Using Modern C++ to Revive an Old Design

JODY HAGINS

MAYSTREET

CppCon 2022

Using Modern C++ to Revive and Old Design

AKA: Coupling and Cohesion are Guiding Lights

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

The Holy Grail

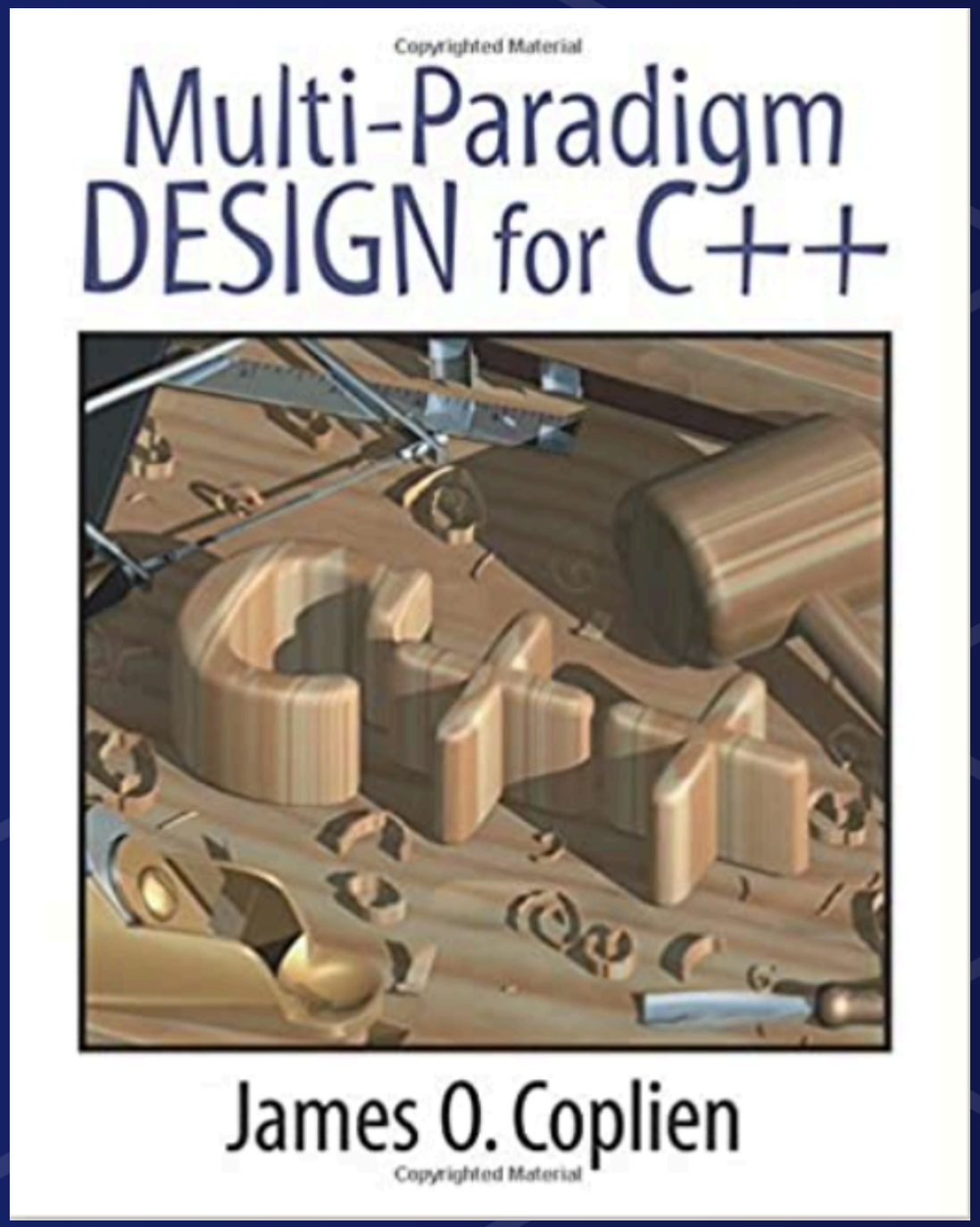
Low Coupling, High Cohesion, Composable, Testable, Reusable, Functional, Modular, Easy to Use, Easy to Change, High Throughput, Low Latency, Optimal Code Generation: Pick All of Them!

The Holy Grail

Low Coupling, High Cohesion, Composable, Testable, Reusable, Functional, Modular, Easy to Use, Easy to Change, High Throughput, Low Latency, Optimal Code Generation: Pick All of Them!

The Holy Grail

C++ Design


Copyrighted Material
Multi-Paradigm
DESIGN for C++
James O. Coplien
Copyrighted Material

Coupling vs. Cohesion

"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

Coupling vs. Cohesion

"One primary goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

Coupling vs. Cohesion

CSE 403 - Washington University

Coupling vs. Cohesion

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Coupling

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Coupling

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Highly coupled have program units dependent on each other.

Coupling

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Highly coupled have program units dependent on each other.

Loosely coupled are made up of units that are independent or almost independent.

Coupling

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Highly coupled have program units dependent on each other.

Loosely coupled are made up of units that are independent or almost independent.

Modules are independent if they can function completely without the presence of the other.

Cohesion

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

Cohesion

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

A component should implement a single logical function or single logical entity.

Cohesion

CSE 403 - Washington University

An indication of the strength of interconnections between program units.

A component should implement a single logical function or single logical entity.

All the parts should contribute to the implementation.

Coupling vs. Cohesion

"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

An indication of the strength of interconnections between program units.

Coupling vs. Cohesion

"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

An indication of the strength of interconnections between program units.

Cohesion is how much one part of a code base forms an atomic program unit

Coupling vs. Cohesion

"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

An indication of the strength of interconnections between program units.

Cohesion is how much one part of a code base forms an atomic program unit

Coupling is how much a single program unit depends upon other program units

Simplified Example

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    // Packet processing code
}
```

Can You Say Code Review?

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    launch_rocket(
        global_rocket_launcher,
        random_coordinates());
    // Packet processing code
}
```

Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (Header(packet).is_compressed()) {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (Header(packet).is_compressed()) {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (Header(packet).is_compressed()) {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (Header(packet).is_compressed()) {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

More or Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (packet.timestamp() >= compression_timestamp &&
        Header(packet).is_compressed())
    {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

More or Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (packet.timestamp() >= compression_timestamp &&
        Header(packet).is_compressed())
    {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

The code block is the program unit that can create both the greatest coupling and the least cohesion.

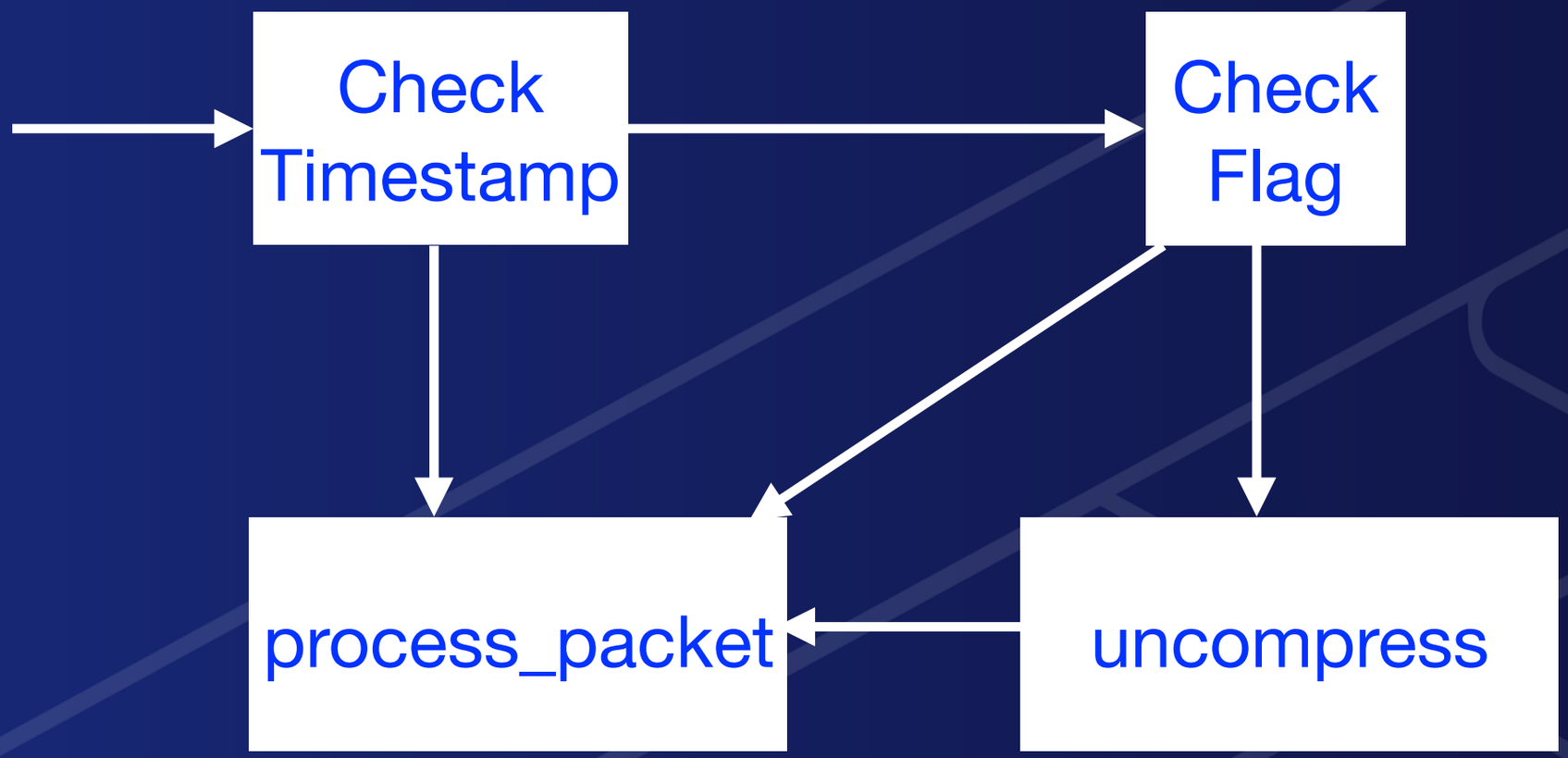
More or Less Scary

```
Result
SomeClass::
process_packet(Packet const & packet)
{
    if (packet.timestamp() >= compression_timestamp &&
        Header(packet).is_compressed())
    {
        return process_packet(uncompress(packet));
    }
    // packet processing code
}
```

The code block is the program unit that can create both the greatest coupling and the least cohesion.

