A Formal Framework for Component Deployment

Y. David Liu
Scott F. Smith
Johns Hopkins University

OOPSLA'06, Portland, Oregon

A Menagerie of Deployment Systems

CLI Assemblies

JSR 277

InstallShield

RPM

OSGi

Dpkg

EJB Manifests

Portage

Bazaar

CORBA D&C

RubyGems

CTAN

CPAN

Foundations?

CLI Assemblies

JSR 277

InstallShield

RPM

OSGi

Dpkg

?

EJB Manifests

Portage

Bazaar

CORBA D&C

RubyGems

CTAN

CPAN

An Analogy: Programming Languages

Fortran

Smalltalk

Pascal

Java

Lisp

Perl

C++

Scala

C#

Scheme

C

Haskel

ML

An Analogy: Foundations of Languages

Fortran

Smalltalk

Pascal

Lisp

Java

λ Calculus
Object Calculi
etc.

Perl

C++

Scala

C#

Scheme

C

Haskel

ML

This Work

CLI Assemblies

JSR 277

InstallShield

RPM

OSGi

Dpkg

Application Buildbox

EJB Manifests

Portage

Bazaar

CORBA D&C

RubyGems

CTAN

CPAN

This Work

An abstract, platform-independent, vendor-independent study of component deployment

- Designing components as deployment units
- Formalizing the entire deployment lifecycle
- Proving deployment invariants

Design objectives: simple (capturing recurring themes) and expressive

This Work

An abstract, platform-independent, vendor-independent study of component deployment

- Designing components as deployment units
- Formalizing the entire deployment lifecycle
- Proving deployment invariants

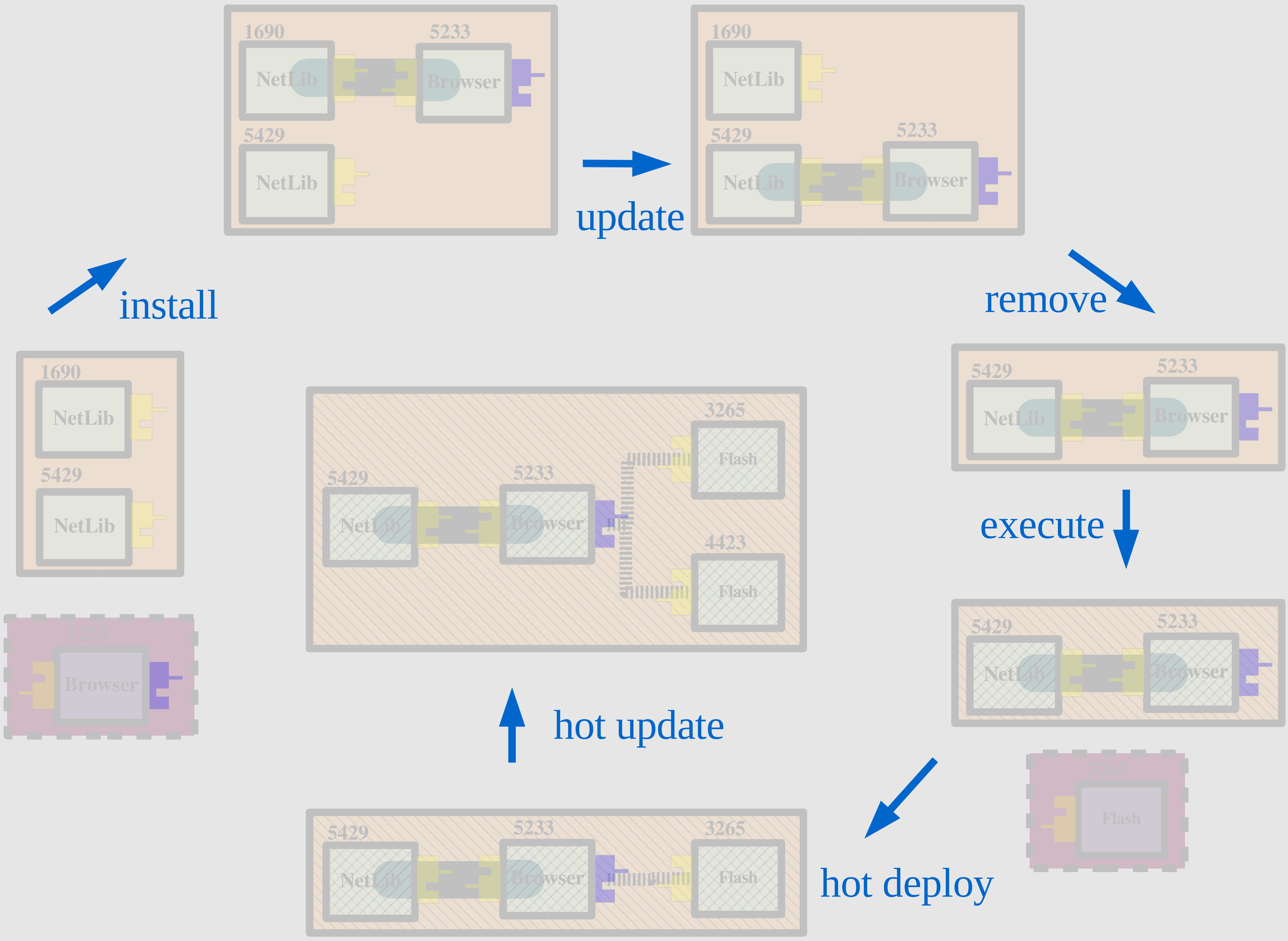
Design objectives: simple (capturing recurring themes) and expressive

This Work

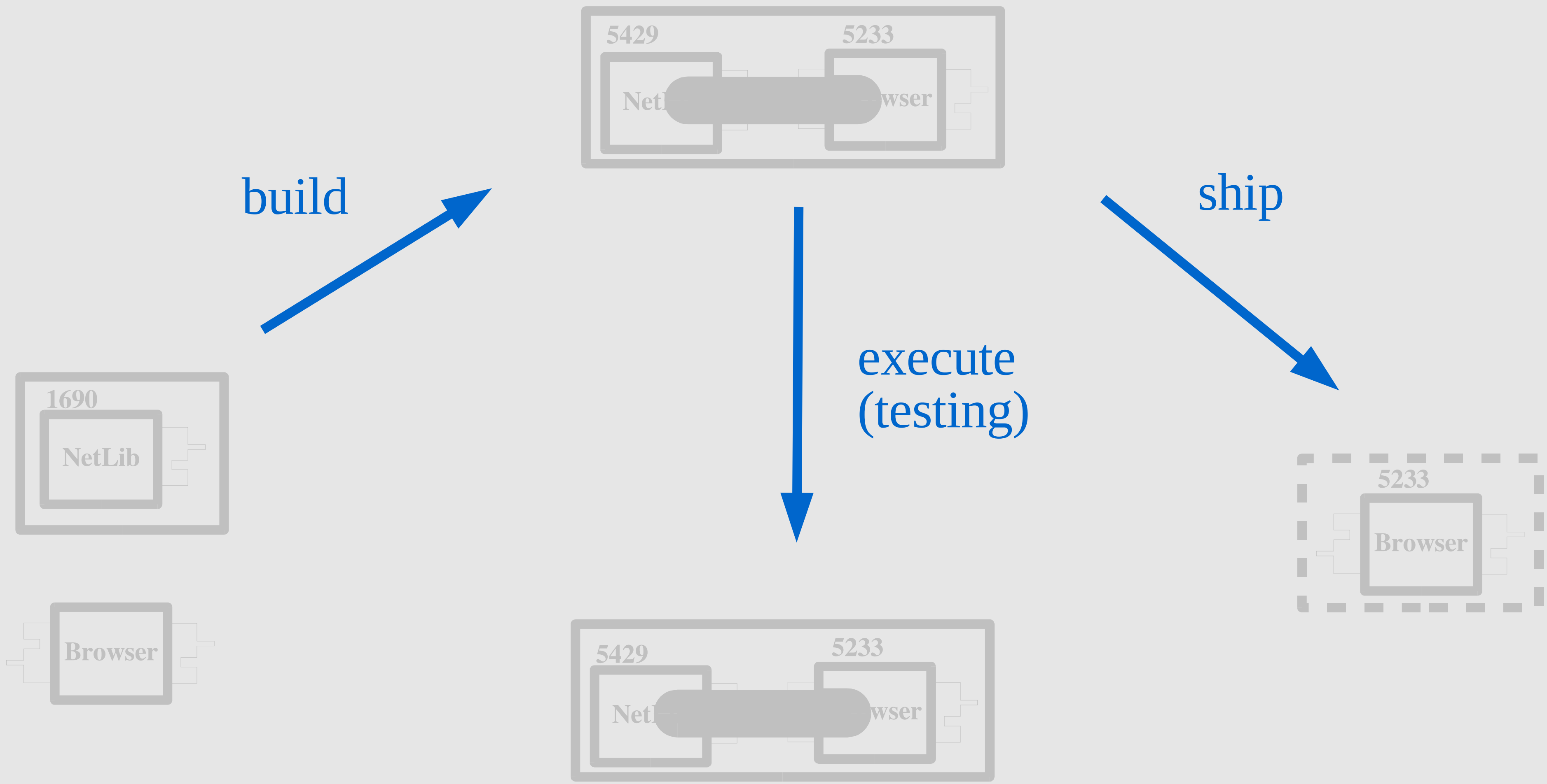
An abstract, platform-independent, vendor-independent study of component deployment

- Designing components as deployment units
- Formalizing the entire deployment lifecycle
- Proving deployment invariants

Design objectives: simple (capturing recurring themes) and expressive



update
remove
execute
hot deploy
hot update
install
1690
5233
5429
4423
3265
NetLib
Browser
Flash


build
ship
execute
(testing)
5429
5233
1690
NetLib
Browser
5233
Browser
5429
5233

This Work

An abstract, platform-independent, vendor-independent study of component deployment

- Designing components as deployment units
- Formalizing the entire deployment lifecycle
- Proving deployment invariants
  - Deployment "never goes wrong"
  - Version compatibility

Design objectives: simple (capturing recurring themes) and expressive

This Work

An abstract, platform-independent, vendor-independent study of component deployment

- Designing components as deployment units
- Formalizing the entire deployment lifecycle
- Proving deployment invariants

Design objectives: simple (capturing recurring themes) and expressive

Why Foundations?

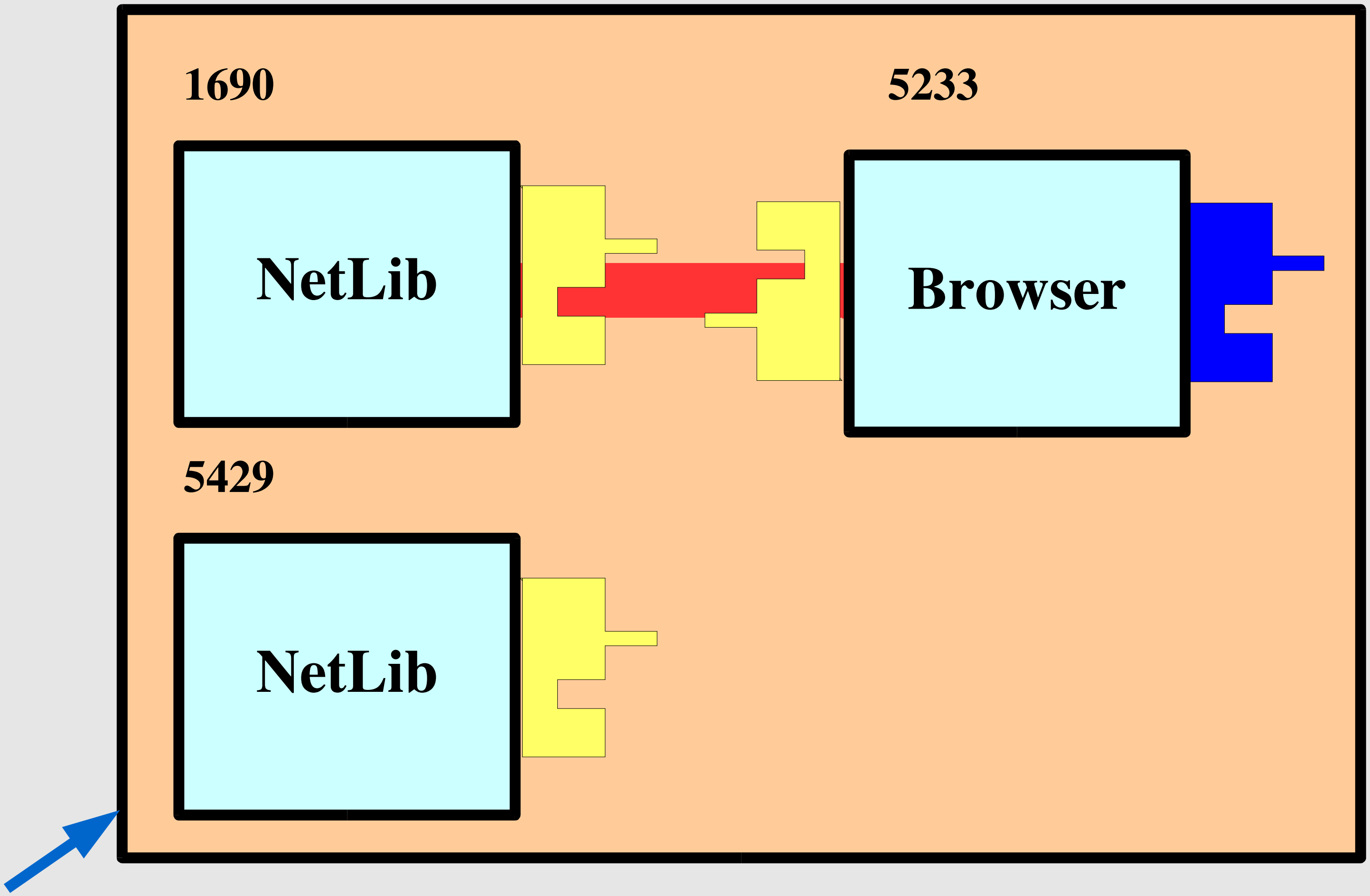
- Fosters next-generation deployment systems
  - Elucidates subtle issues
  - More features proposed from academic research community
  - Deployment systems with provably correct properties
- Complements modularity research
  - *when* and *where* of linking

Why Foundations?

- Fosters next-generation deployment systems
  - Elucidates subtle issues
  - More features proposed from academic research community
  - Deployment systems with provably correct properties
- Complements modularity research
  - ***when*** and ***where*** of linking

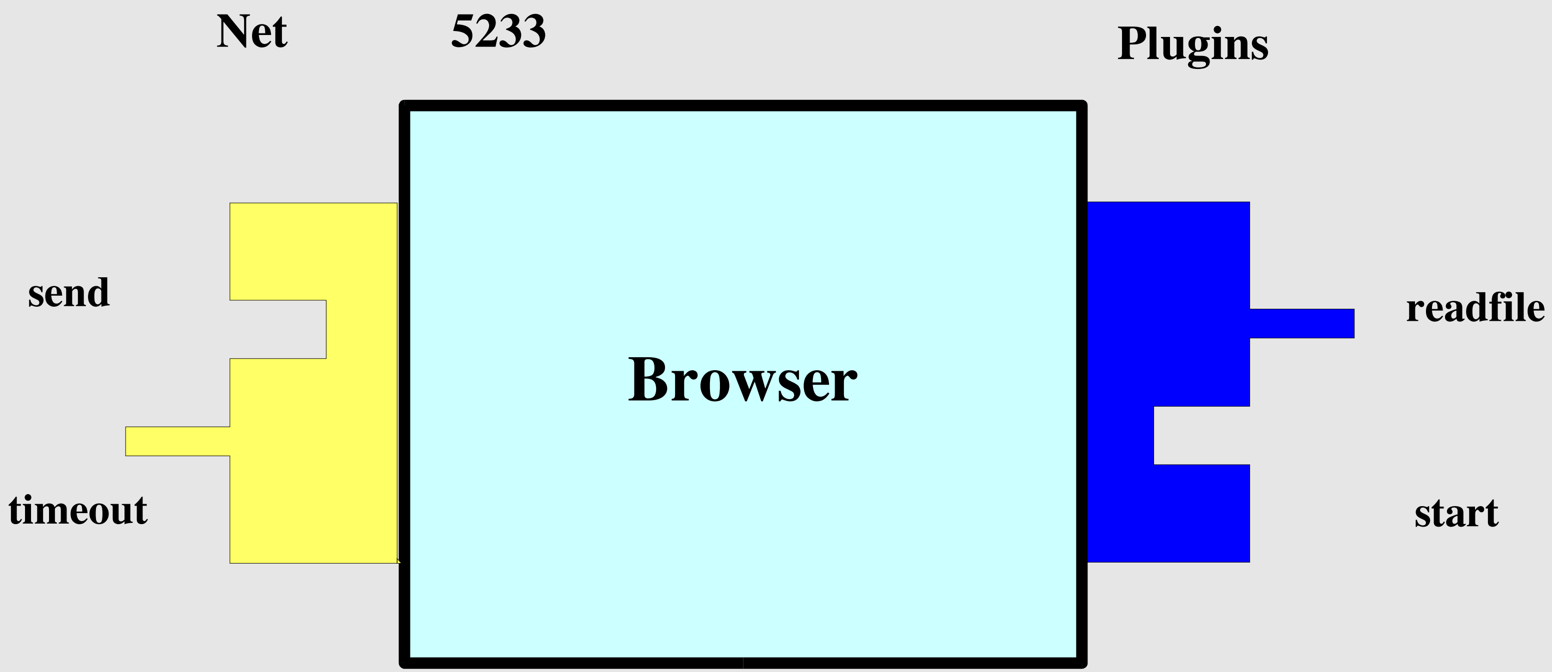
Basics

Application Buildbox



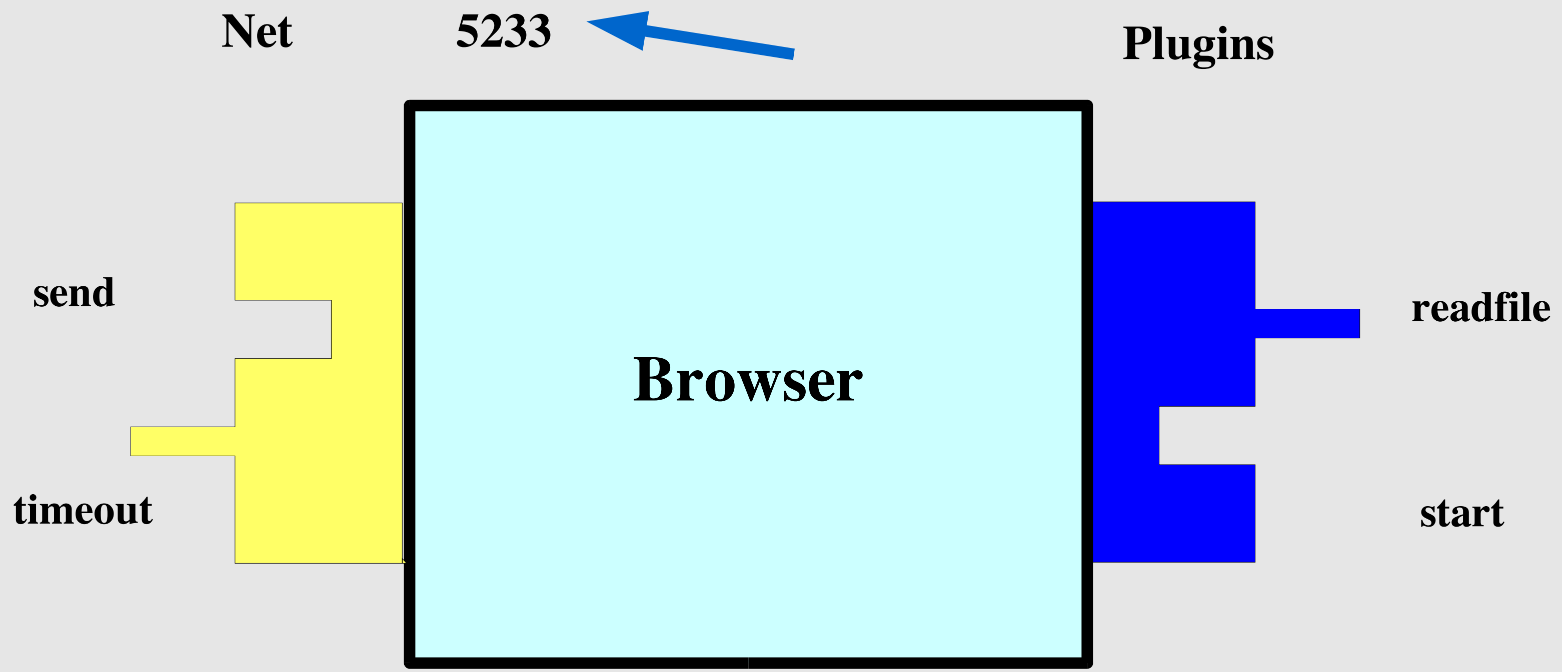
An imaginary box where an application "hatches" throughout the deployment lifecycle

