Examples for *resources* concept of binding-like relations

Denis Buzdalov
Alexey Khoroshilov

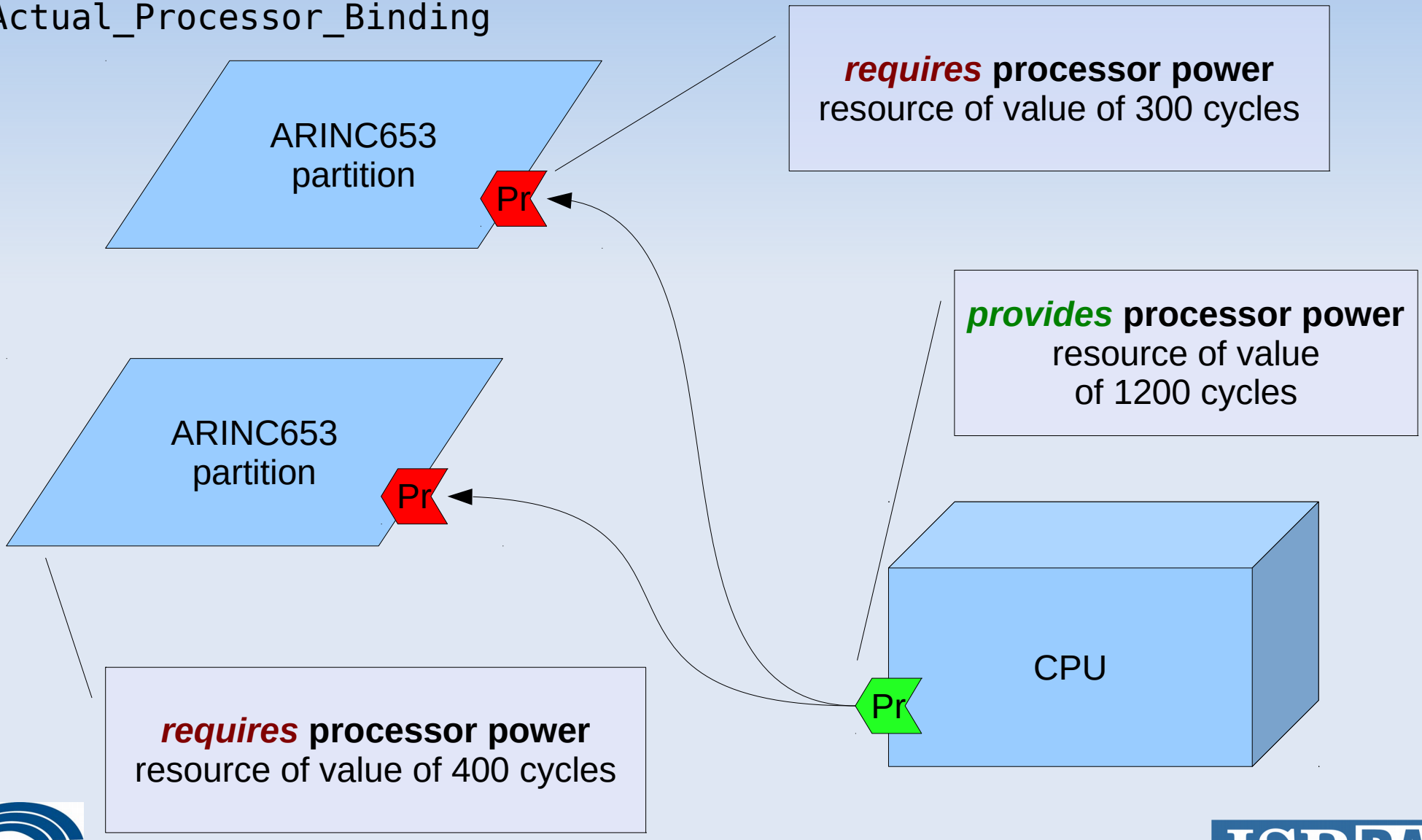
AADL meeting, fall 2016

Resource providing and receiving

- Relation between components
  - Directional: goes from a providing to a receiving point
  - Typed: different relation types have different semantics
  - Paremeterized
    - Receiving only a part of provided resource value
    - Limitations (incompatibility of samely typed resource points)
    - ...

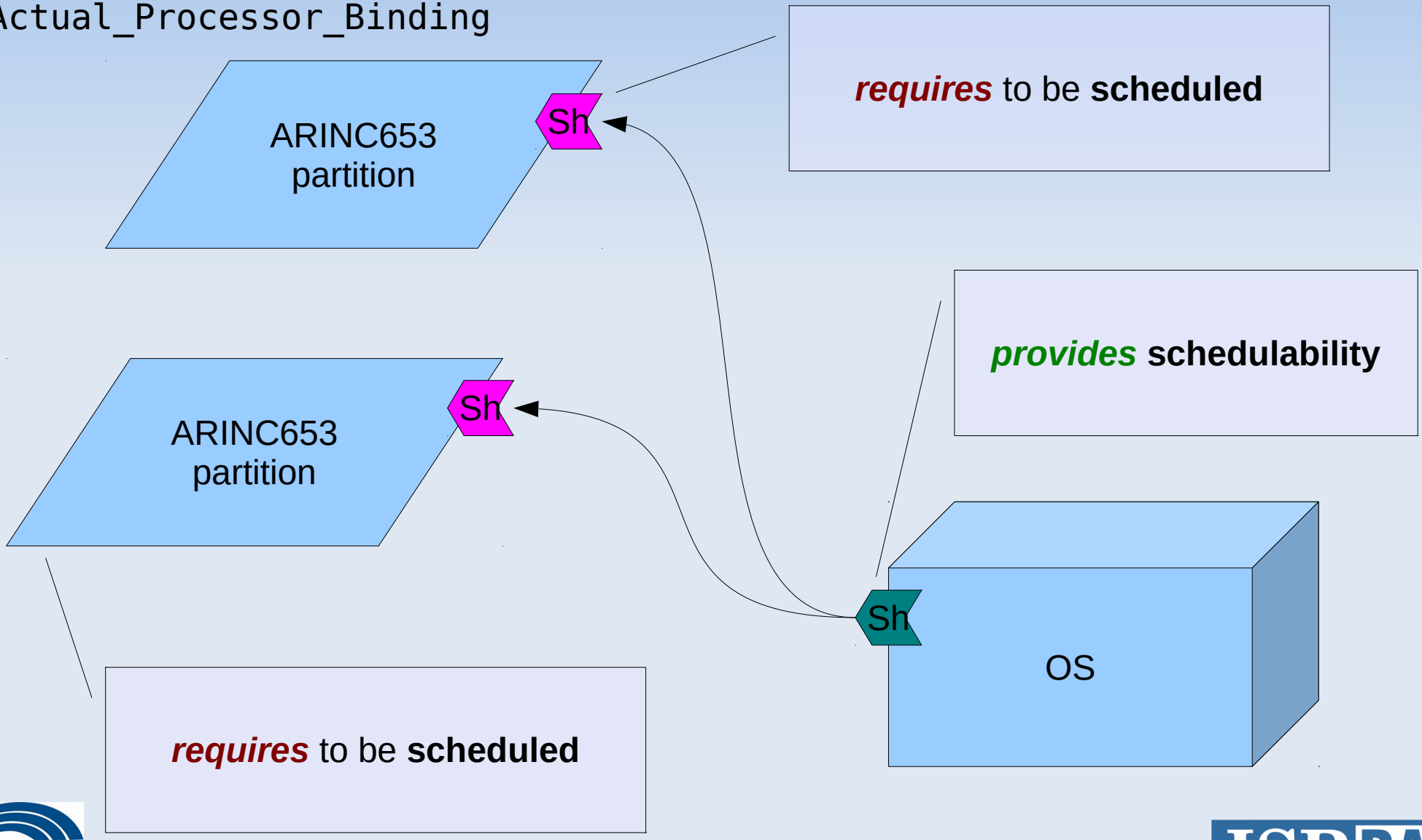
Resource type example: processor power

One side of the `Actual_Processor_Binding`



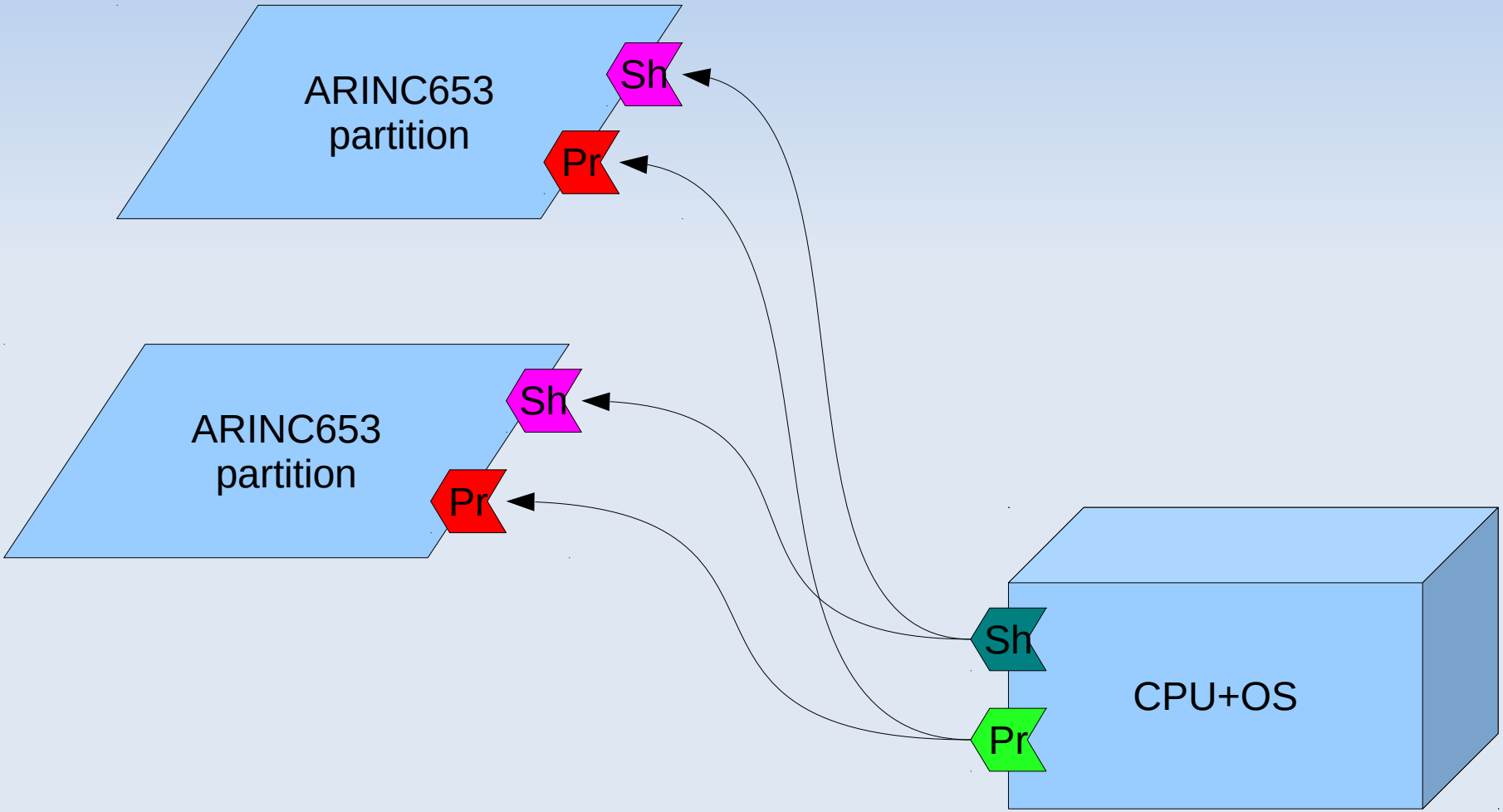
Resource type example: schedulability

Another side of the `Actual_Processor_Binding`



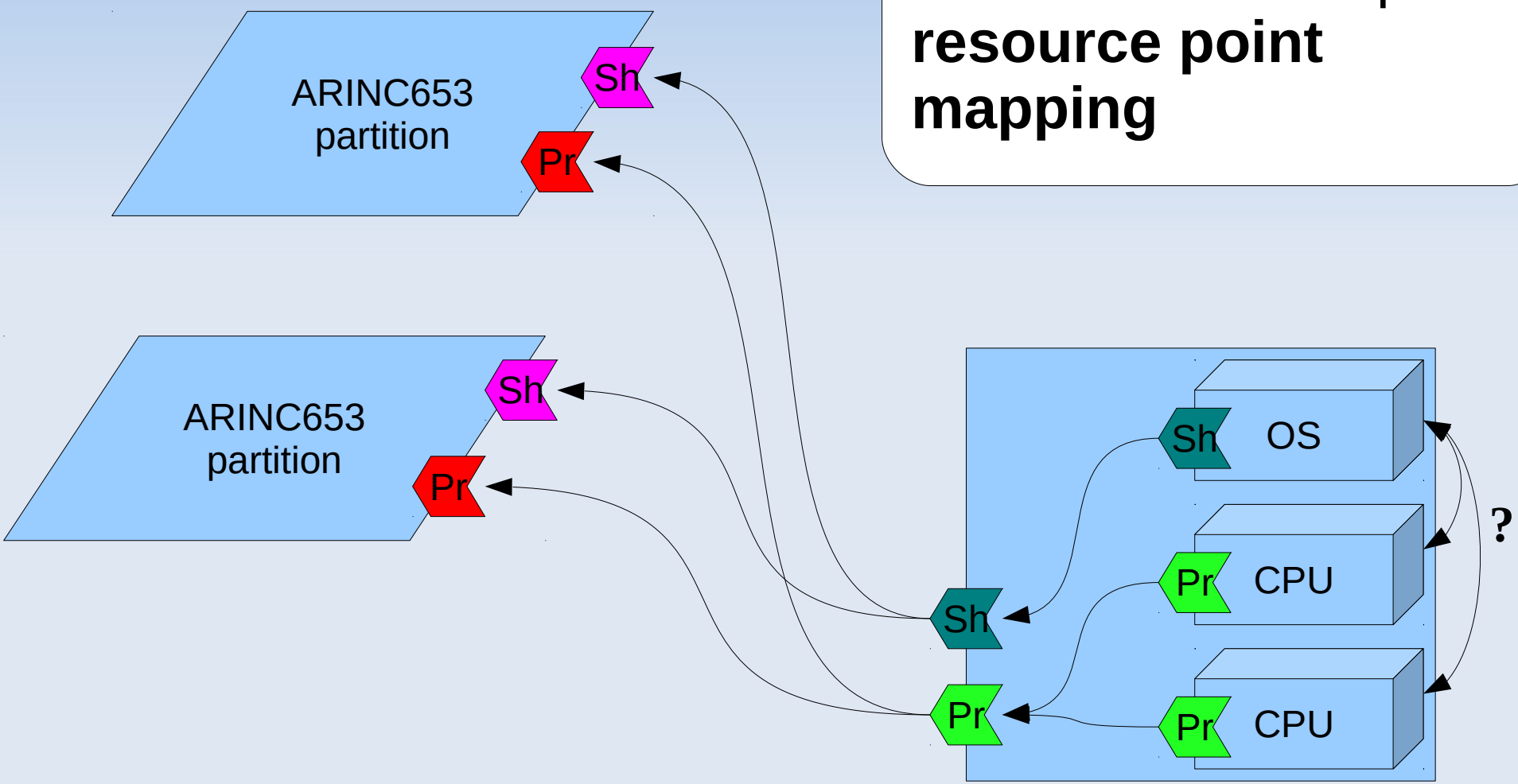
Resource type example: processor power & schedulability