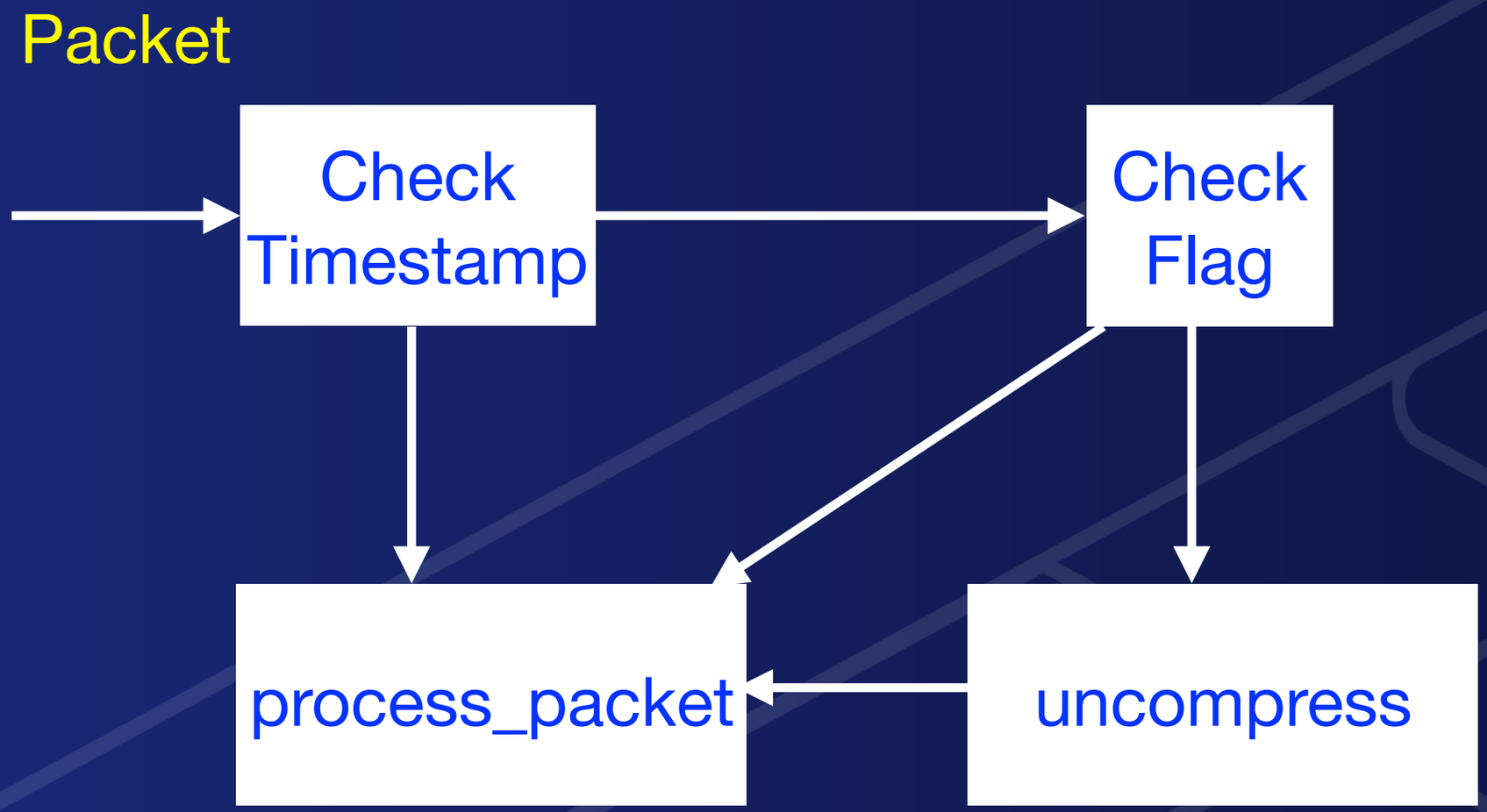
Coupling vs. Cohesion

Compose small, cohesive program units



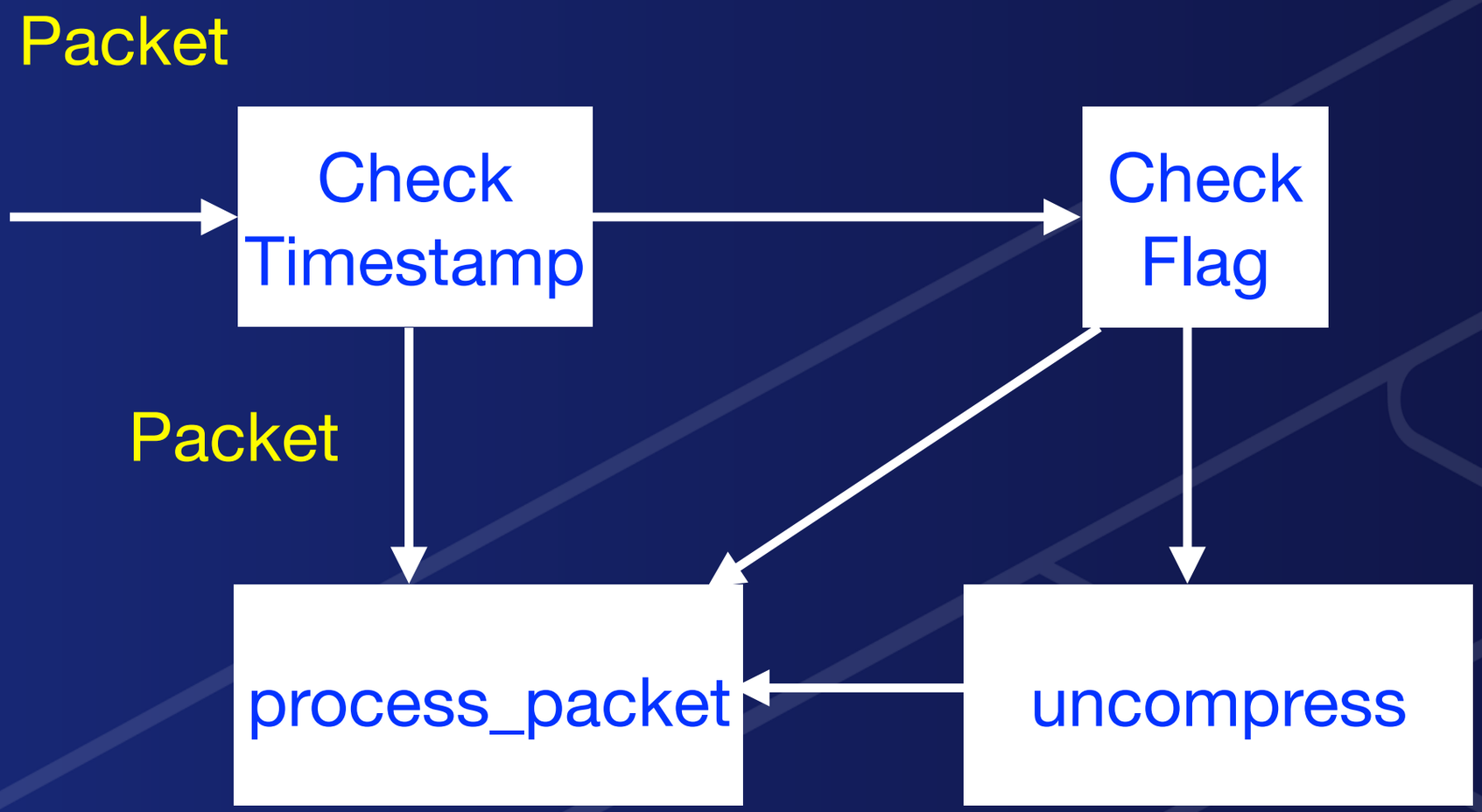
Coupling vs. Cohesion

Compose small, cohesive program units



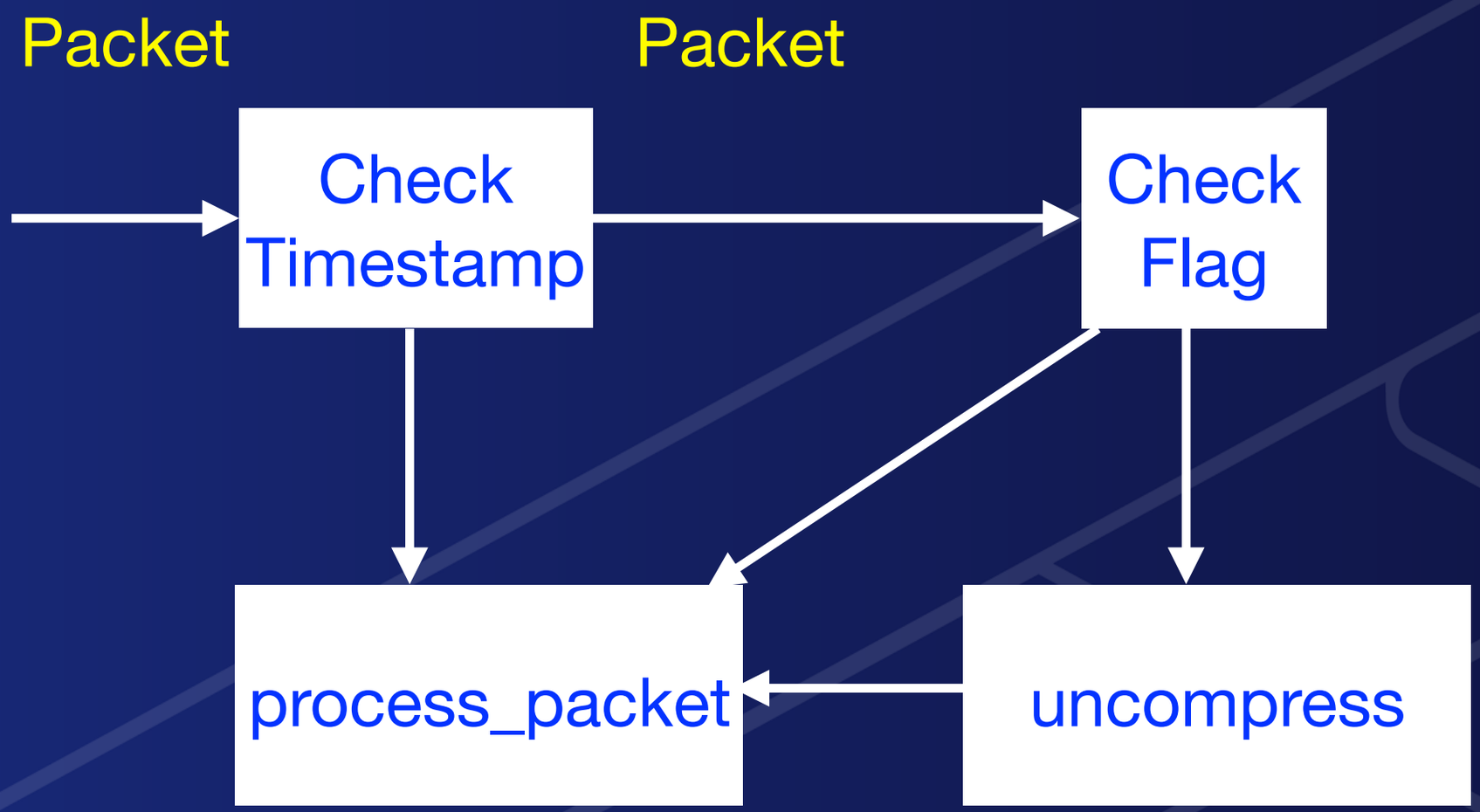
Coupling vs. Cohesion

Compose small, cohesive program units



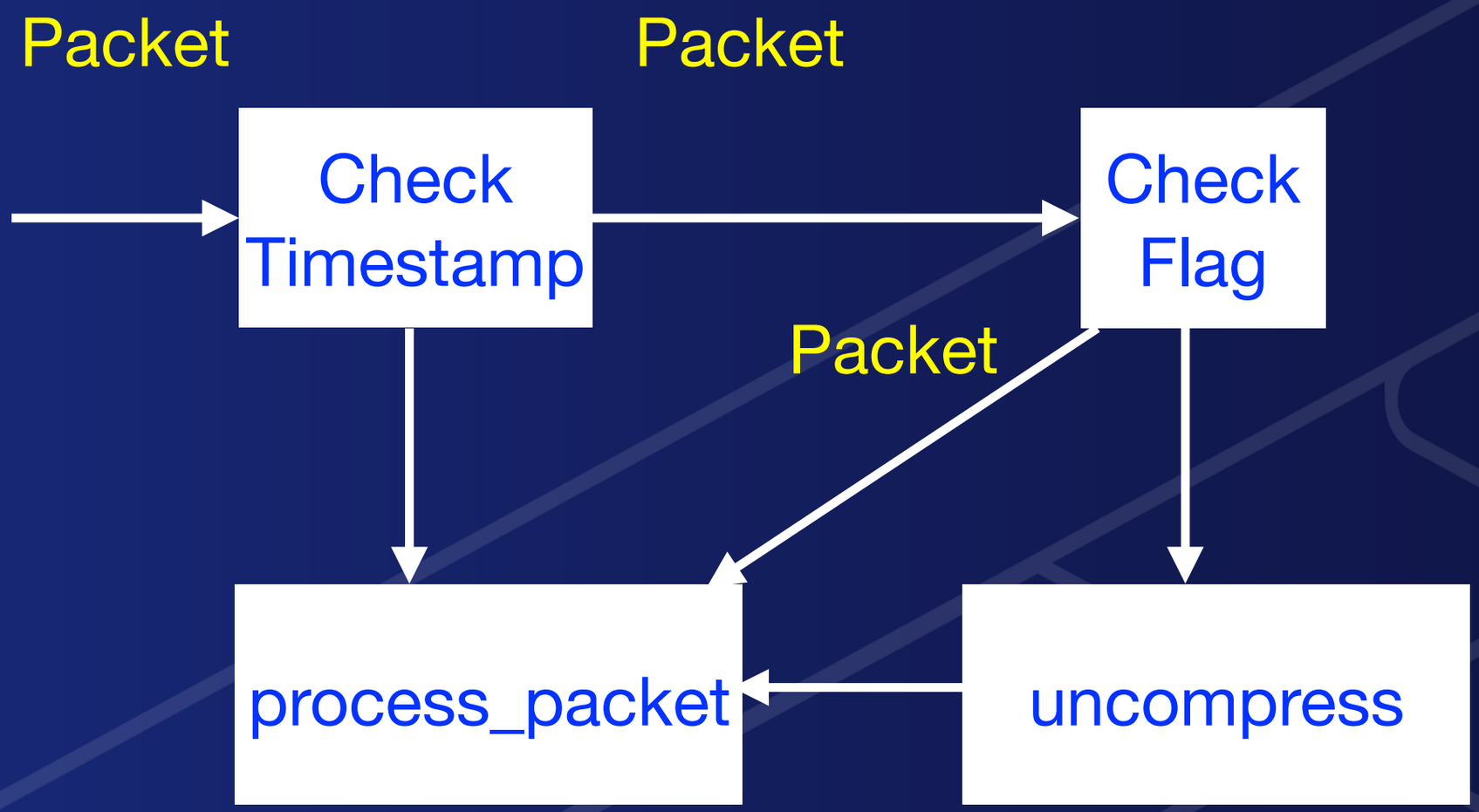
Coupling vs. Cohesion

Compose small, cohesive program units



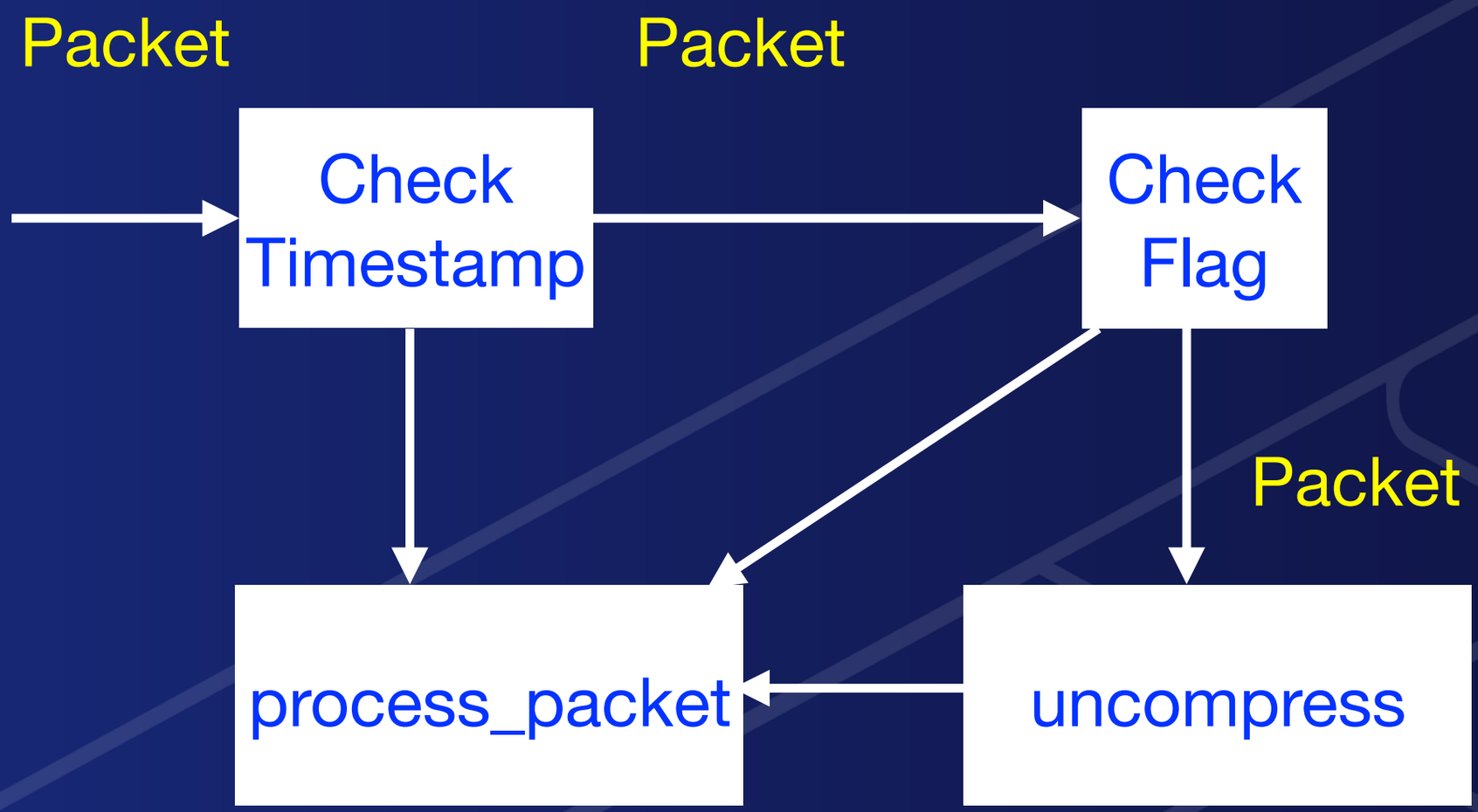
Coupling vs. Cohesion

Compose small, cohesive program units



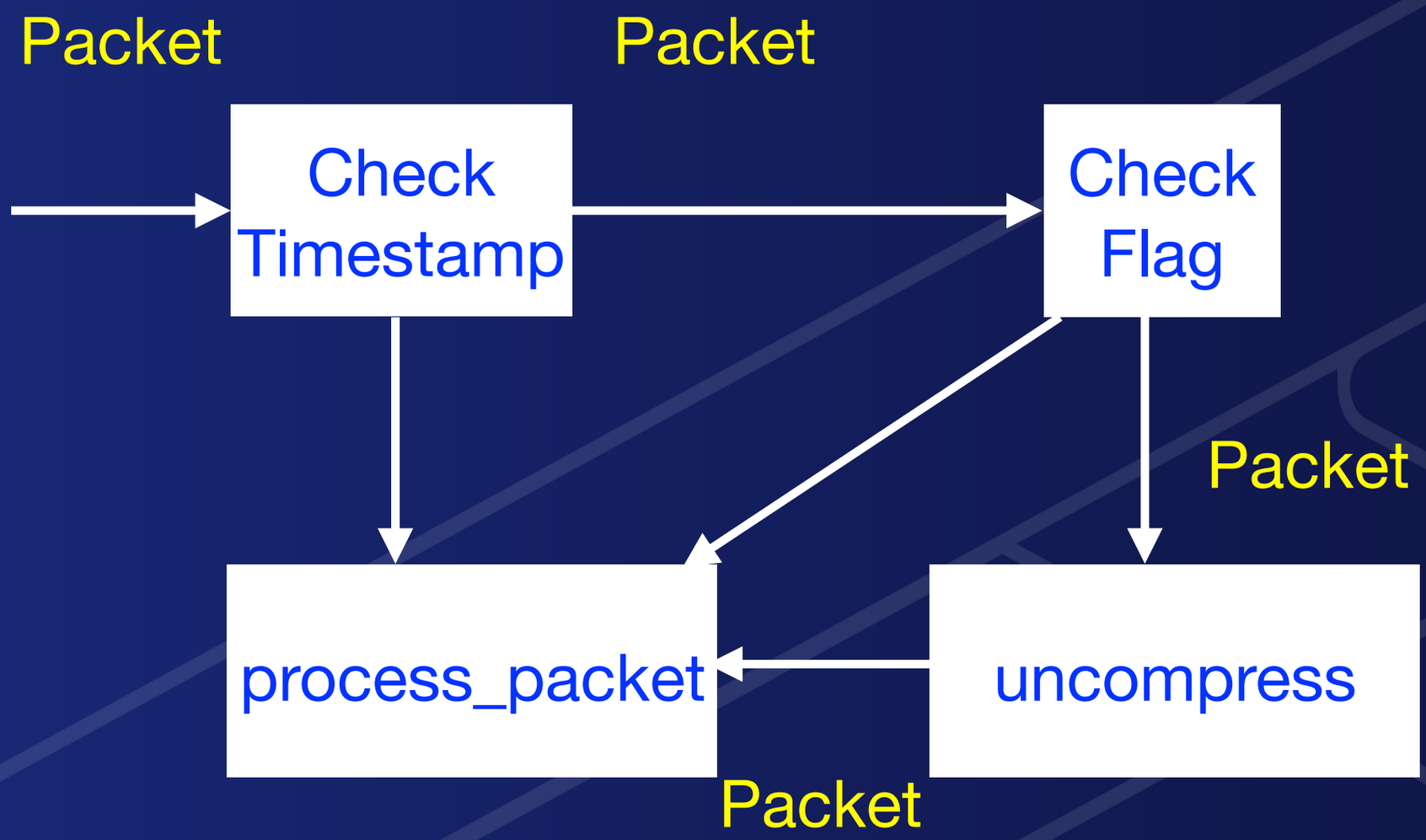
Coupling vs. Cohesion

Compose small, cohesive program units



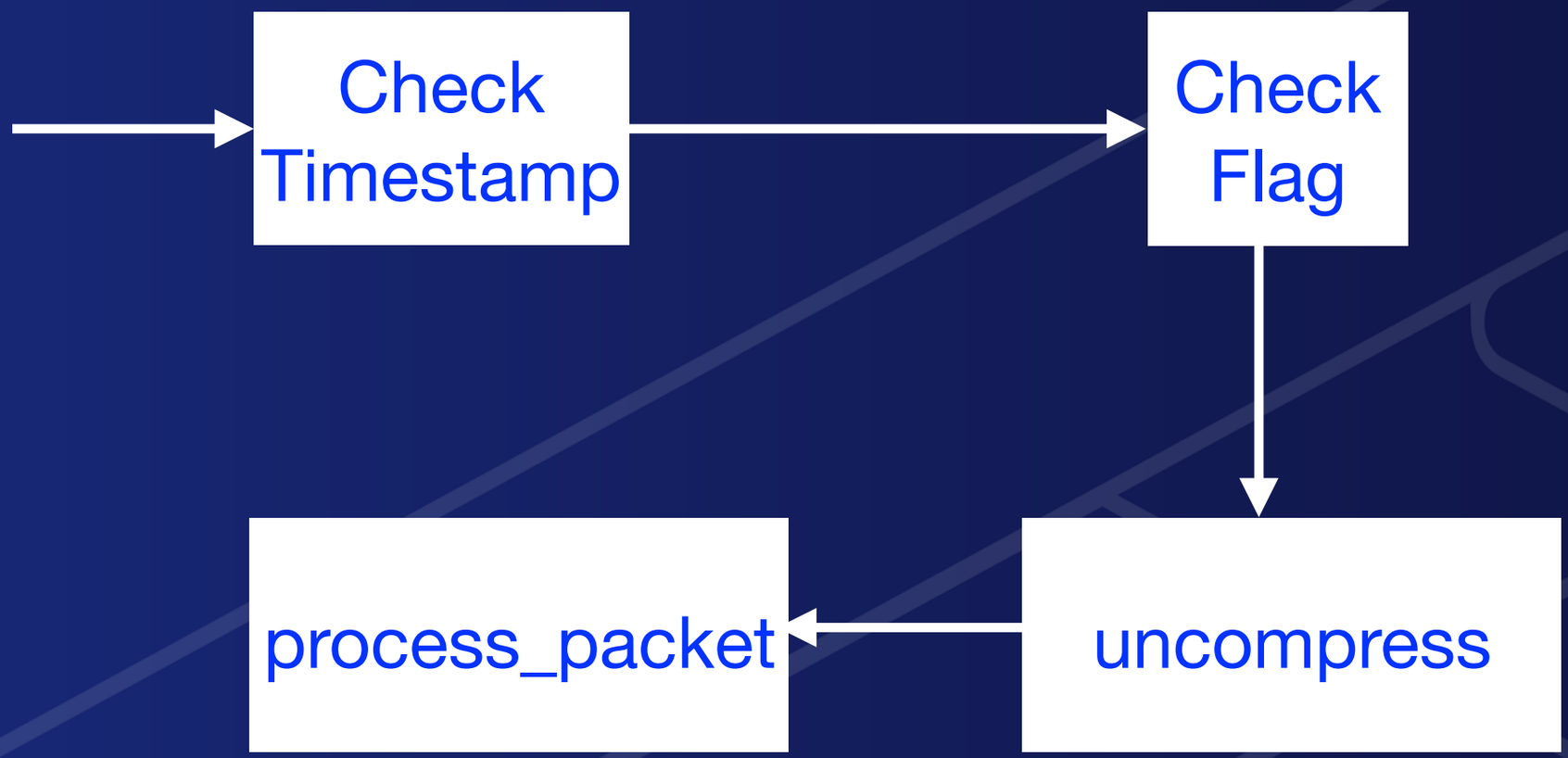
Coupling vs. Cohesion

Compose small, cohesive program units



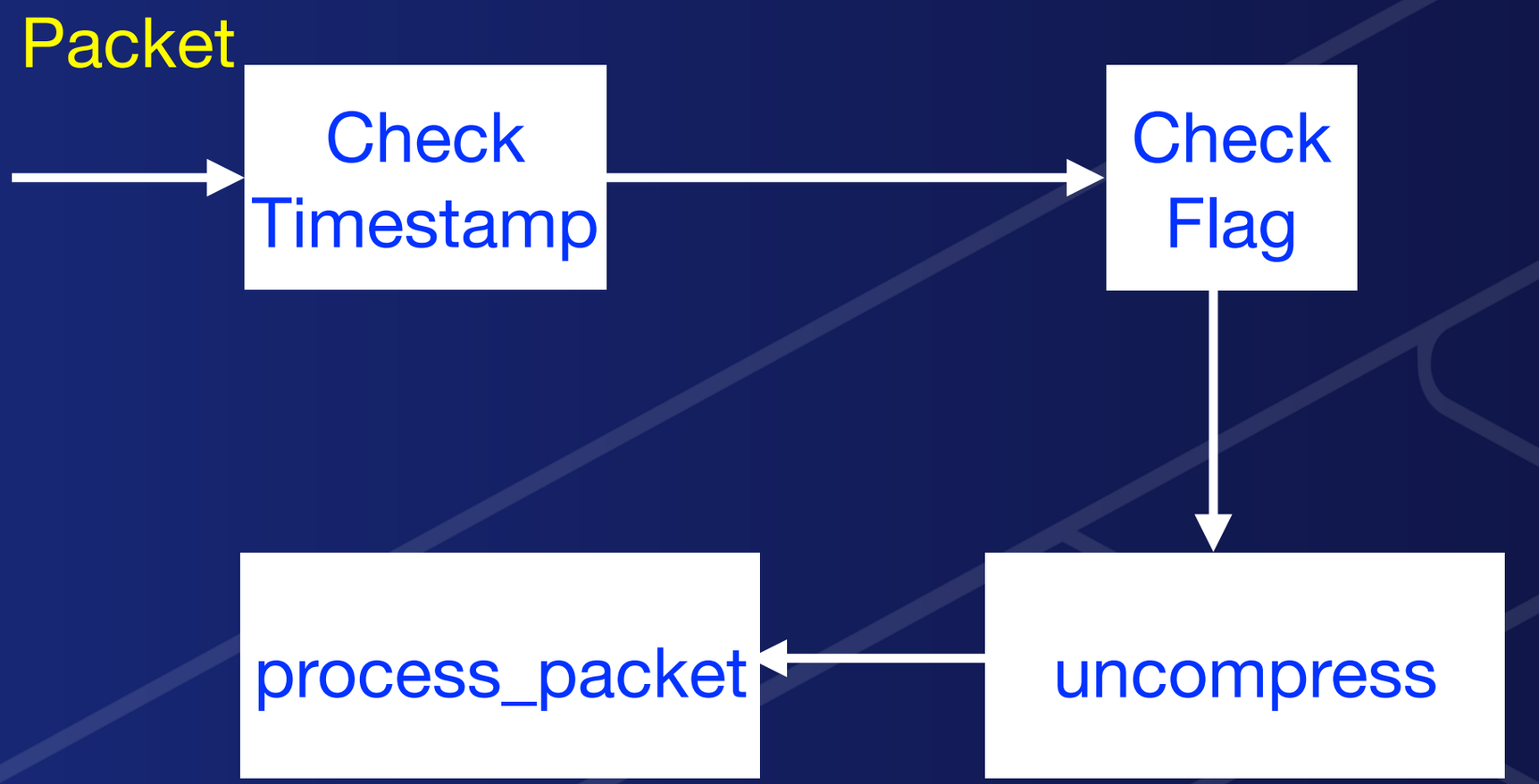
Coupling vs. Cohesion

Compose small, cohesive program units



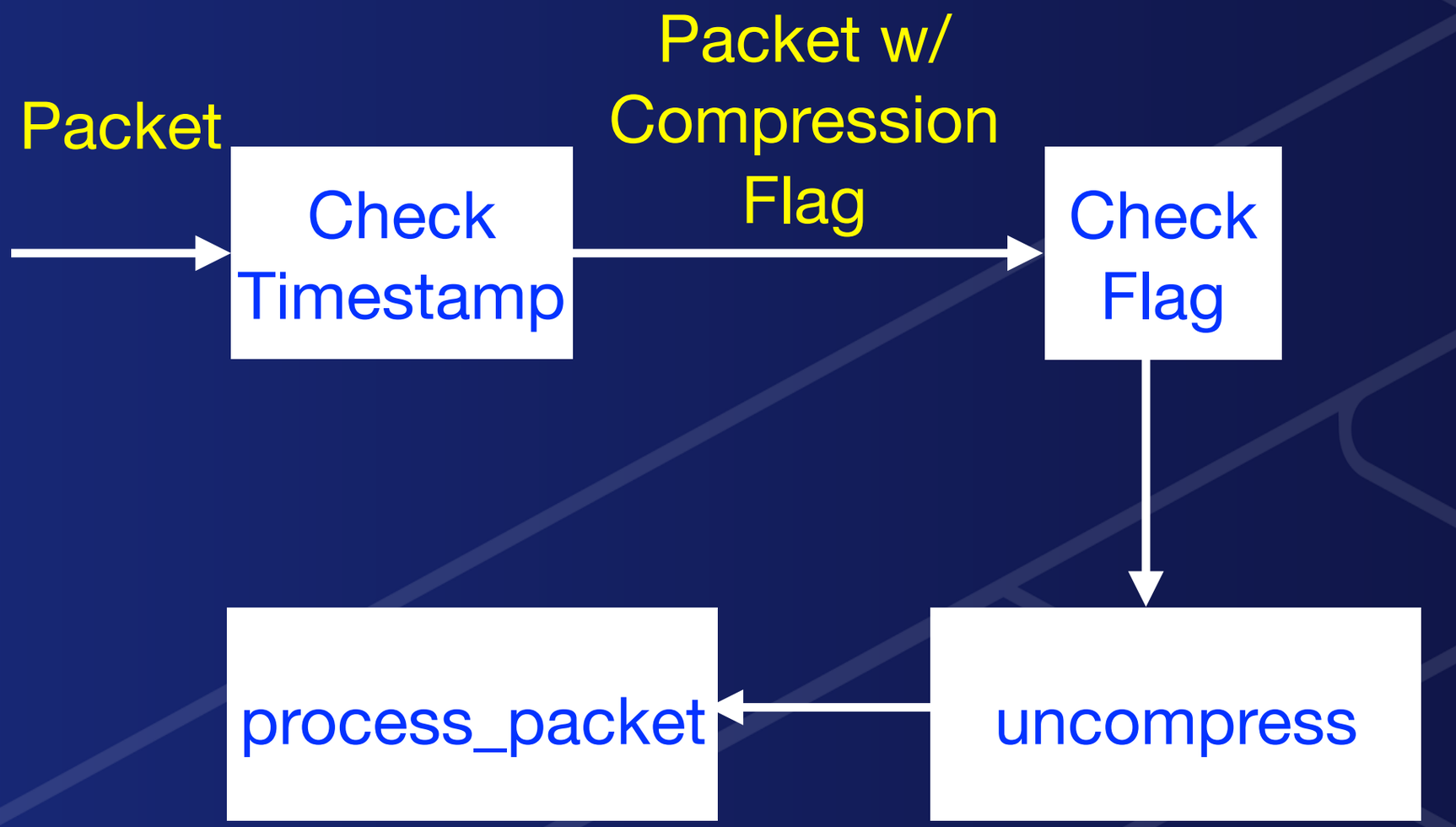
Coupling vs. Cohesion

Compose small, cohesive program units



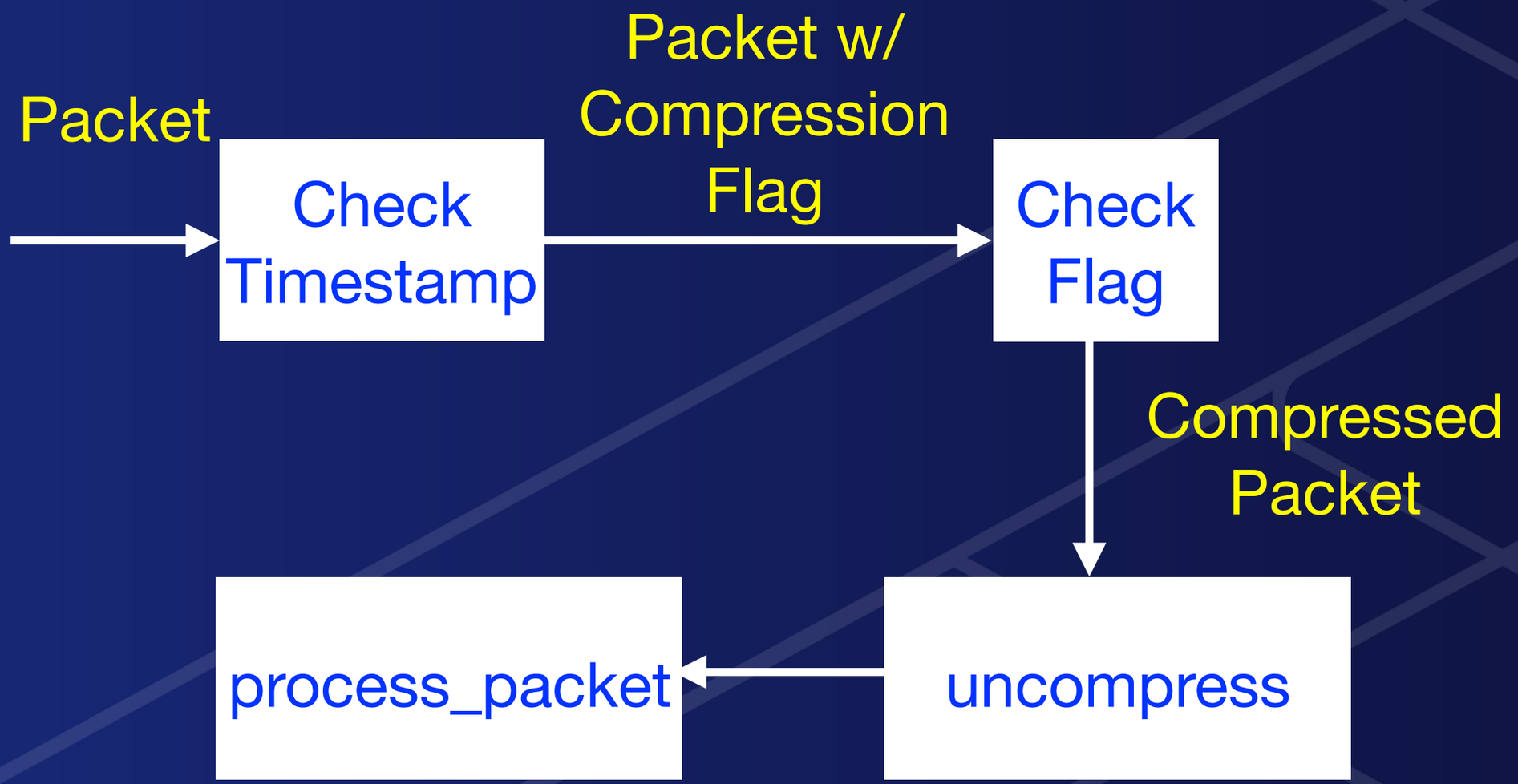
Coupling vs. Cohesion

Compose small, cohesive program units



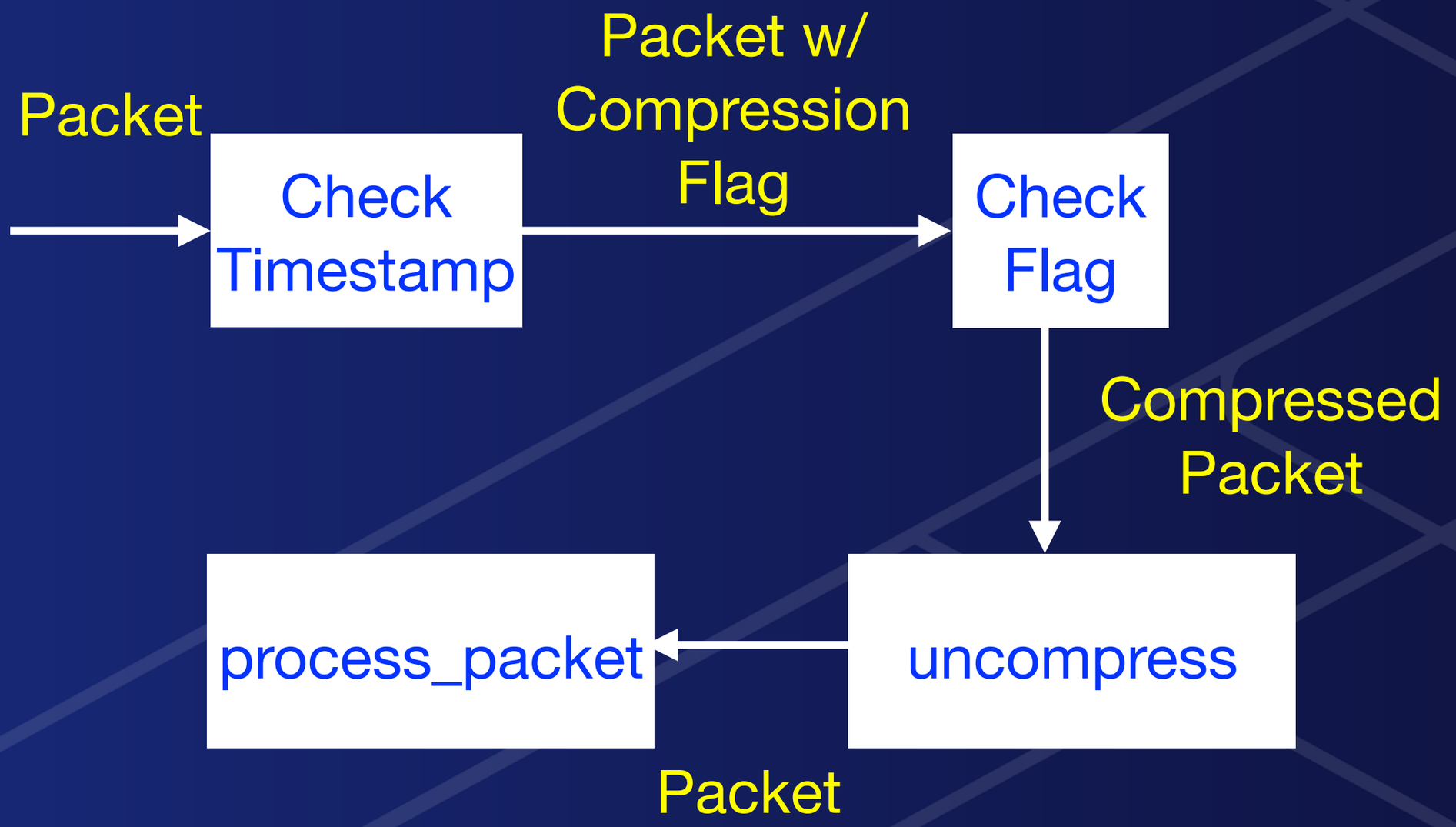
Coupling vs. Cohesion

Compose small, cohesive program units



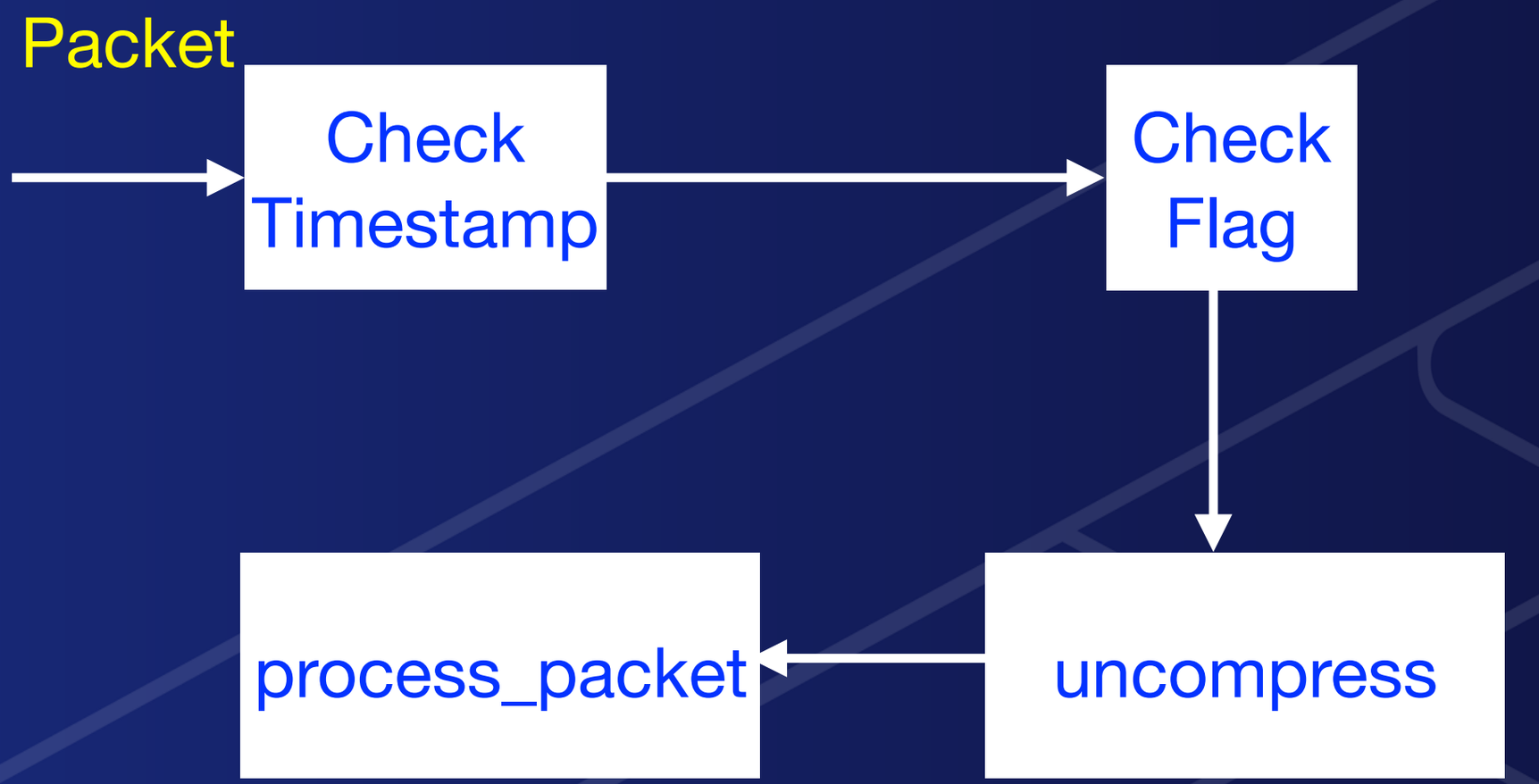
Coupling vs. Cohesion

Compose small, cohesive program units



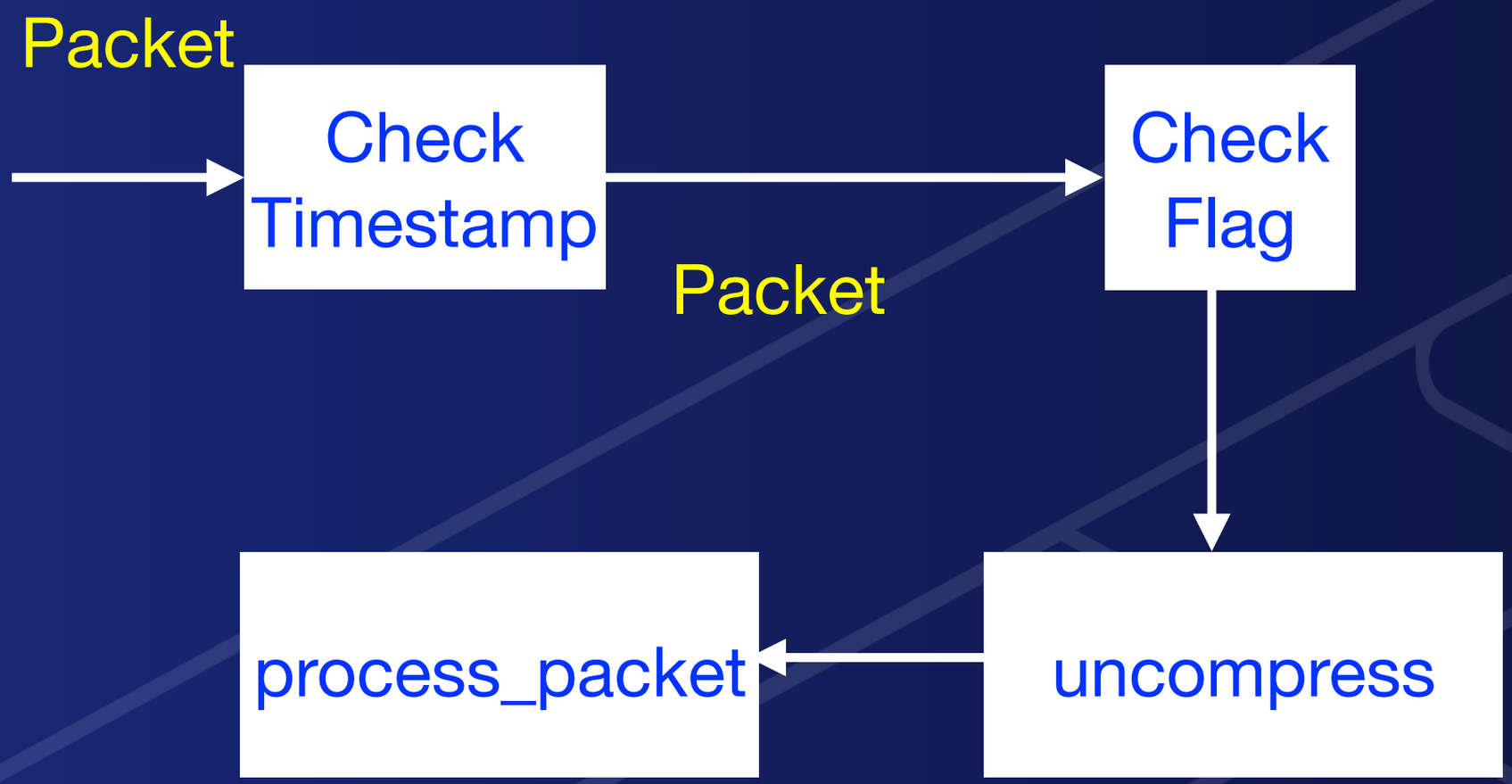
Coupling vs. Cohesion

Compose small, cohesive program units



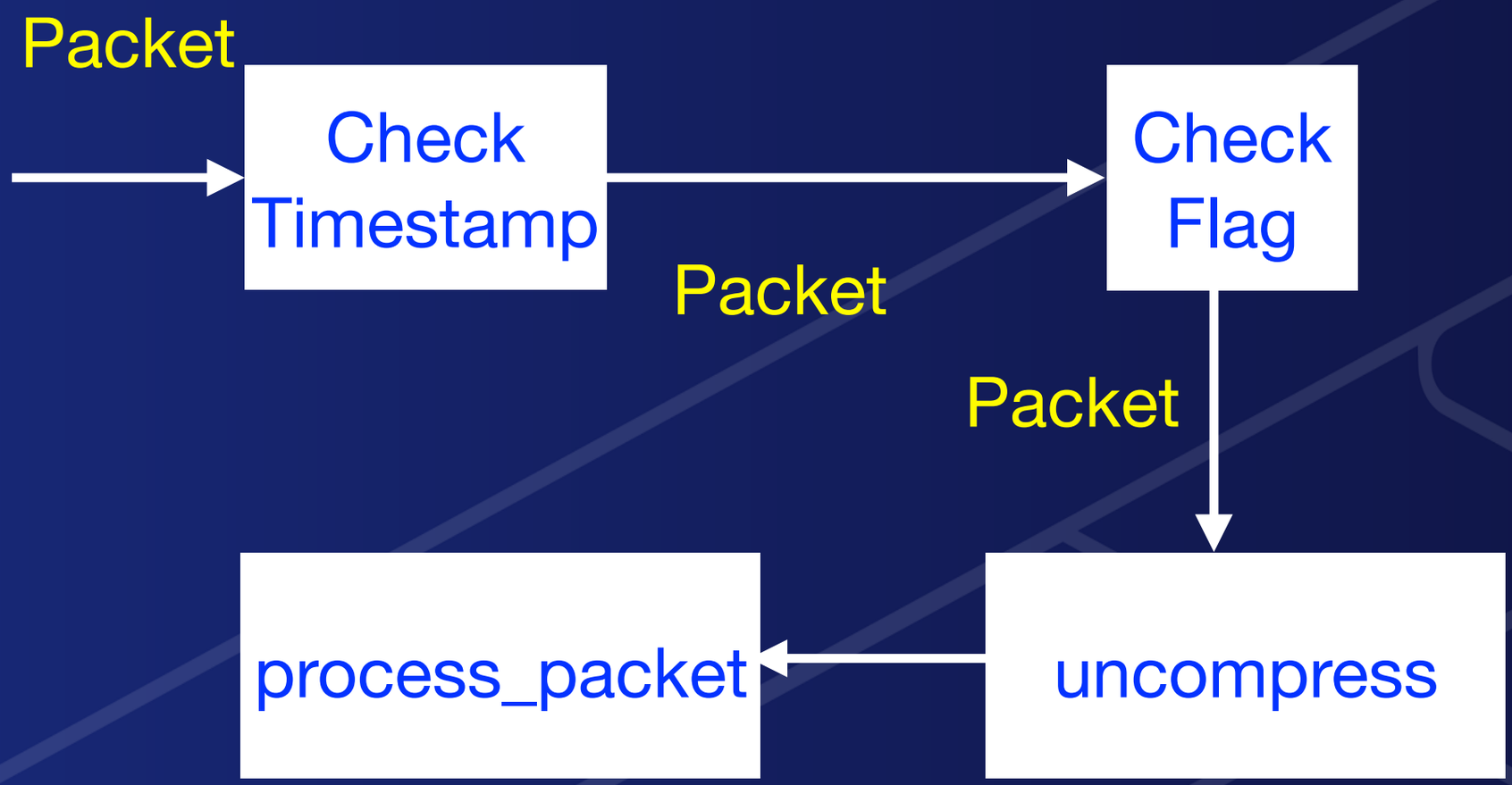
Coupling vs. Cohesion

Compose small, cohesive program units



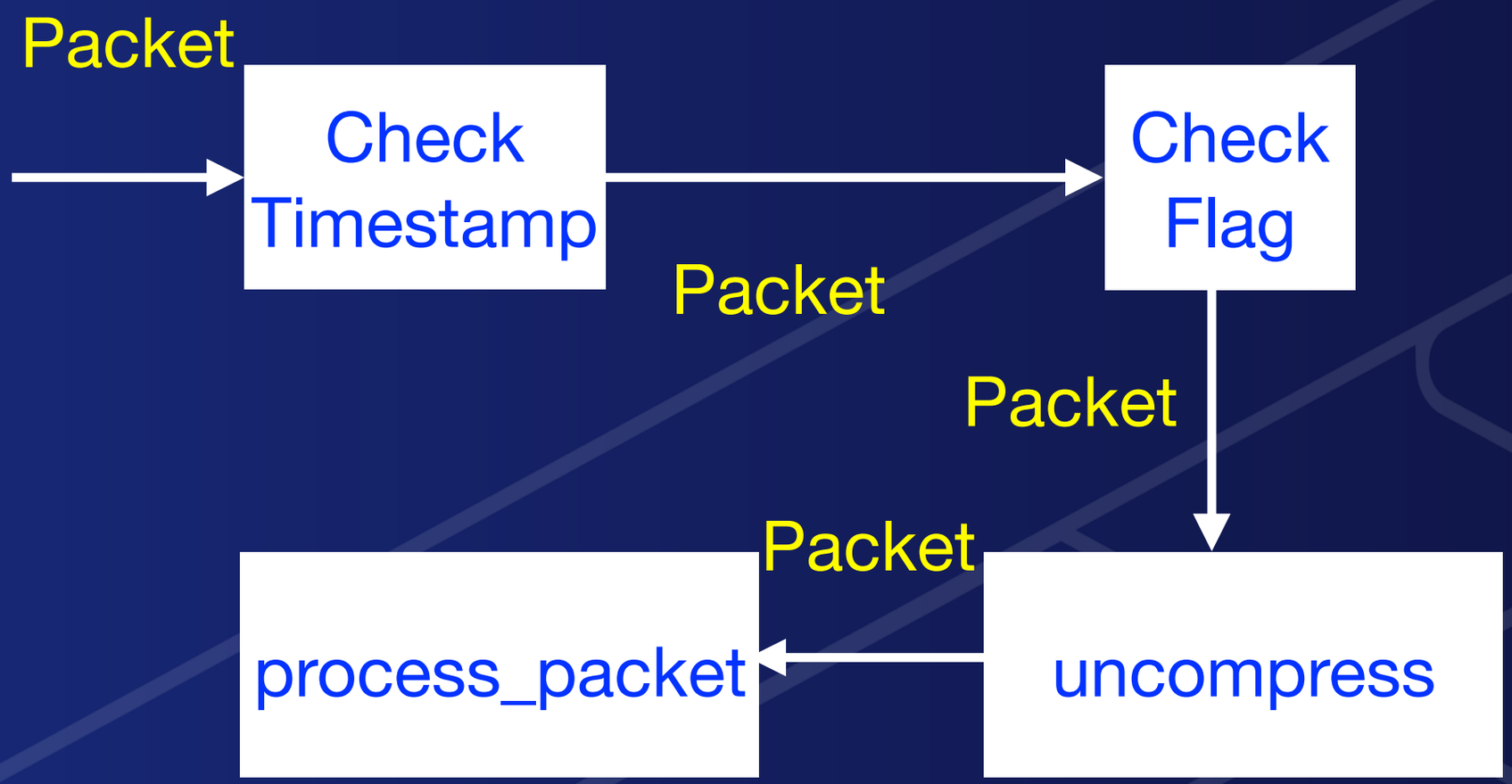
Coupling vs. Cohesion

Compose small, cohesive program units



Coupling vs. Cohesion

Compose small, cohesive program units



A Solution

A Solution

- A Particular Design and How to Implement that Design

A Solution

- A Particular Design and How to Implement that Design
- Emphasis on the former

A Solution

- A Particular Design and How to Implement that Design
- Emphasis on the former
- Less than I want on the latter

A Solution

- A Particular Design and How to Implement that Design
- Emphasis on the former
- Less than I want on the latter
- Key concept - adding a feature should require zero/little change to the existing implementation

A Solution

- A Particular Design and How to Implement that Design
- Emphasis on the former
- Less than I want on the latter
- Key concept - adding a feature should require zero/little change to the existing implementation
- Very low coupling, very high cohesion

A Solution

- A Particular Design and How to Implement that Design
- Emphasis on the former
- Less than I want on the latter
- Key concept - adding a feature should require zero/little change to the existing implementation
- Very low coupling, very high cohesion
- Recount parts of a 32 year quest

Journey Before Destination

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say.

The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

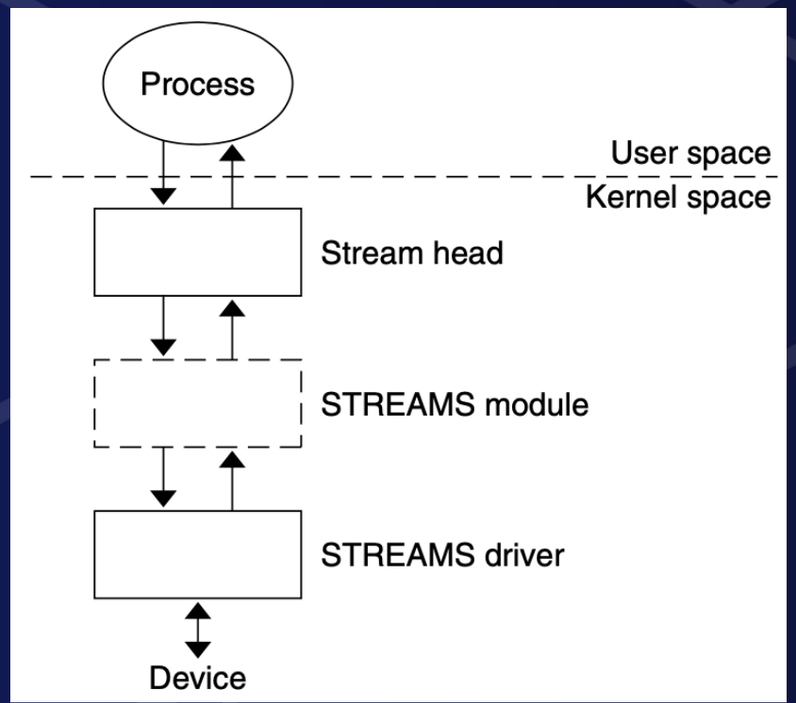
Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

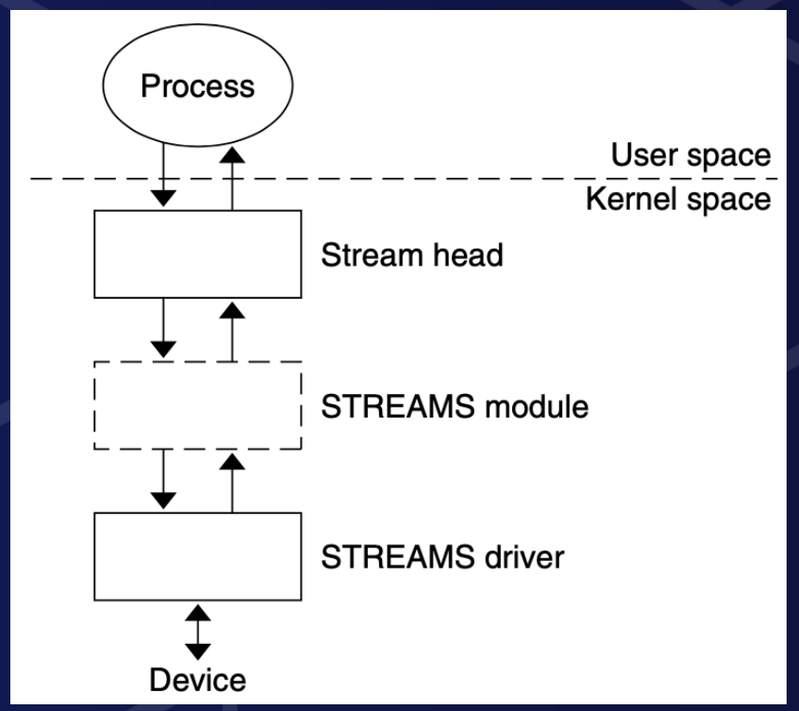
-BRANDON SANDERSON, *OATHBRINGER*

A Three-Way Love Affair

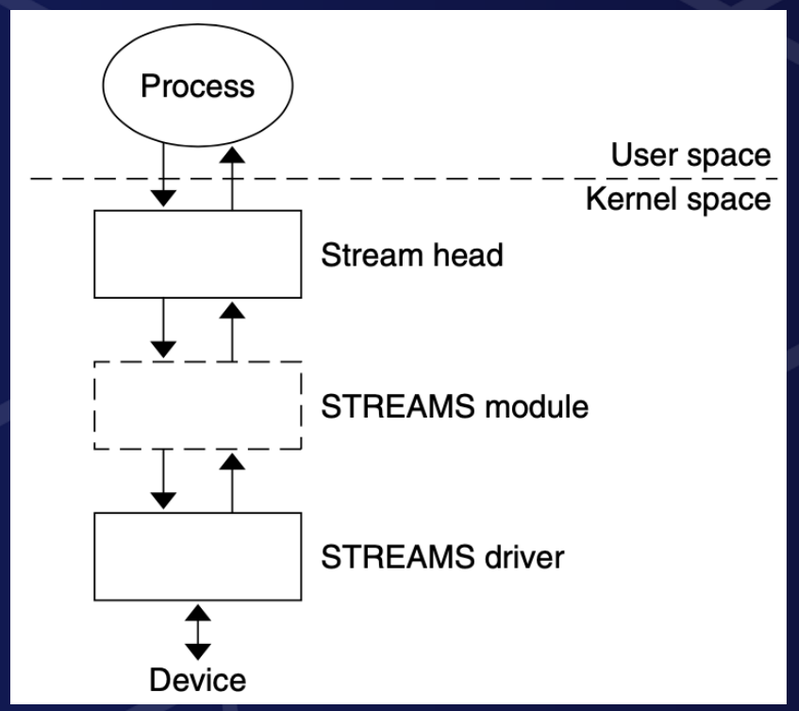
A Three-Way Love Affair



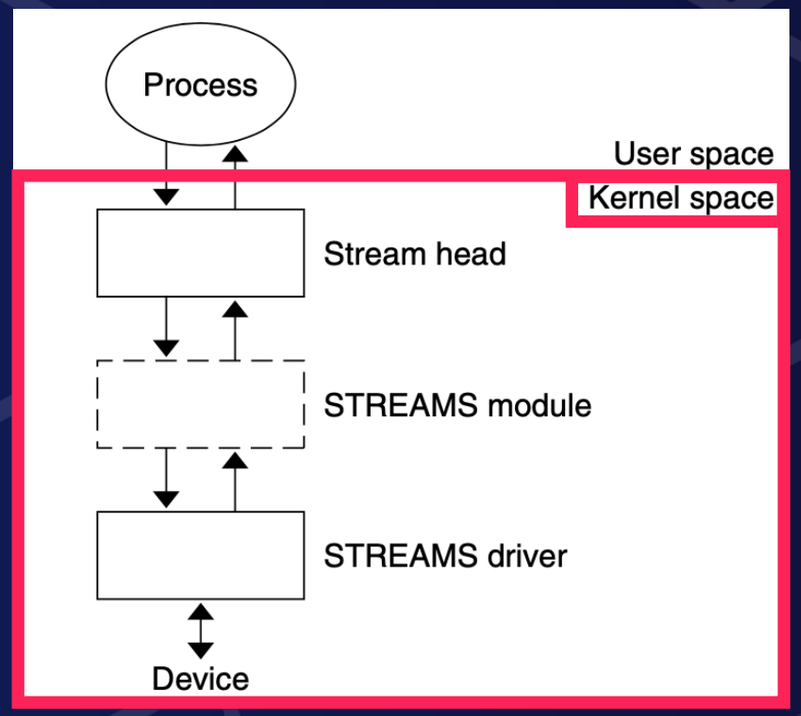
A Three-Way Love Affair



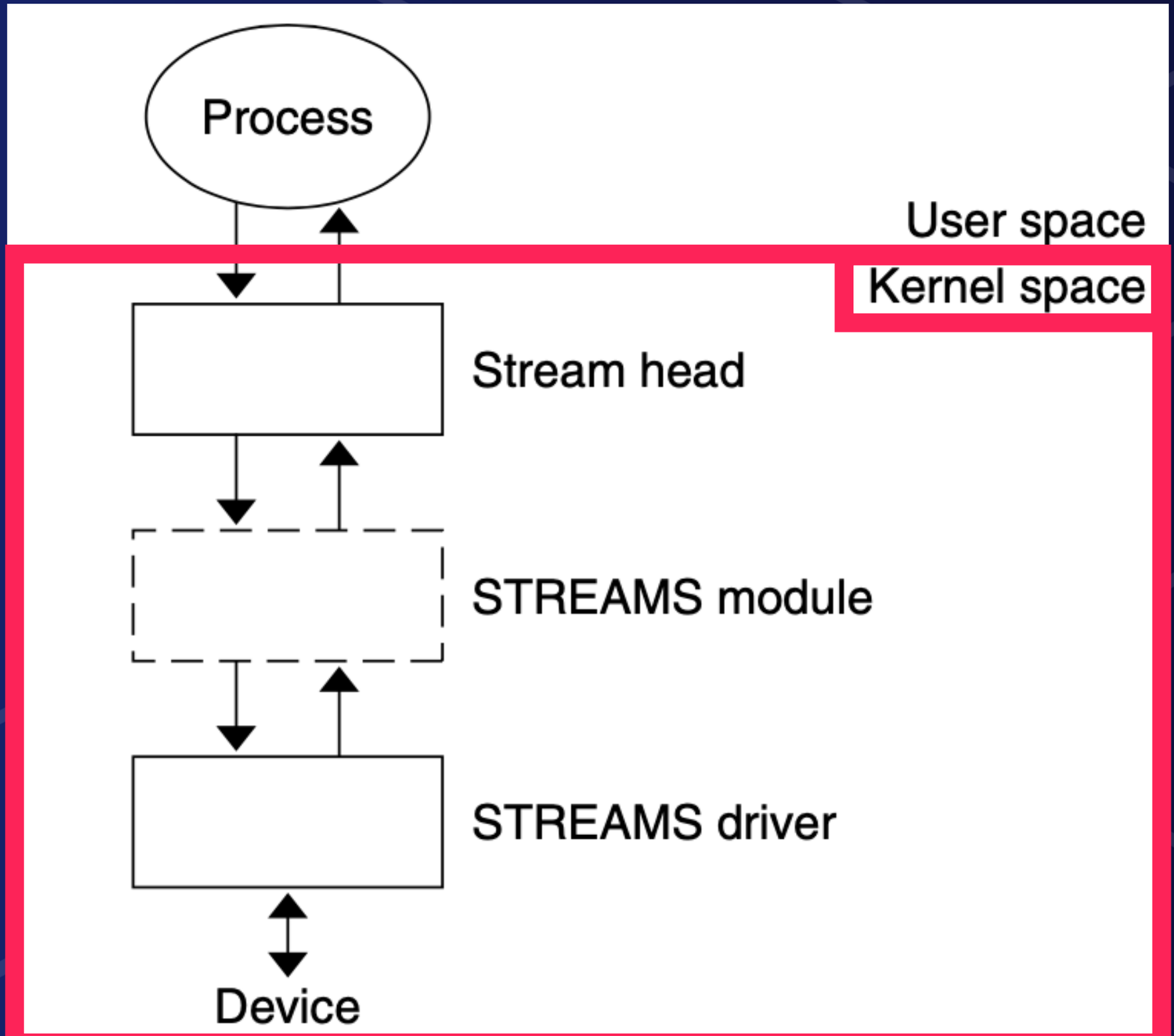
A Three-Way Love Affair



A Three-Way Love Affair



A Three-Way Love Affair



A Beginning...

A Beginning...

tmp = a
a = b
b = tmp

A Beginning...

```
tmp := a;
a := b;
b := tmp;
```

My Early Use of C++

- c / cfront
- get off punch cards

Grad School and the AI Winter

Grad School and Publications

Hagins, Biswas, Yu, Model-Based Diagnosis in the Process-Ontology Framework,
The Second AAAI Workshop on Model Based Reasoning, Boston, MA, July 1990

Biswas, Yu, Hagins, Strobel, Kendall, Cannon, Bezdek, An Efficient Scheme for Characterizing Hydrocarbon Plays for Analogical Analysis
AAPG Annual Convention, San Francisco, CA, June 1990

Biswas, Yu, Hagins, Strobel, Kendall, Cannon, Bezdek, PLAYMAKER: A Knowledge-Based Approach to Characterizing Hydrocarbon Plays
Applications of AI VIII (SPIE), Orlando, FL, April 1990

Biswas, Strobel, Hagins, Kendall, Cannon, Bezdek, An Associational Scheme for Characterizing Hydrocarbon Plays for Analogical Reasoning
IEEE Expert, March 1990

Biswas, Hagins, Debelak, Qualitative Modeling in Engineering Applications
1989 IEEE Conference on Systems, Man, and Cybernetics, Cambridge, MA, November 1989

Weinberg, Hagins, Biswas, Extending Temporal Reasoning in Process-Oriented Qualitative Reasoning
Proceedings of IJCAI-89 Workshop on Model Based Reasoning, Detroit, MI, August 1989

Debelak, Biswas, Hagins, Qualitative Modeling in Chemical Engineering Applications
American Institute of Chemical Engineers: 1989 Summer National Meeting, August 20, 1989

Grad School and Winning the Lottery

SVR4 SMP Unix Kernel

Cool Stuff!!!

Siemens Stromberg-Carlson

Goal

Goal

- Take set of applications running one host

Goal

- Take set of applications running one host
- Add feature(s) so existing applications can be distributed across any number of hosts

Goal

- Take set of applications running one host
- Add feature(s) so existing applications can be distributed across any number of hosts
- Original plan was to design/implement a distributed architecture. Massive code changes.

Goal

- Take set of applications running one host
- Add feature(s) so existing applications can be distributed across any number of hosts
- Original plan was to design/implement a distributed architecture. Massive code changes.
- My goal - add feature with little or no change to existing design and/or implementation

Goal

- Take set of applications running one host
- Add feature(s) so existing applications can be distributed across any number of hosts
- Original plan was to design/implement a distributed architecture. Massive code changes.