Deployment Unit: Assemblage



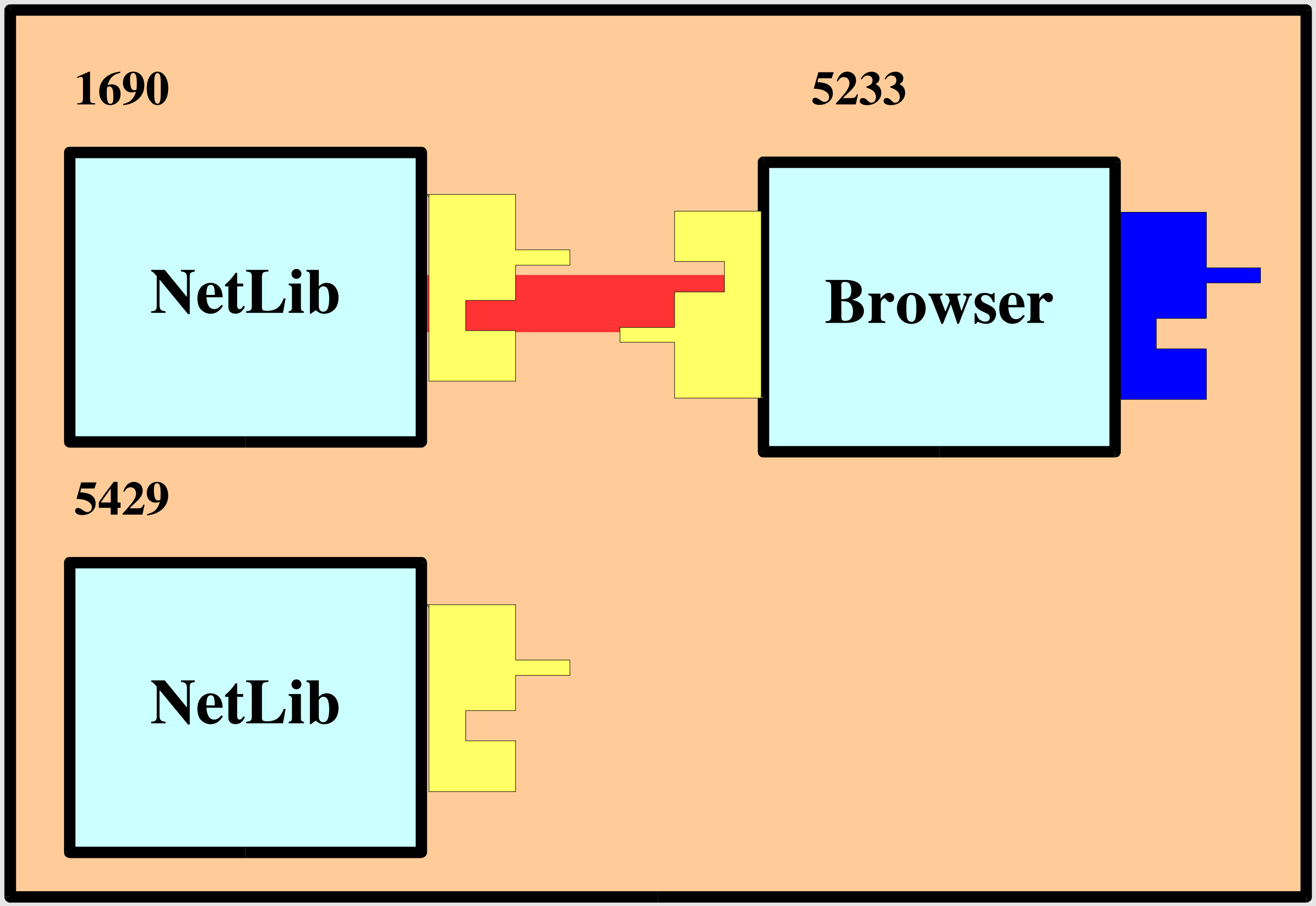
- Real-world analogues: JAR, C .so library, DLL, CLI Assembly
- Assemblages were first developed in [Liu and Smith, ECOOP'04], but without deployment

Version Identifiers



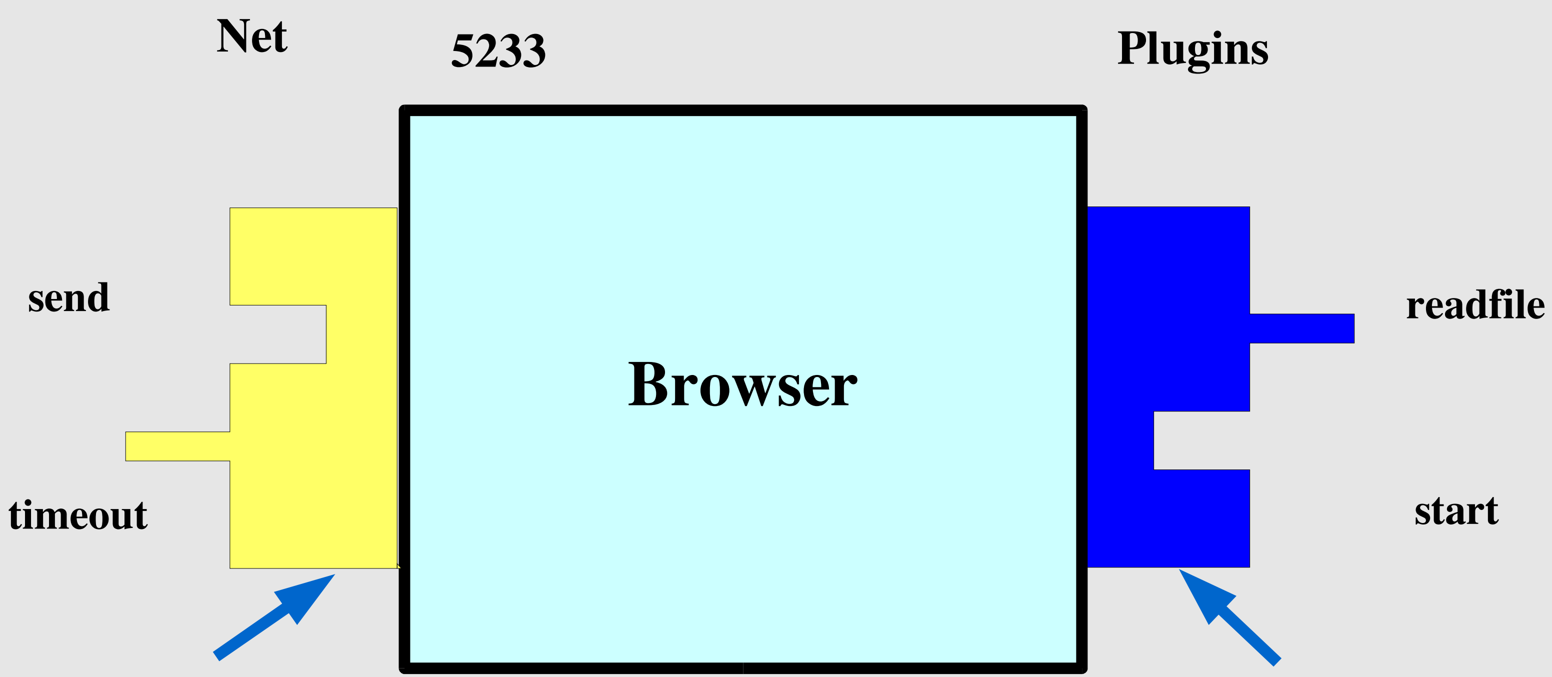
- Globally Unique
- Real-world analogues: COM+ GUID, CLI Assembly strong names

Side-by-Side Deployment



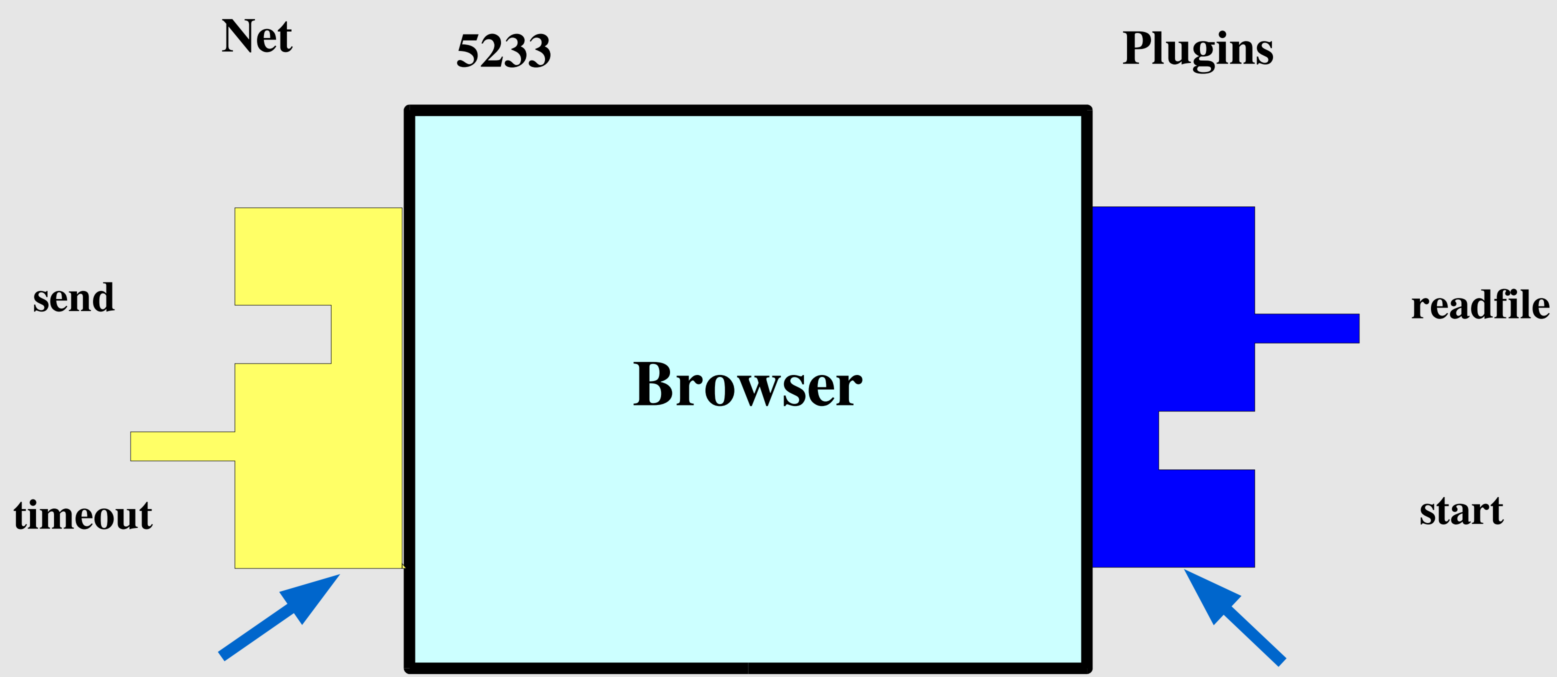
Two versions of the **NetLib** are deployed in the same buildbox

Basic Construct: Assemblage Interfaces



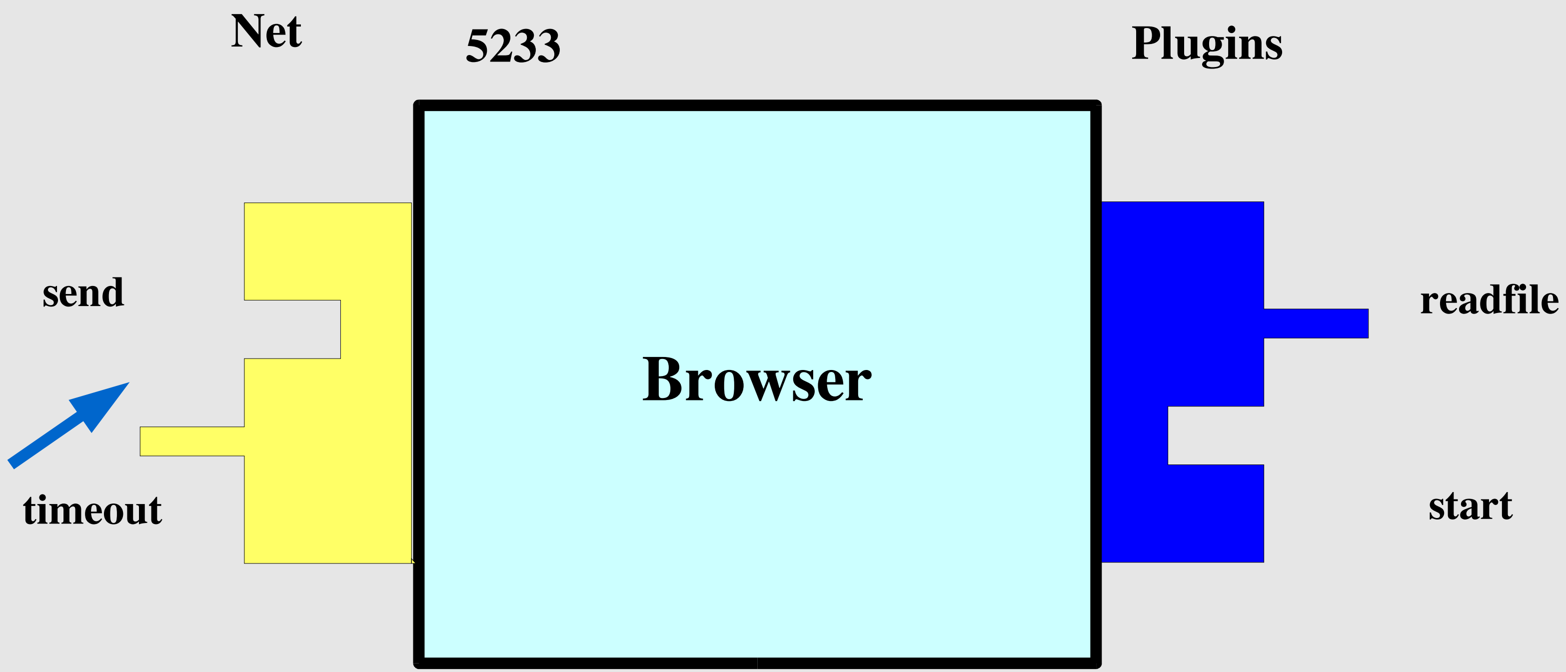
Real-world analogues: Manifest files, Deployment Descriptors

