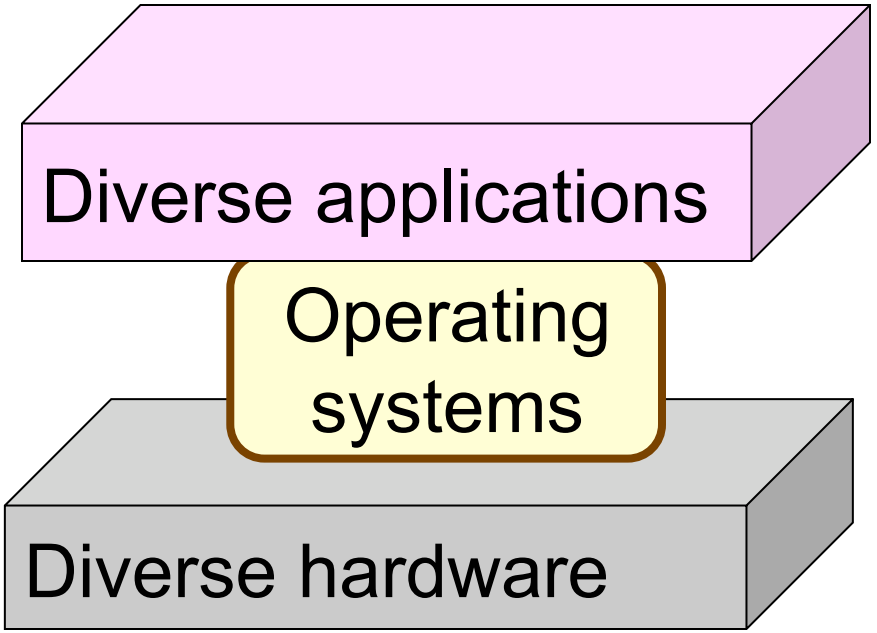
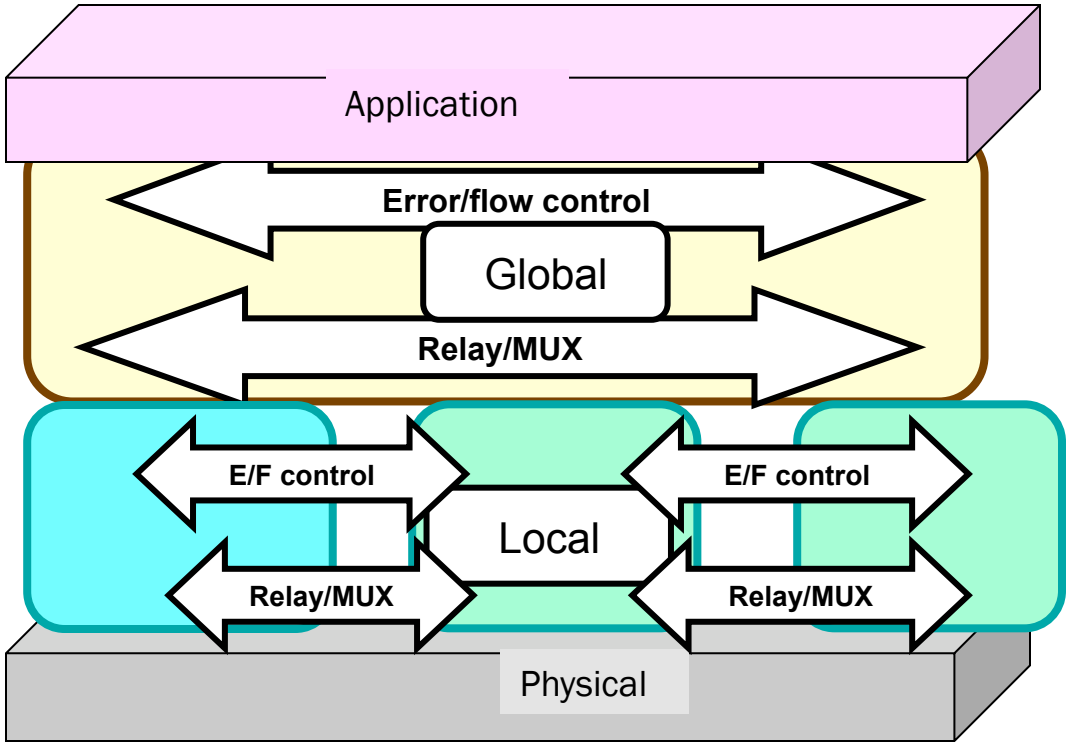
Cmplx Net Arch: Networking OS





NetME and the Internet

- Internet architecture is mix of
  - accident,
  - history, and
  - necessity,
- Sorting these out, extracting lessons learned, is subtle, challenging, …
- Spectacular success has blinded us…
- TCP/IP architecture has become “gravity,” an unchangeable feature of reality

Theory and the Internet

- Tremendous progress last decade in laying theoretical foundations, and…
- Continued incremental progress (patching) but…
- Many robustness issues acute and unaddressed (secure, verify, manage, maintain, …)
- Continued patching may never get us there, but…
- How do we change “gravity”?
- Question our assumptions about
  - The existing Internet architecture
  - The limits of our theory

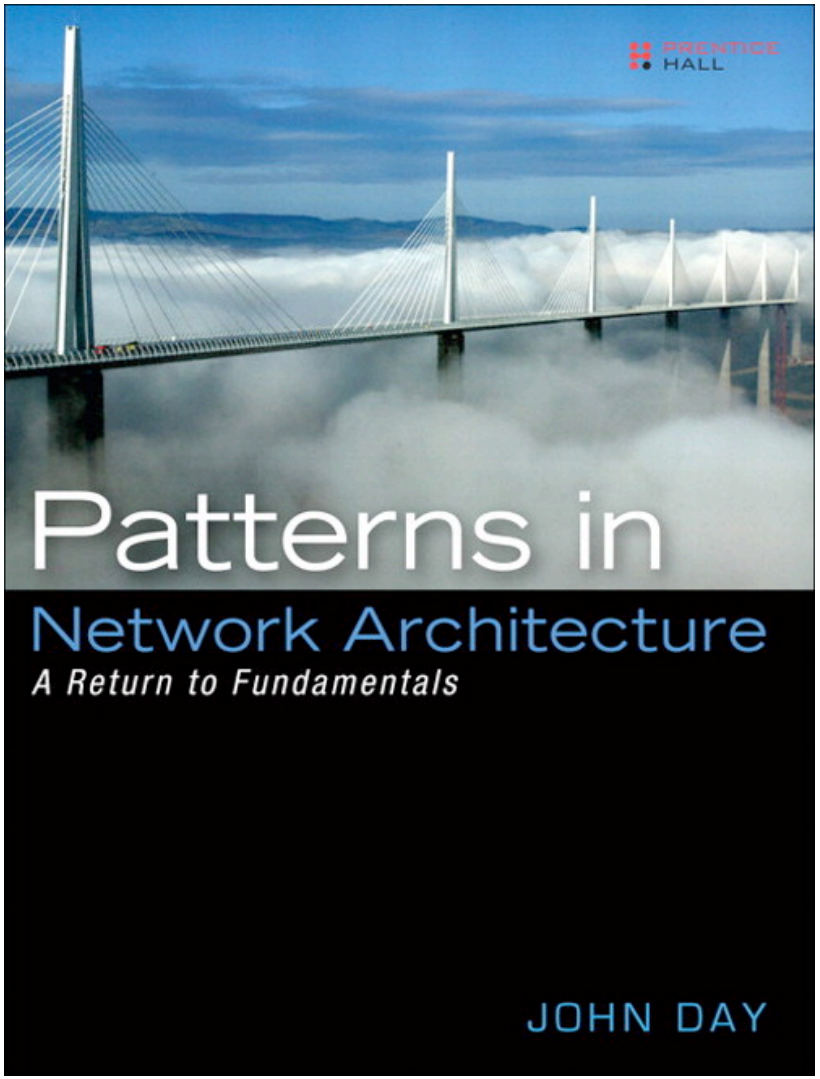
“back to basics”

- Our strategy: Think of Internet as NetOS
- Back to original starting point: operating systems
- Leverage much recent “rethinking”
  - Appears fragmented and incoherent, but…
  - There may be an opportunity for more unification
  - (Just like the theory side)
- Start with Day’s PNA
- Later add ISI/USC Touch et al and others

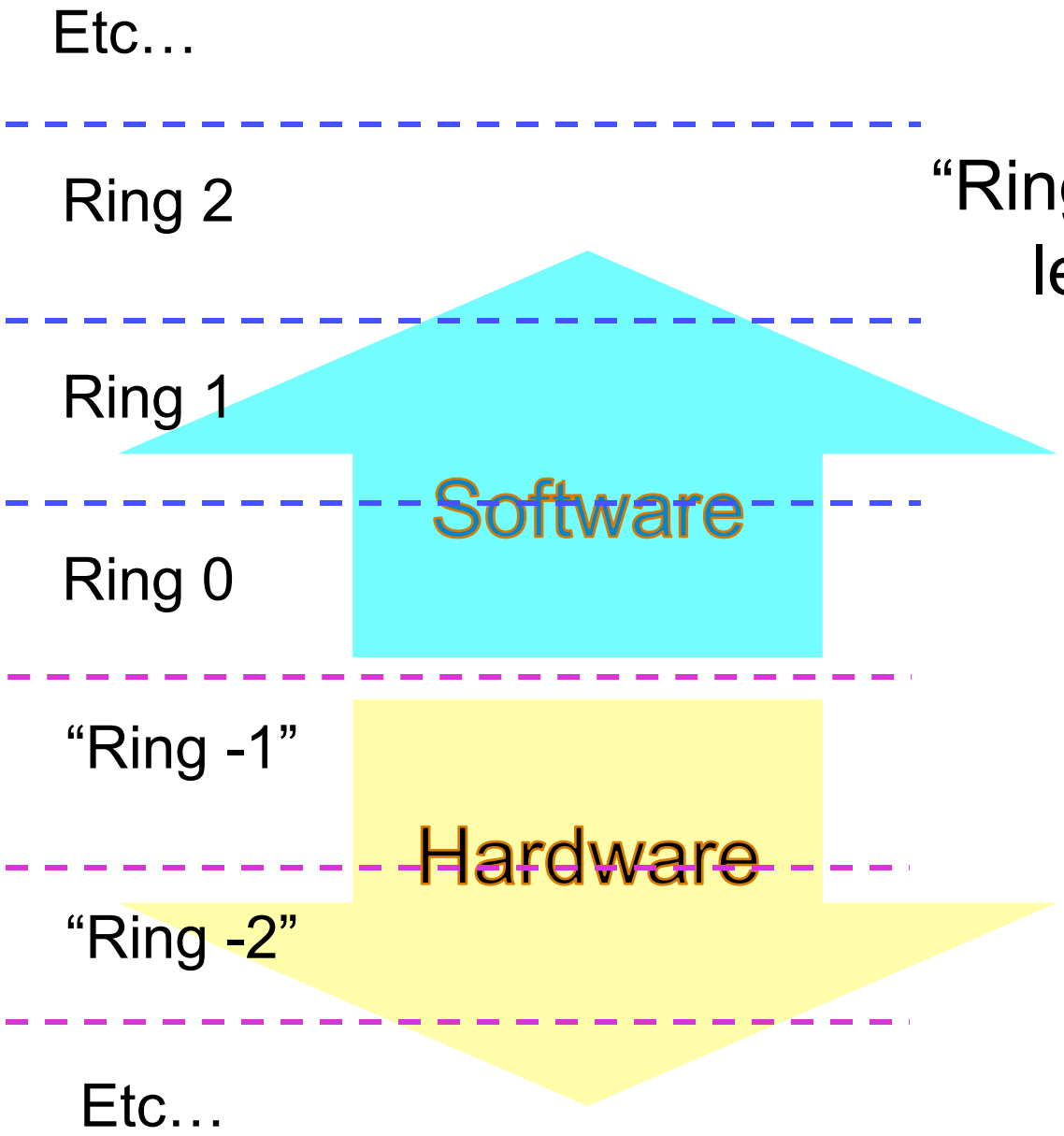
Internet as NetOS

- Decomposition of “NetOS” into an OS kernel, around which we nucleate the “Net” dimension
- Leverage others onthe “OS” part right
- We focus on “policy” for ctrl and mgmt
- Keep chasing “architecture”
- Special cases handled naturally: virtualization, data centers, clouds etc.

PNA



“return to fundamentals”

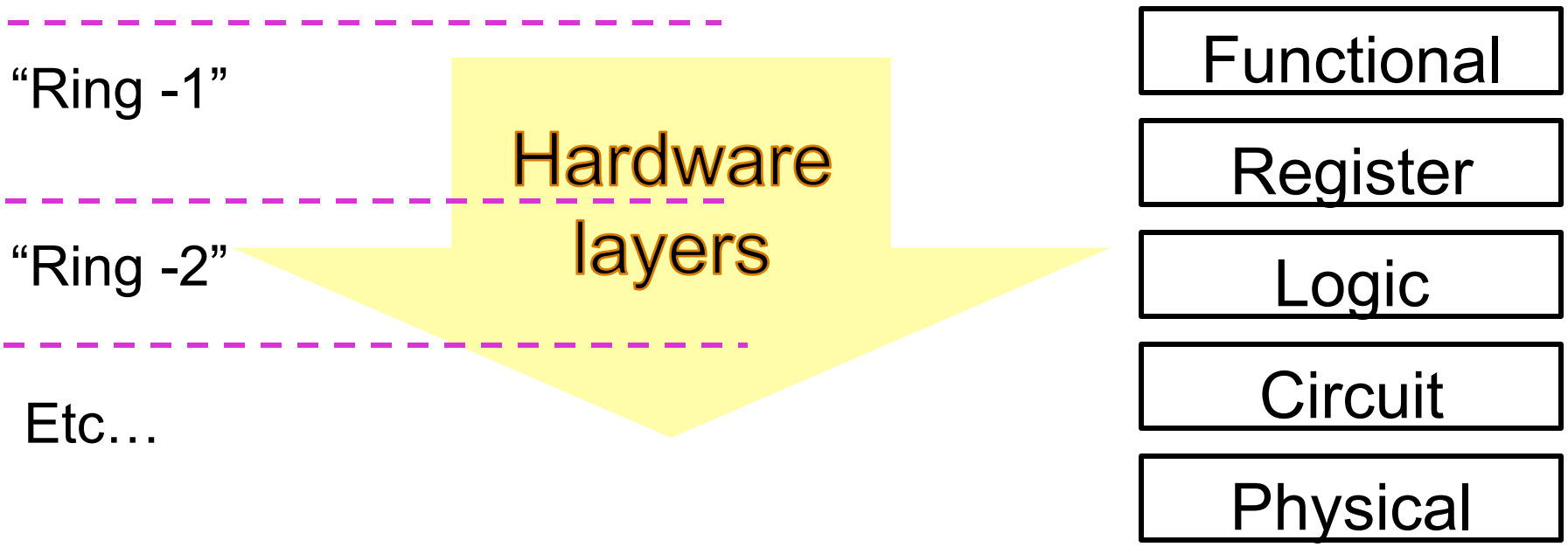


“Rings” are HW defined levels of “protection”

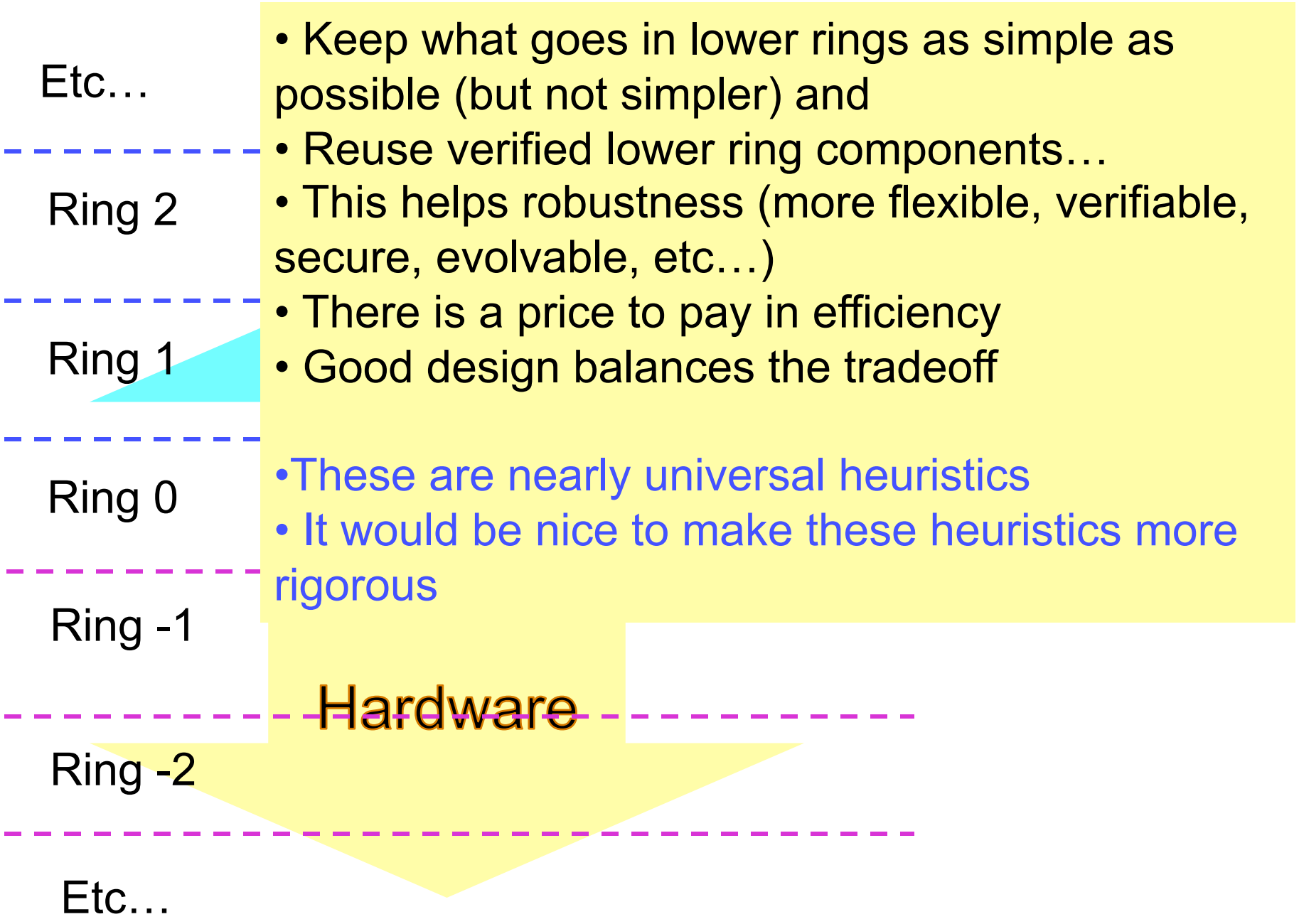
Start at SW/HW interface within a single processor

- Negative rings don't mean the same thing
- They would correspond to abstraction layers in hardware design
- We will temporarily defer HW, but..
- PBD is very compatible
- Often a key design issue in PBD is where to put the HW/SW boundary
- The PNA view of layering can be viewed as a special case of PBD

Platform Based Design (PBD)

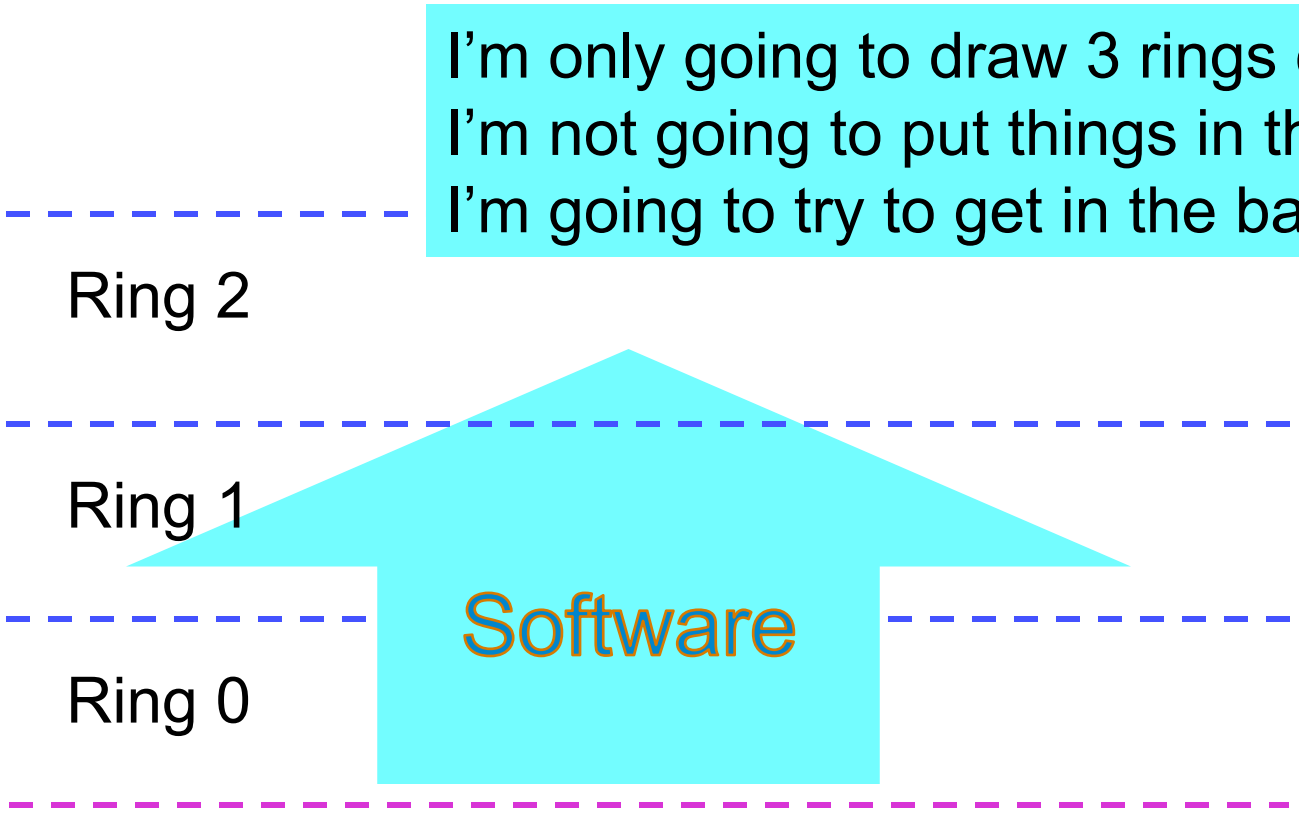


Design heuristics (KISS or E2E)

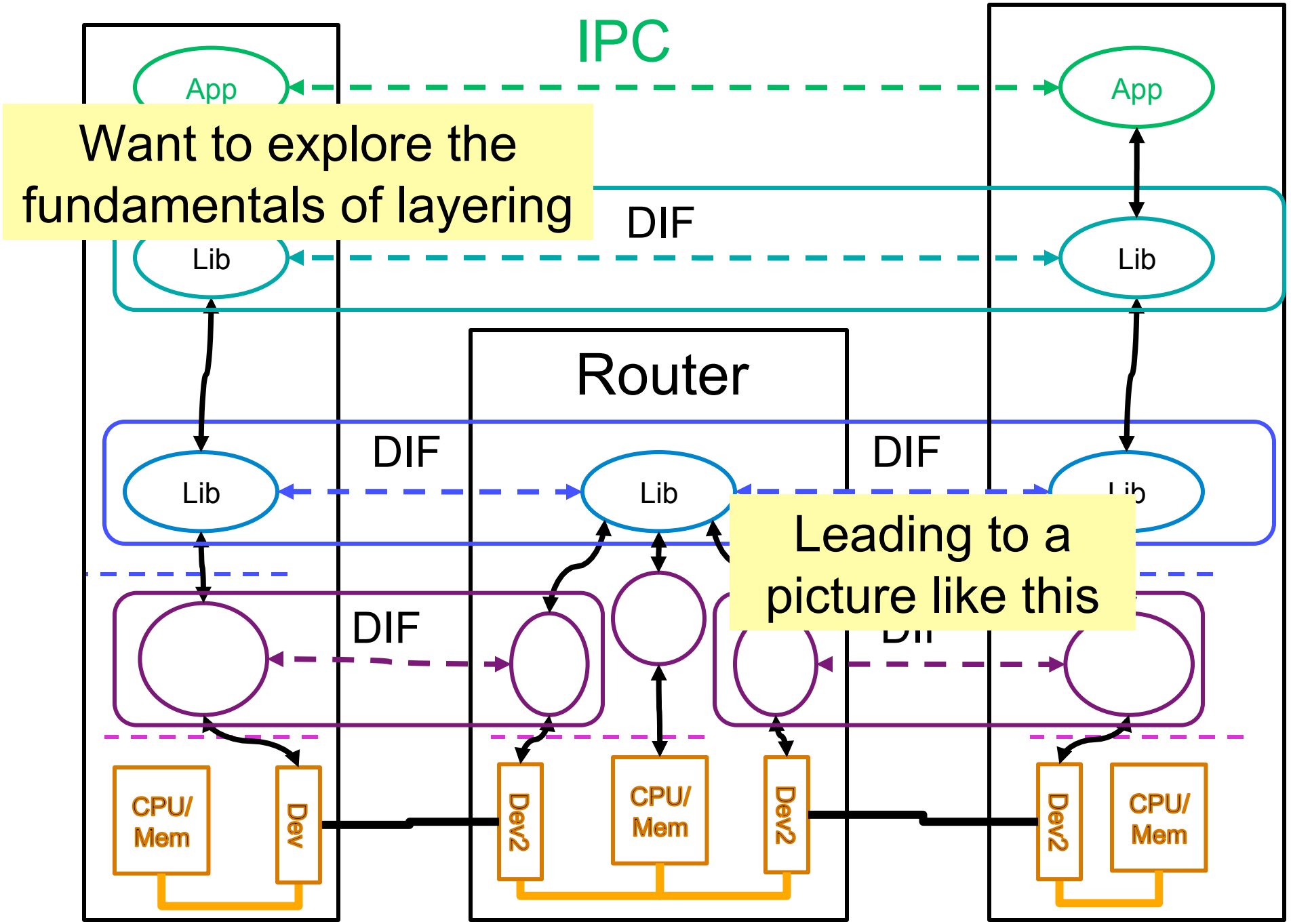


My first mistake…

I'm only going to draw 3 rings of software and I'm not going to put things in the right rings, but I'm going to try to get in the ballpark…



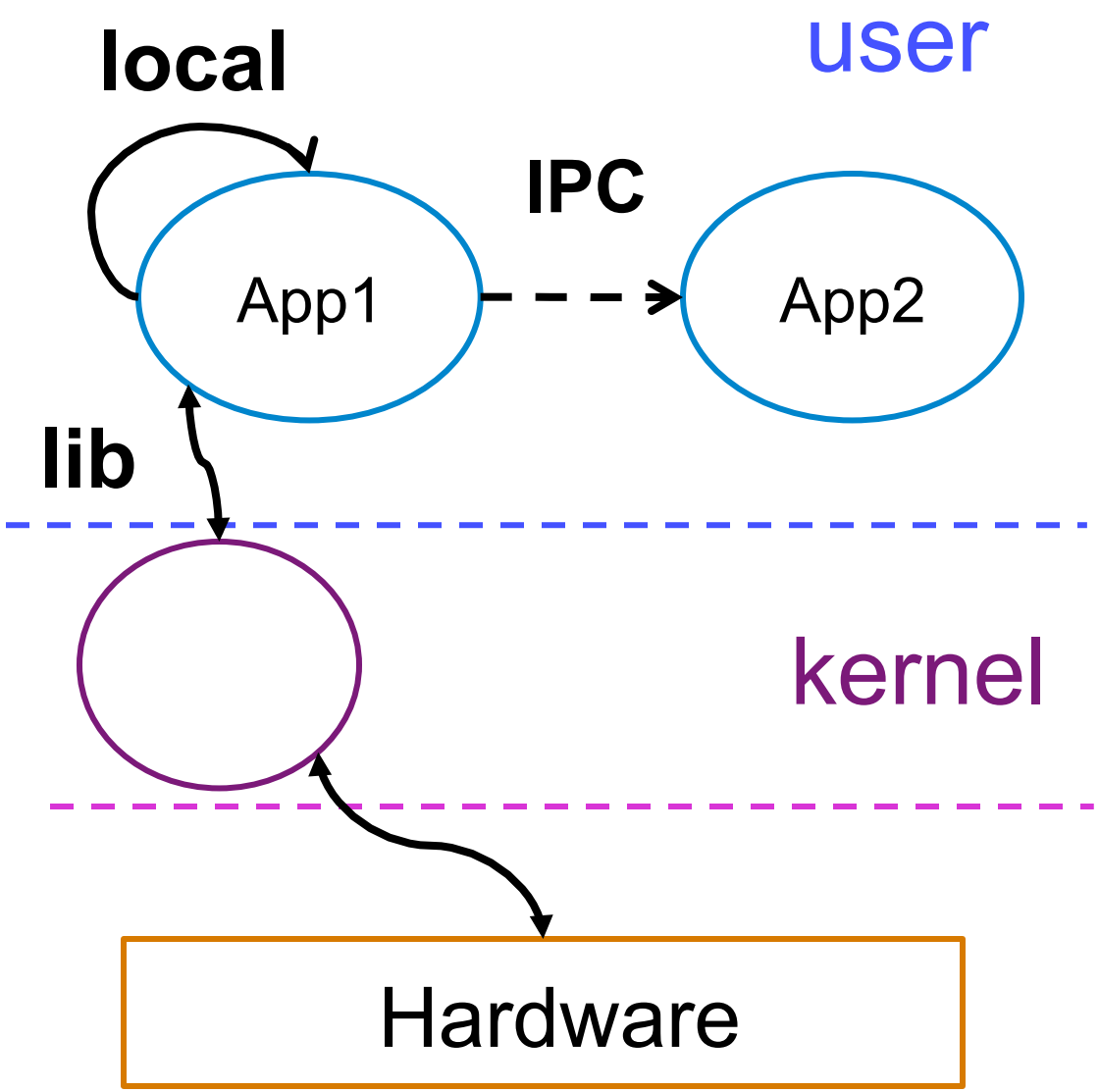
- I'm not going to do a very good job of drawing the HW
- Actually I won't do a good job of drawing anything but I think the hardware will be really bad.
- No rings of hardware.


IPC
App
App
Want to explore the fundamentals of layering
DIF
Lib
Lib
Router
DIF
DIF
Lib
Lib
Lib
Leading to a picture like this
DIF
CPU/ Mem
Dev
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem

Within a single processor

A function call can be

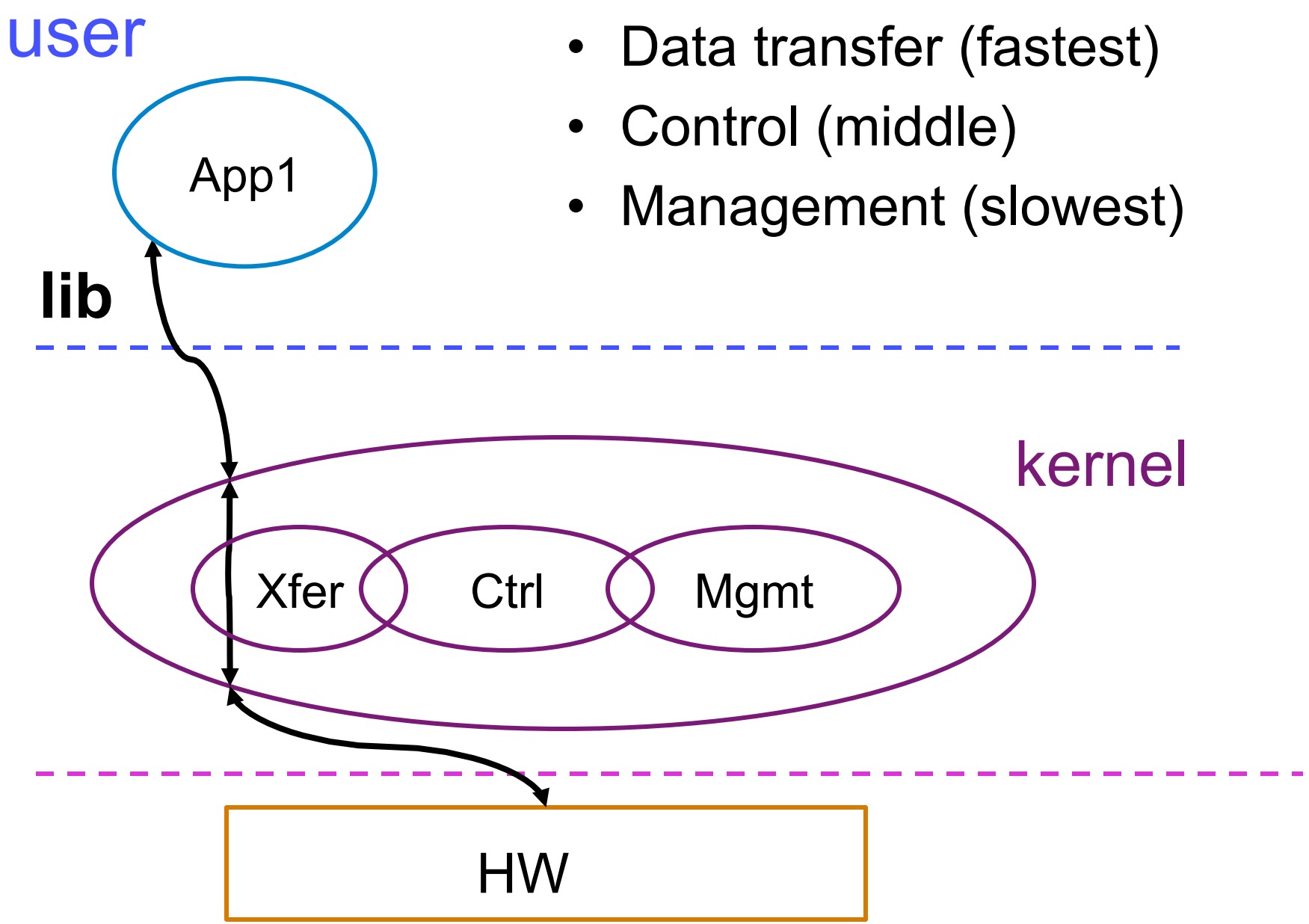
- Local
- Library (system)
- IPC



IPC= InterProcess Communication

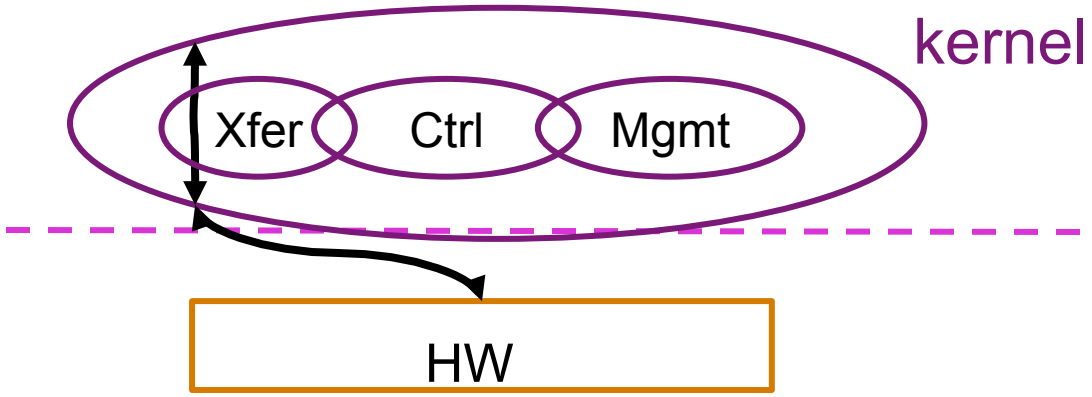
The kernel functions are

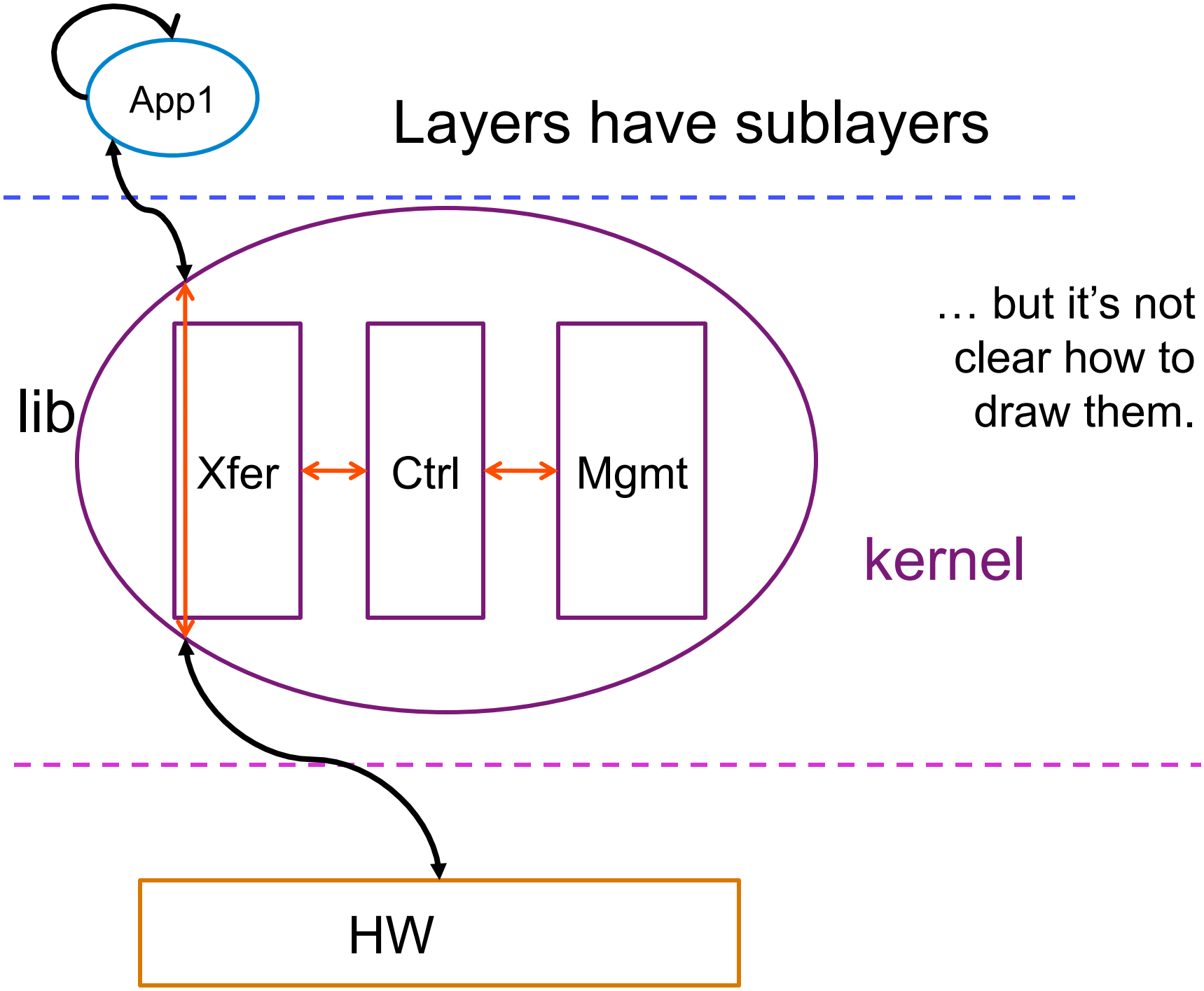
- Data transfer (fastest)
- Control (middle)
- Management (slowest)

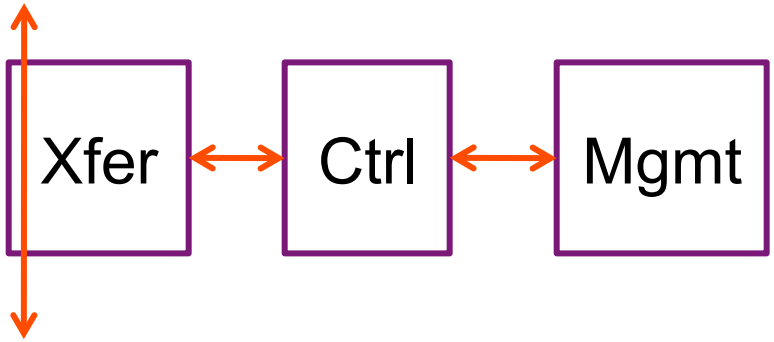


The kernel functions are

- Data transfer (fastest time scale)
  - Within memory (and memory hierarchies)
  - Between devices and memory
  - Between memory and computing elements
- Control (middle time scales)
  - Scheduling/Multiplexing resources
  - In time and space
- Management (slowest time scale)
  - ***What*** resources are available?
  - ***Where*** are they?

