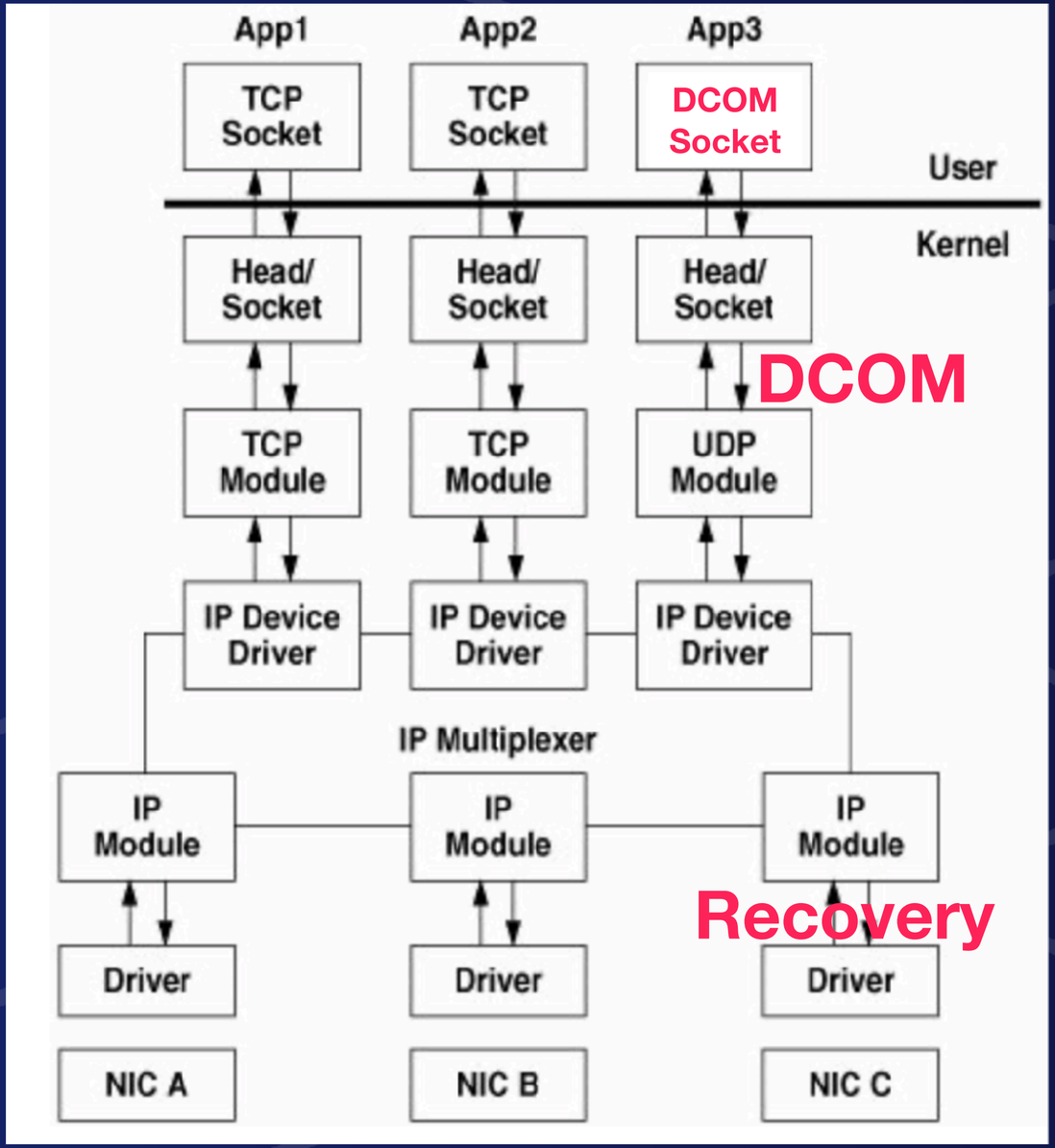
Reliability