Two Kinds of Assemblage Interfaces

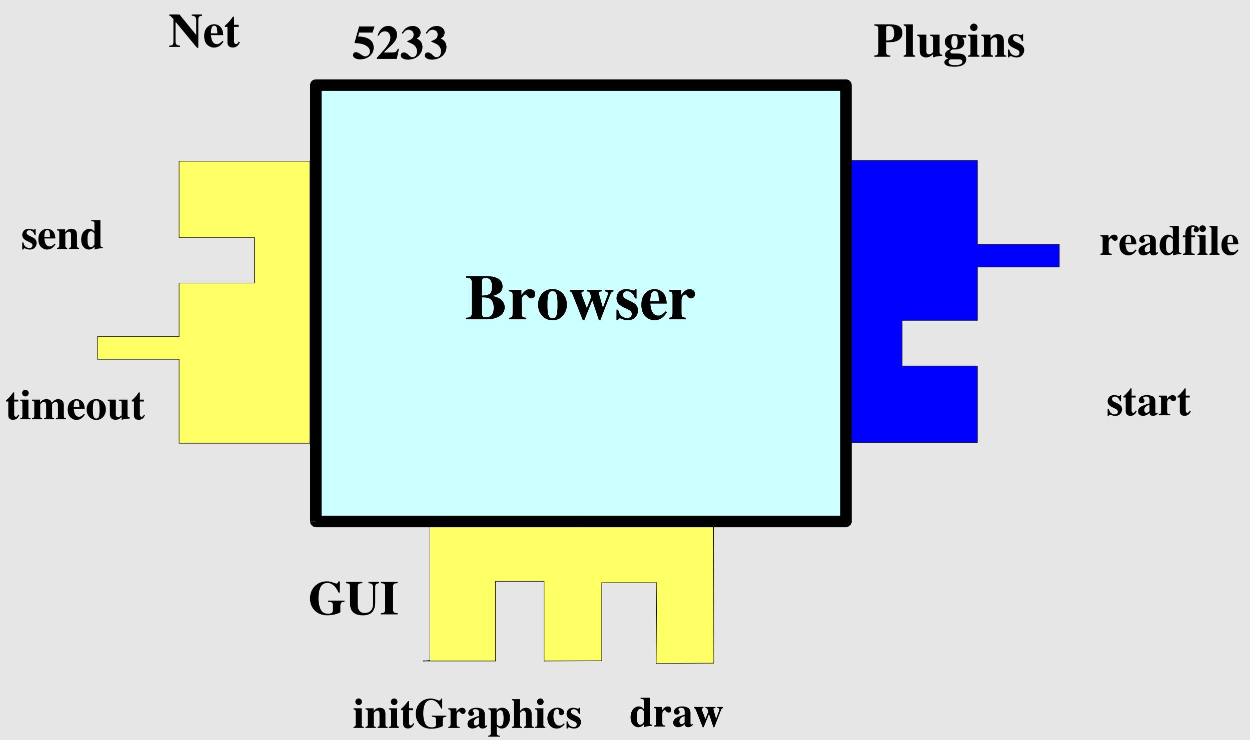


Mixers: regular dependency **Pluggers: hot deployment dependency**

Interfaces are Bi-directional: Imports, Exports

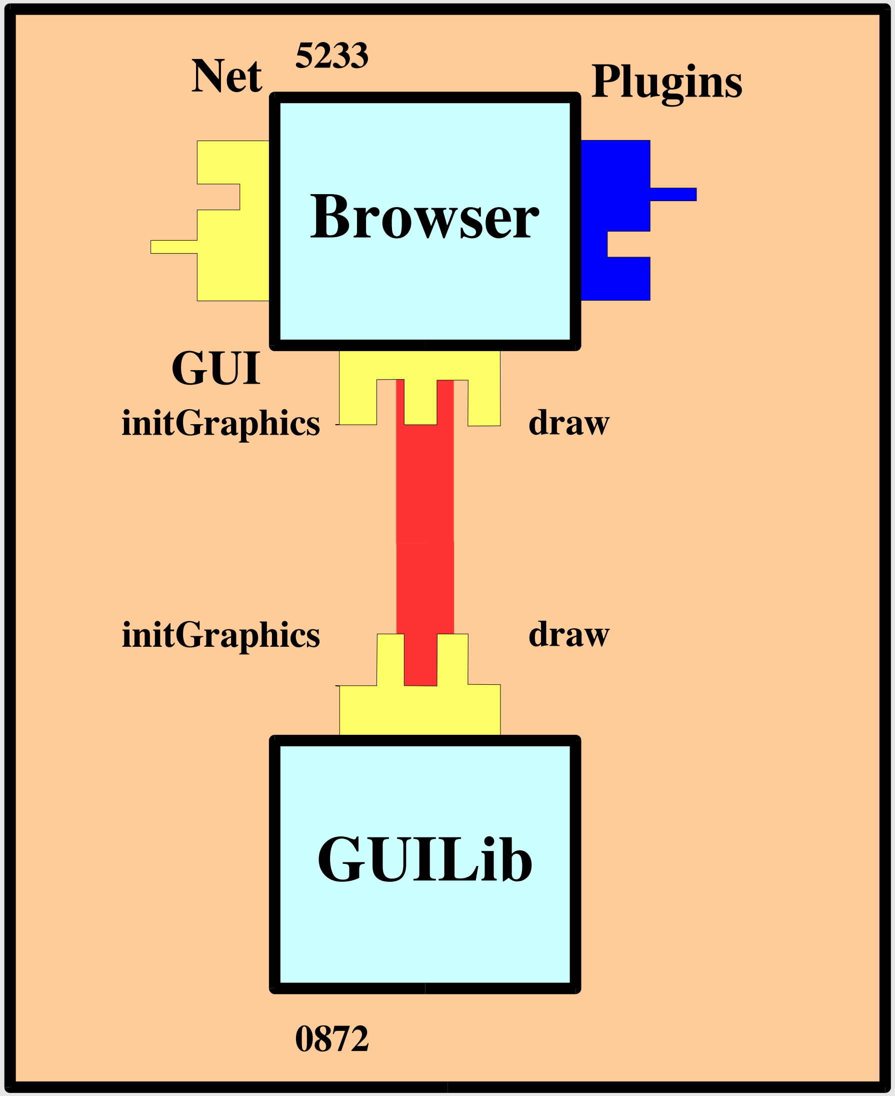


Multiple Interfaces

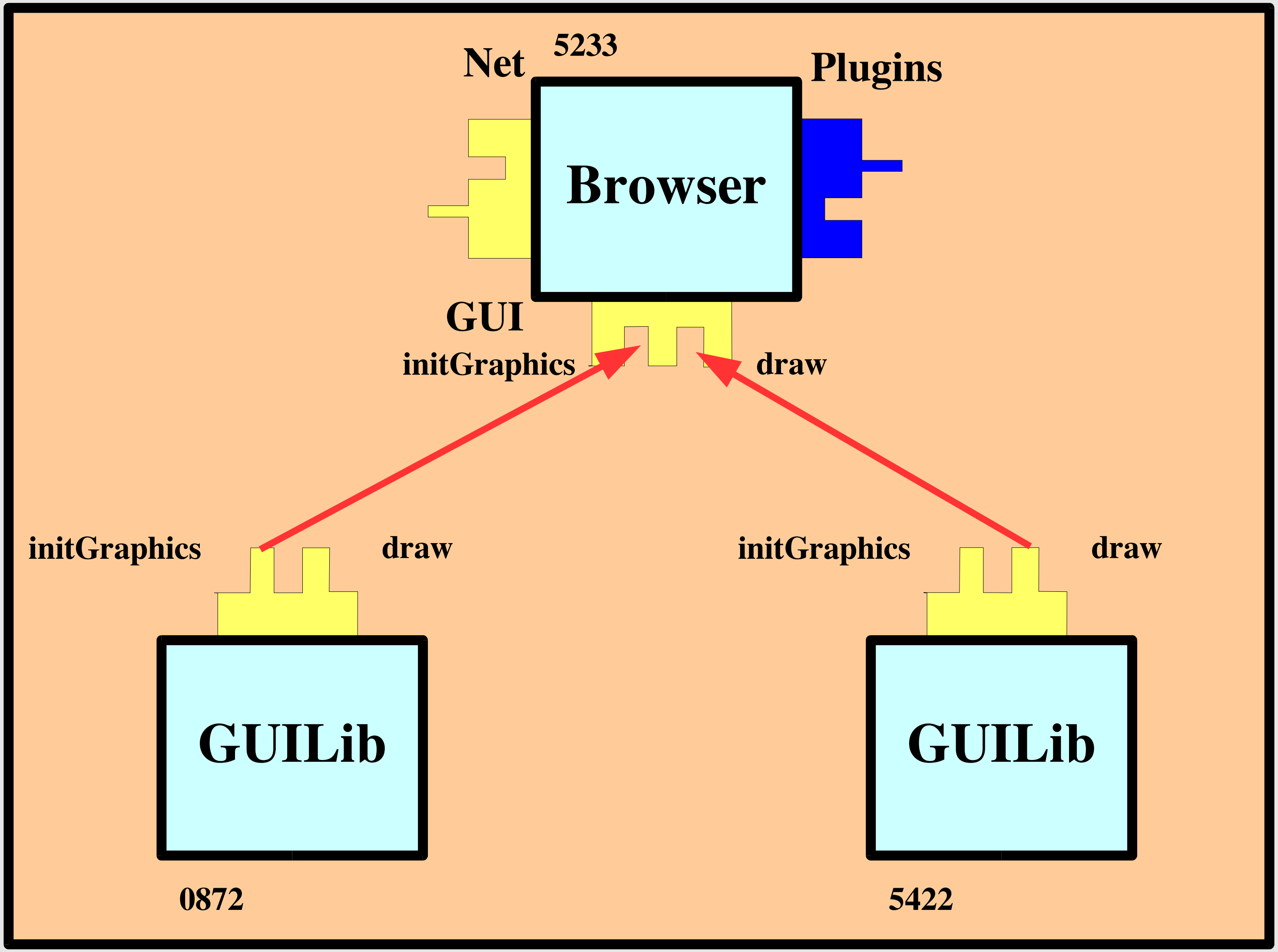


- Name management is crucial for deployment.
- Avoid global name clashes

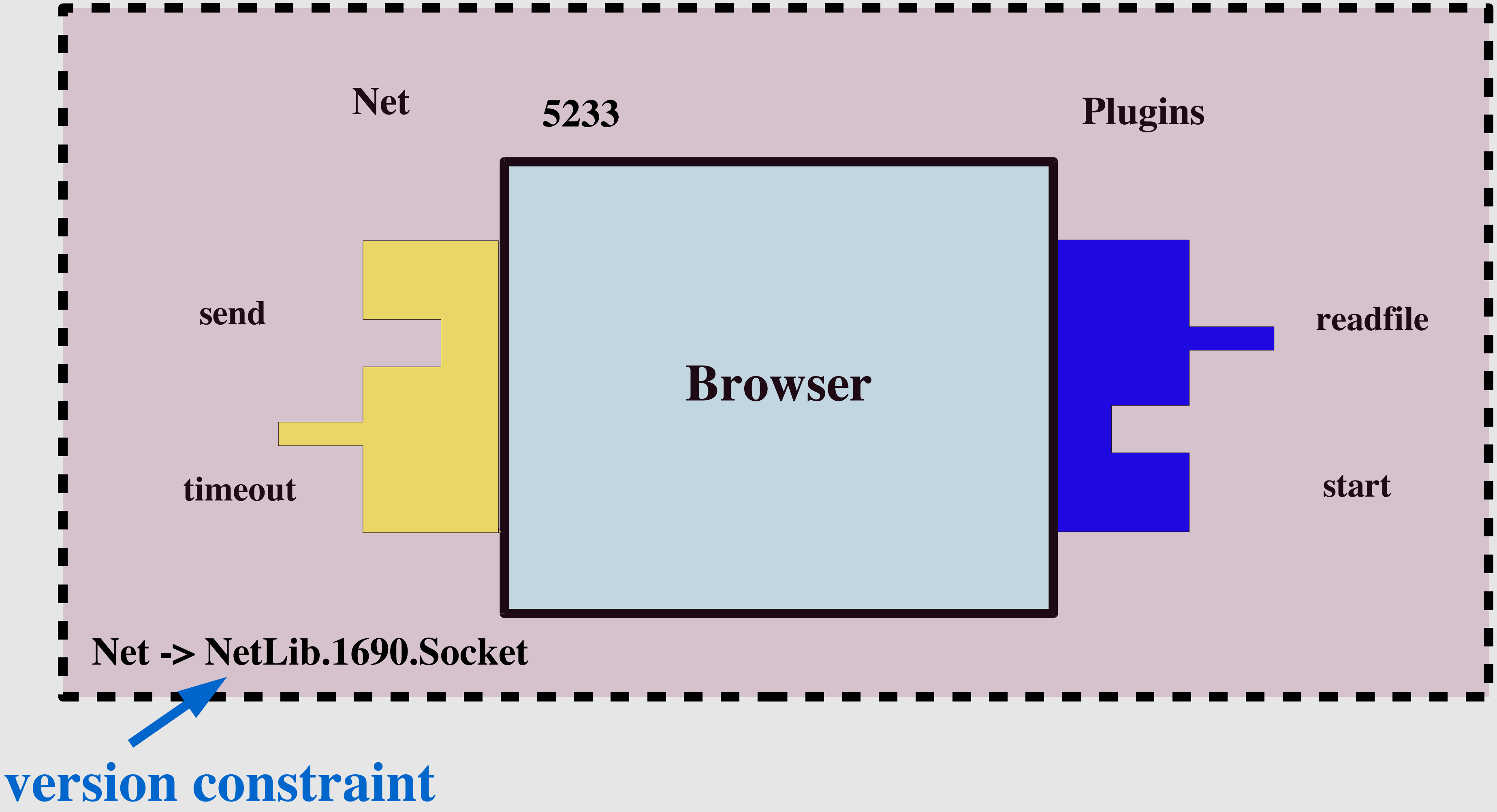
Interface: Unit of Versioning Dependencies



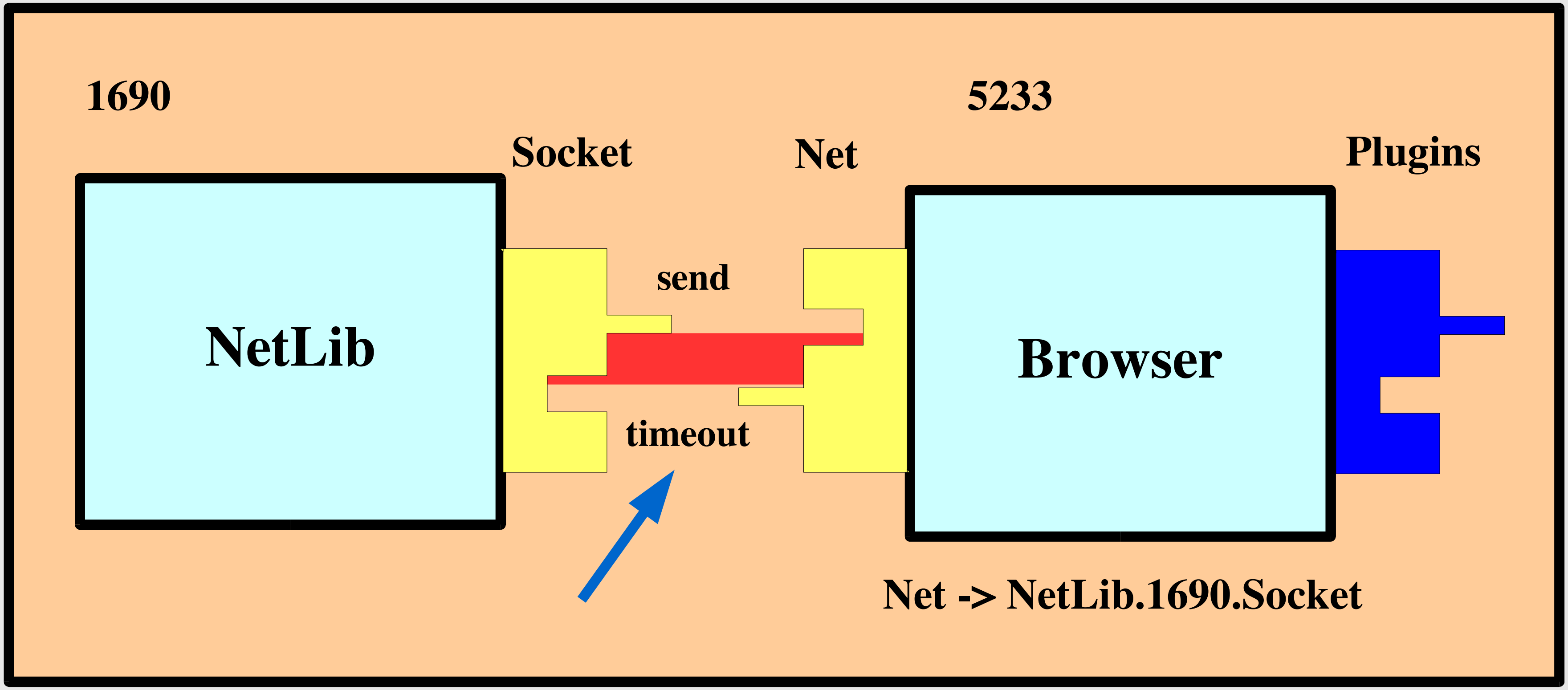
What is NOT Possible...



Assemblages in Shipped Form

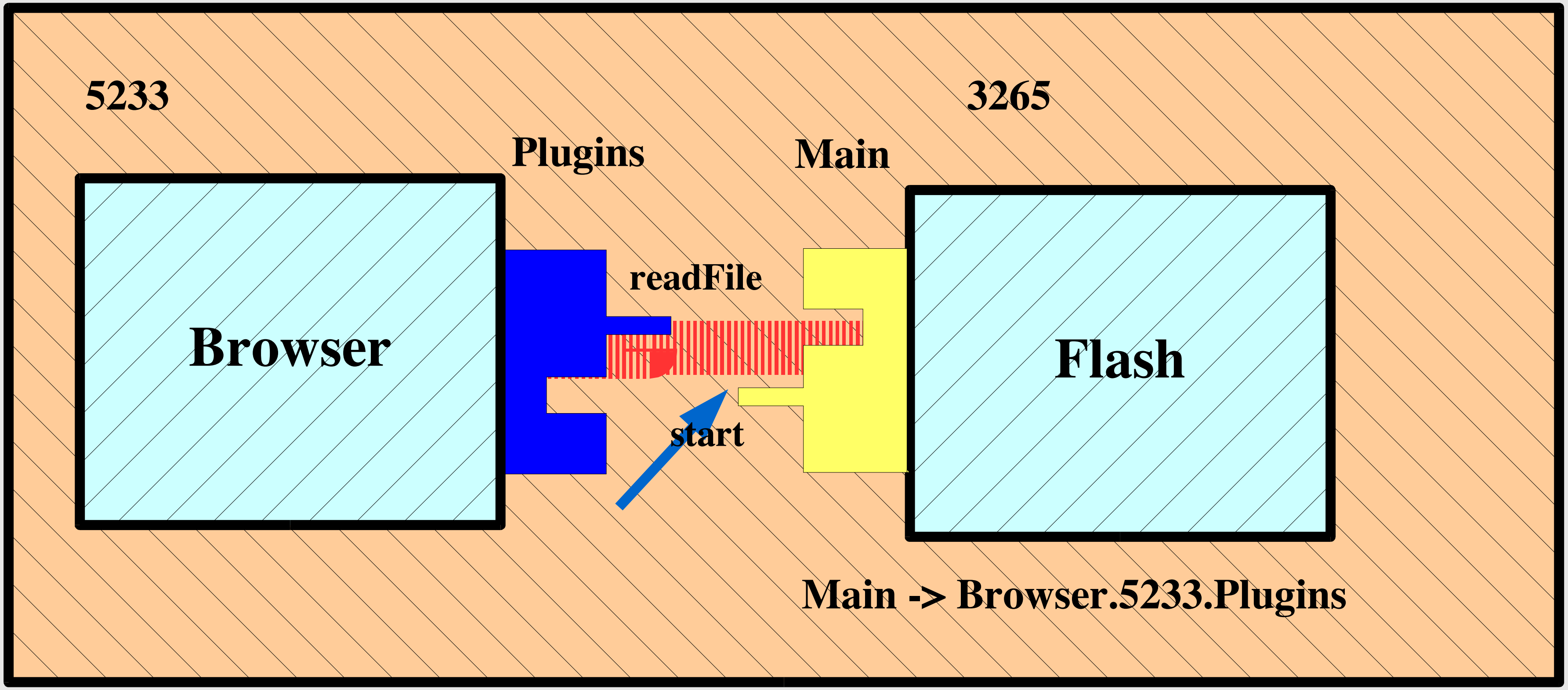


Component Wiring: Mixing



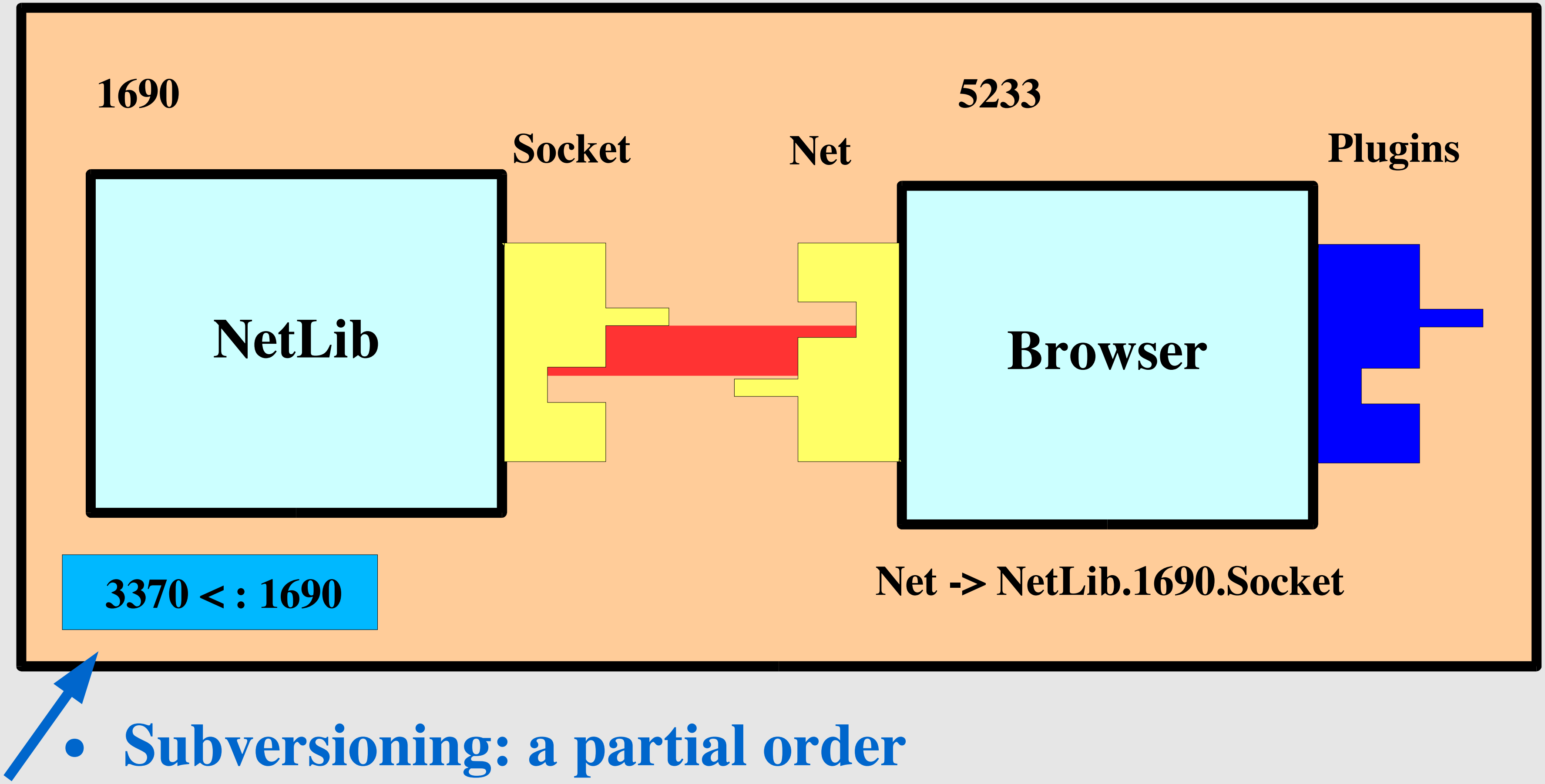
- Between a pair of mixers
- Matching of functionalities
- Matching of version constraints

Component Wiring: Plugging



- Wiring at hot deployment time
- Between a plugger and a mixer
- Matching of functionalities
- Matching of version constraints

Compatibility Set

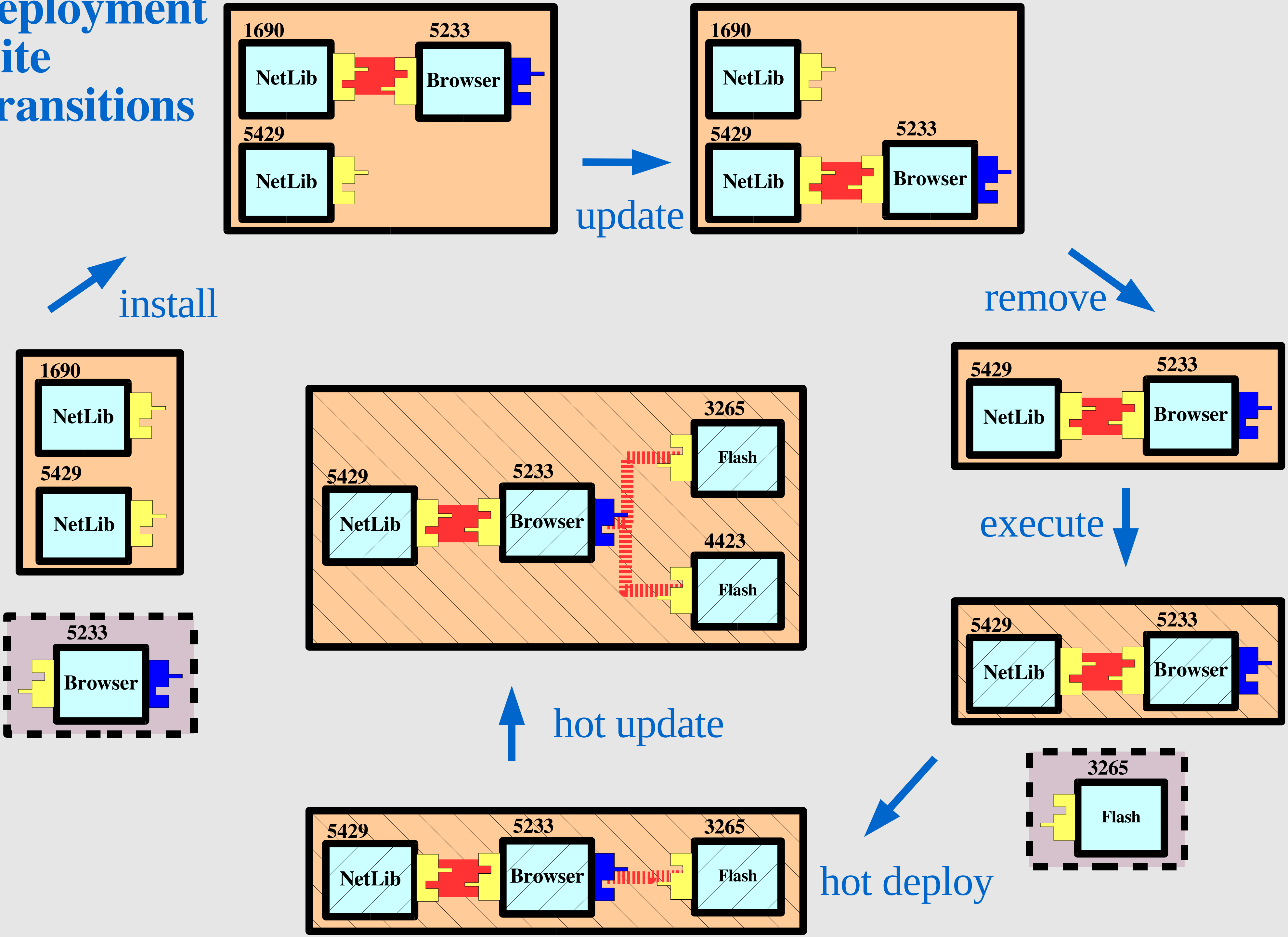


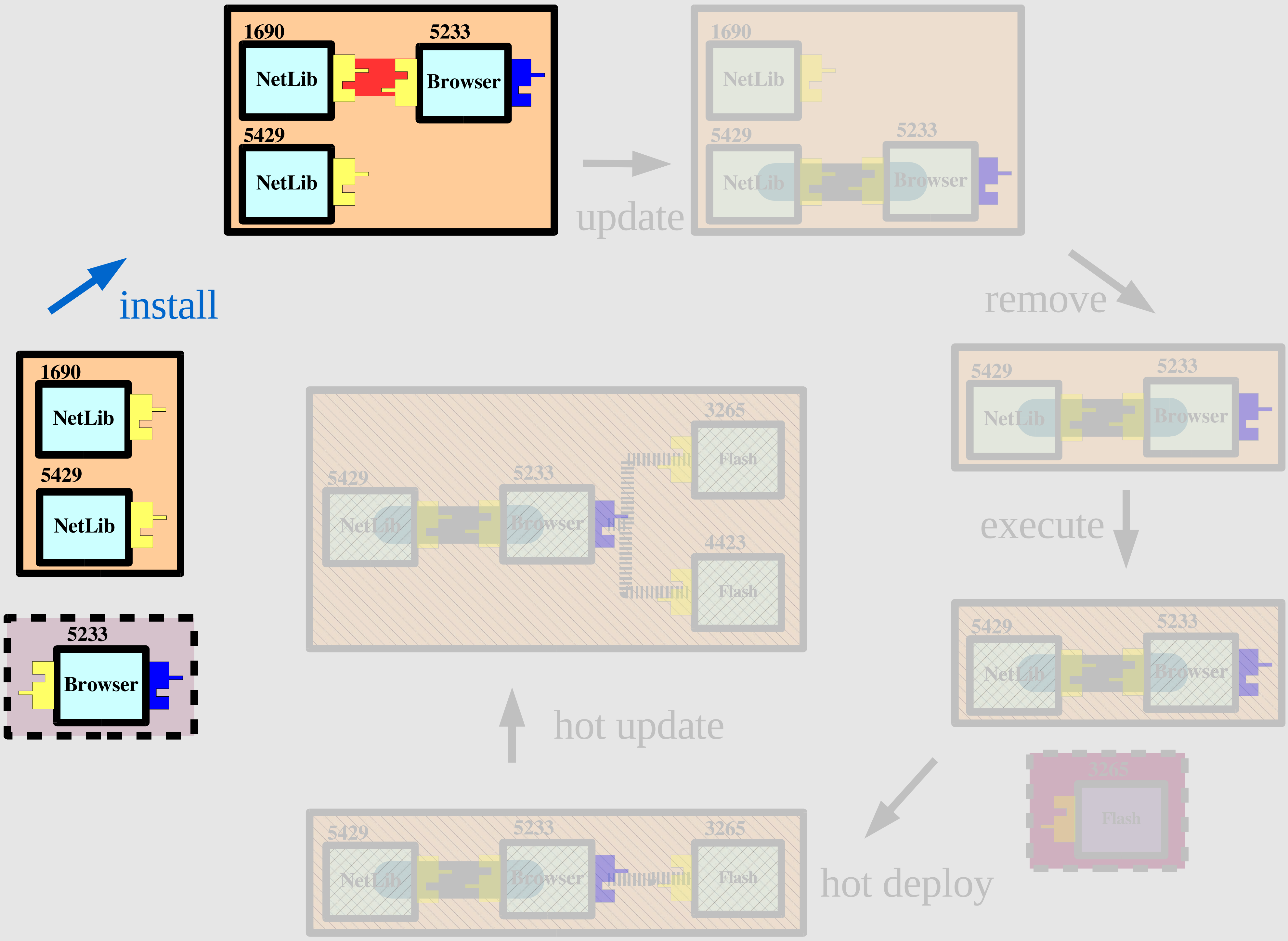
- Subversioning: a partial order
- We do not hardcode the strategy on how two versions are semantically compatible

Act 2:

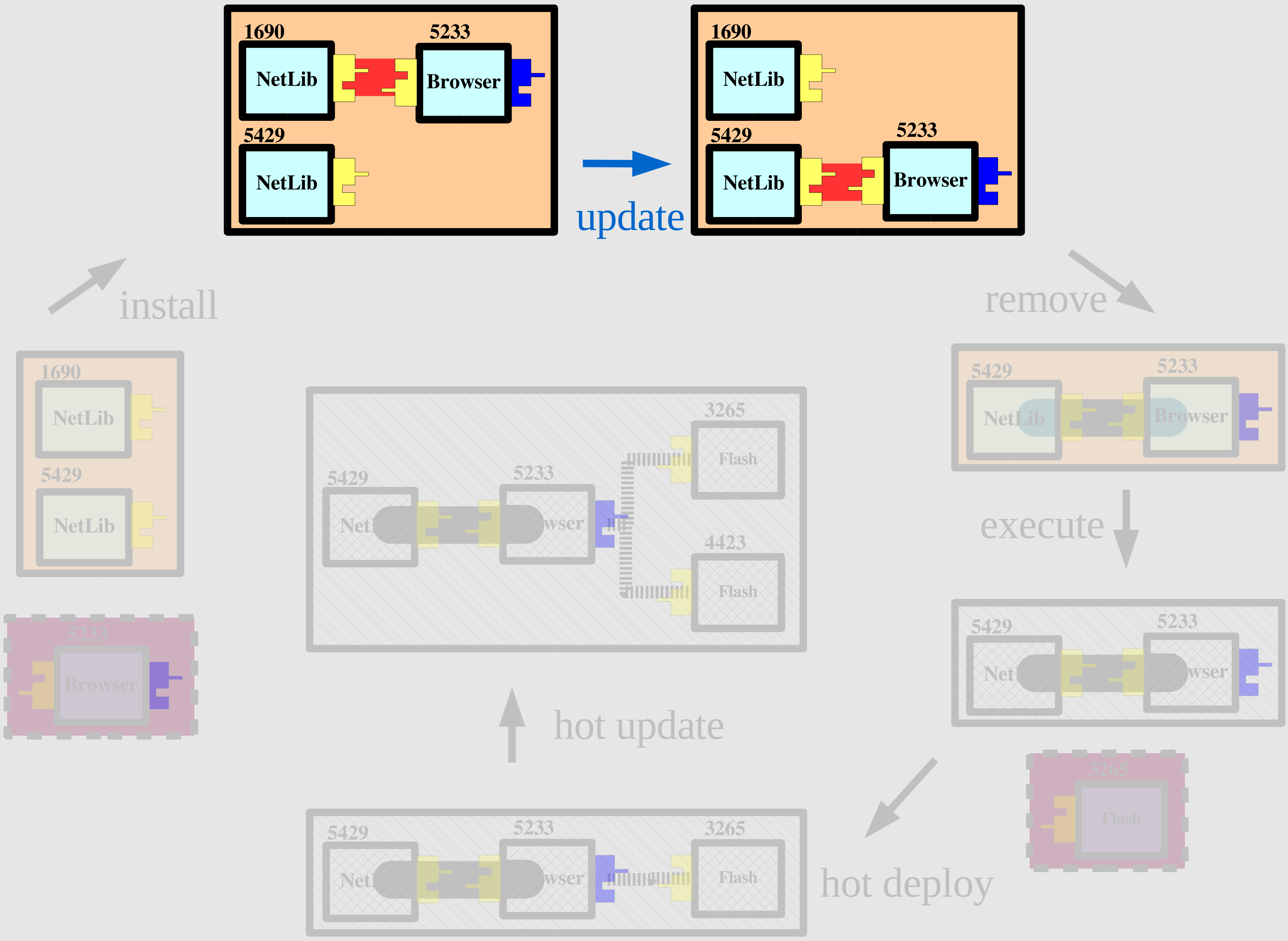
Component Deployment Lifecycle

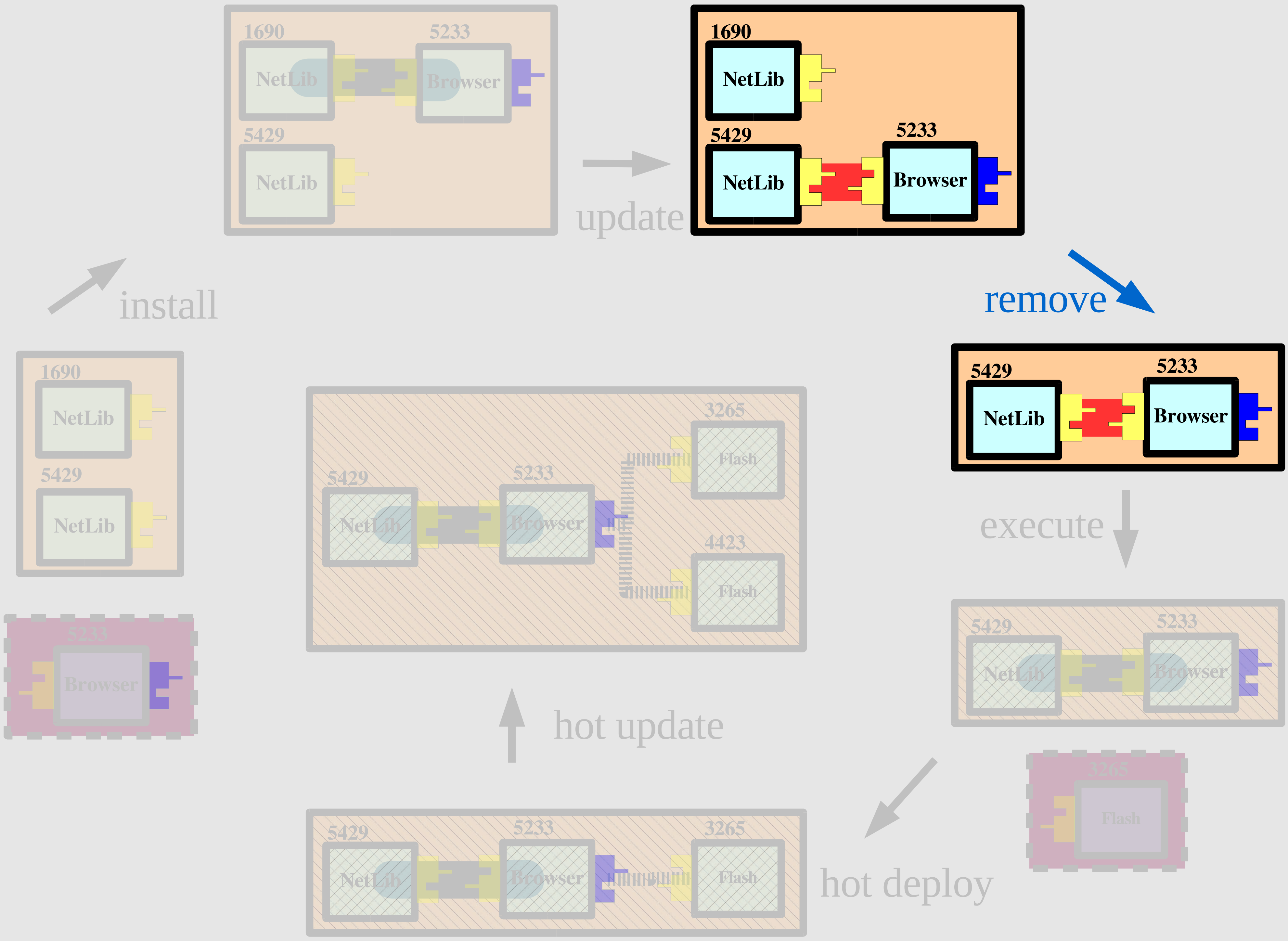
Deployment Site Transitions



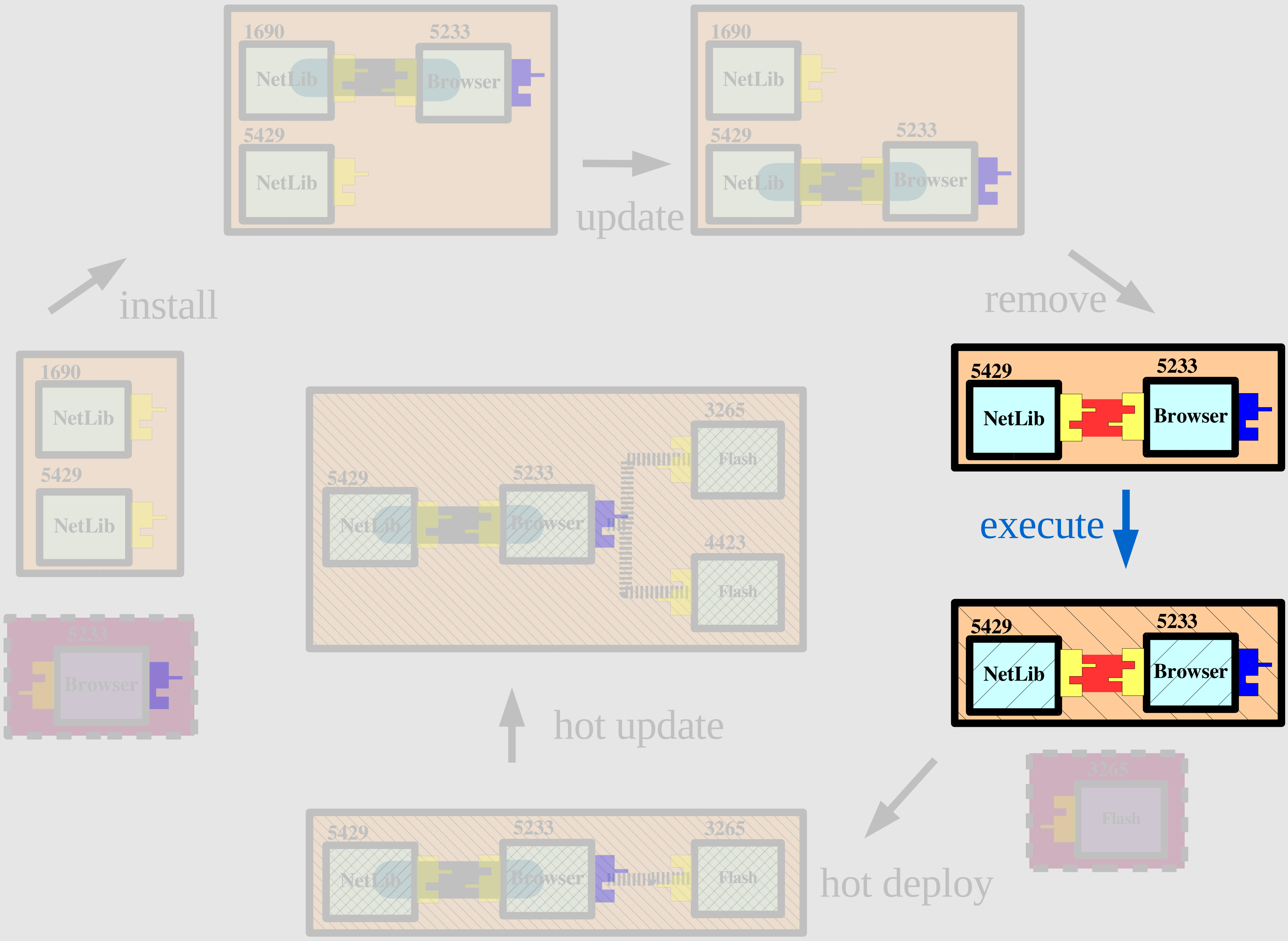


1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
remove
5429
NetLib
5233
Browser
execute
5429
NetLib
5233
Browser
3265
Flash
hot deploy
5429
NetLib
5233
Browser
3265
Flash
hot update
5429
NetLib
5233
Browser
3265
Flash
4423
Flash
install
1690
NetLib
5429
NetLib
5233
Browser

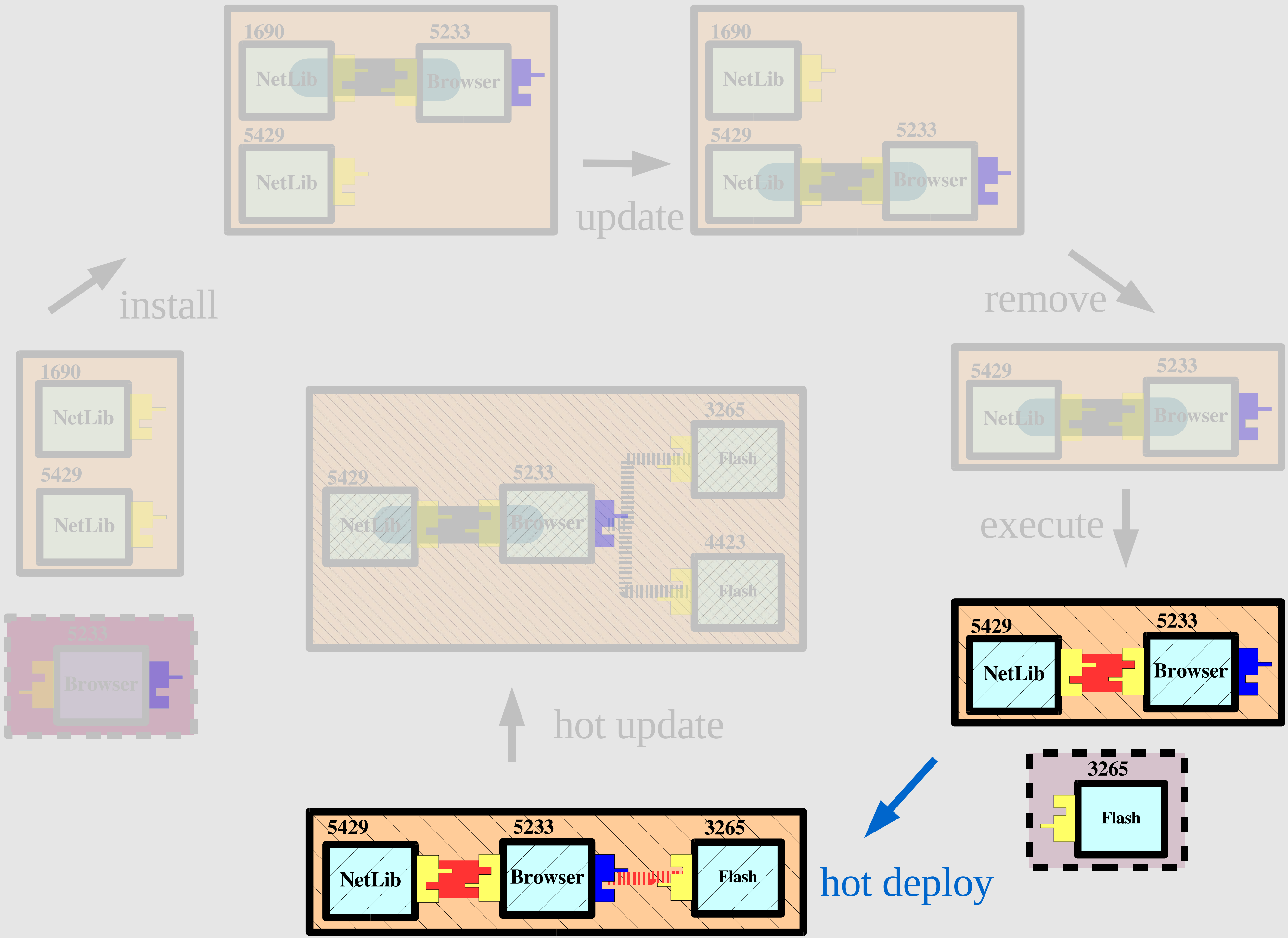

1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
install
remove
1690
NetLib
5429
NetLib
5233
Browser
5429
NetLib
5233
Browser
execute
3265
Flash
5429
Net
5233
wser
4423
Flash
5429
Net
5233
wser
hot update
3265
Flash
5429
Net
5233
wser
3265
Flash
hot deploy



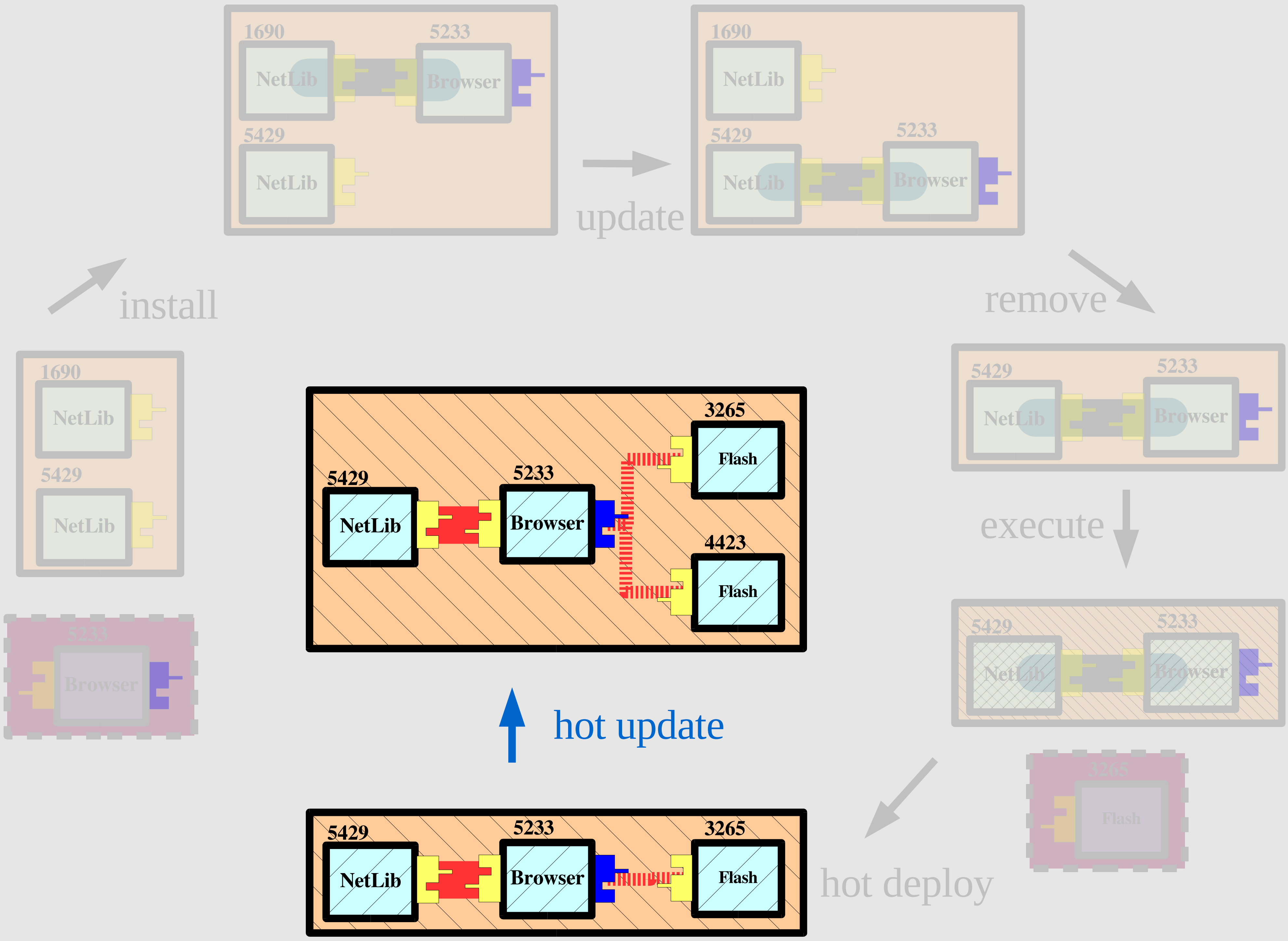
1690
5233
NetLib
Browser
5429
NetLib
update
1690
NetLib
5429
5233
NetLib
Browser
install
remove
5429
5233
NetLib
Browser
1690
NetLib
5429
NetLib
3265
Flash
5429
5233
NetLib
Browser
4423
Flash
execute
5233
Browser
5429
5233
NetLib
Browser
hot update
3265
Flash
5429
5233
3265
NetLib
Browser
Flash
hot deploy



1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
install
remove
1690
NetLib
5429
NetLib
5429
NetLib
5233
Browser
execute
5429
NetLib
5233
Browser
5233
Browser
5429
NetLib
5233
Browser
3265
Flash
4423
Flash
hot update
5429
NetLib
5233
Browser
3265
Flash
hot deploy
3265
Flash