- My goal - add feature with little or no change to existing design and/or implementation
- Fortunately, they had used System V IPC

System V IPC

Message queues

System V message queues allow data to be exchanged in units called messages. Each messages can have an associated priority, POSIX message queues provide an alternative API for achieving the same result; see mq_overview(7).

The System V message queue API consists of the following system calls:

msgget(2)
: Create a new message queue or obtain the ID of an existing message queue. This call returns an identifier that is used in the remaining APIs.

msgsnd(2)
: Add a message to a queue.

msgrcv(2)
: Remove a message from a queue.

msgctl(2)
: Perform various control operations on a queue, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

Semaphore sets

System V semaphores allow processes to synchronize their actions. System V semaphores are allocated in groups called sets; each semaphore in a set is a counting semaphore. POSIX semaphores provide an alternative API for achieving the same result; see sem_overview(7).

The System V semaphore API consists of the following system calls:

semget(2)
: Create a new set or obtain the ID of an existing set. This call returns an identifier that is used in the remaining APIs.

semop(2)
: Perform operations on the semaphores in a set.

semctl(2)
: Perform various control operations on a set, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

Shared memory segments

System V shared memory allows processes to share a region a memory (a "segment"). POSIX shared memory is an alternative API for achieving the same result; see shm_overview(7).

The System V shared memory API consists of the following system calls:

shmget(2)
: Create a new segment or obtain the ID of an existing segment. This call returns an identifier that is used in the remaining APIs.

shmat(2)
: Attach an existing shared memory object into the calling process's address space.

shmdt(2)
: Detach a segment from the calling process's address space.

shmctl(2)
: Perform various control operations on a segment, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

Message queues

System V message queues allow data to be exchanged in units called messages. Each messages can have an associated priority, POSIX message queues provide an alternative API for achieving the same result; see mq_overview(7).

The System V message queue API consists of the following system calls:

msgget(2)
: Create a new message queue or obtain the ID of an existing message queue. This call returns an identifier that is used in the remaining APIs.

msgsnd(2)
: Add a message to a queue.

msgrcv(2)
: Remove a message from a queue.

msgctl(2)
: Perform various control operations on a queue, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

Semaphore sets

System V semaphores allow processes to synchronize their actions. System V semaphores are allocated in groups called sets; each semaphore in a set is a counting semaphore. POSIX semaphores provide an alternative API for achieving the same result; see sem_overview(7).

The System V semaphore API consists of the following system calls:

semget(2)
Create a new set or obtain the ID of an existing set. This call returns an identifier that is used in the remaining APIs.

semop(2)
Perform operations on the semaphores in a set.

semctl(2)
Perform various control operations on a set, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

System V IPC

Shared memory segments

System V shared memory allows processes to share a region a memory (a "segment"). POSIX shared memory is an alternative API for achieving the same result; see shm_overview(7).

The System V shared memory API consists of the following system calls:

shmget(2)
: Create a new segment or obtain the ID of an existing segment. This call returns an identifier that is used in the remaining APIs.

shmat(2)
: Attach an existing shared memory object into the calling process's address space.

shmdt(2)
: Detach a segment from the calling process's address space.

shmctl(2)
: Perform various control operations on a segment, including deletion.

https://man7.org/linux/man-pages/man7/sysvipc.7.html

Message Queue

```c
int msgget(key_t key, int msgflg)
{ /* implementation */ }

int msgsnd(int msqid, const void *msgp, size_t msgsz,
    int msgflg)
{ /* implementation */ }

ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long
msgtyp, int msgflg)
{ /* implementation */ }

int msgctl(int msqid, int cmd, struct msqid_ds *buf)
{ /* implementation */ }
```

Semaphore

```c
int semget(key_t key, int nsems, int semflg)
{ /* implementation */ }

int semop(int semid, struct sembuf *sops, size_t nsops)
{ /* implementation */ }

int semctl(int semid, int semnum, int cmd, ...)
{ /* implementation */ }
```

Relink

```
g++ blah blah blah -ldcom.a
```

Relink

```
g++ blah blah blah -ldcom.a
```

Kernel STREAMS module: DCOM

Relink

```
g++ blah blah blah -ldcom.a
```

Kernel STREAMS module: DCOM

But, even without the STREAMS implementation,
we are left with a stunning result

Relink

```
g++ blah blah blah -ldcom.a
```

Kernel STREAMS module: DCOM

But, even without the STREAMS implementation,
we are left with a stunning result

The original plan was to rewrite

Relink

```
g++ blah blah blah -ldcom.a
```

Kernel STREAMS module: DCOM

But, even without the STREAMS implementation,
we are left with a stunning result

The original plan was to rewrite

We didn't change a single line of existing code!!!

Relink

```
g++ blah blah blah -ldcom.a
```

Kernel STREAMS module: DCOM

But, even without the STREAMS implementation,
we are left with a stunning result

The original plan was to rewrite

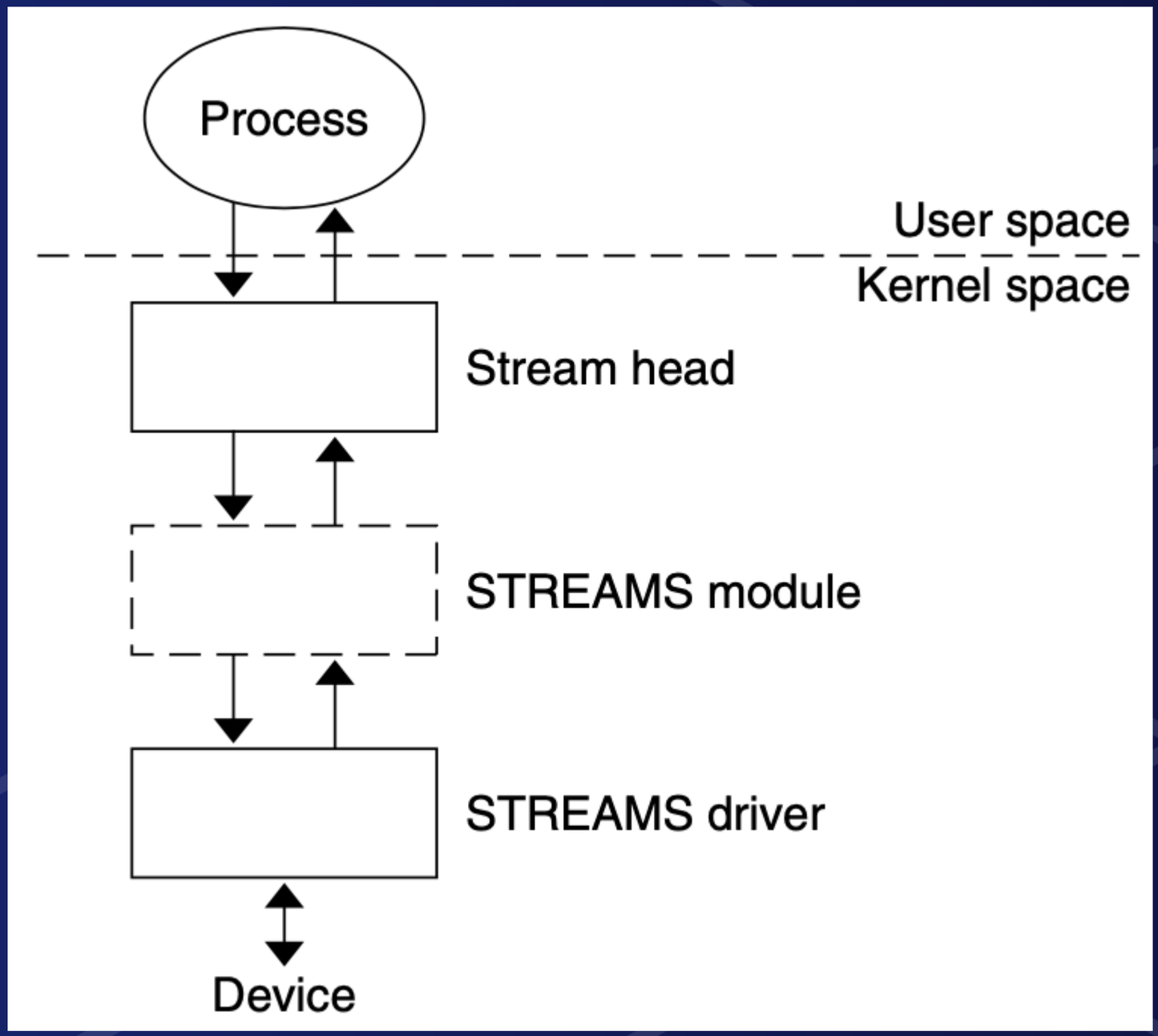
We didn't change a single line of existing code!!!