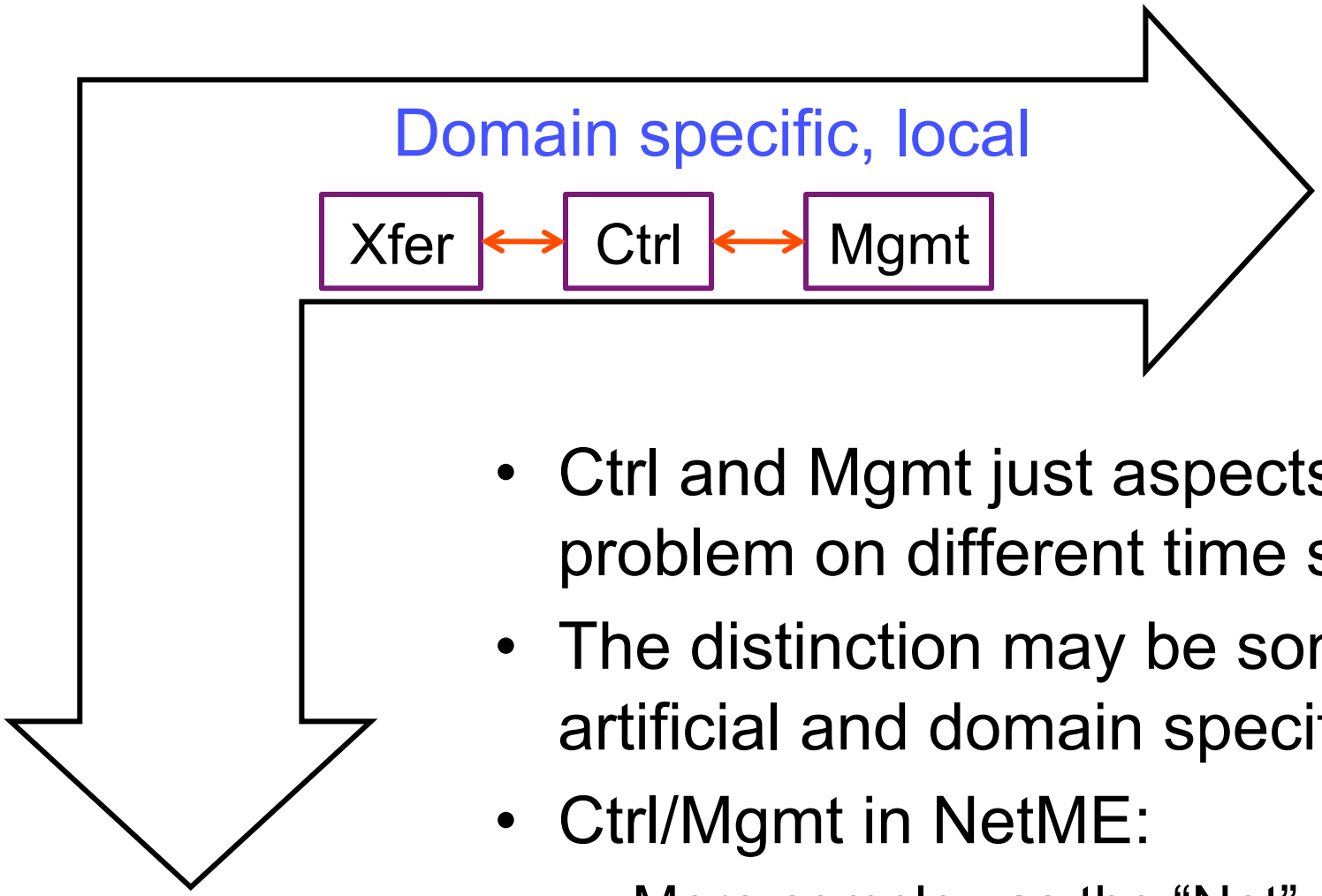



App1
Layers have sublayers
lib
Xfer
Ctrl
Mgmt
… but it's not clear how to draw them.
kernel
HW

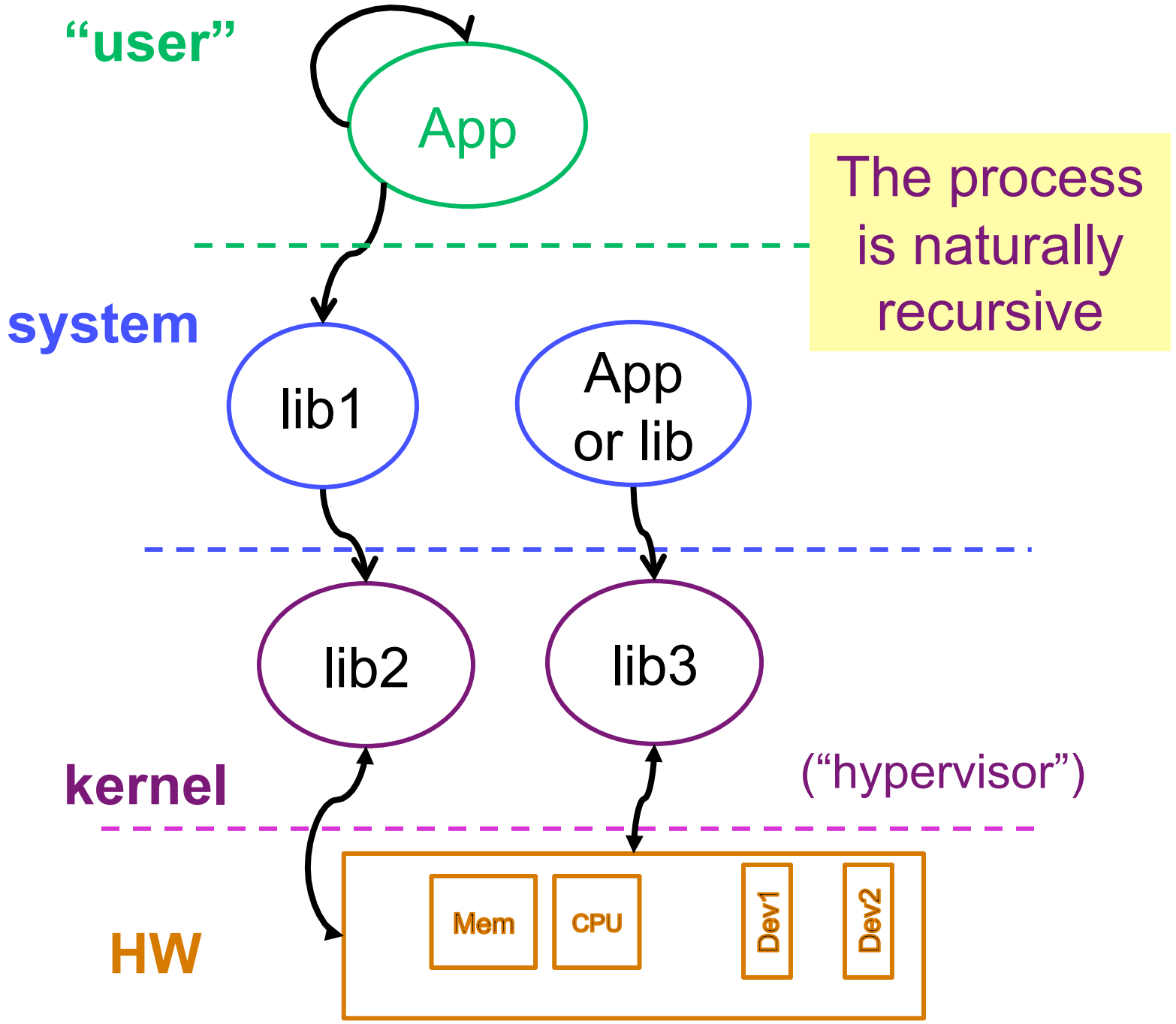


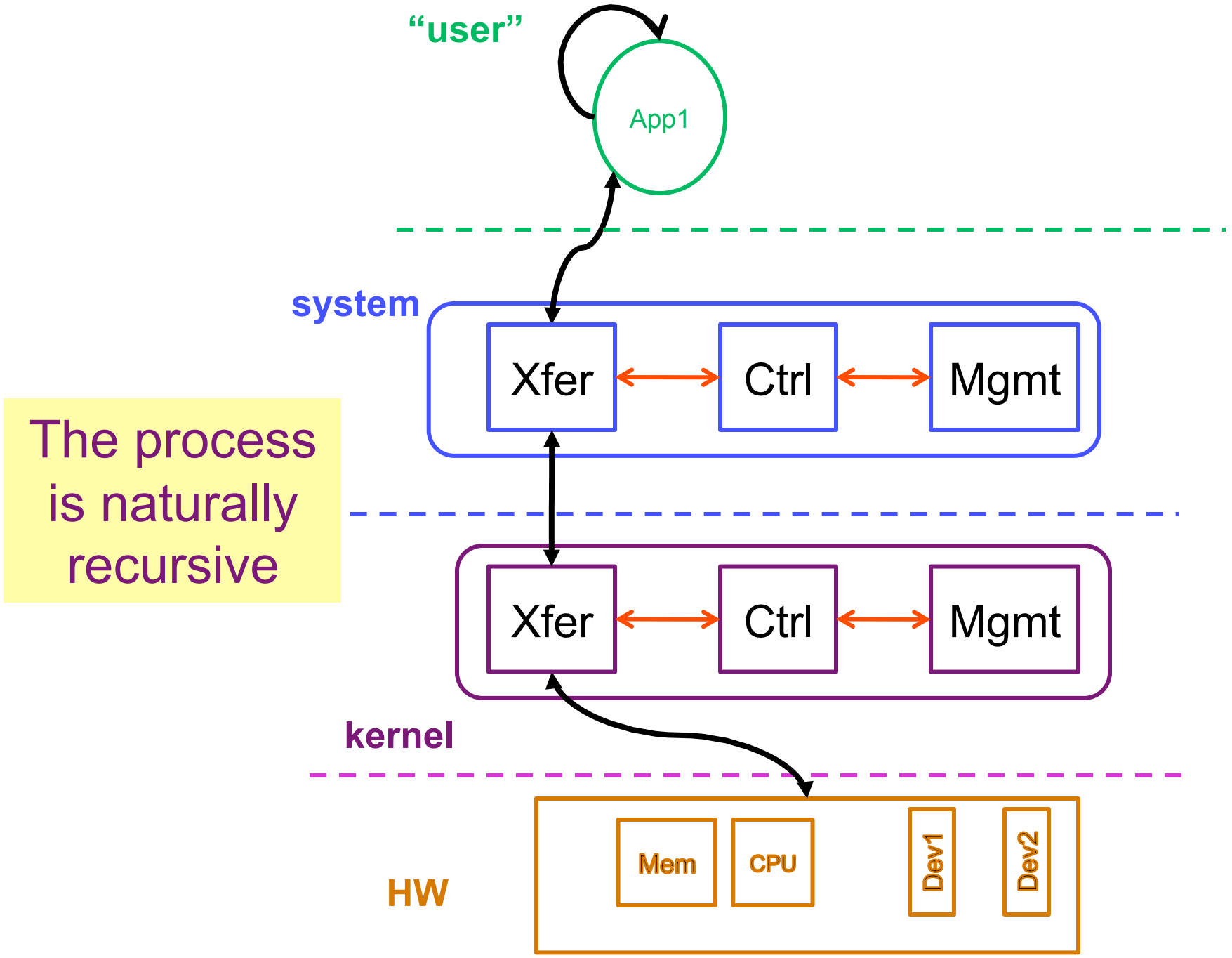
Universal functions?

- Transfer or transform (fastest)
  - Domain specific (data, power, goods, etc)
  - Depends on demand and supply of resources
- Control (middle)
  - Schedule/MUX resources in time and space
  - Flow and error control
- Management (slowest)
  - ***What*** resources are available?
  - ***Where*** are they?
  - Cost? Risk? etc



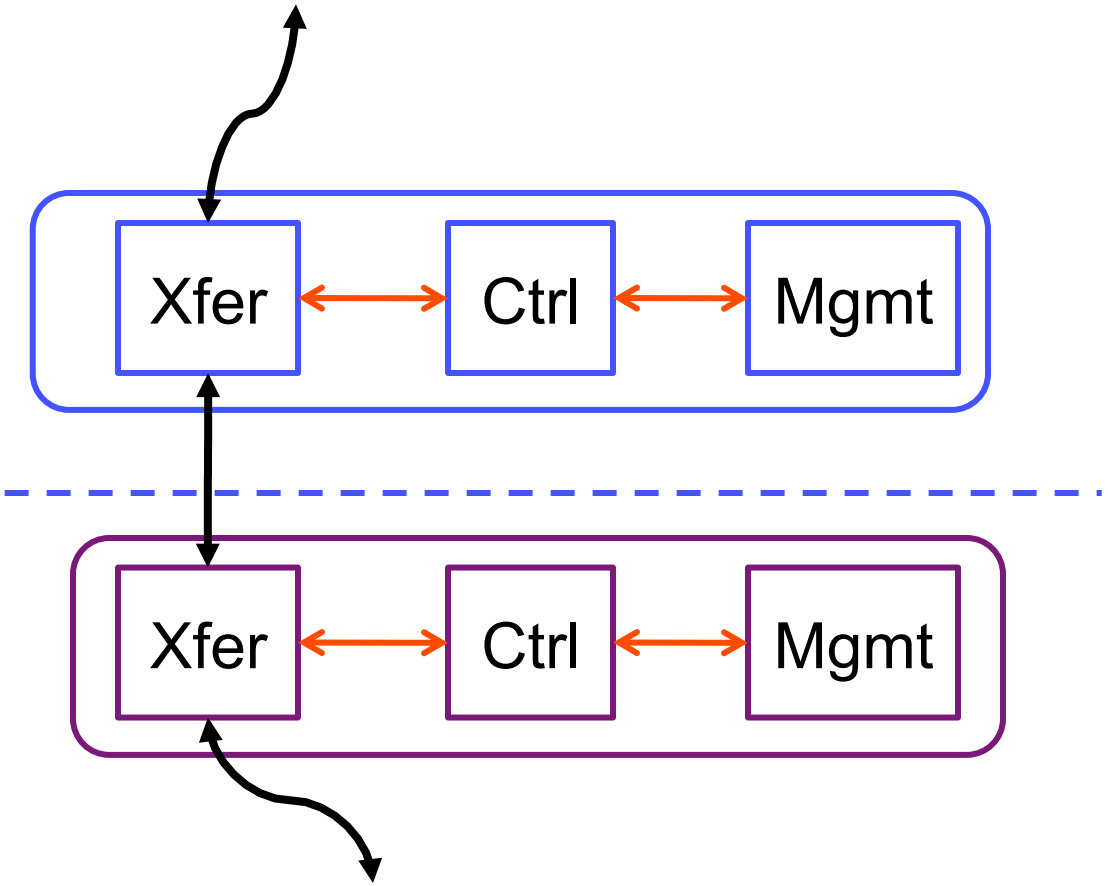
- Ctrl and Mgmt just aspects of a single problem on different time scales
- The distinction may be somewhat artificial and domain specific
- Ctrl/Mgmt in NetME:
  - More complex as the “Net” part grows
  - Will be our focus/goal of a unified theory
  - From physics to information to computation to control

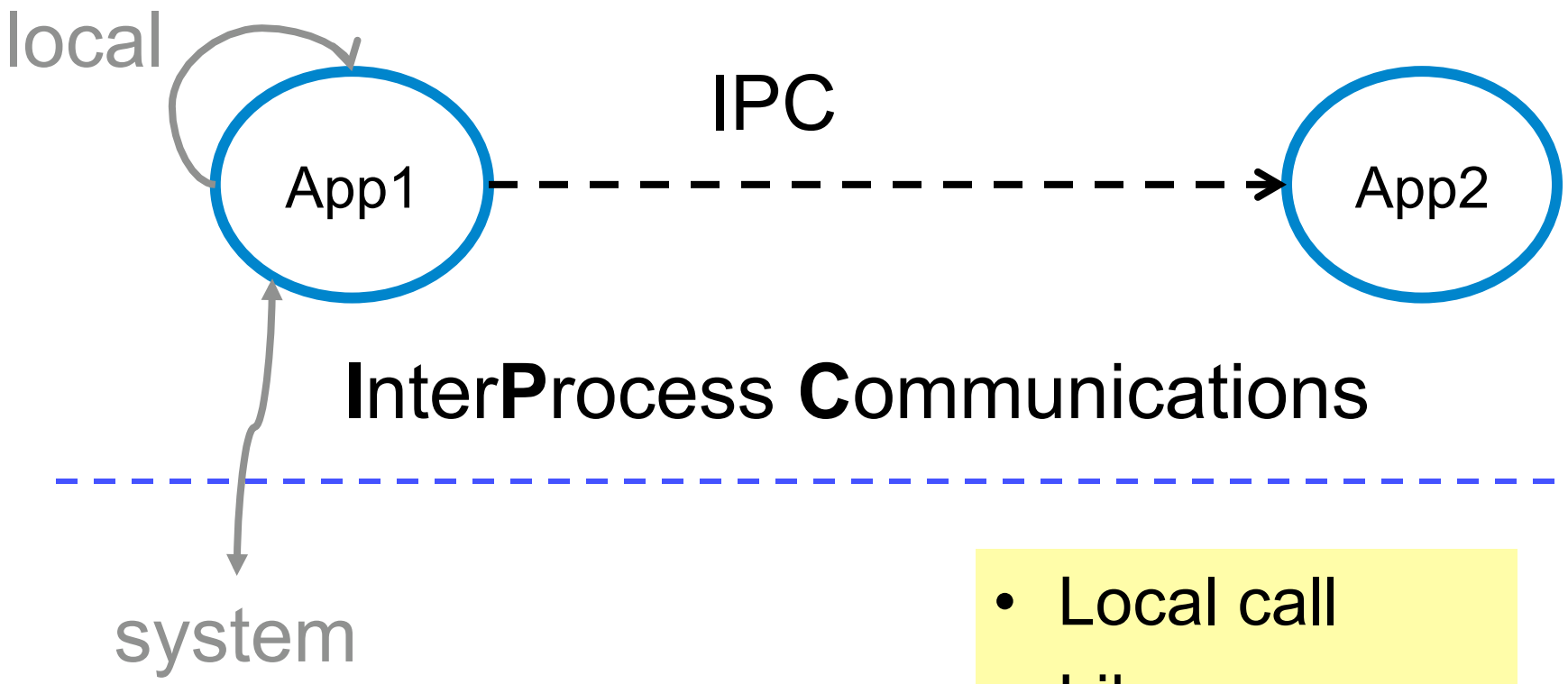

"user"
App
The process is naturally recursive
system
lib1
App or lib
lib2
lib3
kernel
("hypervisor")
HW
Mem
CPU
Dev1
Dev2


"user"
App1
system
Xfer
Ctrl
Mgmt
The process is naturally recursive
Xfer
Ctrl
Mgmt
kernel
HW
Mem
CPU
Dev1
Dev2

Layers have sublayers

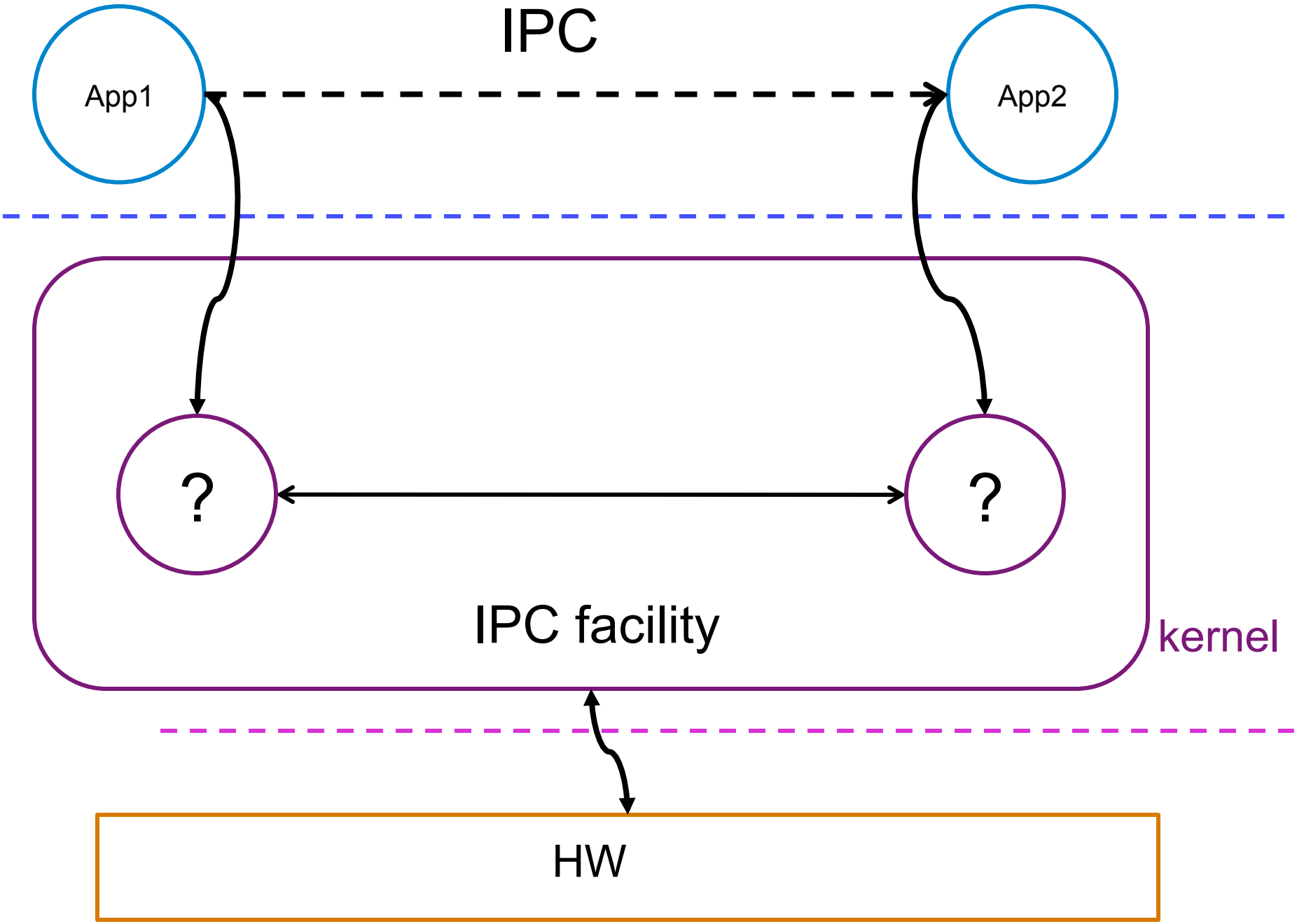
Layers are naturally recursive

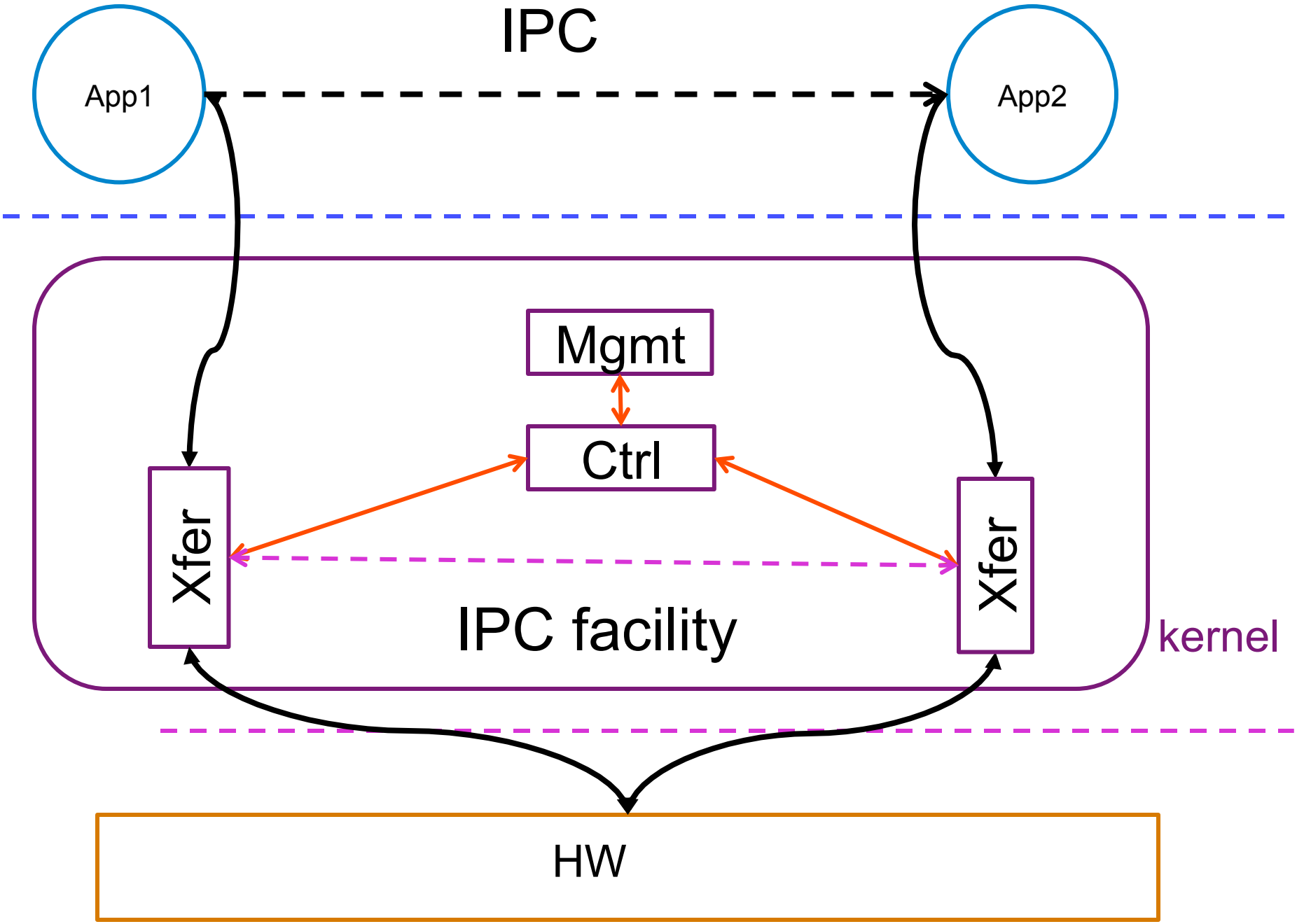




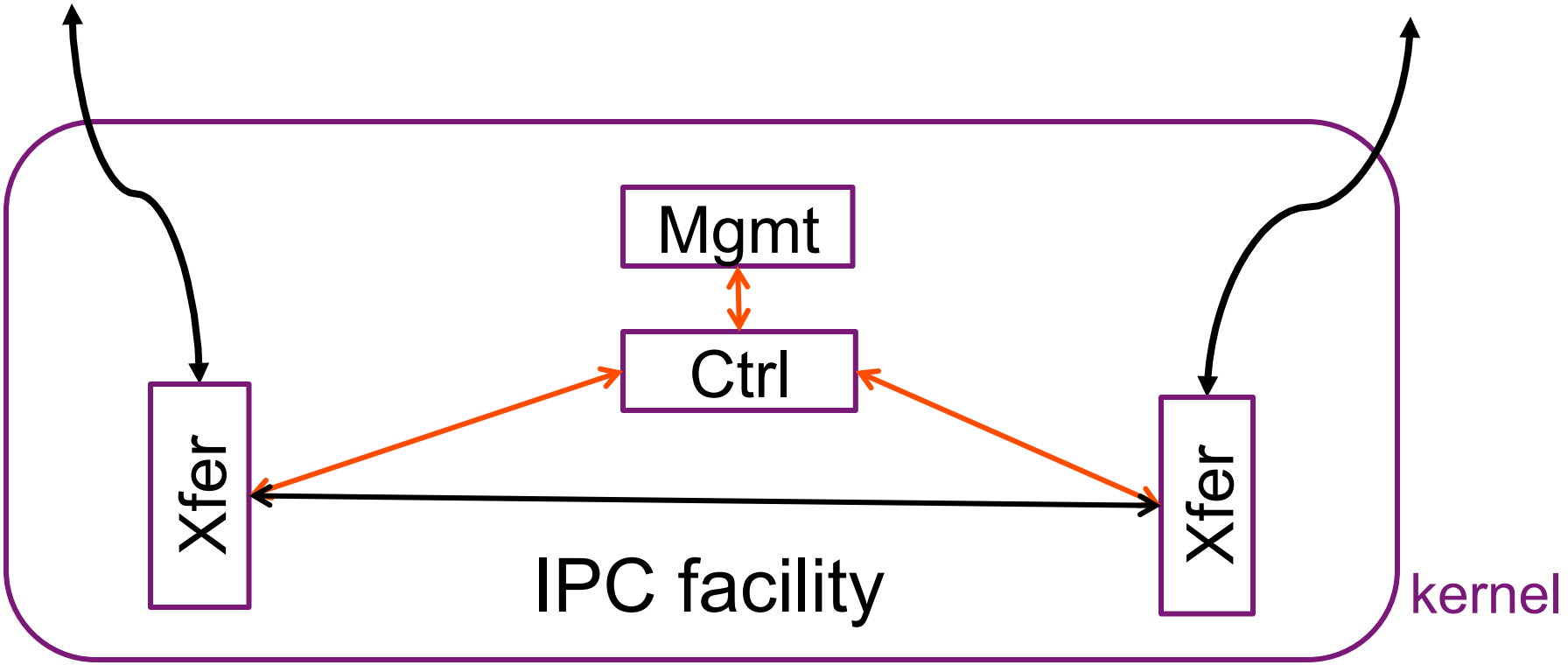
- Local call
- Library (system) call
- IPC

Want them all to behave similarly.


IPC
App1
App2
?
?
IPC facility
kernel
HW

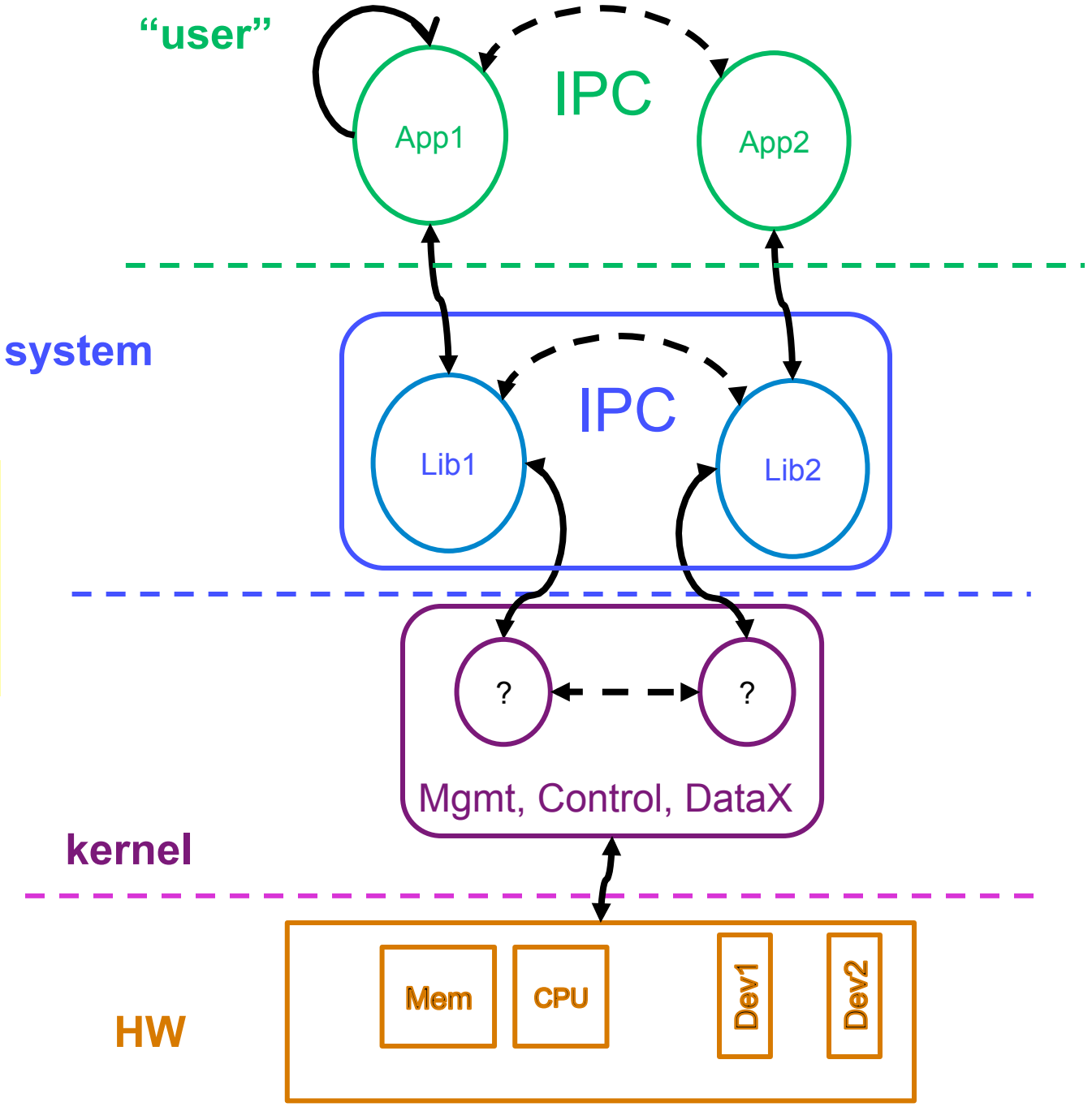

IPC
App1
App2
Mgmt
Ctrl
Xfer
Xfer
IPC facility
kernel
HW

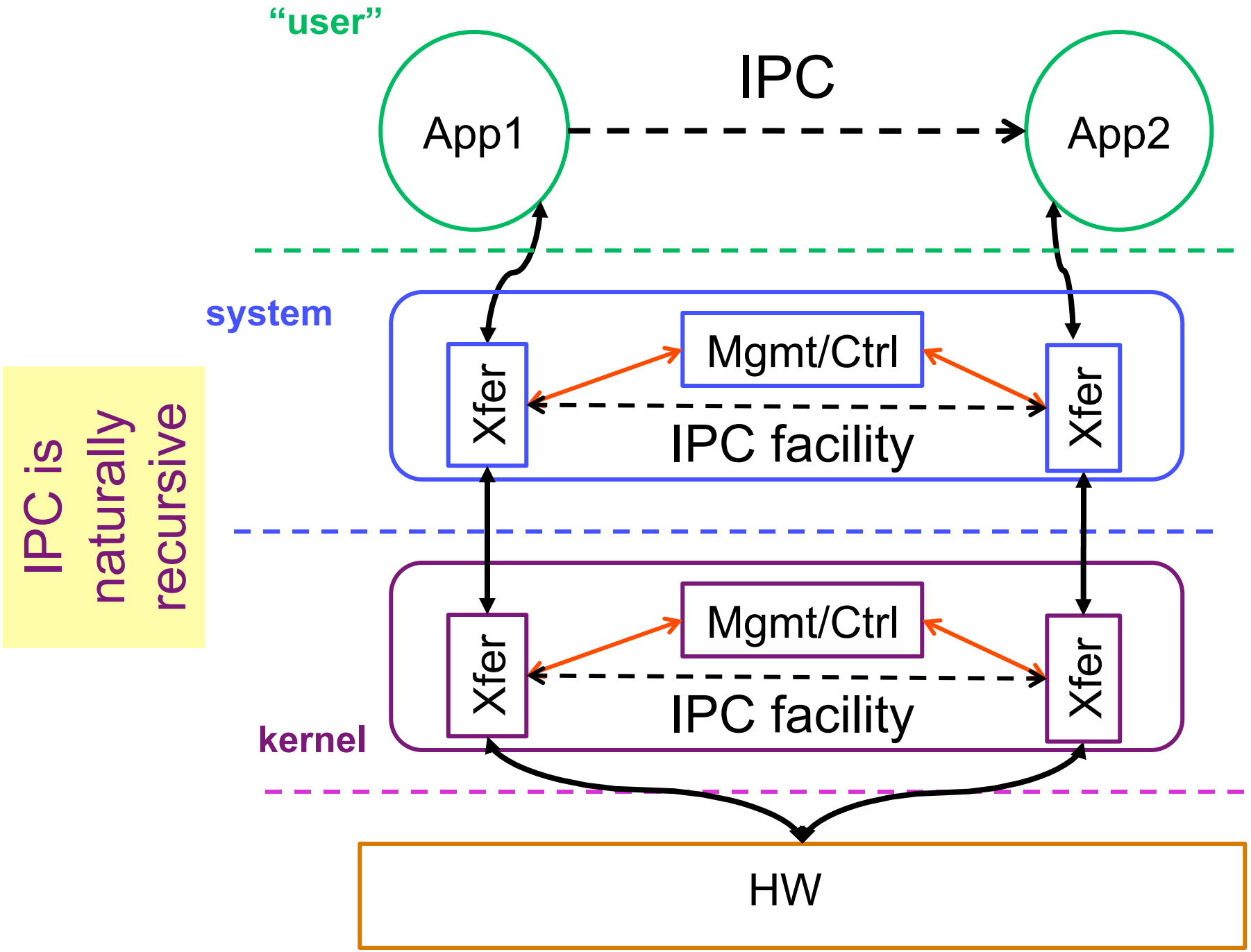
Layers have sublayers



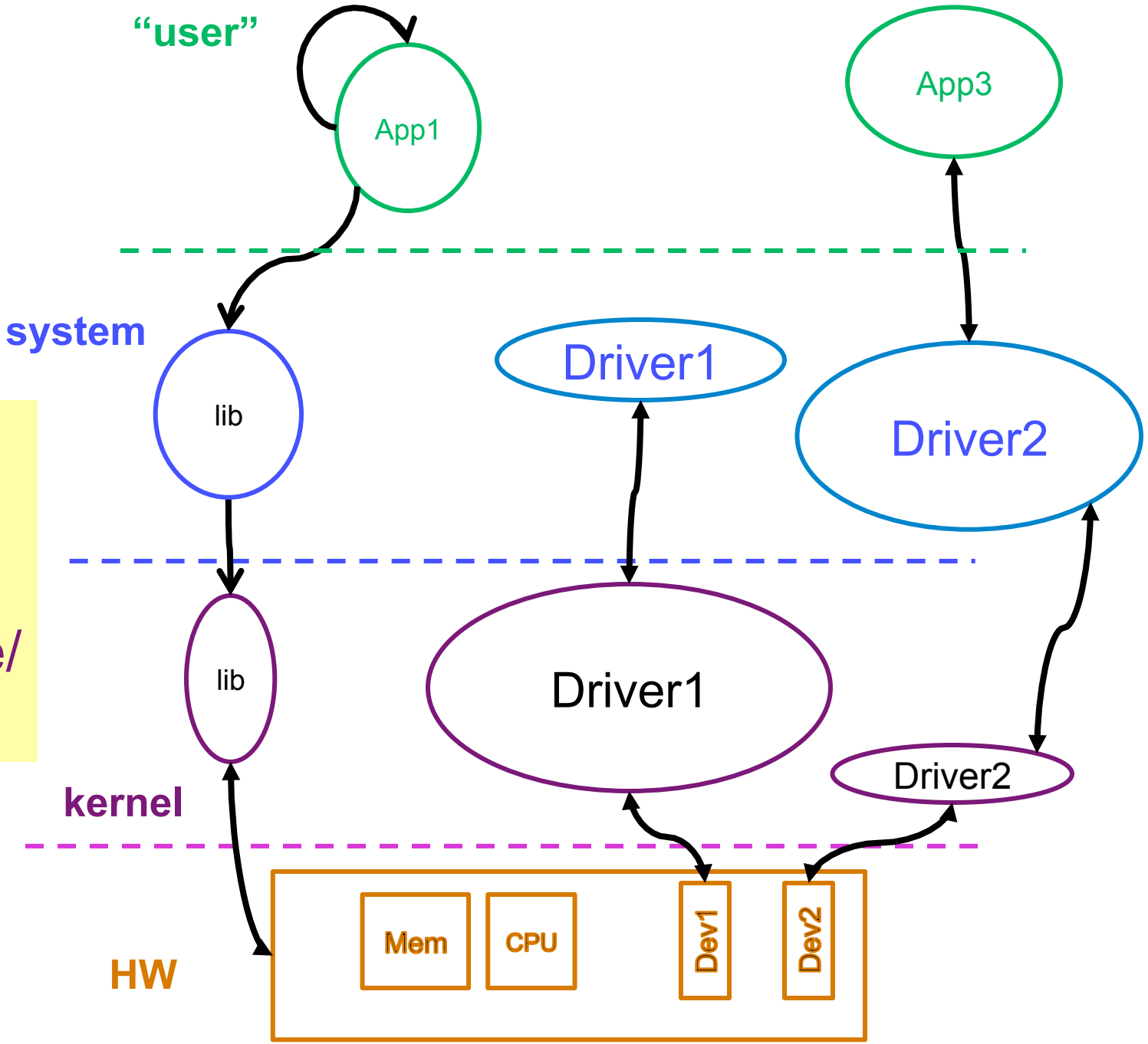
… but it’s not clear how to draw them.

IPC is naturally recursive

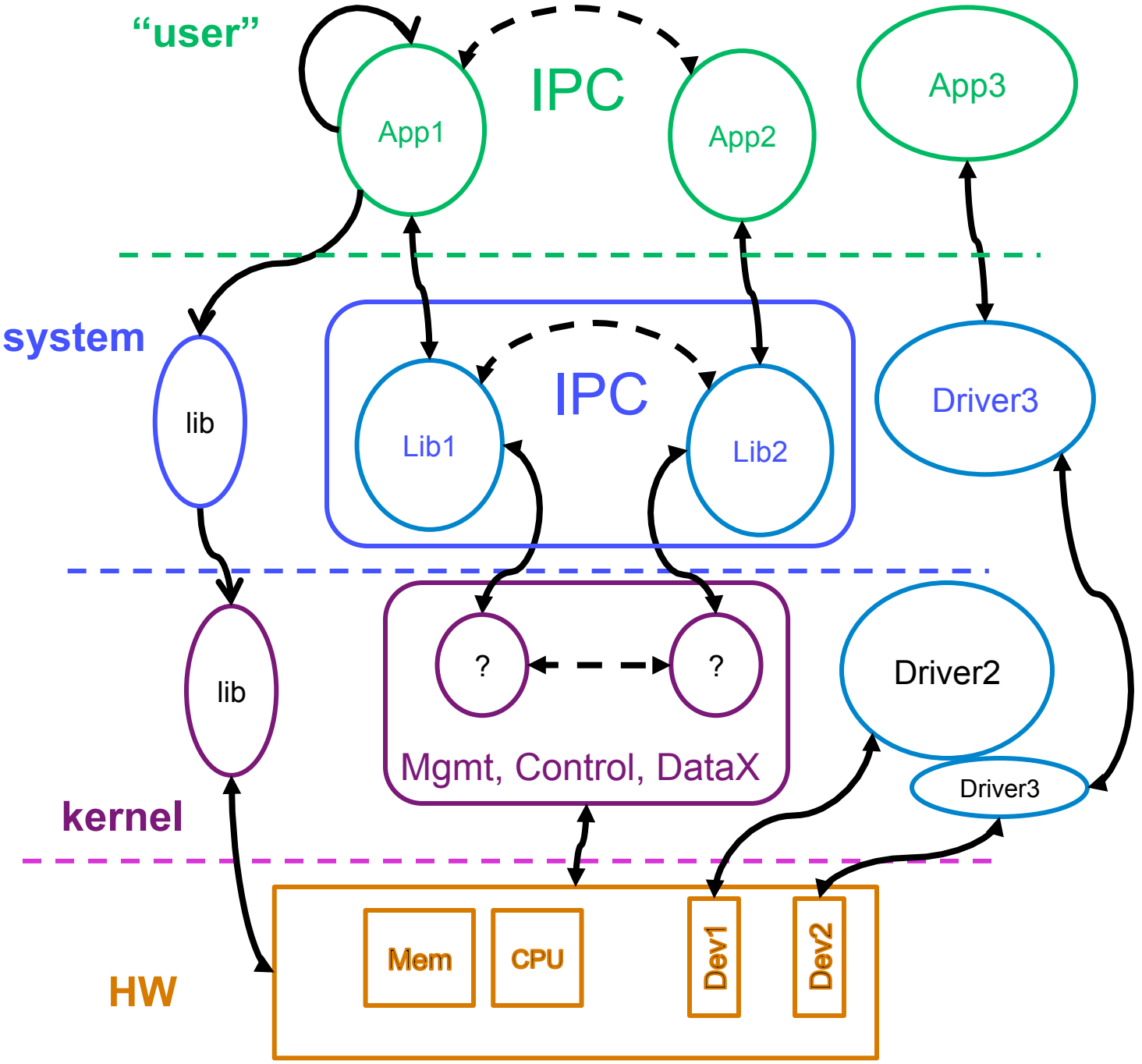



"user"
IPC
App1
App2
system
Xfer
Mgmt/Ctrl
Xfer
IPC facility
IPC is naturally recursive
Xfer
Mgmt/Ctrl
Xfer
IPC facility
kernel
HW

Design choices effect performance/ robustness

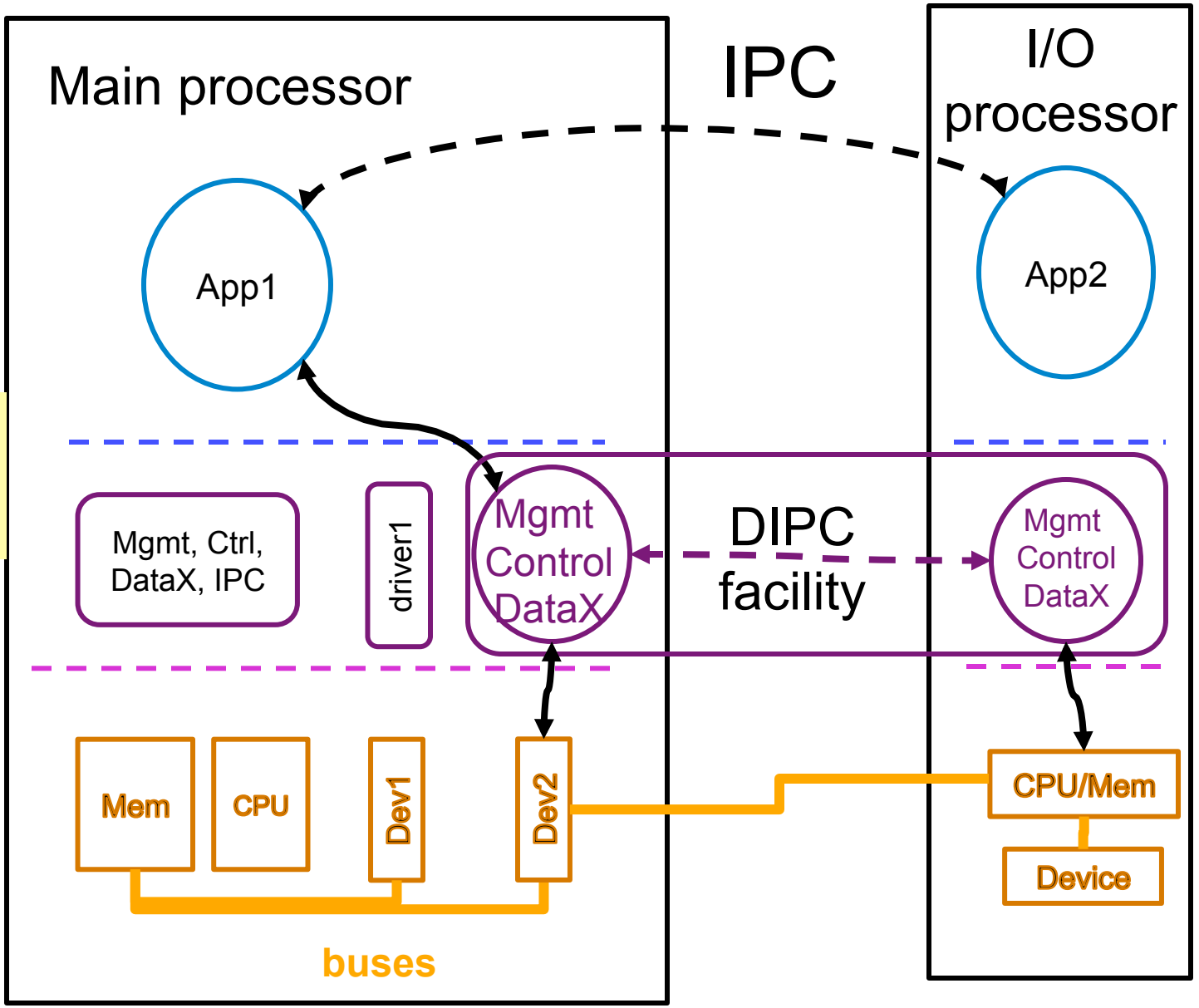


layers are naturally recursive



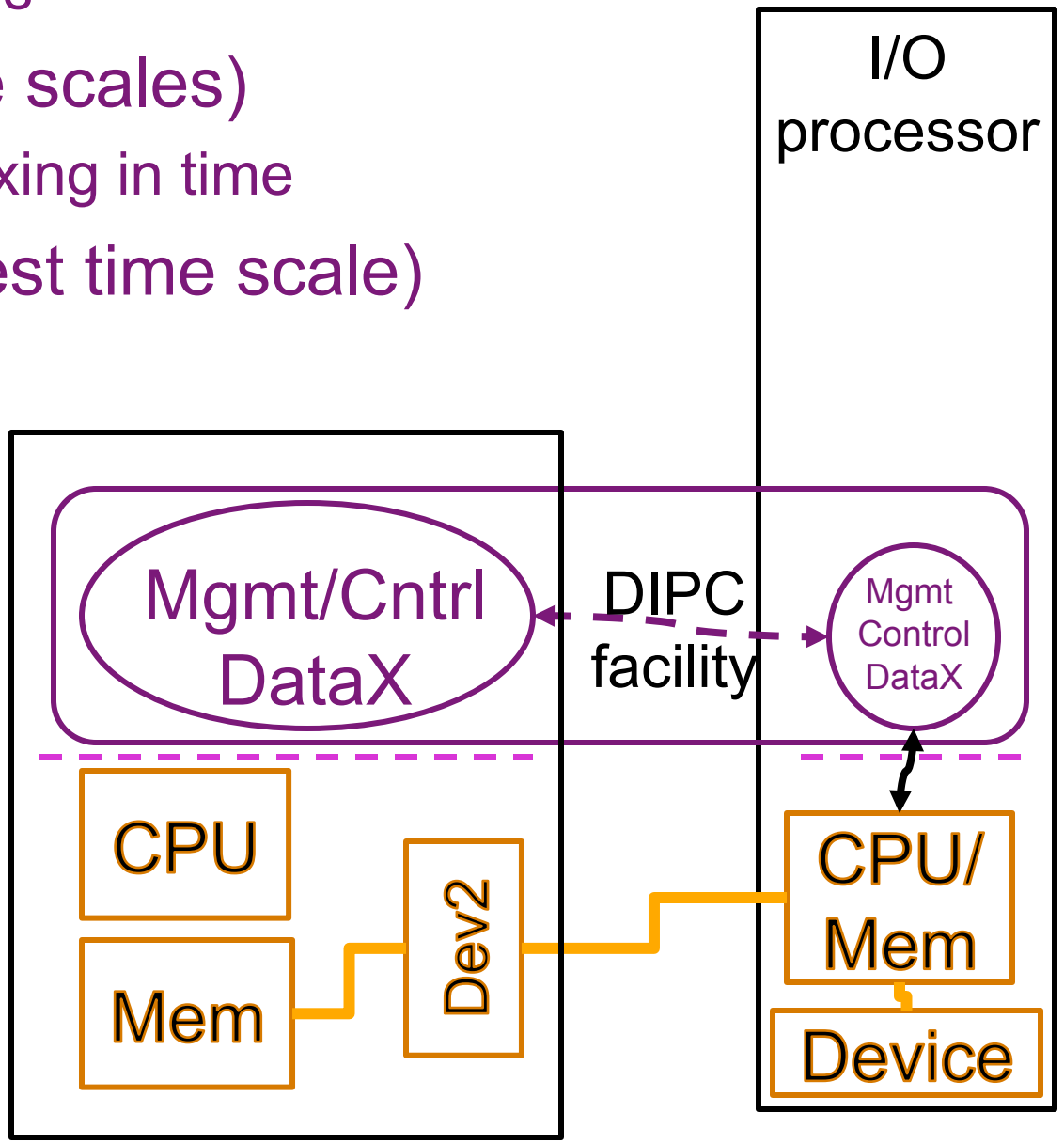
What happens in a computer ***system***?