Two sides of the `Actual_Processor_Binding`



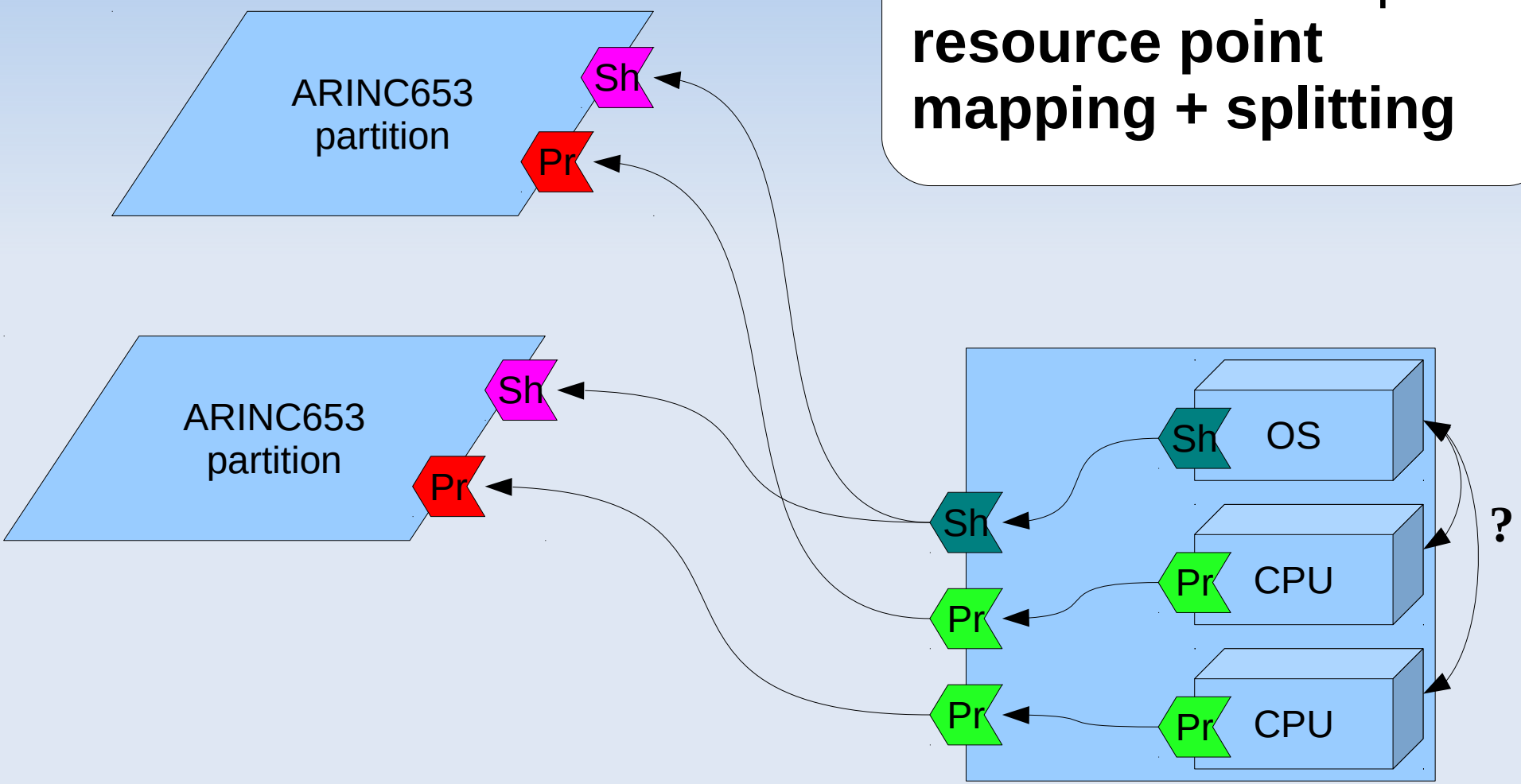
Resource type example: processor power & schedulability

Refinement example: **resource point mapping**



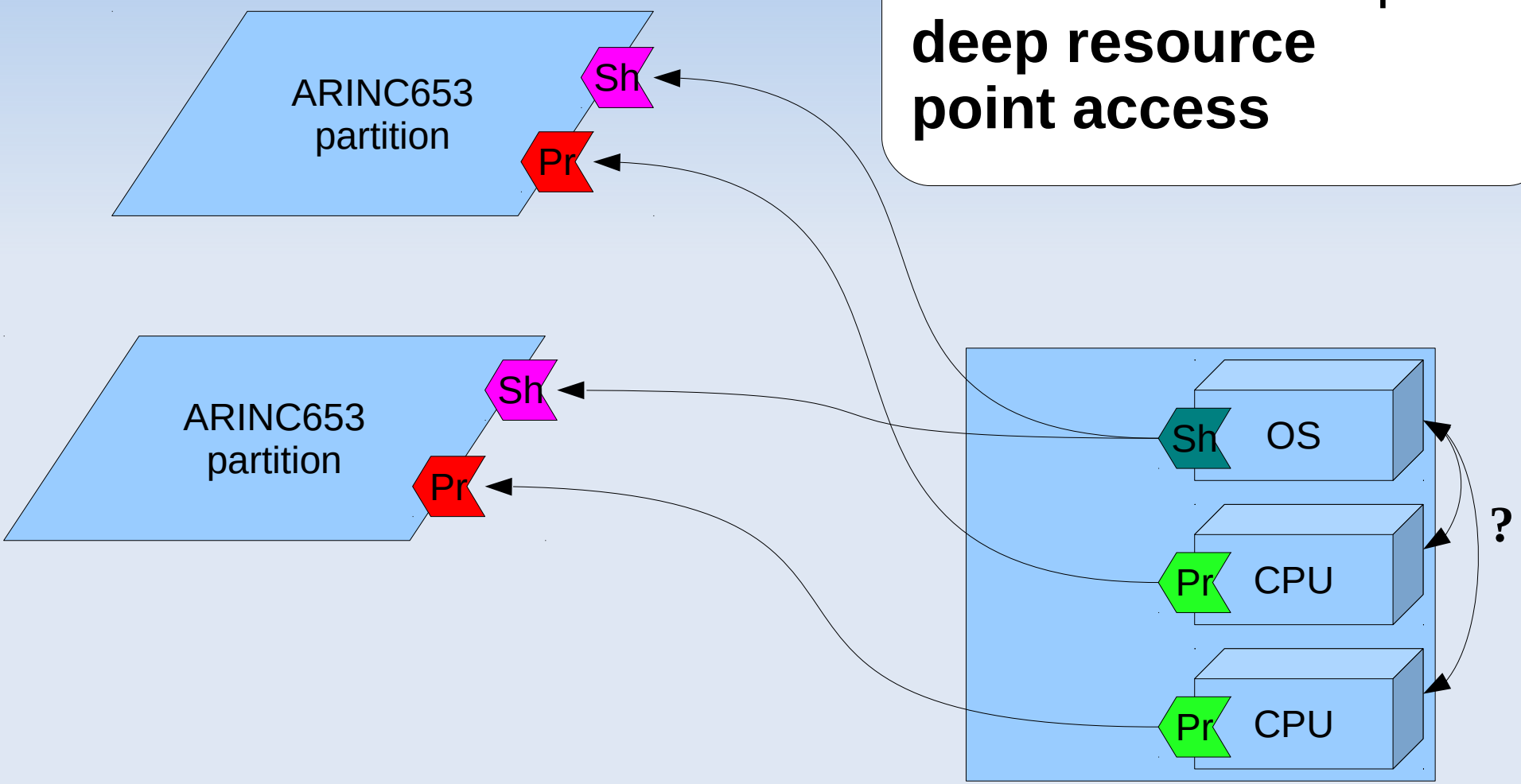
Resource type example: processor power & schedulability