Coupling vs. Cohesion

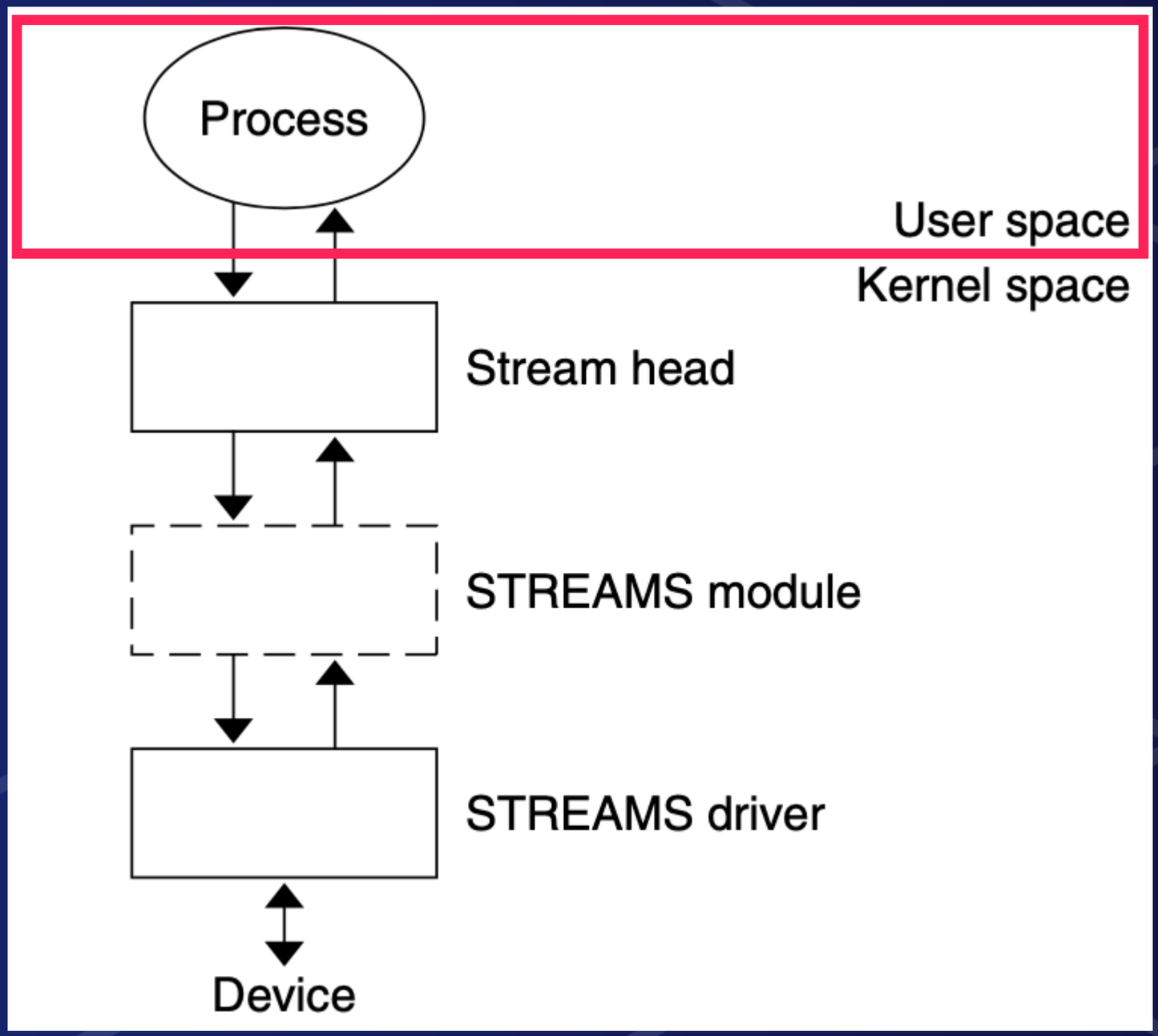
"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

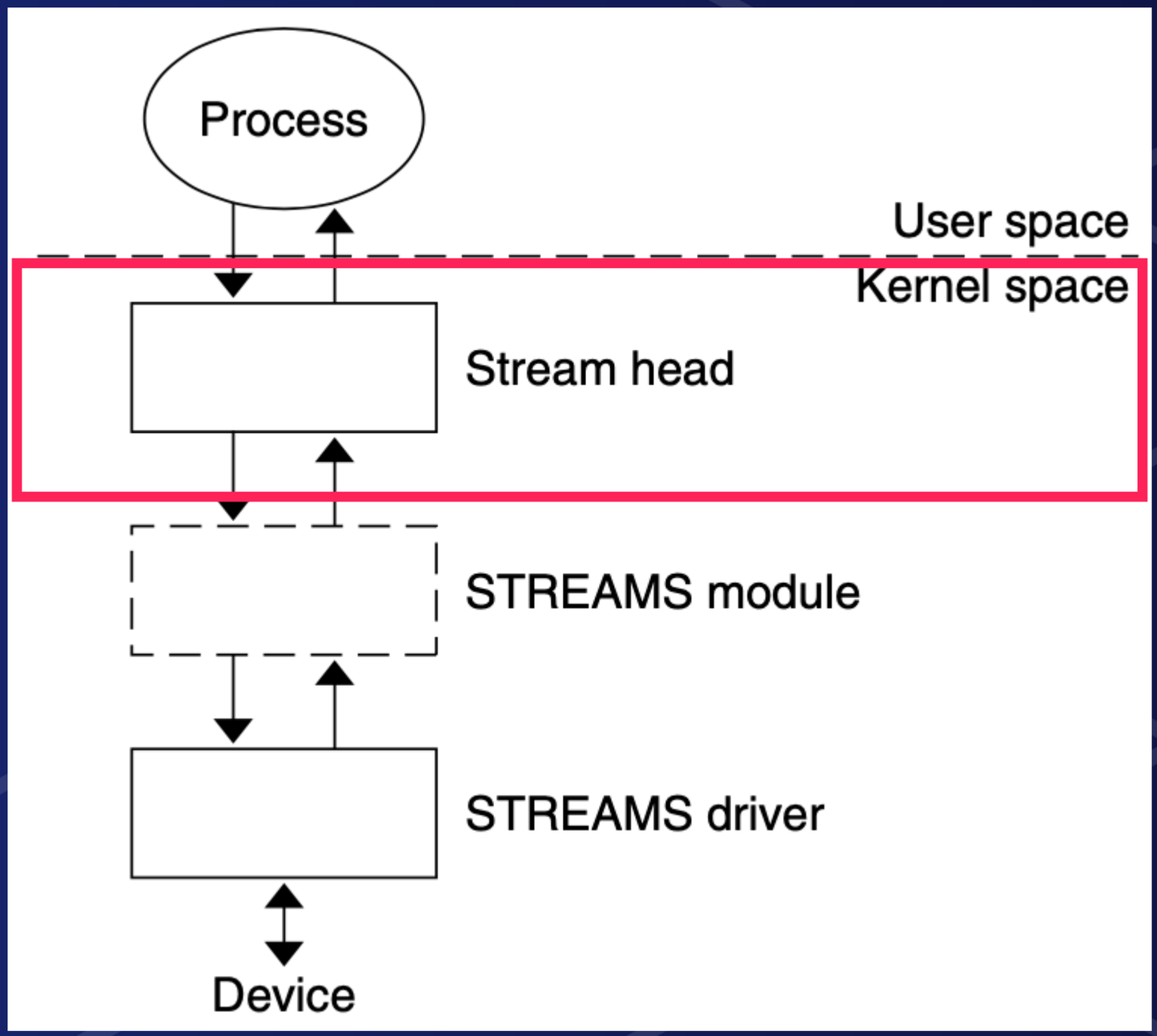
Love at First Sight



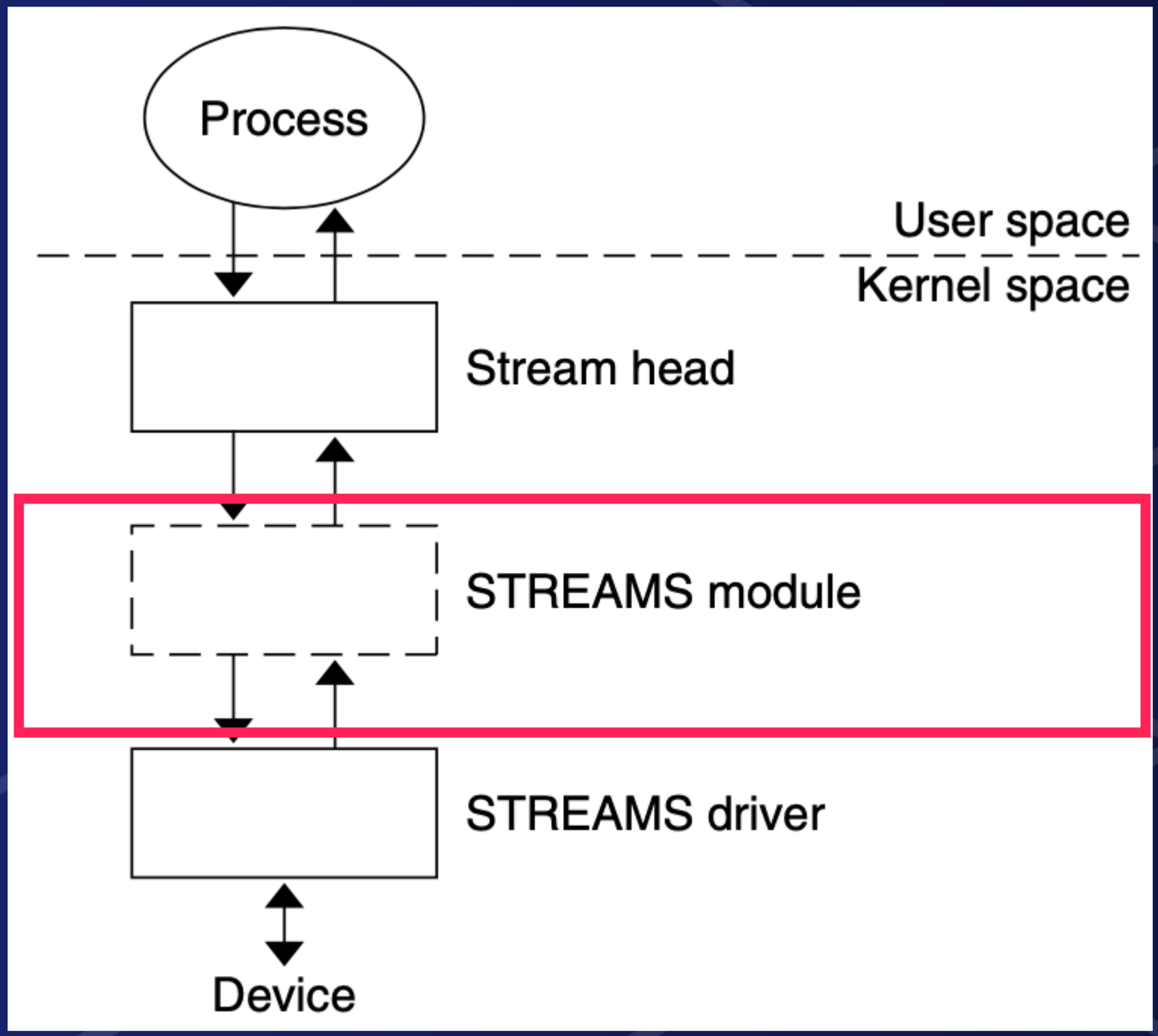
Love at First Sight



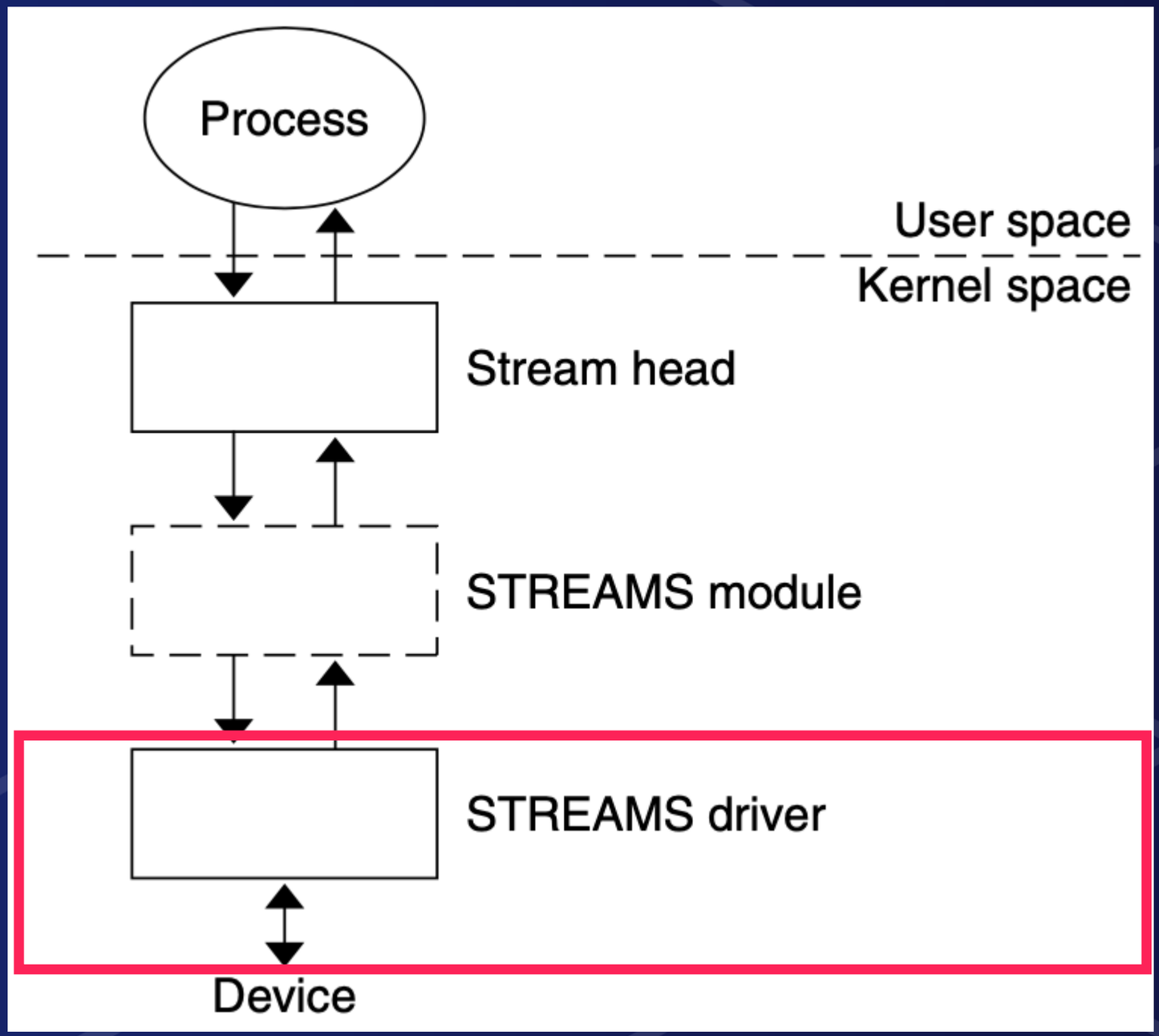
Love at First Sight



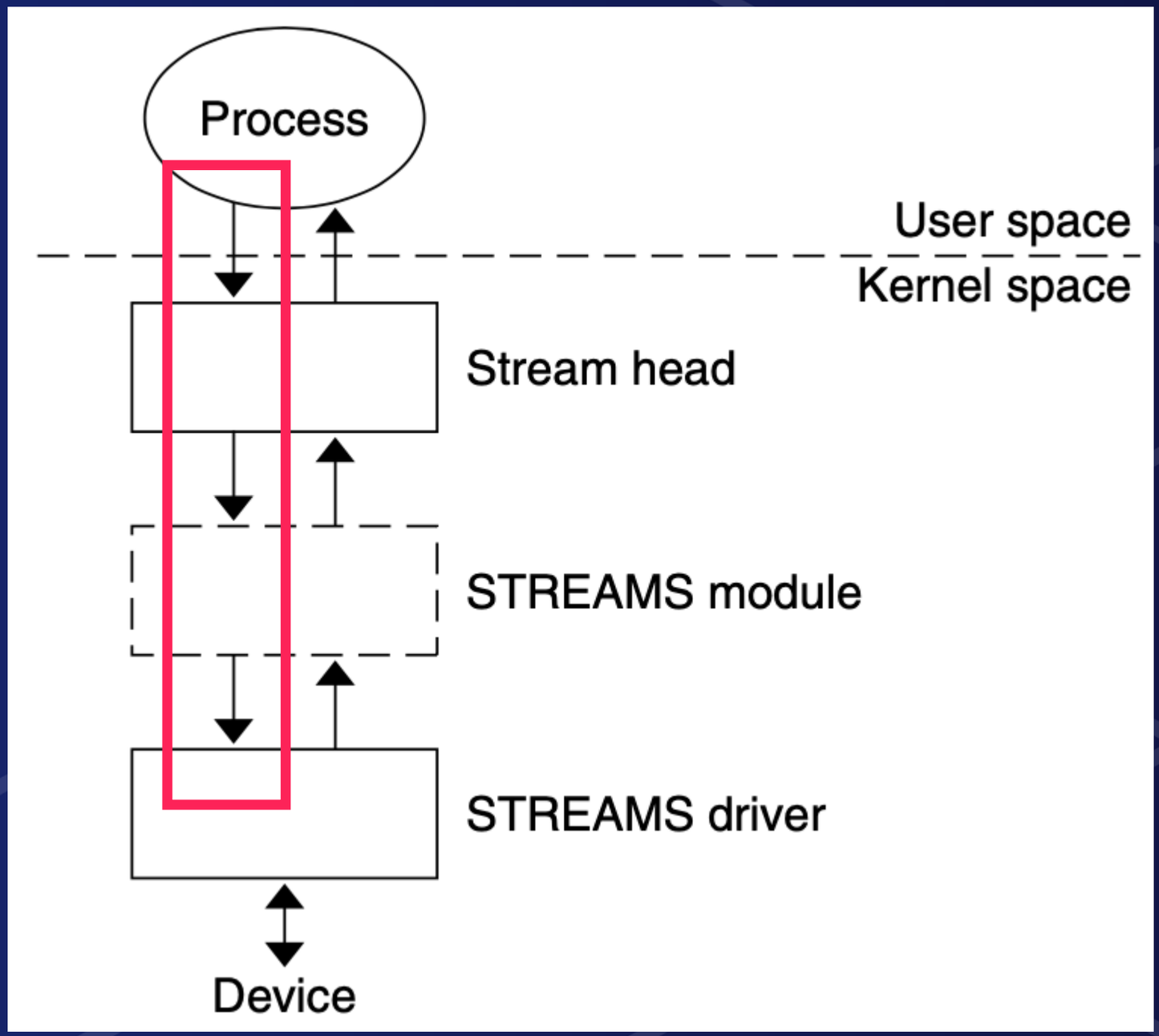
Love at First Sight



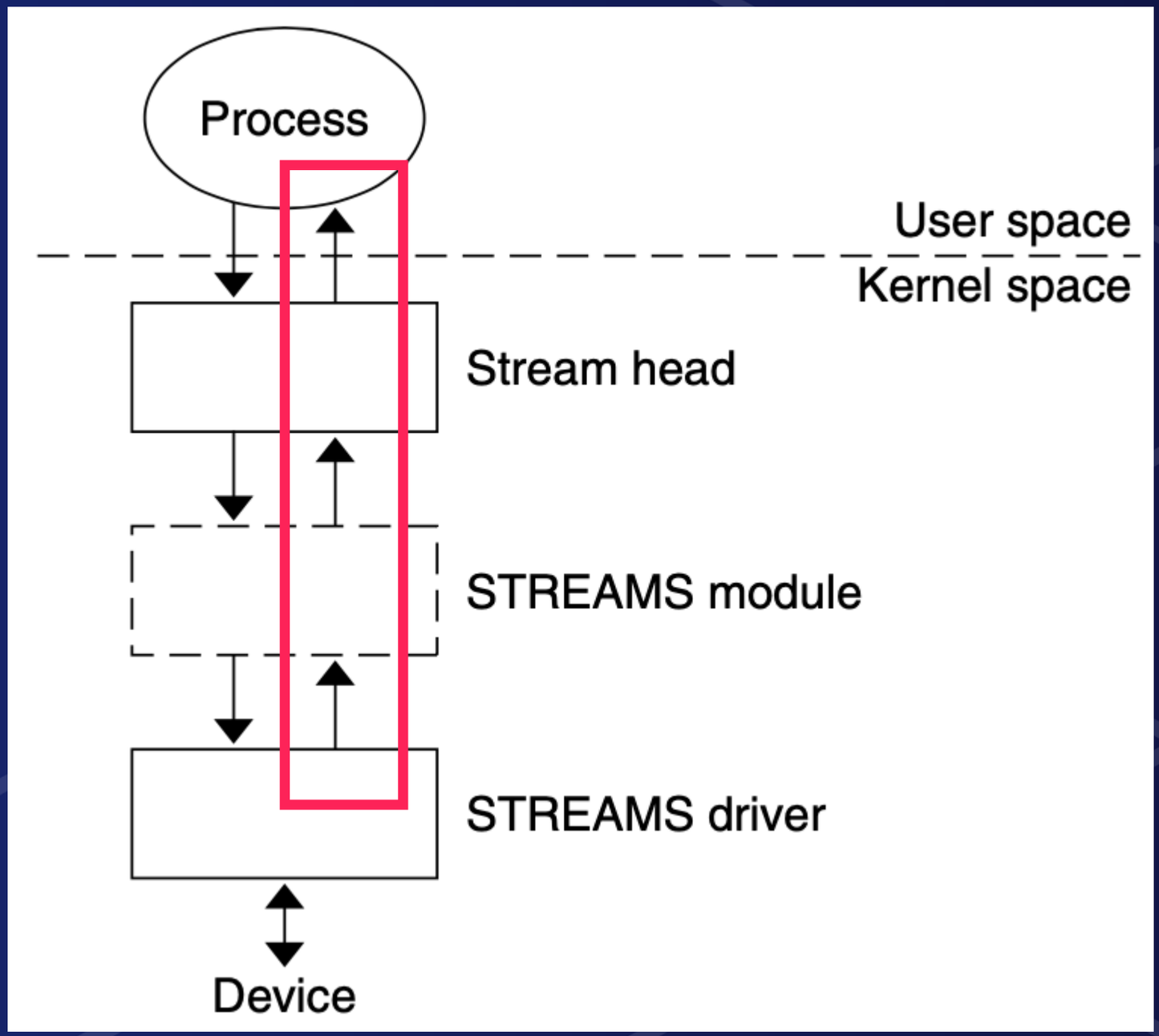
Love at First Sight



Love at First Sight



Love at First Sight



STREAMS Resources

STREAMS Resources

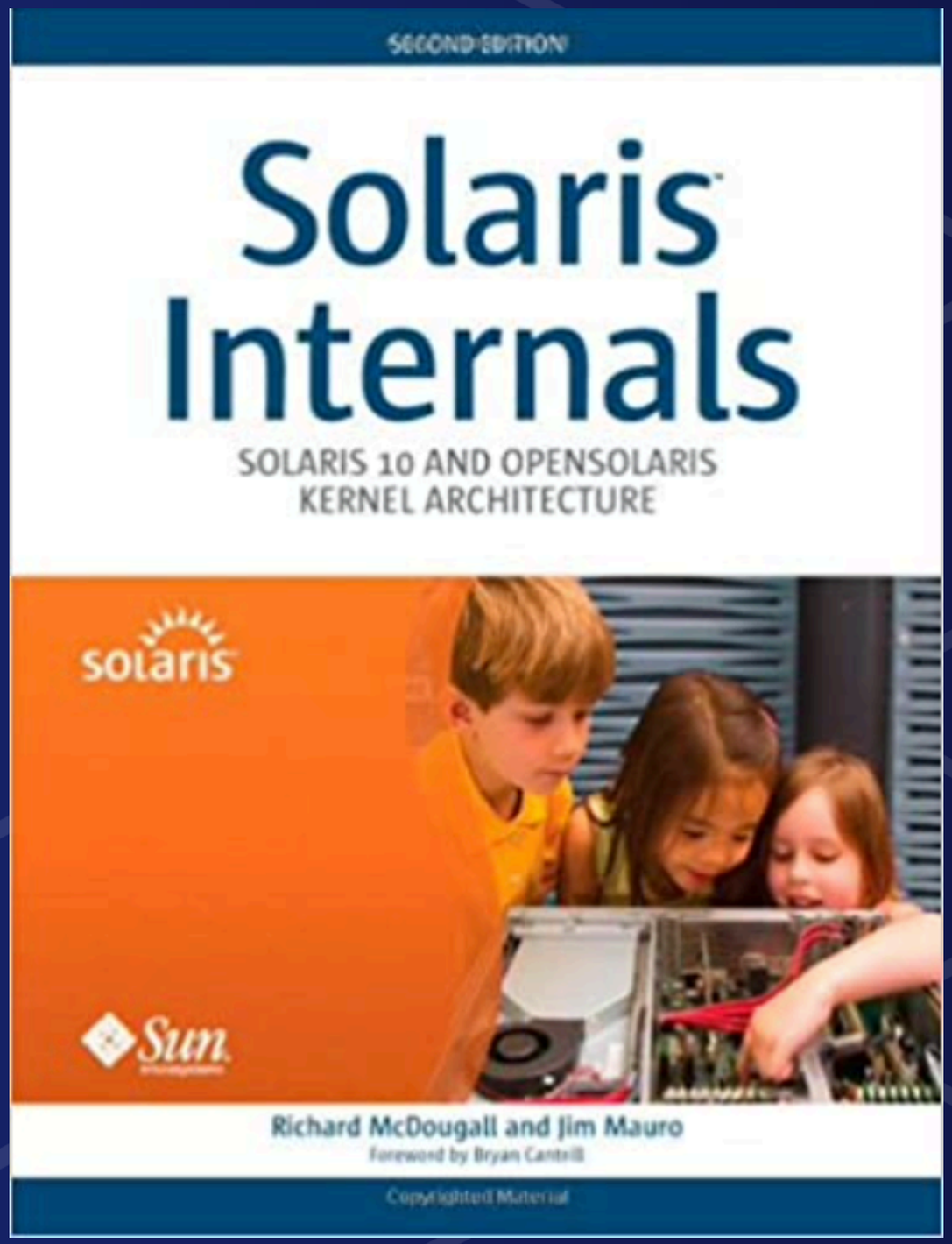
STREAMS Programming Guide - Oracle
https://docs.oracle.com/cd/E26502_01/html/E35856/index.html

STREAMS Resources

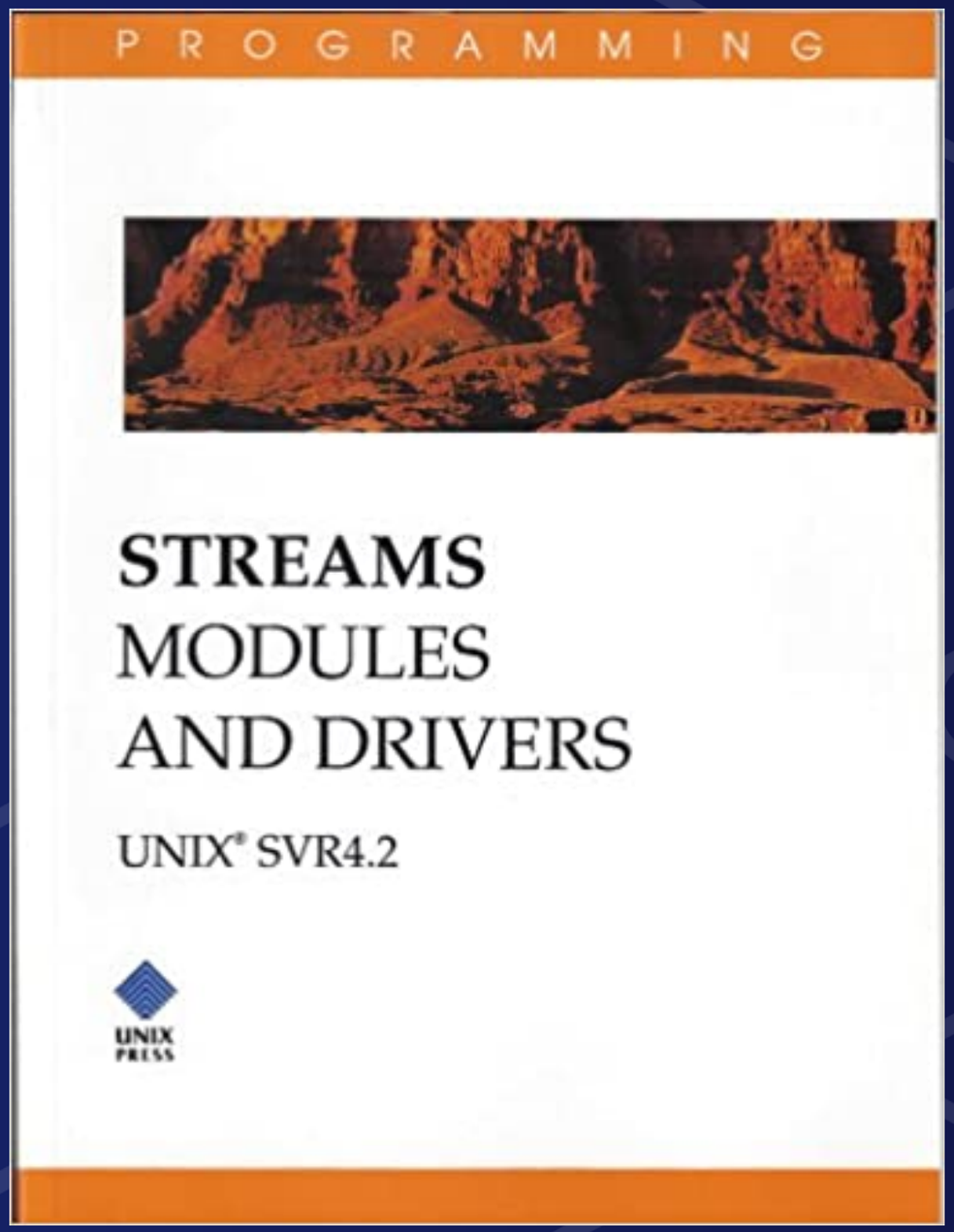
STREAMS Programming Guide - Oracle
https://docs.oracle.com/cd/E26502_01/html/E35856/index.html