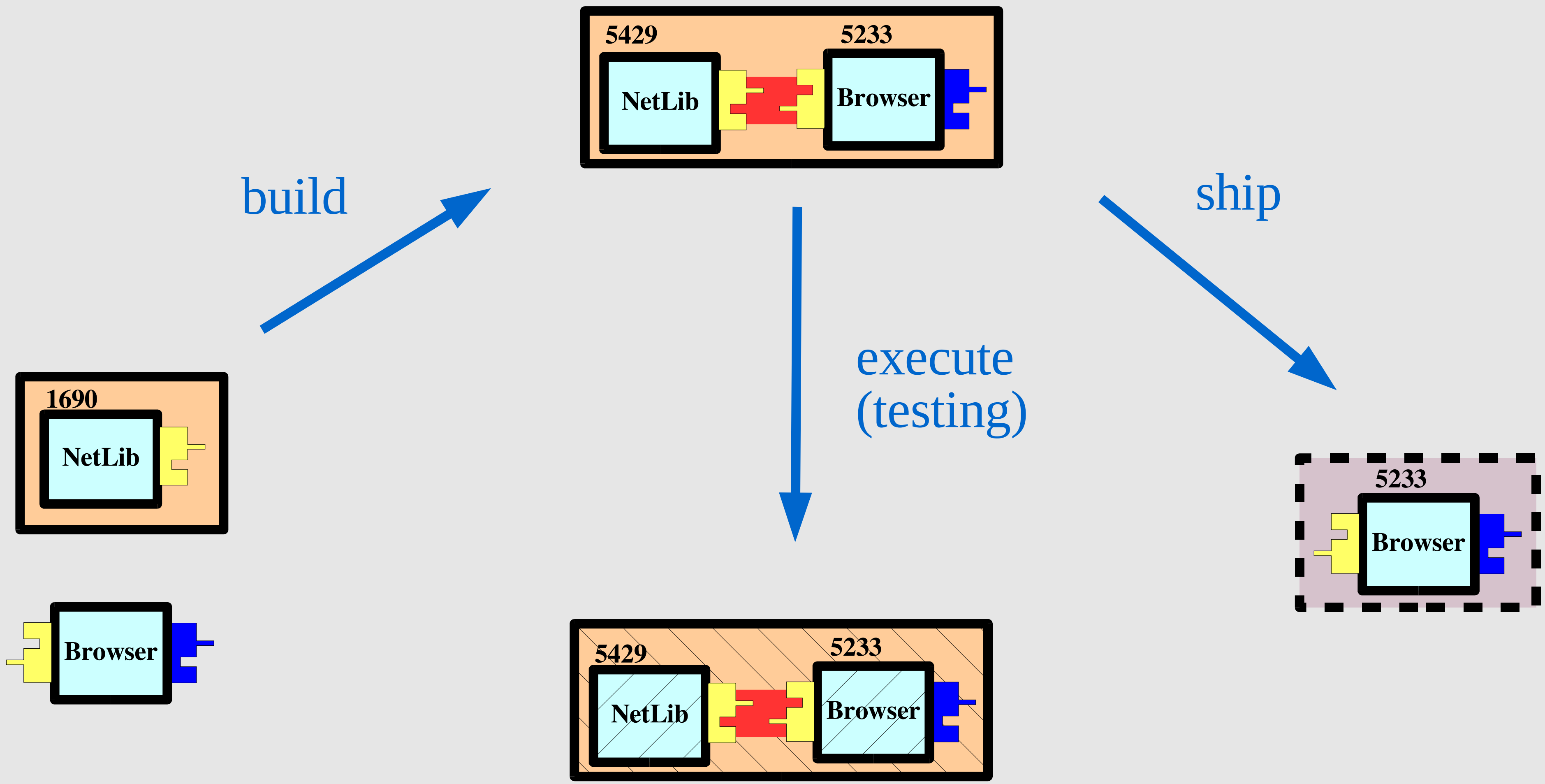


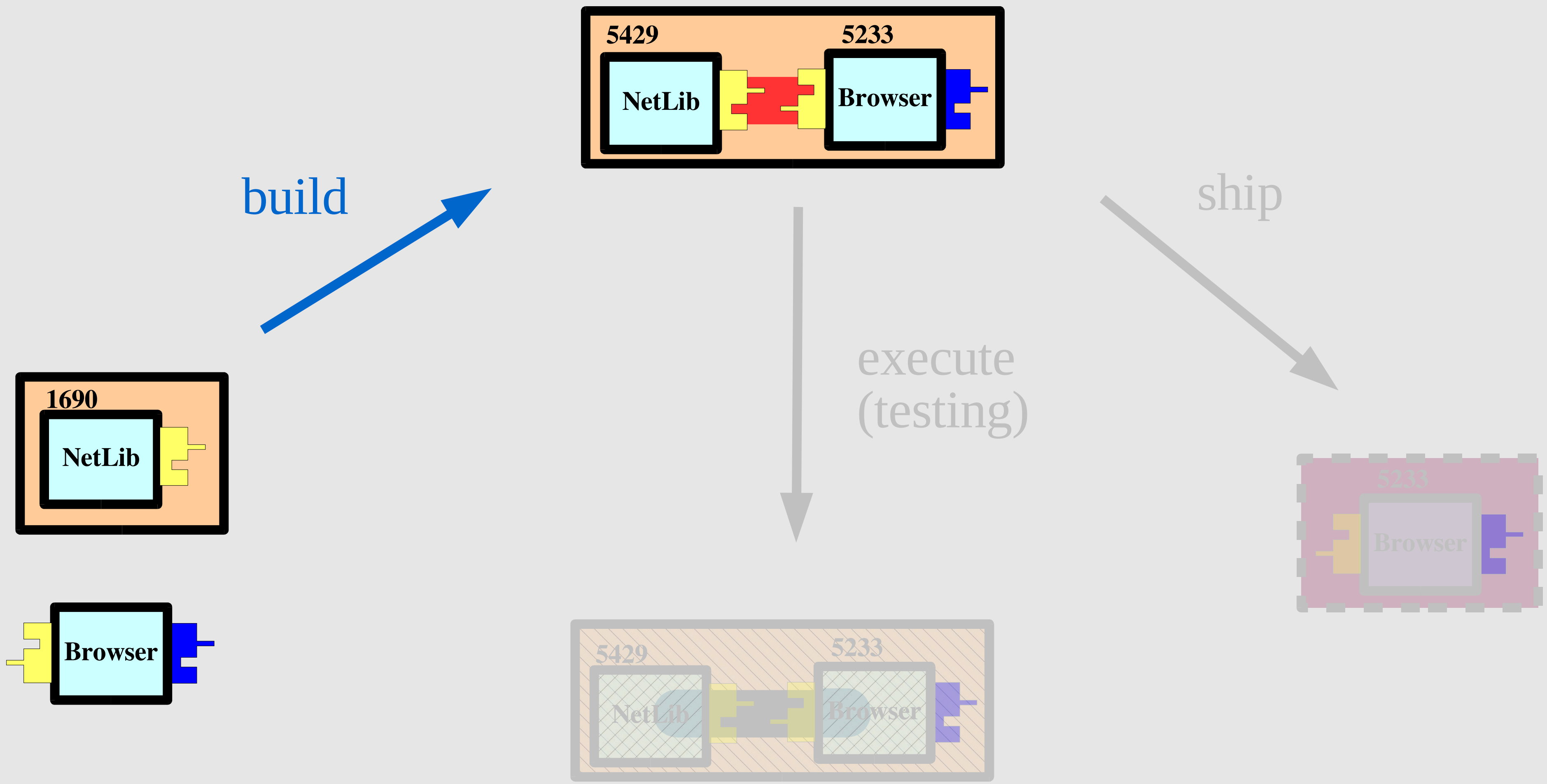
1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
install
remove
1690
NetLib
5429
NetLib
5233
Browser
5429
NetLib
5233
Browser
3265
Flash
4423
Flash
execute
5429
NetLib
5233
Browser
hot update
3265
Flash
5429
NetLib
5233
Browser
3265
Flash
hot deploy



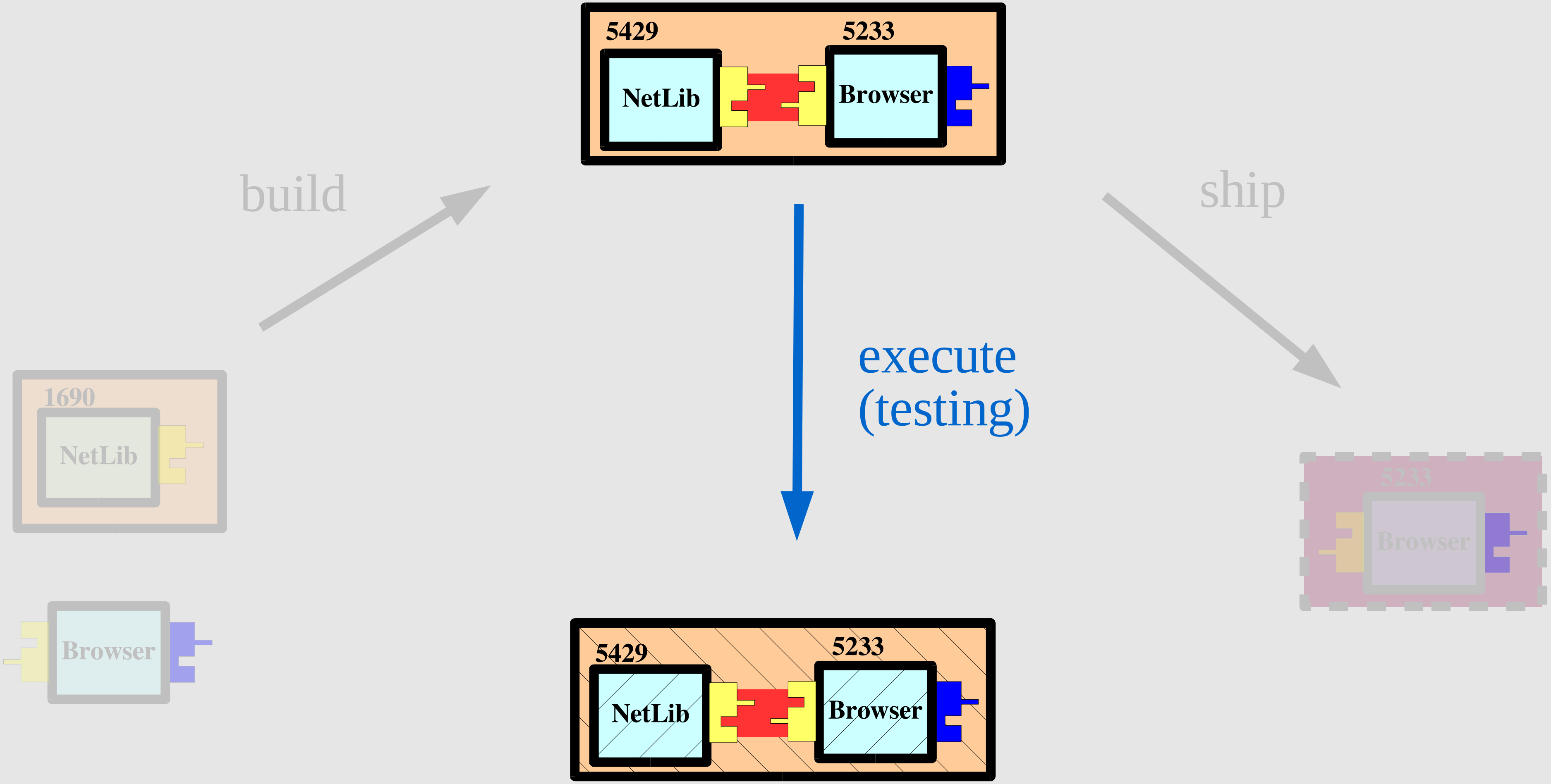
1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
install
remove
1690
NetLib
5429
NetLib
5429
NetLib
5233
Browser
5233
Browser
3265
Flash
5429
NetLib
5233
Browser
4423
Flash
execute
5429
NetLib
5233
Browser
hot update
3265
Flash
5429
NetLib
5233
Browser
3265
Flash
hot deploy

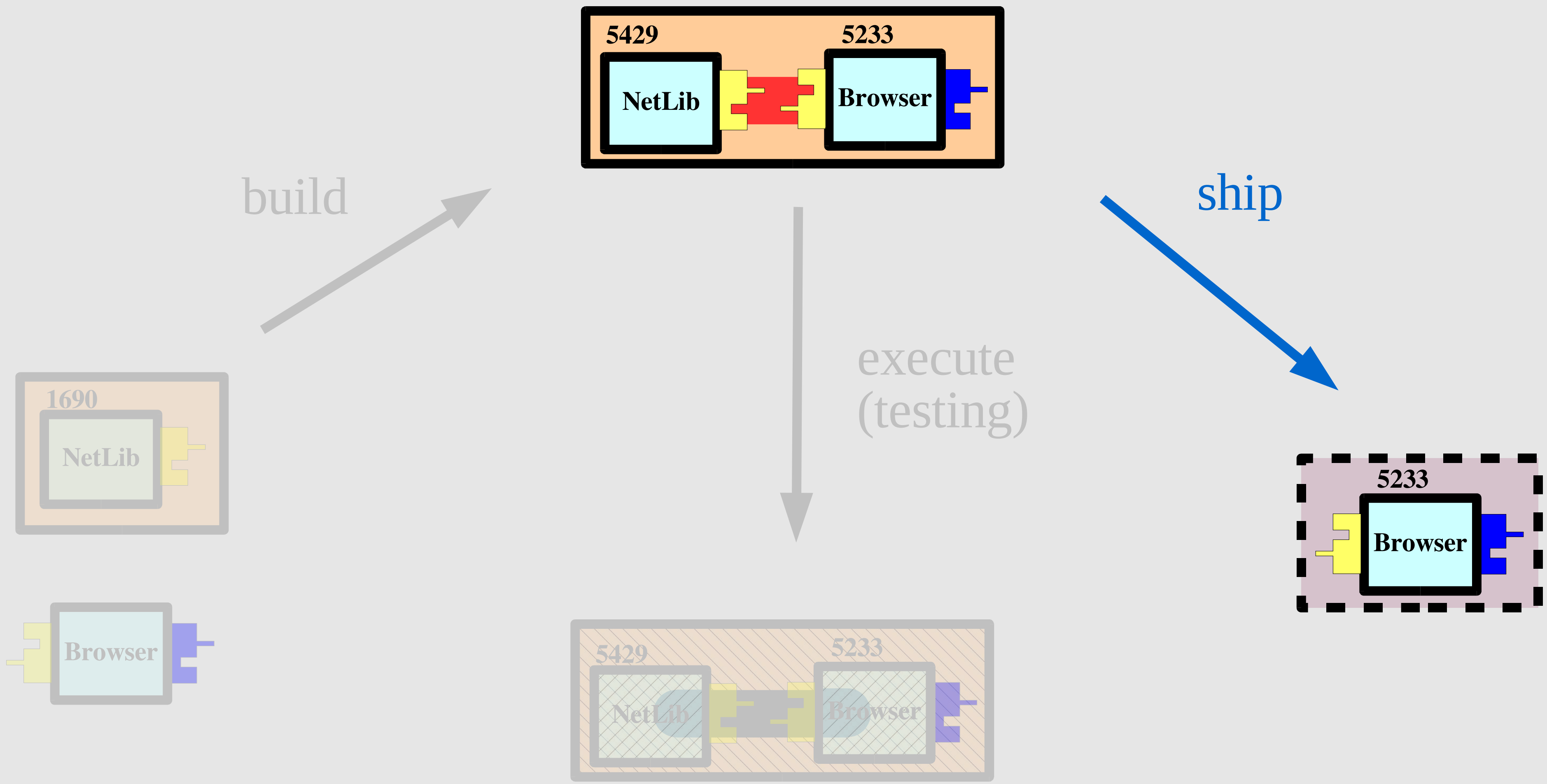
Development Site Transitions





1690
NetLib
Browser
build
5429
5233
NetLib
Browser
ship
execute
(testing)
5233
Browser
5429
5233
NetLib
Browser

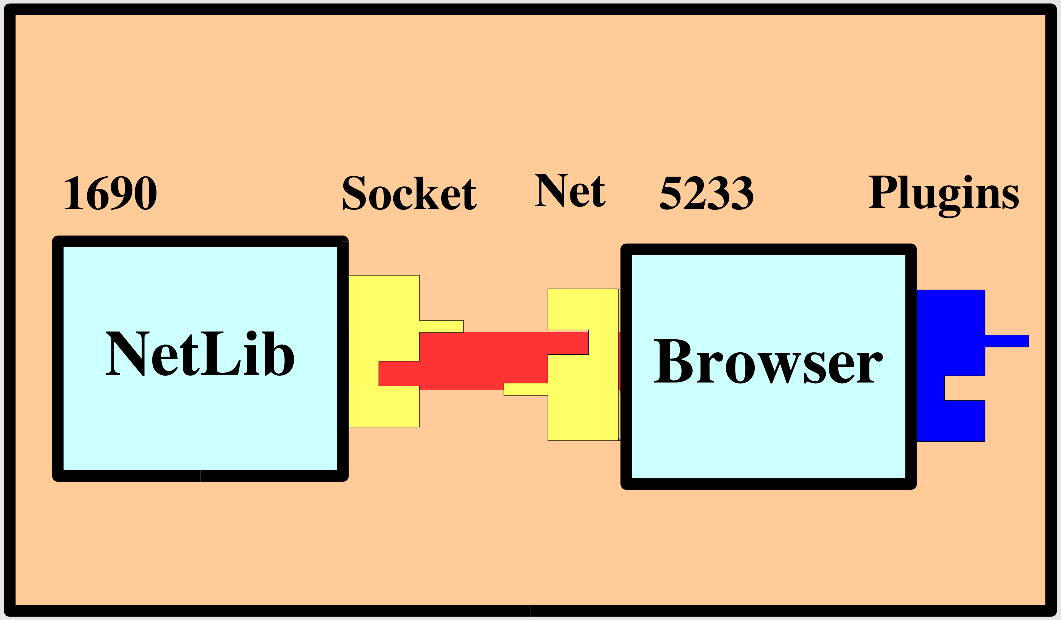

5429
5233
NetLib
Browser
build
ship
execute
(testing)
1690
NetLib
Browser
5233
Browser
5429
5233
NetLib
Browser


5429
5233
NetLib
Browser
build
ship
execute
(testing)
1690
NetLib
Browser
5233
Browser
5429
5233
NetLib
Browser

Formalism Choice

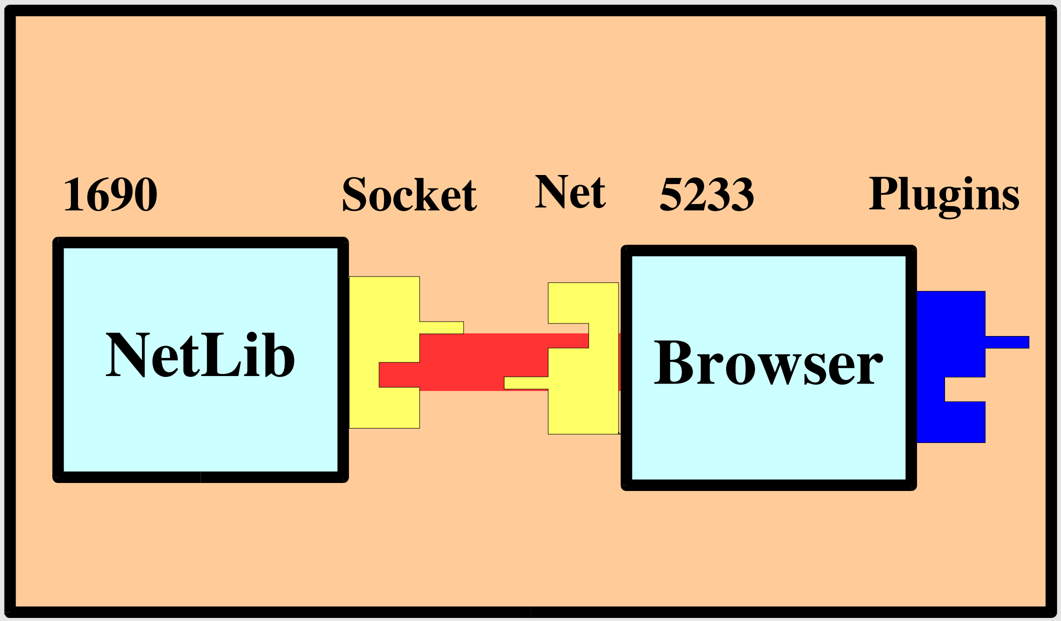
- Labelled Transition System (LTS) for deployment operations
  - Each transition step is an application buildbox evolution step
  - Labels are "commands" which deployment system users can trigger
- Run-time behaviors captured via a minimalistic programming language

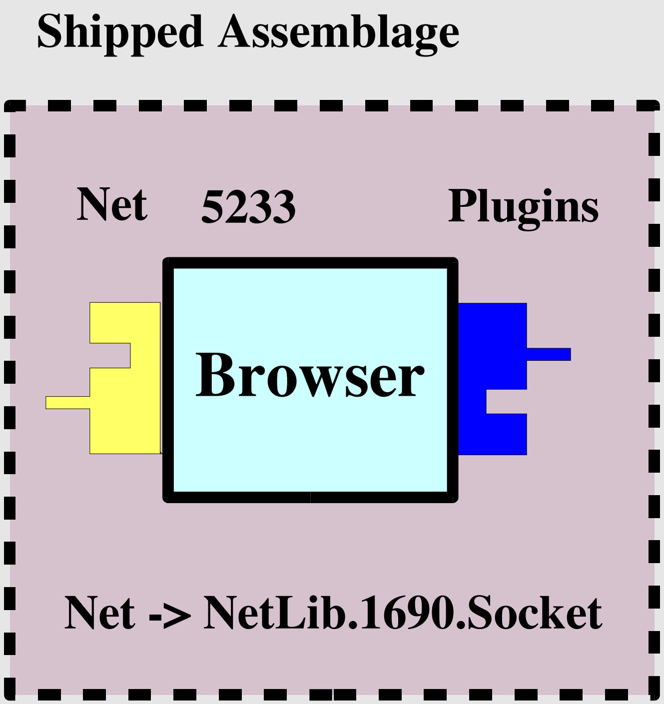
Shipping a Component



ship (*Browser*, 5233, {*Net*})

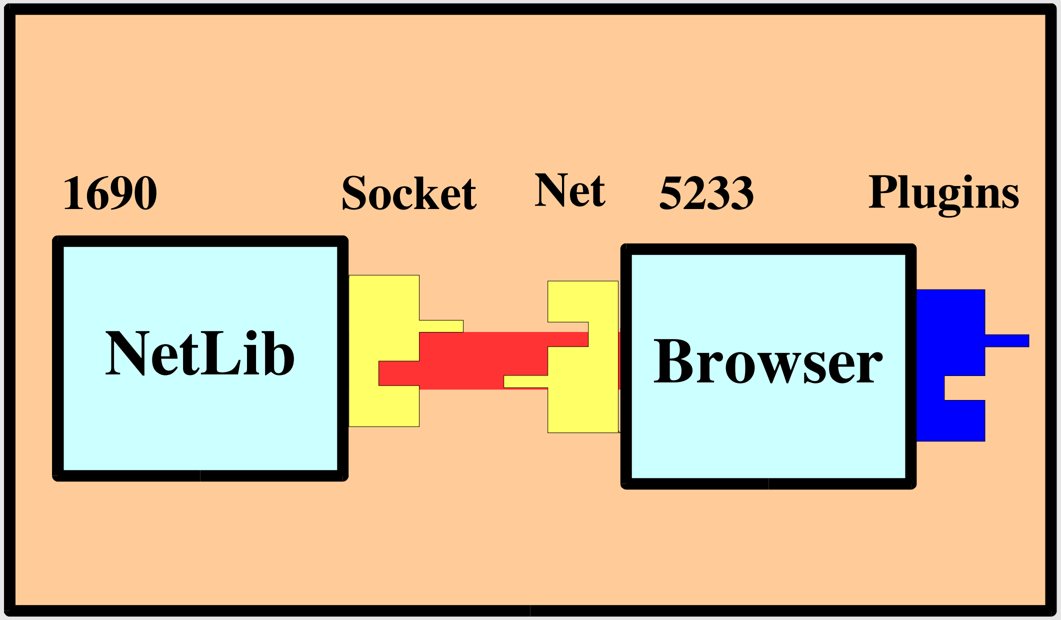
Shipping a Component



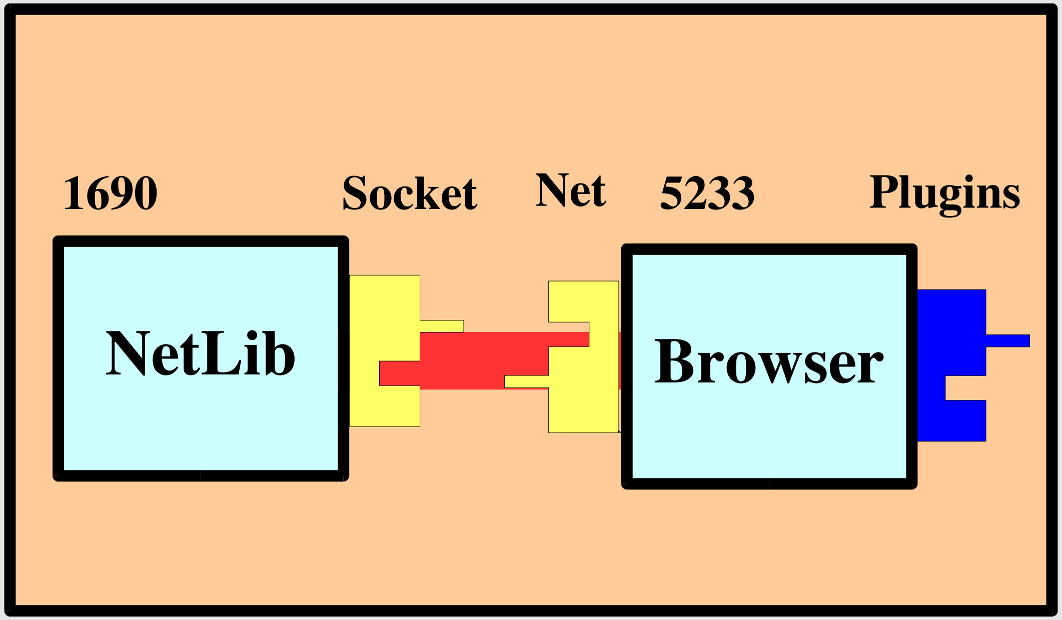


ship (*Browser, 5233, {Net}*)

Why Not Always Ship the Entire Closure?