Distributed IPC.



- Data transfer (fastest time scale)
  - Between “processors”
- Control (middle time scales)
  - Scheduling/Multiplexing in time
- Management (slowest time scale)
  - What? Where?

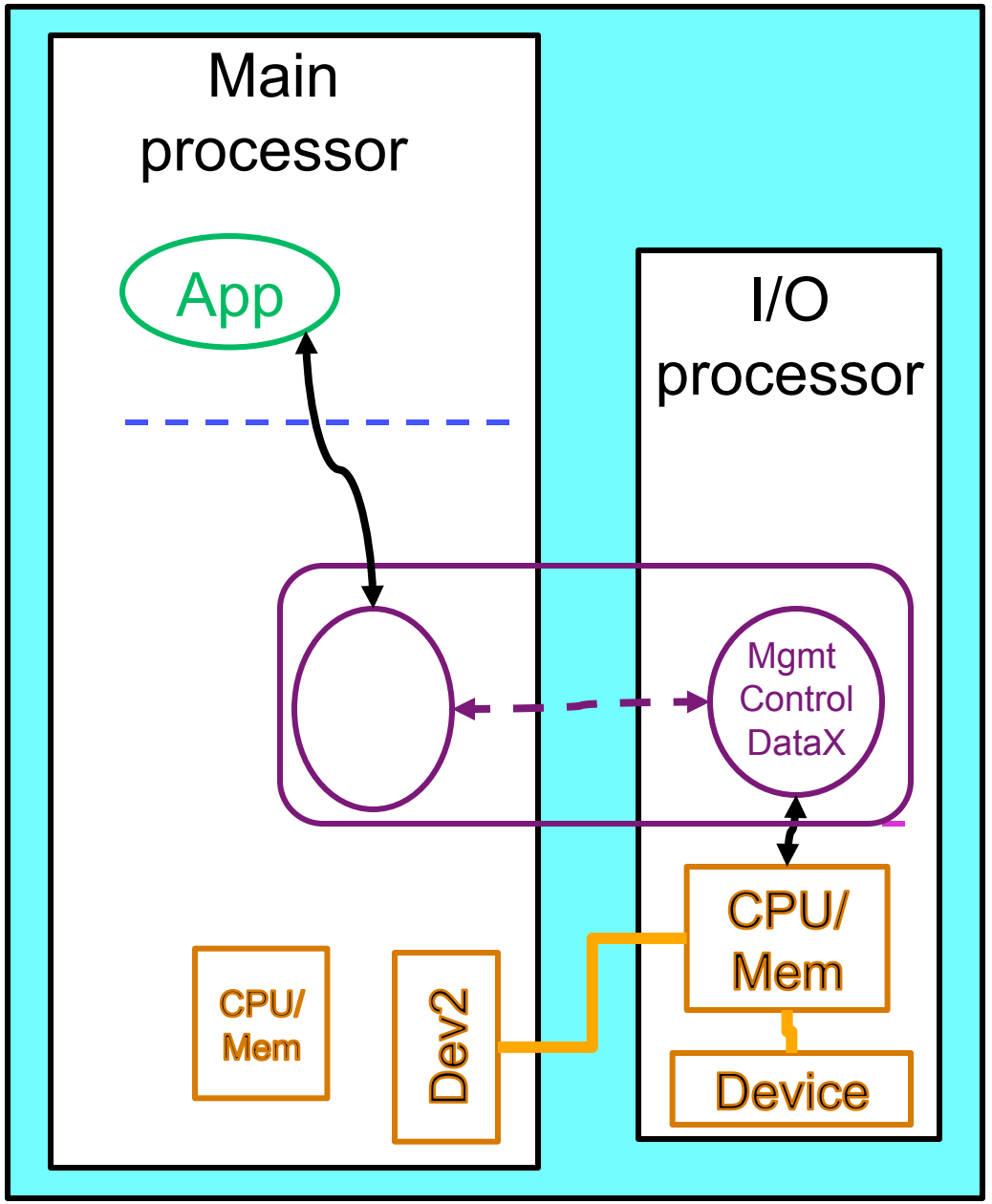
Mgmt and Ctrl become more complex



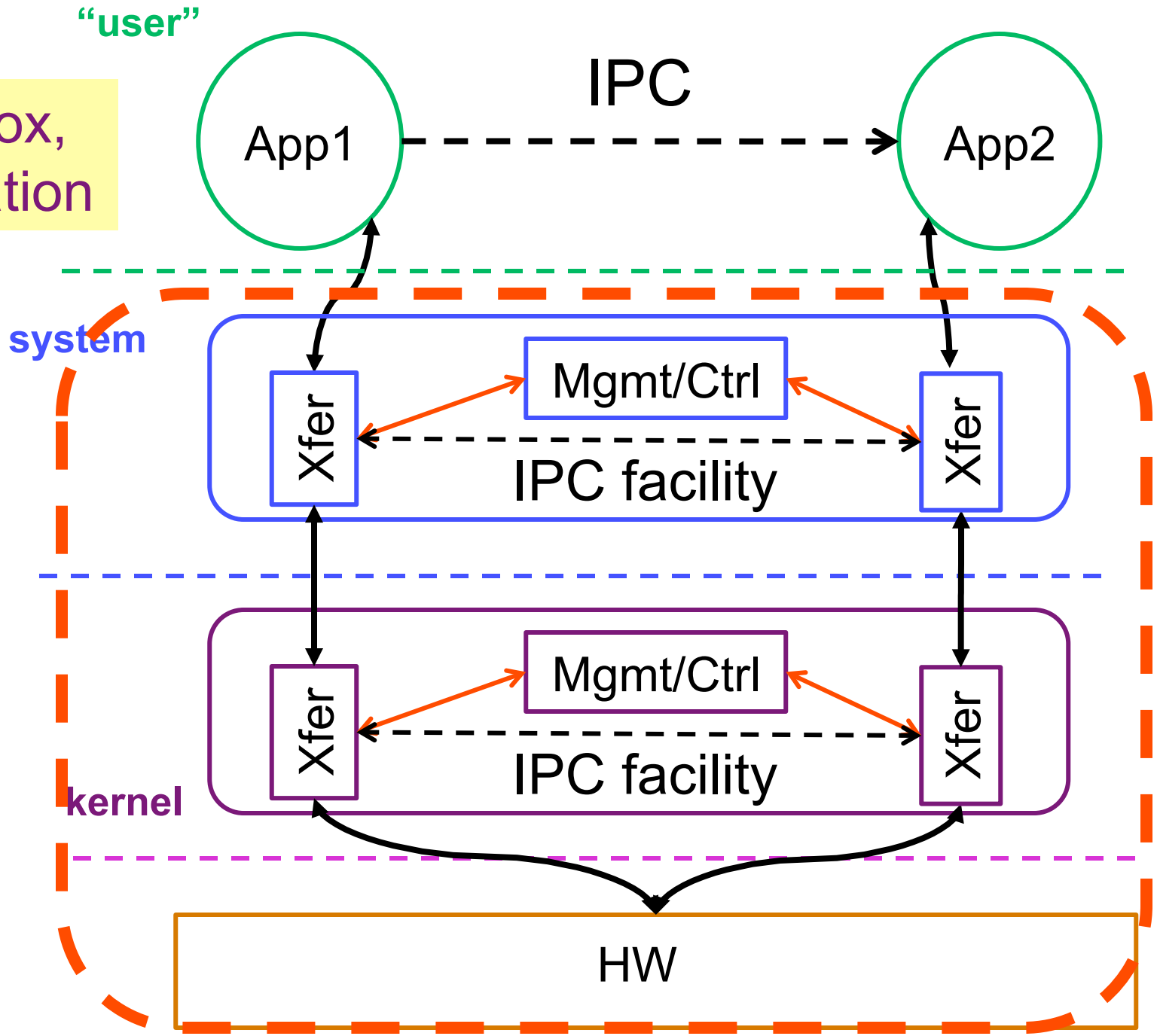
Any layer's functions are

- Data transfer (fastest time scale)
  - Within/between memory, computing, devices
  - Between processors
  - Between virtualized resources (in higher layers)
- Control (middle time scales)
  - Scheduling/Multiplexing resources in time/space
  - Real and virtualized
- Management (slowest time scale)
  - ***What*** resources are available?
  - ***Where*** are they?
  - Real and virtualized

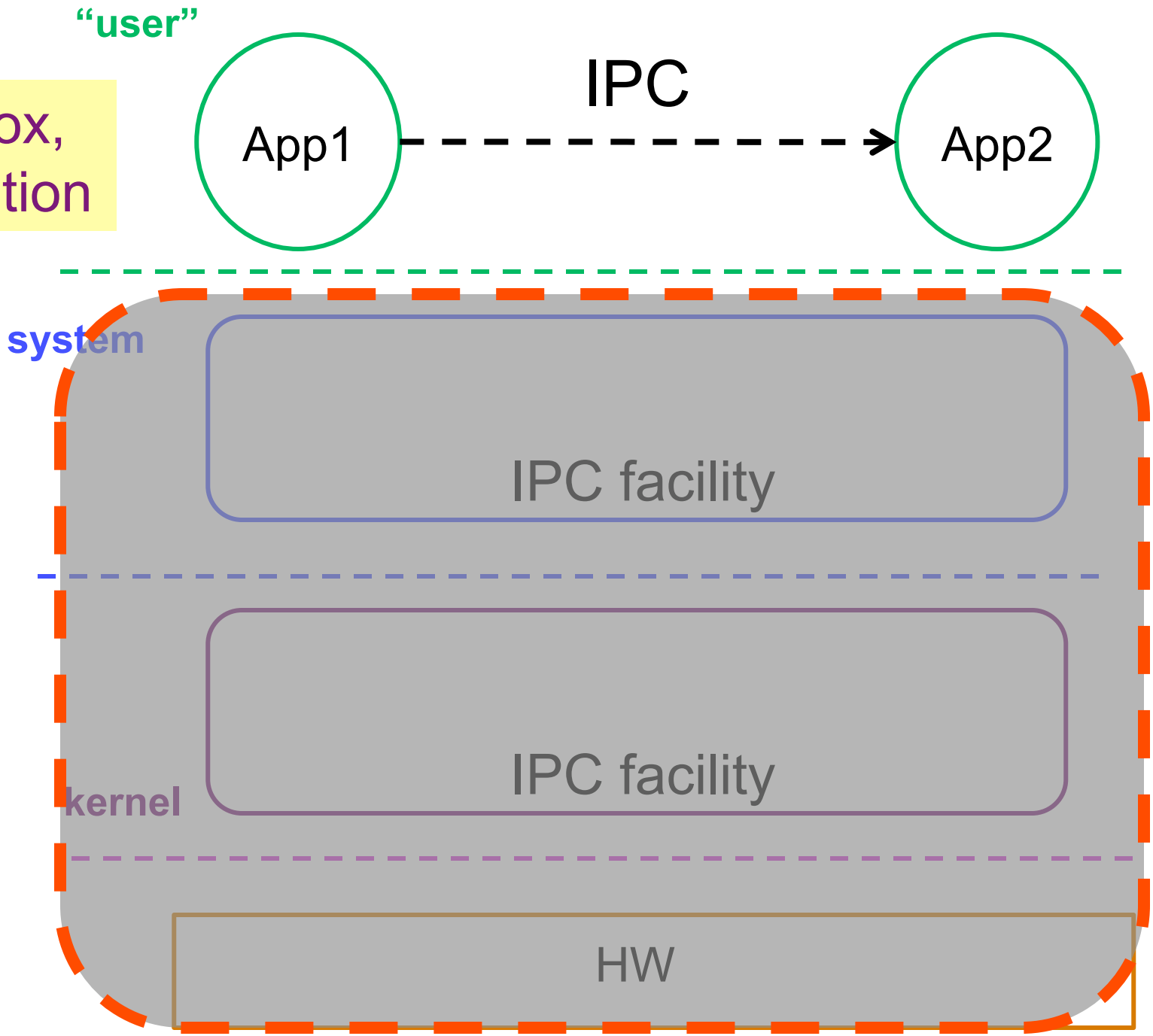
Might be all in the same “box”.



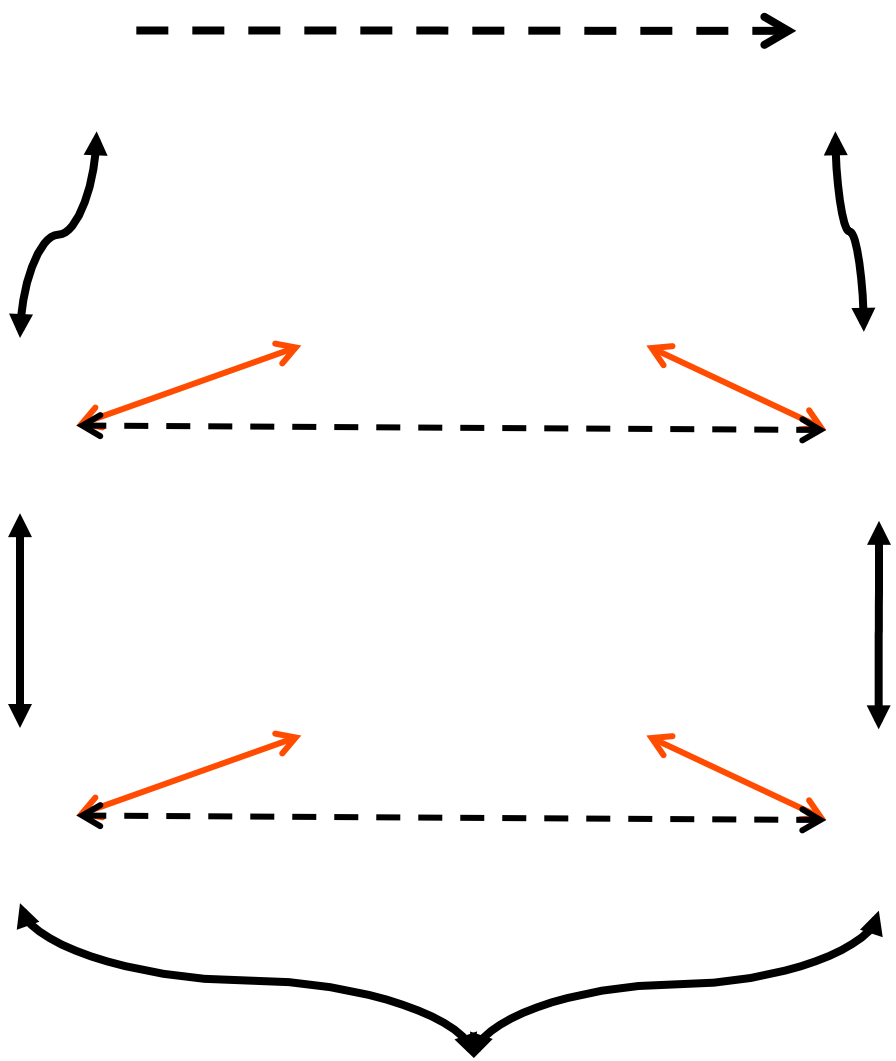
Black box, virtualization



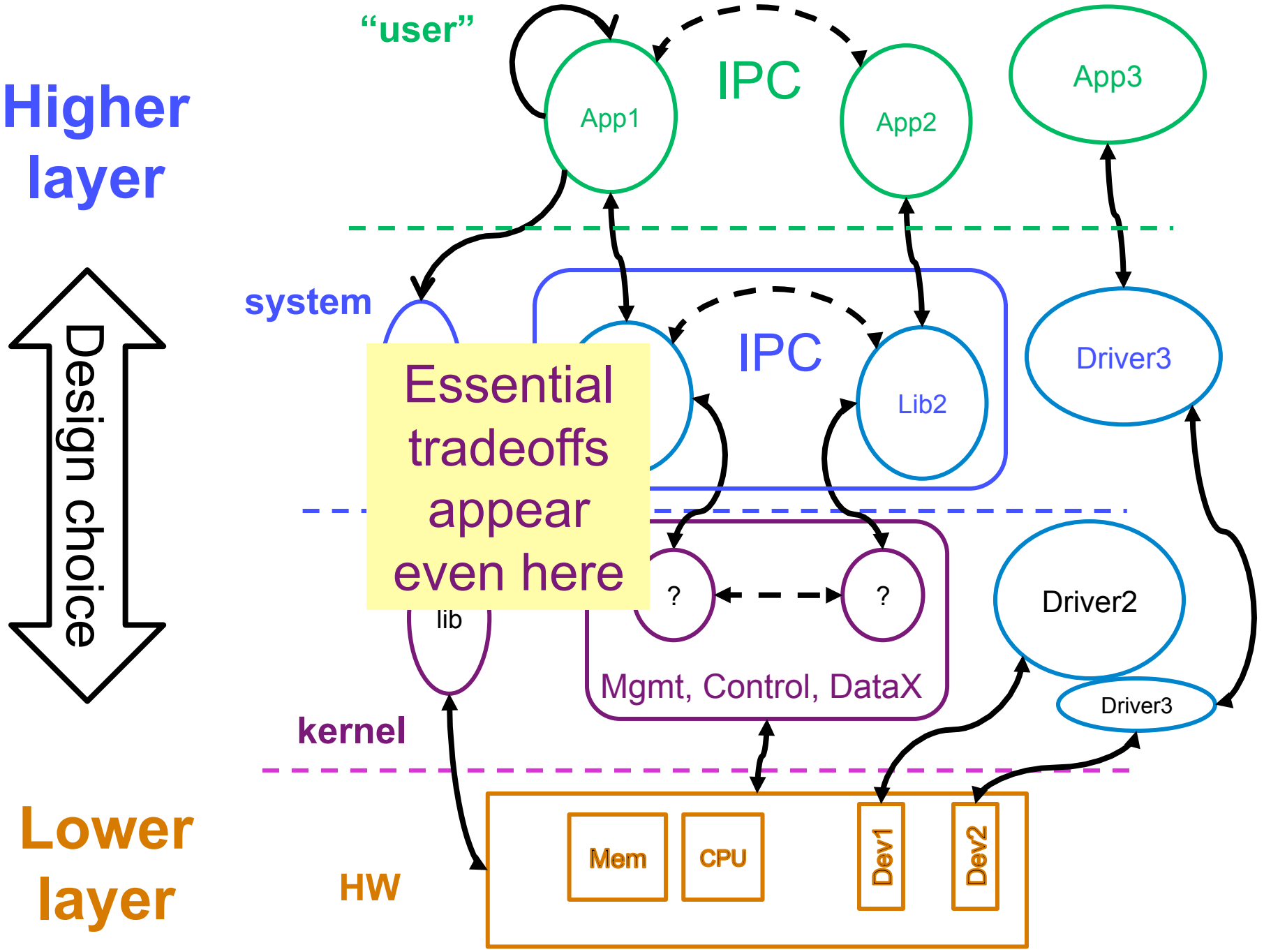
Black box, virtualization

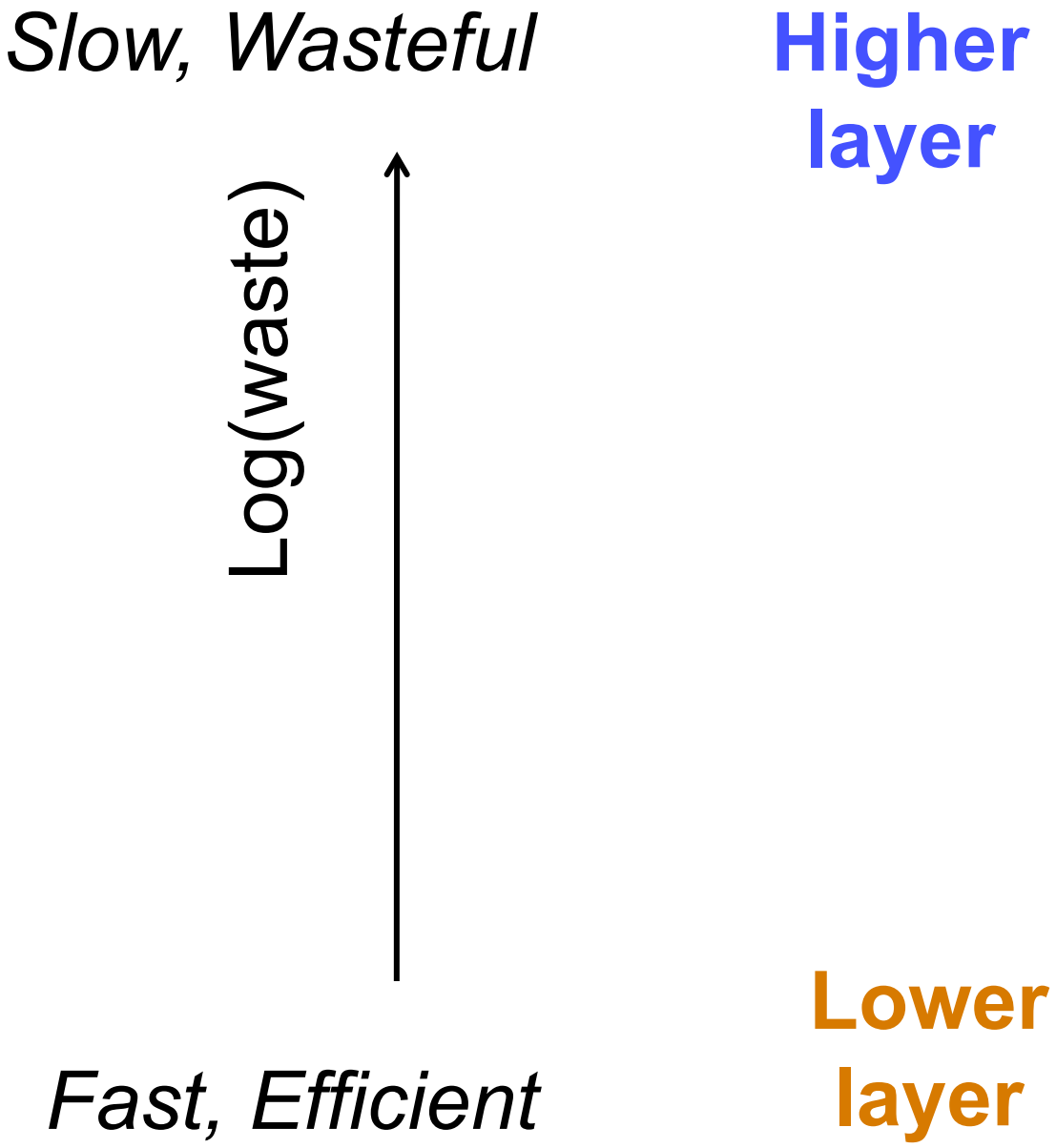


All these signals are “virtual”

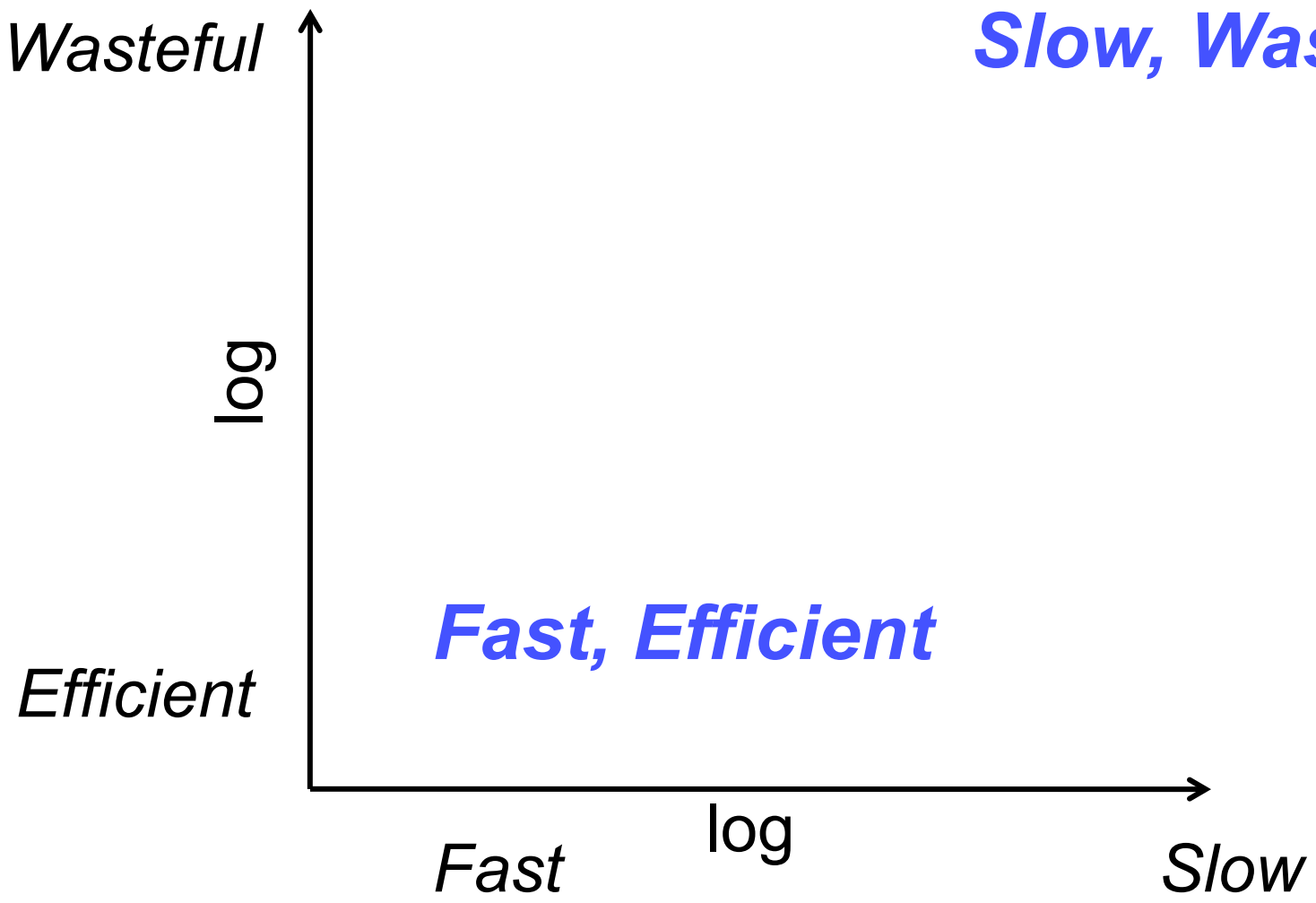


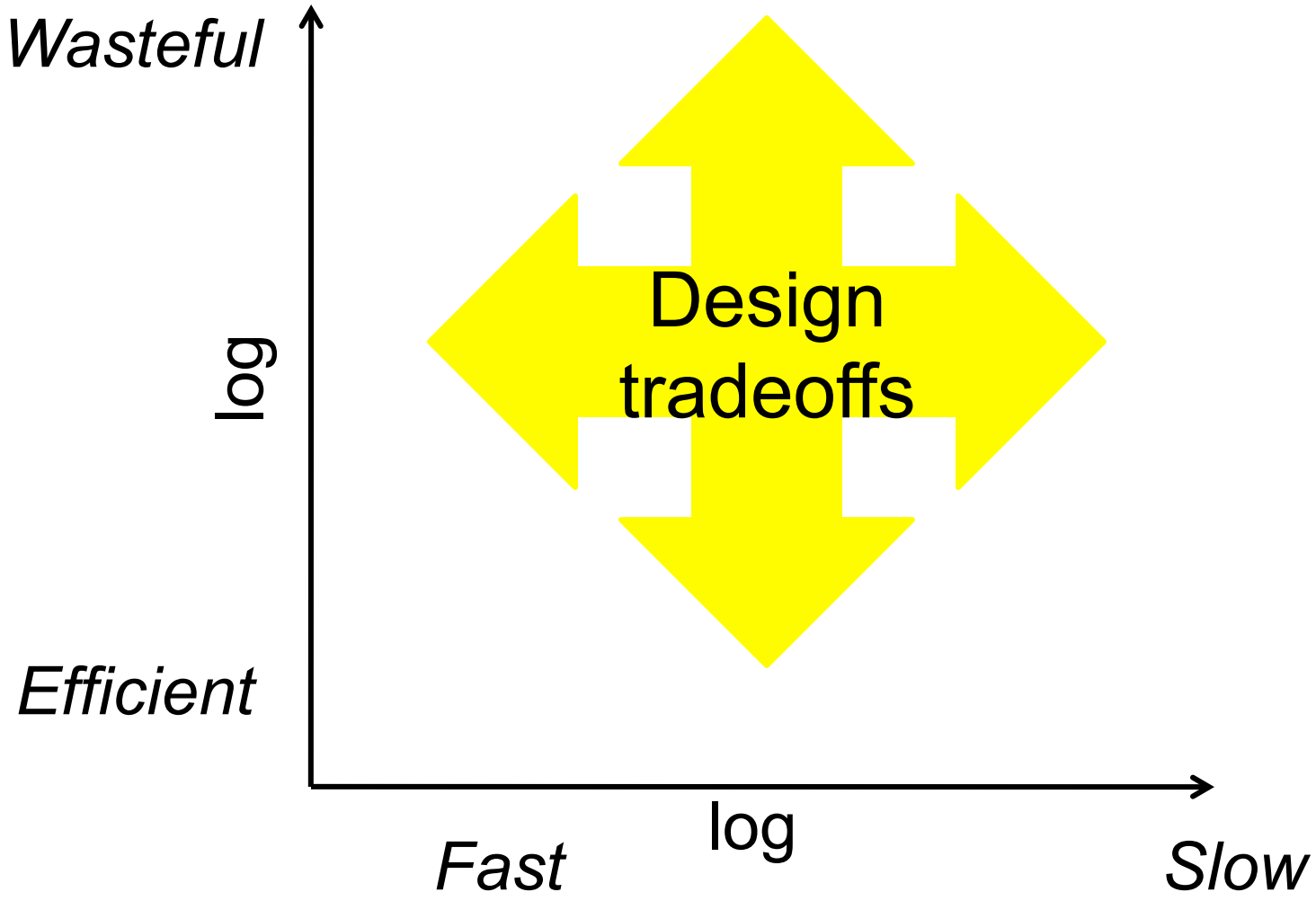
The only “real” signals are not shown


Higher layer
Design choice
Lower layer
"user"
App1
IPC
App2
App3
system
IPC
Lib2
Driver3
Essential tradeoffs appear even here
lib
?
?
Mgmt, Control, DataX
Driver2
Driver3
kernel
HW
Mem
CPU
Dev1
Dev2

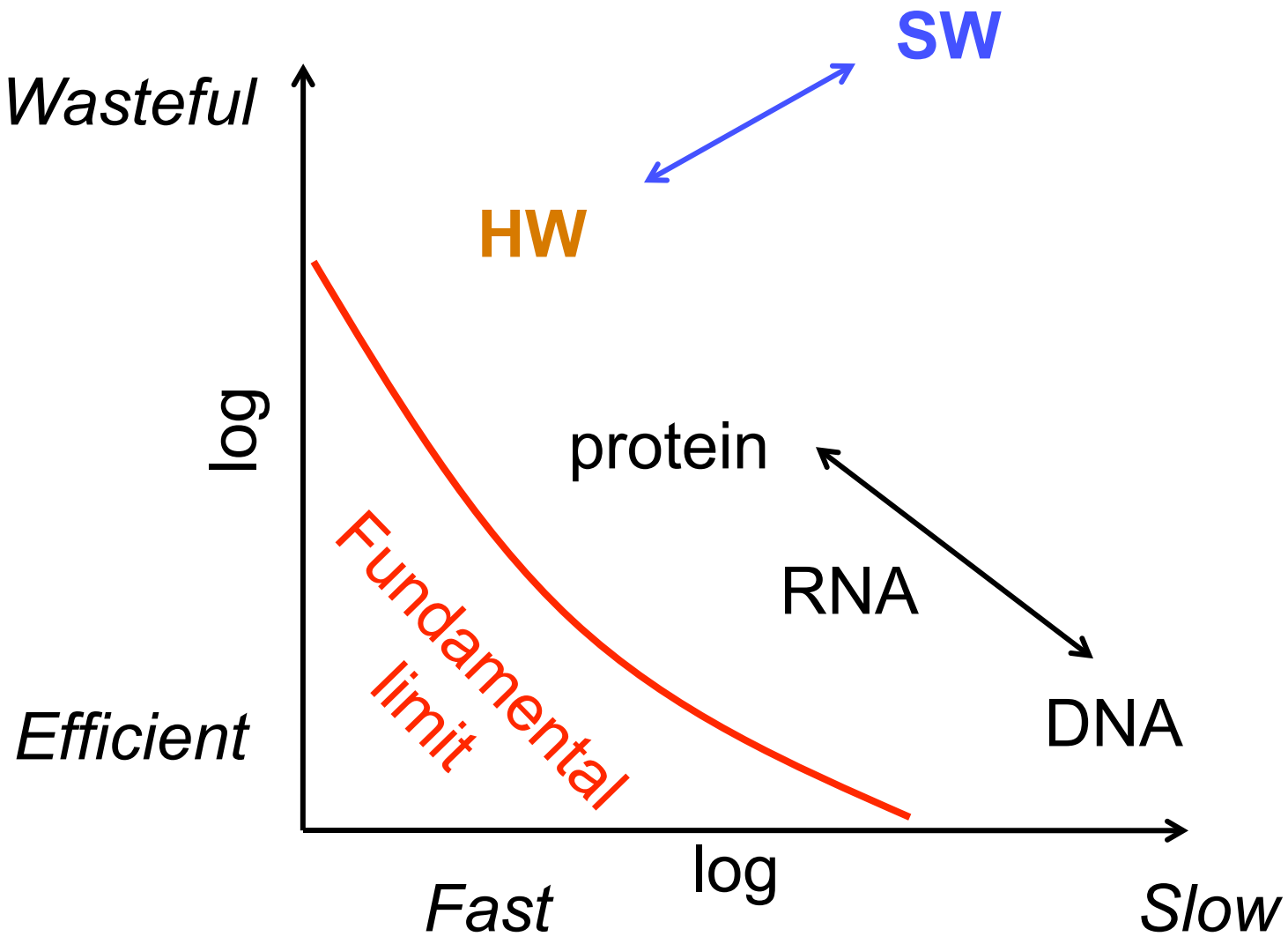

Slow, Wasteful
Higher layer
Log(waste)
Lower layer
Fast, Efficient

Expand dimensions

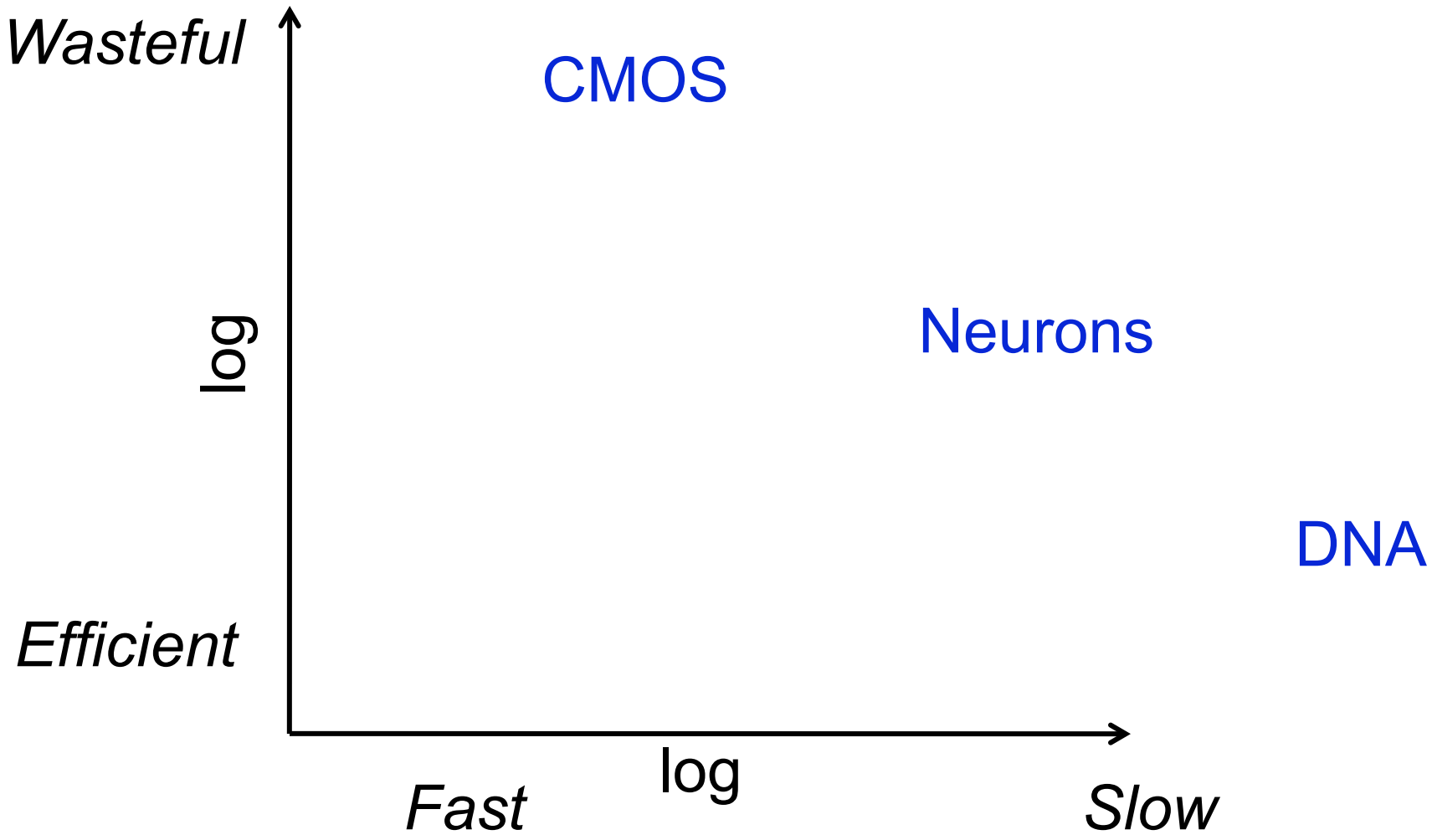



Wasteful
log
Efficient
Design
tradeoffs
Fast
log
Slow

Tradeoffs are universal,
but the details are not.