Lots of other PDF resources available online
google is your friend

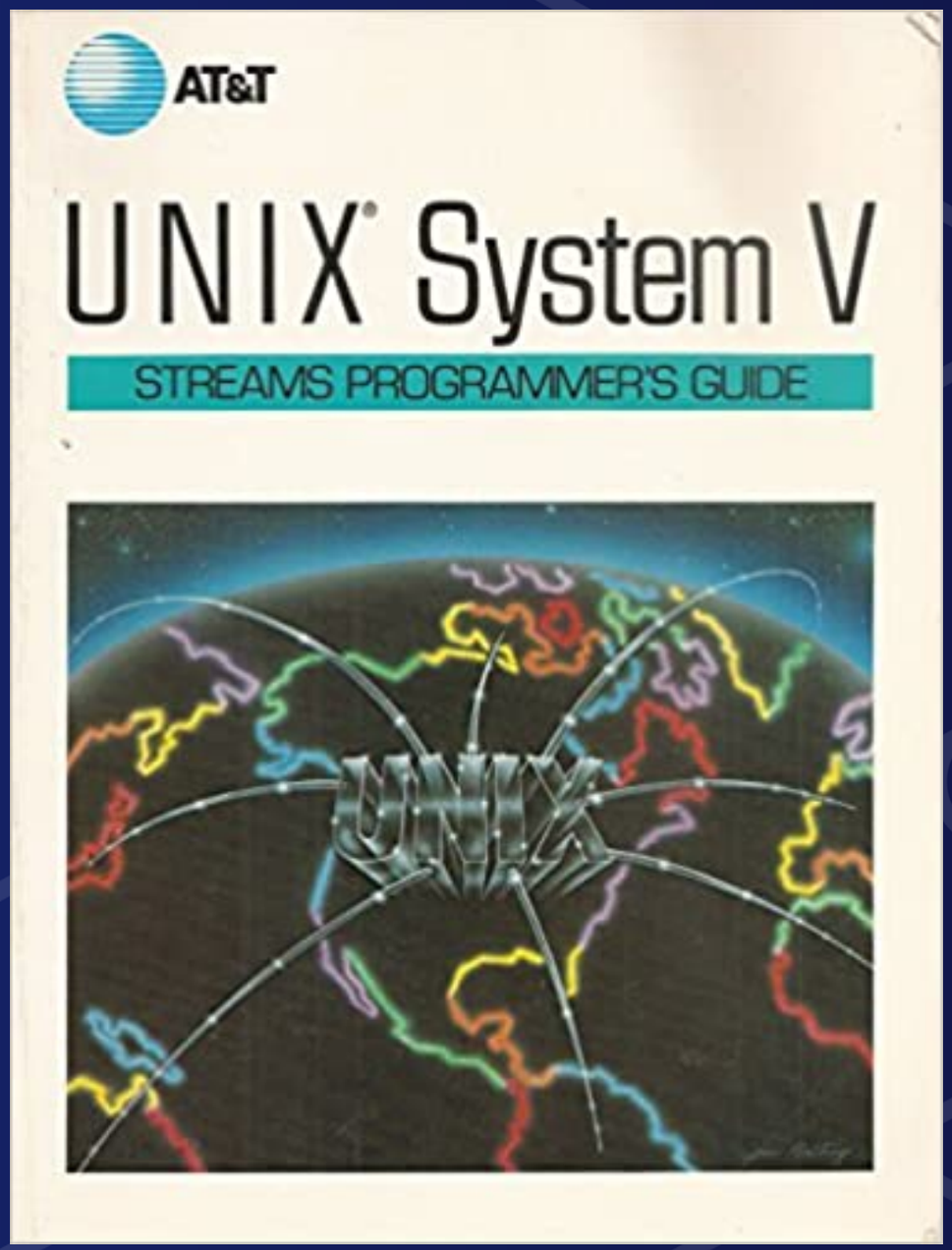
STREAMS Resources



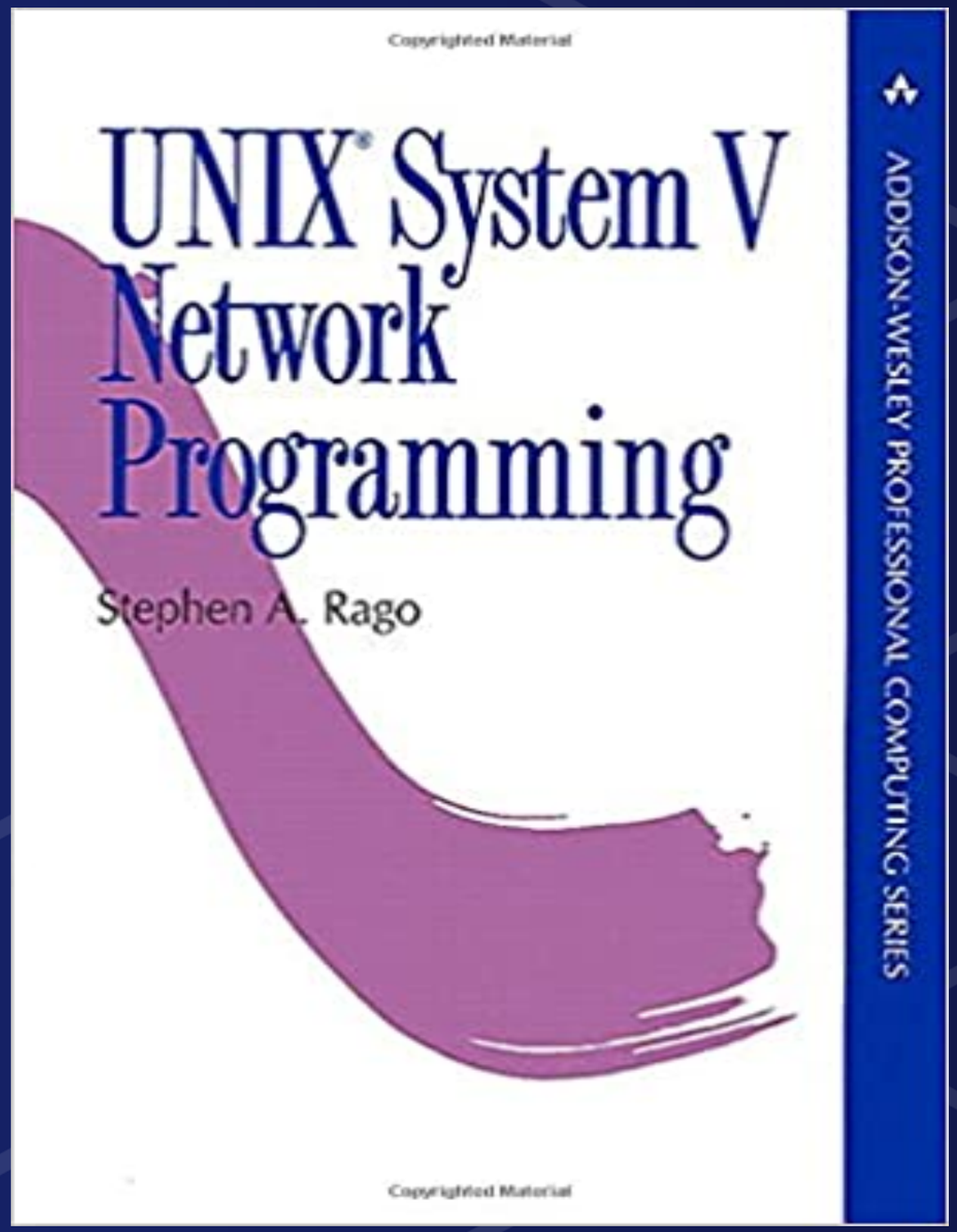
STREAMS Resources



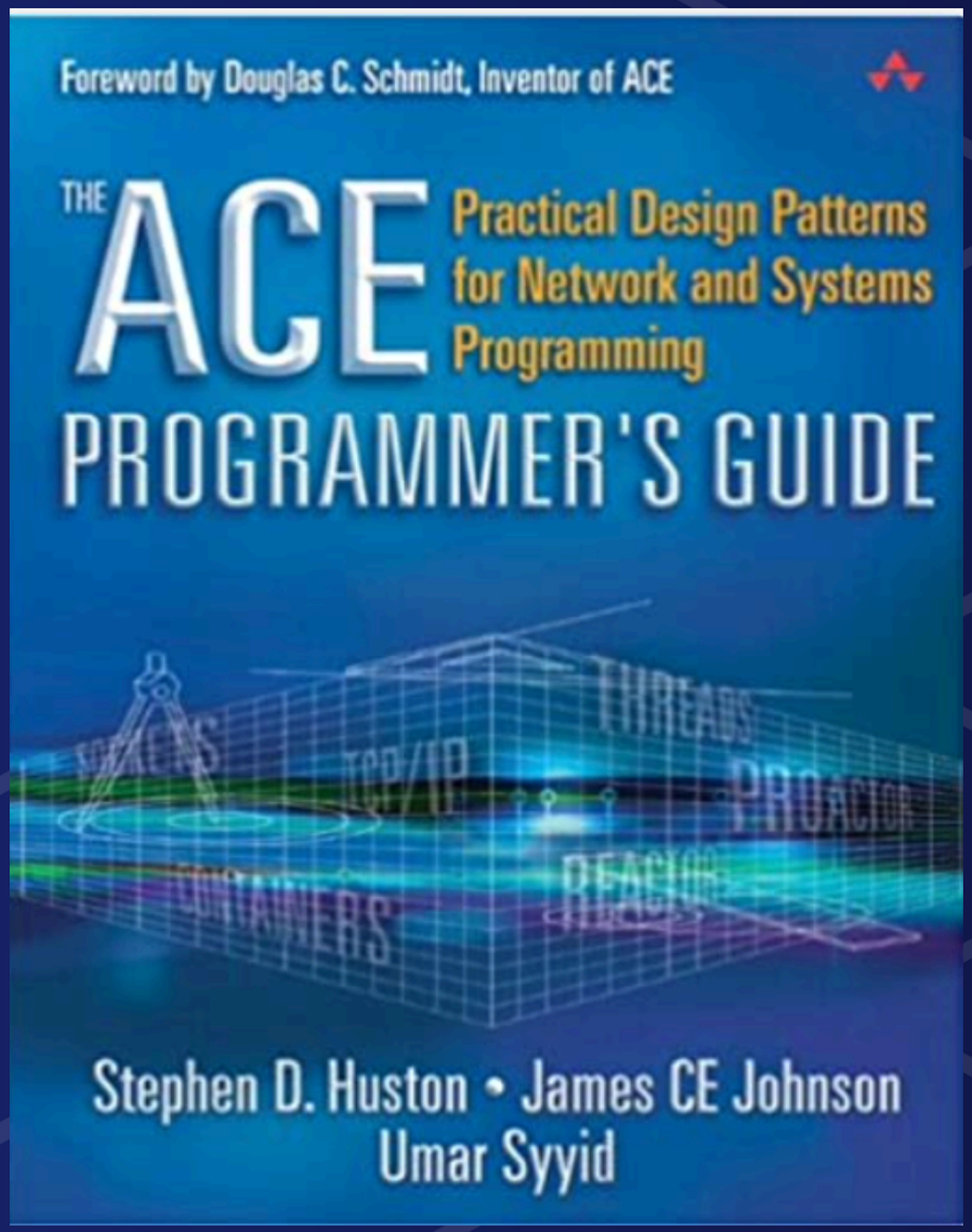
STREAMS Resources



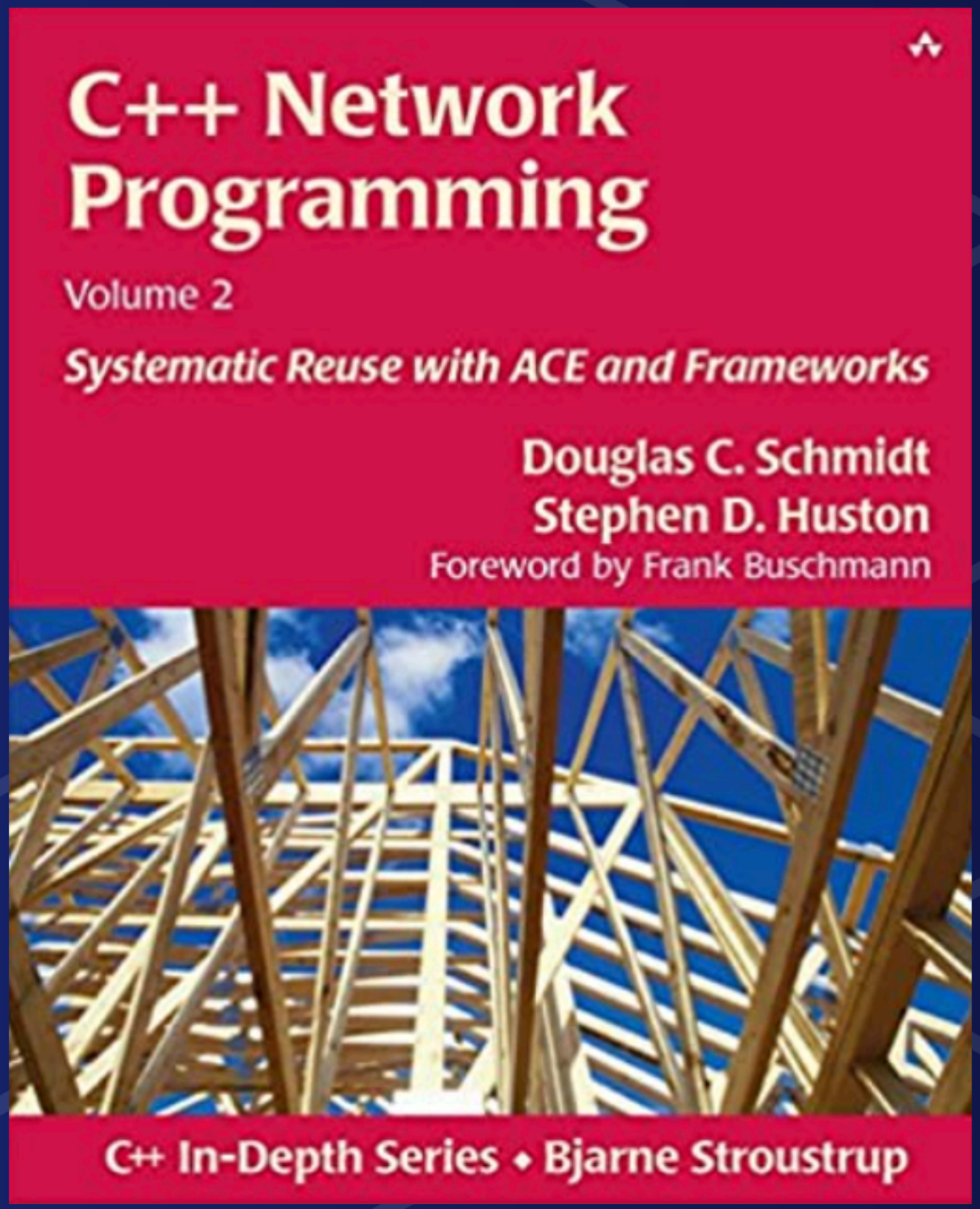
STREAMS Resources



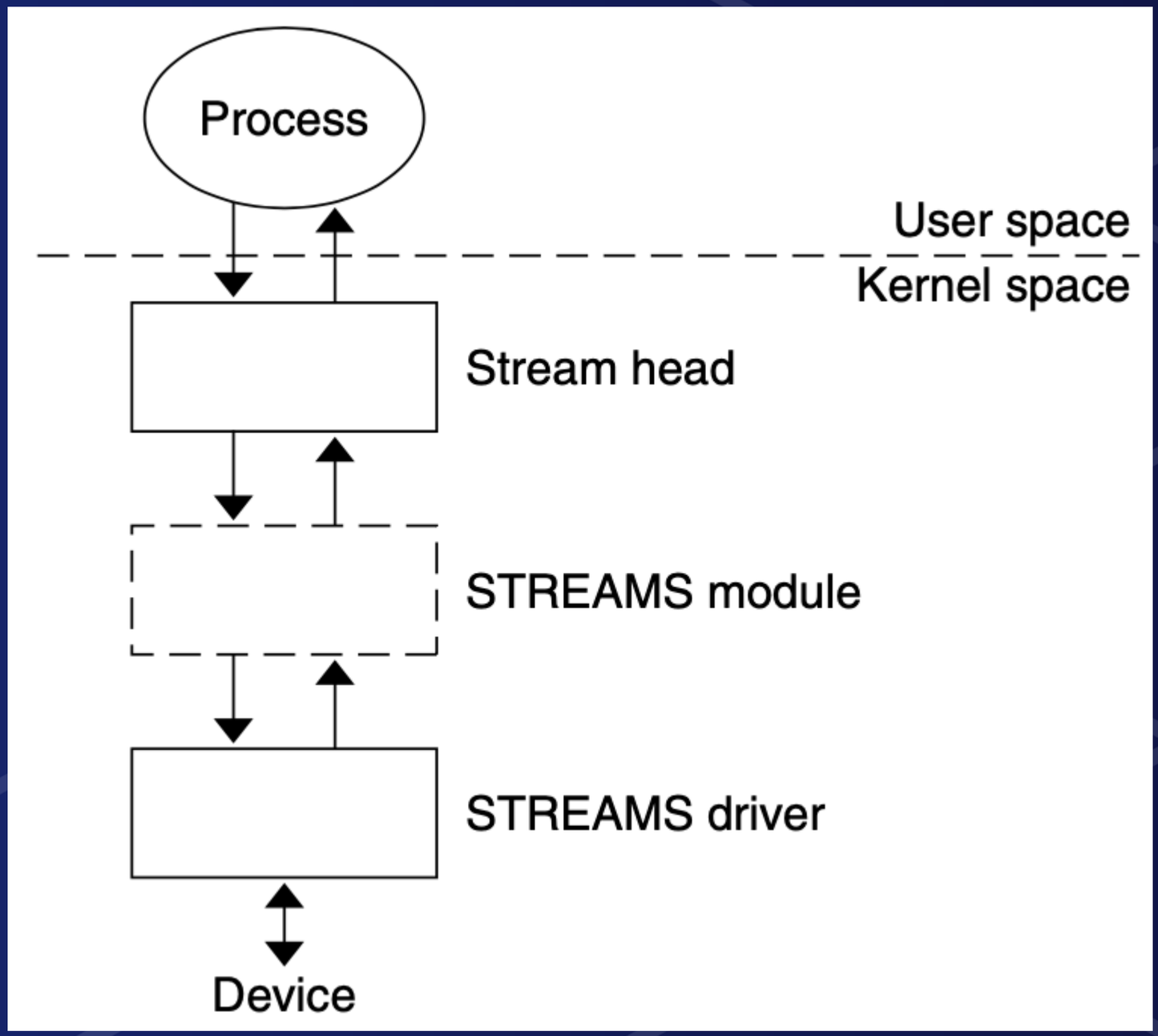
STREAMS Resources



STREAMS Resources



Love at First Sight



Love at Second Sight?

```
#include <sys/fcntl.h>
#include <stdio.h>

main()
{
        char buf[1024];
        int fd, count;

        if ((fd = open("/dev/ttya", O_RDWR)) < 0) {
            perror("open failed");
            exit(1);
        }
        while ((count = read(fd, buf, sizeof(buf))) > 0) {
            if (write(fd, buf, count) != count) {
                perror("write failed");
                break;
            }
        }
        exit(0);
}
```

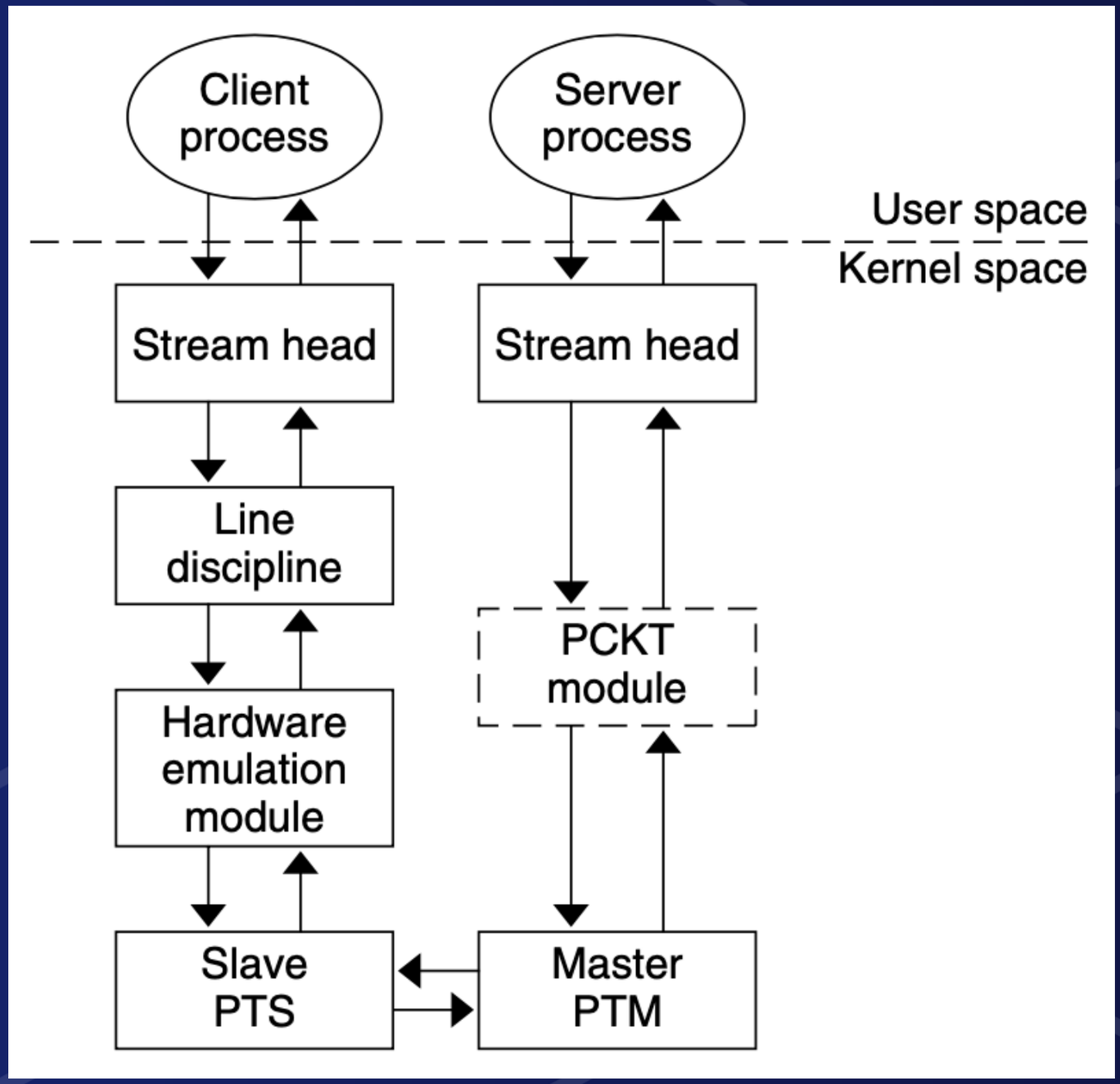
Love at Second Sight?

https://www.youtube.com/watch?v=89eEa3RvCF4

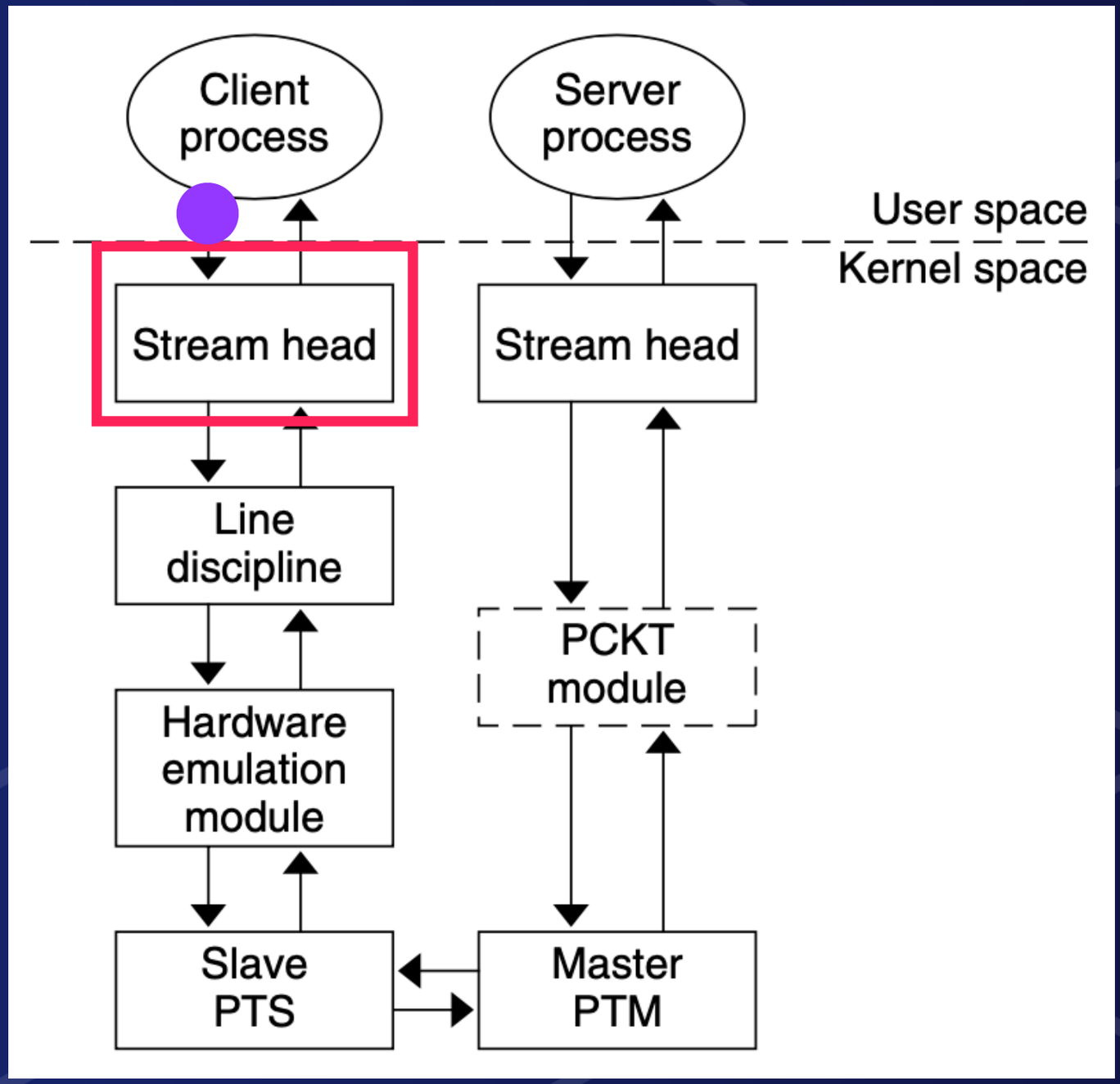
Love at Second Sight?

```
#include <sys/fcntl.h>
#include <stdio.h>

main()
{
        char buf[1024];
        int fd, count;

        if ((fd = open("/dev/ttya", O_RDWR)) < 0) {
            perror("open failed");
            exit(1);
        }
        while ((count = read(fd, buf, sizeof(buf))) > 0) {
            if (write(fd, buf, count) != count) {
                perror("write failed");
                break;
            }
        }
        exit(0);
}
```