Refinement example:
resource point mapping + splitting

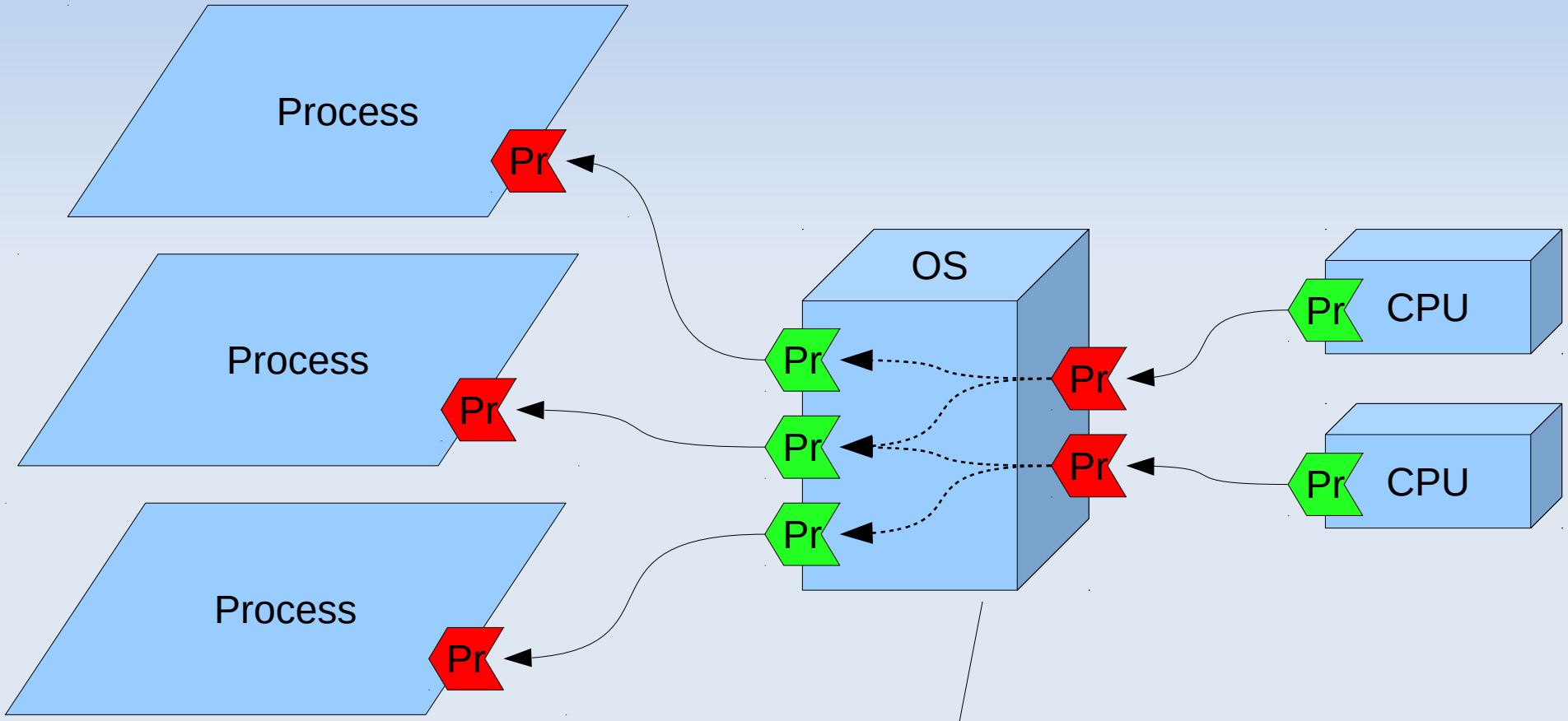


Resource type example: processor power & schedulability

Refinement example:
deep resource point access

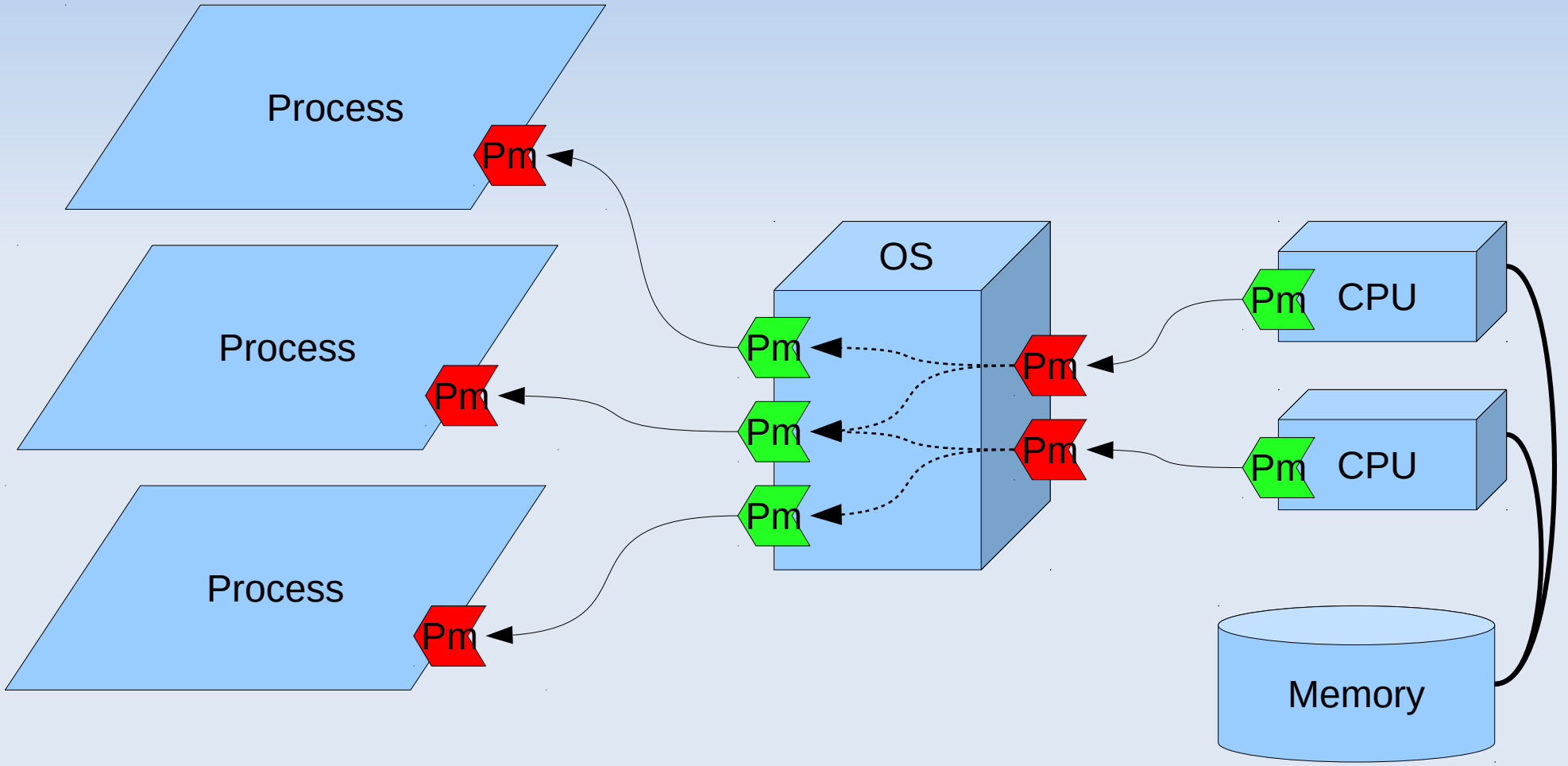


Resource type example: single processor power

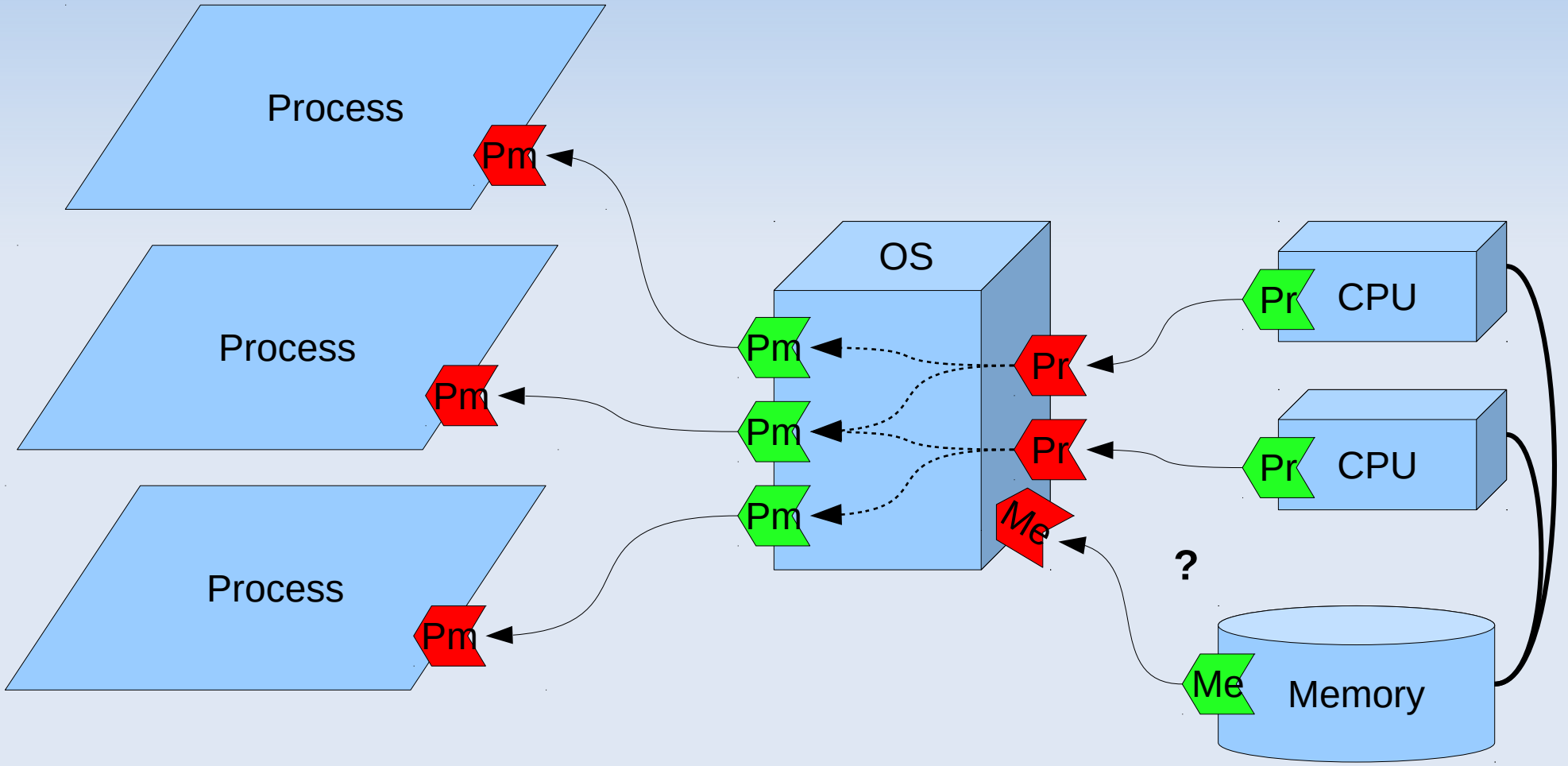


Operating system is considered as a **resource converter**

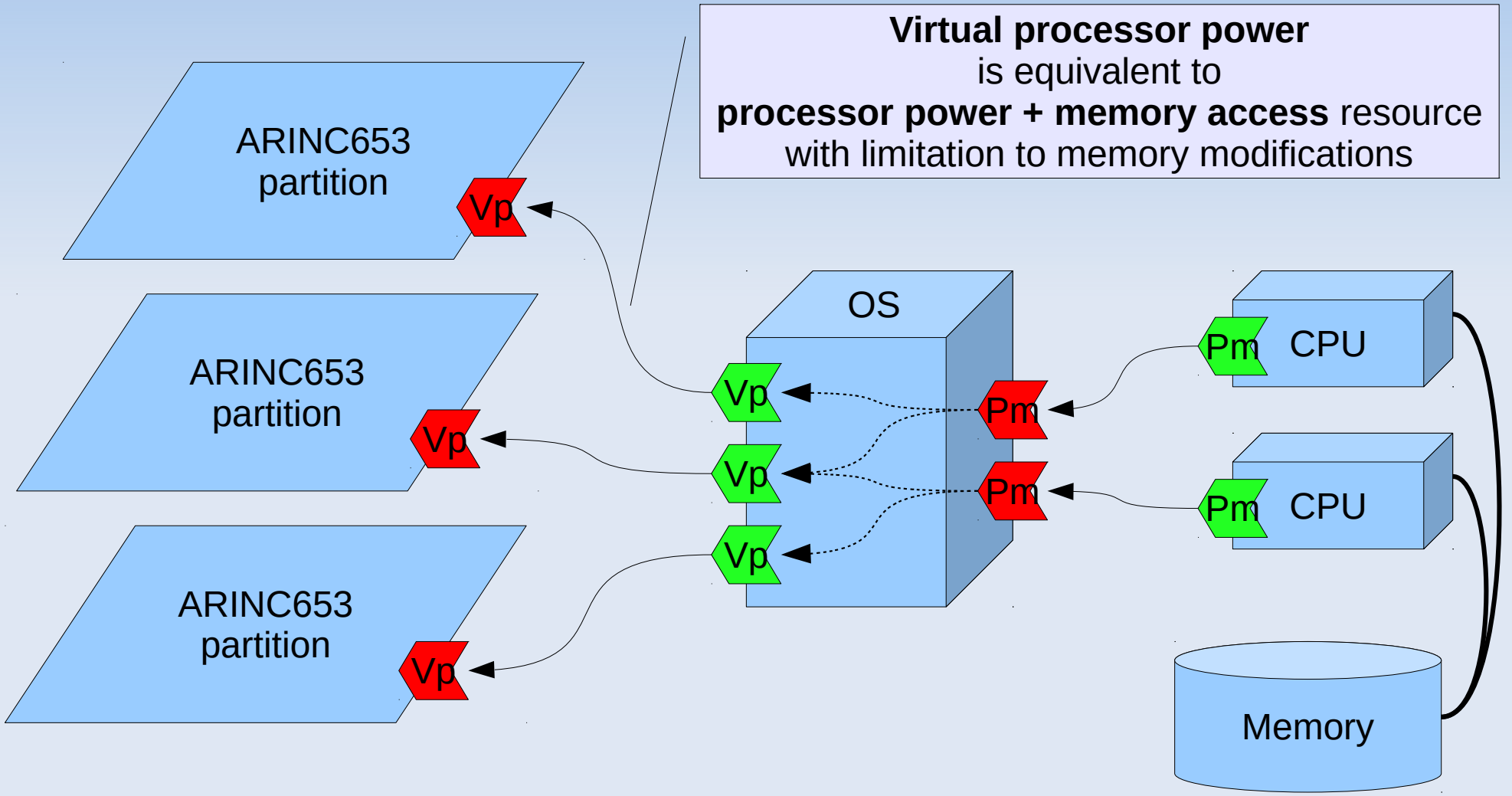
Resource type example: processor power + memory access



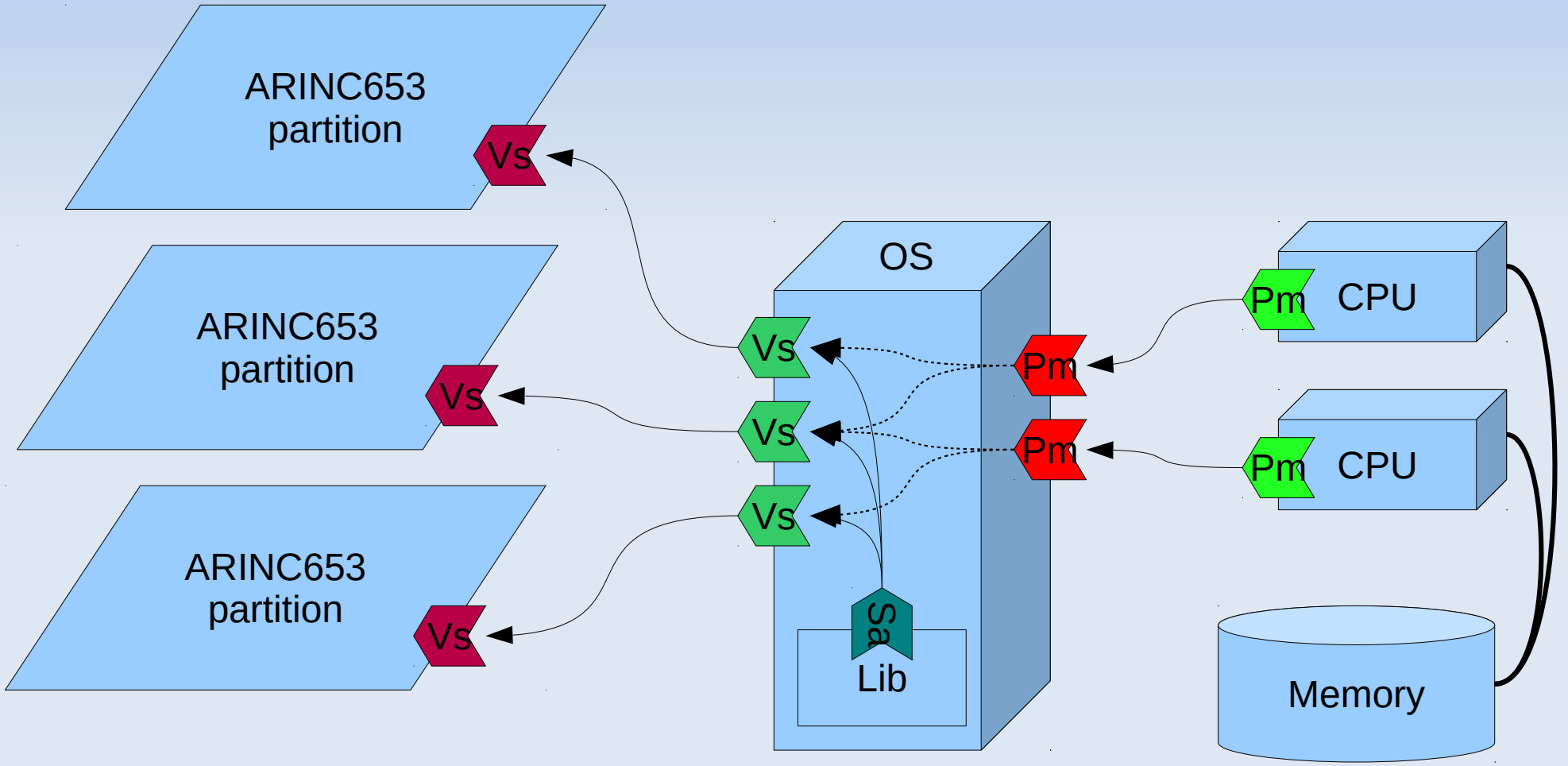
Resource type example: processor power & memory access