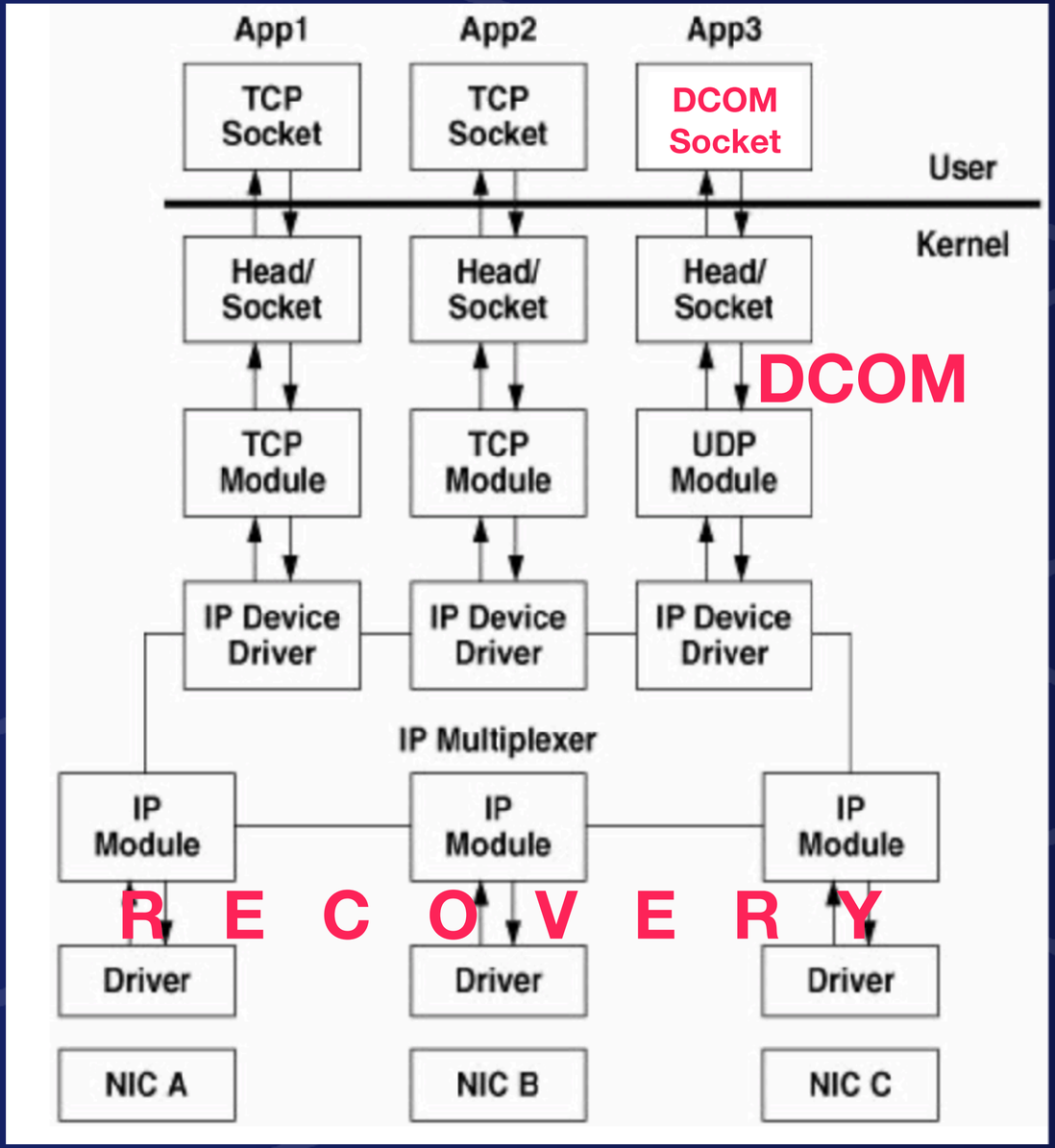
Reliability



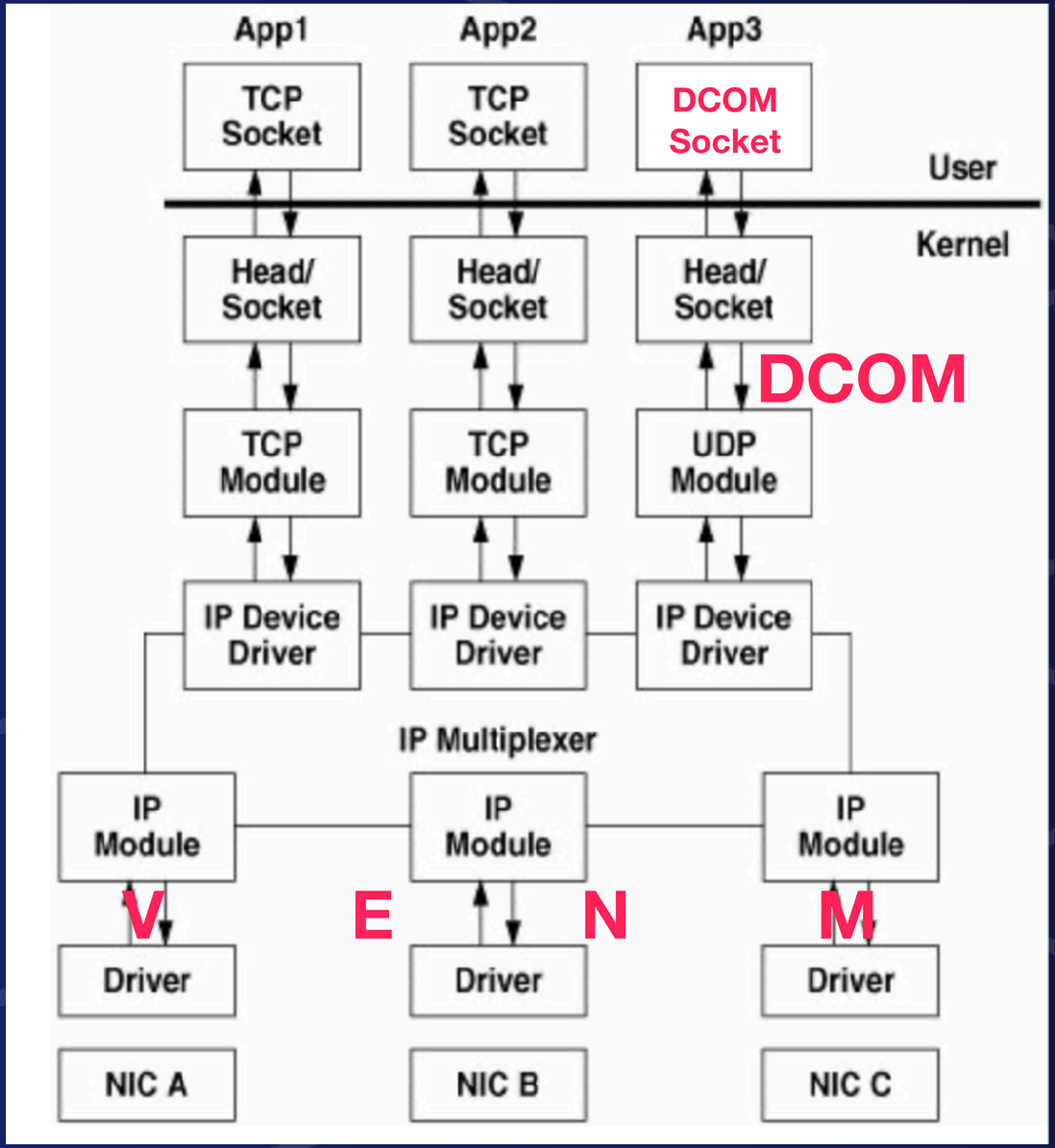
Reliability



Reliability



Reliability



Reliability