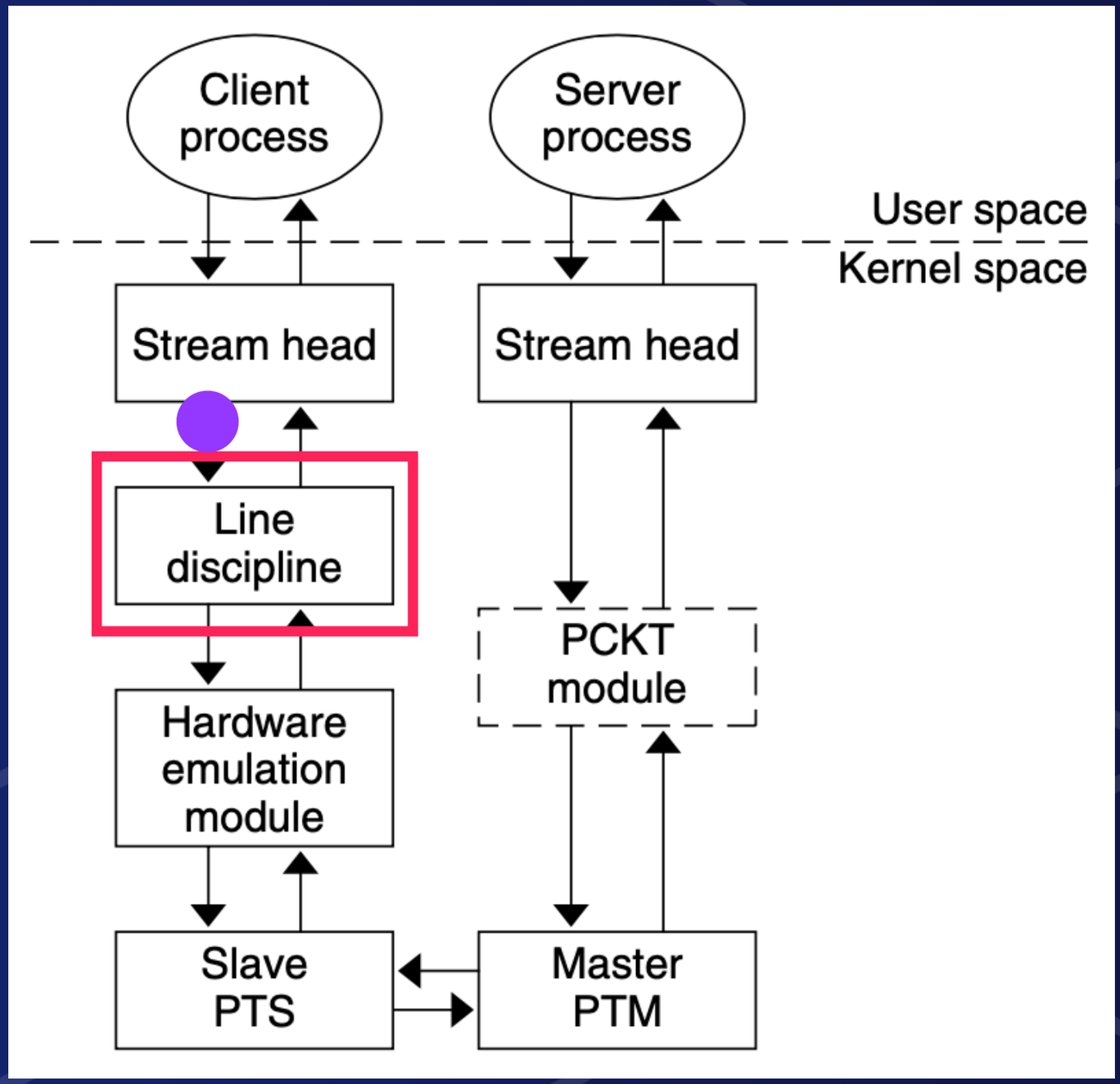
Pseudo-TTY in STREAMS



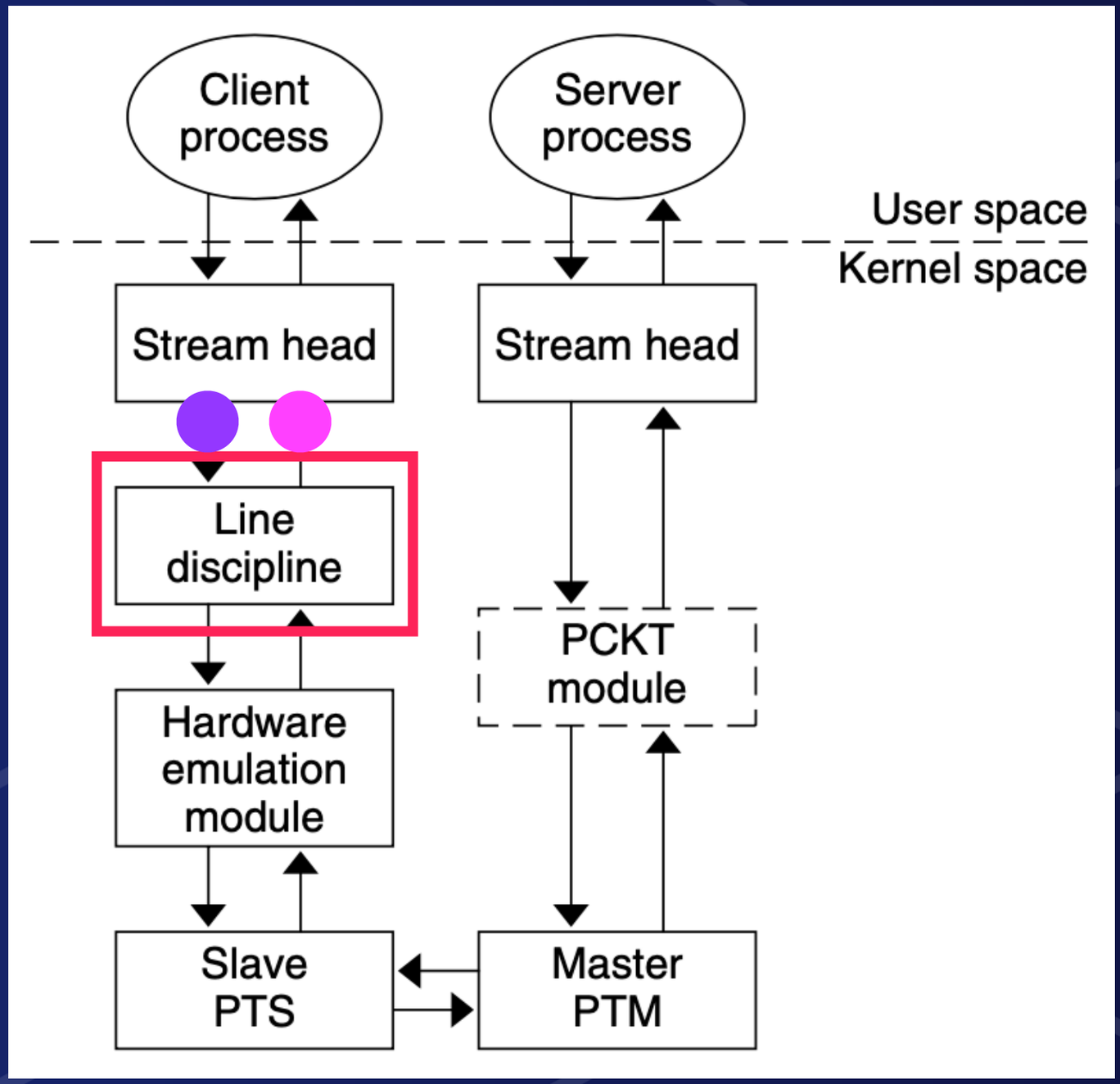
Pseudo-TTY in STREAMS



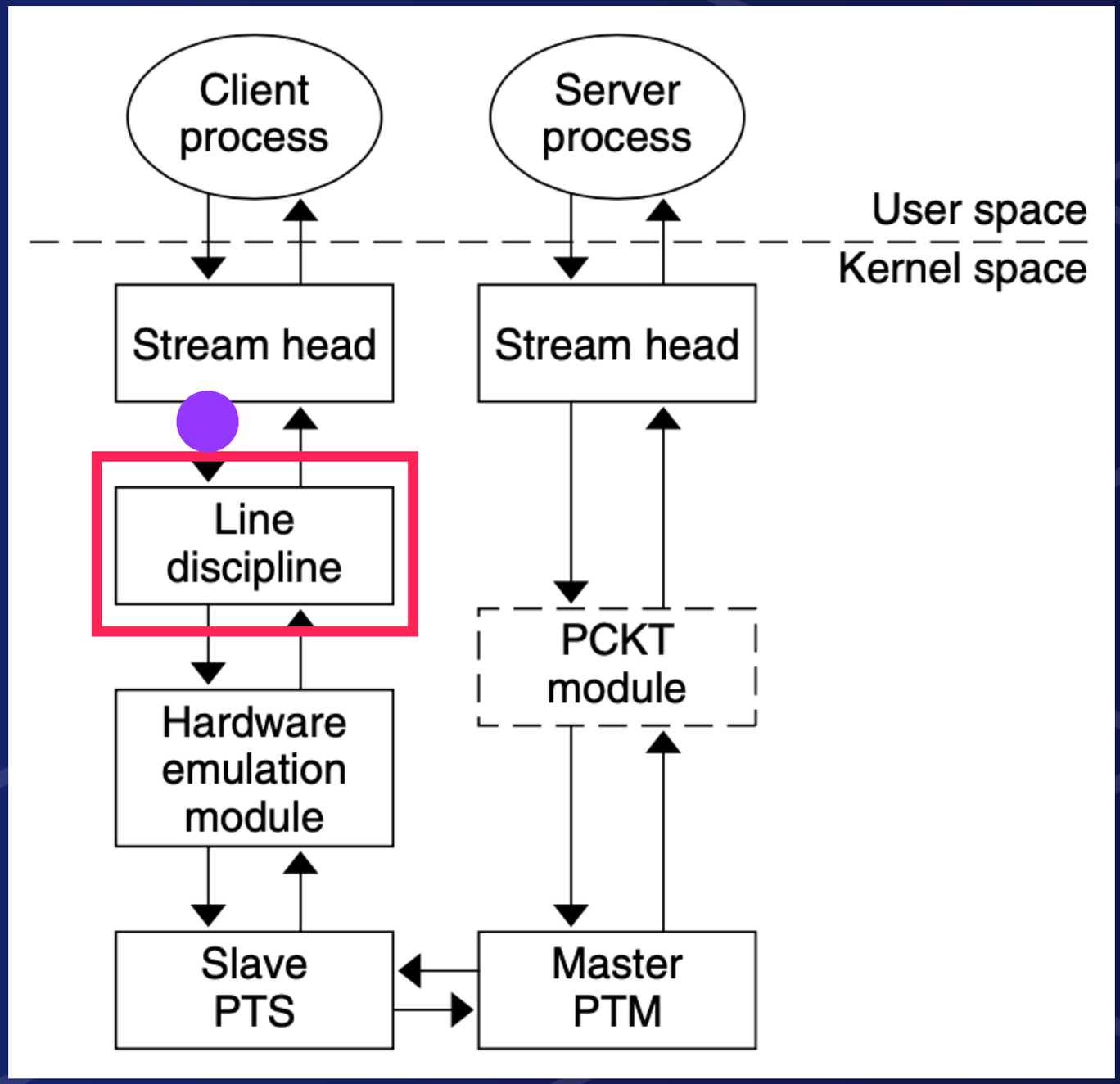
Pseudo-TTY in STREAMS



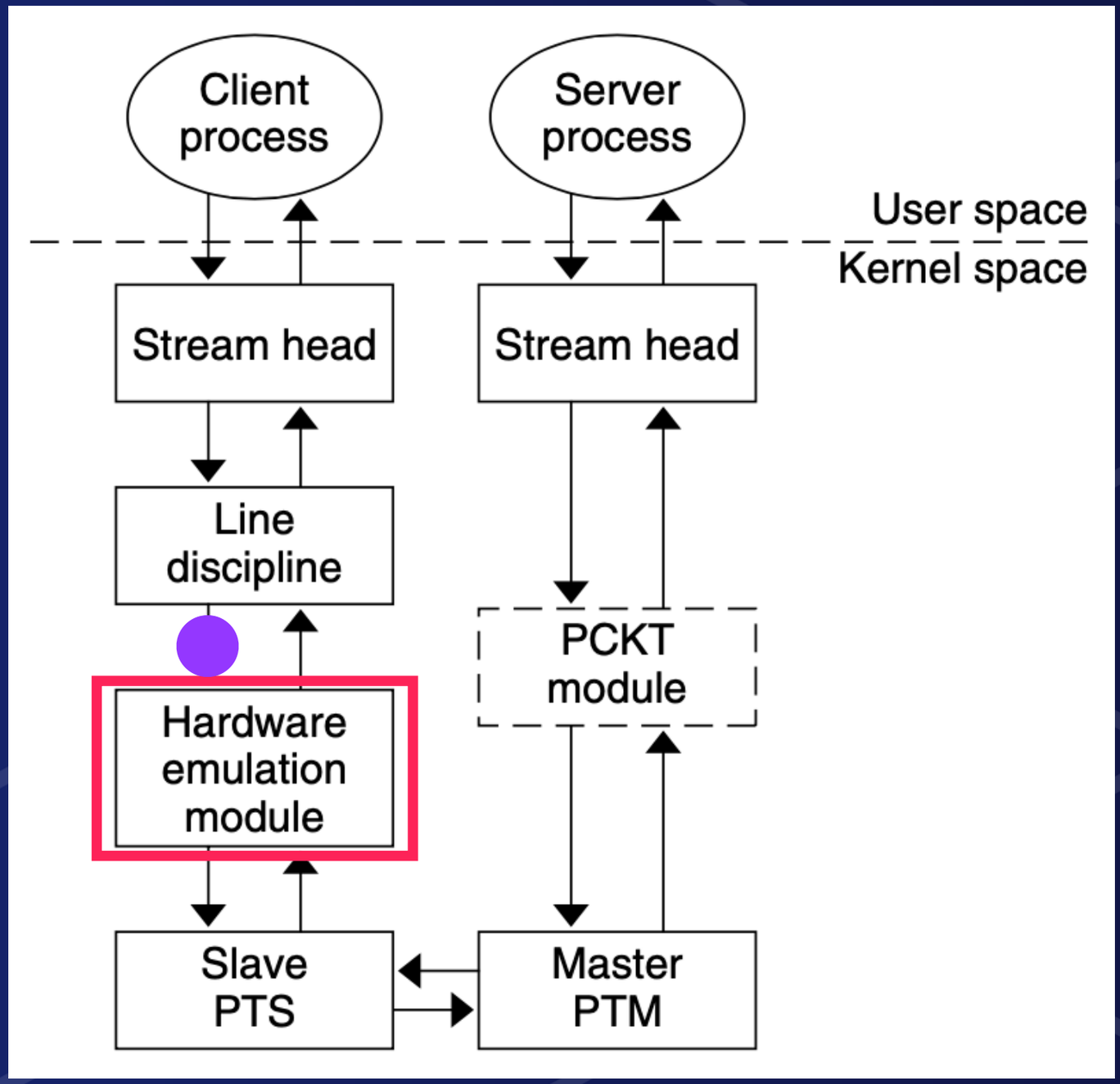
Pseudo-TTY in STREAMS



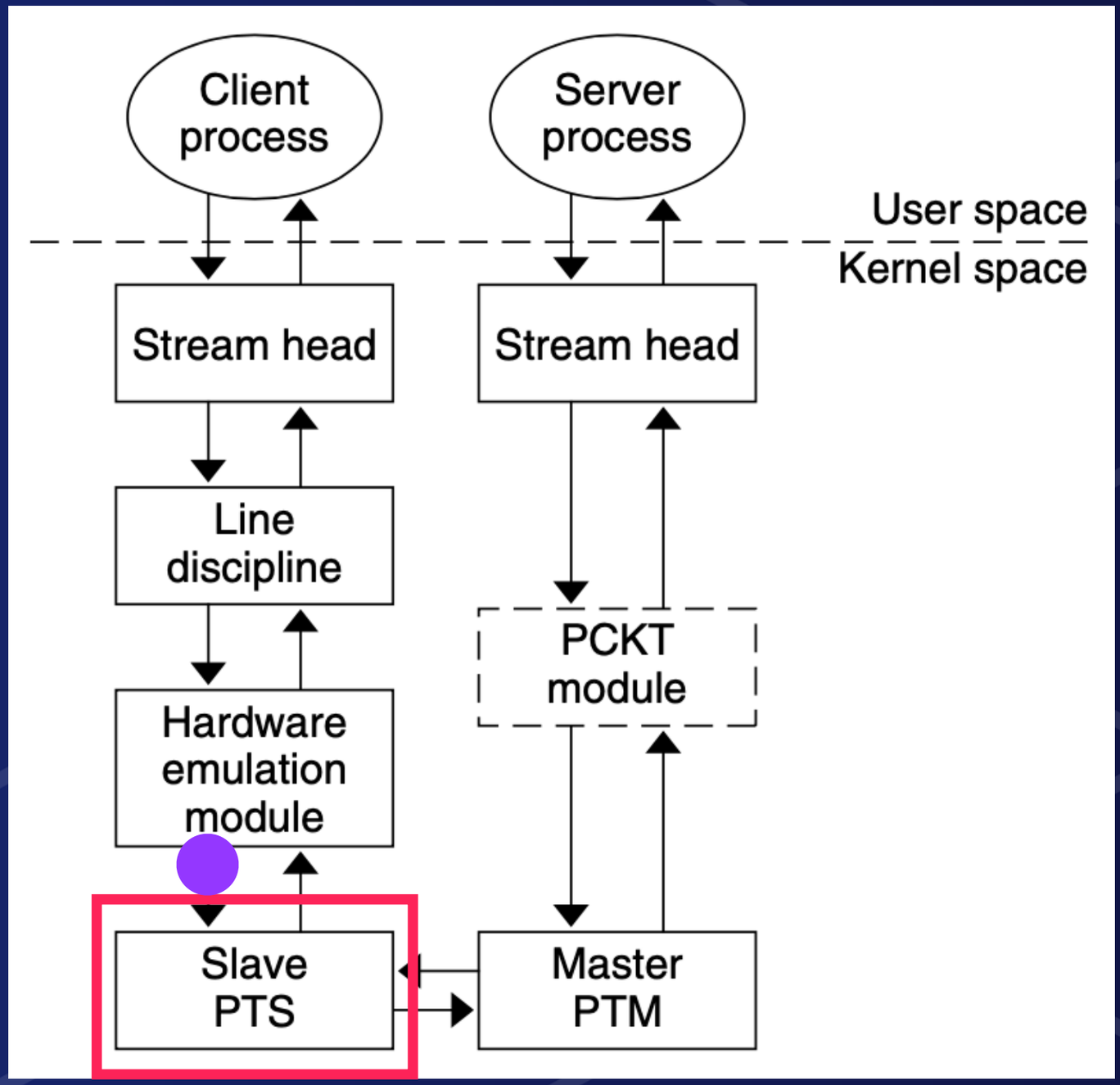
Pseudo-TTY in STREAMS



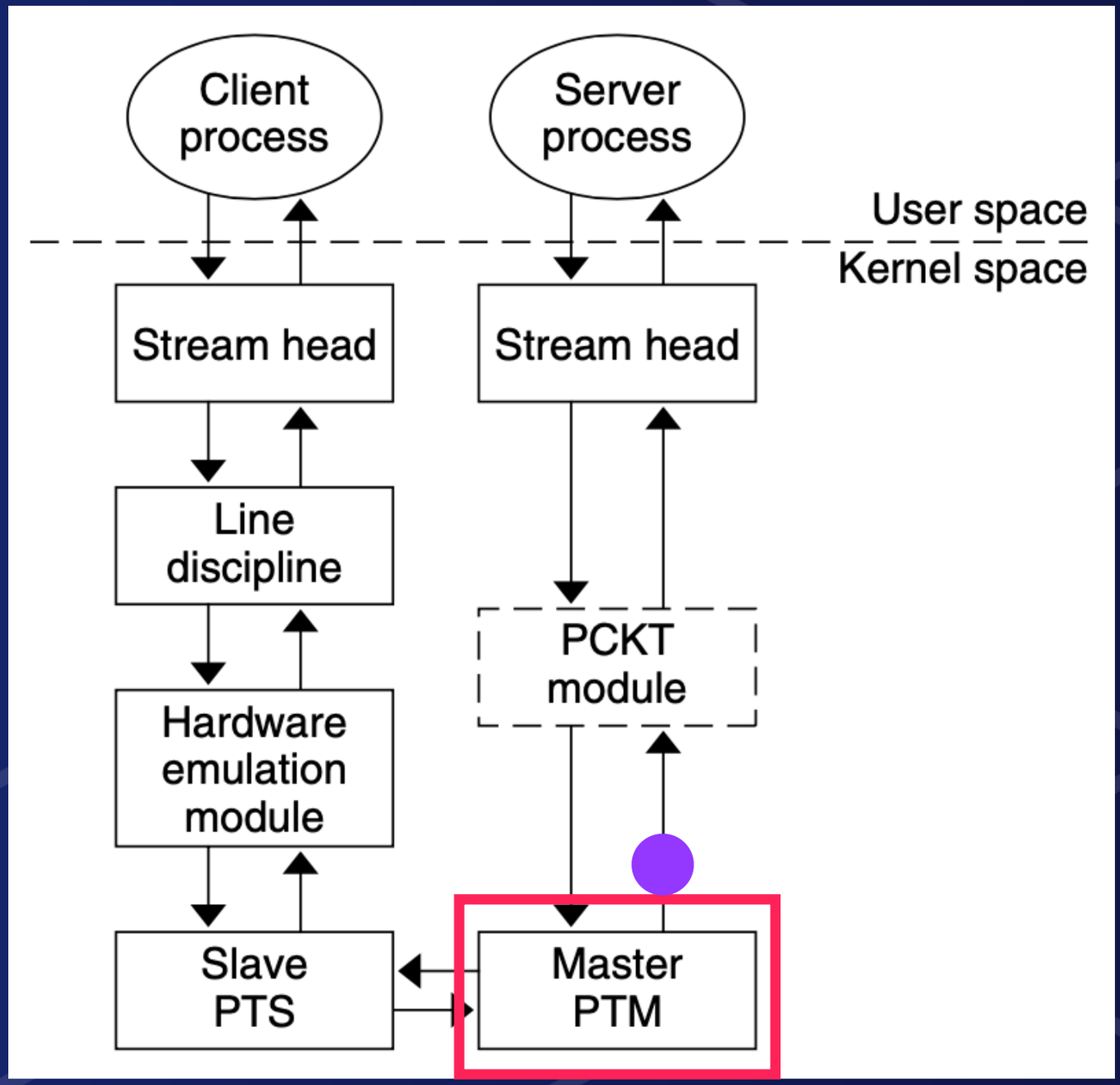
Pseudo-TTY in STREAMS



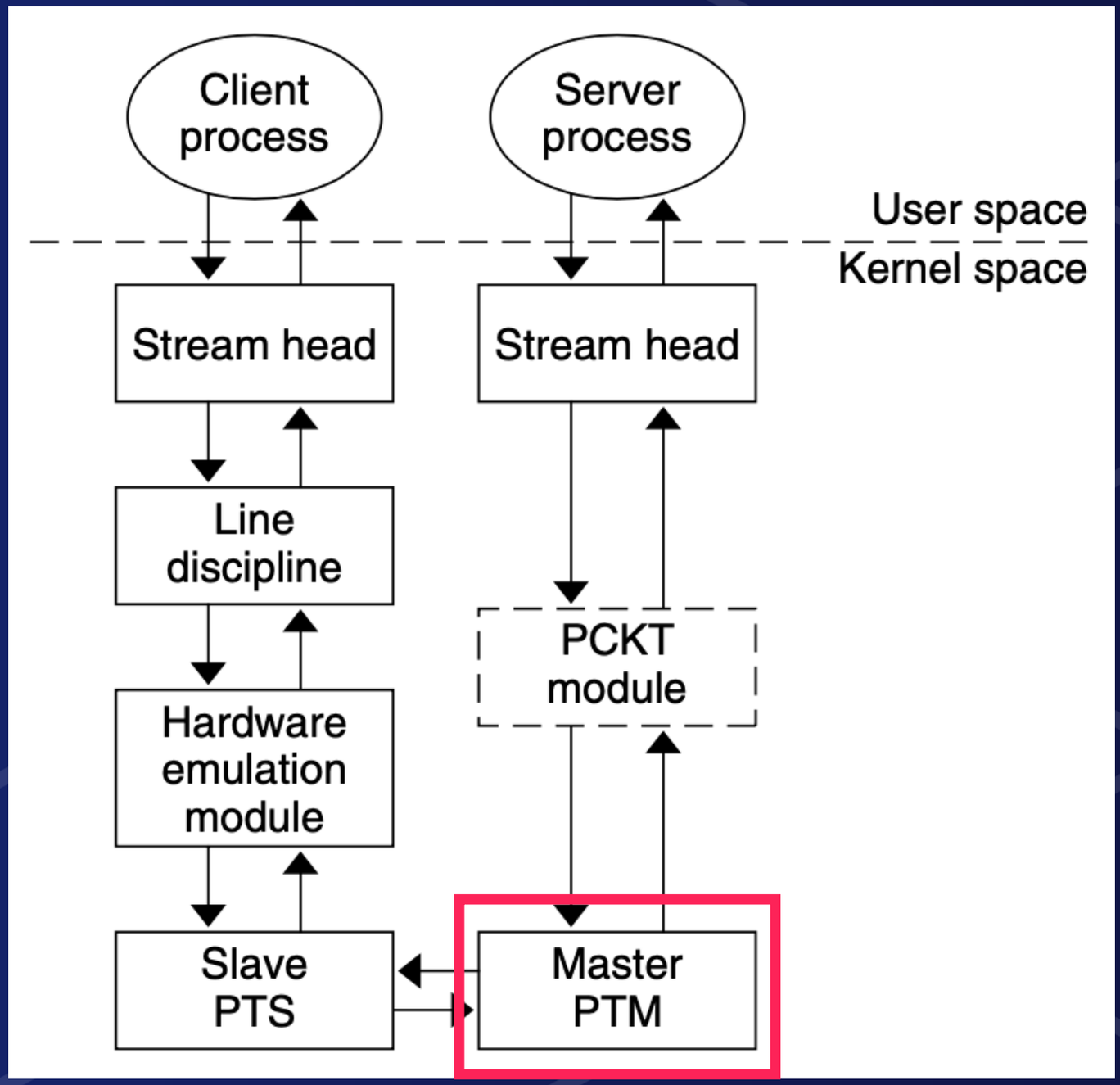
Pseudo-TTY in STREAMS



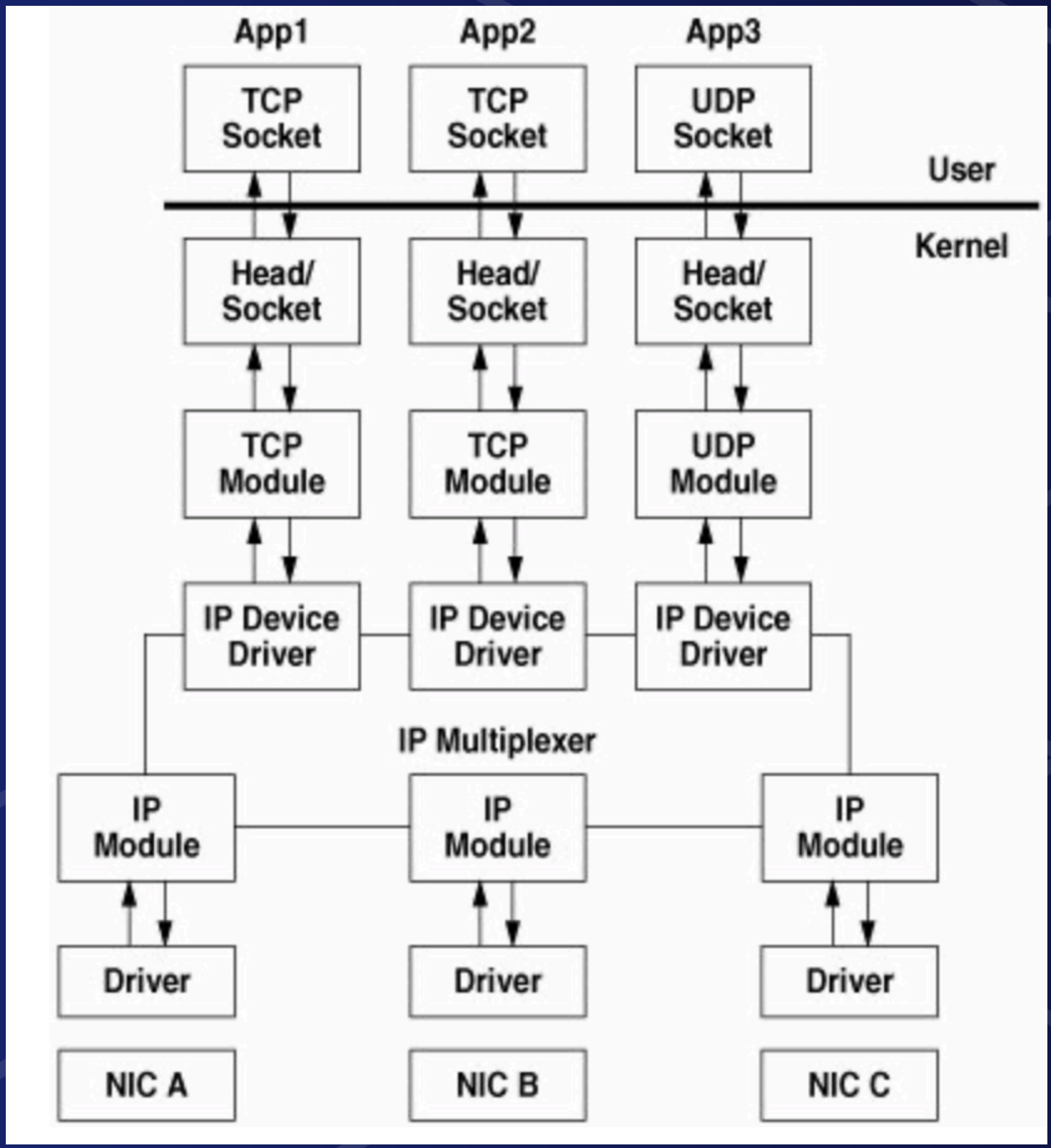
Pseudo-TTY in STREAMS



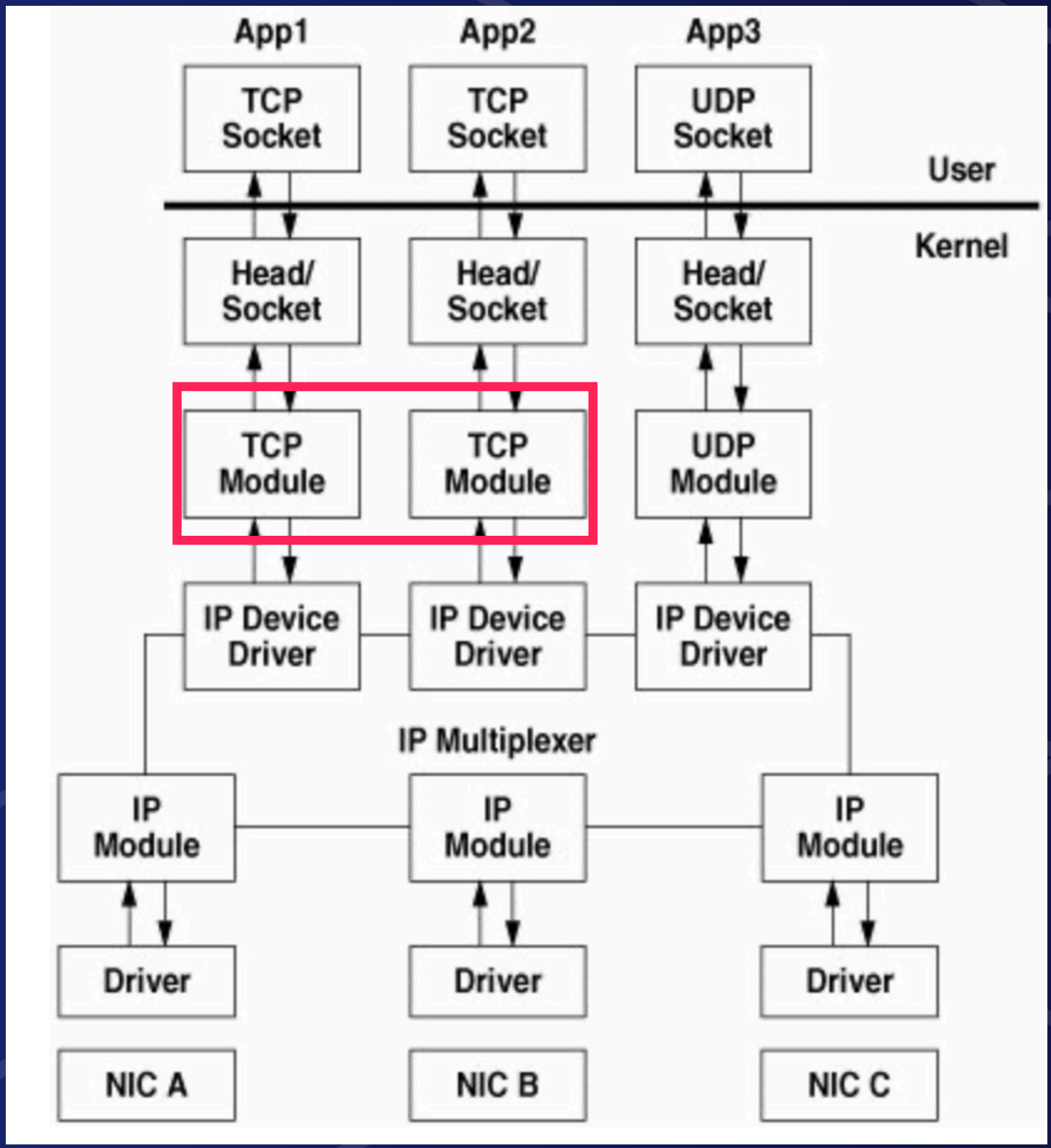
Pseudo-TTY in STREAMS



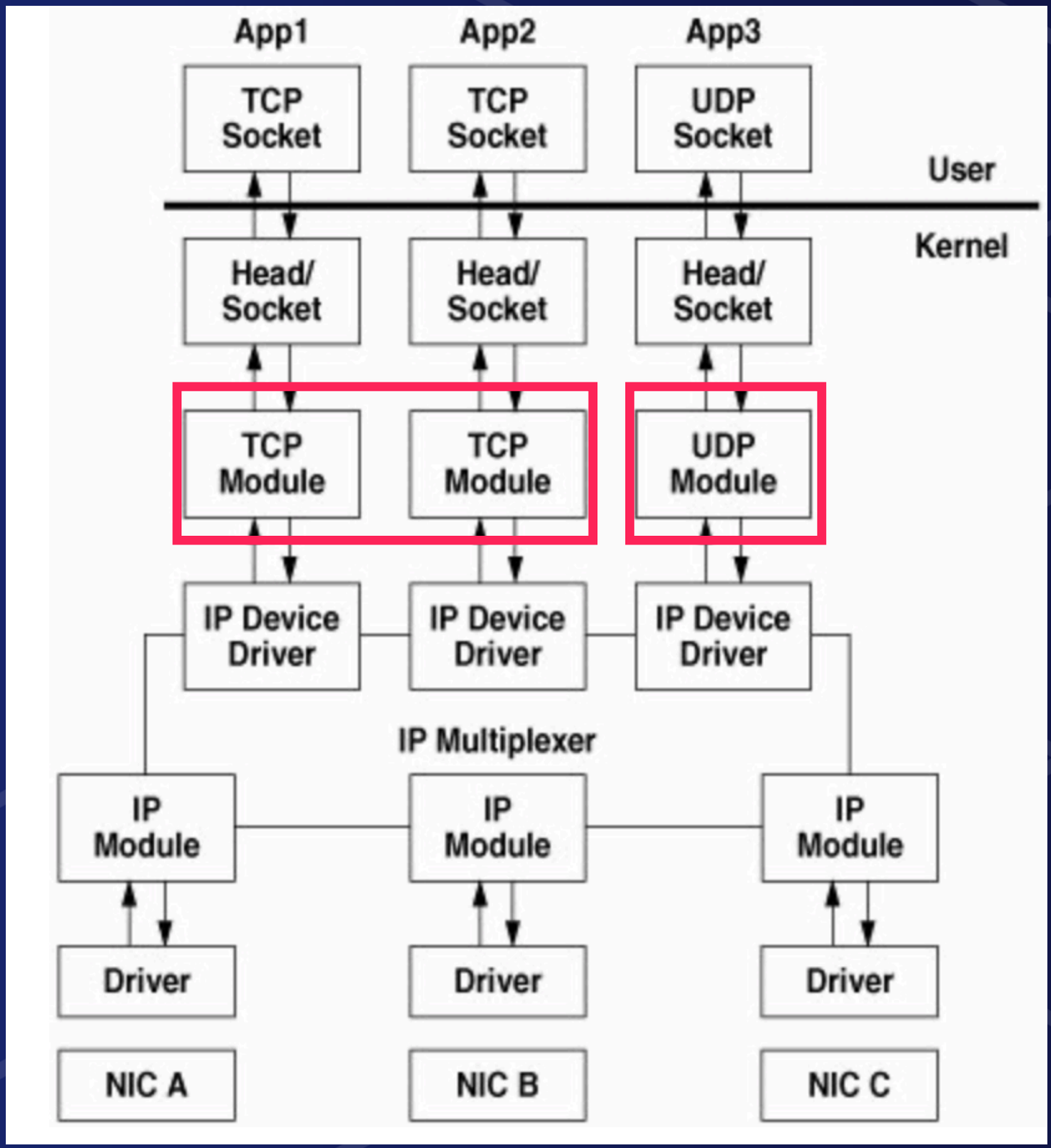
TCP/UDP in STREAMS



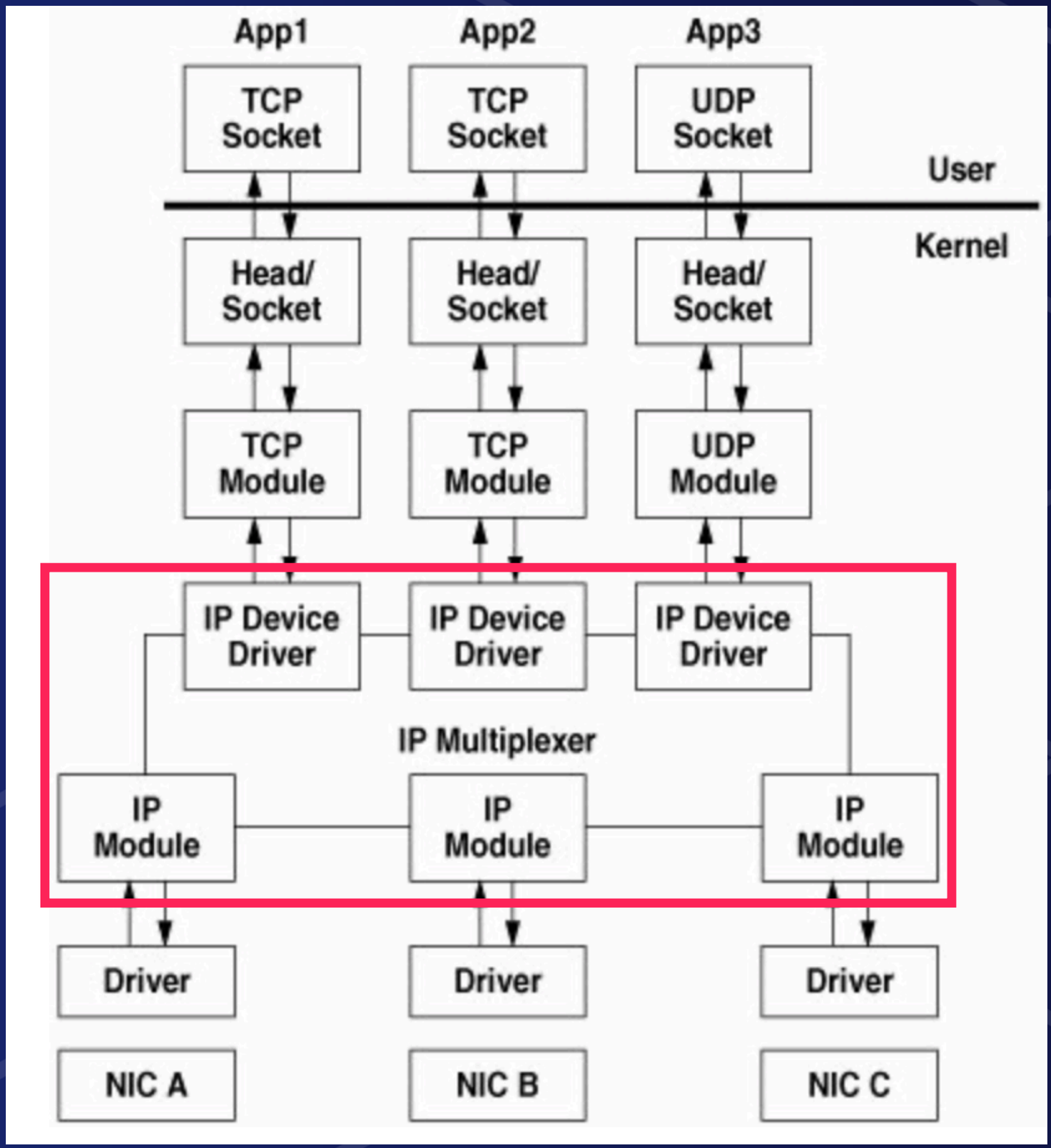
TCP/UDP in STREAMS



TCP/UDP in STREAMS



TCP/UDP in STREAMS



Reliability

Remember the library to intercept SYSV IPC?

Reliability