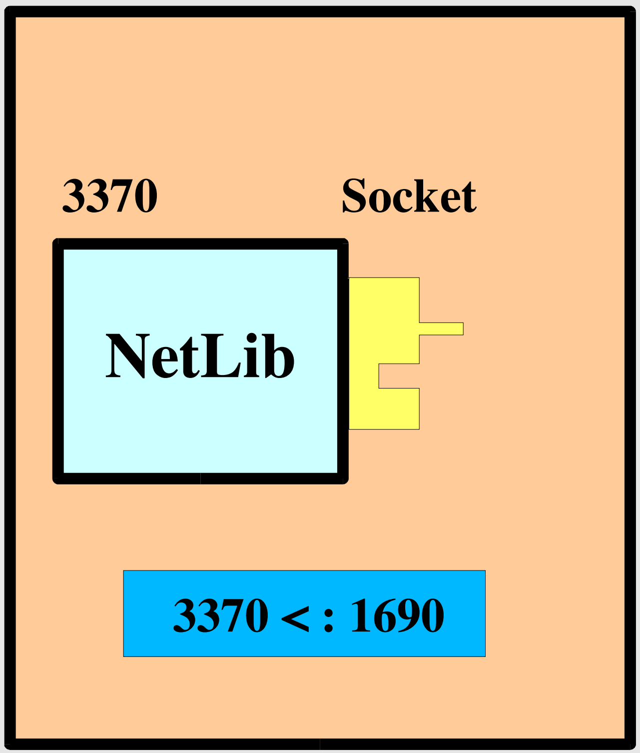


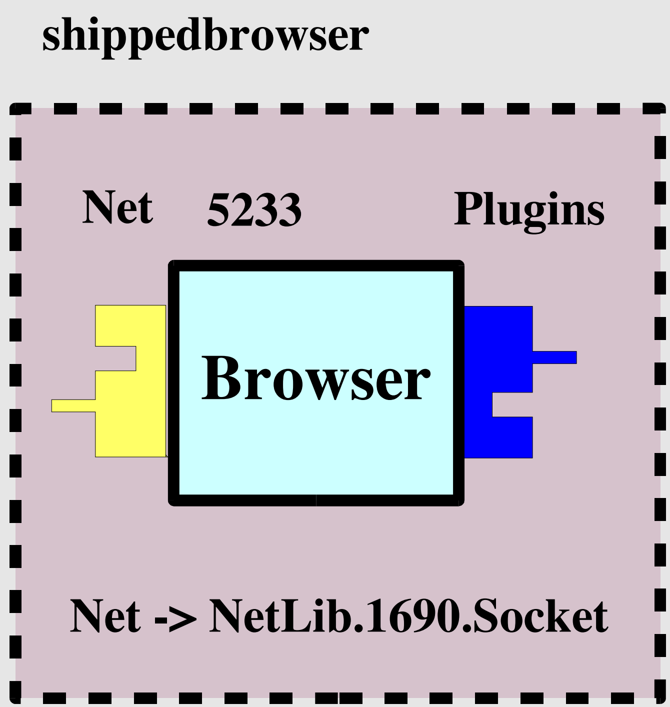
Why Not Always Ship the Entire Closure?



- Components are independently deployable units!
  - Off-the-shelf commercial components, libraries
  - Updates, patches
- Sometimes not realistic, such as native code

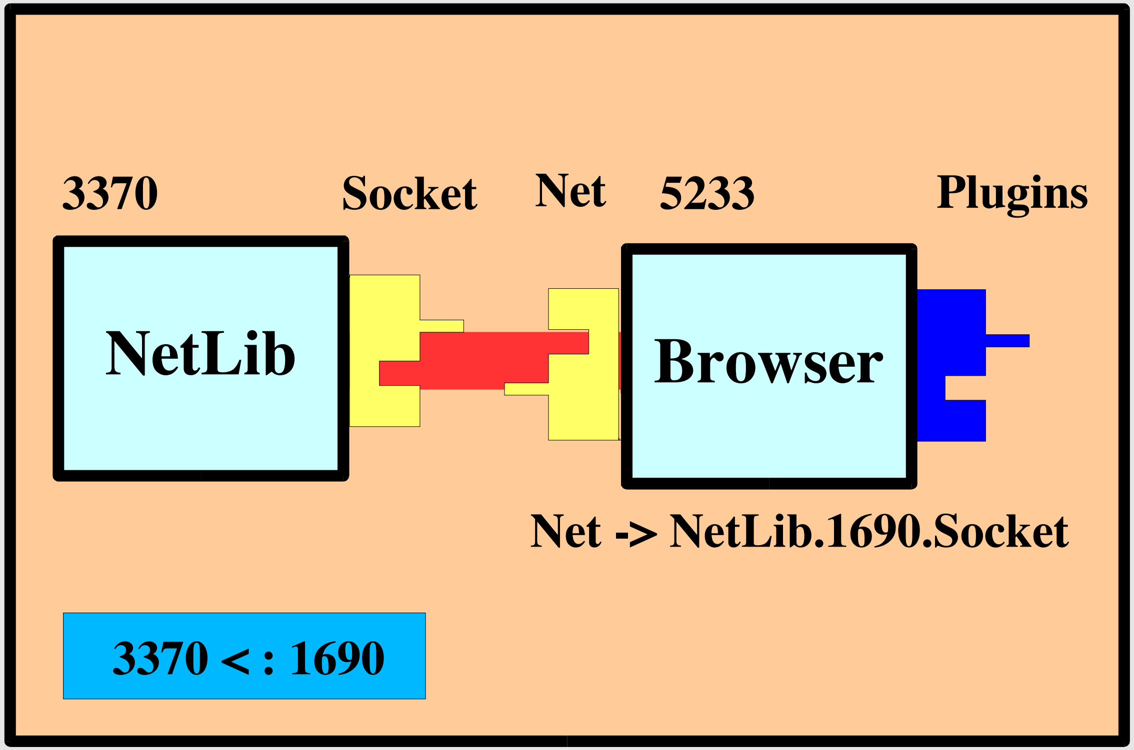
Installing a Component





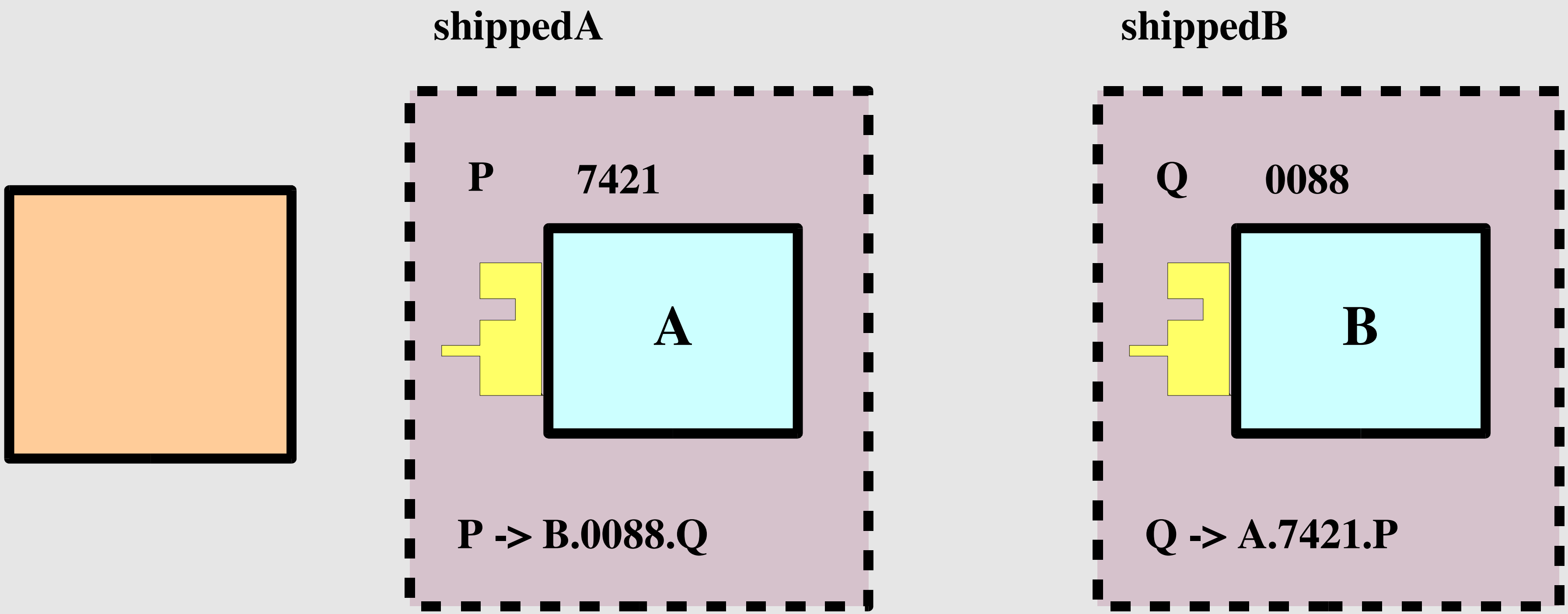
install *(shippedbrowser)*

Installing a Component



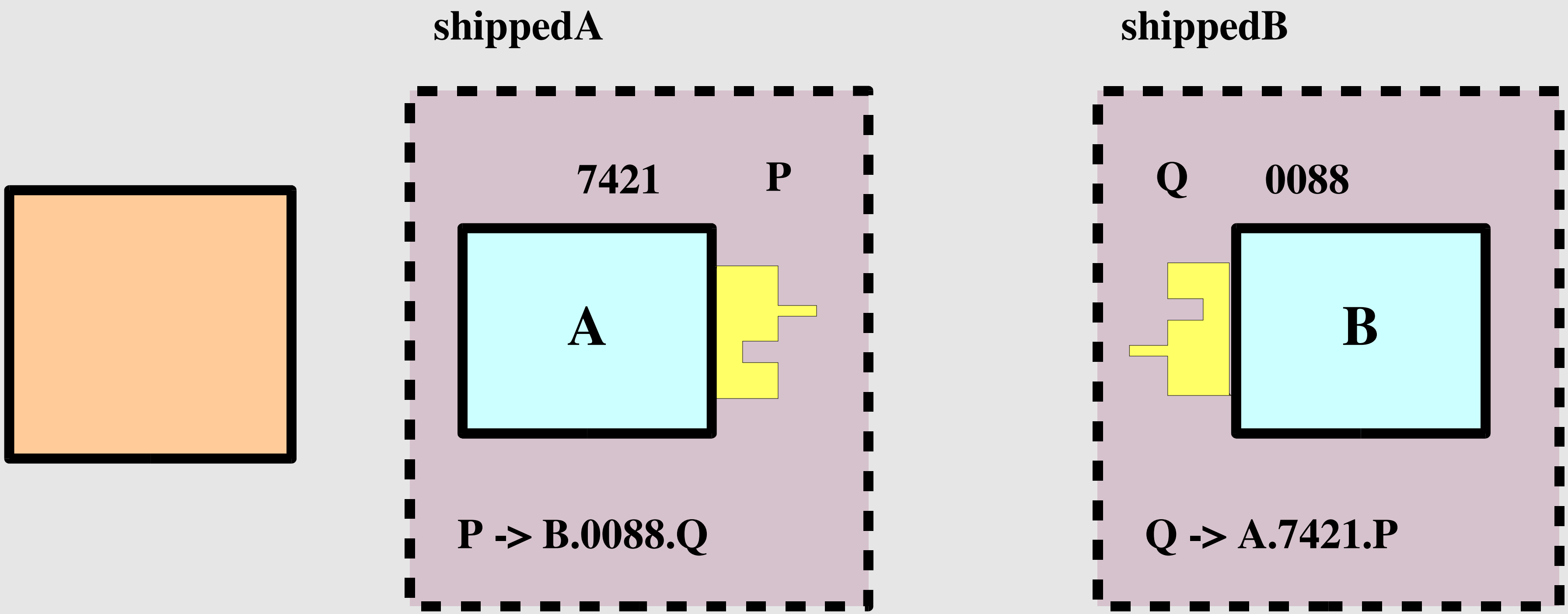
install *(shippedbrowser)*

Cyclic Dependencies



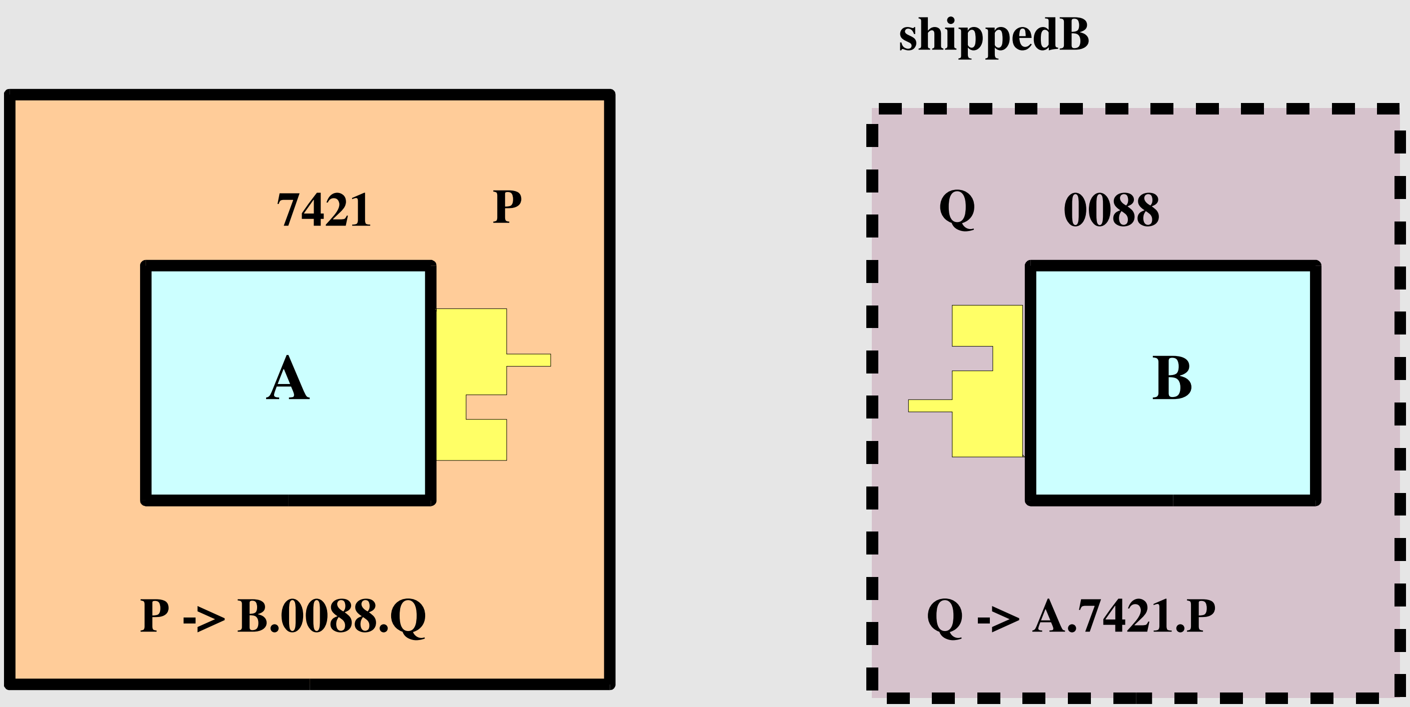
Example: System.dll and System.xml.dll in .NET