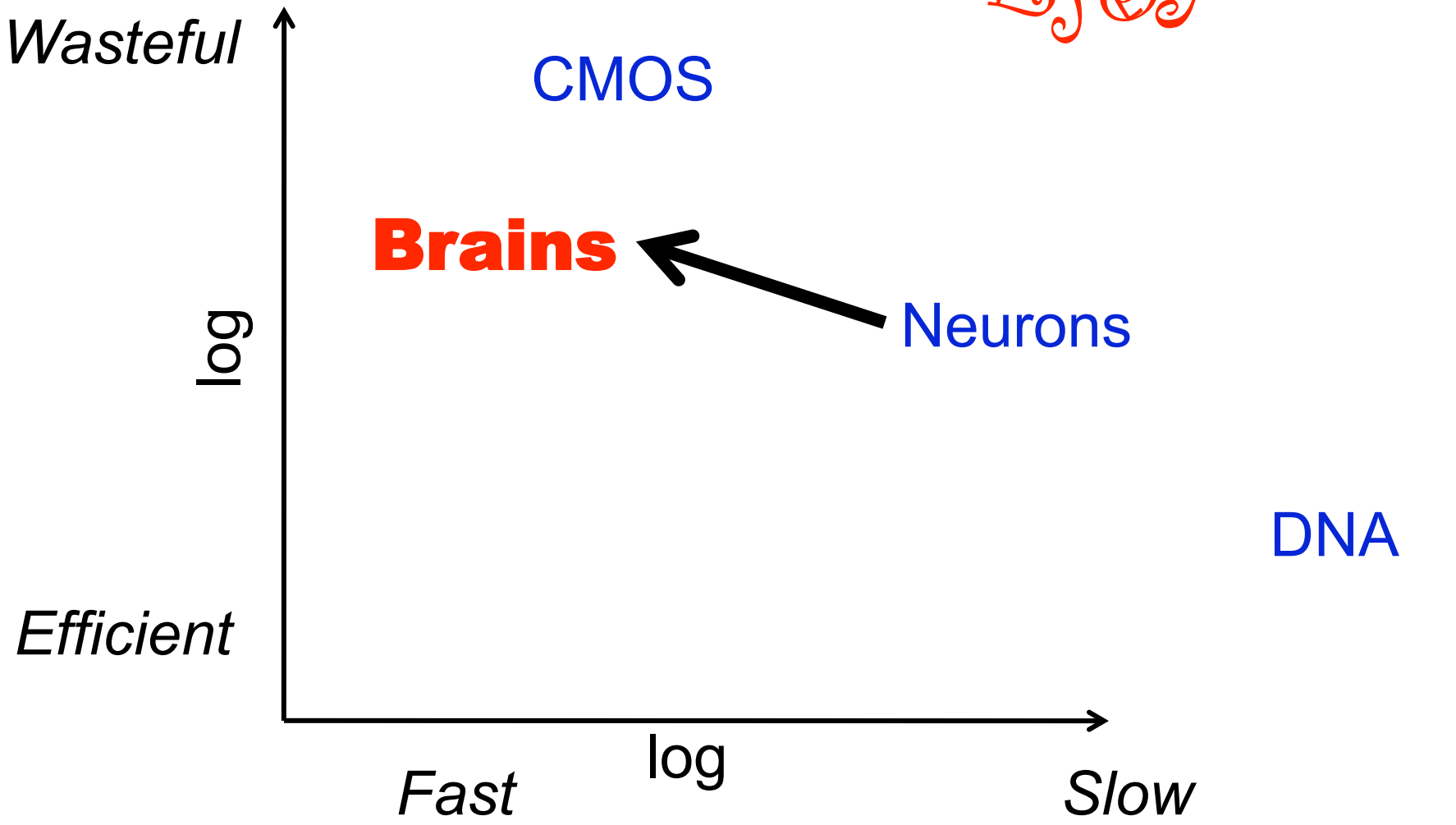


Computational hardware substrates

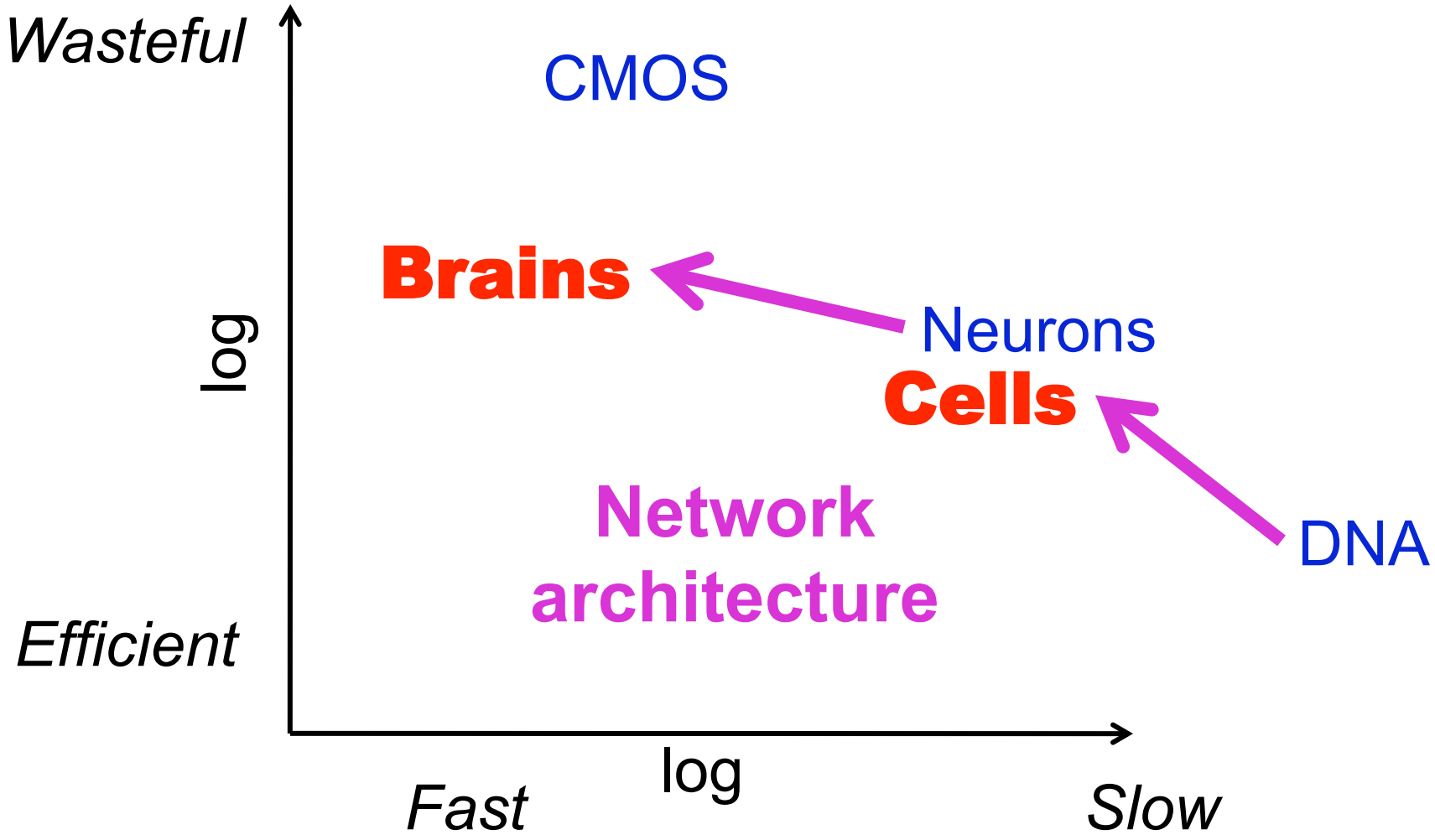


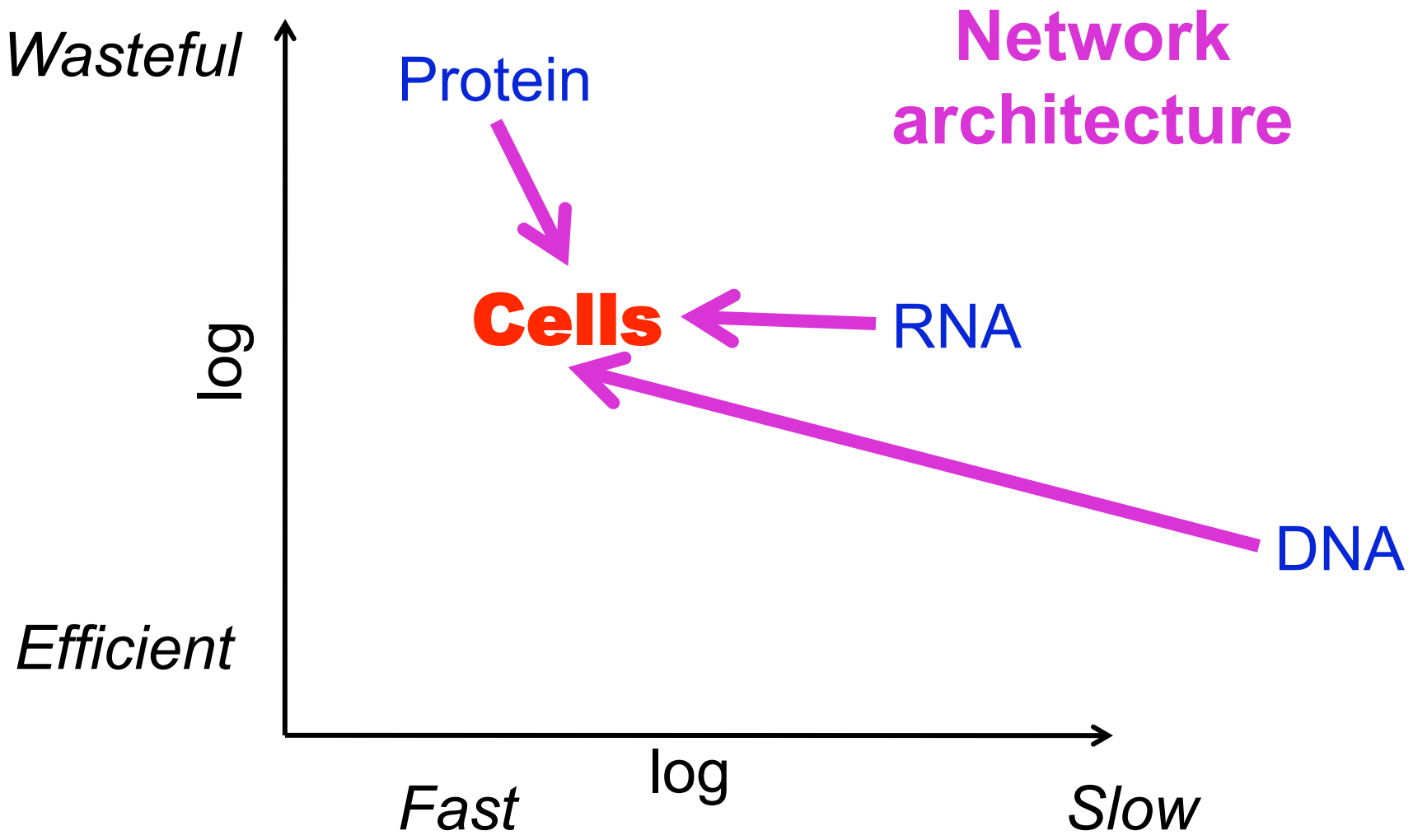
Some tasks: HARD for computers

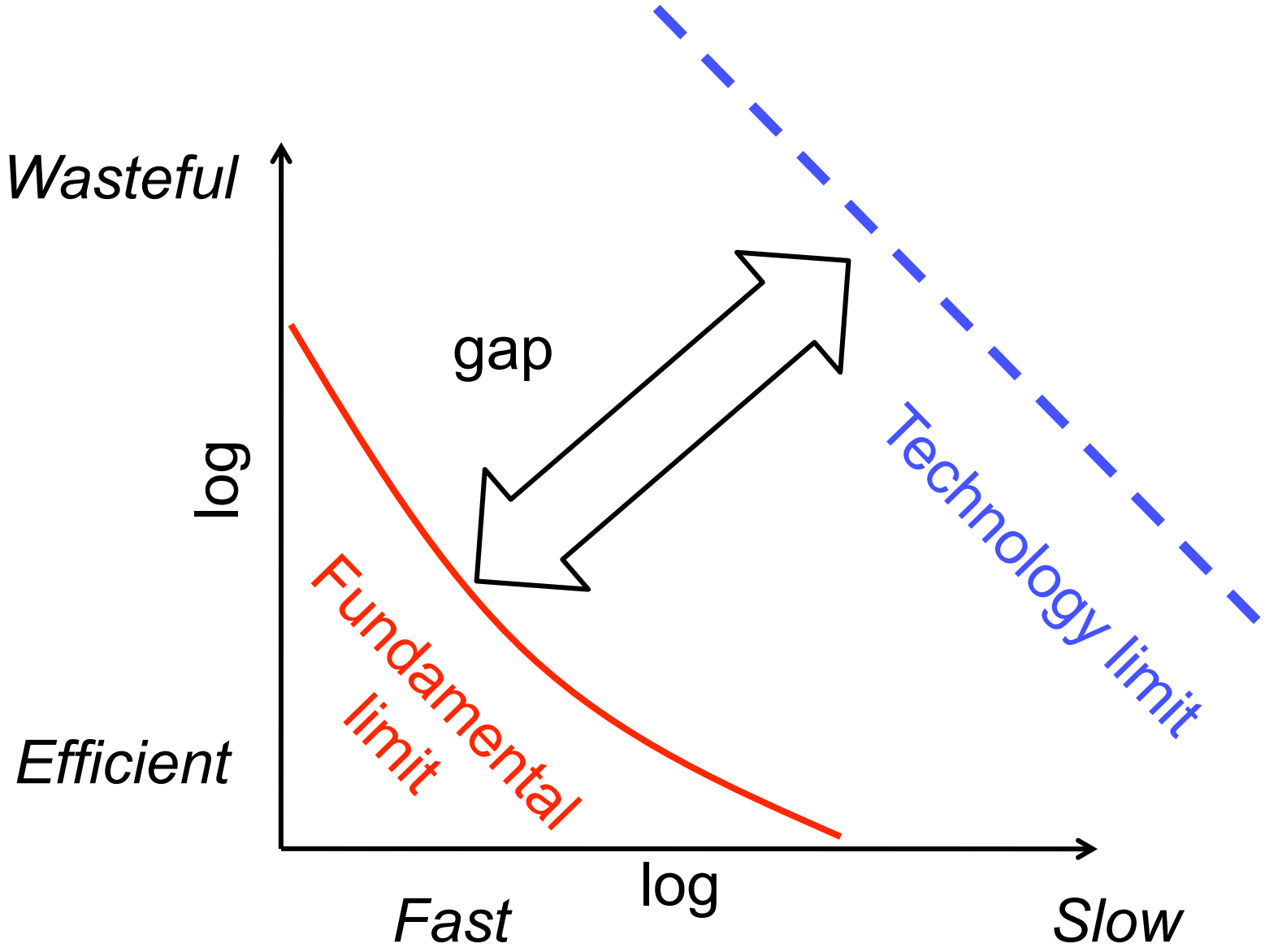
EASY for us

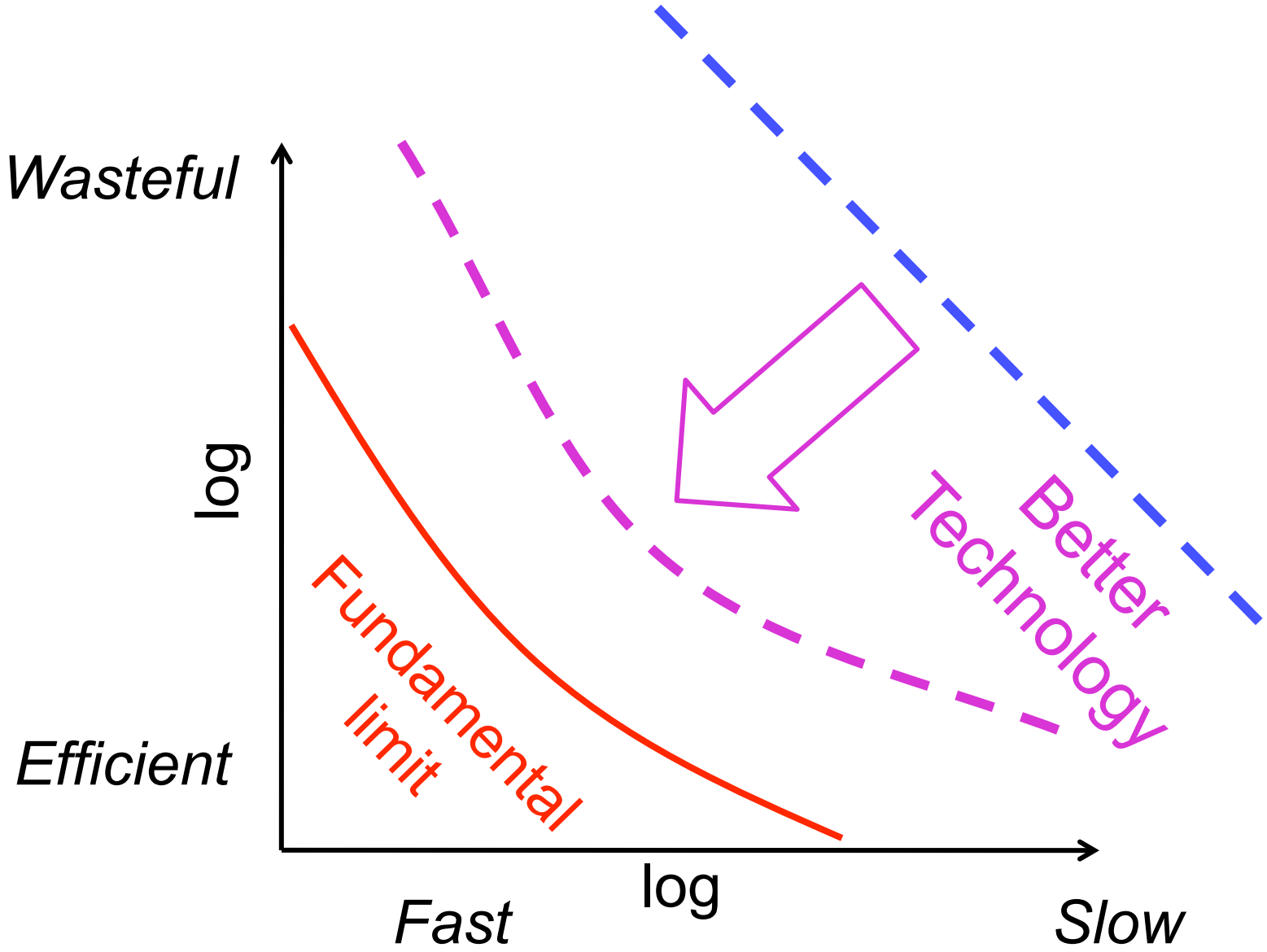


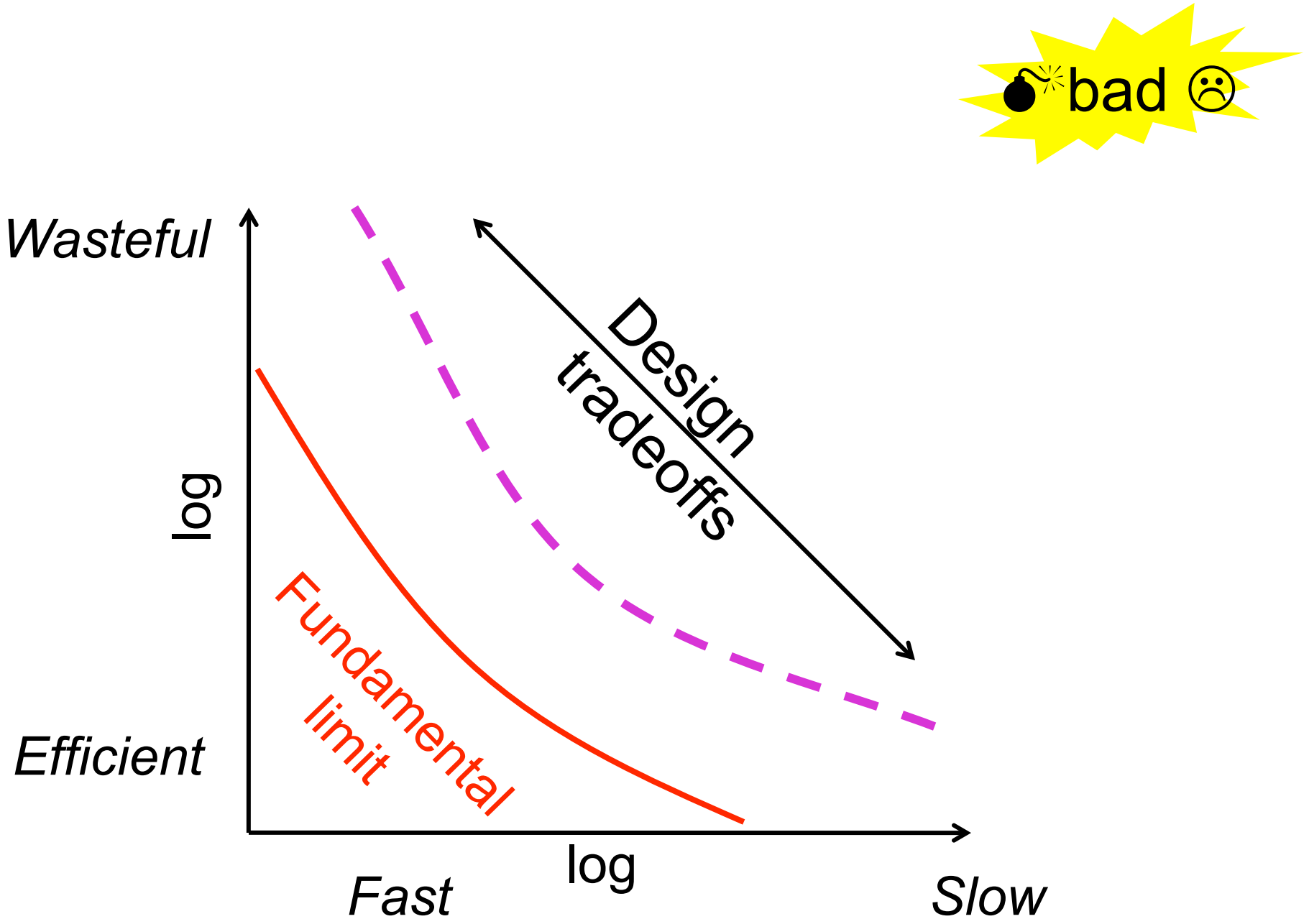
What makes this possible?




Wasteful
Efficient
log
Protein
Network
architecture
Cells
RNA
DNA
Fast
log
Slow

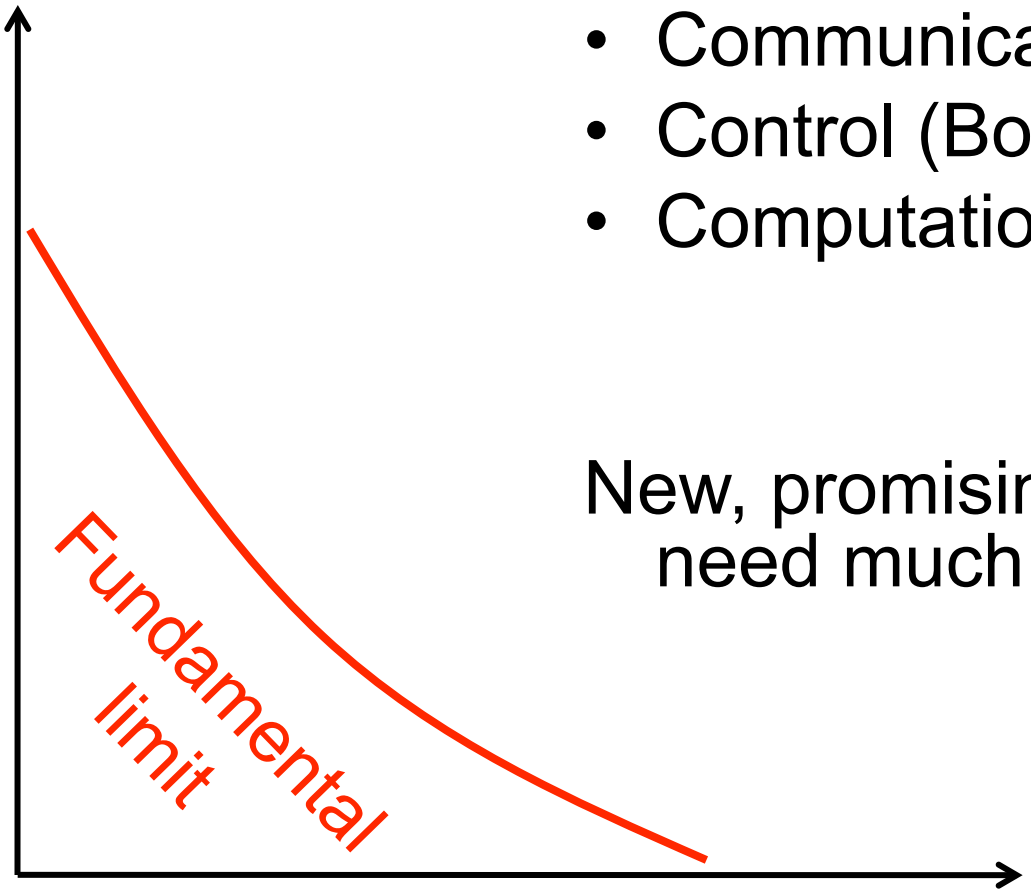

Wasteful
log
Efficient
gap
Technology limit
Fundamental limit
Fast
log
Slow


Wasteful
log
Efficient
Fundamental
limit
Better
Technology
log
Fast
Slow


bad
Wasteful
log
Efficient
Design
tradeoffs
Fundamental
limit
Fast
log
Slow

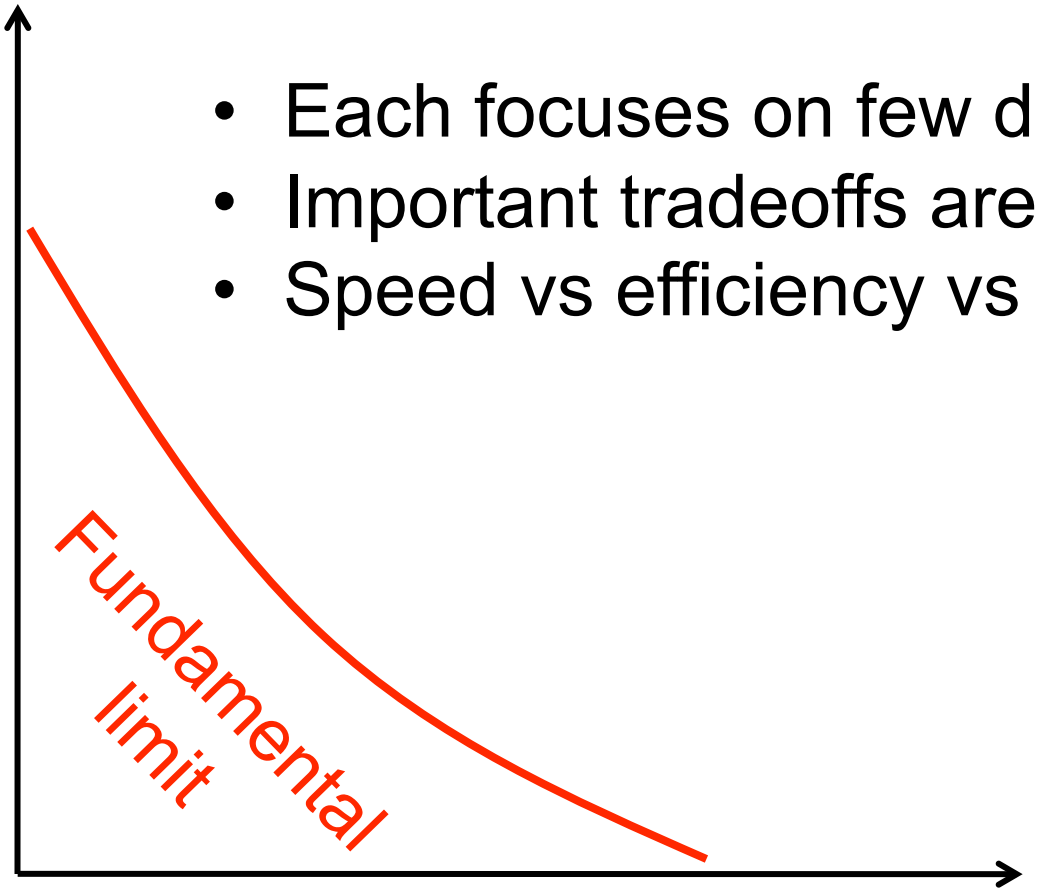
Existing hard limits have restrictive assumptions and few dimensions

- Thermodynamics (Carnot)
- Communications (Shannon)
- Control (Bode)
- Computation (Turing)

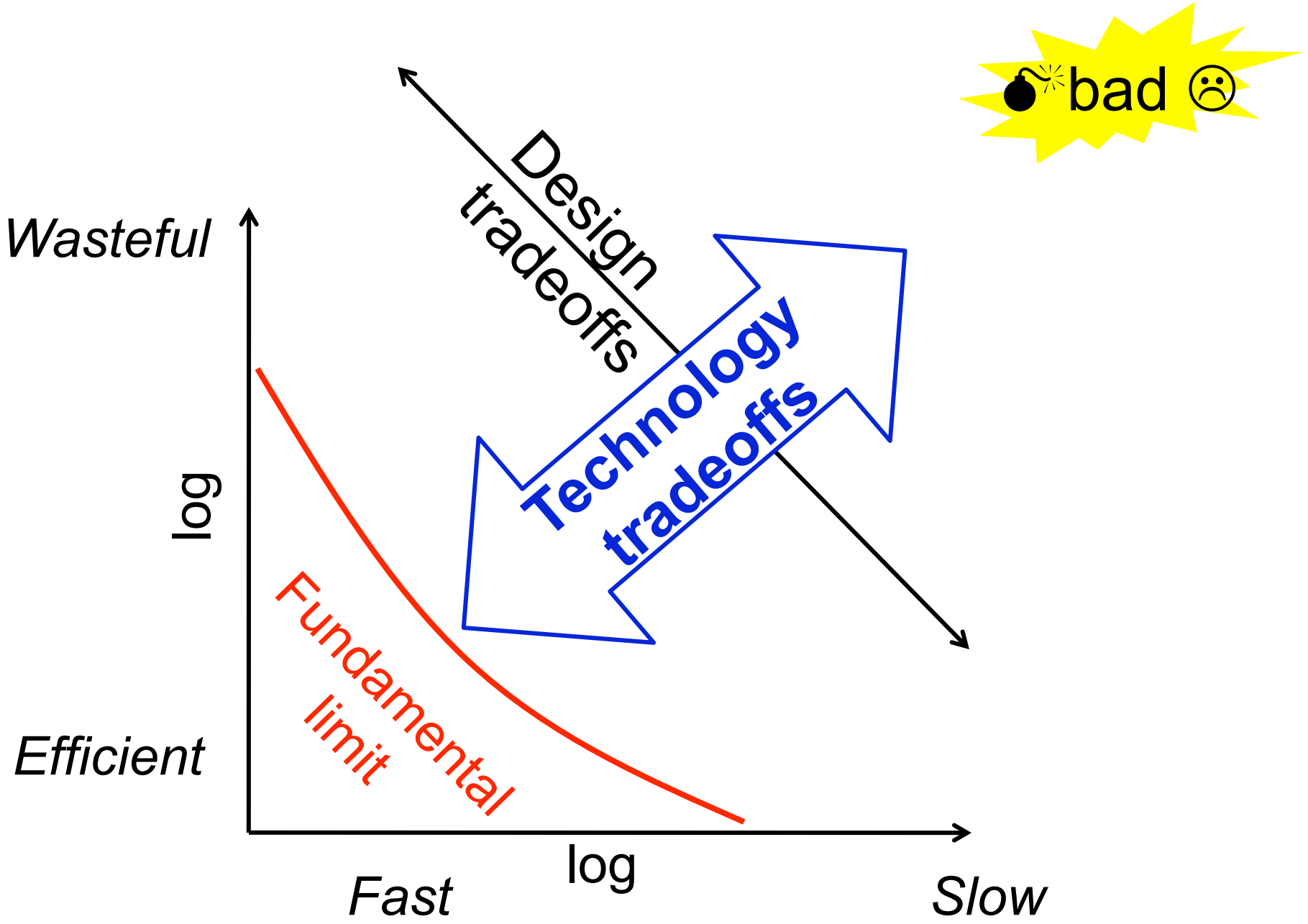


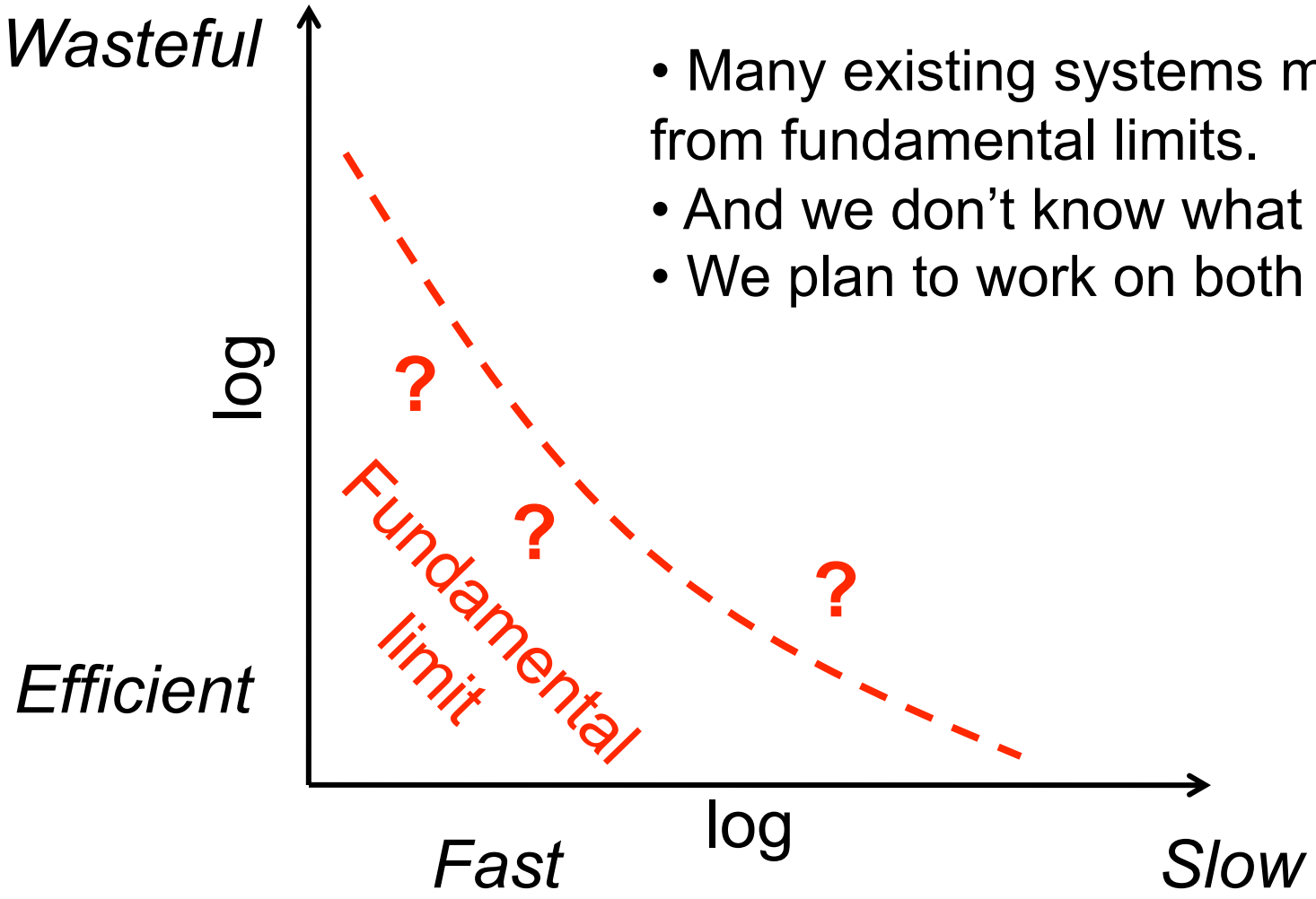
New, promising unifications but need much more

- Thermodynamics (Carnot)
- Communications (Shannon)
- Control (Bode)
- Computation (Turing)



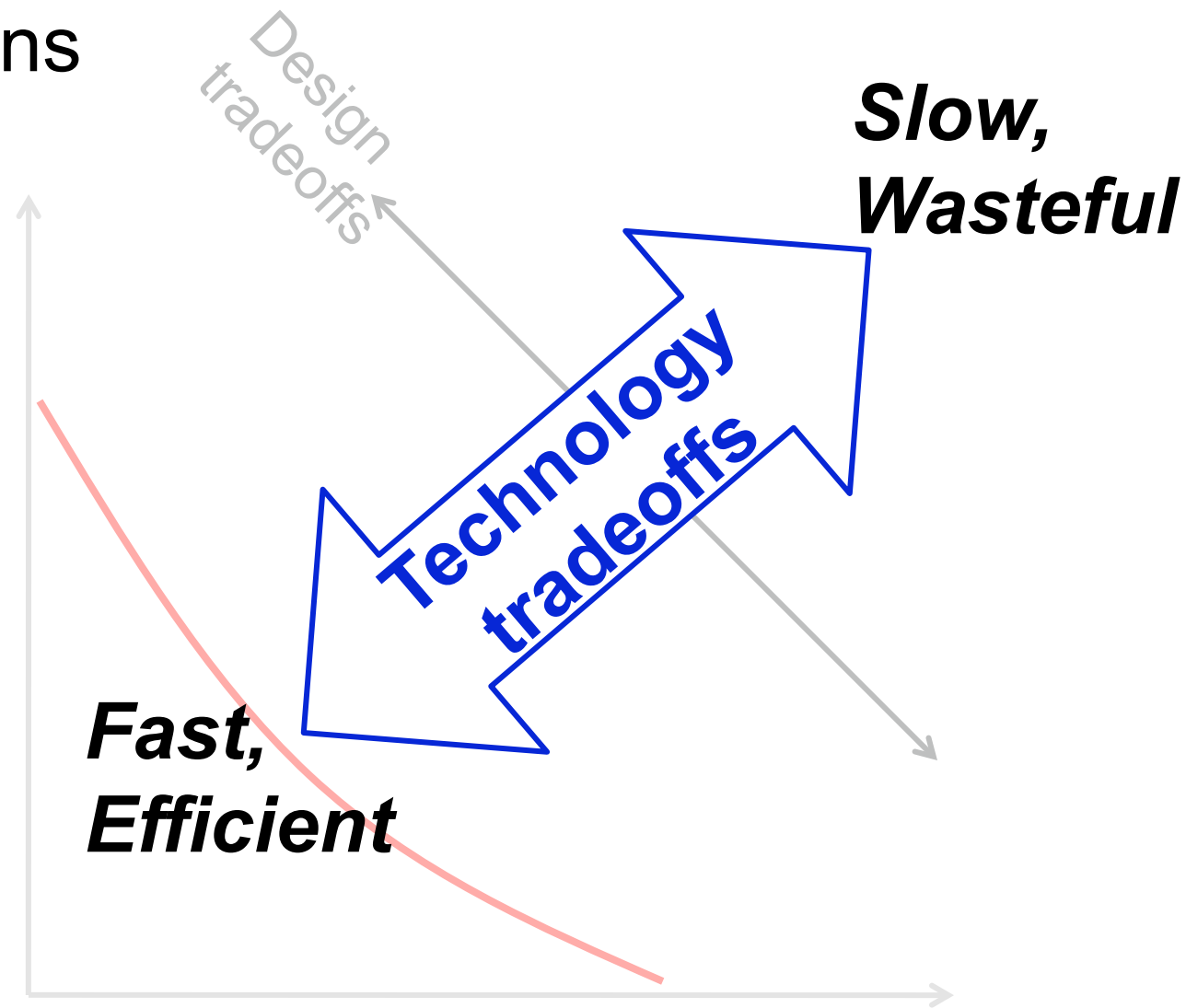
- Each focuses on few dimensions
- Important tradeoffs are across these areas
- Speed vs efficiency vs robustness vs …


bad
Design tradeoffs
Technology tradeoffs
Wasteful
log
Efficient
Fundamental limit
log
Fast
Slow

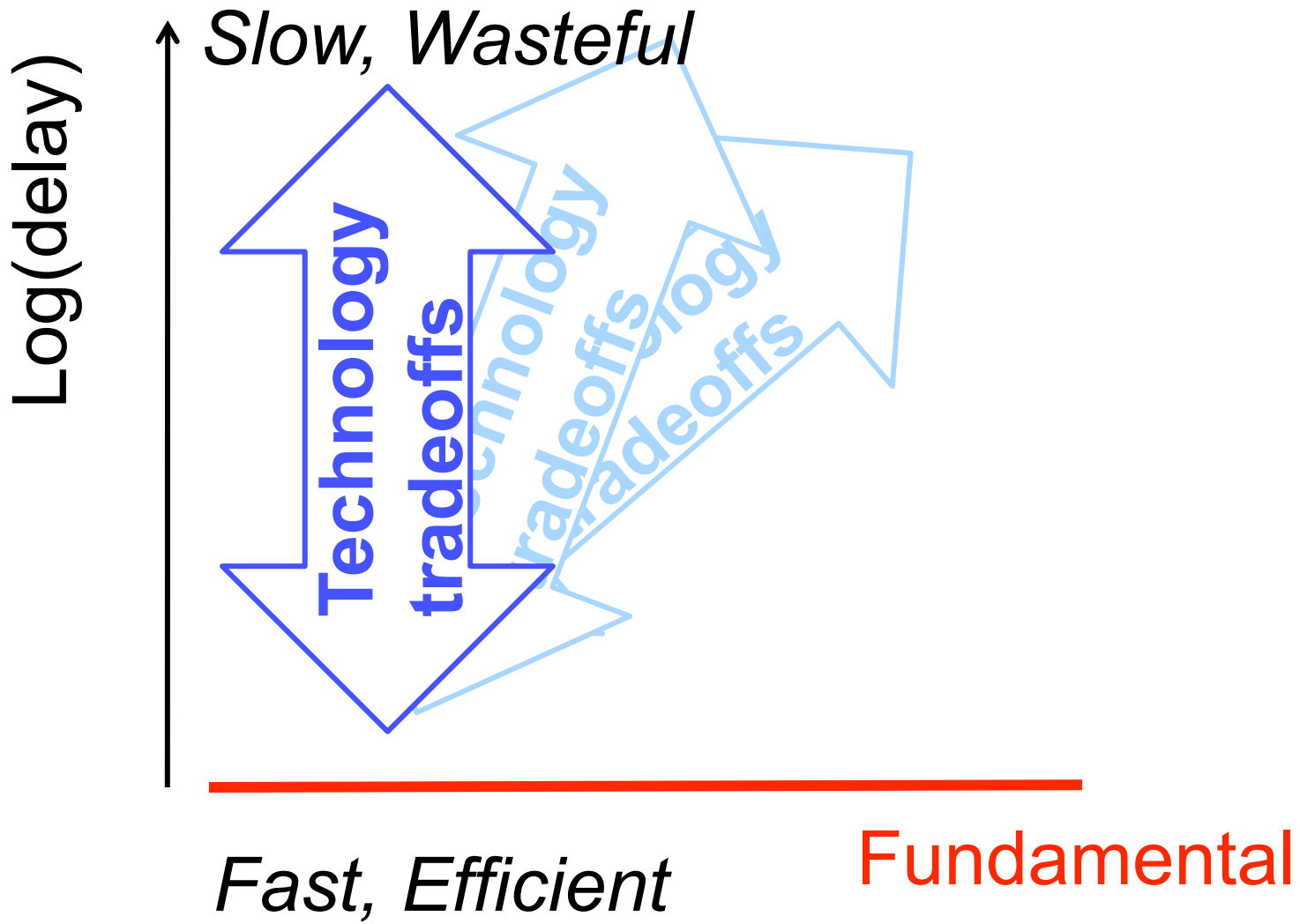


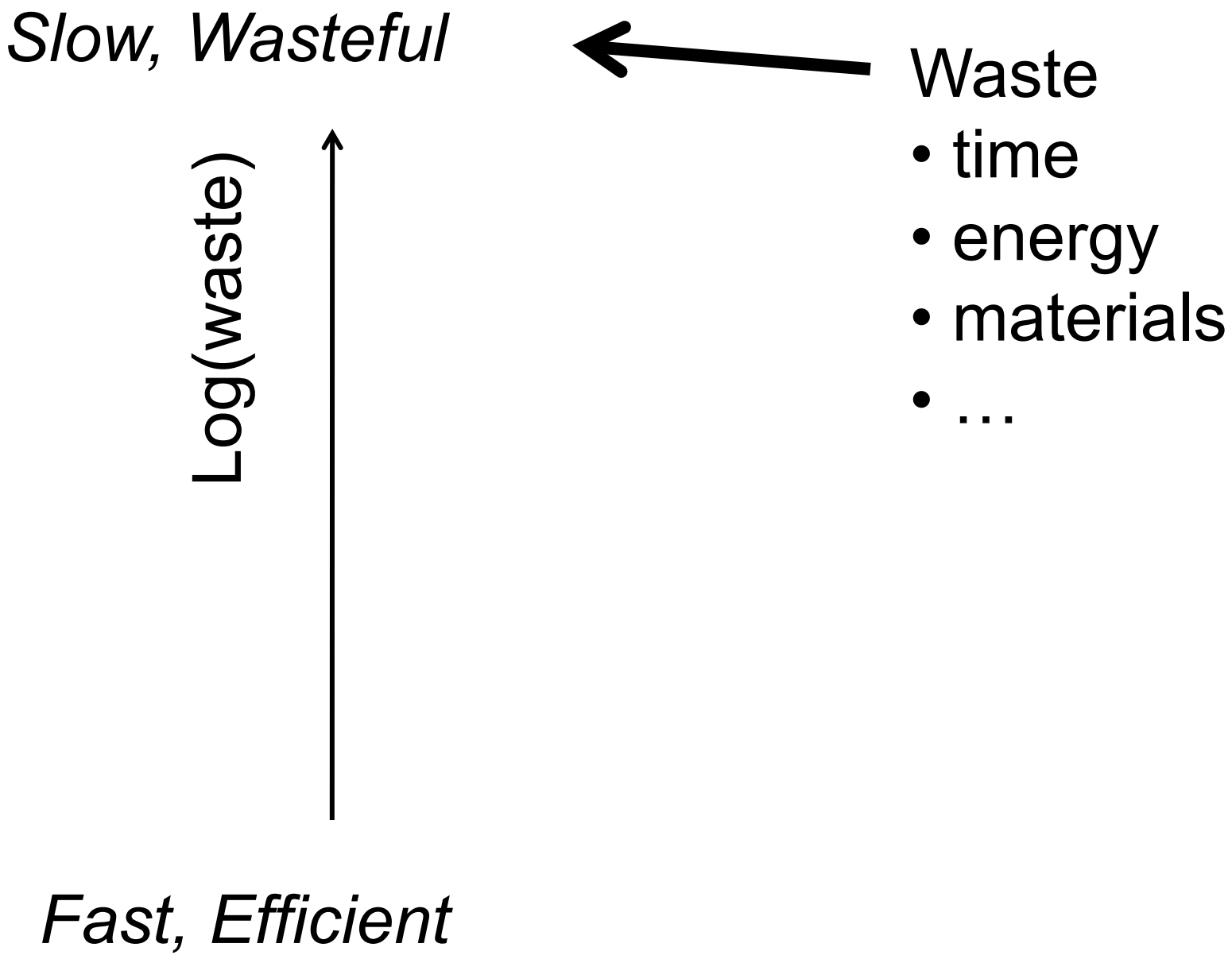
- Many existing systems may be far from fundamental limits.
- And we don't know what they are..
- We plan to work on both ends

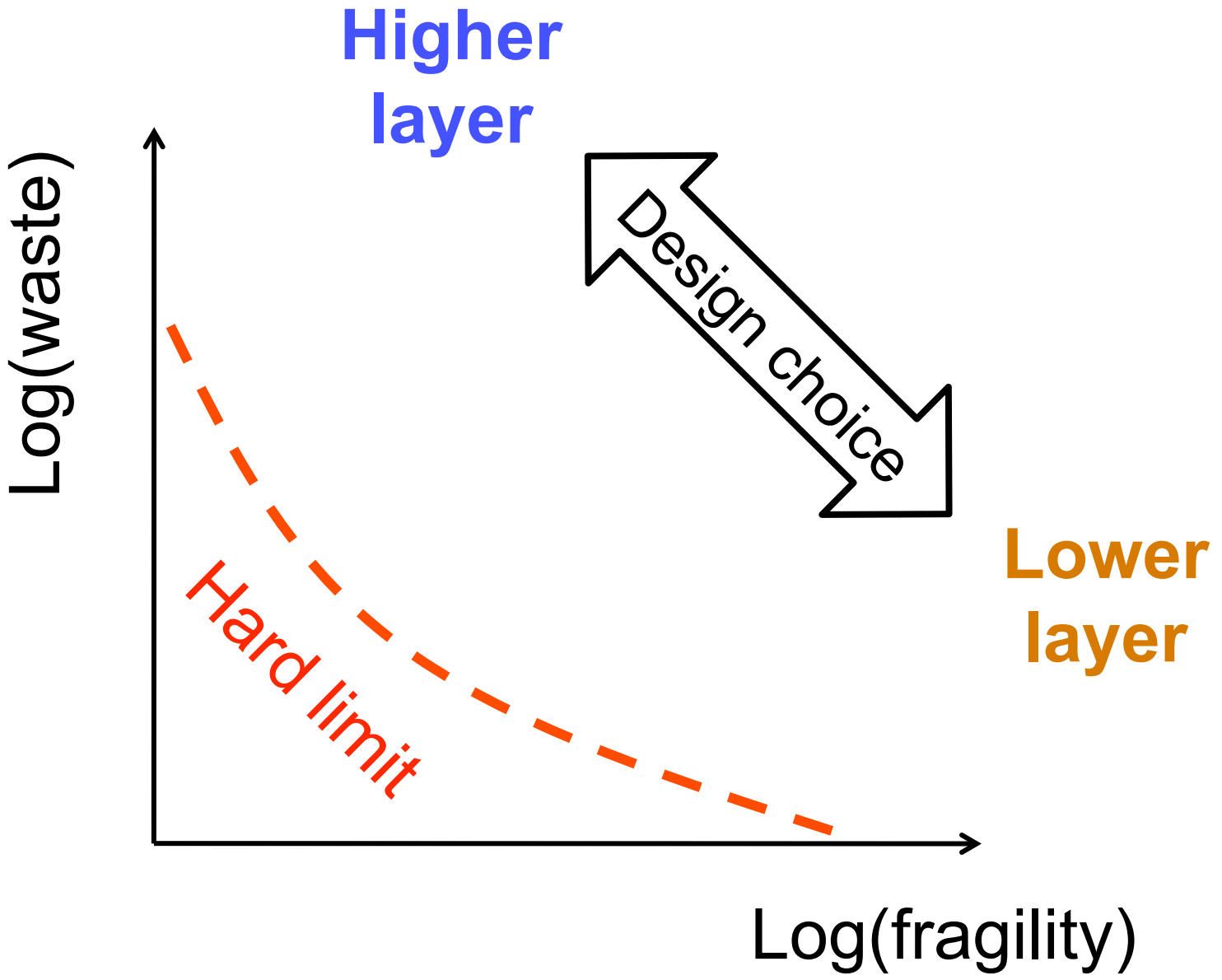
Collapse dimensions



Collapse dimensions




Slow, Wasteful
Log(waste)
Fast, Efficient
Waste
• time
• energy
• materials
• ...