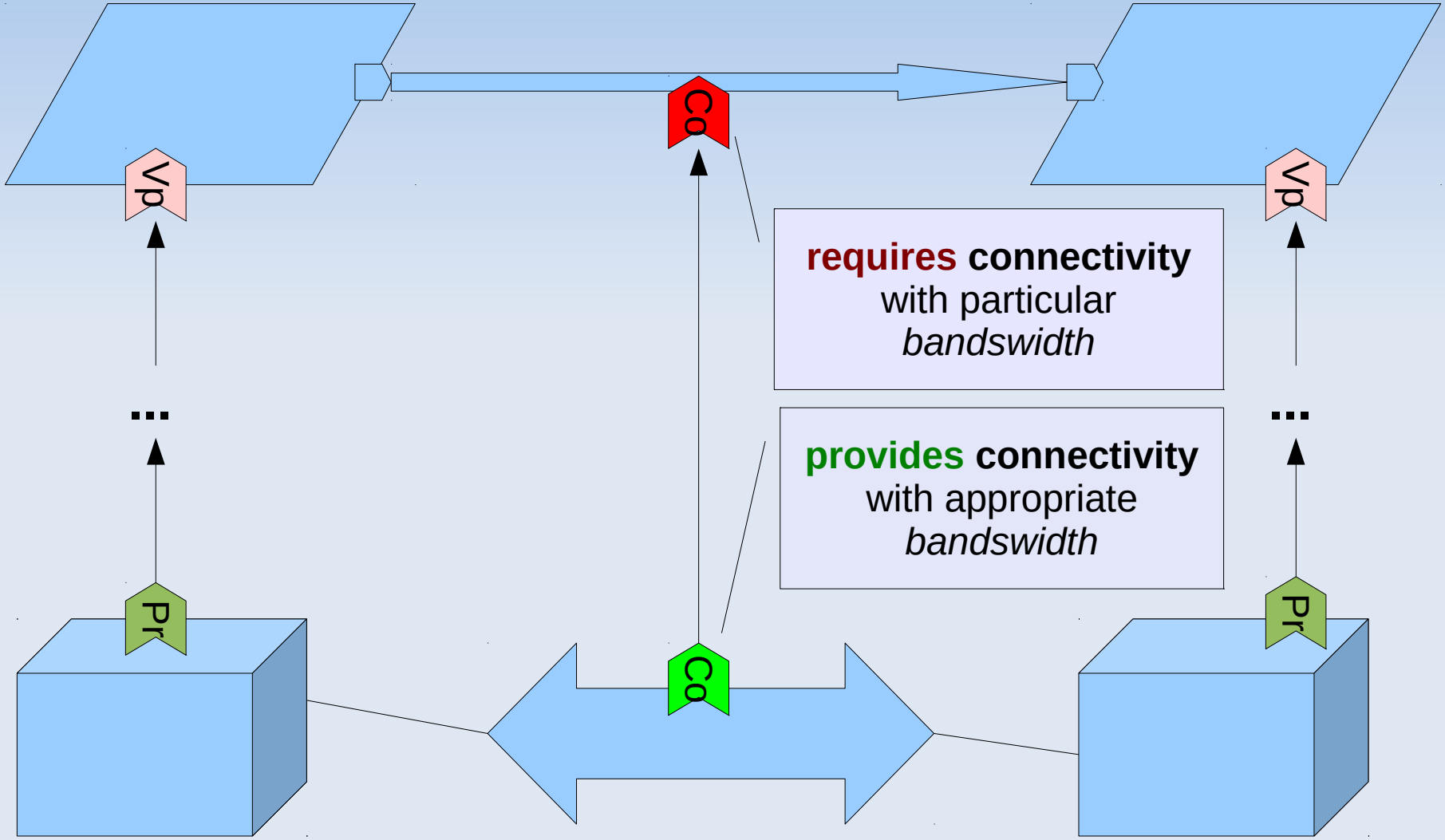
Resource type example: virtual processor



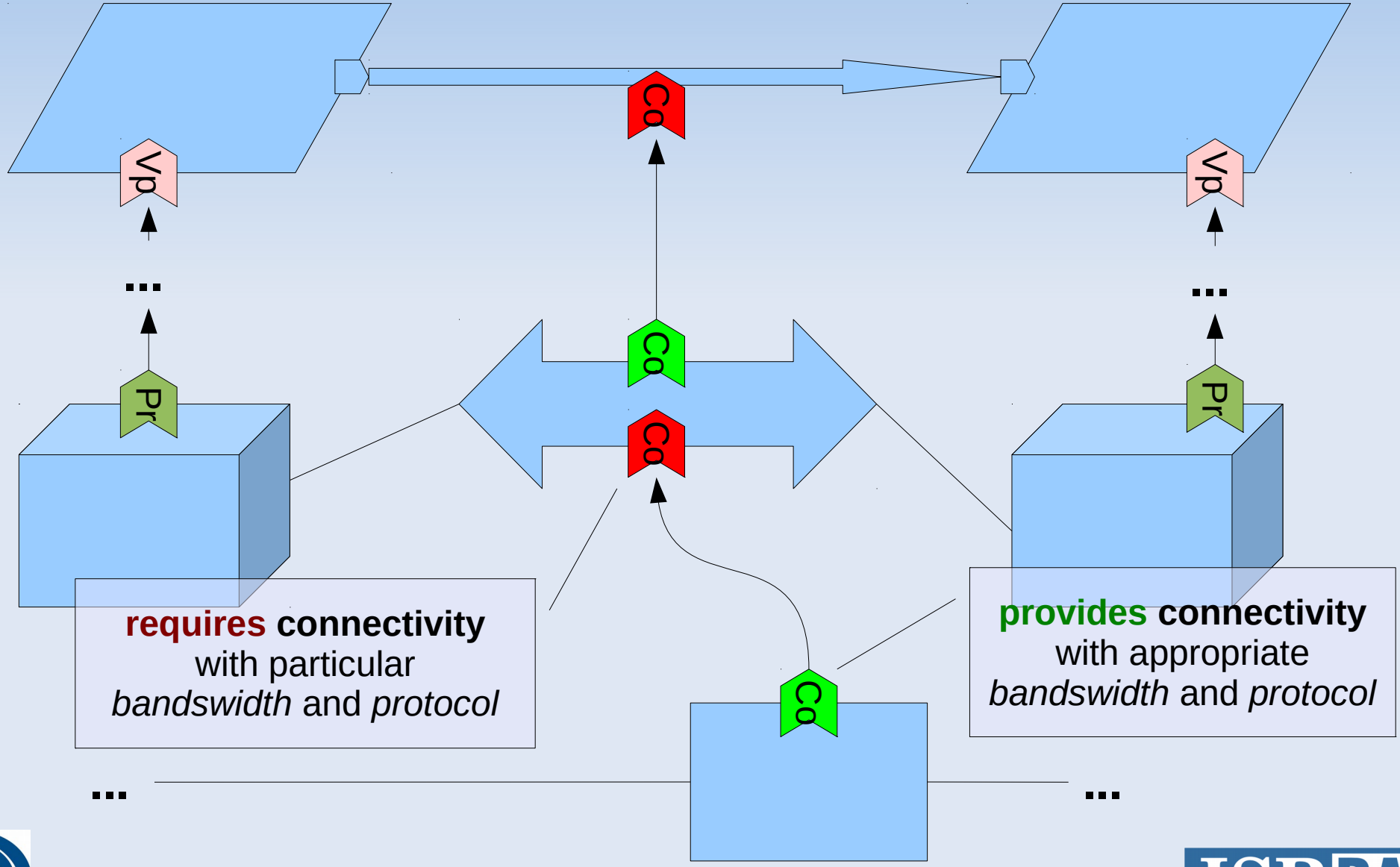
Resource type example: virtual processor+subprogram access