Cyclic Dependencies



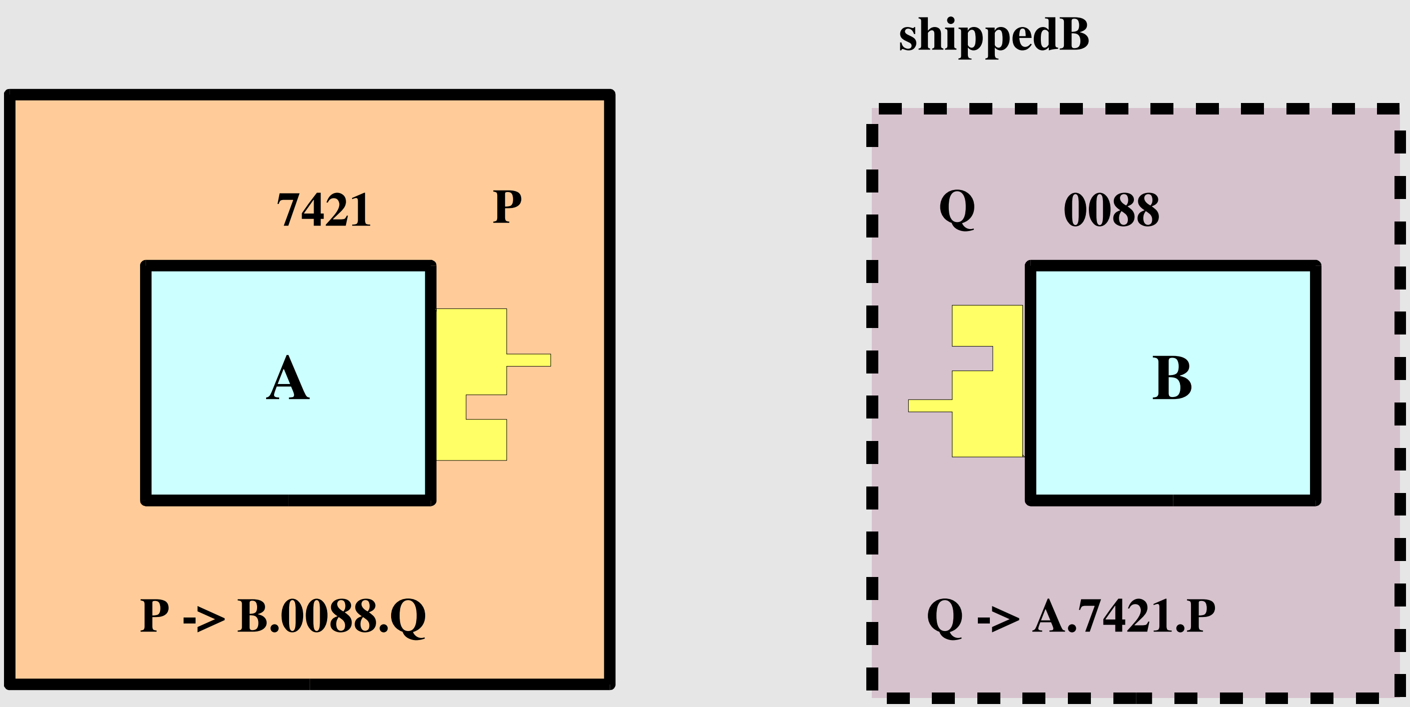
install *(shippedA)*

Cyclic Dependencies



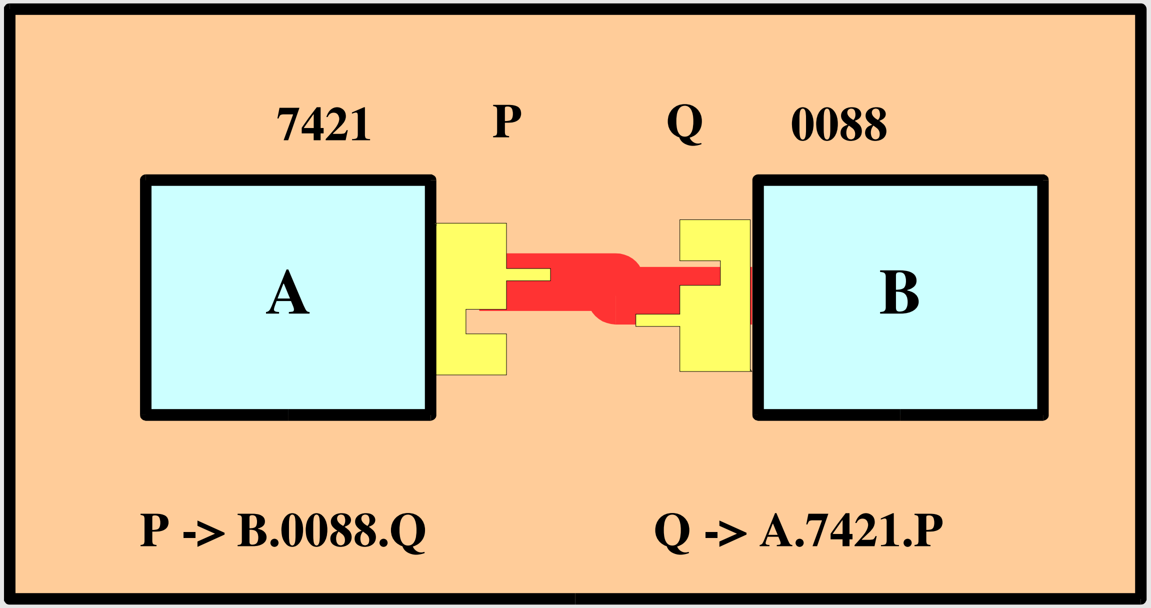
install *(shippedA)*

Cyclic Dependencies



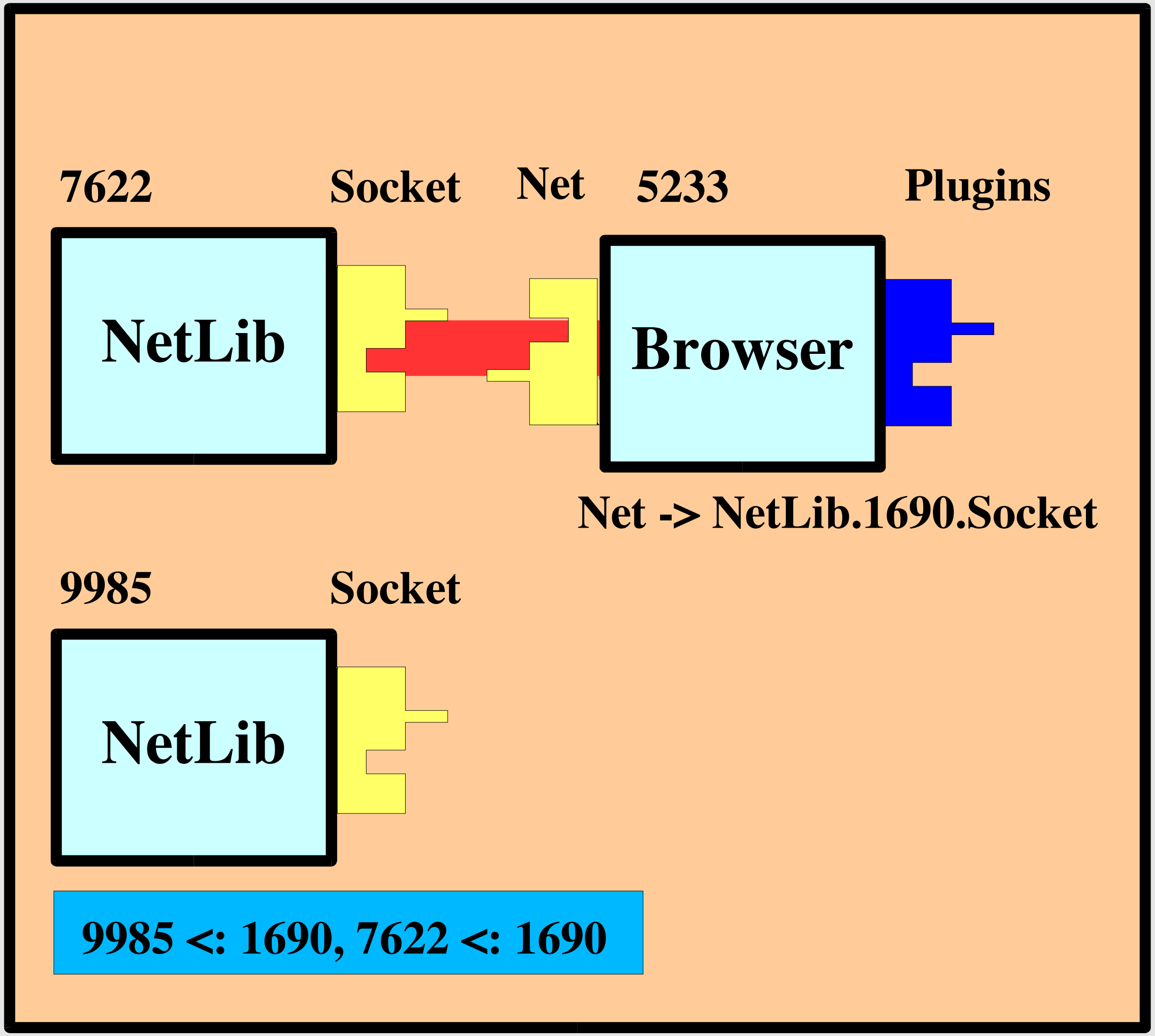
install *(shippedB)*

Cyclic Dependencies



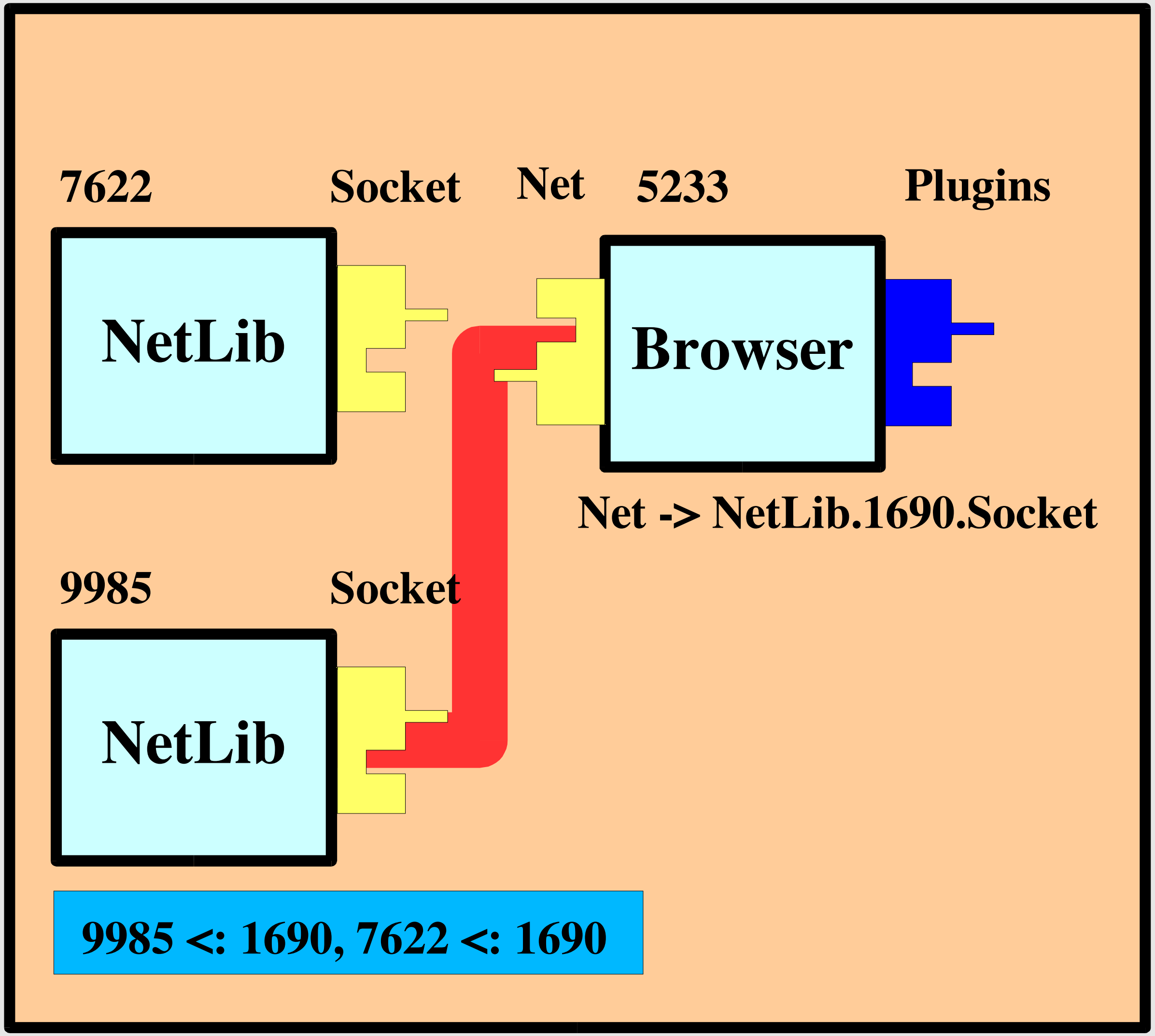
install *(shippedB)*

Updating a Component



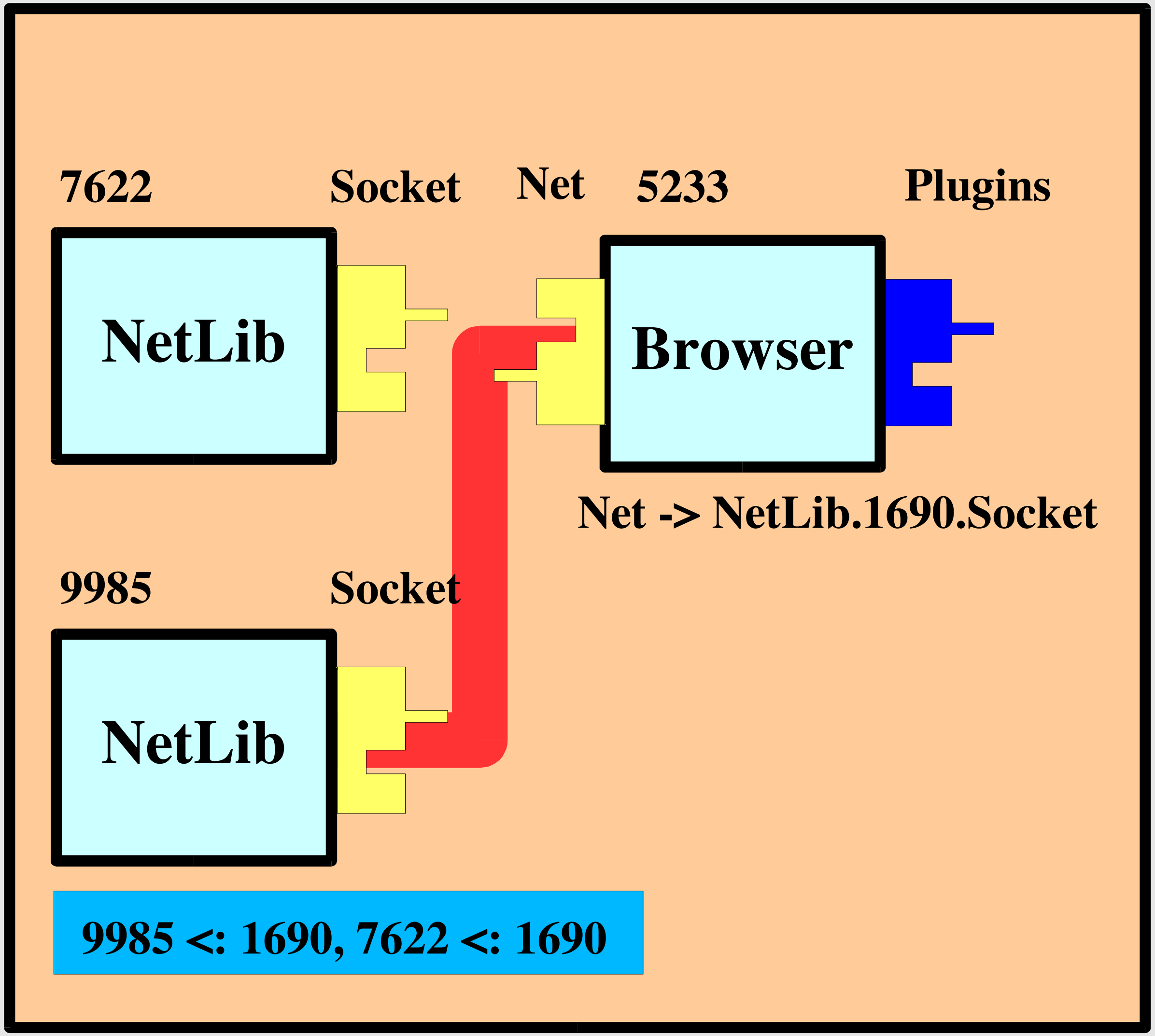
update *(NetLib, 7622, 9985)*

Updating a Component



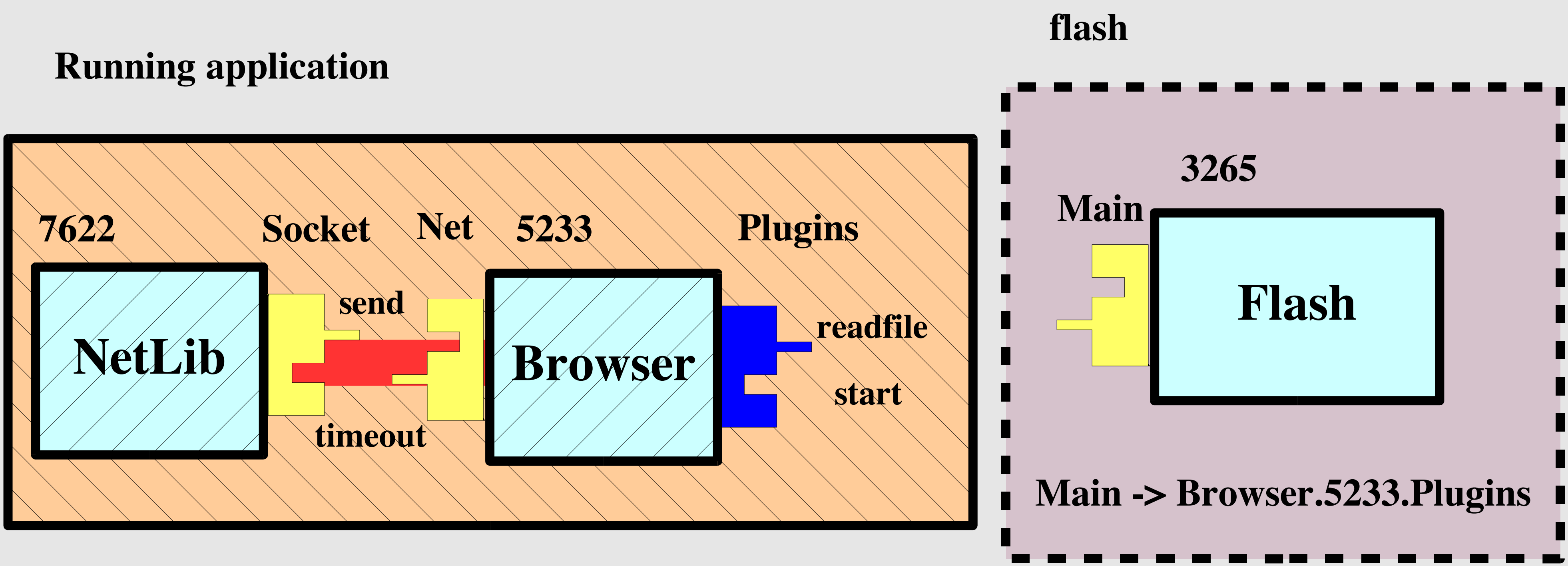
update *(NetLib, 7622, 9985)*

Updating a Component



an update is not necessarily an upgrade

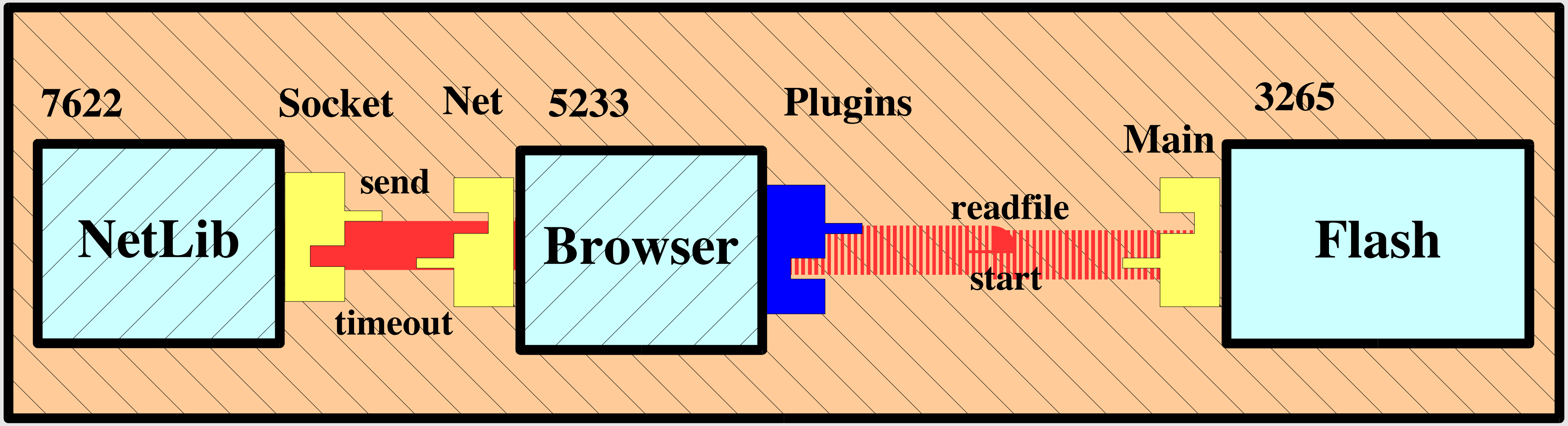
Hot Deployment



h = plugin flash with Plugins >> Main;

Hot Deployment

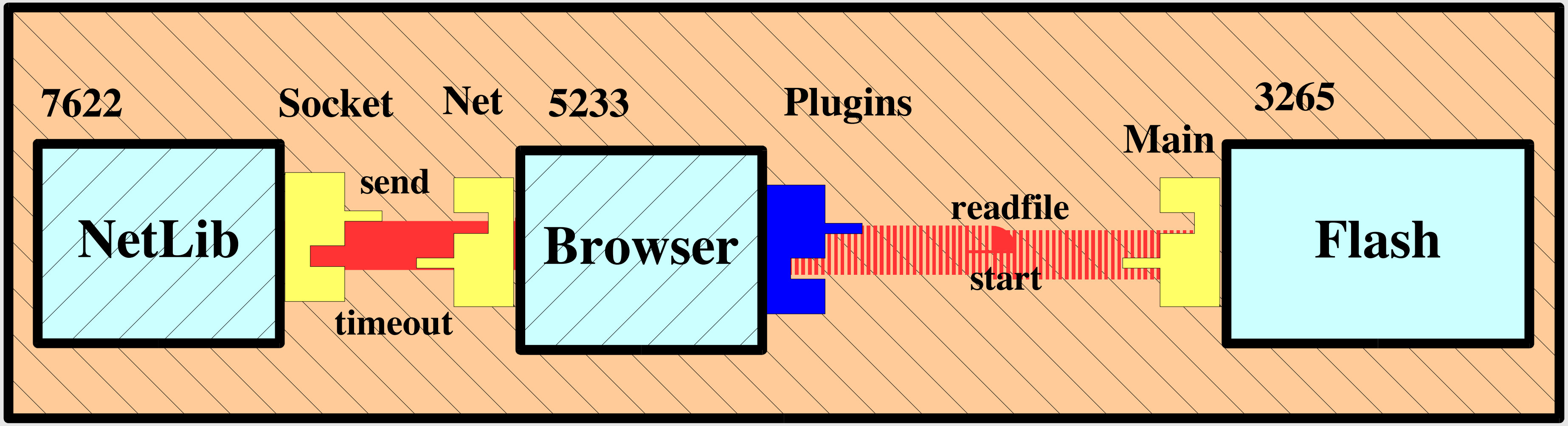
Running application



h = plugin flash with Plugins >> Main;

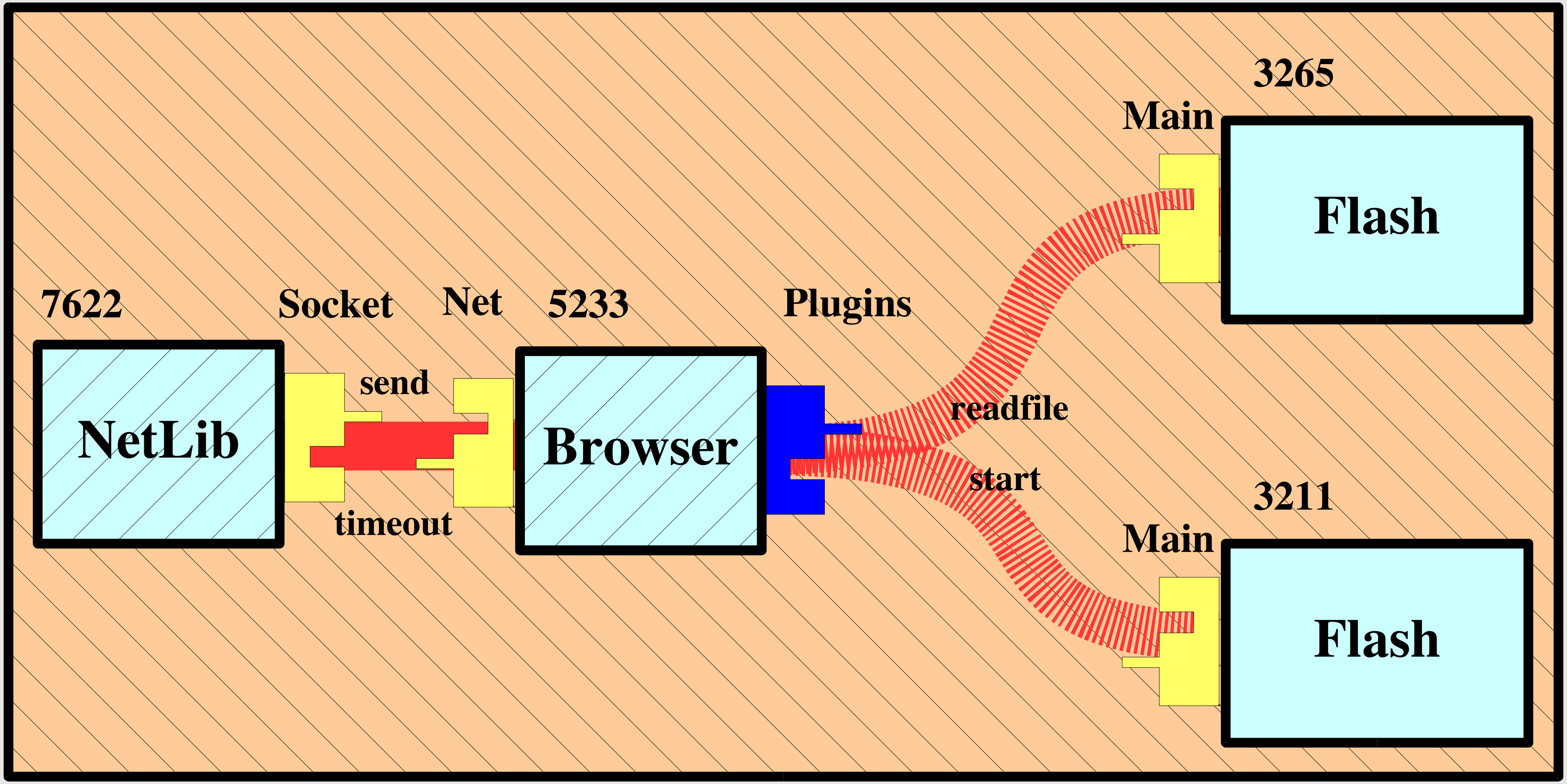
Hot Deployment

Running application



h = plugin flash with Plugins >> Main;
h..start();

Multiple Plugins: Hot Update



h1 = plugin flash1 with Plugins >> Main;

...