Higher layer
Design choice
Lower layer
Hard limit
Log(waste)
Log(fragility)

[**a system**]
can be *robust*
for a given
[**property**]
and a set of
[**perturbations**]

Yet
be ***fragile for***
a different
[**property**]
or
[**perturbation**]

Log(robustness)

Log(fragility)

Question: Human complexity

Robust

- ☺ Efficient, flexible metabolism
- ☺ Regeneration & renewal
- ☺ Rich microbial symbionts
- ☺ Immune systems
- Complex societies
- Advanced technologies

Yet Fragile

- ☹ Obesity and diabetes
- ☹ Cancer
- ☹ Parasites, infection
- ☹ Inflammation, Auto-Im.
- Epidemics, war, …
- Catastrophic failures

Mechanism?

Robust

- ☺ Efficient, flexible metabolism
- ☺ Regeneration & renewal
  - ☺ Fat accumulation
  - ☺ Insulin resistance
  - ☺ Inflammation

Yet Fragile

- ☹ Obesity and diabetes
- ☹ Cancer
  - ☹ Fat accumulation
  - ☹ Insulin resistance
  - ☹ Inflammation

Fluctuating energy

Static energy

Implications/ Generalizations

Robust	Yet Fragile
☺ Efficient, flexible metabolism	☹ Obesity and diabetes
☺ Rich microbial symbionts	☹ Parasites, infection
☺ Immune systems	☹ Inflammation, Auto-Im.
☺ Regeneration & renewal	☹ Cancer
Complex societies	☠ Epidemics, war, …
Advanced technologies	Catastrophic failures

- Fragility = Hijacking, side effects, unintended… of mechanisms evolved for robustness
- Complexity is driven by control, robust/fragile tradeoffs
- Math: New robust/fragile conservation laws

- Resilience/safety/security Engineering/Economics: "Human error" and "human nature" is often a symptom of bad system architecture

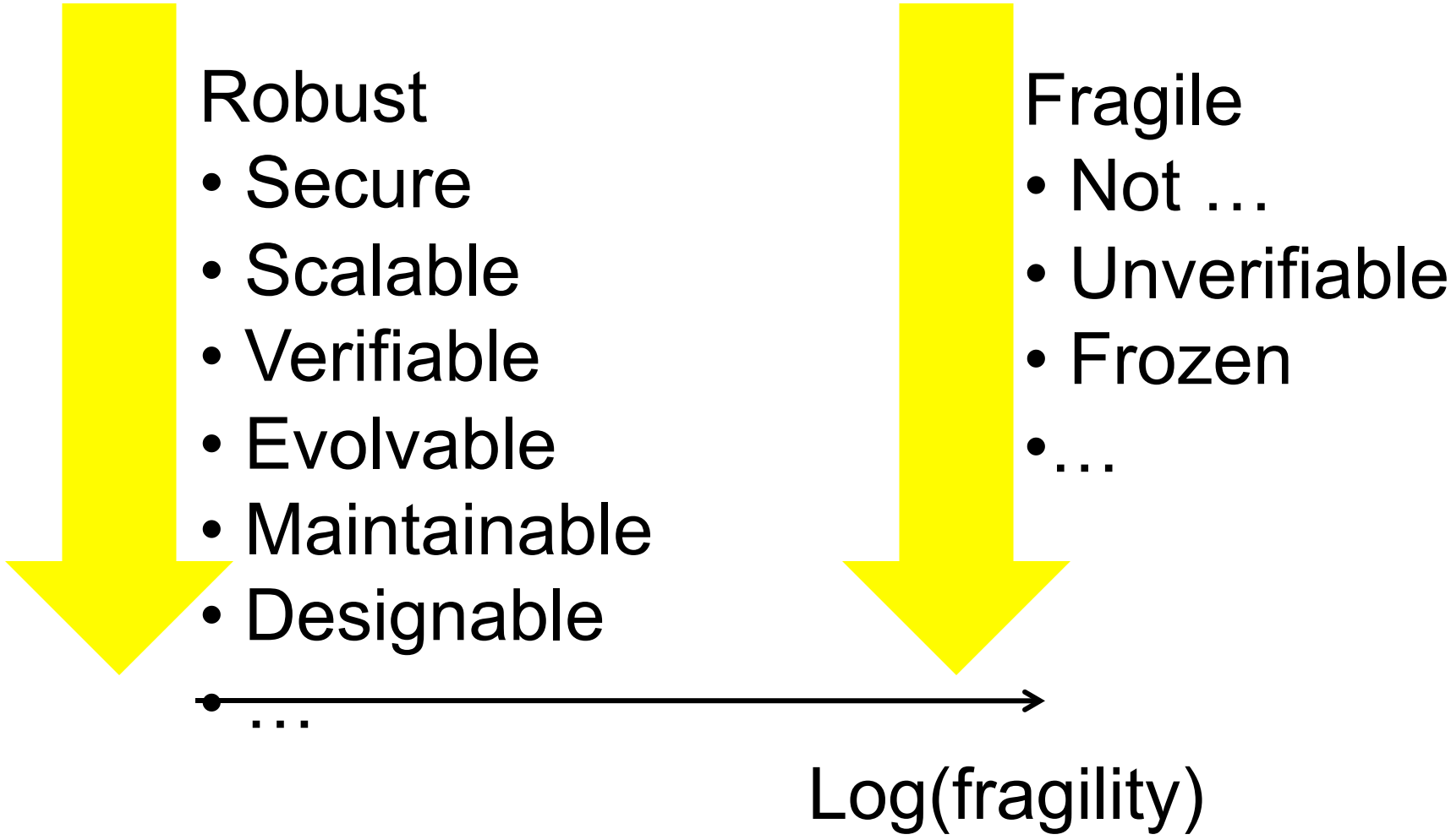
Other dimensions

Robust
- Secure
- Scalable
- Verifiable
- Evolvable
- Maintainable
- Designable
- …

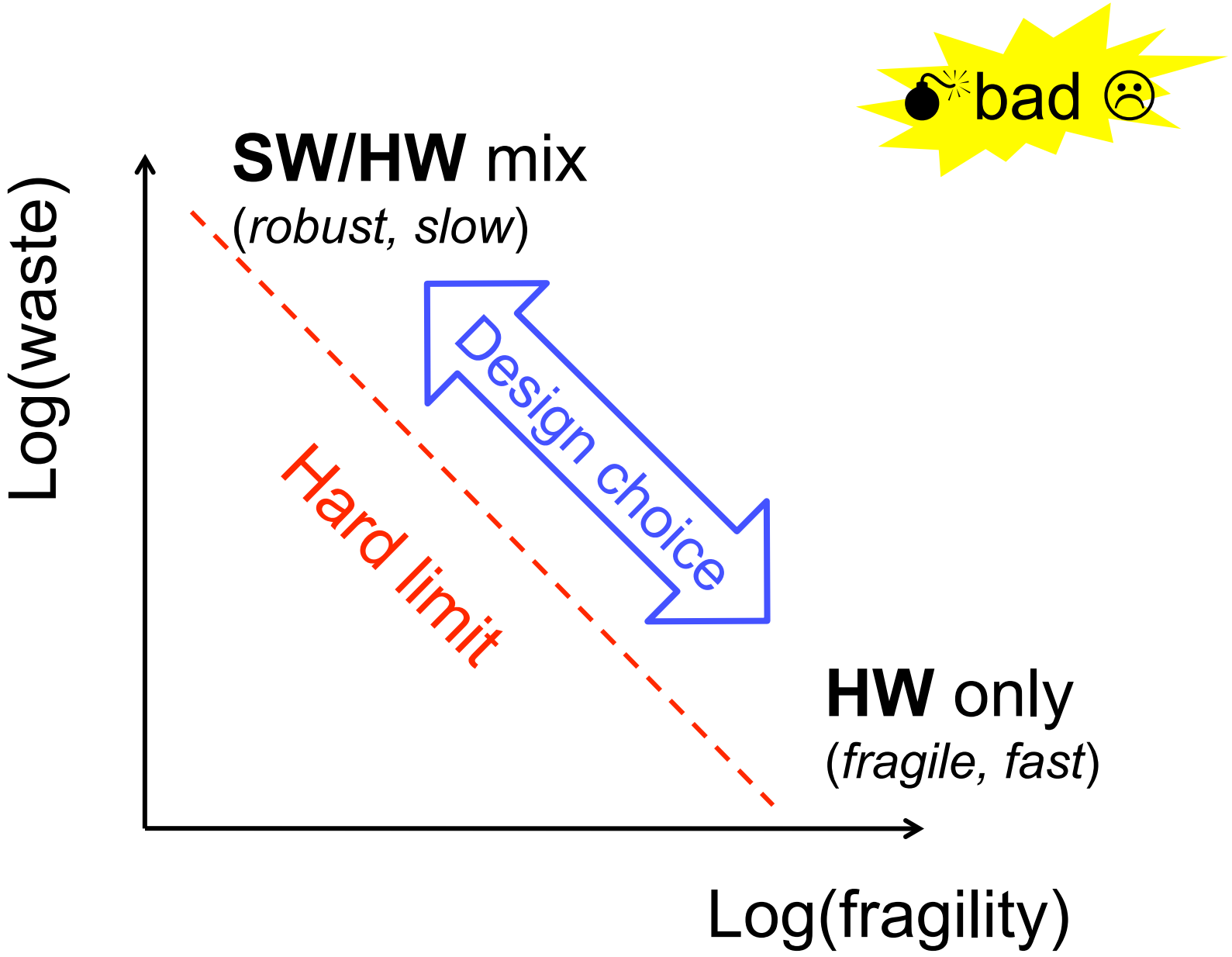
Fragile
- Insecure
- Not scalable
- Unverifiable
- Frozen
- …

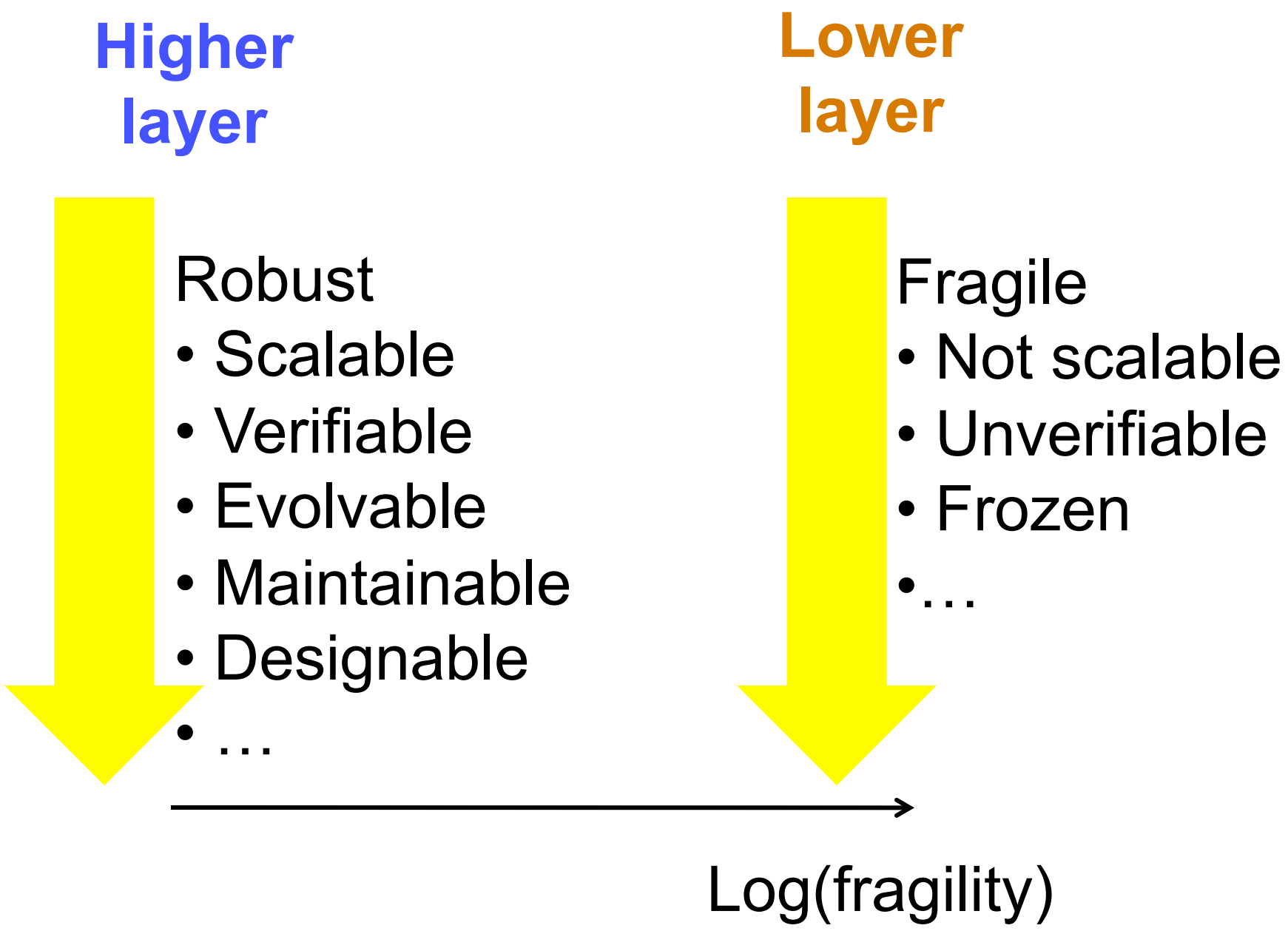
Log(fragility)

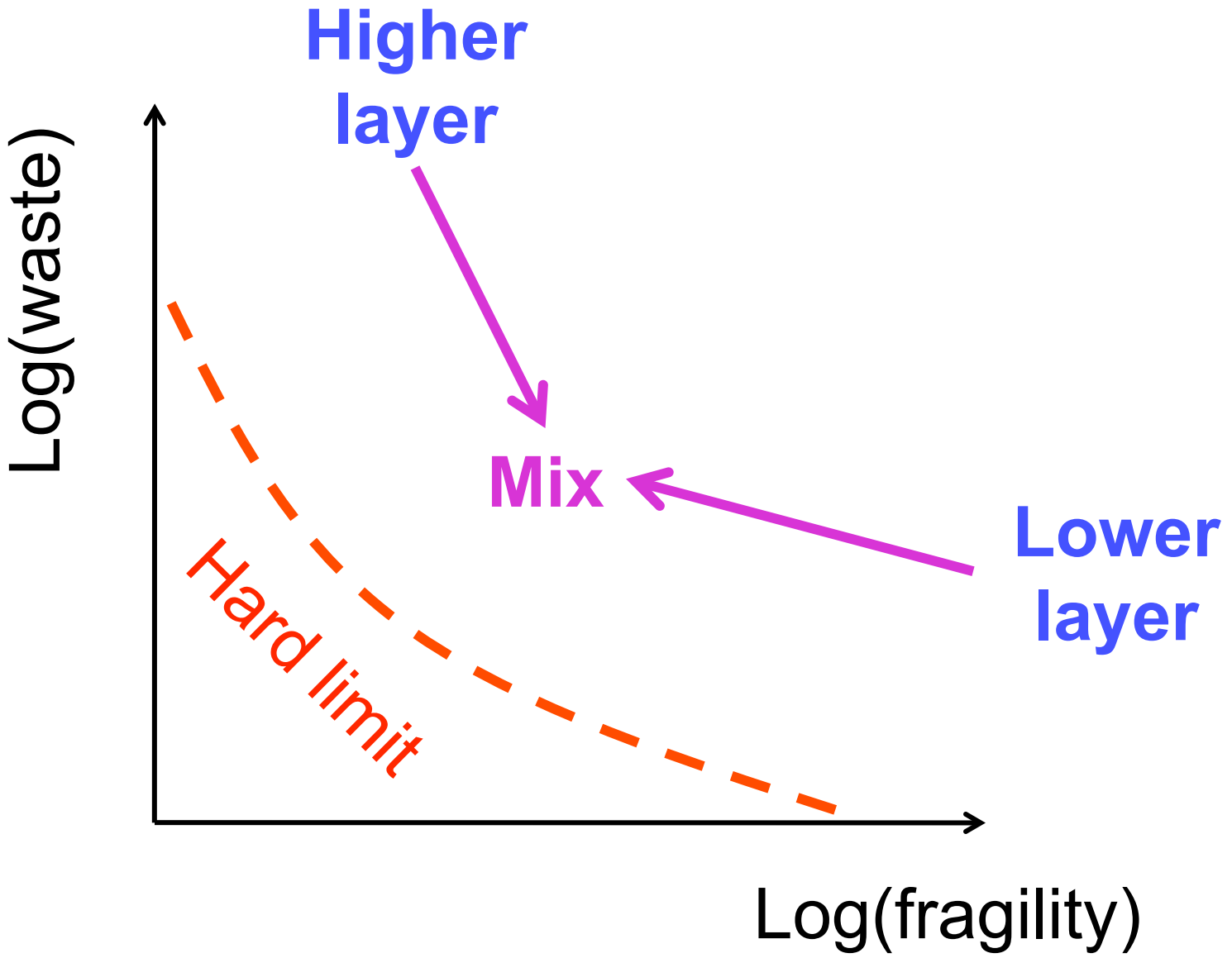
Collapse other dimensions

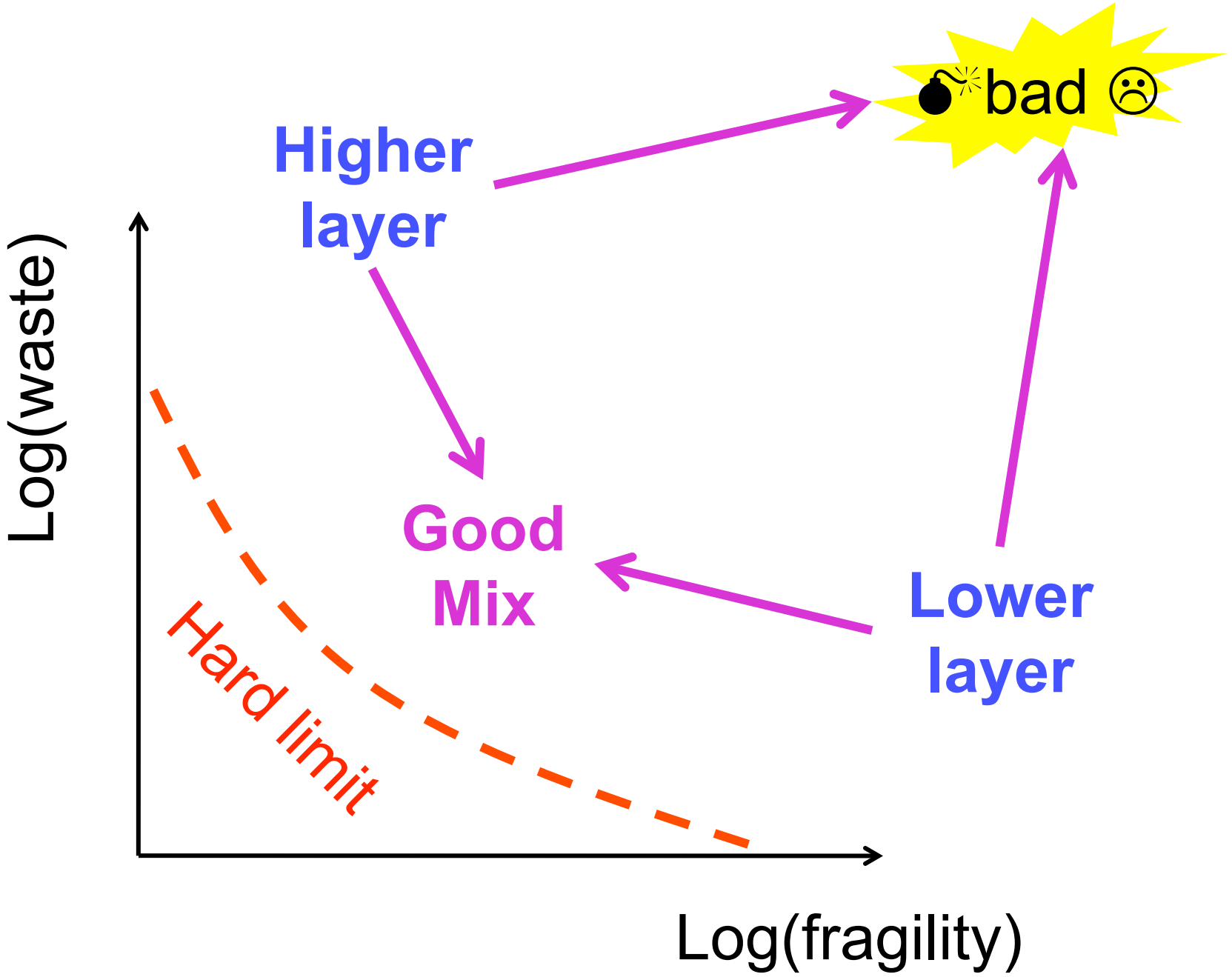


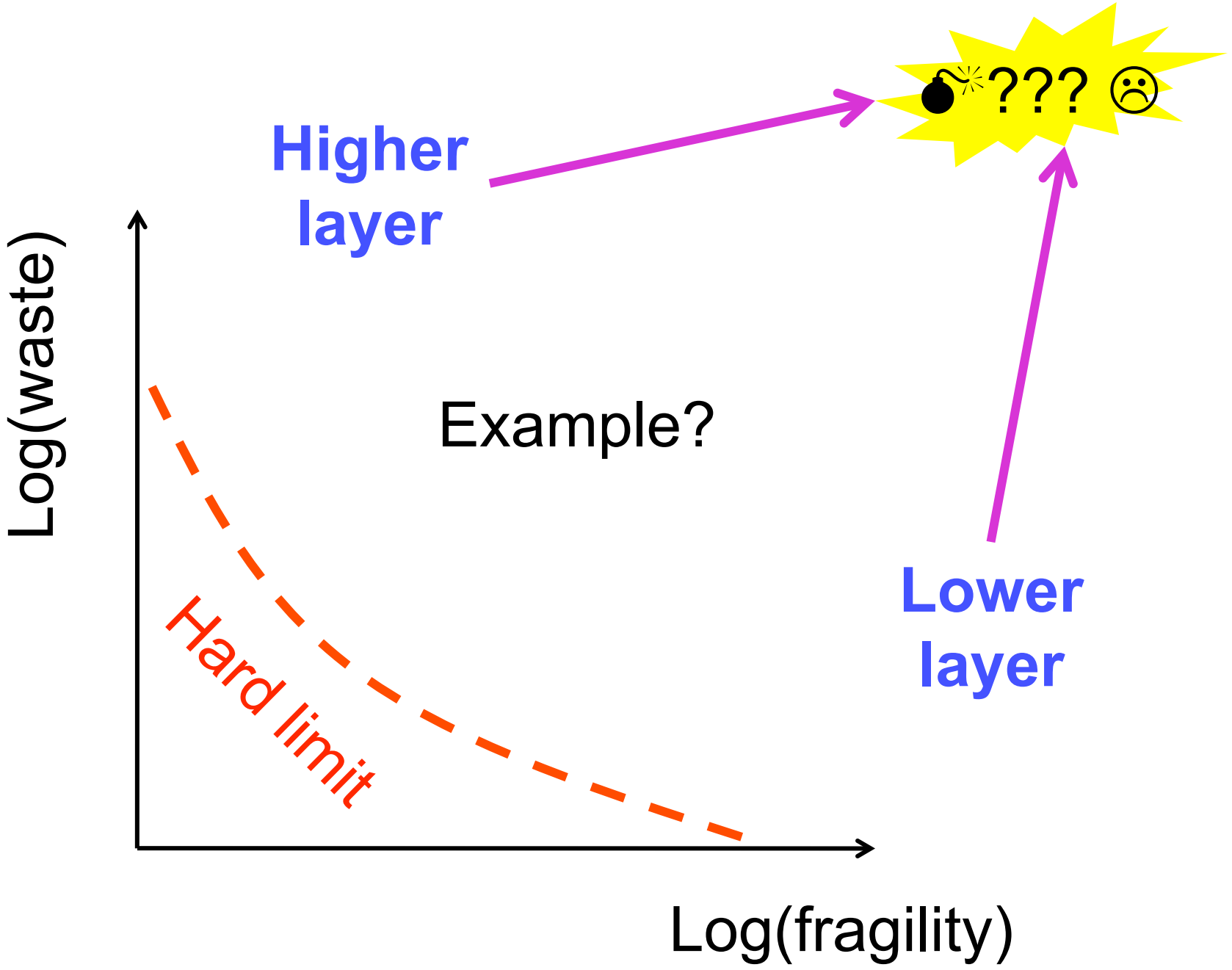
Log(fragility)


bad ☹
SW/HW mix
(robust, slow)
Design choice
Hard limit
HW only
(fragile, fast)
Log(waste)
Log(fragility)


Higher layer
Lower layer
Robust
• Scalable
• Verifiable
• Evolvable
• Maintainable
• Designable
• …
Fragile
• Not scalable
• Unverifiable
• Frozen
•…
Log(fragility)

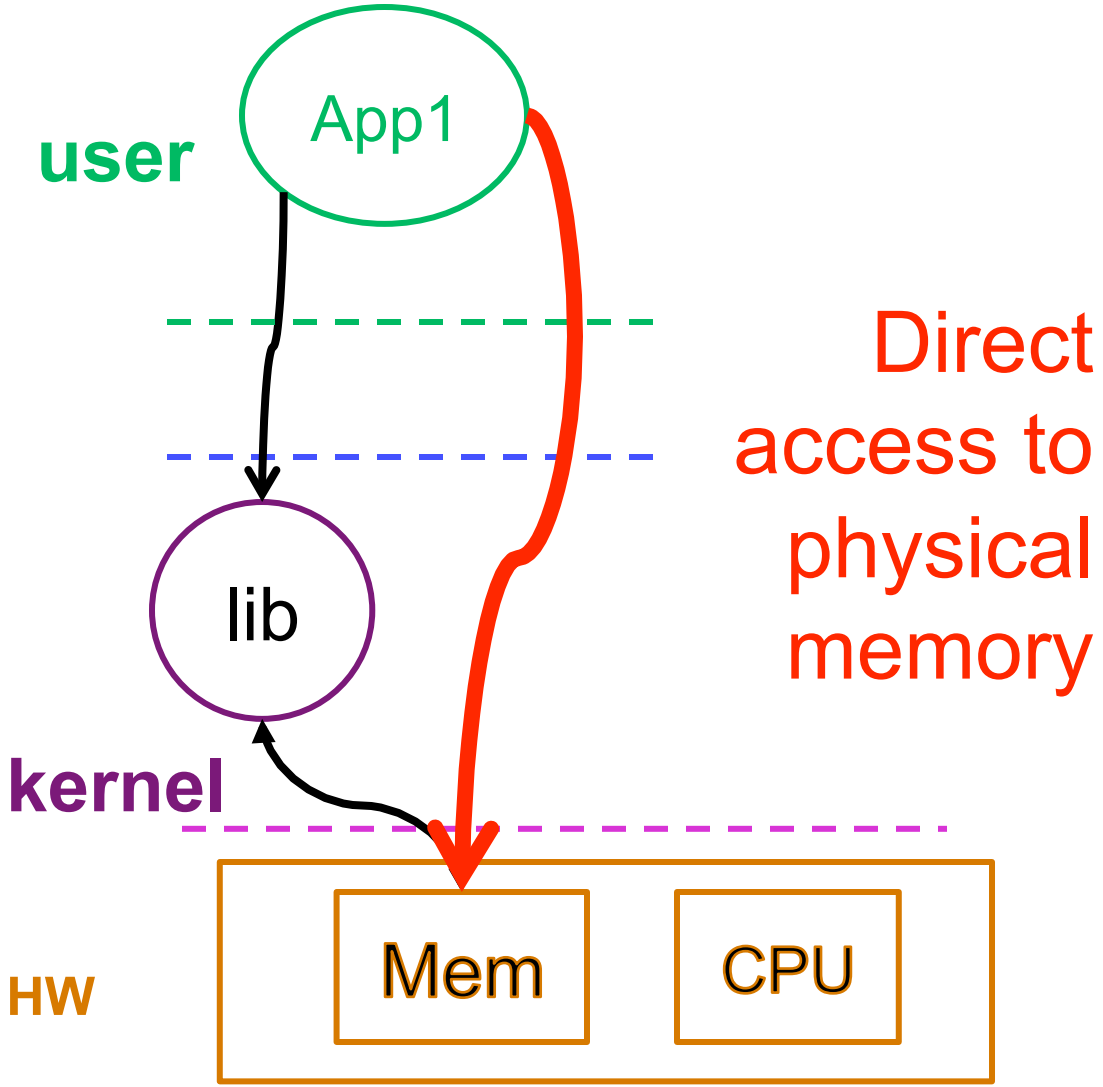

Higher layer
Log(waste)
Mix
Lower layer
Hard limit
Log(fragility)


bad
Higher layer
Good Mix
Lower layer
Hard limit
Log(waste)
Log(fragility)


???
Higher layer
Example?
Lower layer
Log(waste)
Hard limit
Log(fragility)

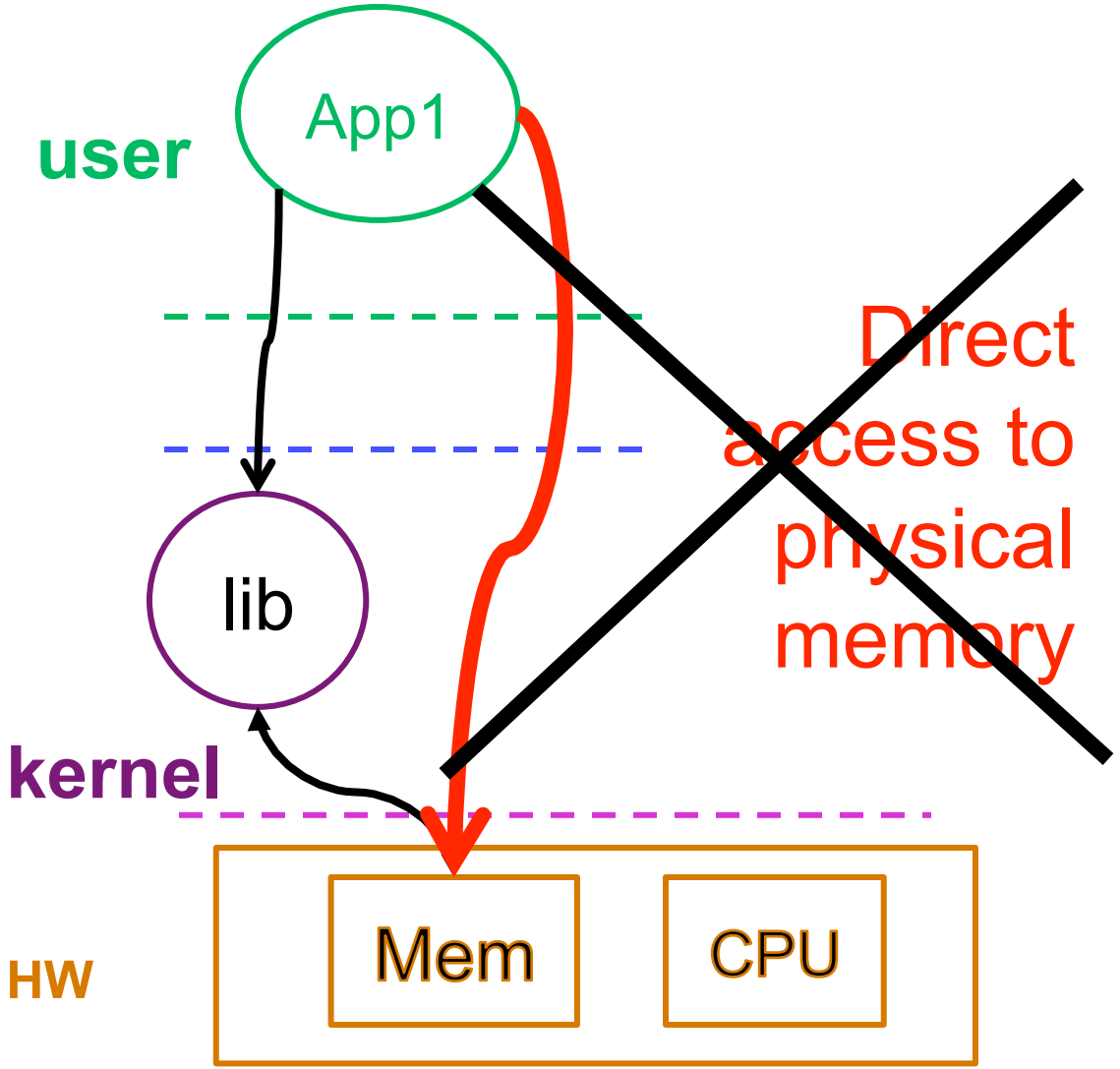
Don’t cross layers

- Robust
- Secure
- Scalable
- Verifiable
- Evolvable
- Maintainable
- Designable
- …



Separate logical names and physical addresses

~~Robust~~
- ~~Secure~~
- ~~Scalable~~
- ~~Verifiable~~
- ~~Evolvable~~
- ~~Maintainable~~
- ~~Designable~~
- ~~…~~

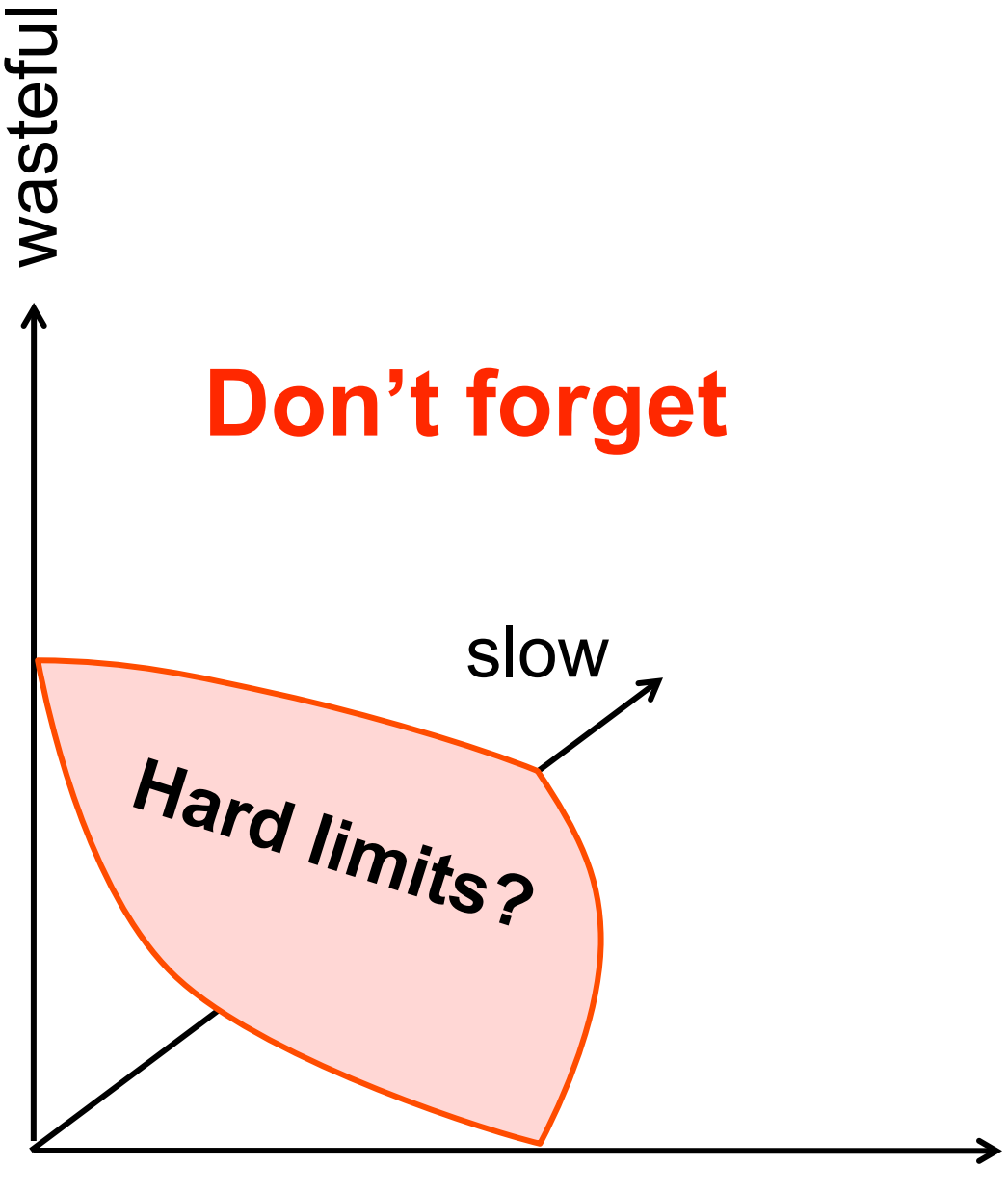


Separate logical names and physical addresses

Naming and addressing are important topics in OS

Needs to be an even richer topic in networking

So, finally, let's look at a minimal network

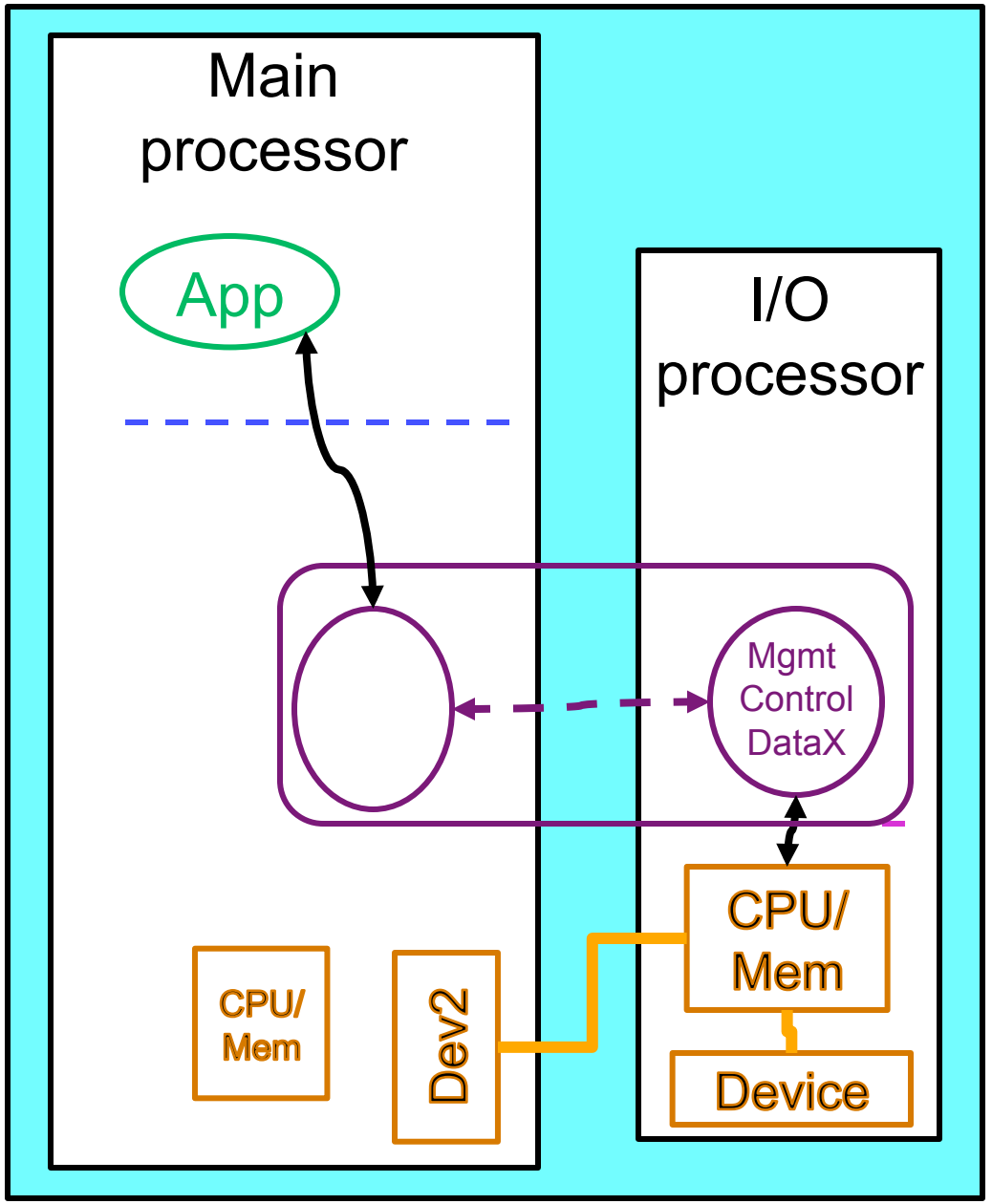


Don’t forget

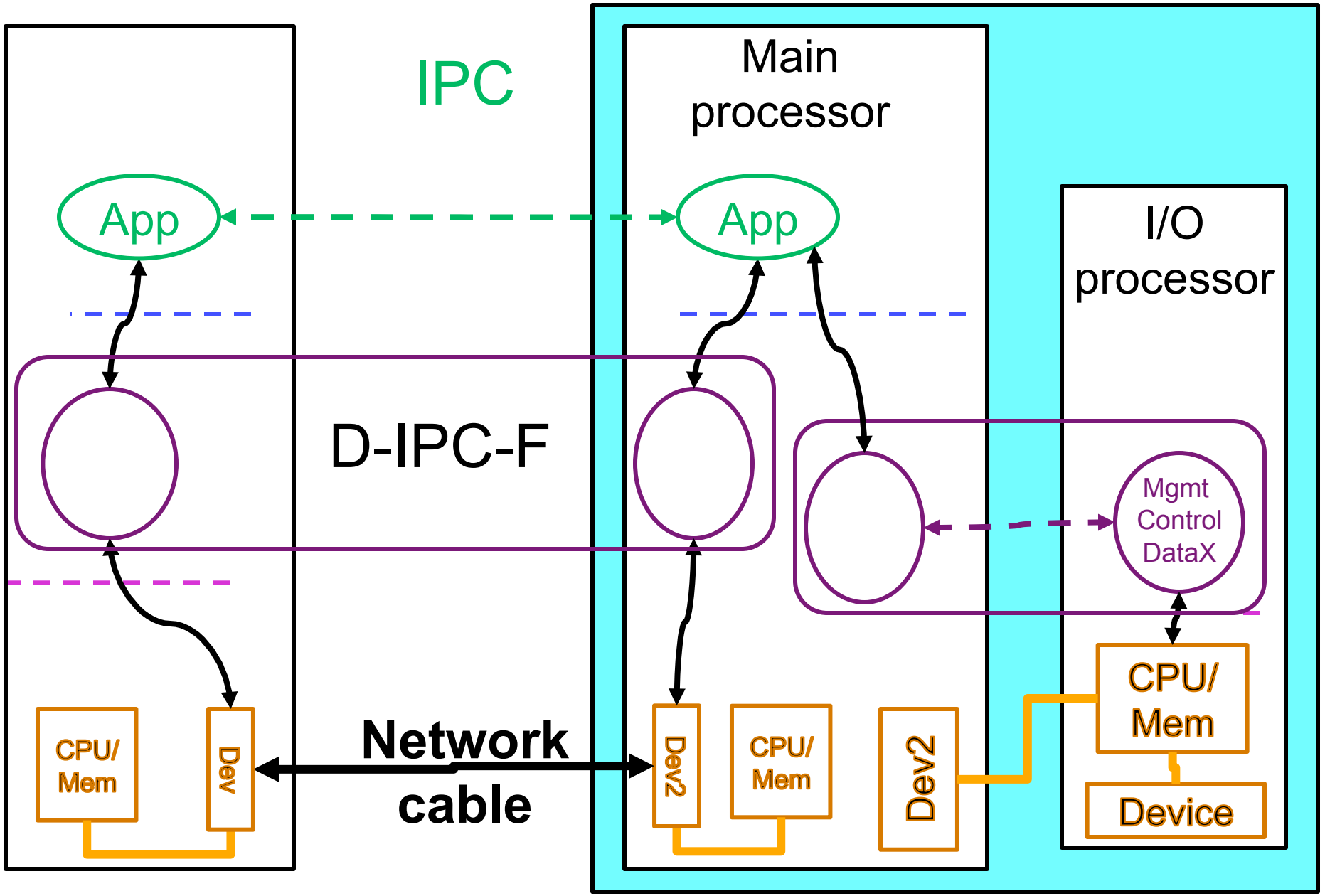
- Thermo (Carnot)
- Comms (Shannon)
- Control (Bode)
- Computation (Turing)