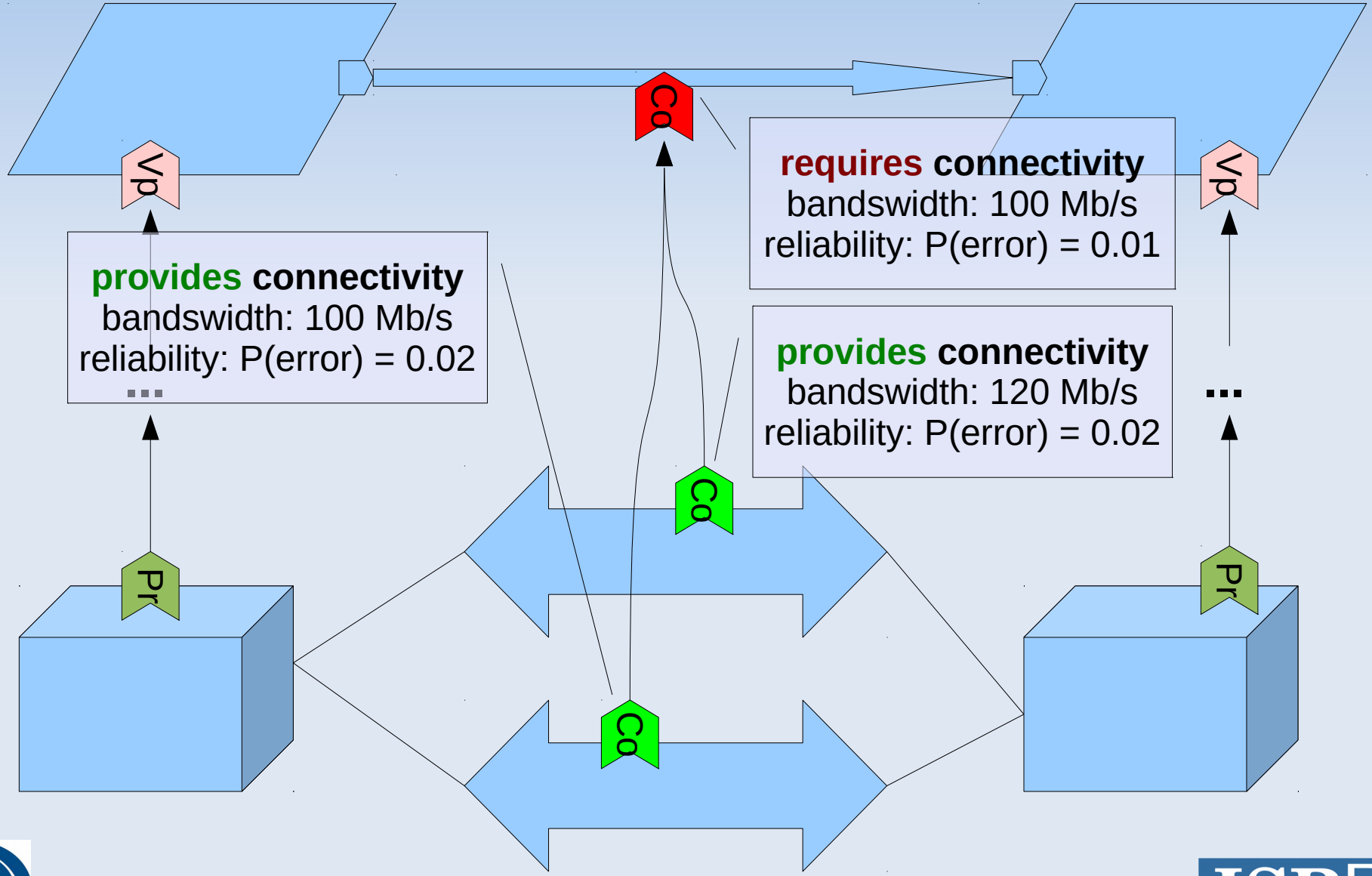
Resource type example: connectivity



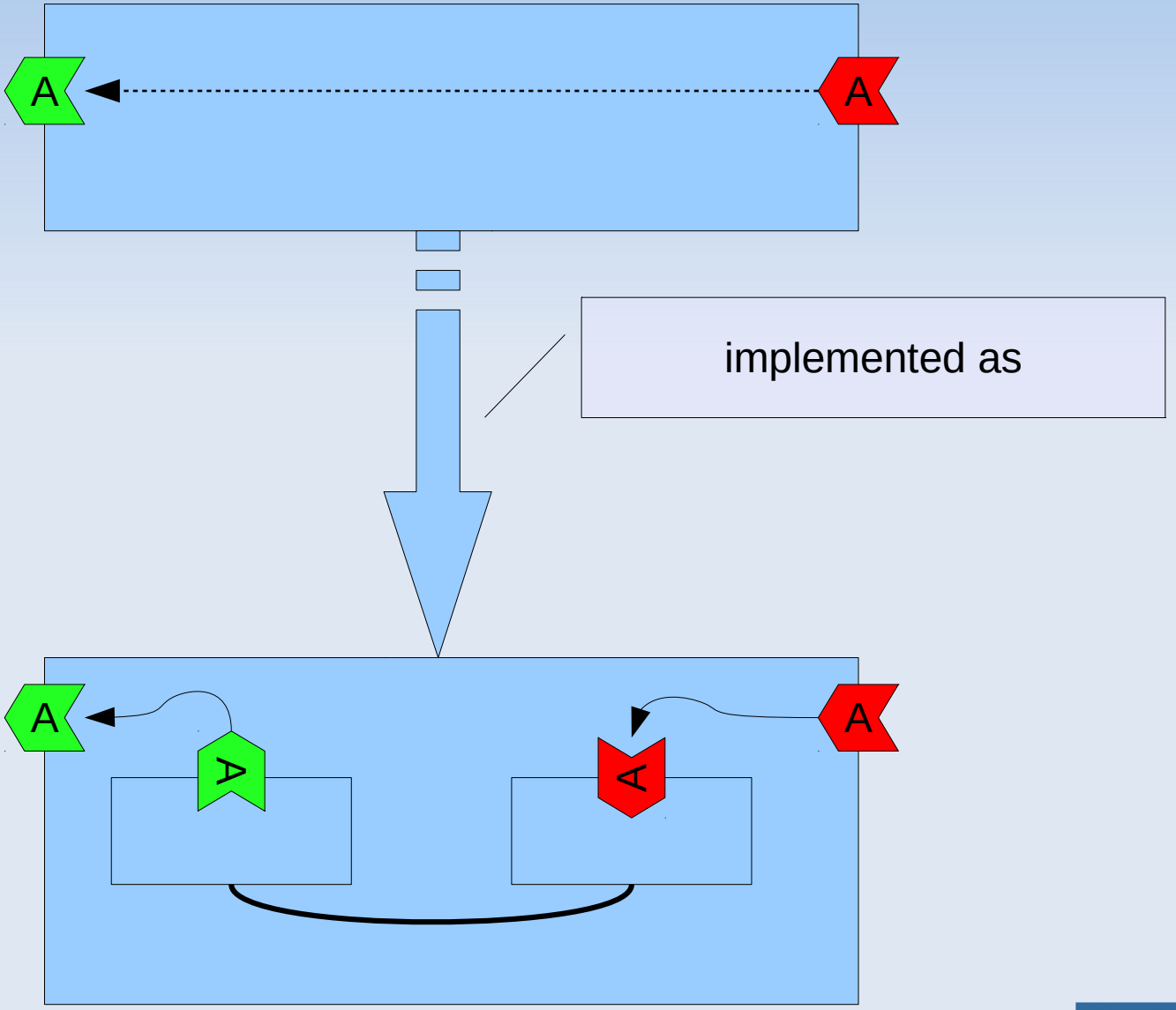
Resource type example: connectivity



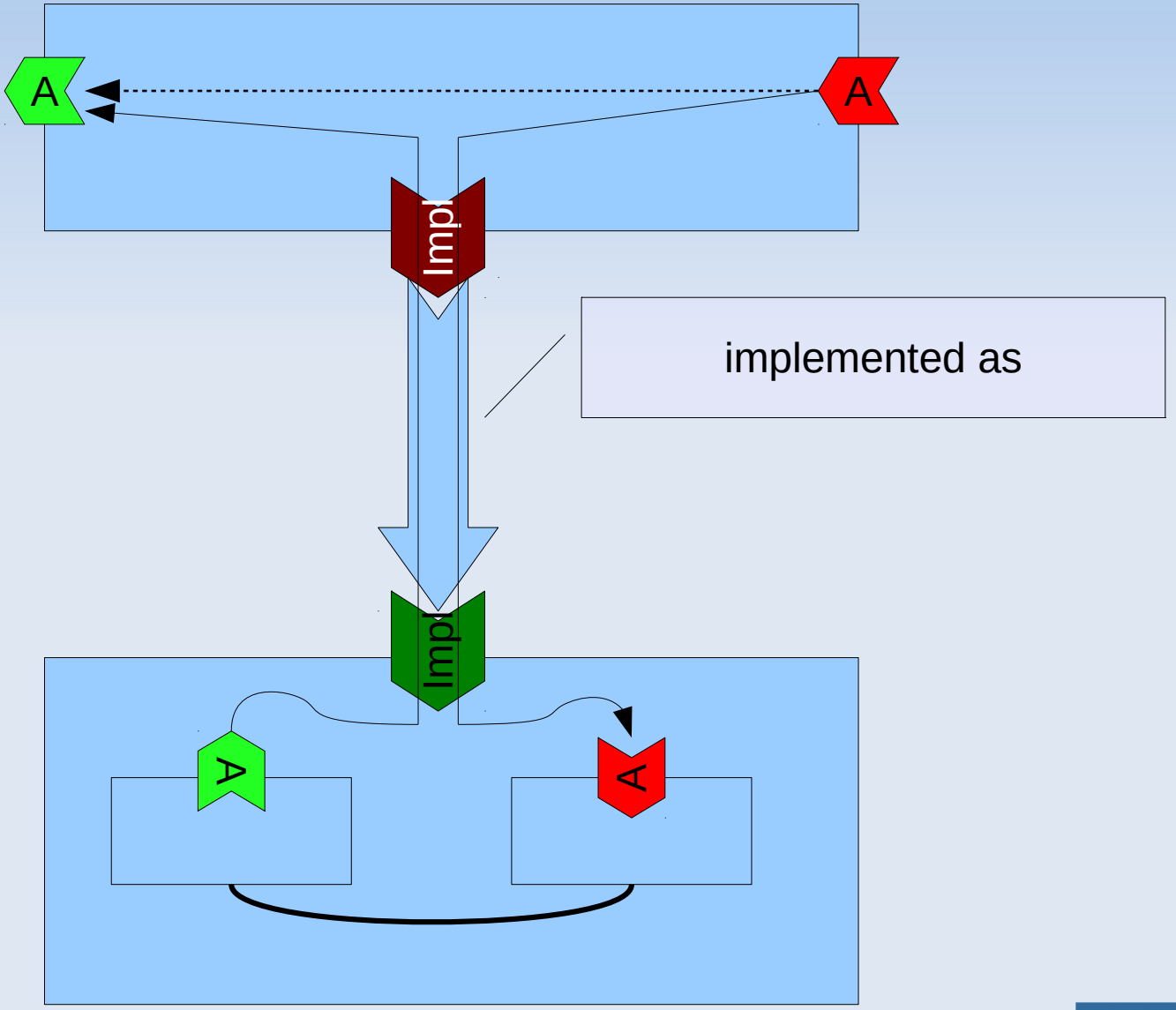
Resource type example: connectivity



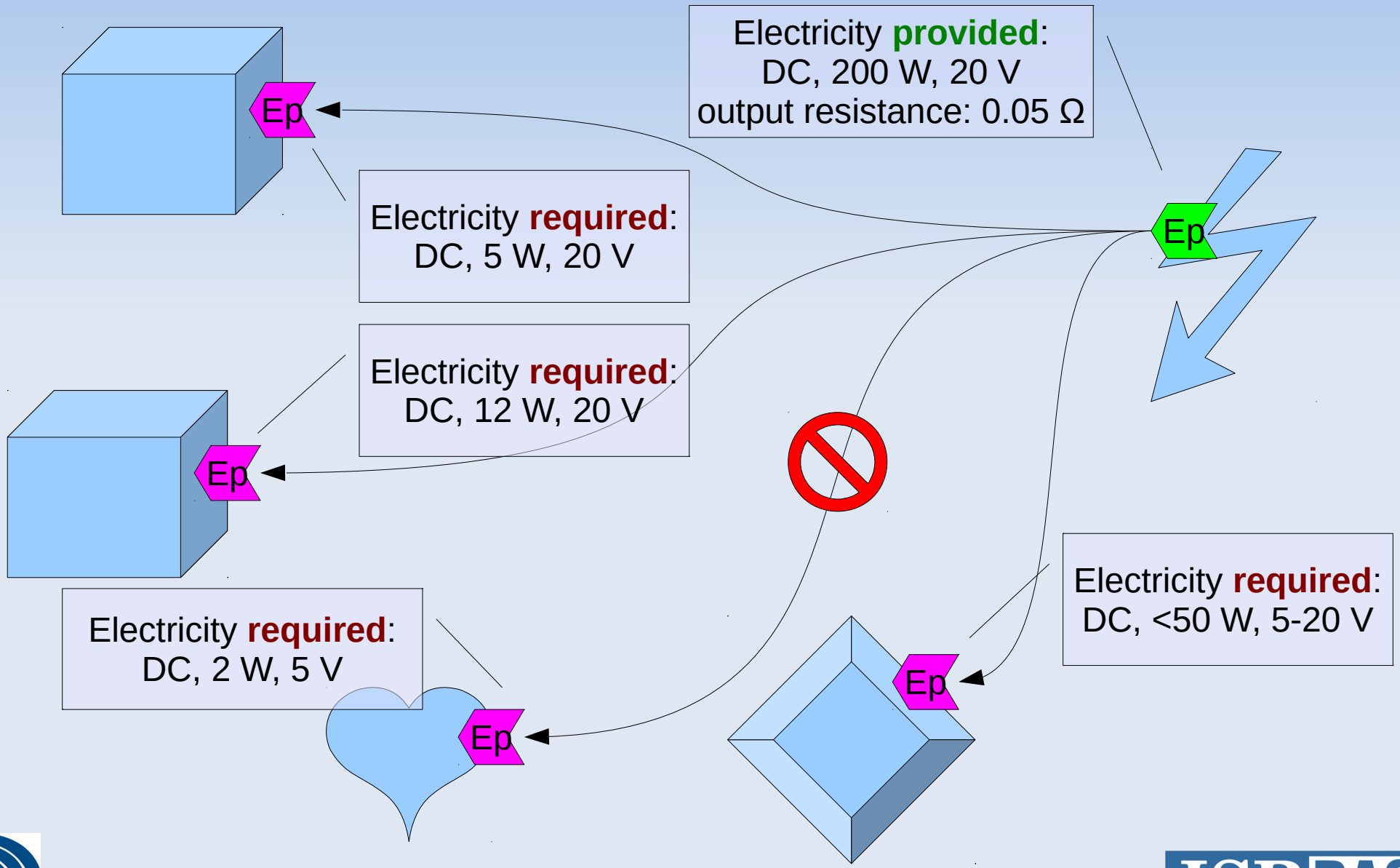
Implementing as