Remember the library to intercept SYSV IPC?

New requirement - reliability

Reliability

Remember the library to intercept SYSV IPC?

New requirement - reliability

Multiple network cards

Reliability

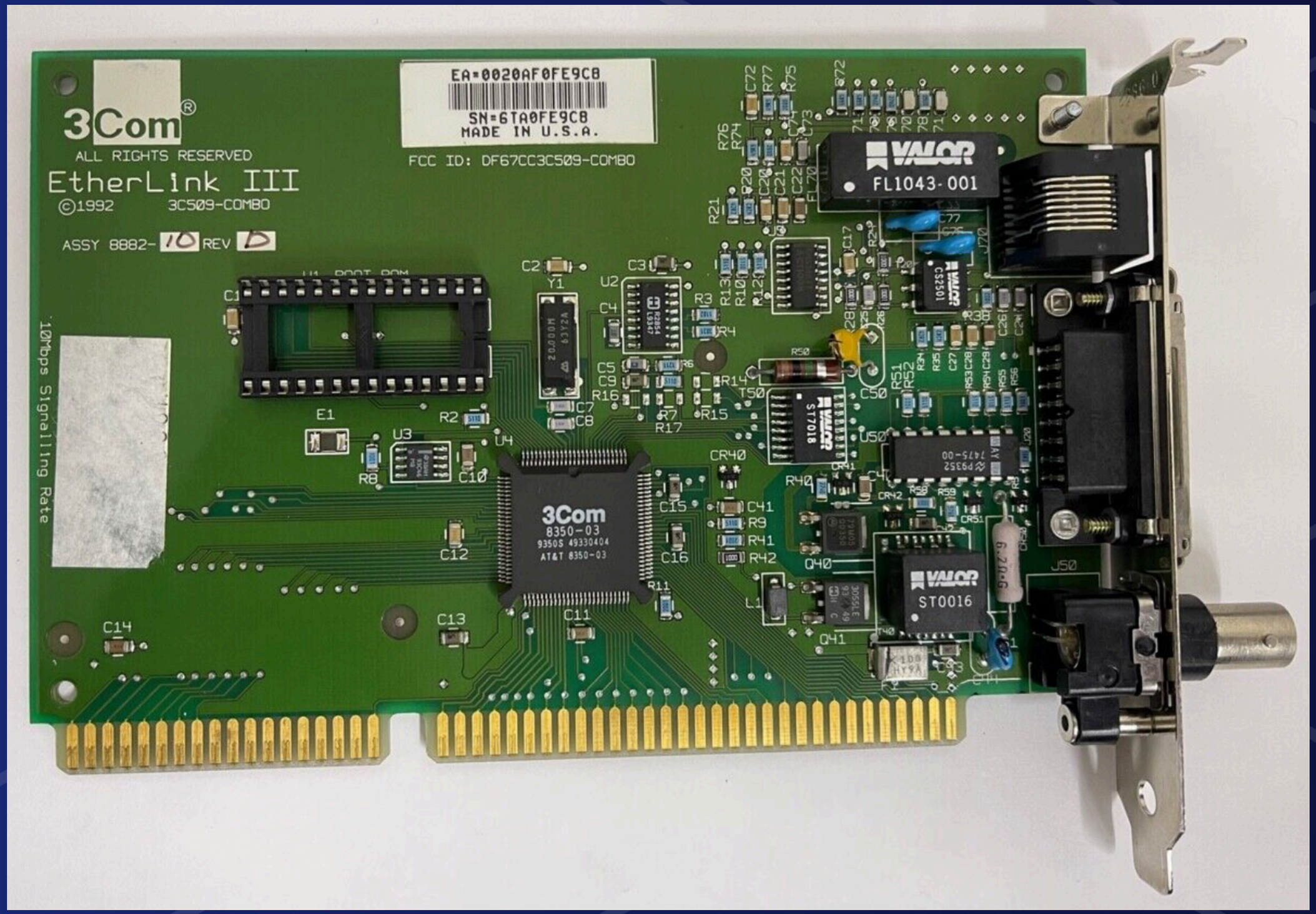
Remember the library to intercept SYSV IPC?

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

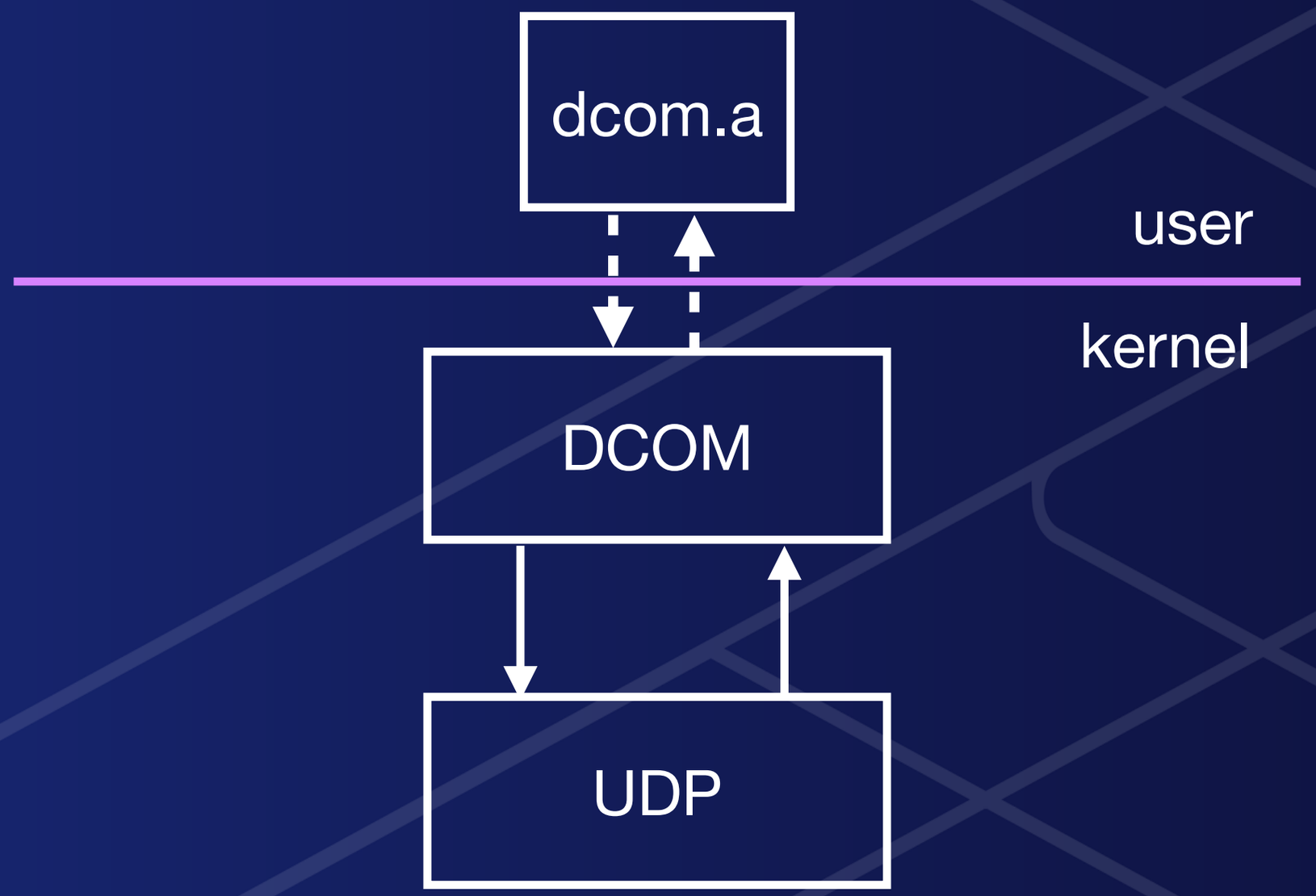
Reliability

Remember the library to intercept SYSV IPC?

dcom.a

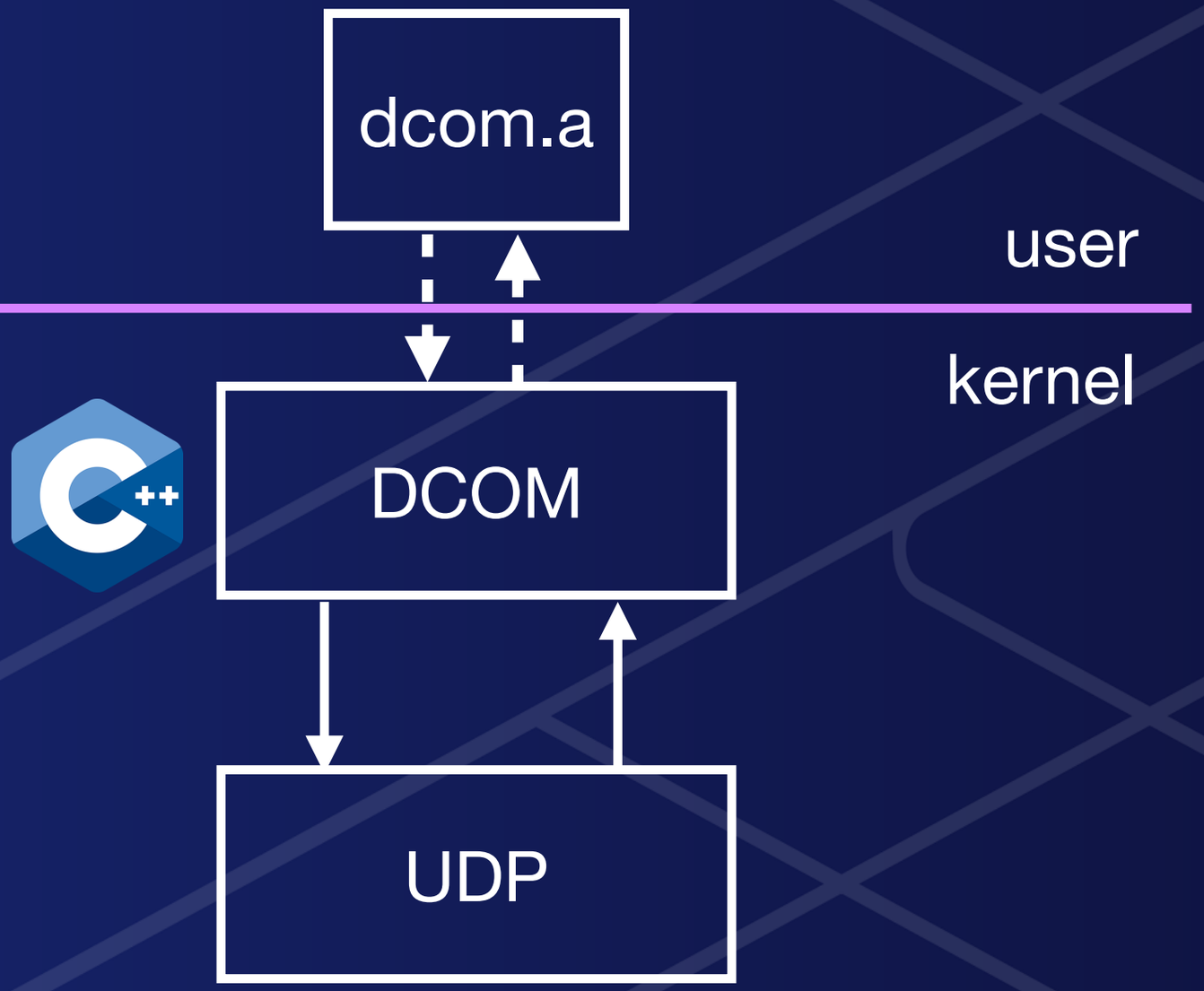
Reliability

Remember the library to intercept SYSV IPC?



Reliability

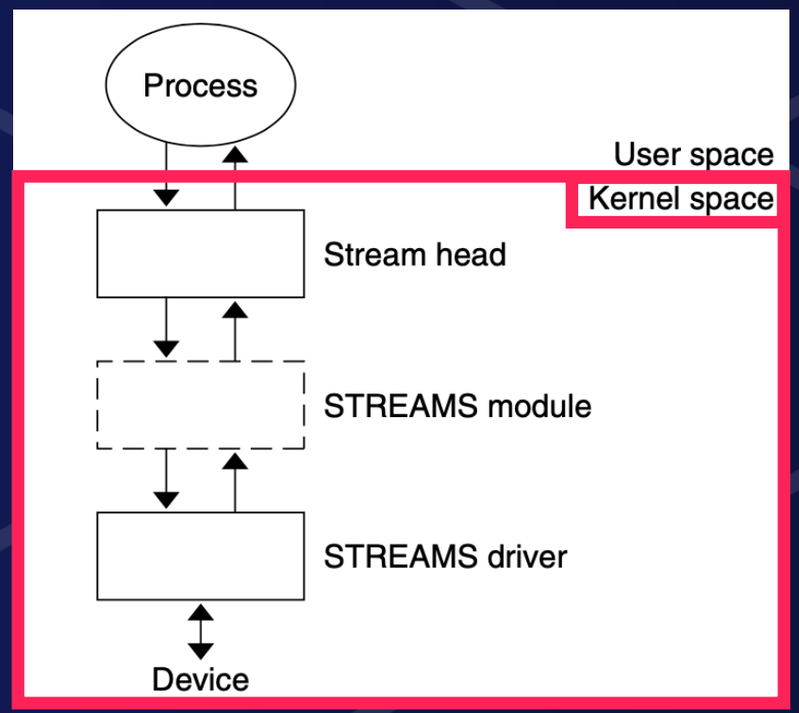
Remember the library to intercept SYSV IPC?



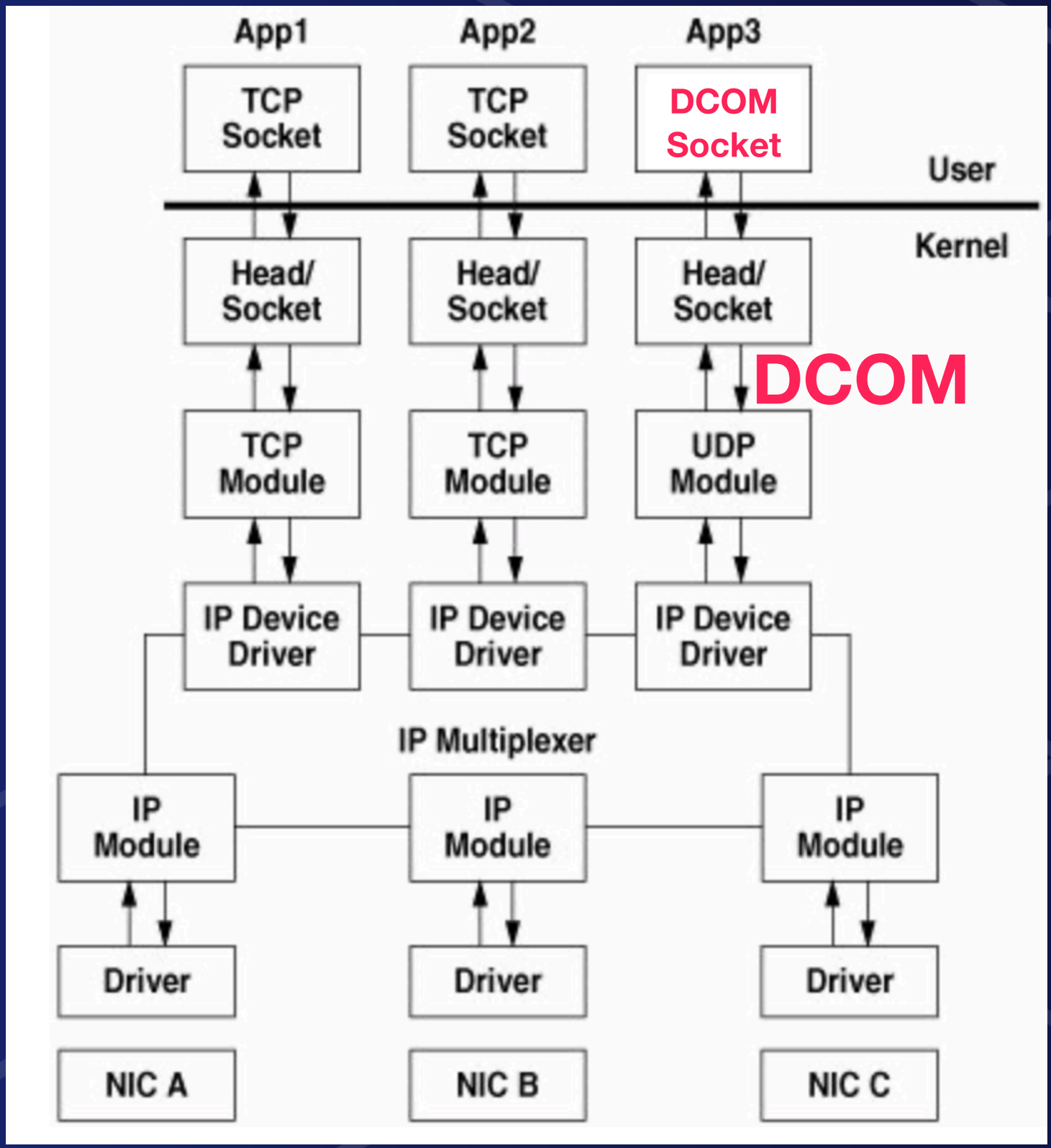
Me and C++ and STREAMS Sitting in a Tree

https://www.youtube.com/watch?v=Pd0VBm8gU5o

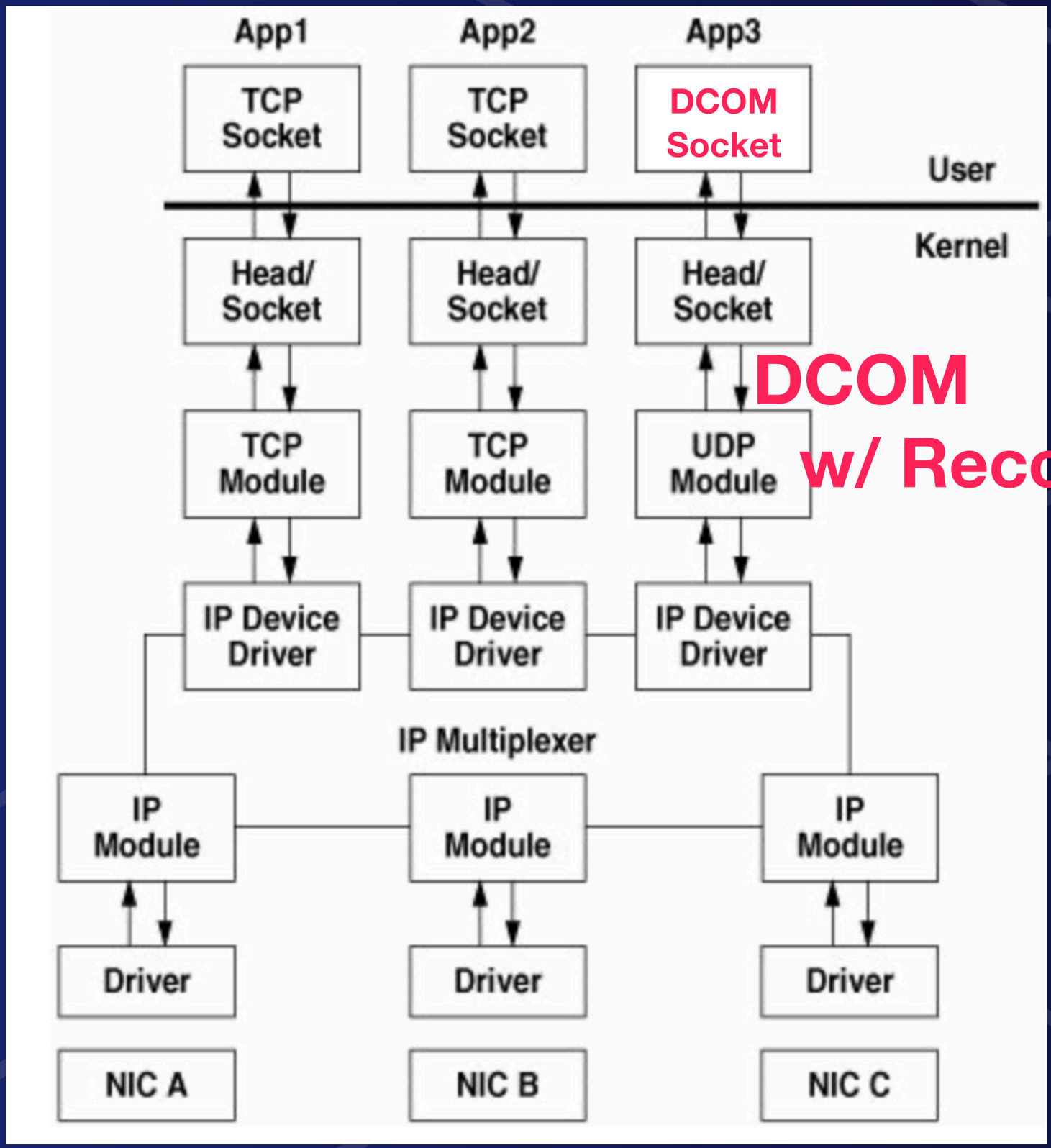
Me and C++ and STREAMS Sitting in a Tree