- Each focuses on few dimensions
- Important tradeoffs are across these areas
- Speed vs efficiency vs robustness vs …

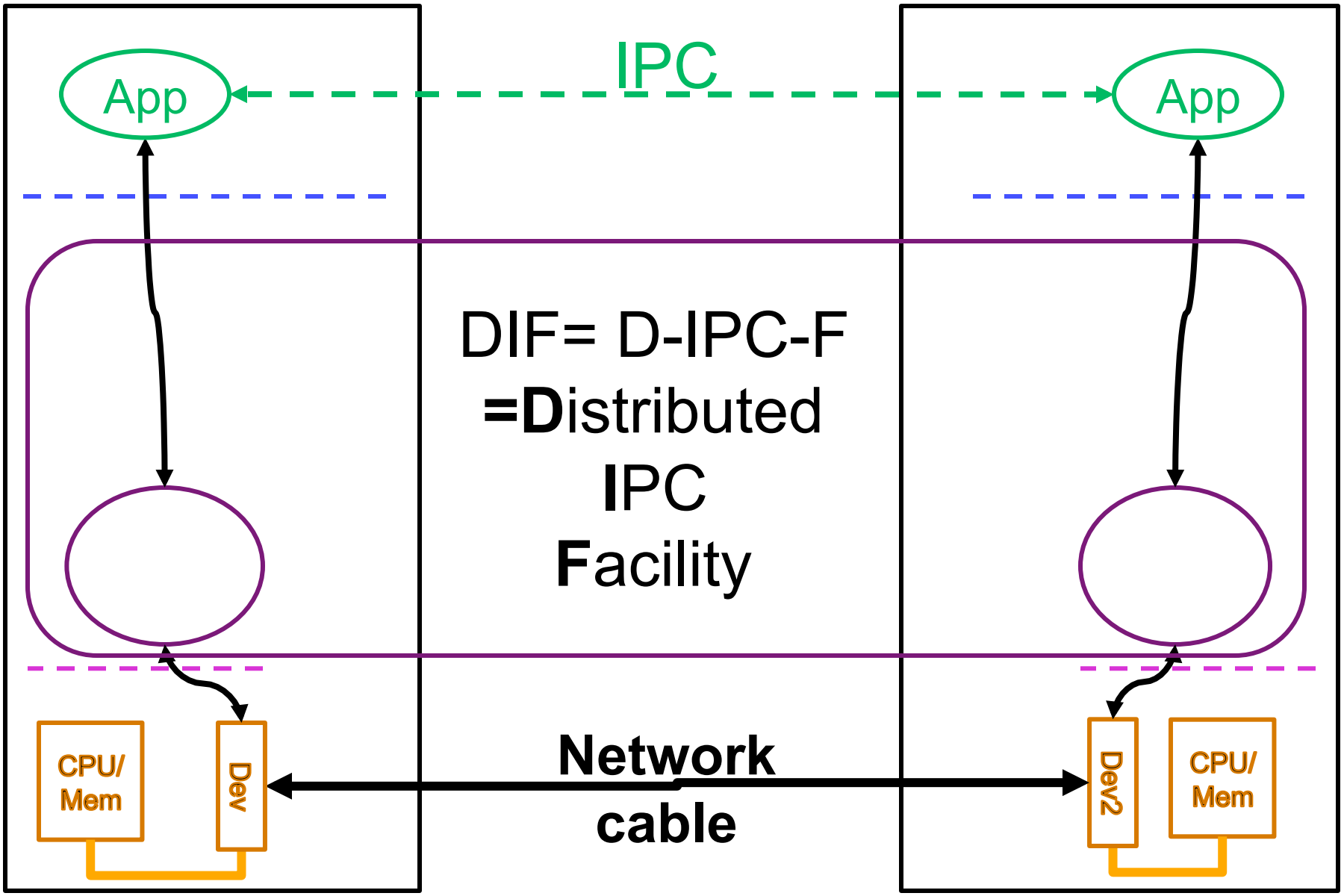
Might be all in the same “box”.



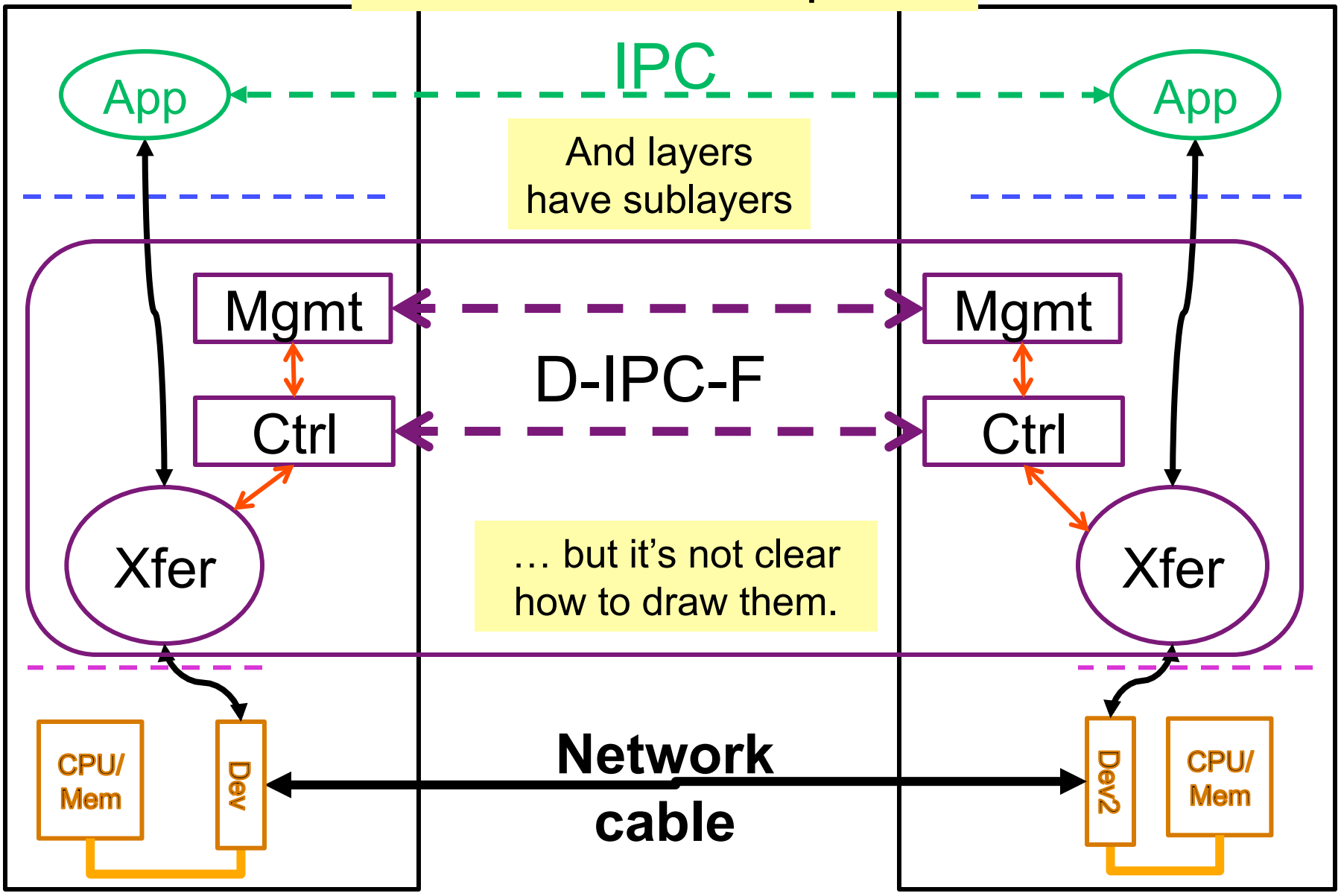
A network with another “box”…



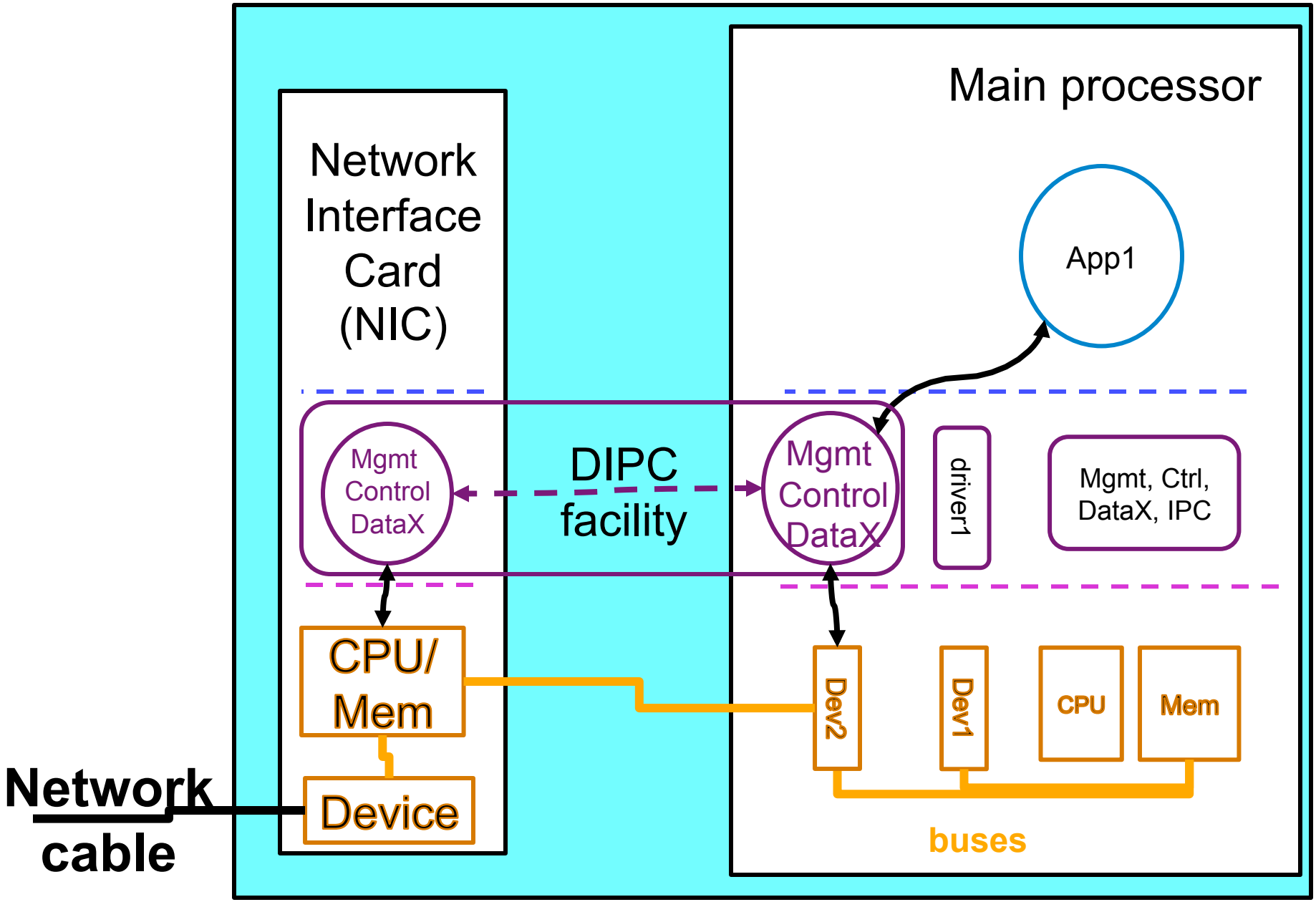
A minimal network without a NIC.



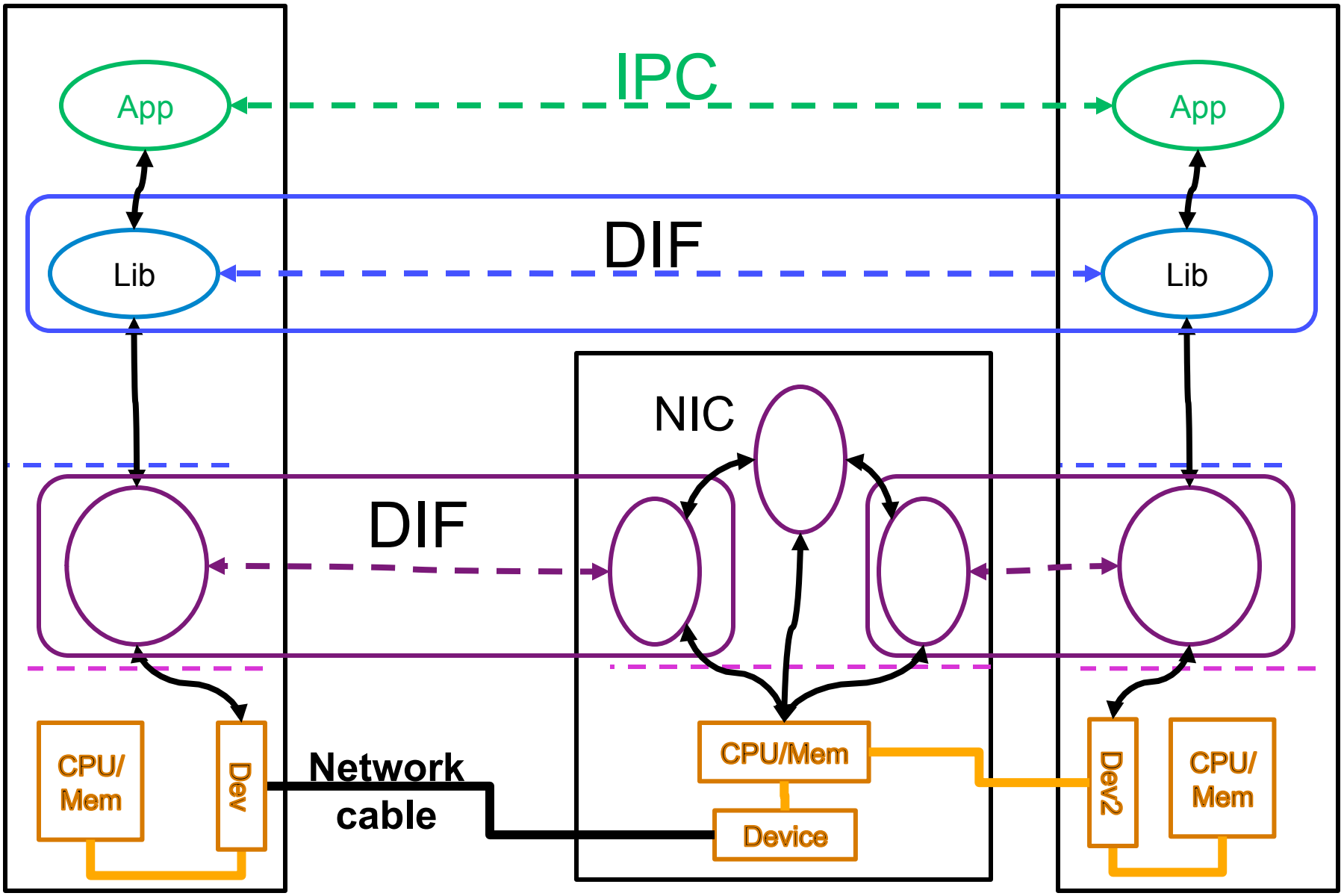
Mgmt and Cntrl become even more complex



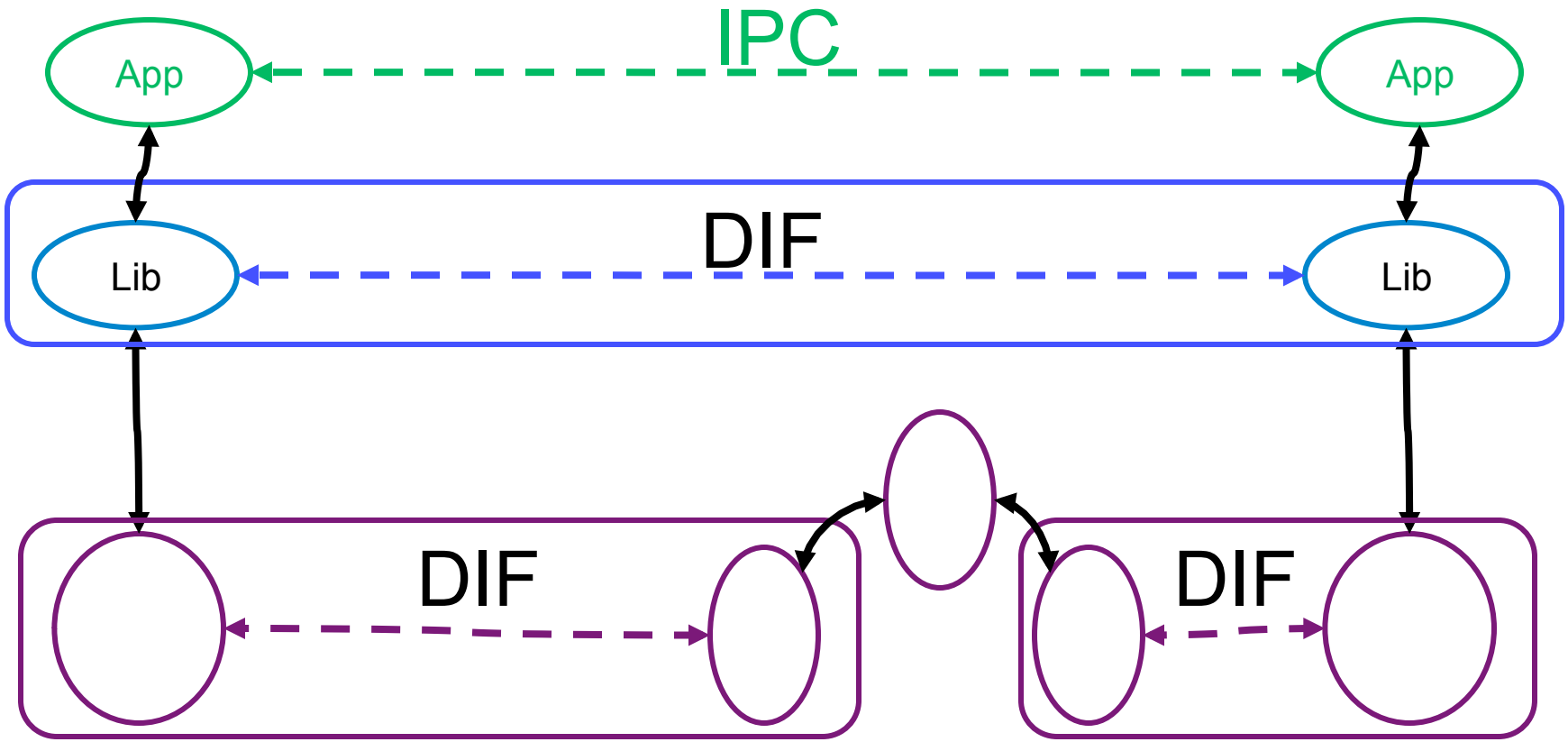
What is a NIC?


Main processor
Network Interface Card (NIC)
App1
Mgmt Control DataX
DIPC facility
Mgmt Control DataX
driver1
Mgmt, Ctrl, DataX, IPC
CPU/ Mem
Dev2
Dev1
CPU
Mem
Device
buses
Network cable

A minimal network with a NIC

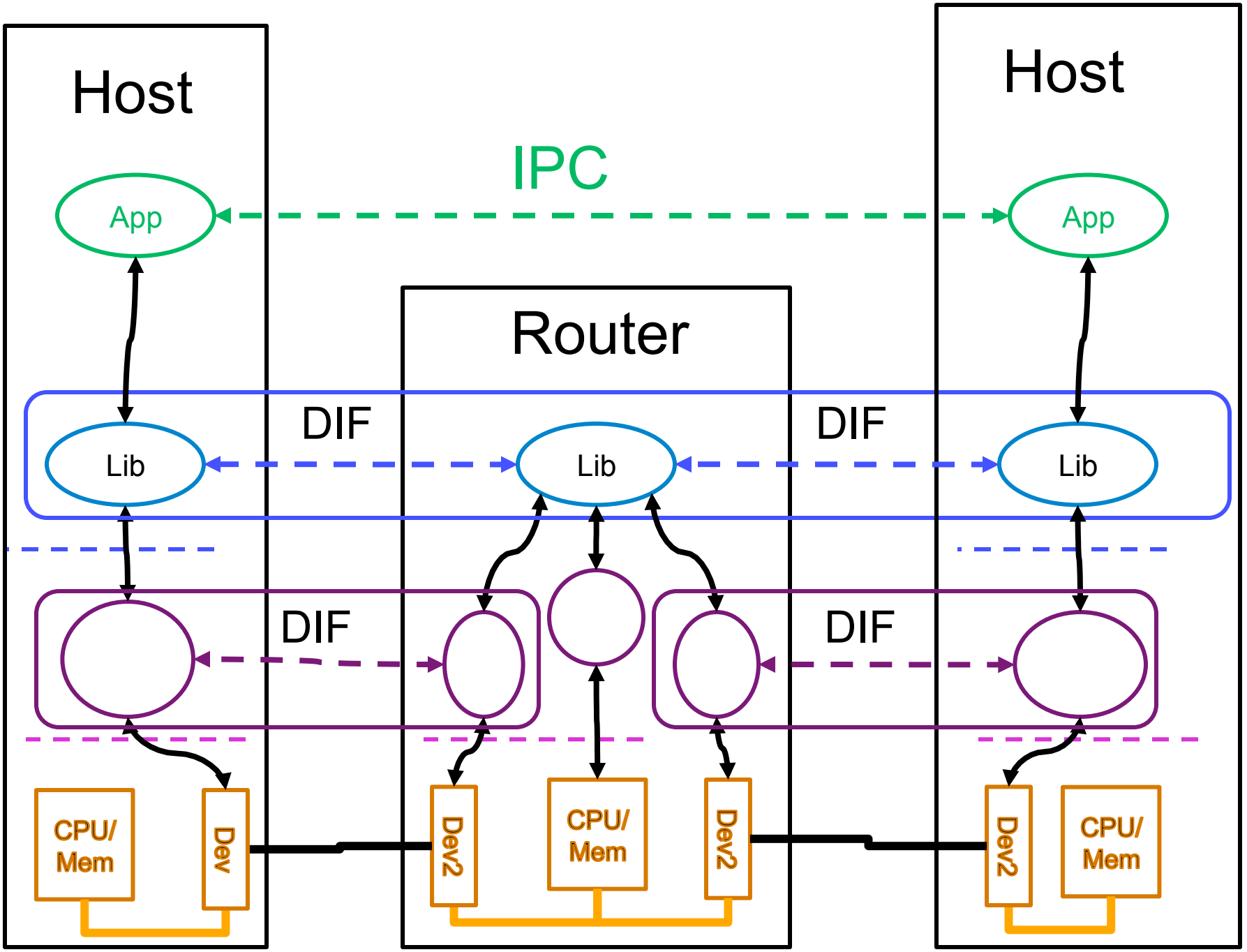


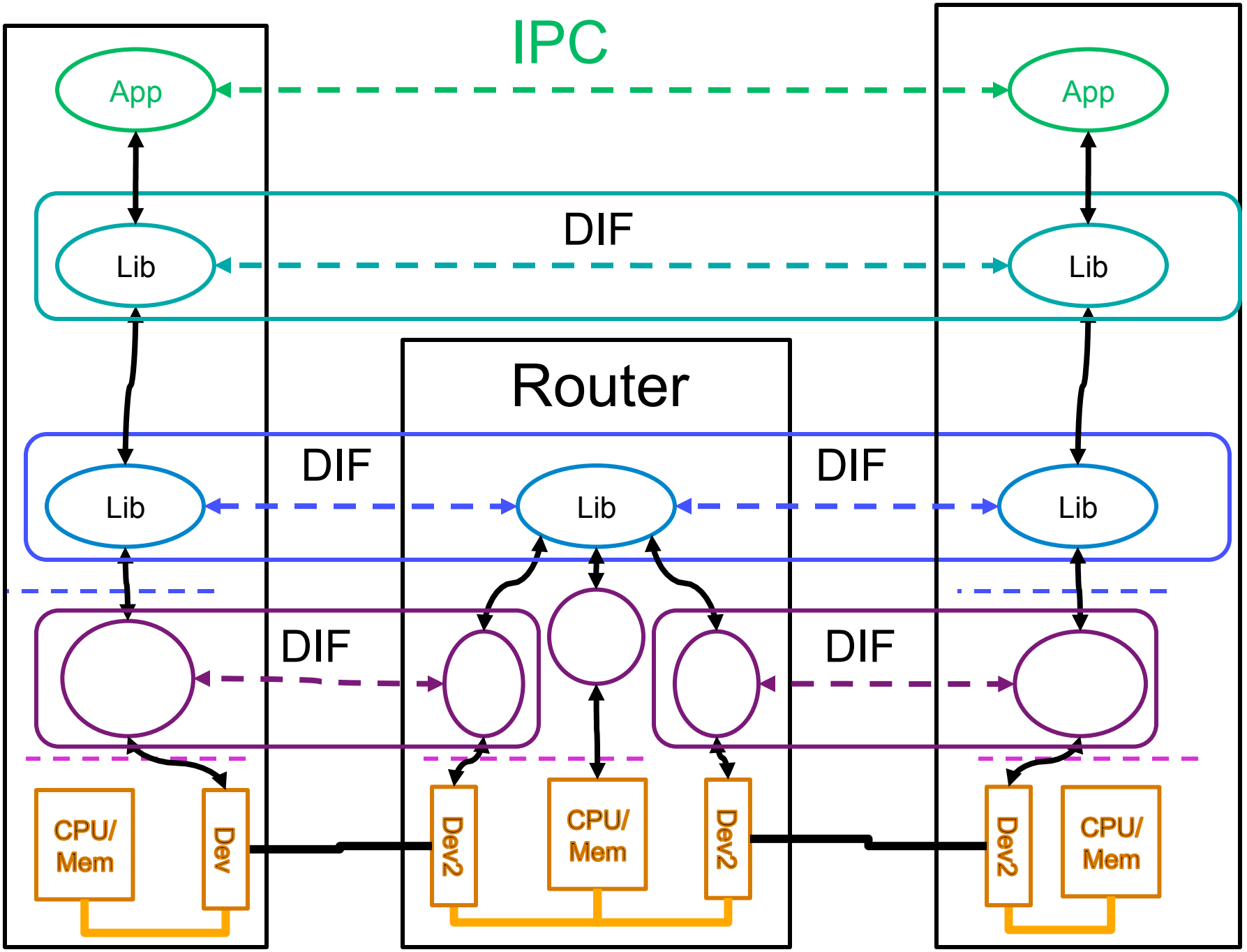
More layers



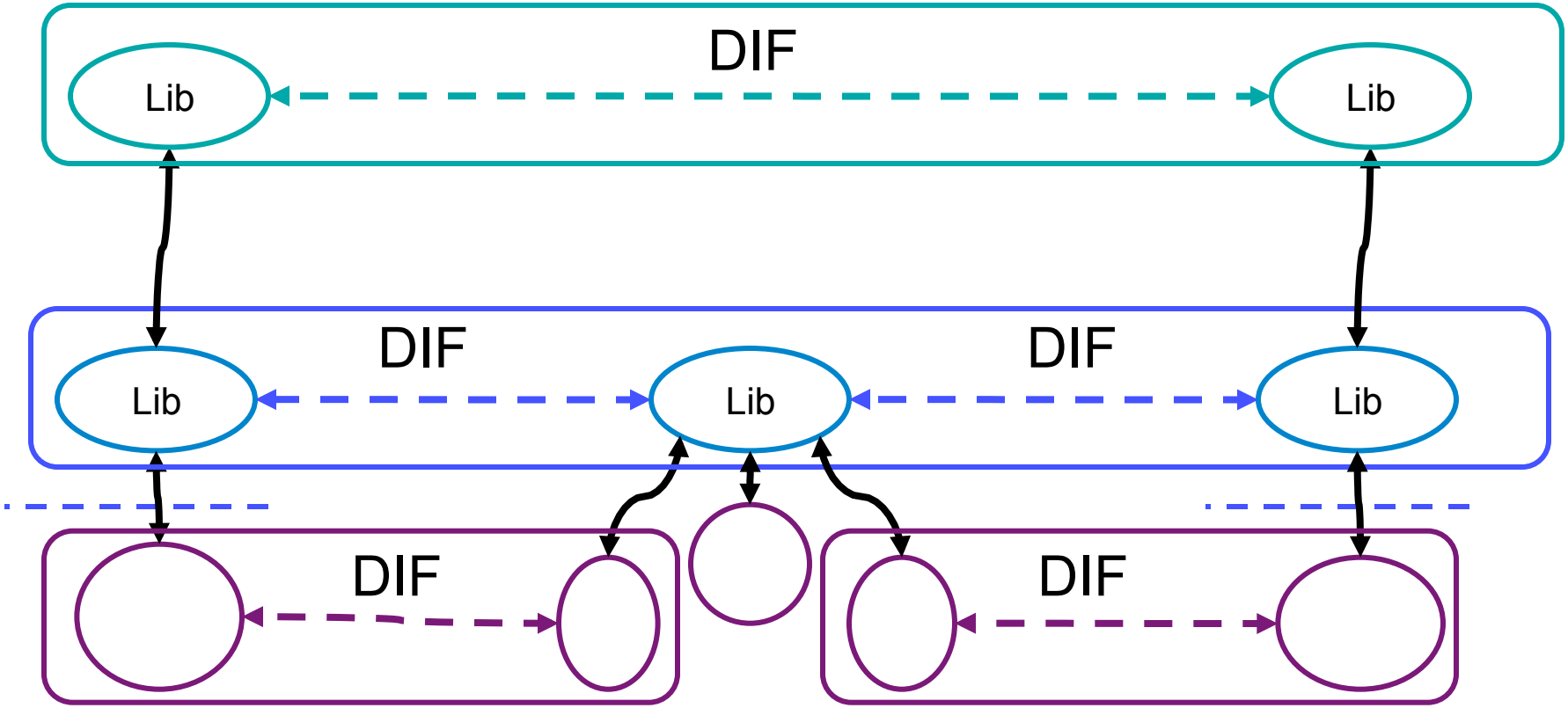
Different scopes

DIF= **D**istributed **I**PC **F**acility


Host
Host
IPC
App
App
Router
DIF
DIF
Lib
Lib
Lib
DIF
DIF
CPU/
Mem
Dev
Dev2
CPU/
Mem
Dev2
Dev2
CPU/
Mem


IPC
App
App
DIF
Lib
Lib
Router
DIF
DIF
Lib
Lib
Lib
DIF
DIF
CPU/
Mem
Dev
Dev2
CPU/
Mem
Dev2
Dev2
CPU/
Mem

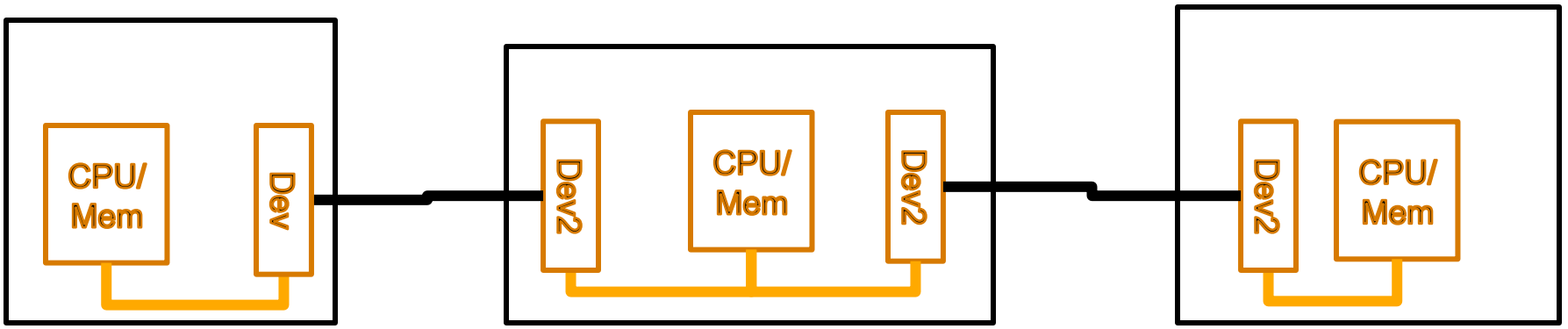
How many layers?

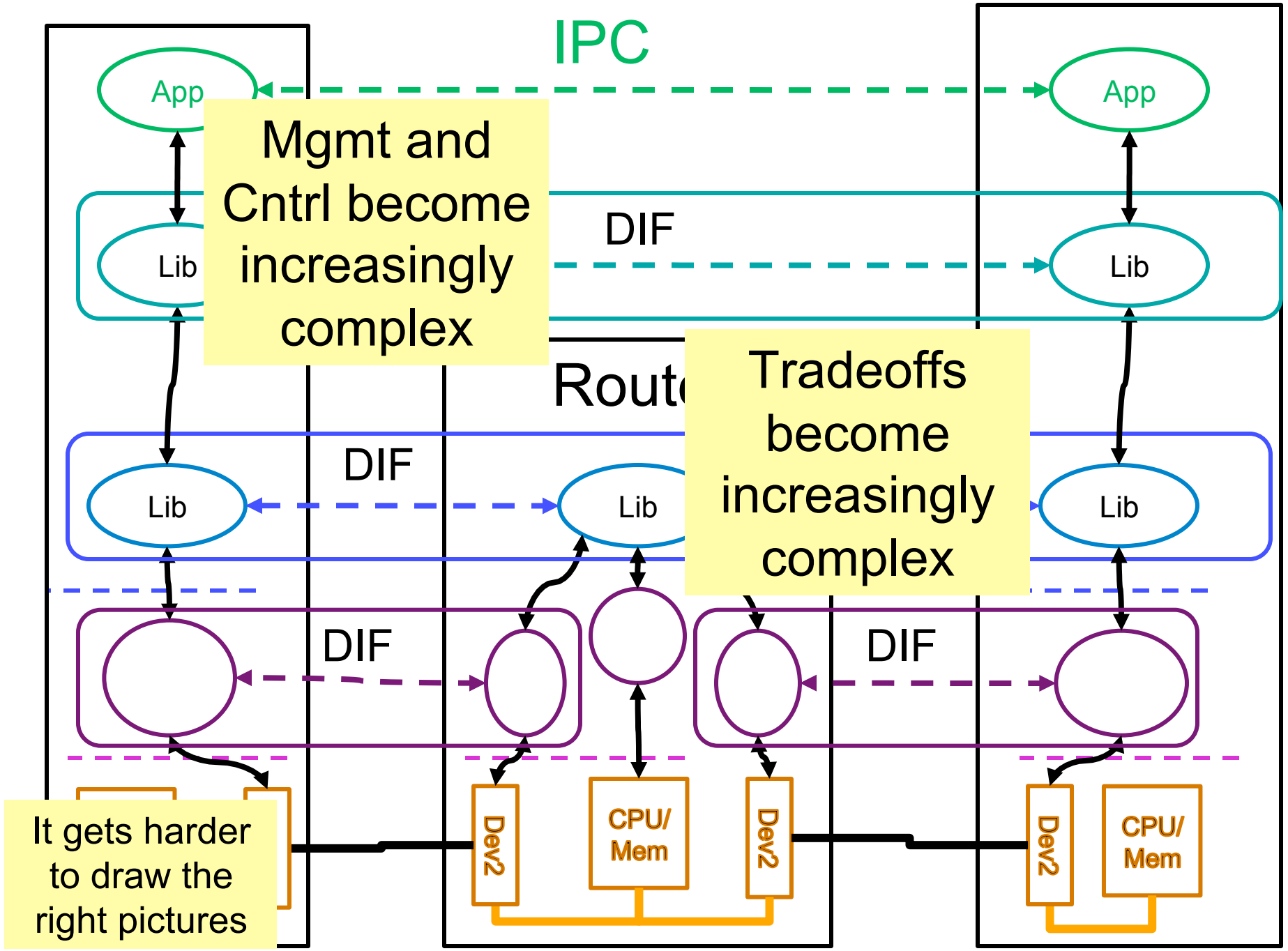


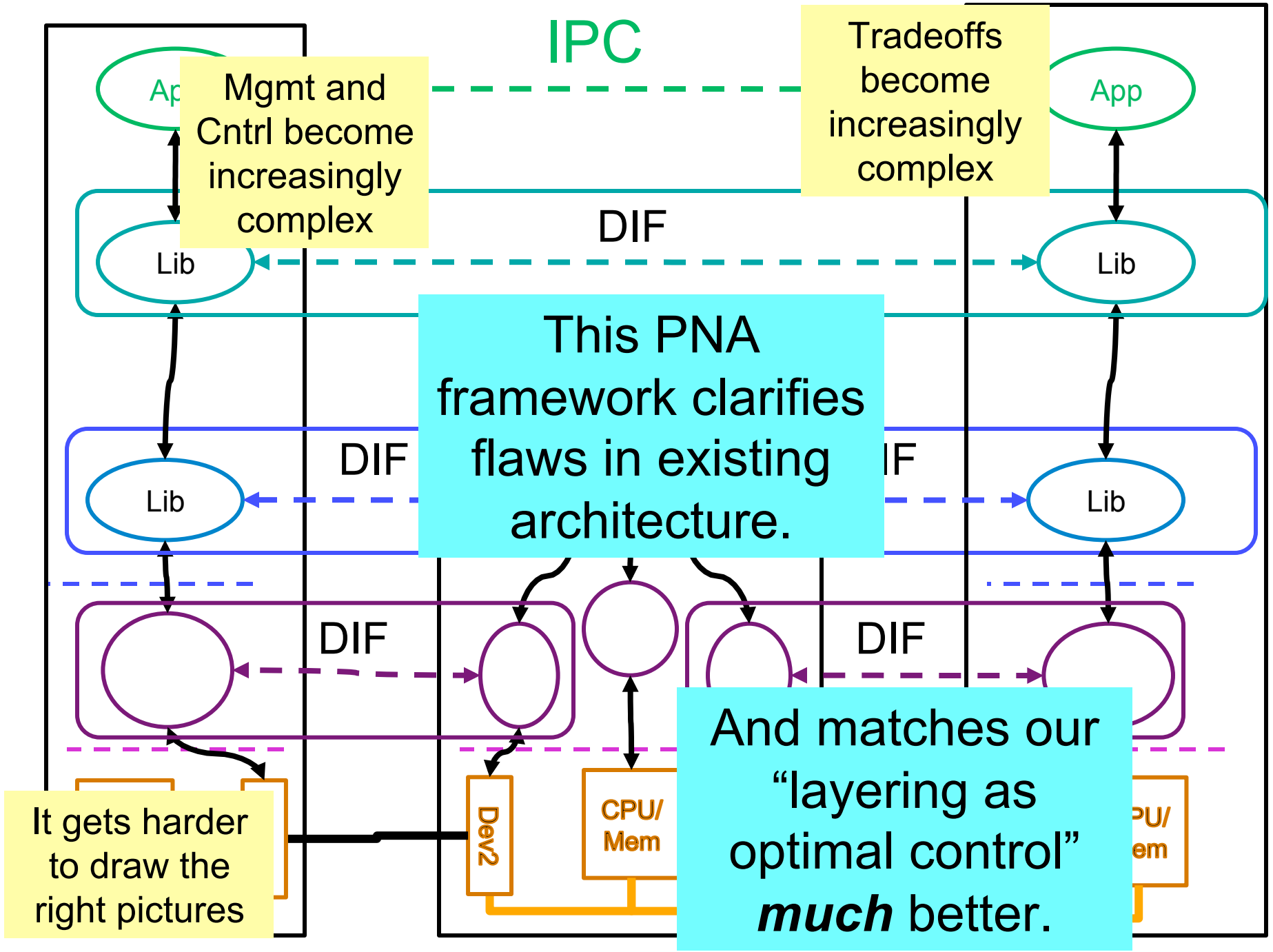
How many layers?