Resource type example: functionality proxying

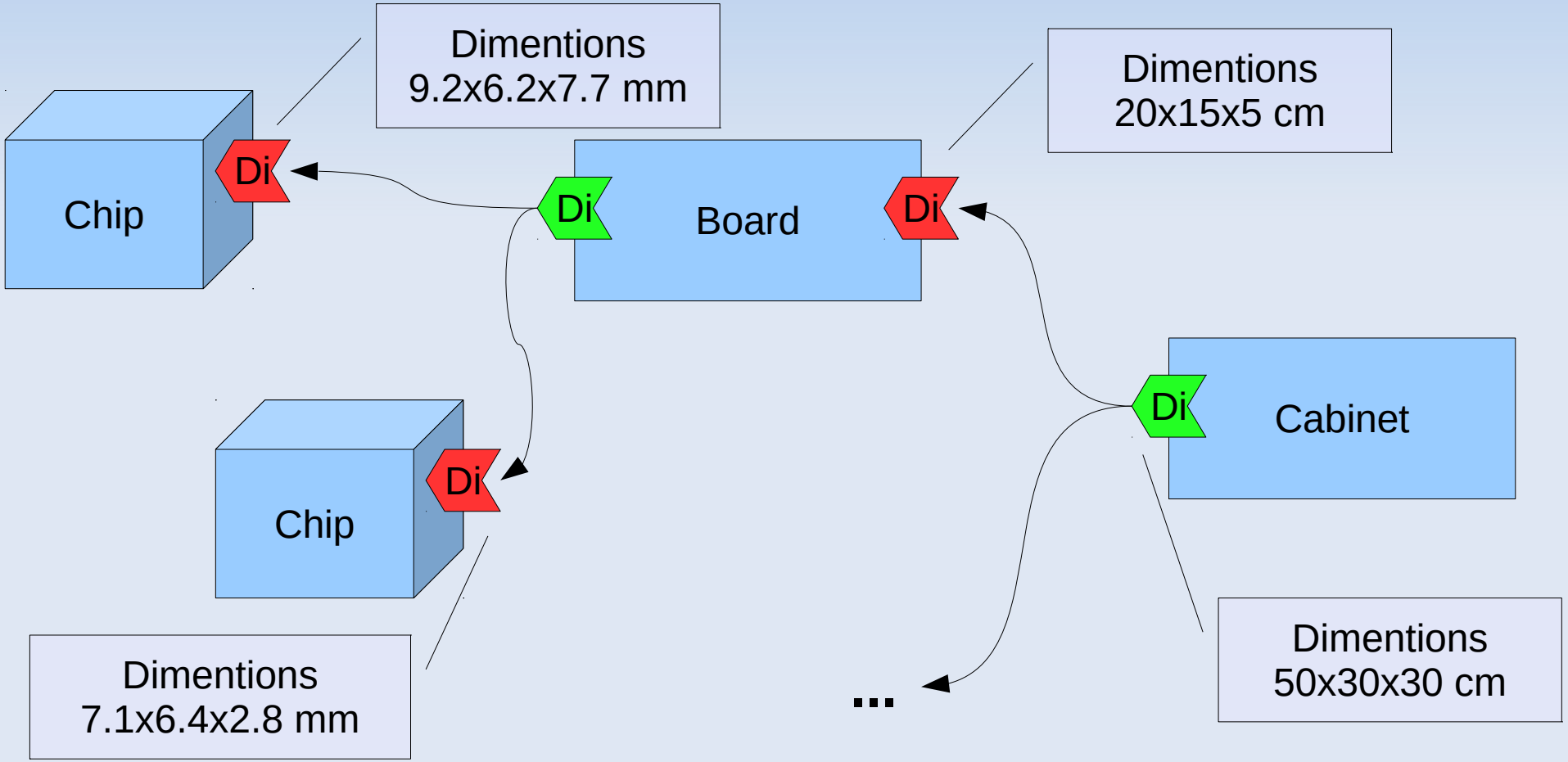


Exotic resource types: electric power

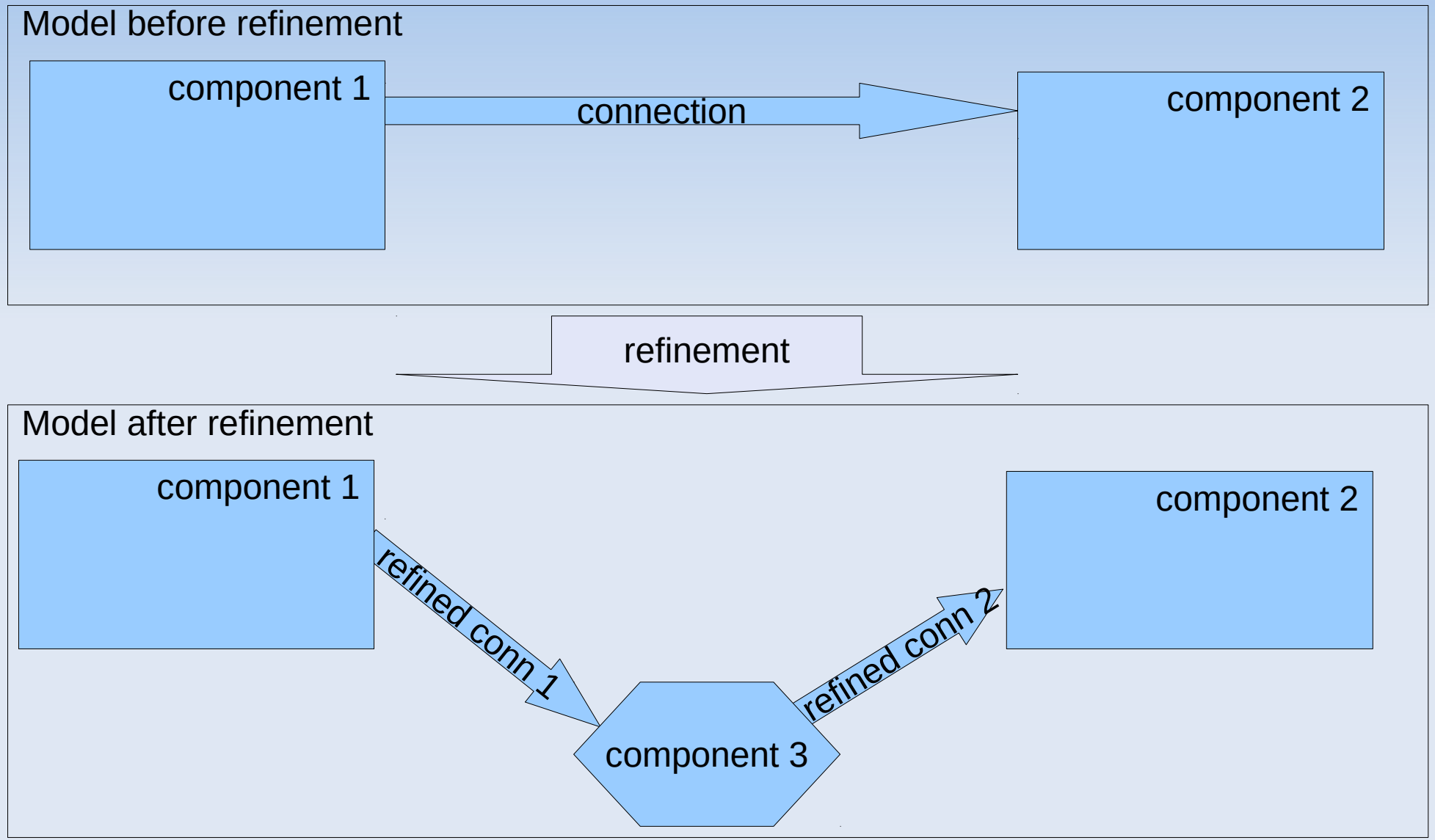


Exotic resource types: physical containment (deployment)

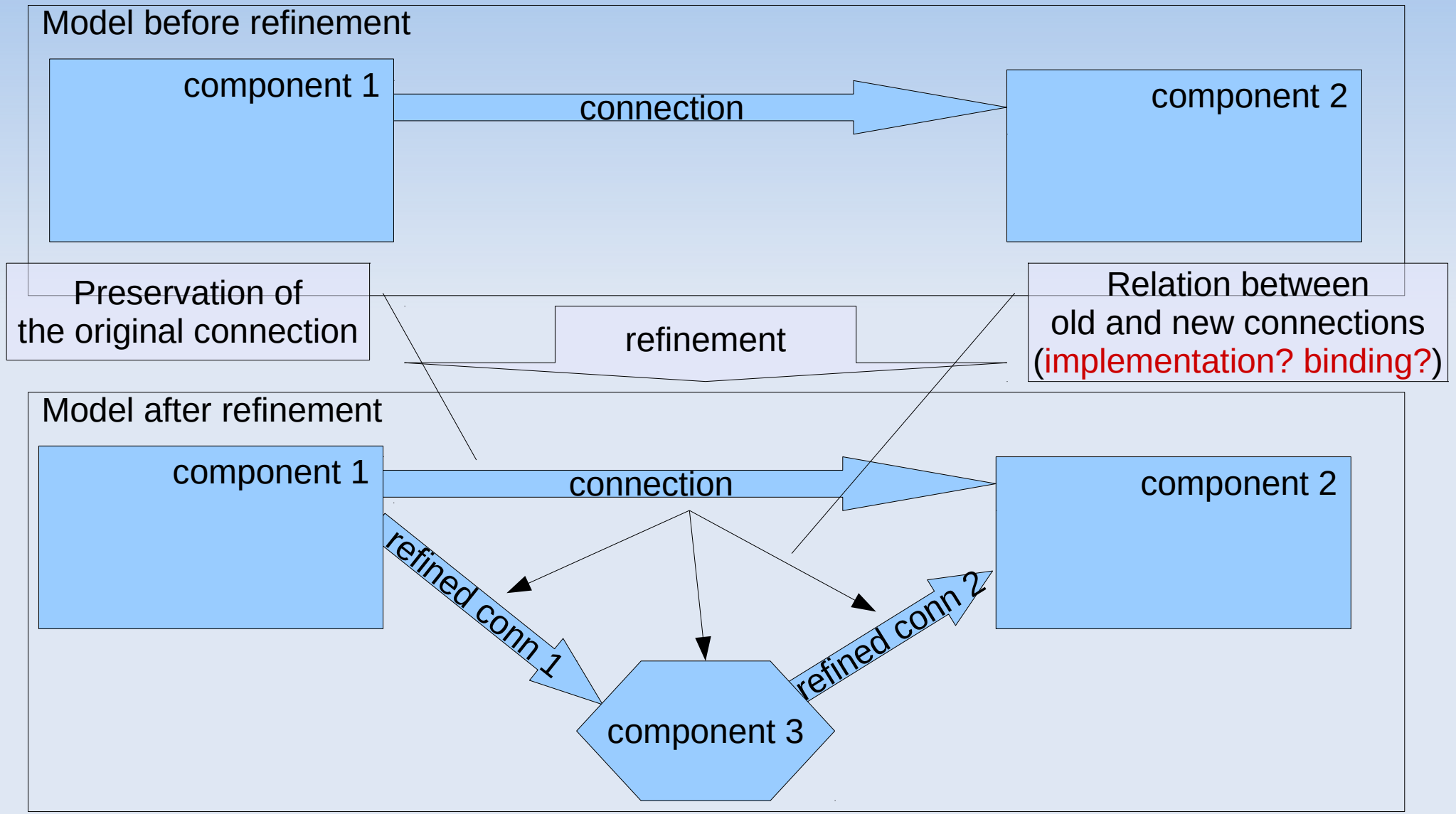
- Not to be confused with *model containment*