Reliability

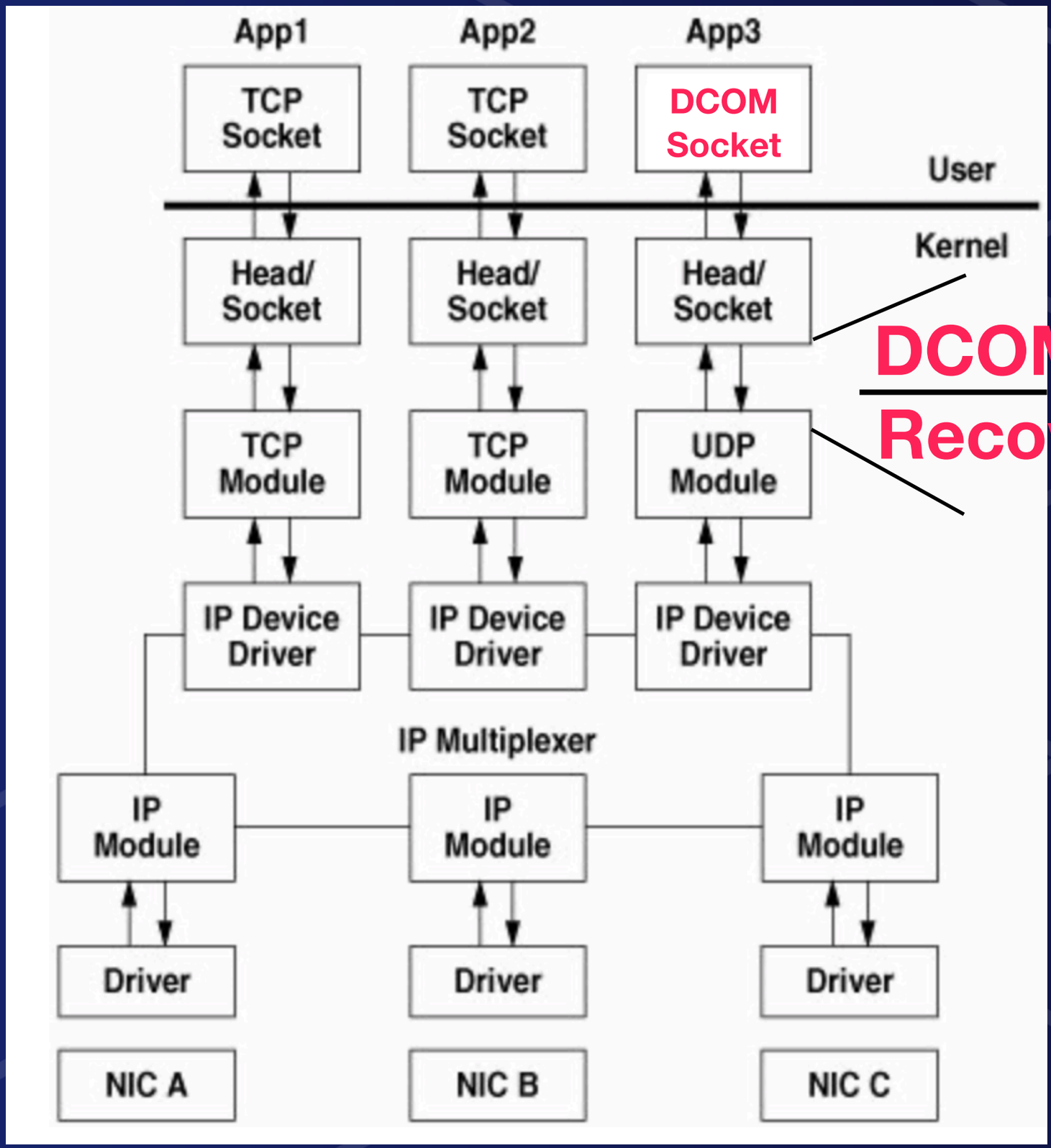


Reliability



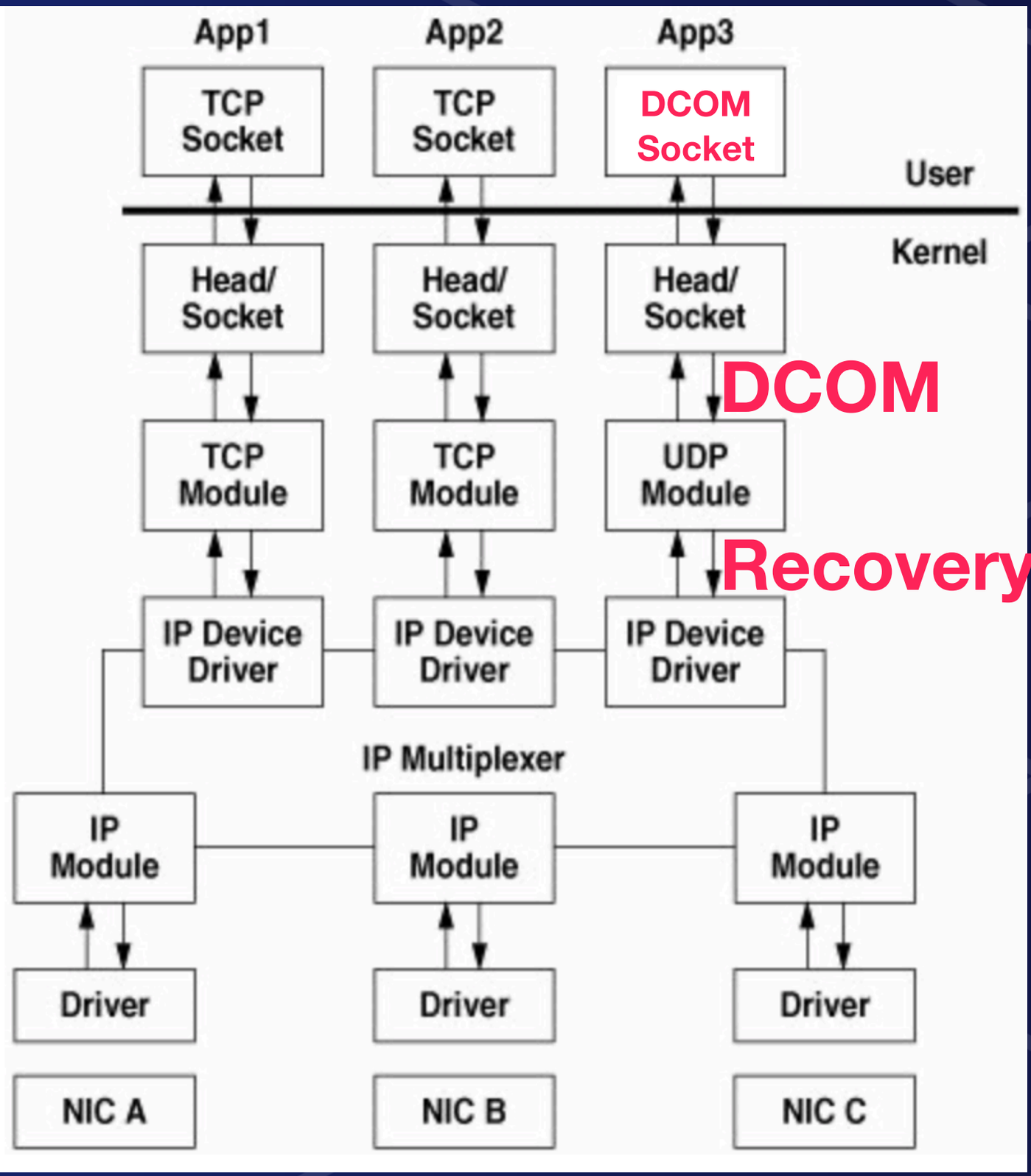
DCOM w/ Recovery

Reliability

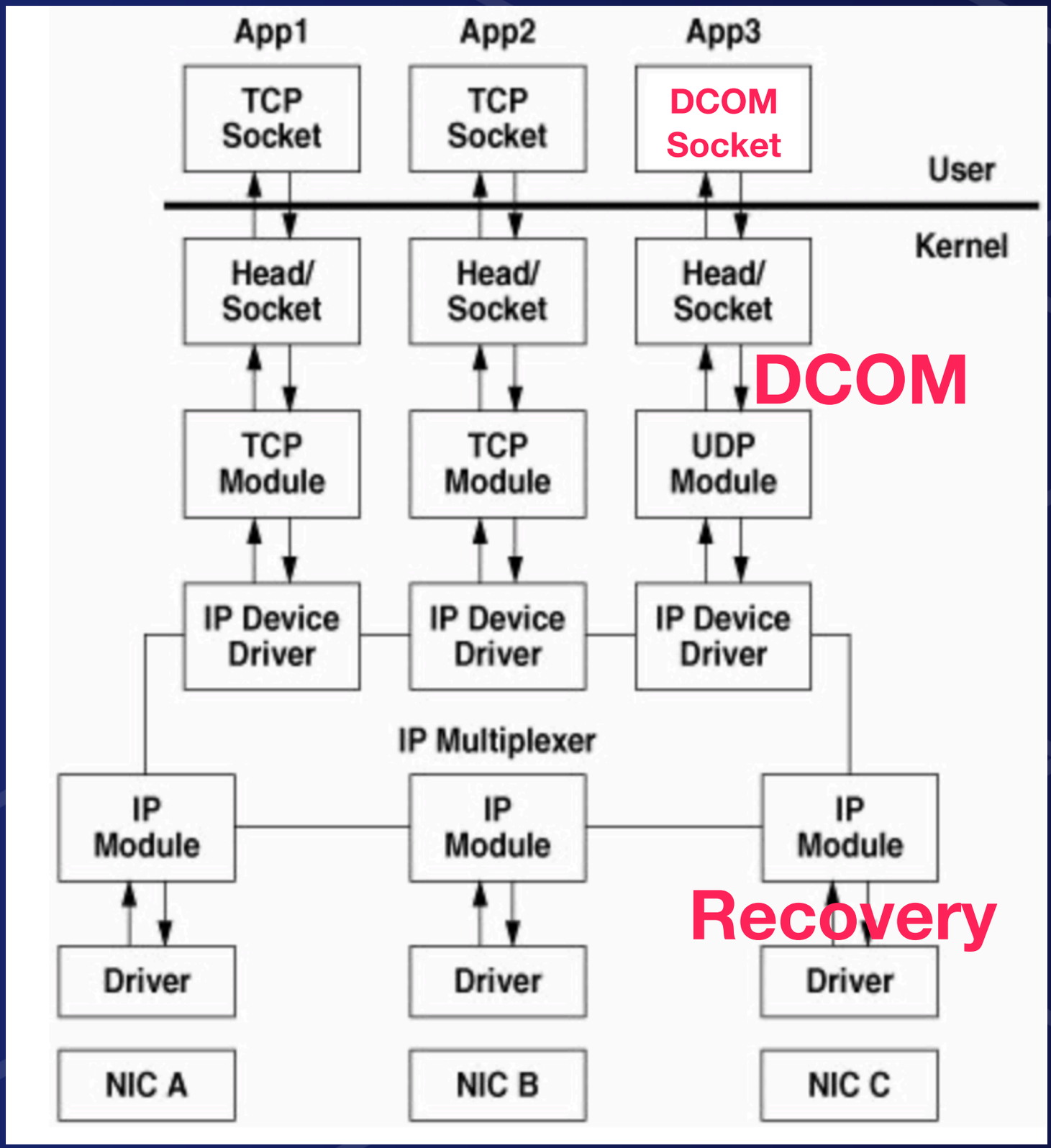


DCOM Recovery

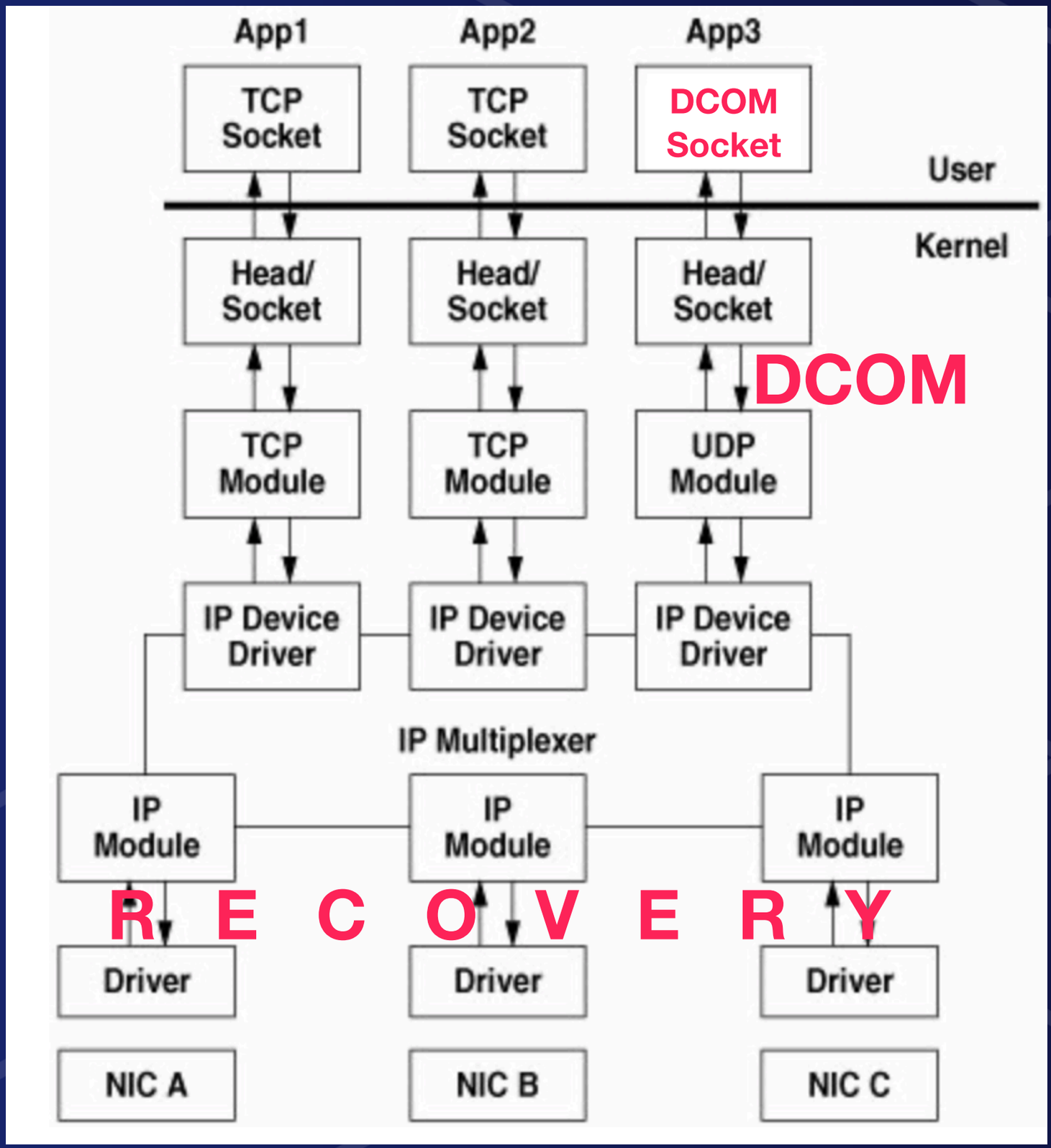
Reliability



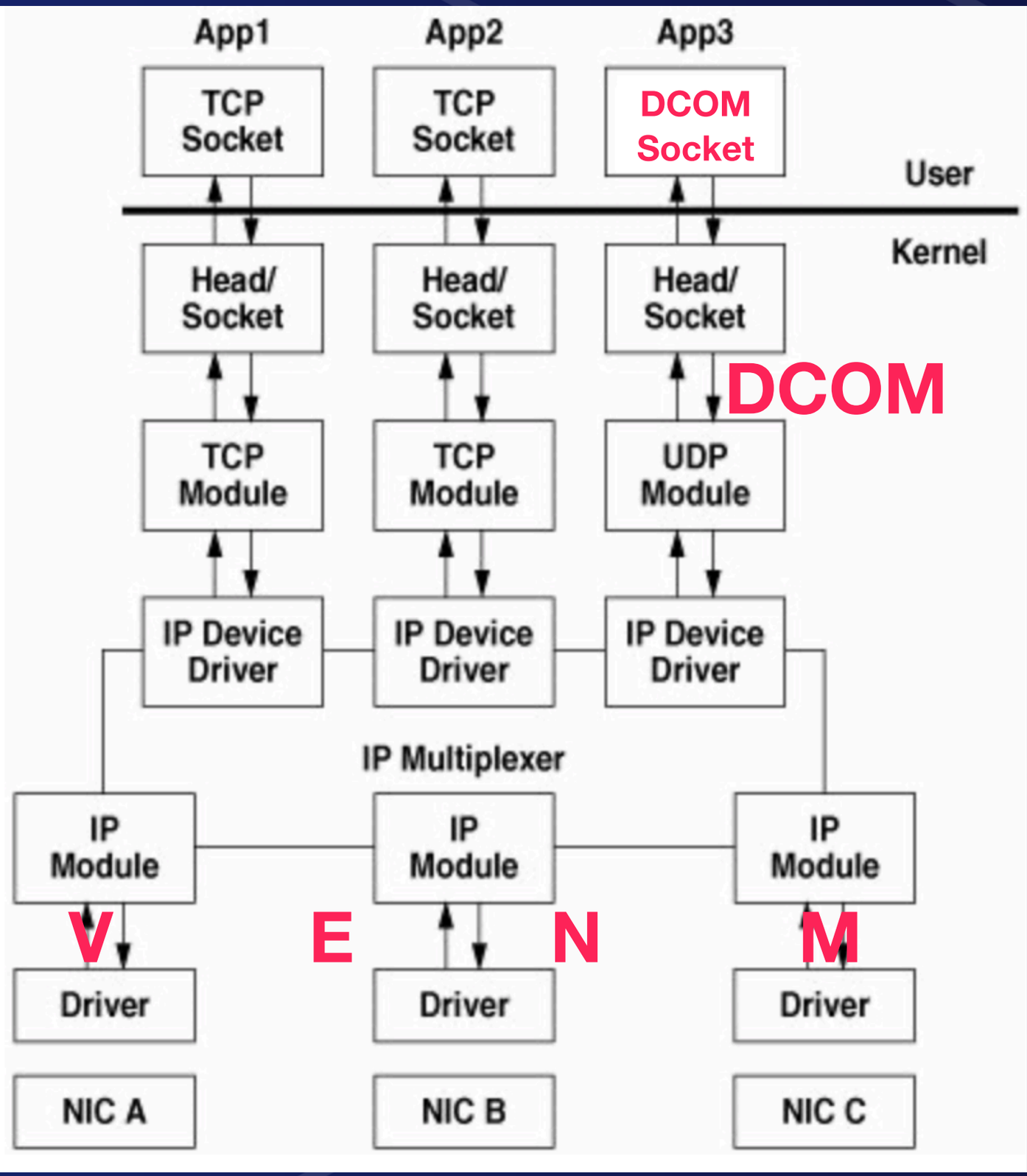
Reliability



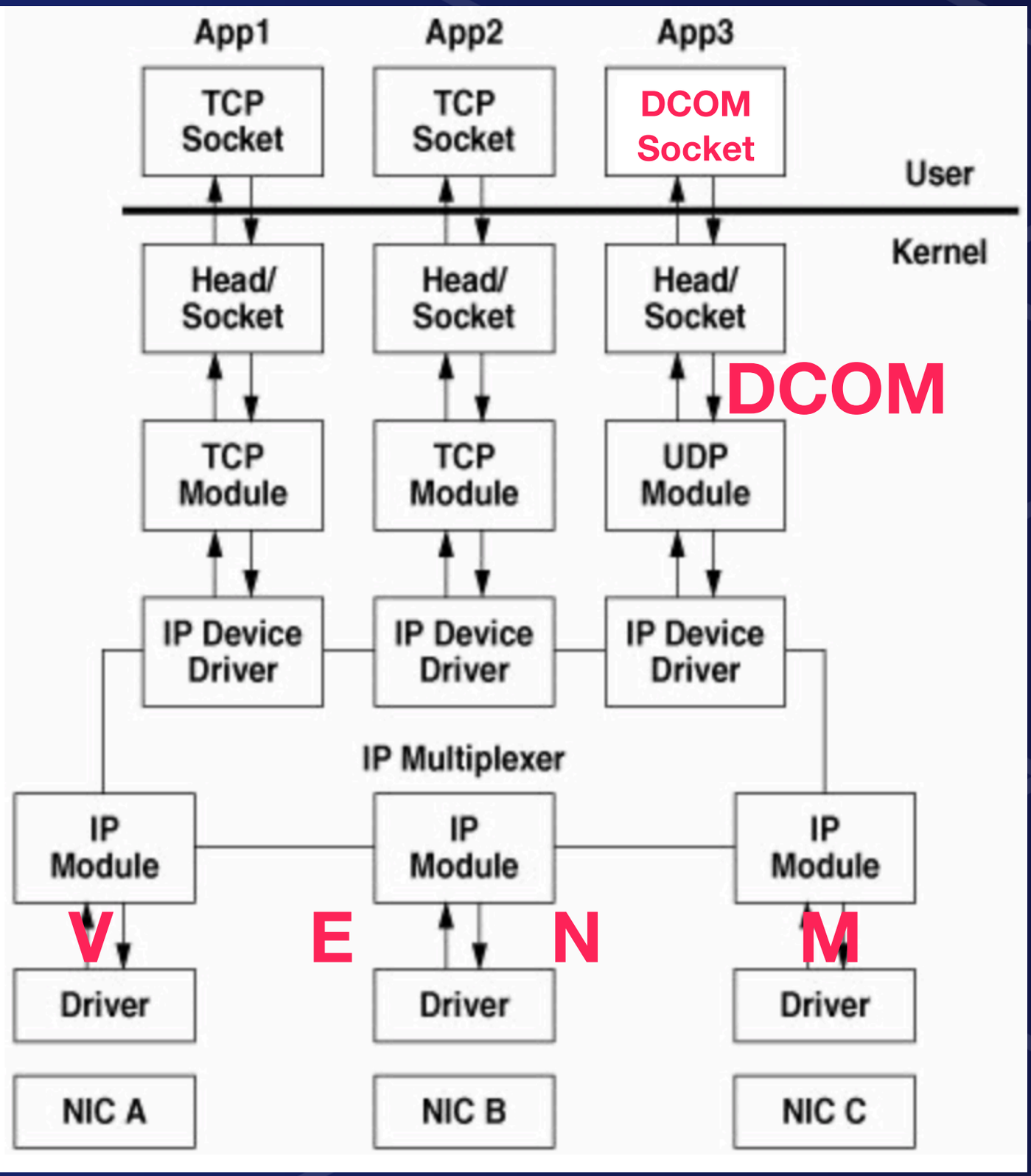
Reliability



Reliability



Reliability



Reliability

We got what we wanted - reliability in light of hardware failure

Reliability

We got what we wanted - reliability in light of hardware failure

We got extra - almost N-times performance in non-failure conditions

Reliability

We got what we wanted - reliability in light of hardware failure

We got extra - almost N-times performance in non-failure conditions

Didn't change a single line of code in our applications

Reliability

We got what we wanted - reliability in light of hardware failure

We got extra - almost N-times performance in non-failure conditions

Didn't change a single line of code in our applications

Not just our applications - every application

Reliability

We got what we wanted - reliability in light of hardware failure

We got extra - almost N-times performance in non-failure conditions

Didn't change a single line of code in our applications

Not just our applications - every application

ftp was the demo app

Coupling vs. Cohesion

"One goal of design is to minimize coupling between parts and to maximize cohesion within them."

Multi-Paradigm Design for C++ *James Coplien*

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

What's the Catch?

If this design is so great, why don't I know anything about it?

What's the Catch?

If this design is so great, why don't I know anything about it?

My personal opinions and guesses...

What's the Catch?

If this design is so great, why don't I know anything about it?

STREAMS is kernel only

What's the Catch?

If this design is so great, why don't I know anything about it?

STREAMS is kernel only

The user space interface has none of the awesomeness

What's the Catch?

If this design is so great, why don't I know anything about it?

STREAMS is kernel only

The user space interface has none of the awesomeness

You must load and run your modules in the kernel

What's the Catch?

If this design is so great, why don't I know anything about it?

STREAMS is kernel only

The user space interface has none of the awesomeness

You must load and run your modules in the kernel

Recommendation was no more than 6-7 modules

What's the Catch?

If this design is so great, why don't I know anything about it?

Poor Linux Support

What's the Catch?

If this design is so great, why don't I know anything about it?

Poor Linux Support

Initial LiS implementation soured many

What's the Catch?

If this design is so great, why don't I know anything about it?

Poor Linux Support

Initial LiS implementation soured many

Linux Fast-STREAMS 2006 - great throughput

What's the Catch?

If this design is so great, why don't I know anything about it?

Poor Linux Support

Initial LiS implementation soured many

Linux Fast-STREAMS 2006 - great throughput

Still kernel only

What's the Catch?

If this design is so great, why don't I know anything about it?

Lack of user-space availability

What's the Catch?

If this design is so great, why don't I know anything about it?

Lack of user-space availability

Some implementations available - ACE

What's the Catch?

If this design is so great, why don't I know anything about it?

Lack of user-space availability

Some implementations available - ACE

OK on throughput, not on latency

What's the Catch?

If this design is so great, why don't I know anything about it?

Lack of user-space availability

Some implementations available - ACE

OK on throughput, not on latency

Java style object oriented - difficult to compose

What's the Catch?

If this design is so great, why don't I know anything about it?

Performance

What's the Catch?

If this design is so great, why don't I know anything about it?

Performance

Full implementation is hard

What's the Catch?

If this design is so great, why don't I know anything about it?

Performance

Full implementation is hard

Can require custom scheduler for service routines

What's the Catch?

If this design is so great, why don't I know anything about it?

Performance

Full implementation is hard

Can require custom scheduler for service routines

Less chance for compiler optimizations

What's the Catch?

If this design is so great, why don't I know anything about it?

Performance

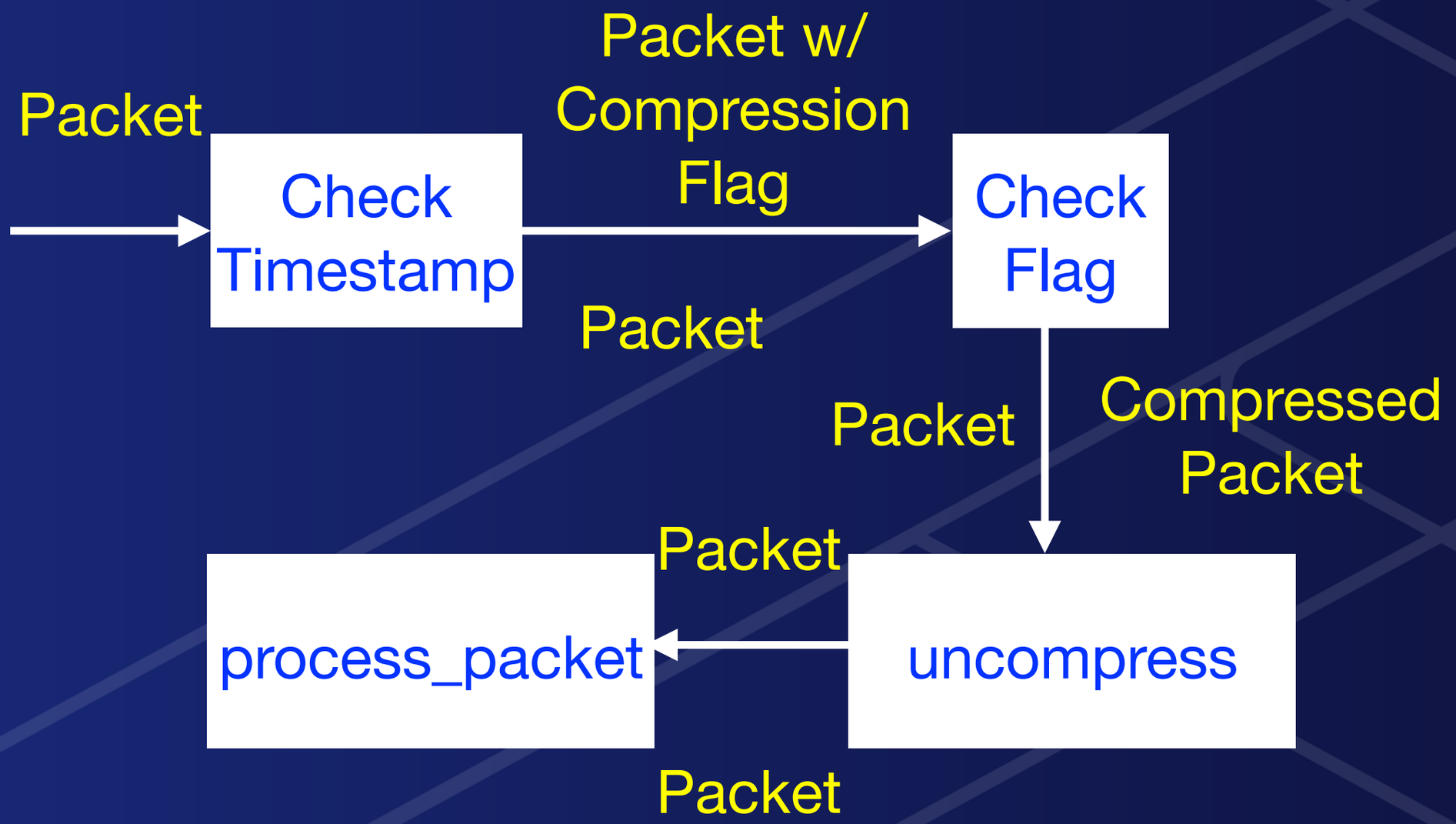
Full implementation is hard

Can require custom scheduler for service routines

Less chance for compiler optimizations

Function calls and runtime checks via opaque messages

Coupling vs. Cohesion



Ever Seen an Ugly Baby?

All parents think their baby is beautiful

Ever Seen an Ugly Baby?

All parents think their baby is beautiful

I am under no such illusion

Ever Seen an Ugly Baby?

All parents think their baby is beautiful

I am under no such illusion

Difficult to use wrong :-)

Ever Seen an Ugly Baby?

All parents think their baby is beautiful

I am under no such illusion

Difficult to use wrong :-)

Difficult to use right :-(

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Timestamp

```
auto check_timestamp = [](auto & fw, Packet const & pkt)
-> decltype(
    add_tag<HasCompressionFlag>(fw, pkt),
    bool{supports_compression(pkt)},
    void())
{
  if (supports_compression(pkt)) {
    put_next(fw, add_tag<HasCompressionFlag>(fw, pkt));
  } else {
    put_next(fw, pkt);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
     check_tagged<HasCompressionFlag>(fw, ev),
     bool{should_compress(event_for(fw, ev))},
     void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
     check_tagged<HasCompressionFlag>(fw, ev),
     bool{should_compress(event_for(fw, ev))},
     void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Check Flag

```
auto check_flag = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<HasCompressionFlag>(fw, ev),
    bool{should_compress(event_for(fw, ev))},
    void())
{
  if (should_compress(event_for(fw, ev))) {
    put_next(
        fw,
        add_tag<Compressed>(
            fw,
            remove_tag<HasCompressionFlag>(fw, ev)));
  } else {
    put_next(fw, ev);
  }
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
     check_tagged<Compressed>(fw, ev),
     uncompress(event_for(fw, ev)),
     void())
{
  put_next(
       fw,
       remove_tag<Compressed>(
            fw,
            tag_as(fw, ev)(
                 uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
     check_tagged<Compressed>(fw, ev),
     uncompress(event_for(fw, ev)),
     void())
{
  put_next(
       fw,
       remove_tag<Compressed>(
            fw,
            tag_as(fw, ev)(
                 uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
     check_tagged<Compressed>(fw, ev),
     uncompress(event_for(fw, ev)),
     void())
{
  put_next(
       fw,
       remove_tag<Compressed>(
            fw,
            tag_as(fw, ev)(
                 uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Uncompress

```
auto uncompress = [](auto & fw, auto const & ev)
-> decltype(
    check_tagged<Compressed>(fw, ev),
    uncompress(event_for(fw, ev)),
    void())
{
  put_next(
      fw,
      remove_tag<Compressed>(
          fw,
          tag_as(fw, ev)(
              uncompress(event_for(fw, ev)))));
};
```

Process Packet

```
auto process_packet = [](auto & fw, Packet const & pkt)
{
  dependency<ExchangeFooSesson>(fw).process_packet(pkt);
};
```

Building the Stream

```
auto strm = StreamHead
     | check_timestamp
     | check_flag
     | uncompress
     | process_packet
     ;
```

Building the Stream

```
auto strm = StreamHead
     | check_timestamp
     | check_flag
     | uncompress
     | process_packet
     ;
```

These are all one-way; auto deduced - Easy Button

Building the Stream

```
auto strm = StreamHead
     | check_timestamp
     | check_flag
     | uncompress
     | process_packet
     ;
```

These are all one-way; auto deduced - Easy Button

Can make modules directly with much more optons

Building the Stream

```
auto strm = StreamHead
     | check_timestamp
     | check_flag
     | uncompress
     | process_packet
     ;
```

These are all one-way; auto deduced - Easy Button

Can make modules directly with much more optons

Can create modules whose put takes variadic arguments

The Holy Grail?

No - but it looks like a grail, and acts like a grail,
and when I close my eyes I can hear
the sound of coconuts clapping together

Journey Before Destination

The most important words a man can say are, “I will do better.” These are not the most important words any man can say. I am a man, and they are what I needed to say. The ancient code of the Knights Radiant says “journey before destination.” Some may call it a simple platitude, but it is far more. A journey will have pain and failure. It is not only the steps forward that we must accept. It is the stumbles. The trials. The knowledge that we will fail. That we will hurt those around us. But if we stop, if we accept the person we are when we fall, the journey ends. That failure becomes our destination. To love the journey is to accept no such end. I have found, through painful experience, that the most important step a person can take is always the next one.

-BRANDON SANDERSON, *OATHBRINGER*

Many Unanswered Questions

Many Unanswered Questions

The answer to most of them is, yes, I have a working framework

Many Unanswered Questions

The answer to most of them is, yes, I have a working framework

I've had a "working" framework for a long time,
I just still don't like the impositions on the user

Many Unanswered Questions

The answer to most of them is, yes, I have a working framework

I've had a "working" framework for a long time,
I just still don't like the impositions on the user

C++ 20 promises to help greatly
but I can't use C++ 20 at work, so I've held off

Many Unanswered Questions

The answer to most of them is, yes, I have a working framework

I've had a "working" framework for a long time,
I just still don't like the impositions on the user

C++ 20 promises to help greatly
but I can't use C++ 20 at work, so I've held off

Other Questions?

MAYSTREET

CppCon 2022

Using Modern C++ to Revive and Old Design

AKA: Coupling and Cohesion are Guiding Lights

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