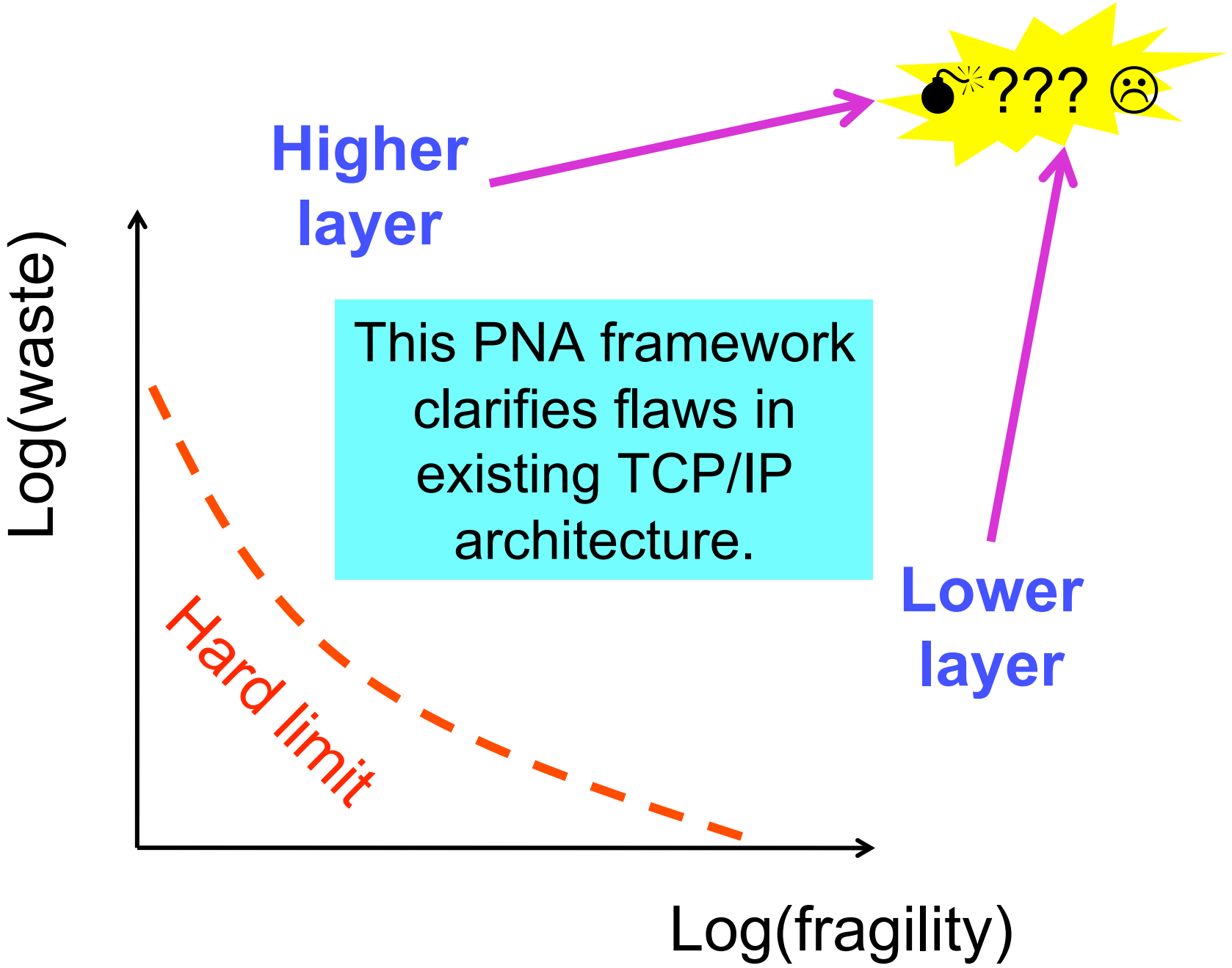
As many as you need to map distribute applications

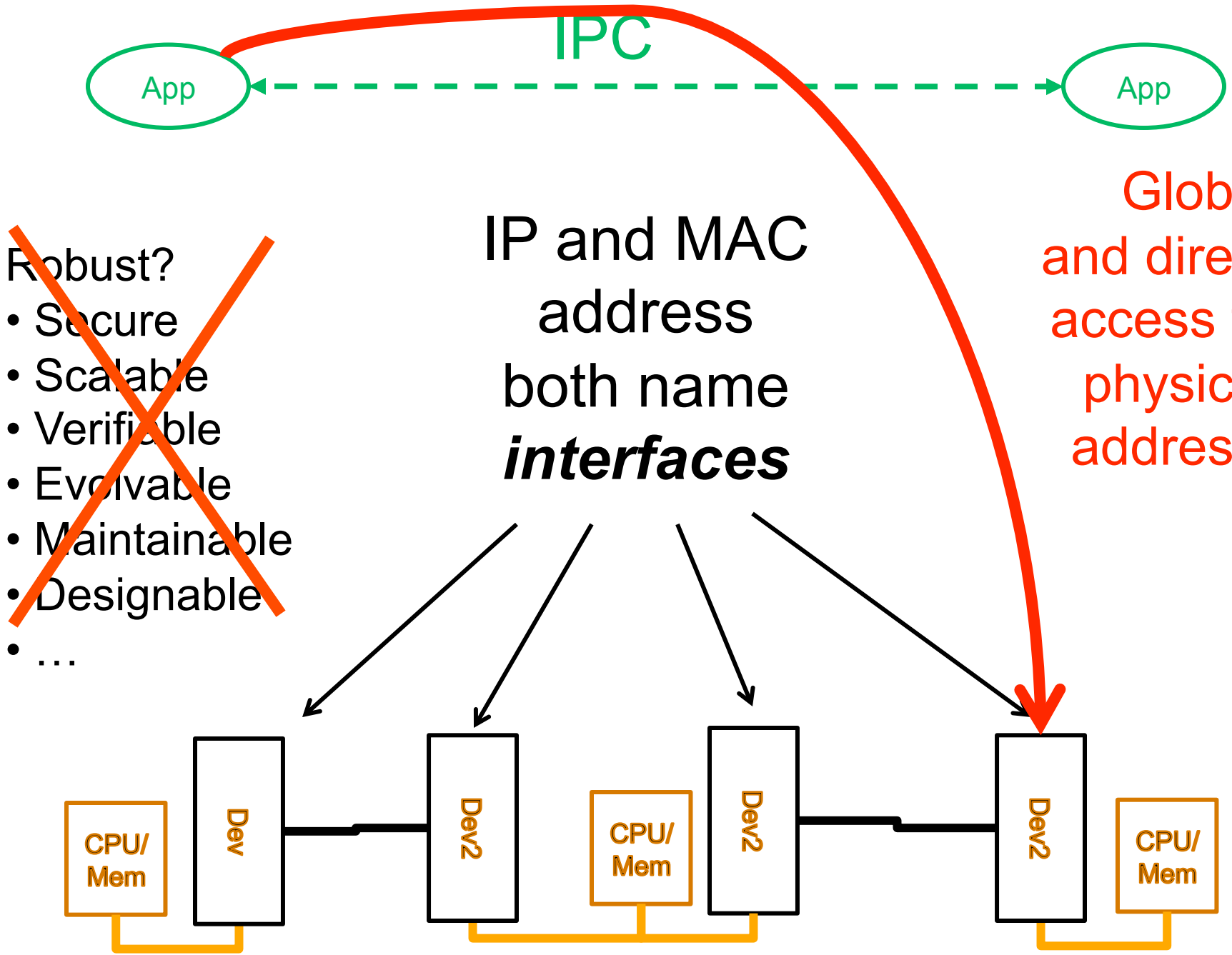
Onto distributed resources



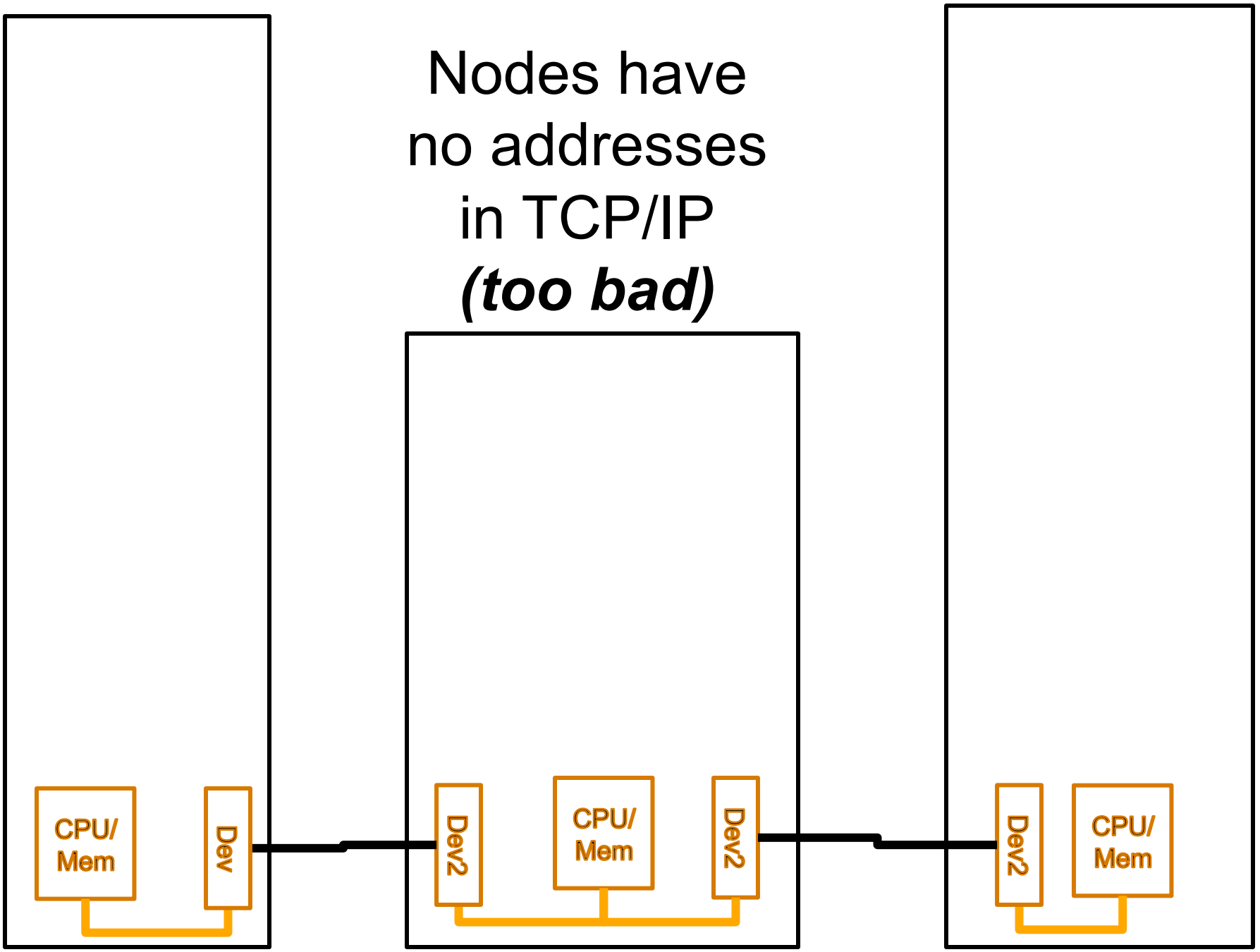

IPC
App
App
Mgmt and Cntrl become increasingly complex
DIF
Lib
Lib
Route
Tradeoffs become increasingly complex
DIF
Lib
Lib
Lib
DIF
DIF
It gets harder to draw the right pictures
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem


IPC
Mgmt and Cntrl become increasingly complex
Tradeoffs become increasingly complex
App
App
DIF
Lib
Lib
This PNA framework clarifies flaws in existing architecture.
DIF
Lib
Lib
DIF
DIF
Dev2
CPU/ Mem
It gets harder to draw the right pictures
And matches our "layering as optimal control" ***much*** better.

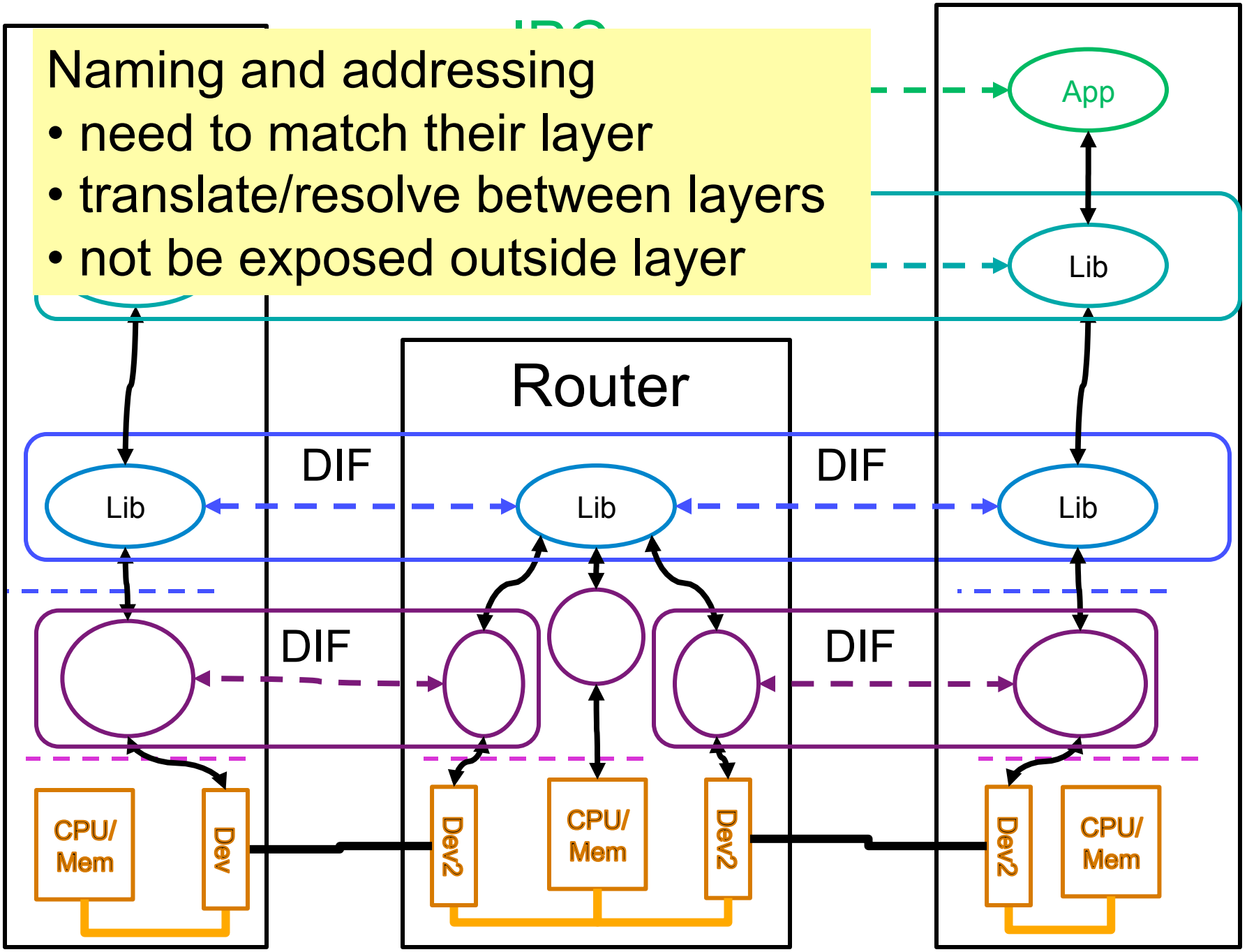

???
Higher layer
Lower layer
This PNA framework clarifies flaws in existing TCP/IP architecture.
Log(waste)
Log(fragility)
Hard limit


IPC
App
App
Global and direct access to physical address!
Robust?
• Secure
• Scalable
• Verifiable
• Evolvable
• Maintainable
• Designable
• …
IP and MAC address both name ***interfaces***
CPU/ Mem
Dev
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem

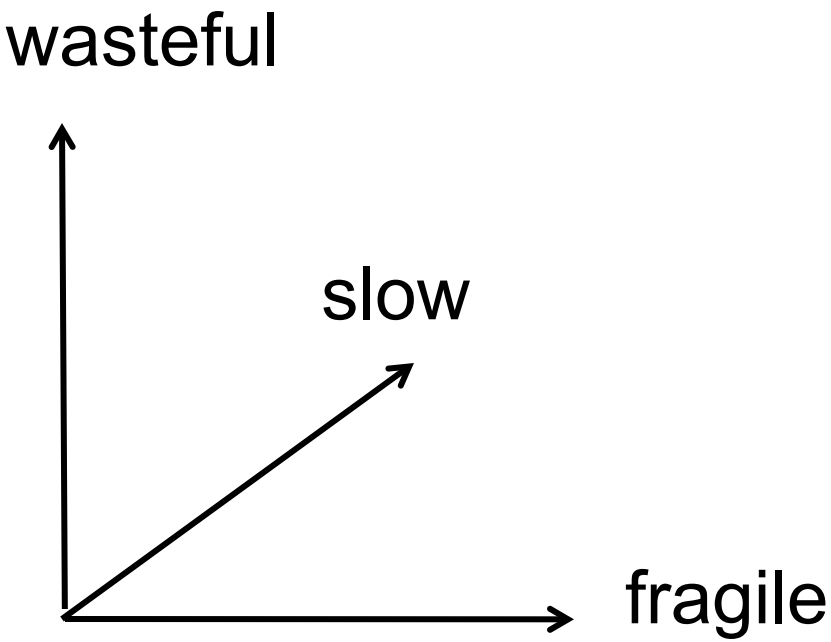
Nodes have
no addresses
in TCP/IP
(too bad)



Naming and addressing
- need to match their layer
- translate/resolve between layers
- not be exposed outside layer



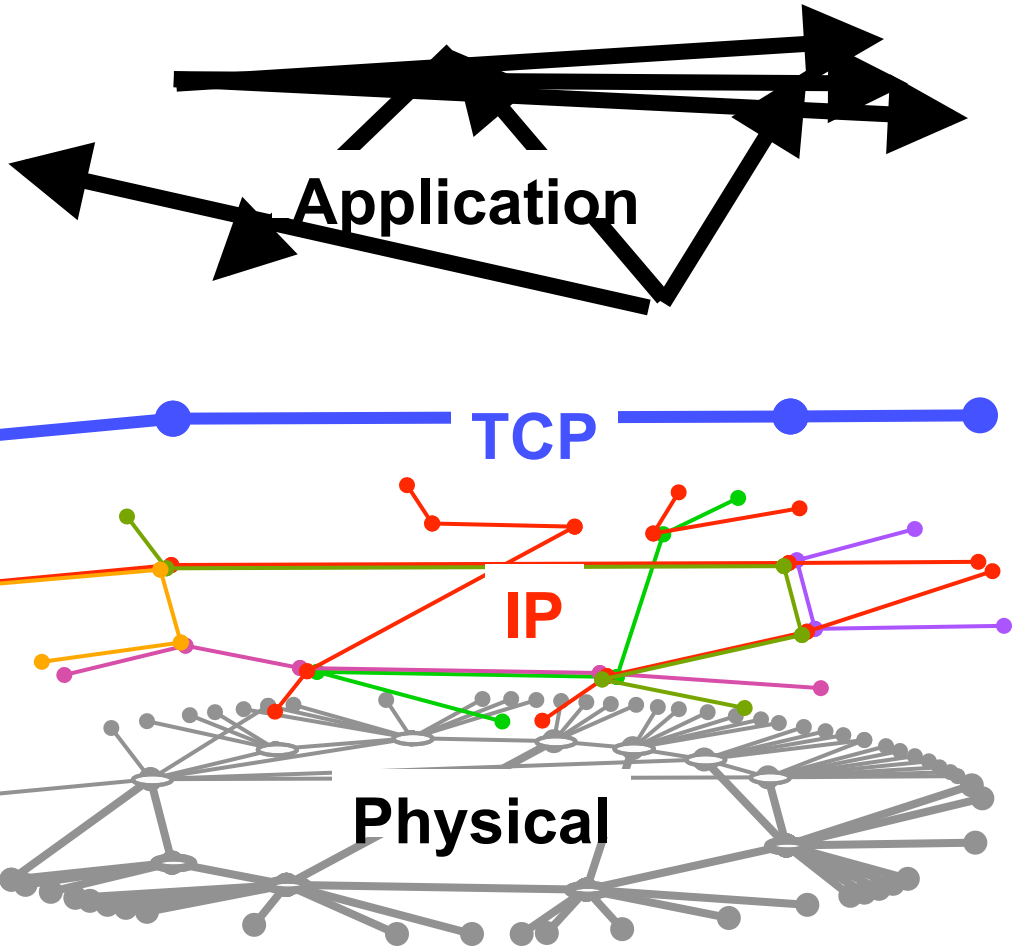
Naming and addressing
- need to match their layer
- translate/resolve between layers
- not be exposed outside layer
- ***familiar tradeoffs here***



Tradeoffs
- Addressing complexity
- Table sizes
- Forwarding
- Optimal routes
- Table updates

Naming and addressing
- need to match their layer
- translate/resolve between layers
- not be exposed outside layer

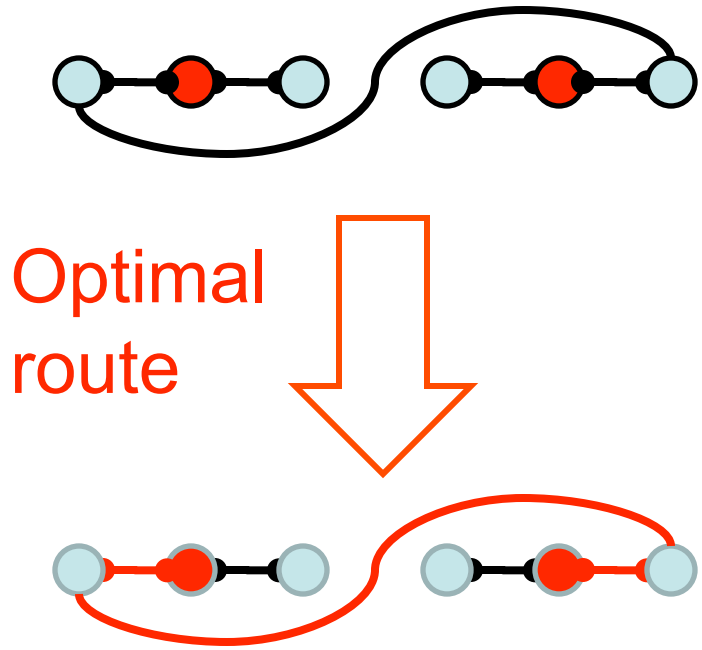
Architecture issues
- DNS
- NATS
- Firewalls
- Multihoming
- Mobility
- Routing table size
- Overlays
- …



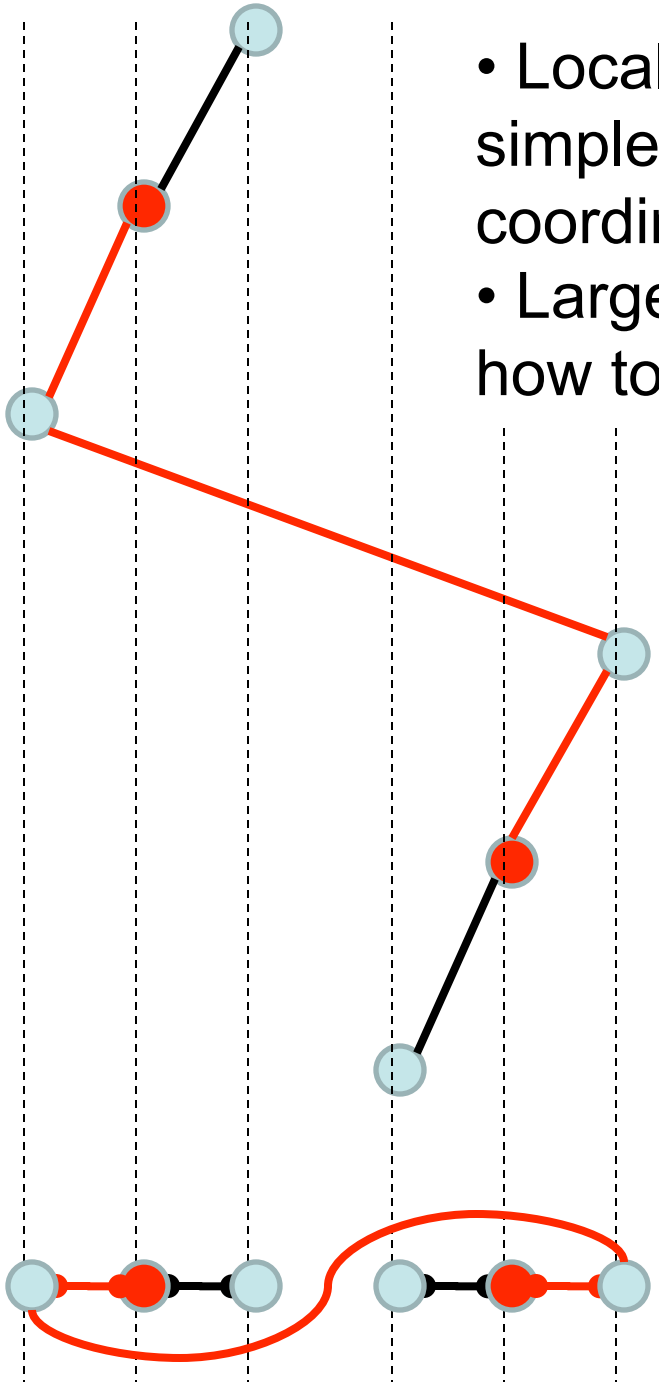
Trivial toy example

Consider a 1 dimensional geography

- Assume some link connectivity
- Optimal route might be indirect
- Consider route between red nodes



A graph in “1d”



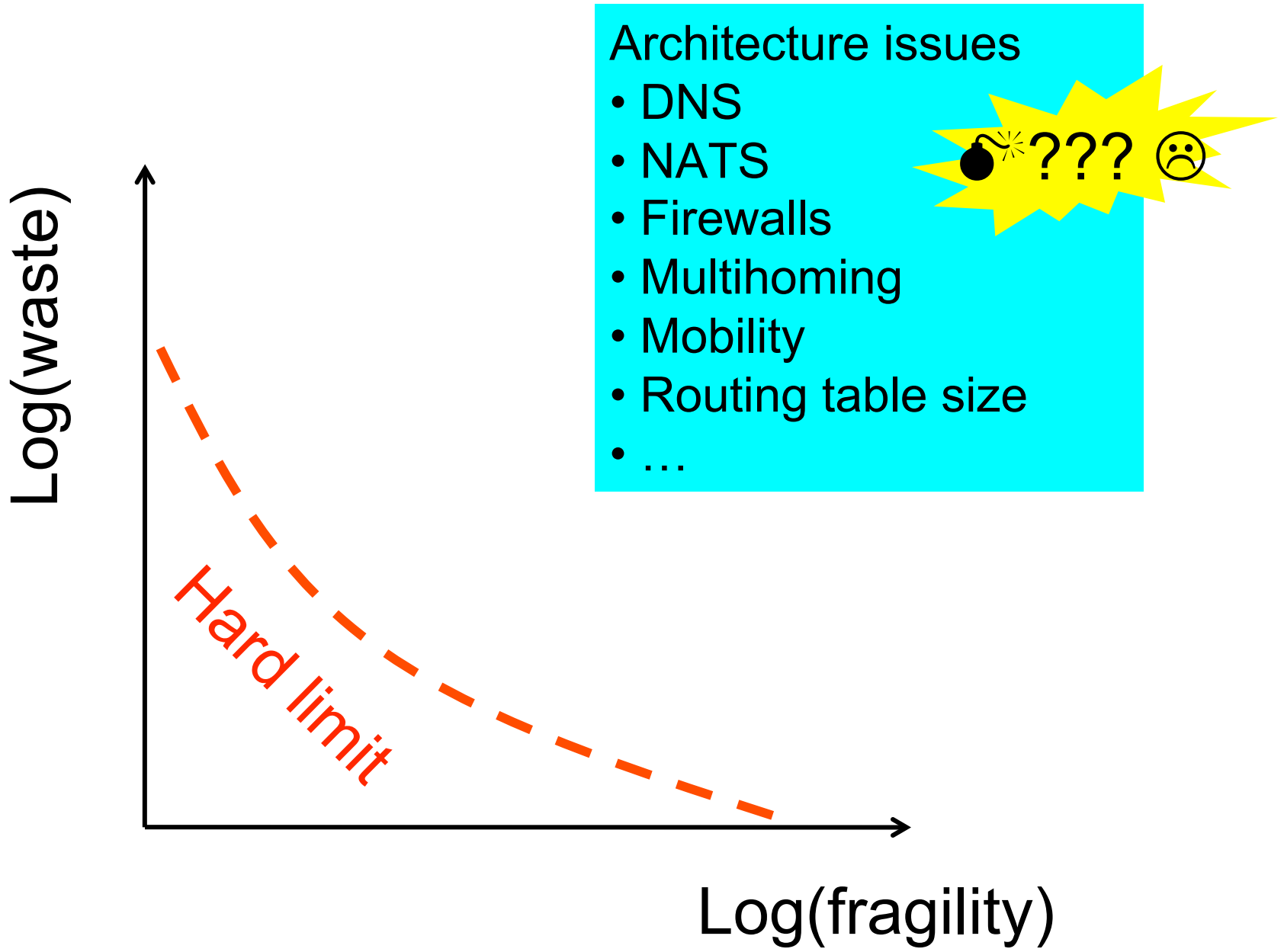
- Local, greedy routing using simple norms and “virtual coordinates” is globally optimal
- Large and growing literature on how to do this systematically

Add a virtual dimension

Other insecurities in TCP/IP

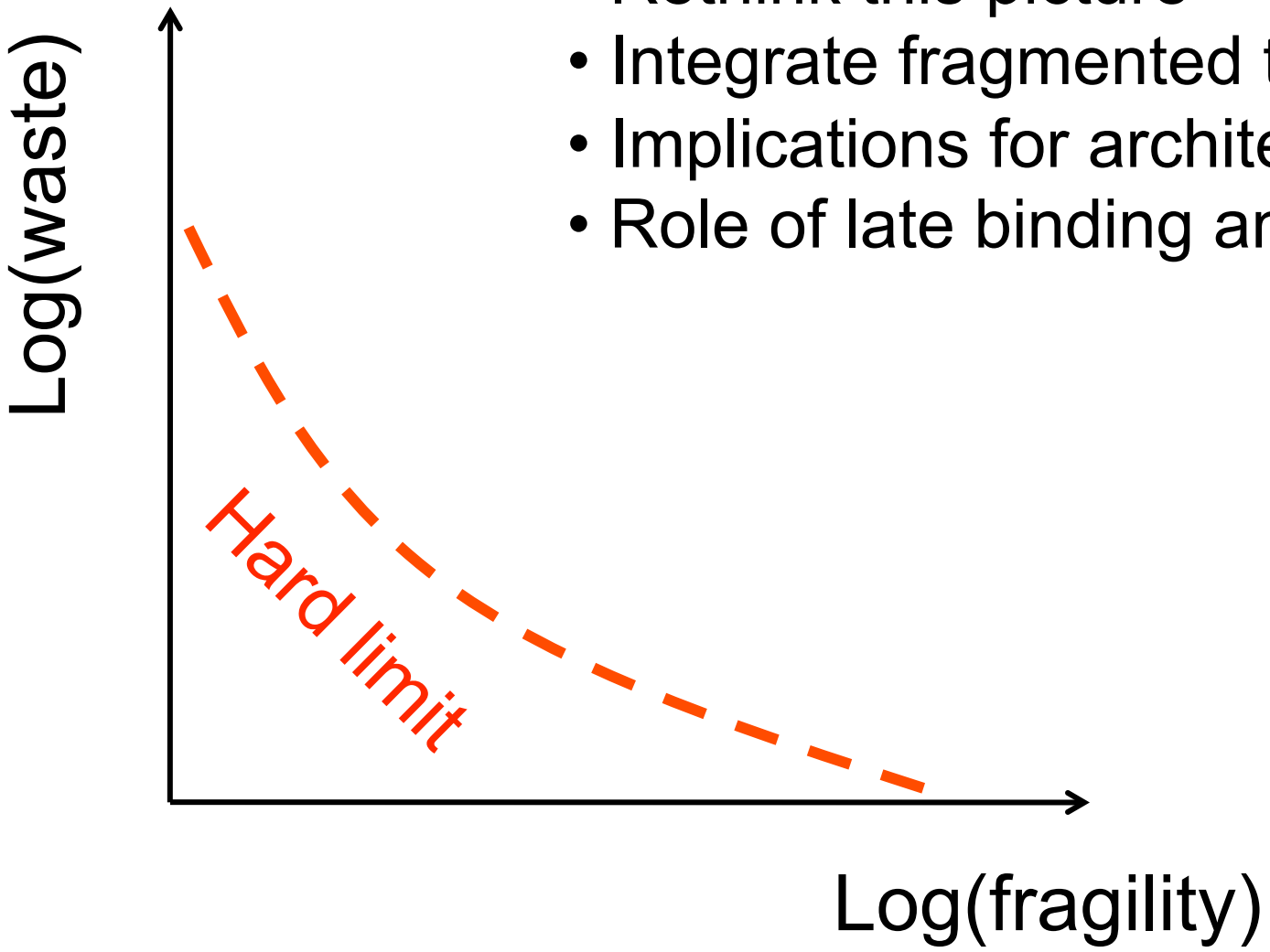
- port-scanning attacks
- connection-opening attacks
- data-transfer attacks
- Etc etc

These are hard to fix in existing architecture


Architecture issues
• DNS
• NATS
• Firewalls
• Multihoming
• Mobility
• Routing table size
• …
???
Log(waste)
Hard limit
Log(fragility)

Fundamentals:

- Rethink this picture
- Integrate fragmented theories
- Implications for architecture
- Role of late binding and ctrl

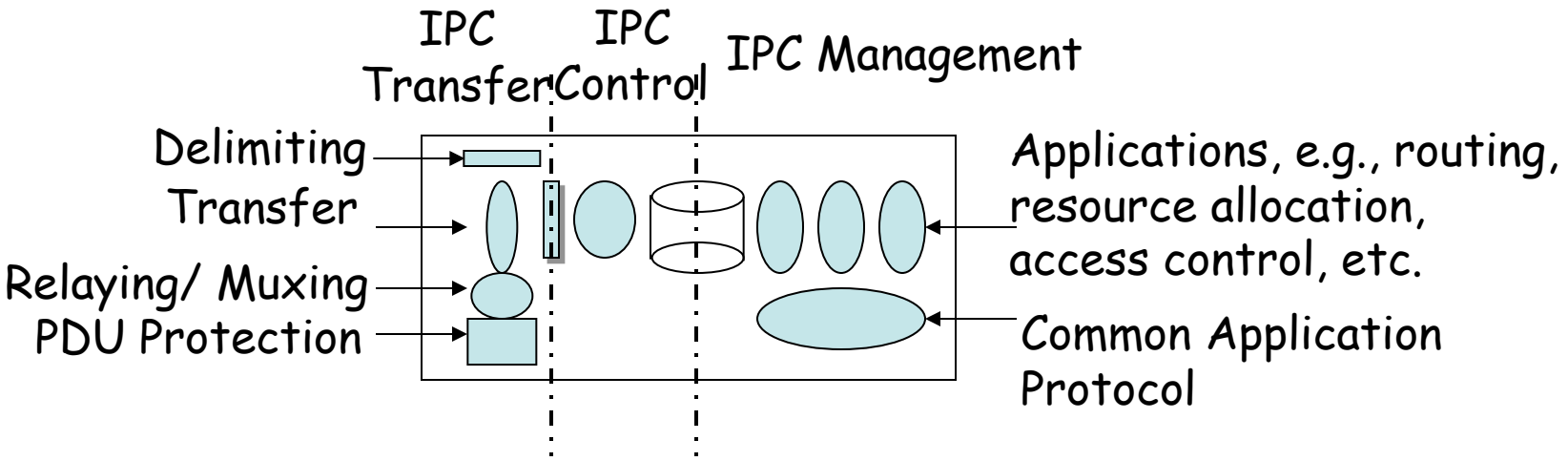


Next steps?
Or near term branch points

- More deeply into OS and PNA
- Rethink a TCP/IP equivalent assuming the OS/ PNA is done by someone else
  - Do TCP/IP as integrated DIF layer
  - Rethink fundamentals of flow/error control and routing
  - Role of information theory methods?
- Rethink a more wireless end system replacement for IP (G4 wireless)…
- What blend of PNA plus control theory would be the right IT infrastructure for smartgrid?

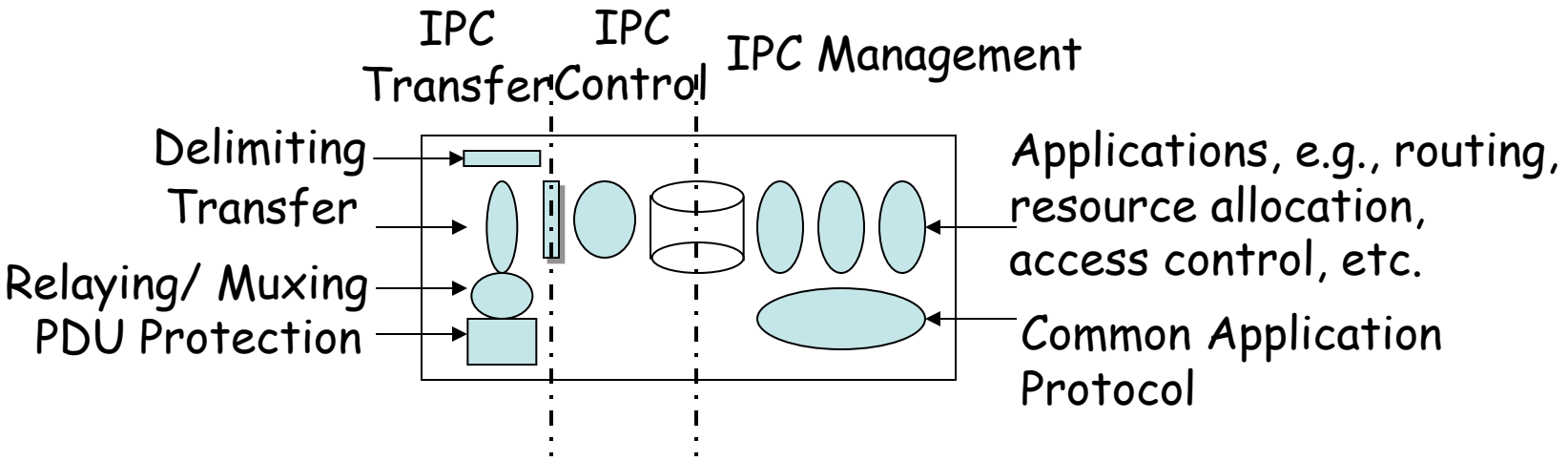
- More deeply into OS and PNA

Start with this picture from PNA



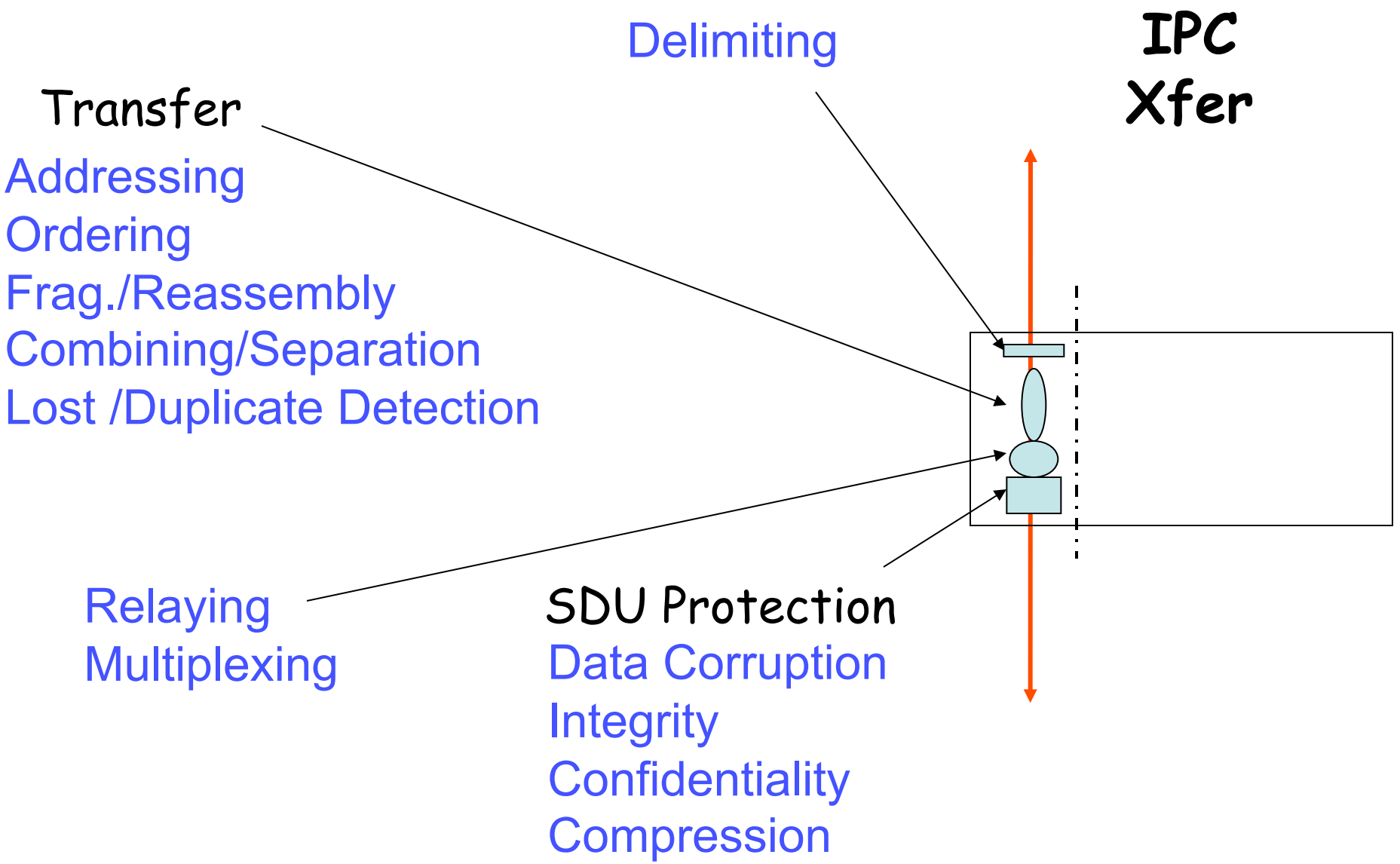
- More deeply into OS and PNA

Start with this picture from PNA

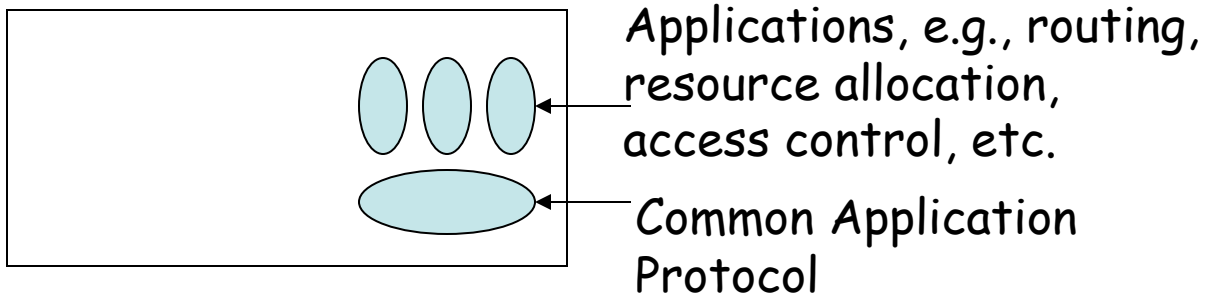


And categorize these

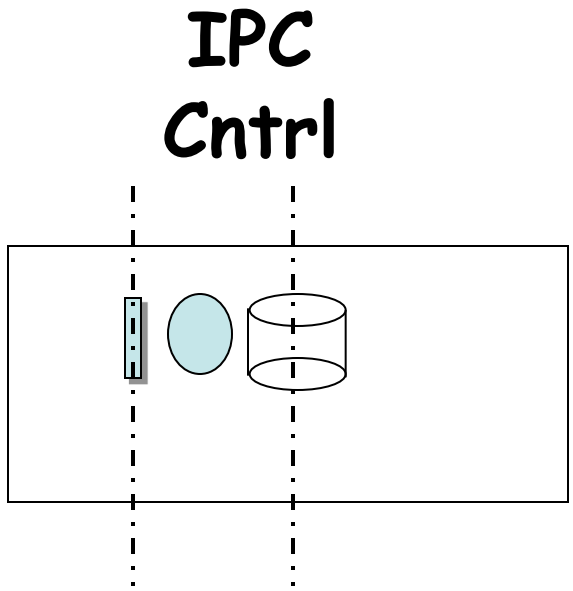
- Delimiting
- Initial State Synch
- Policy Selection
- Addressing
- Flow/Connection Identifier
- Relaying
- Multiplexing
- Ordering
- Frag./Reassembly
- Combining/Separation
- Data Corruption
- Lost /Duplicate Detection
- Flow Control
- Forward Error Cor.
- Ack/Retran Control
- Compression
- Authentication
- Access Control
- Integrity
- Confidentiality
- Nonrepudiation
- Activity


Delimiting
IPC
Xfer
Transfer
Addressing
Ordering
Frag./Reassembly
Combining/Separation
Lost /Duplicate Detection
Relaying
Multiplexing
SDU Protection
Data Corruption
Integrity
Confidentiality
Compression

IPC Mgmt

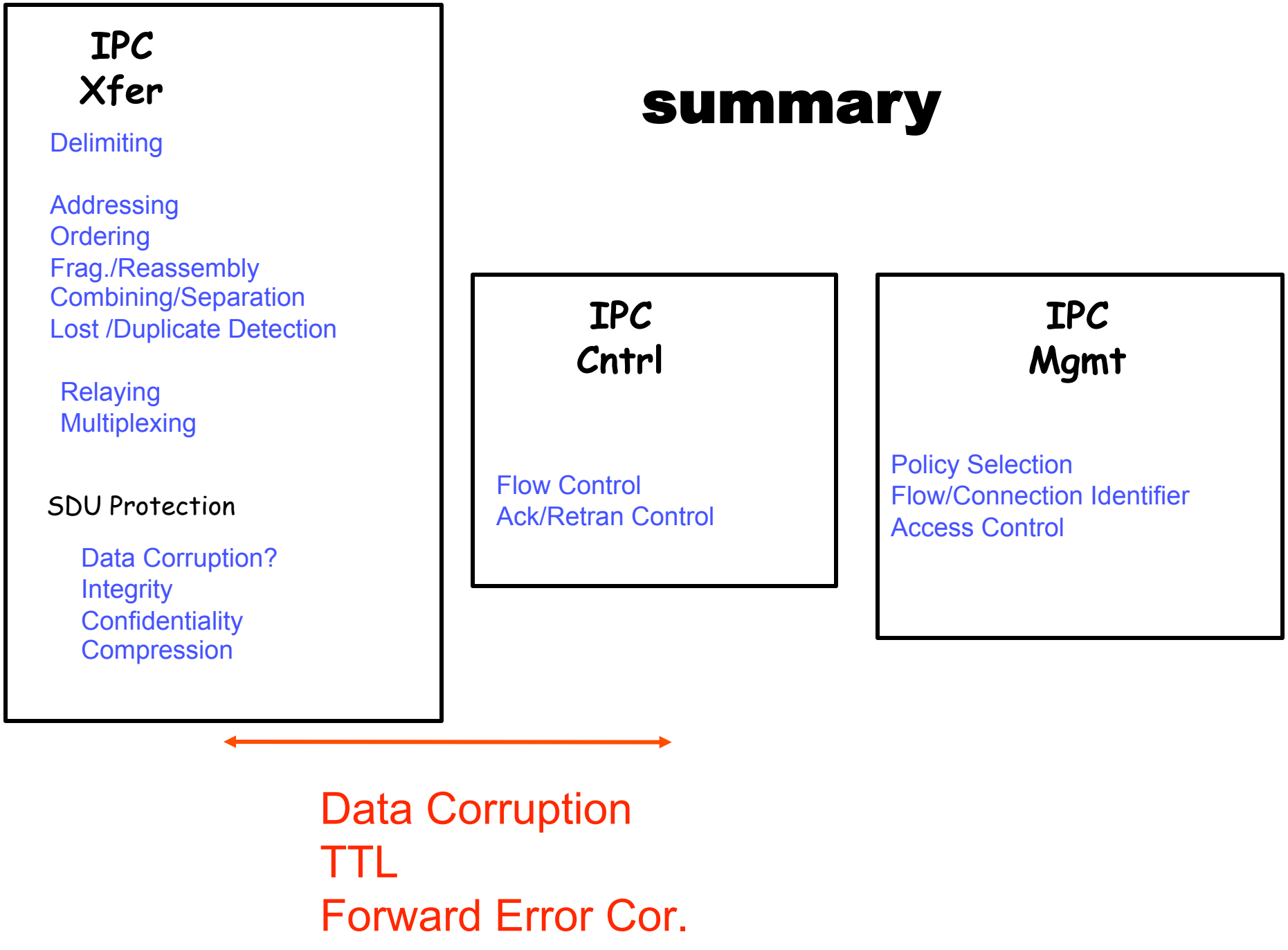


Routing
Policy Selection
Flow/Connection Identifier
Access Control



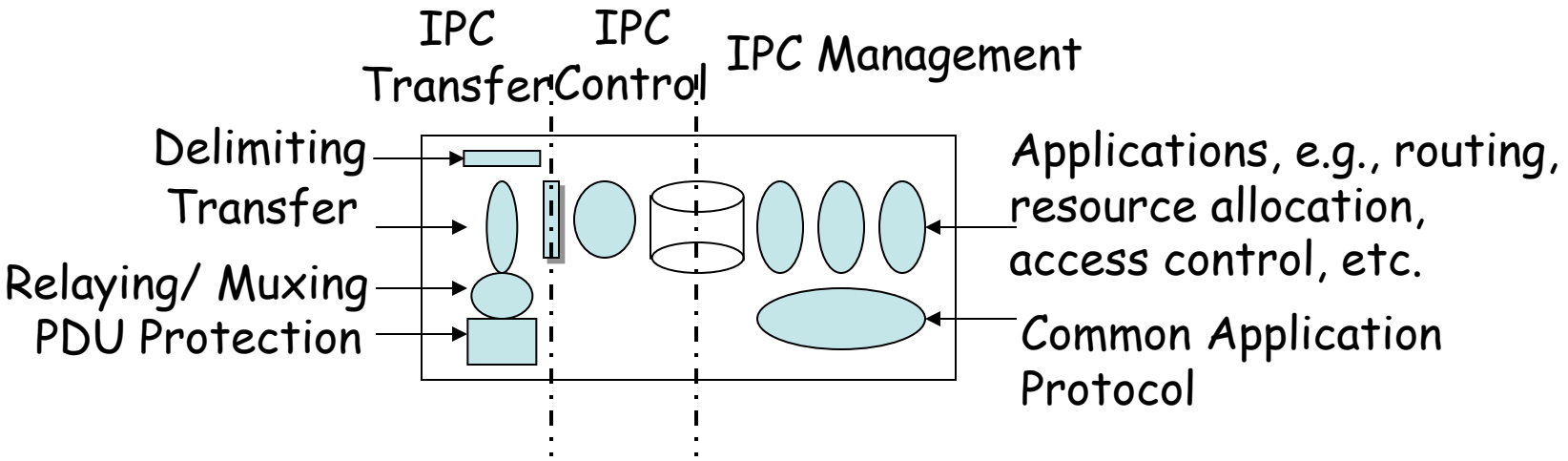
Flow Control
Ack/Retran Control

summary


IPC
Xfer
Delimiting
Addressing
Ordering
Frag./Reassembly
Combining/Separation
Lost /Duplicate Detection
Relaying
Multiplexing
SDU Protection
Data Corruption?
Integrity
Confidentiality
Compression
IPC
Cntrl
Flow Control
Ack/Retran Control
IPC
Mgmt
Policy Selection
Flow/Connection Identifier
Access Control
Data Corruption
TTL
Forward Error Cor.