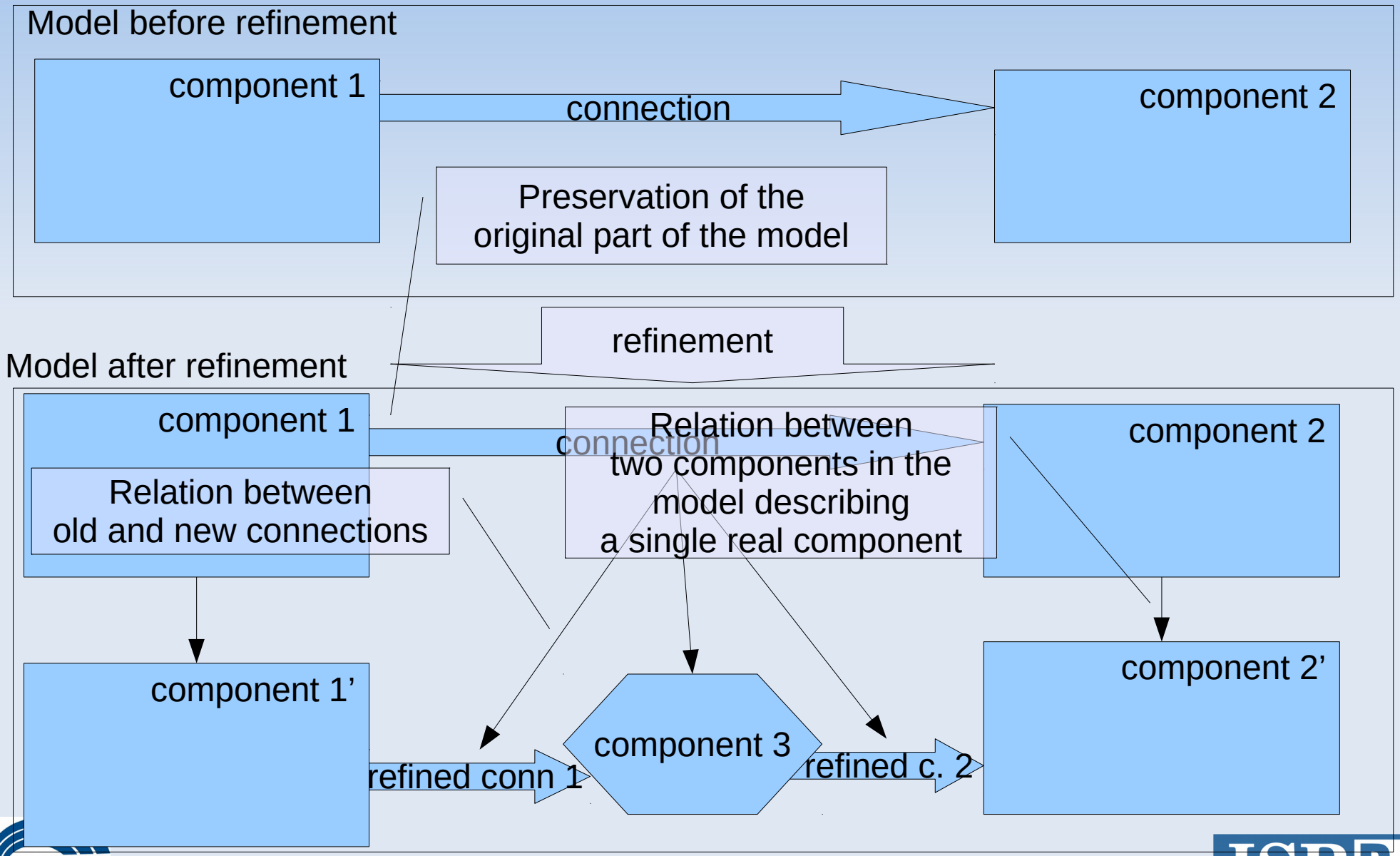
Connections refinement variant 1



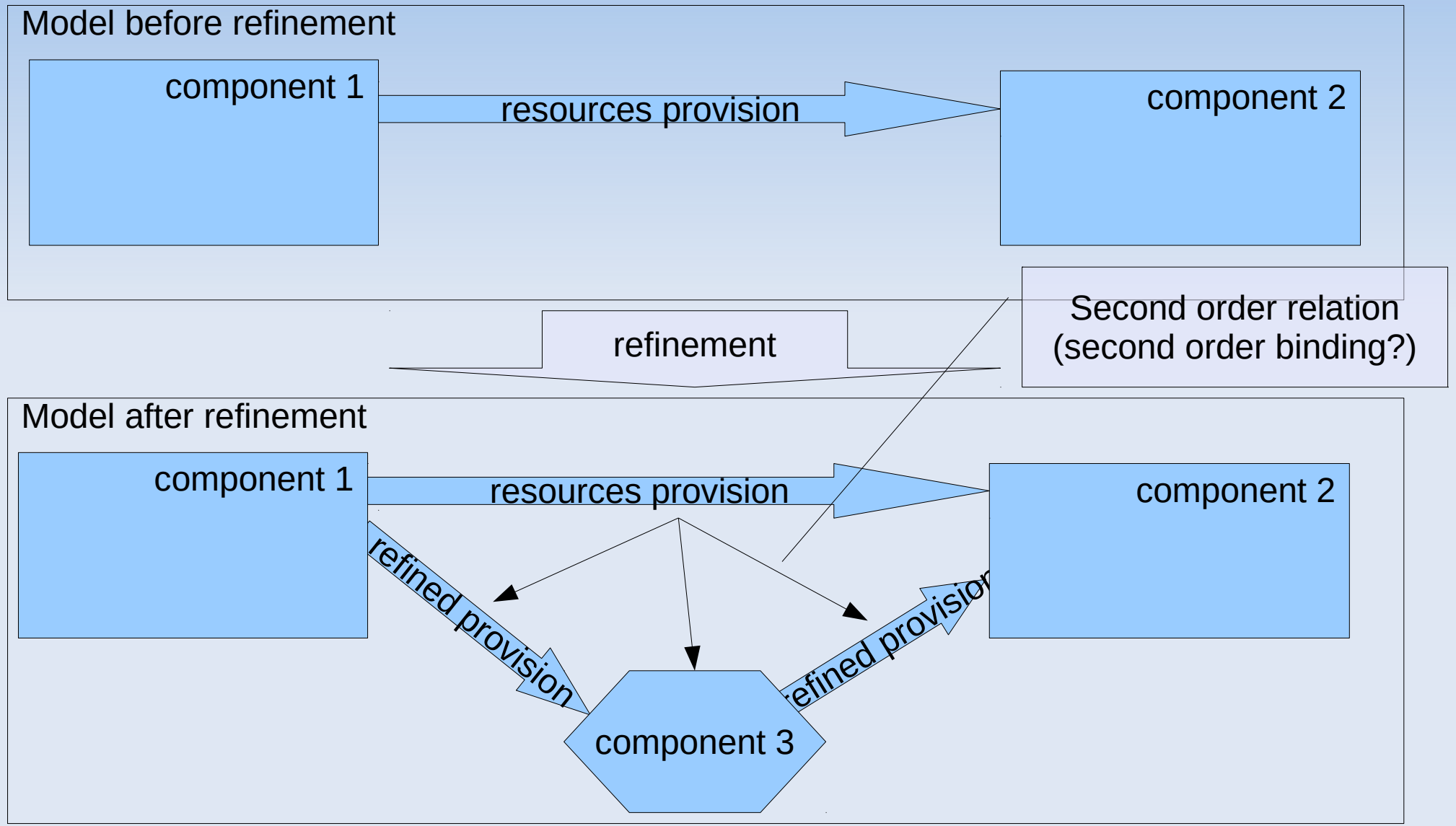
Connections refinement variant 2



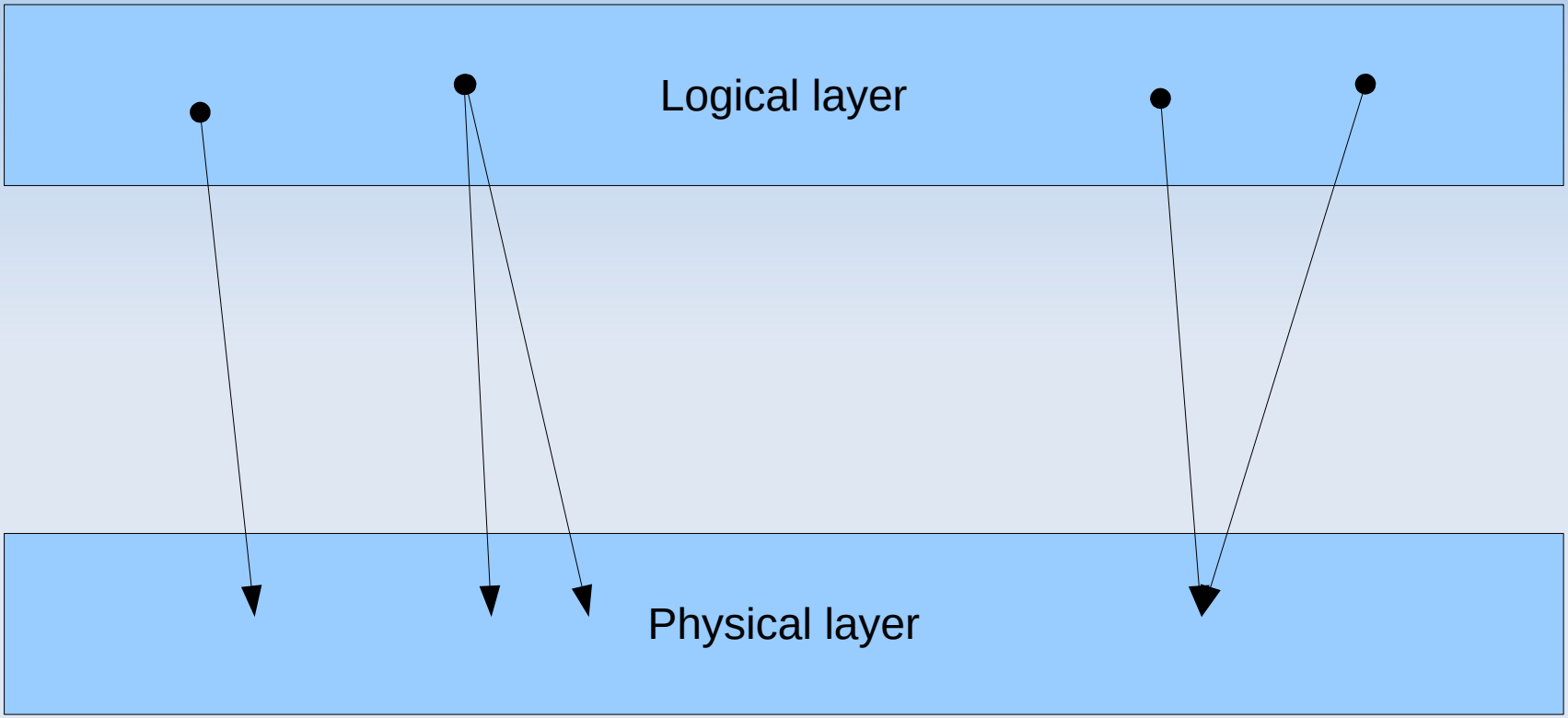
Connections refinement variant 3



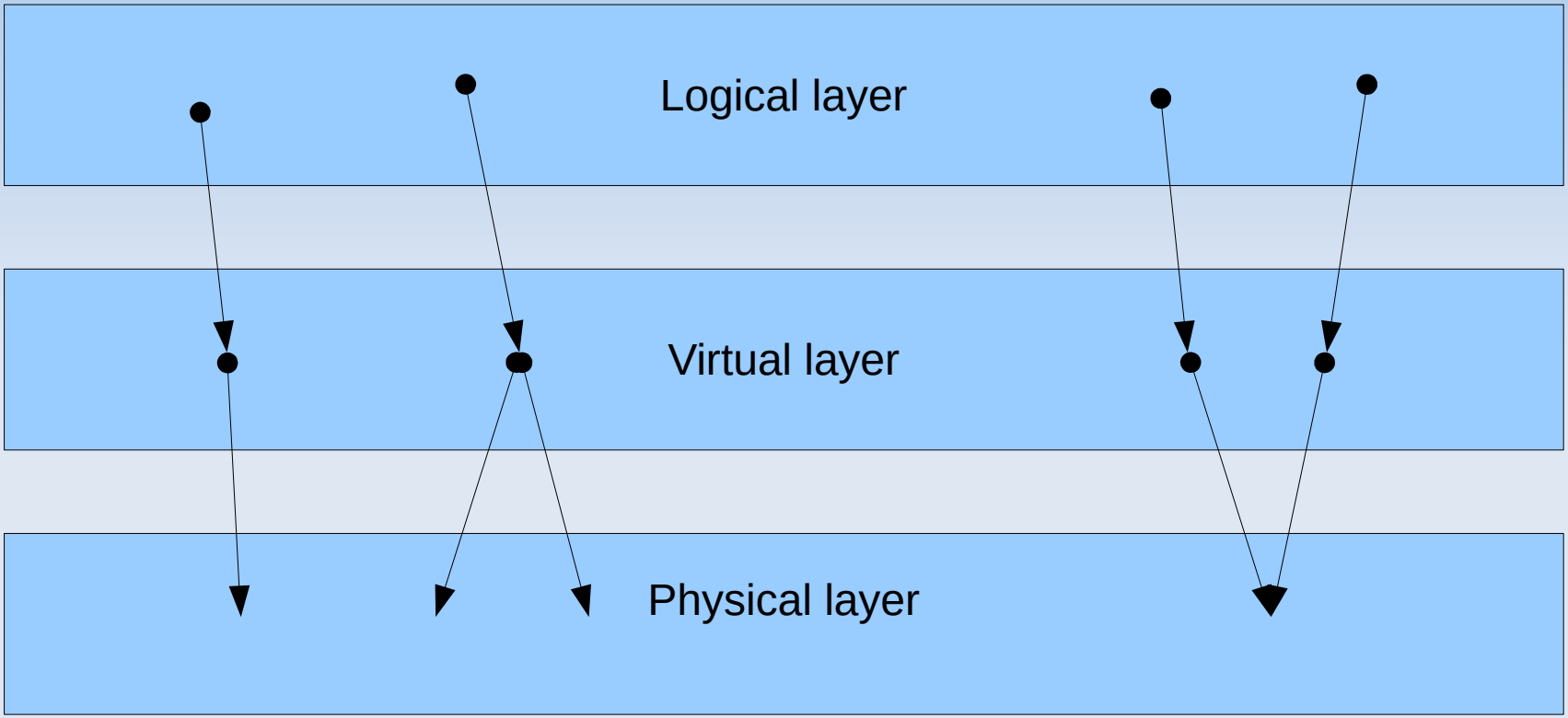
Parallels: resources provision or bindings refinement



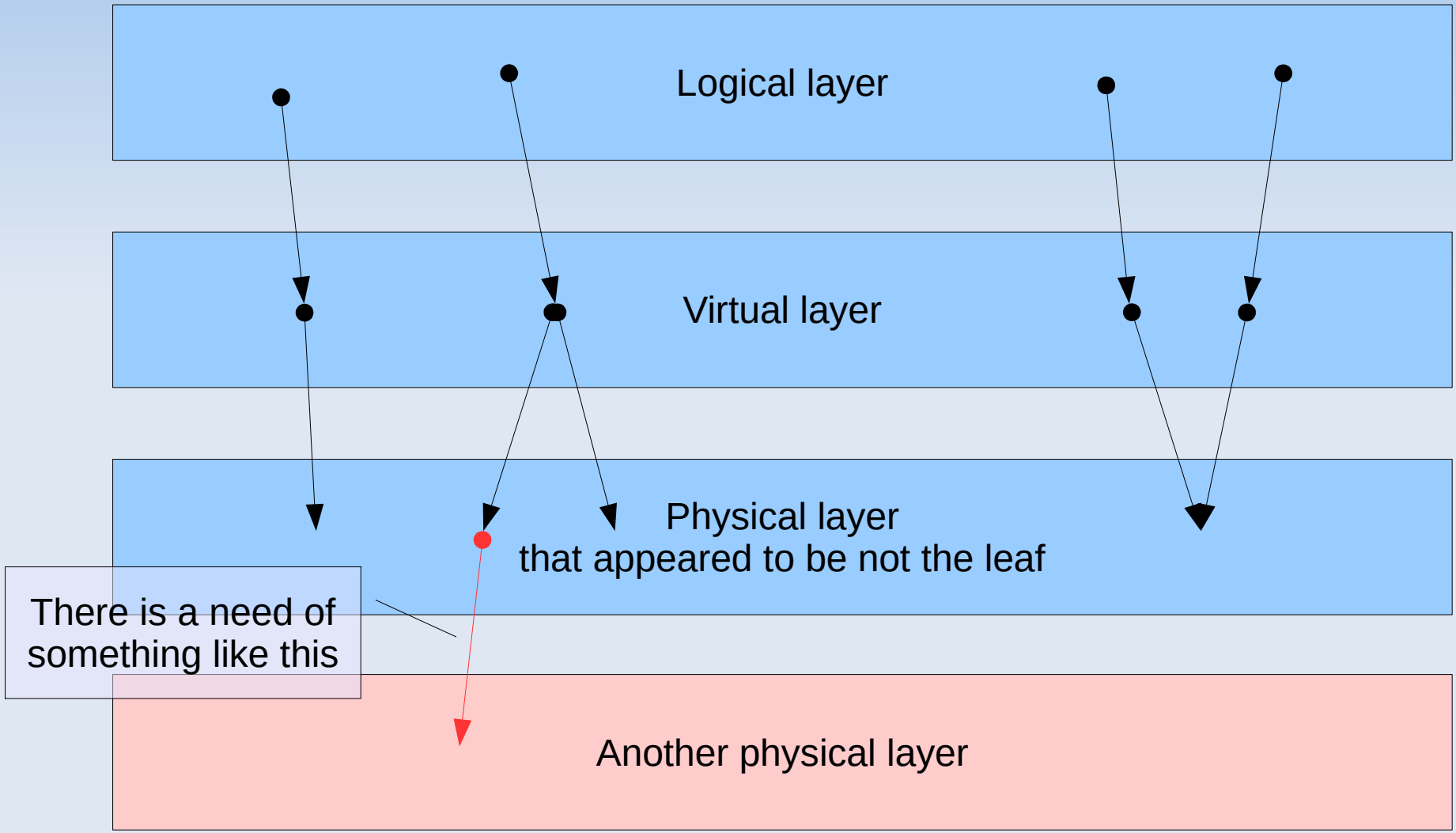
Layered view
Two-layer approach



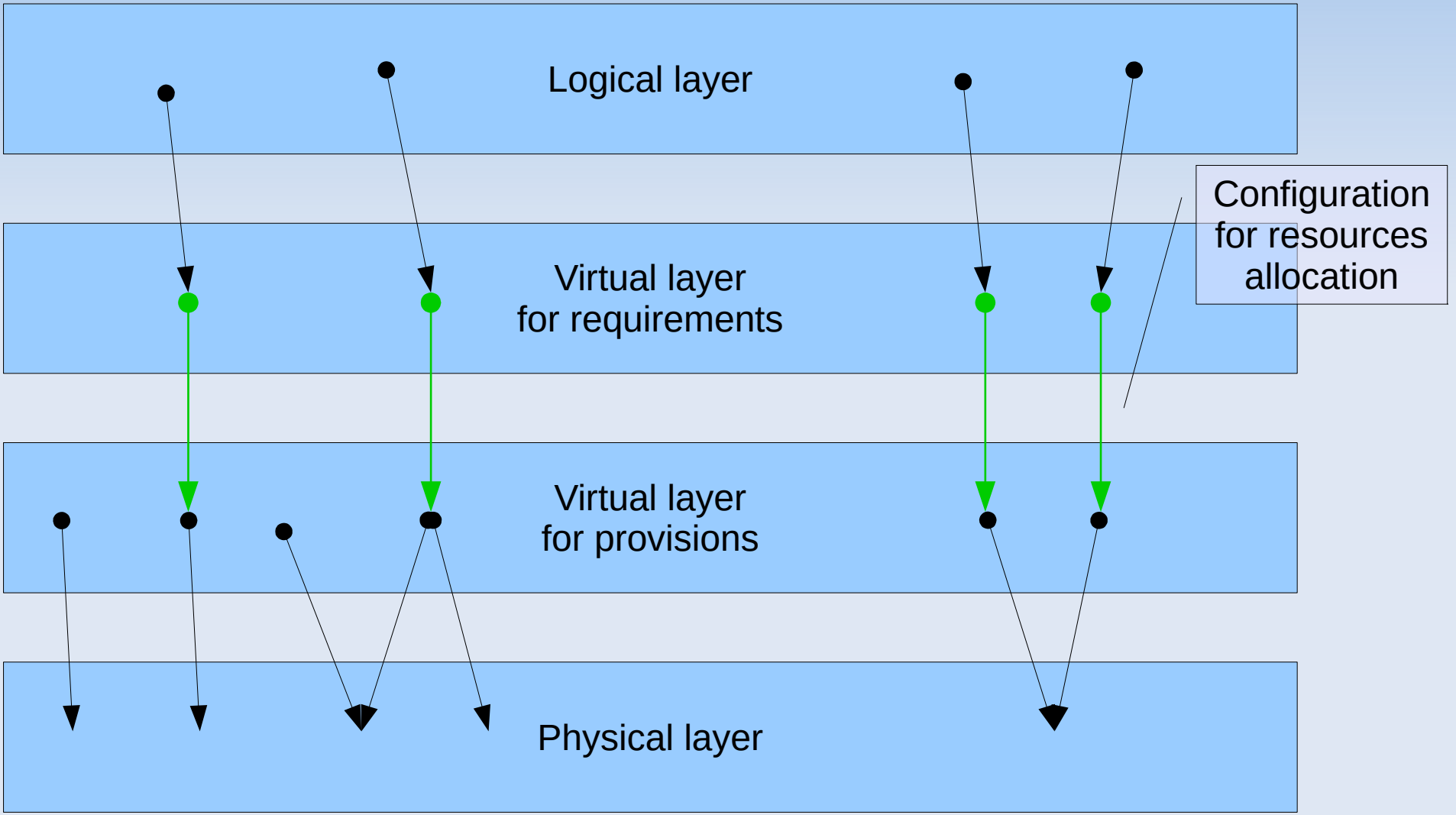
Layered view
Three-layer approach



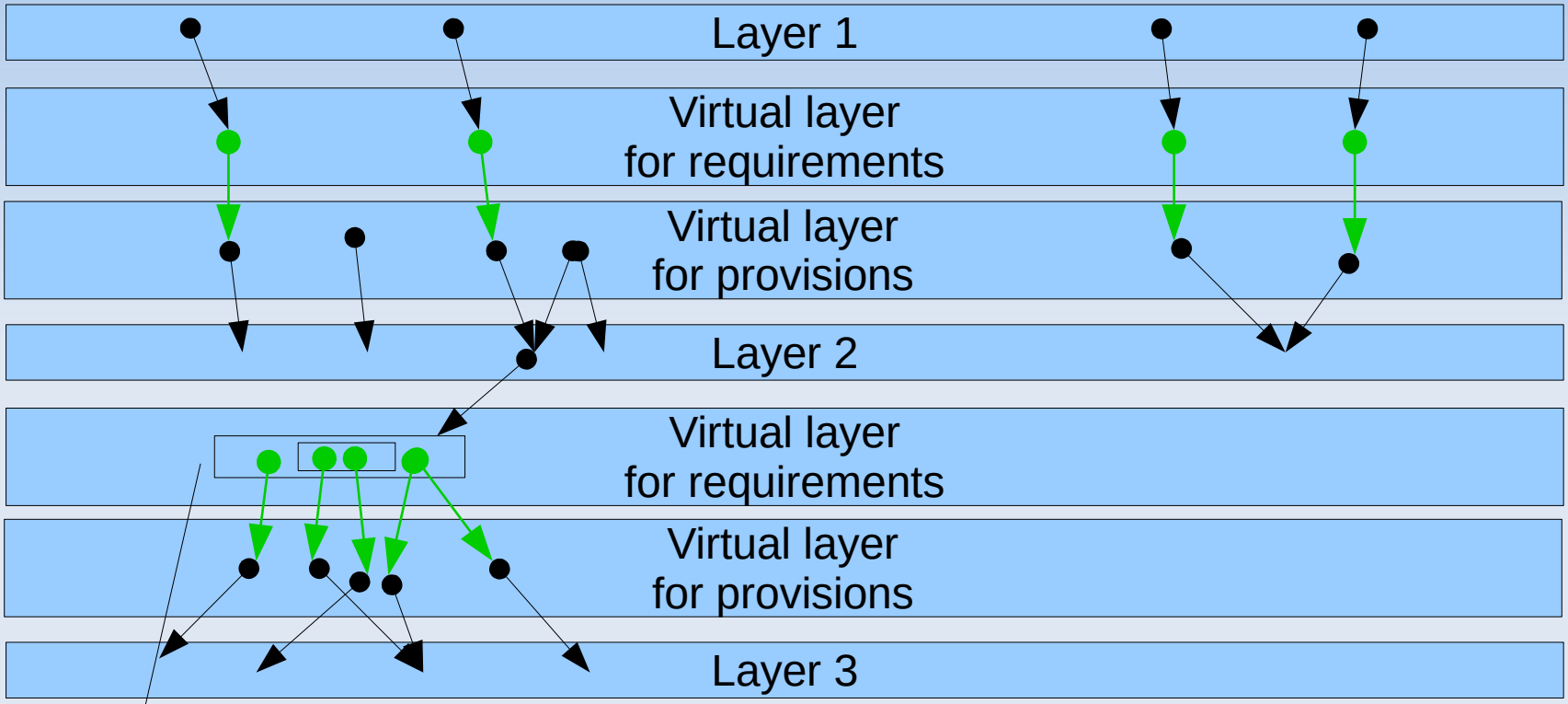
Layered view
Problems of three-layer approach



Layered view
Four-layer approach



Layered view
(1 + 3 * n)-layer approach

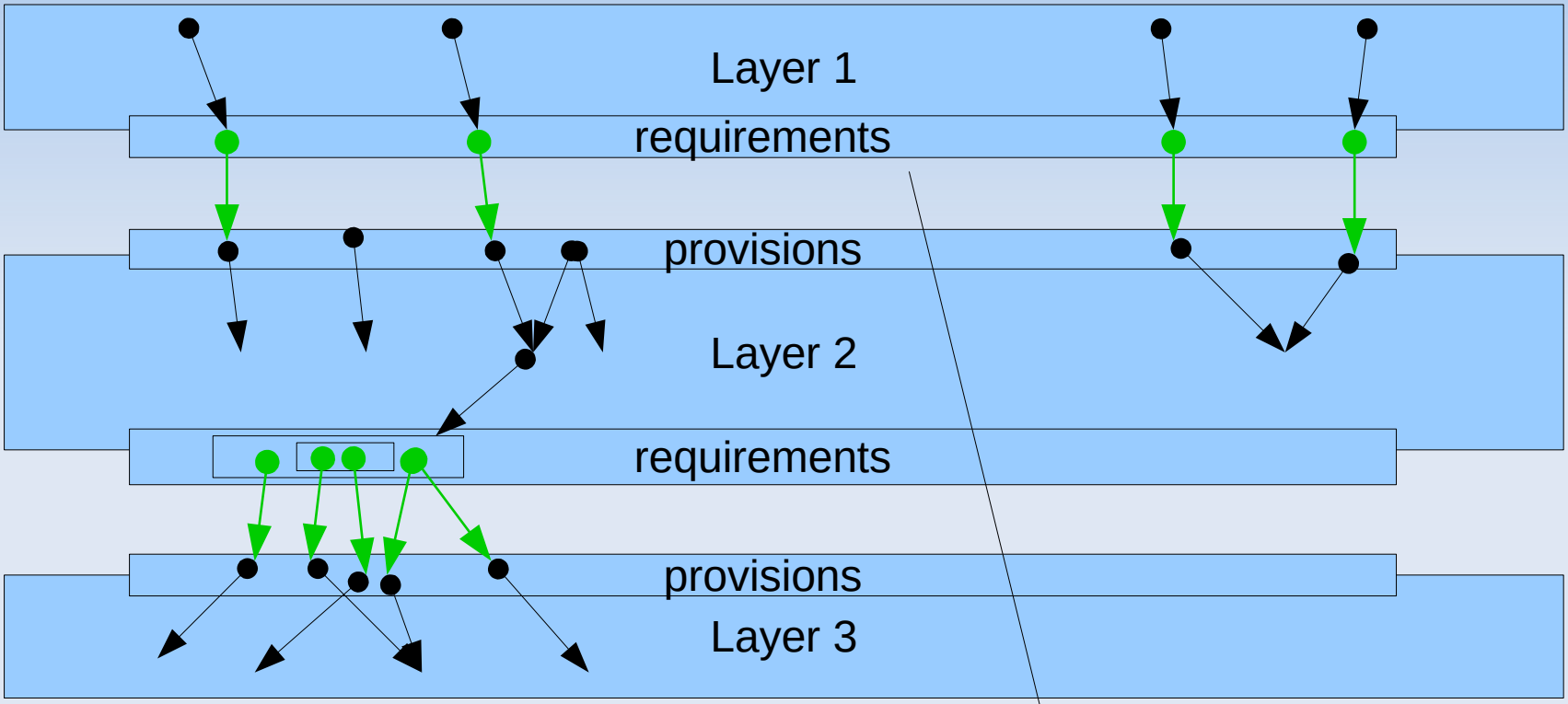


They may be structured

etc...

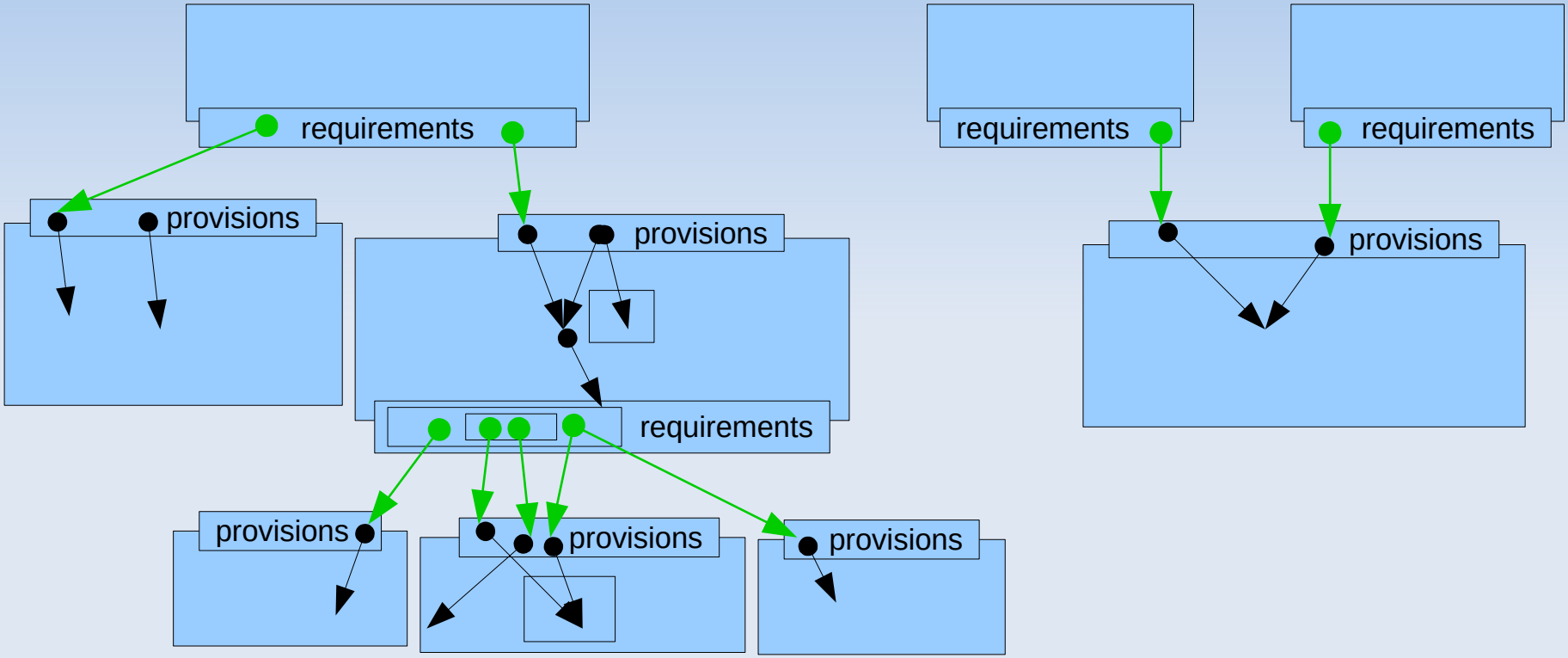
As many layers as it is needed

Layered view
Virtual layers are *interfaces*



Requirements and provisions are in fact parts of layers but not separate layers

Super-layered view
Interfaces belong to *components*



Thank you!

Denis Buzdalov
buzdalov@ispras.ru