• More deeply into OS and PNA

- Lots more in Day's book
- Many details have been started
- ISI/USC (Touch) has a similar approach, get them to tell us about it
- Collaborate so we can focus on ctrl/mgmt



Rethink a TCP/IP equivalent

- “Clean slate” but with basic physical and app layers
- Assuming the OS/PNA parts are done by someone else but allow better control
- Do TCP/IP as integrated DIF layer
  - Split it the “other way” or “right way”
  - What are the alternatives
  - Rethink flow/error control and routing
  - Rethink naming and addressing
- Role of information theory methods?

Rethink a more wireless end system replacement for IP (G4 wireless)…

- There is a proposal out there using IP for G4
- What alternatives does PNA offer?
- Focus on ctrl and mgmt

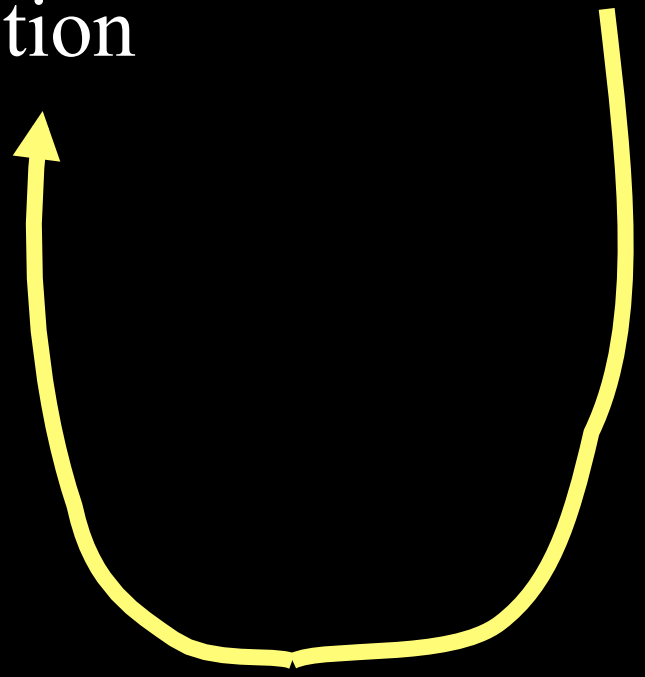
IT infrastructure for smartgrid

- What blend of PNA plus control theory would be the right IT infrastructure for smartgrid?
- Need real time, guarantees of QOS
- Need better security throughout
- Other issues?
- Coordinate with Steven's class

In the real (vs virtual) world

What matters:

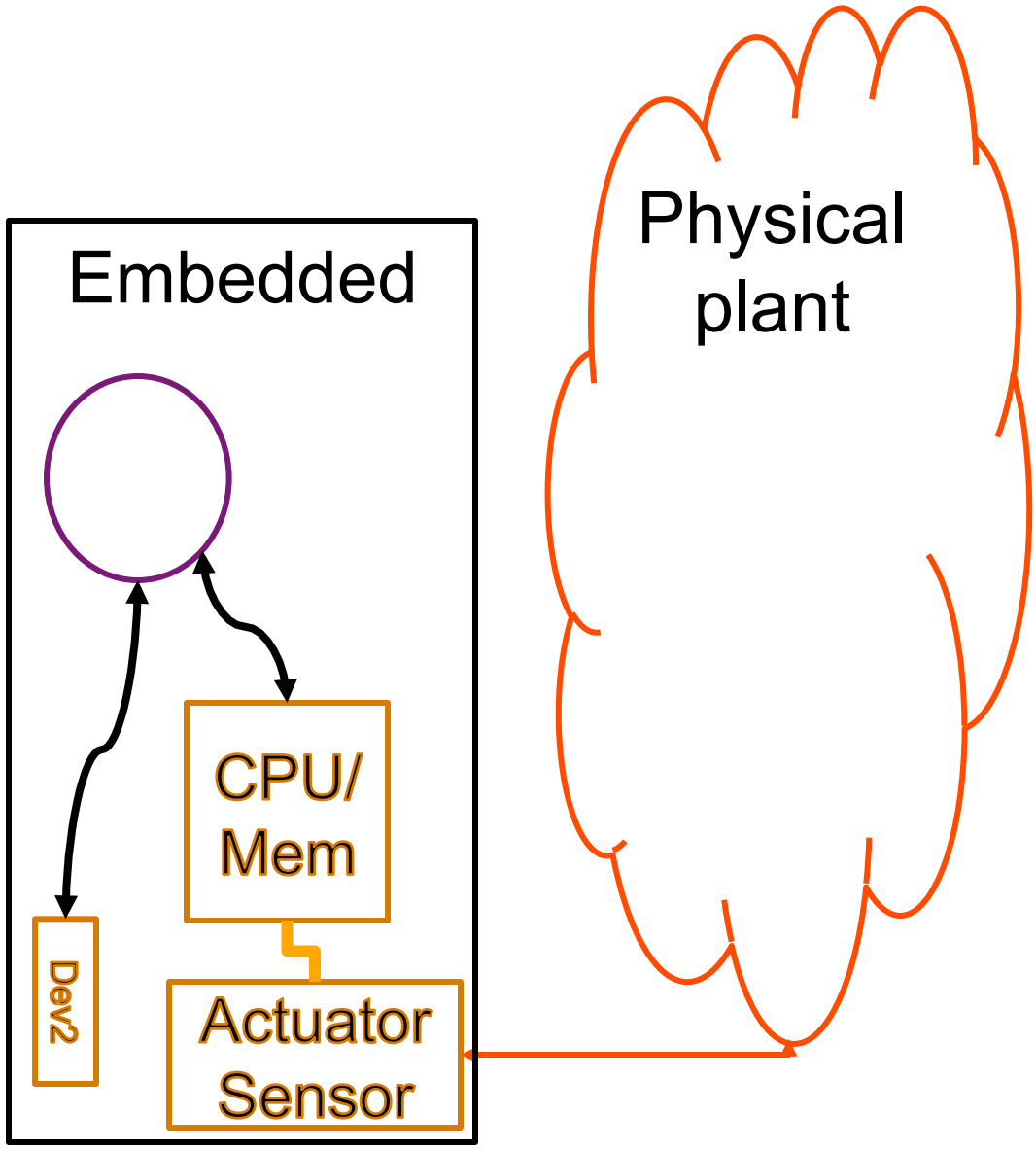
- Action



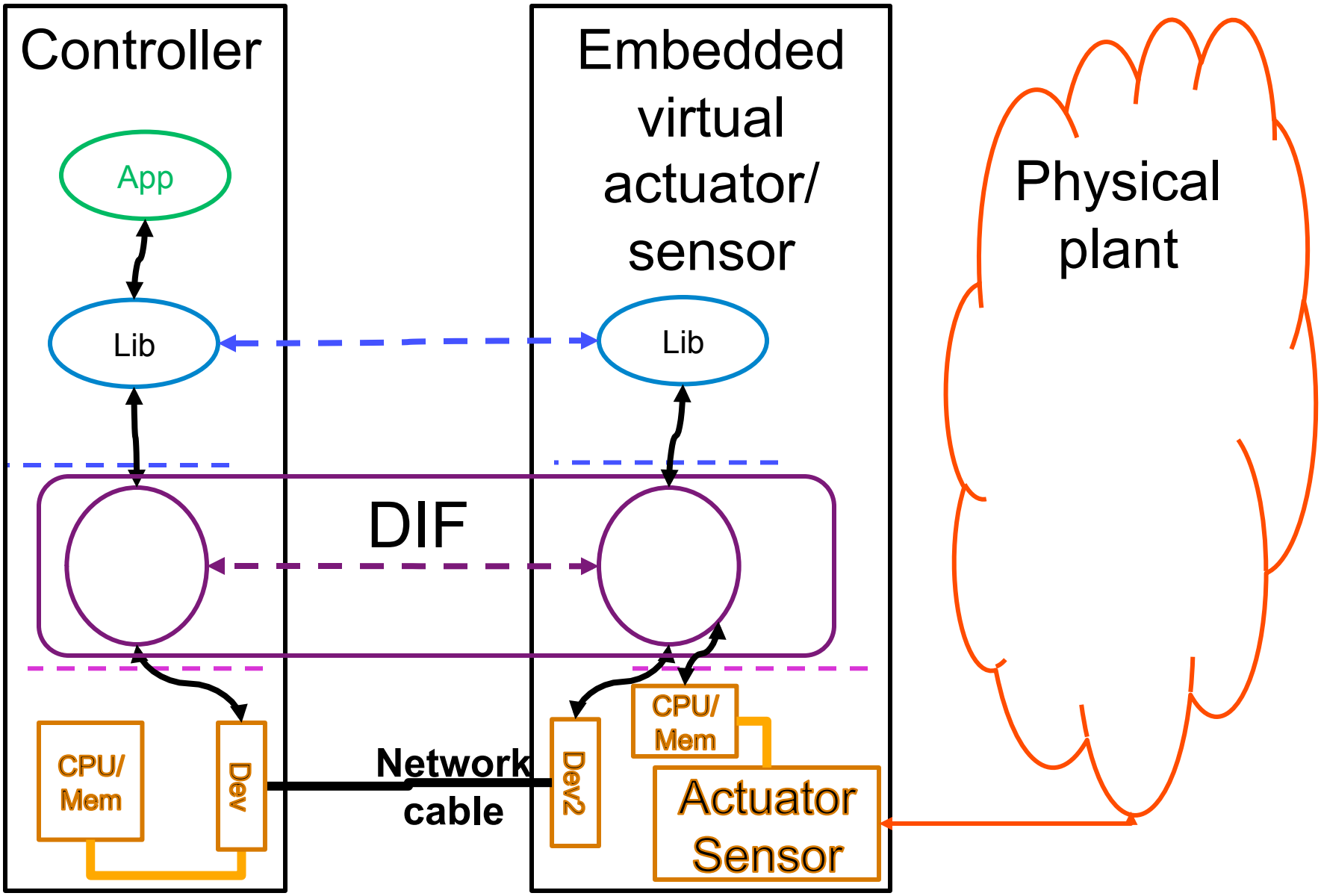
What doesn't:

- Data
- Information
- Computation
- Learning
- Decision
- …

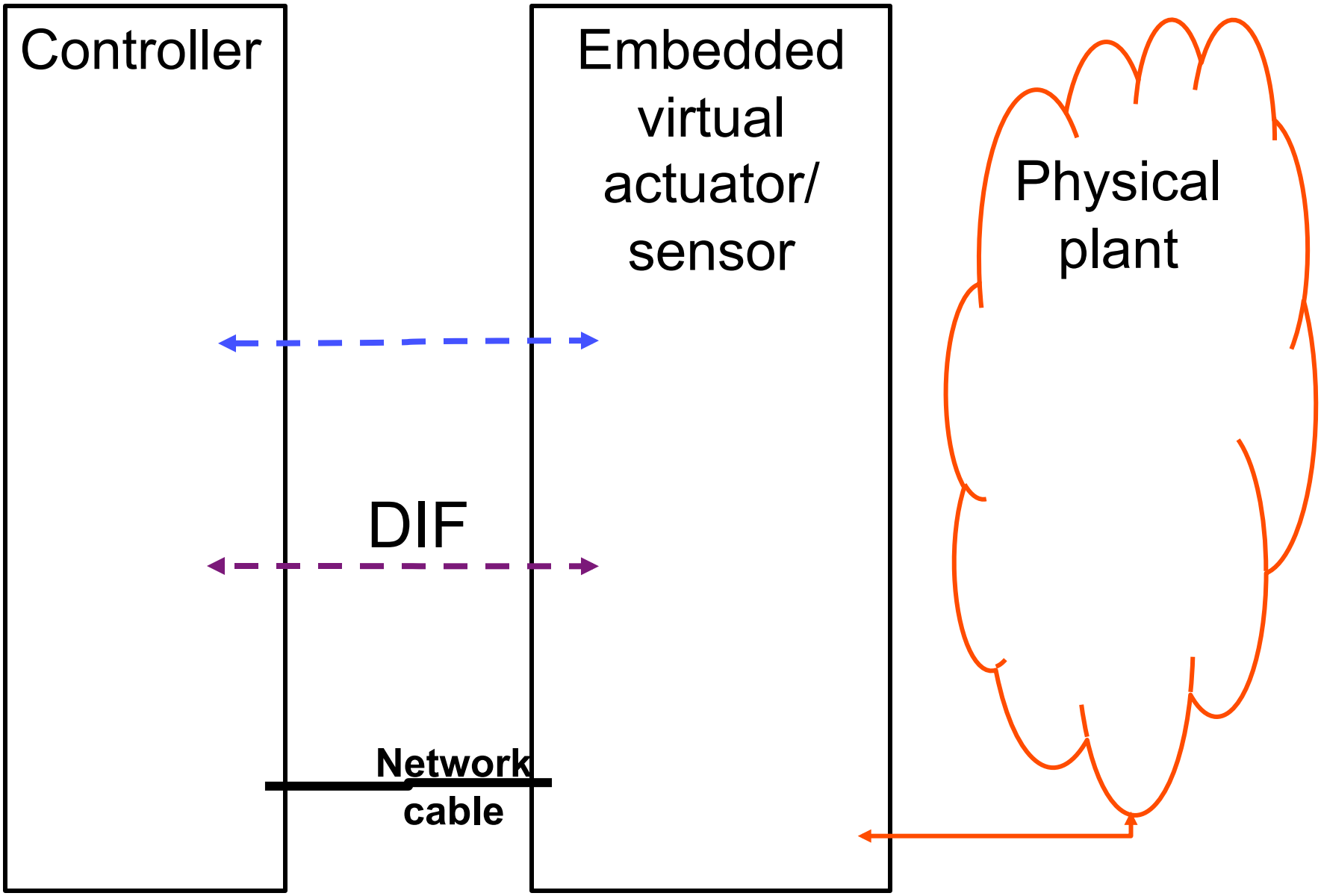
Embedded



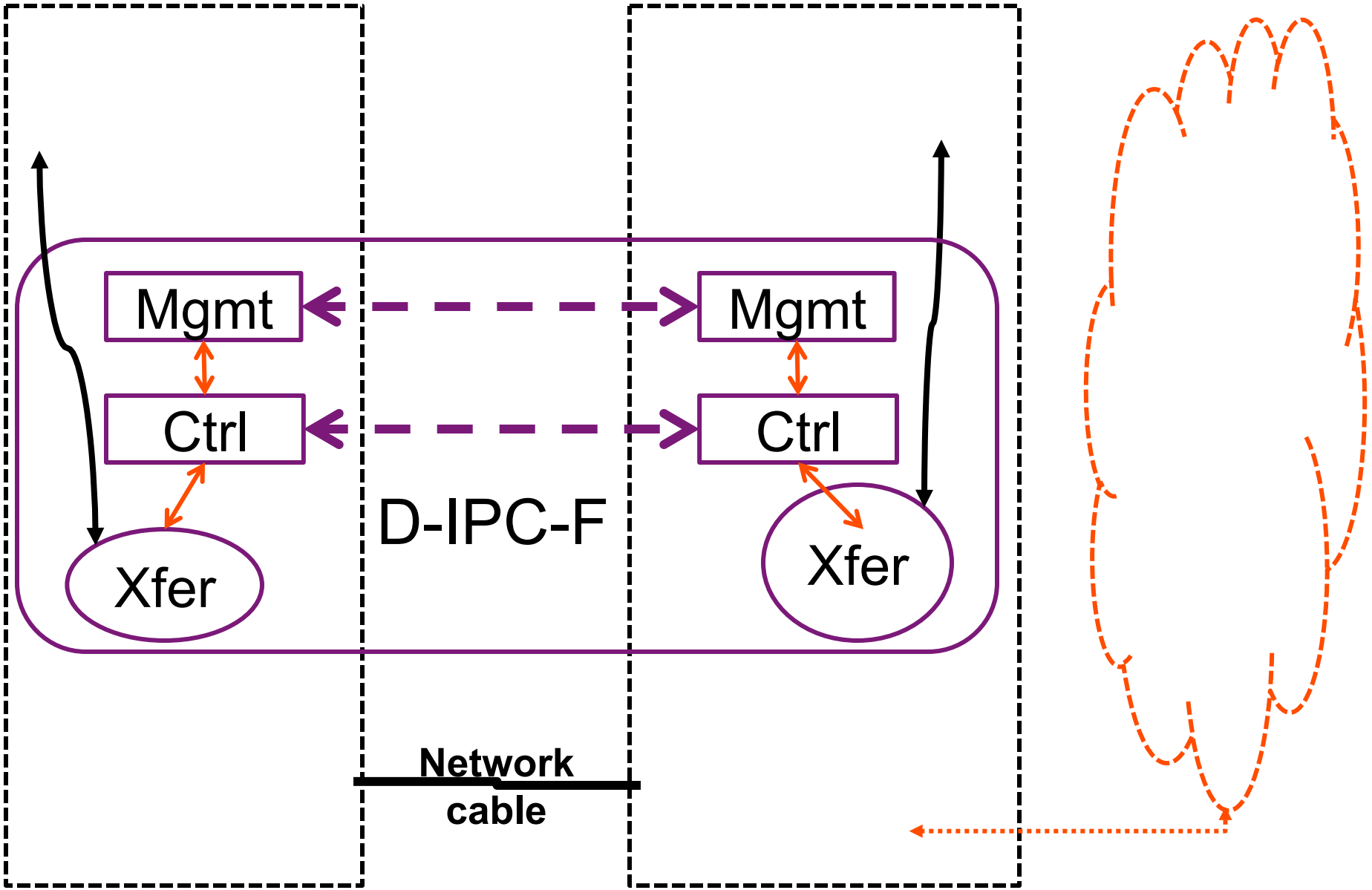
Networked embedded



Meta-layering of cyber-phys control

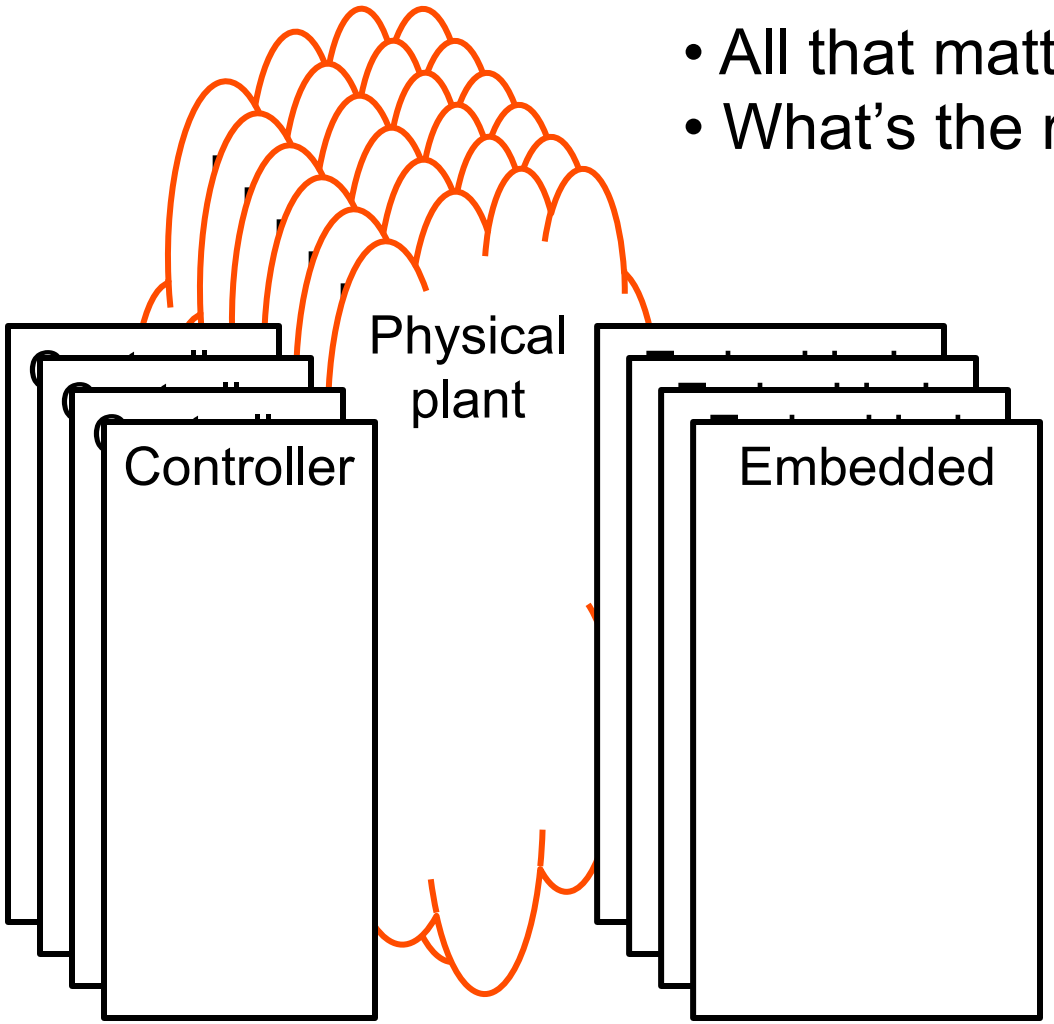


Micro-layering of D-IPC-F



Smartgrid and cyberphys

- Everything is networked.
- Flows of data and power.
- All that matters is ***action***.
- What's the right architecture?



Biology versus the Internet

Similarities

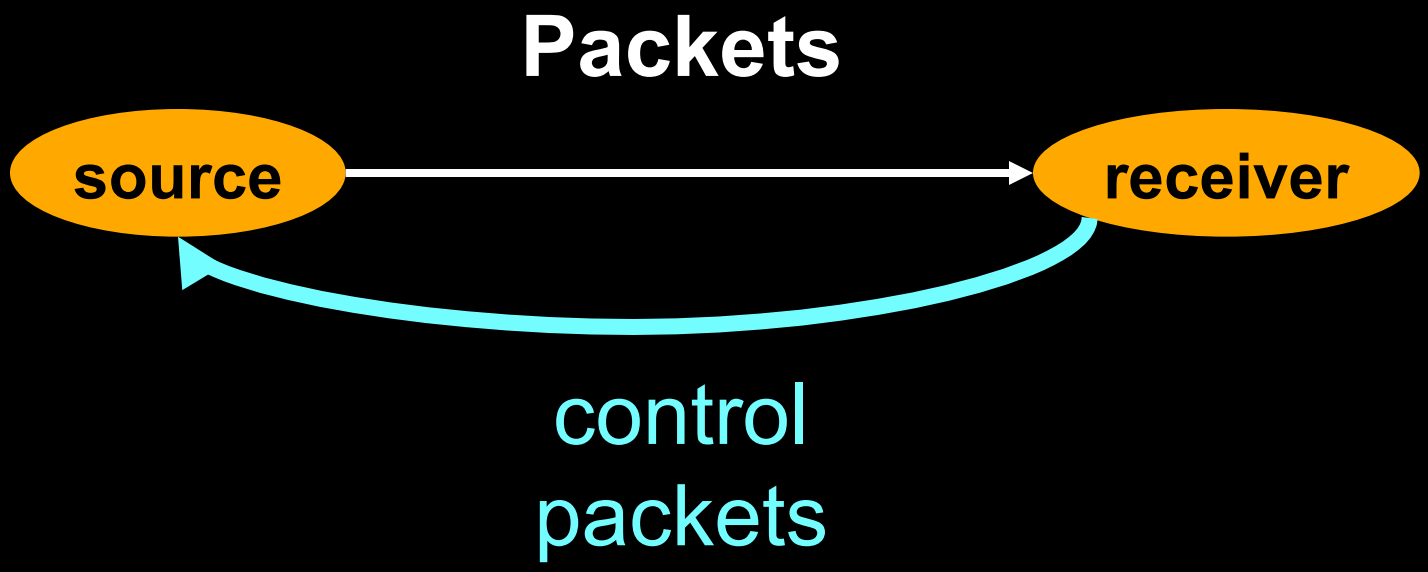
- Evolvable architecture
- **Robust yet fragile**
- **Constraints/deconstrain**
- **Layering, modularity**
- **Hourglass with bowties**
- Feedback
- Dynamics
- Distributed/decentralized
- ***Not*** scale-free, edge-of-chaos, self-organized criticality, etc

Differences

- Metabolism
- Materials and energy
- **Autocatalytic feedback**
- Feedback complexity
- Development and regeneration
- >4B years of evolution

Focus on bacterial biosphere

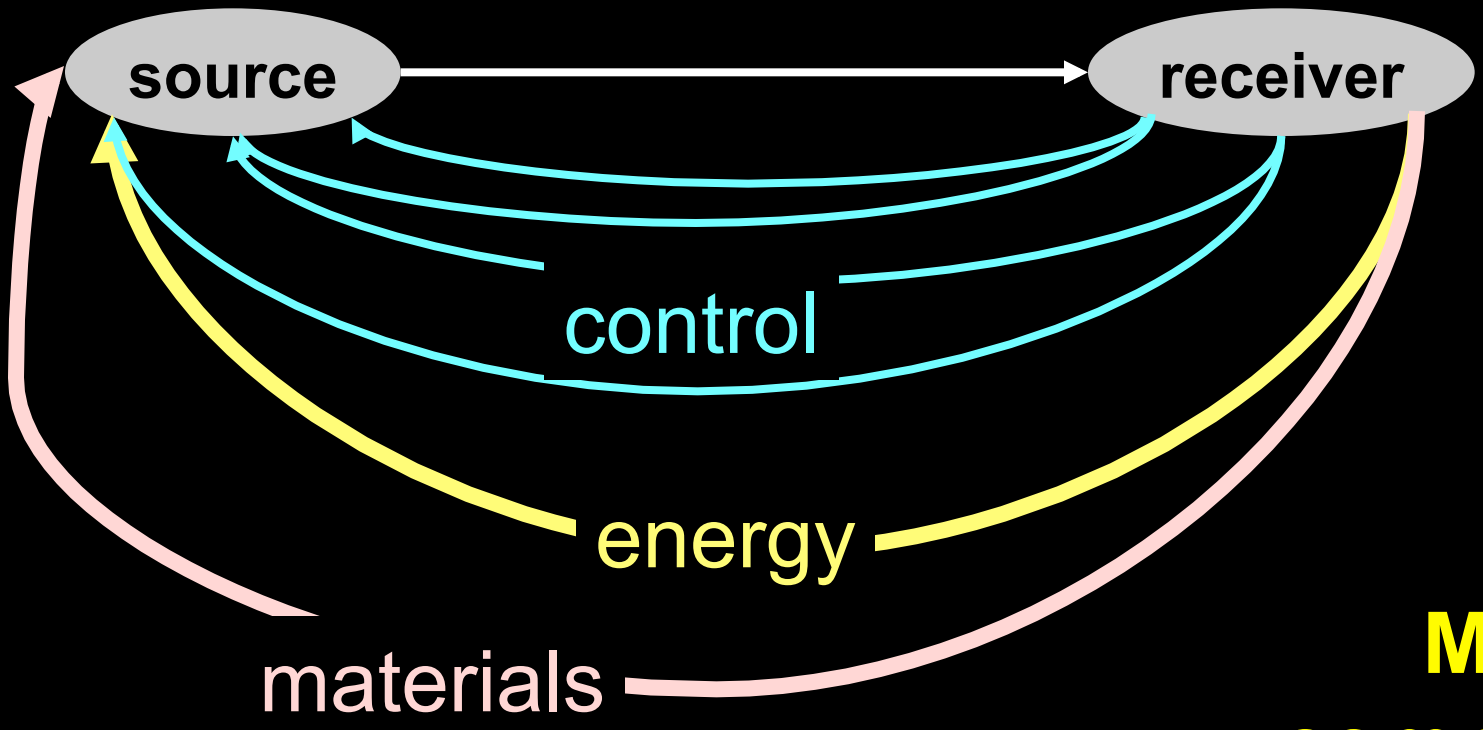
Control of the Internet



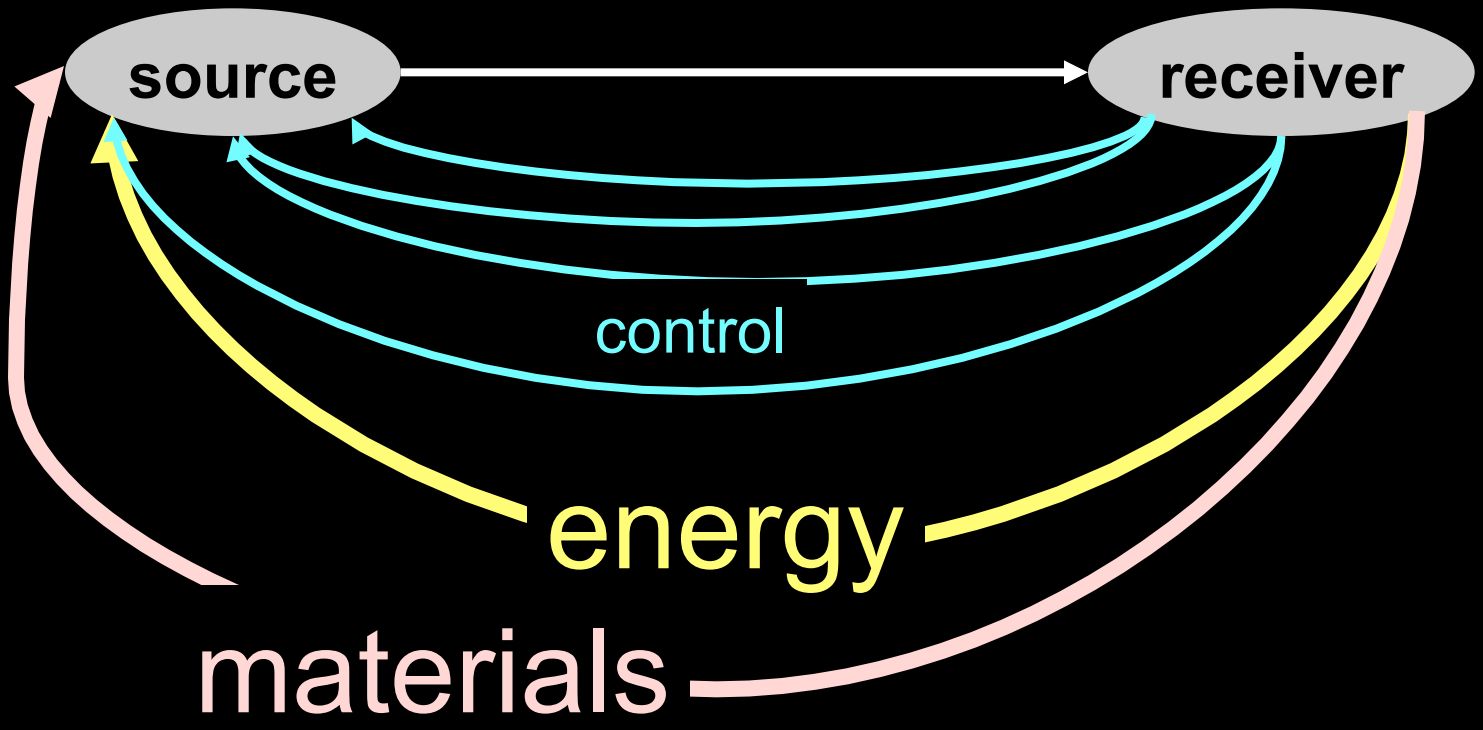
signaling
gene expression
metabolism
lineage

Biological pathways

signaling
gene expression
metabolism
lineage

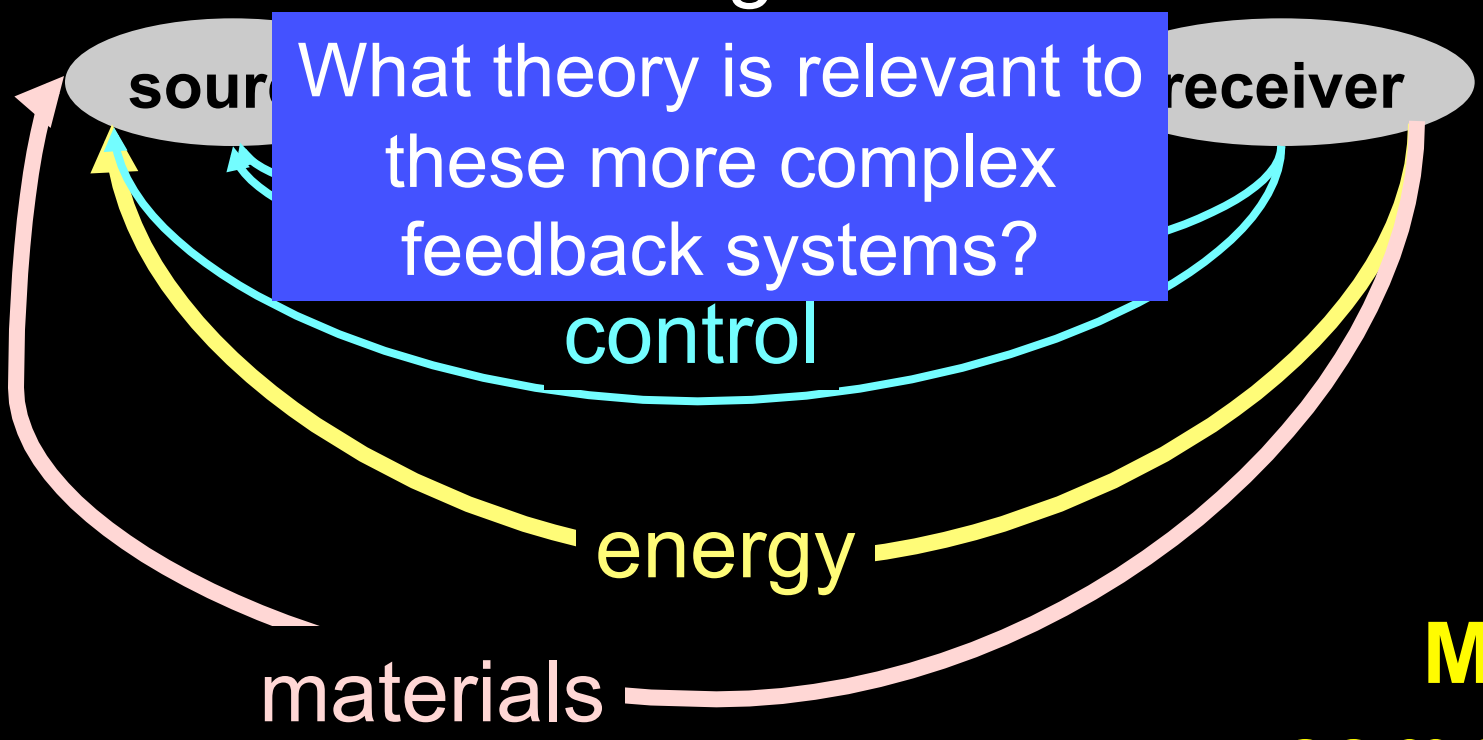


More complex feedback



Autocatalytic feedback

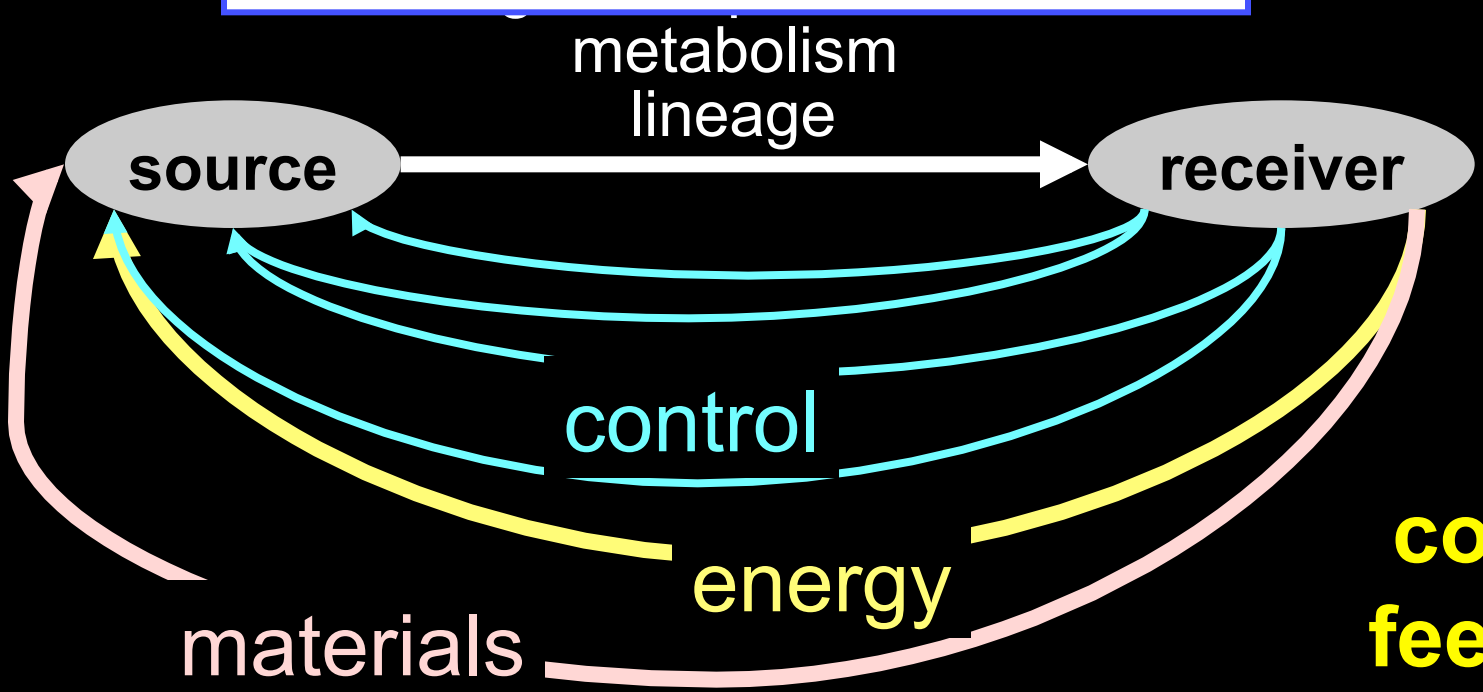
signaling
gene expression
metabolism
lineage



More complex feedback

What theory is relevant to these more complex feedback systems?

$$\frac{1}{\pi}\int_0^{\infty}\ln\left|S(j\omega)\right|\frac{z}{z^2+\omega^2}d\omega \geq \ln\left|\frac{z+p}{z-p}\right|$$



More complex feedback