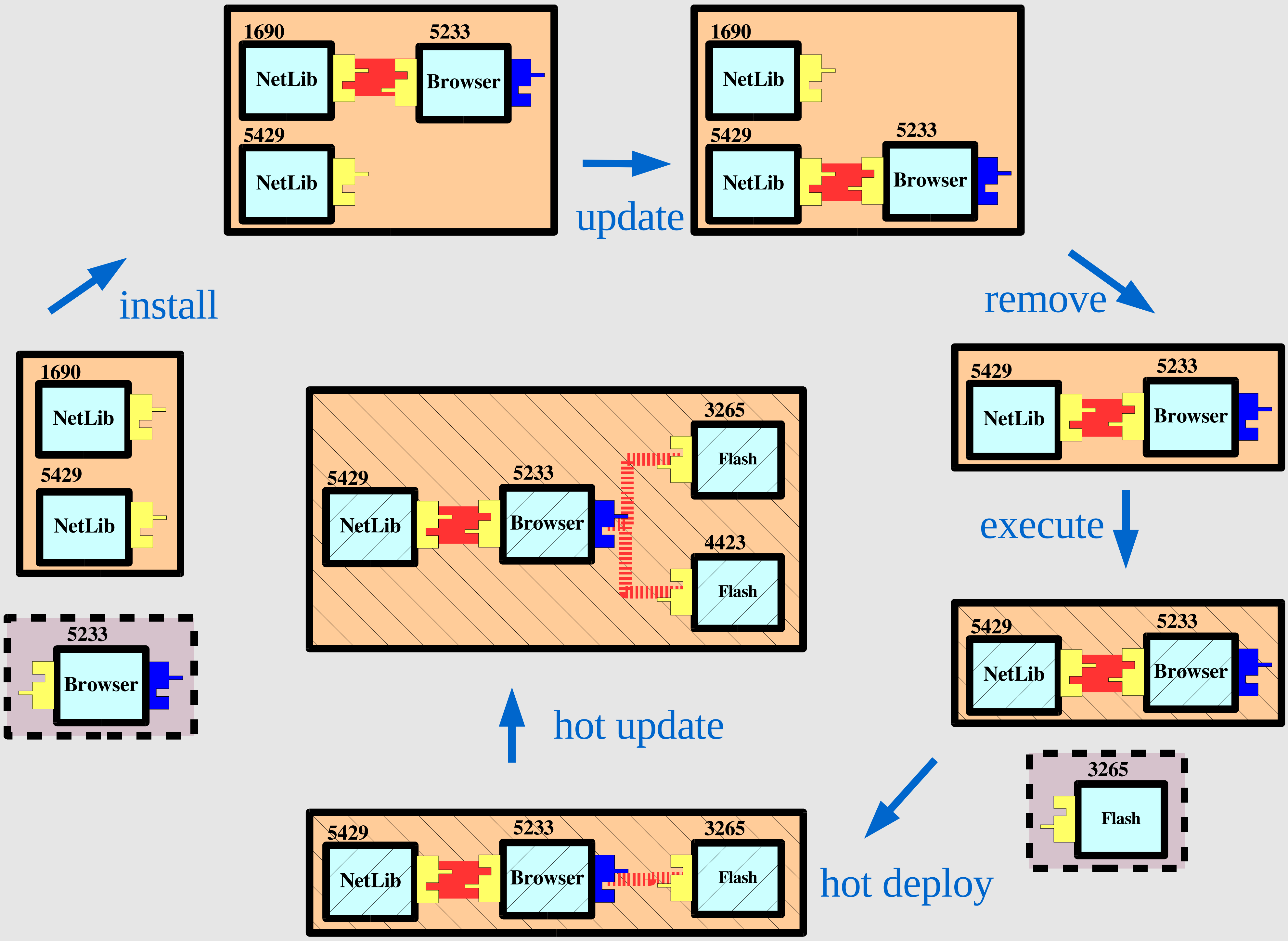
h2 = plugin flash2 with Plugins >> Main;

Act 3:

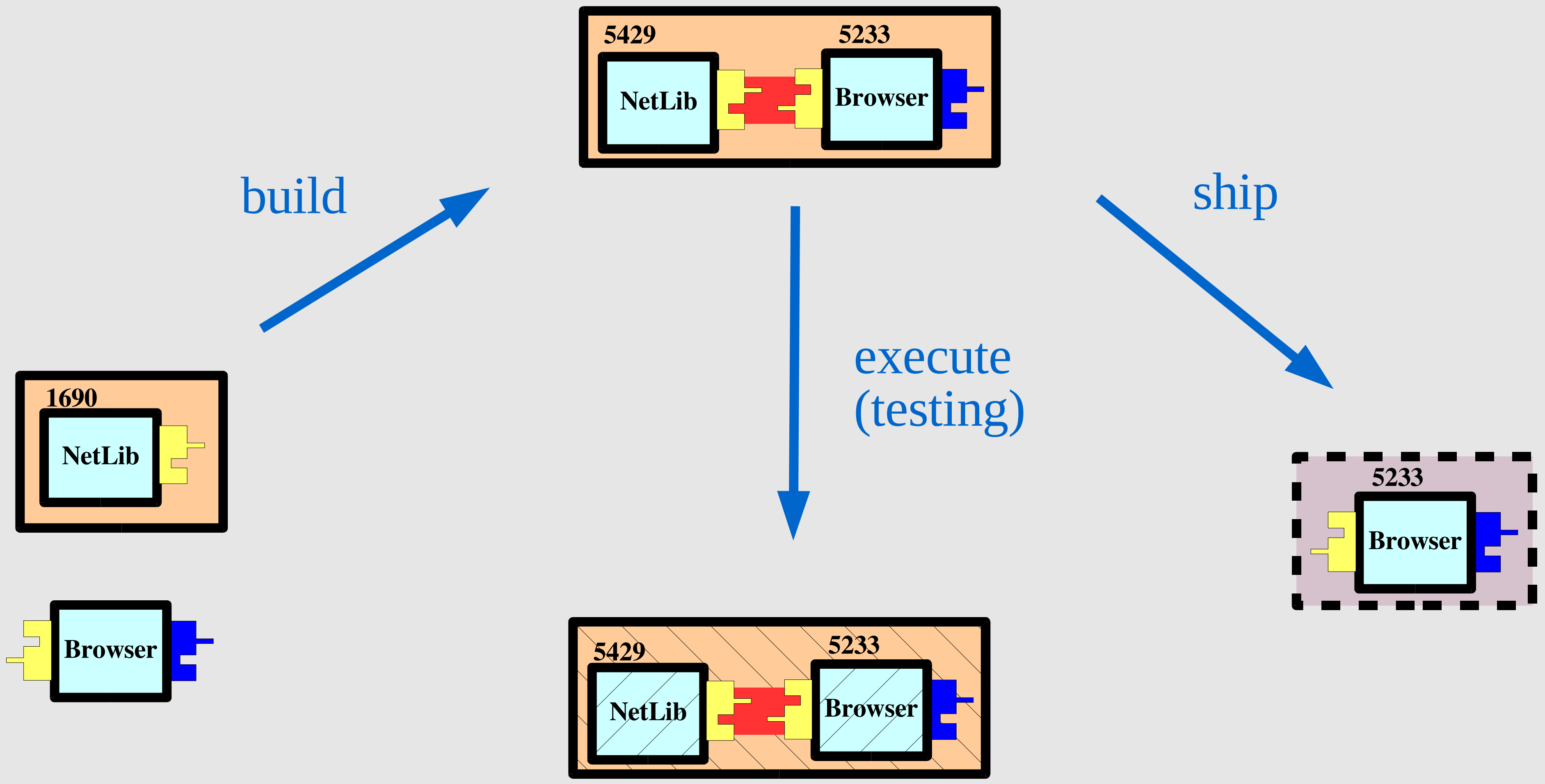
Invariants, Invariants!

Theorems: Buildbox Well-formedness

- Theorem: no deployment operations can turn a well-formed buildbox into a non-well-formed one.
- Theorem: no reductions at run time can turn a well-formed buildbox into a non-well-formed one.



1690
NetLib
5233
Browser
5429
NetLib
update
1690
NetLib
5429
NetLib
5233
Browser
remove
5429
NetLib
5233
Browser
execute
5429
NetLib
5233
Browser
3265
Flash
hot deploy
5429
NetLib
5233
Browser
3265
Flash
hot update
5429
NetLib
5233
Browser
3265
Flash
4423
Flash
install
1690
NetLib
5429
NetLib
5233
Browser

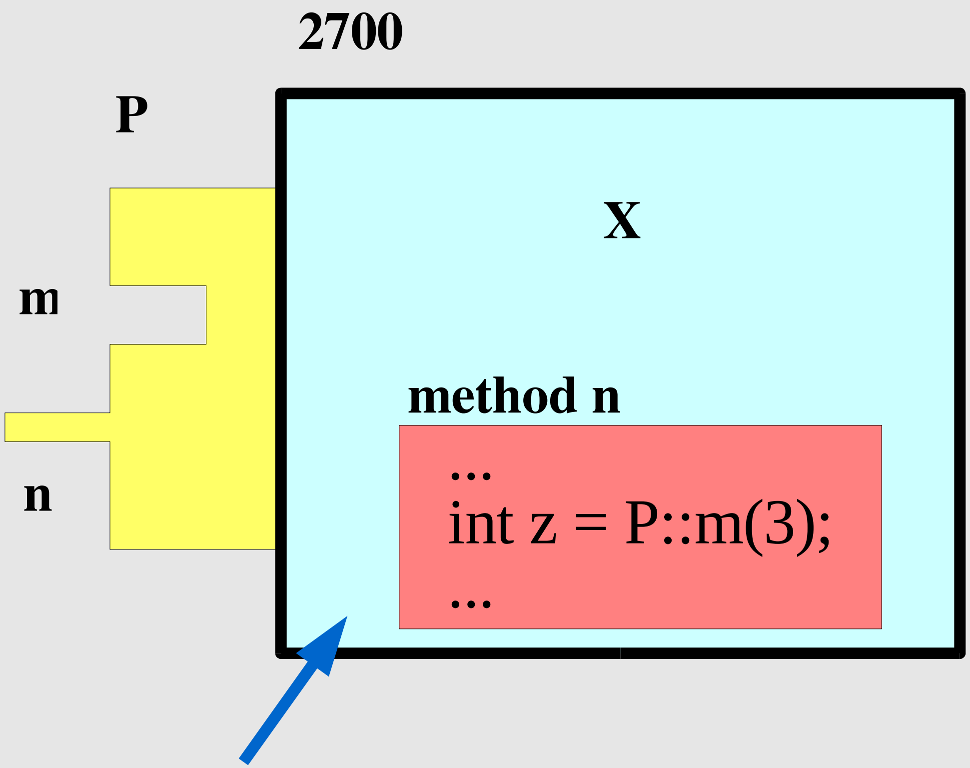


5429
5233
NetLib
Browser
build
ship
execute
(testing)
1690
NetLib
Browser
5233
Browser
5429
5233
NetLib
Browser

Specifying Version Compatibility

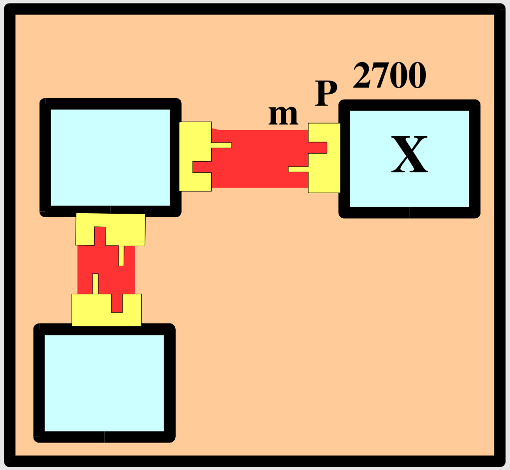
How do a ***deployment-site run*** and ***a pre-shipping test-run*** correspond?

Suppose we have a component X...

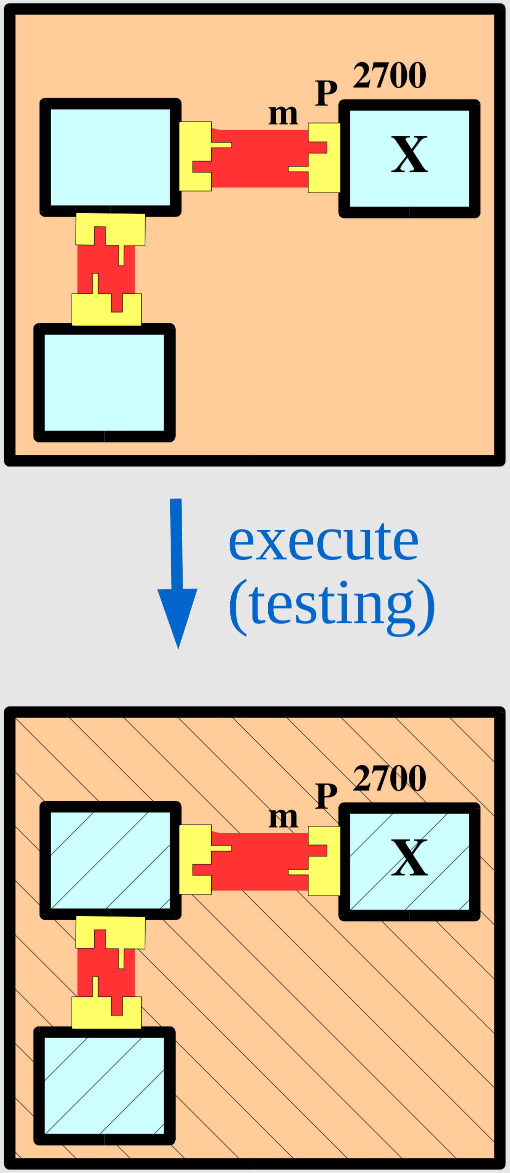


locating method m imported/exported from P

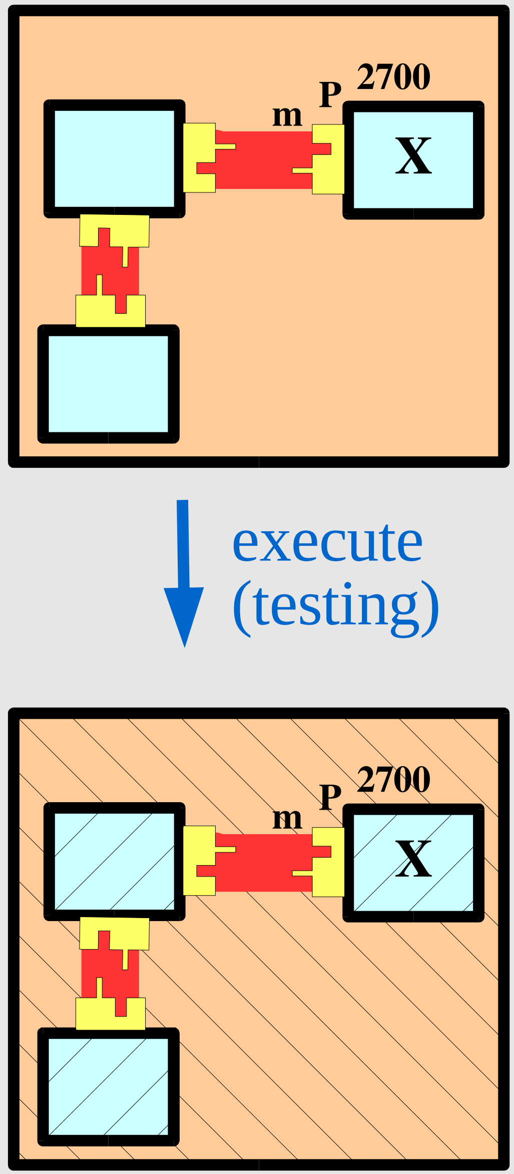
On The Development Site



On The Development Site

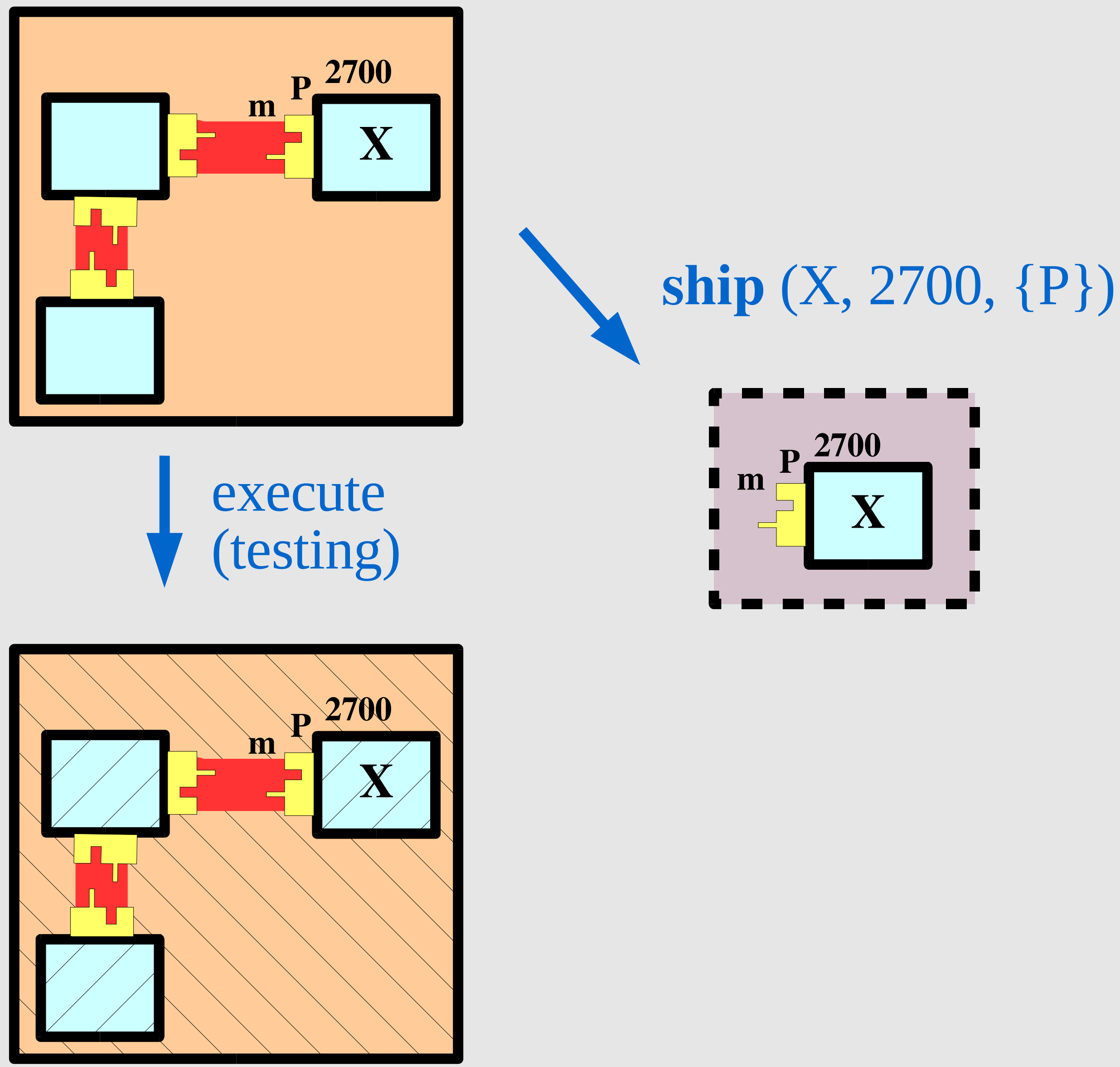


On The Development Site



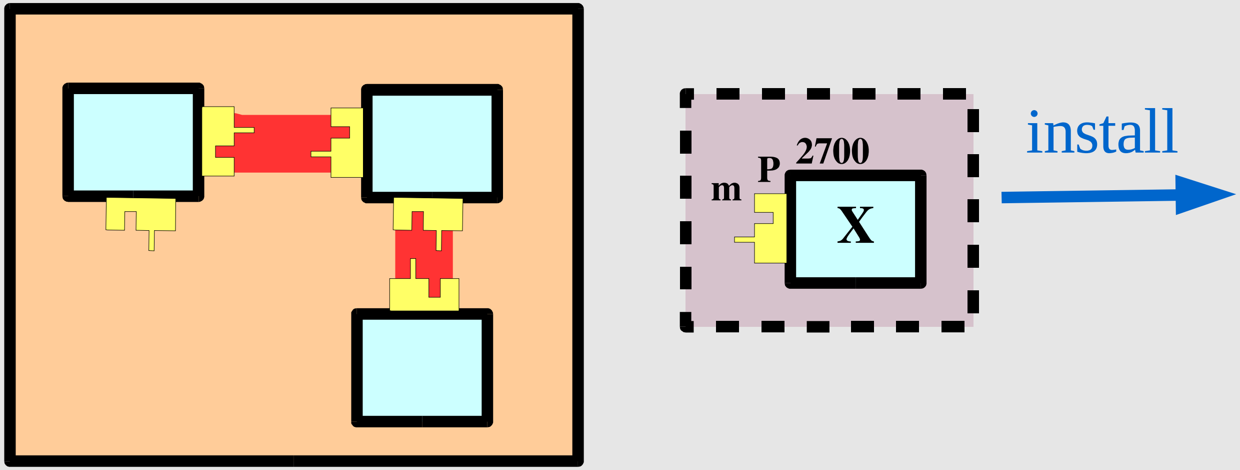
at run time
P::m is bound to
assemblage Y version v

On The Development Site

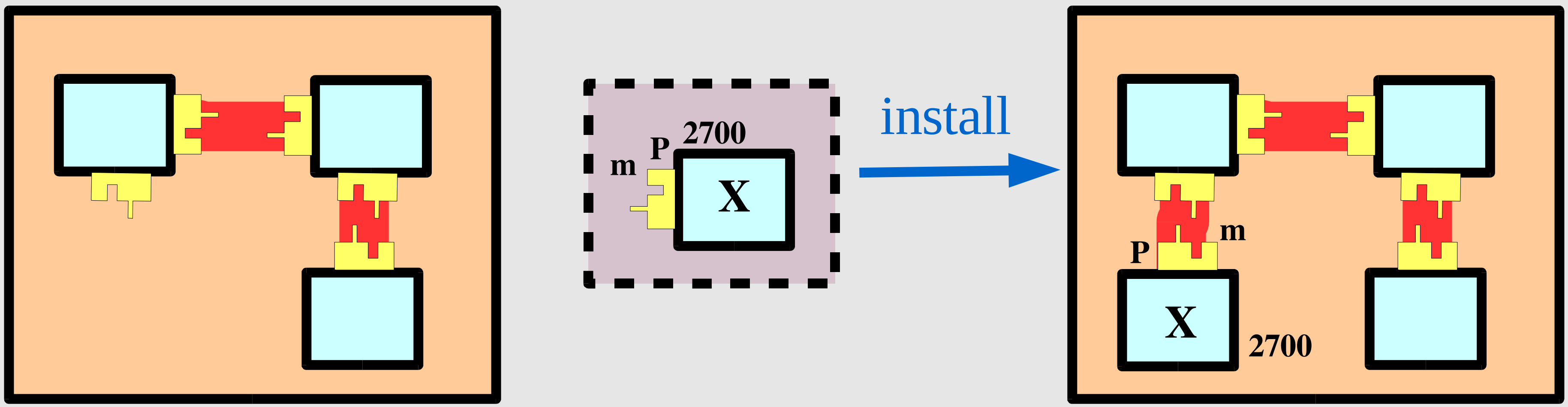


at run time
P::m is bound to
assemblage Y version v

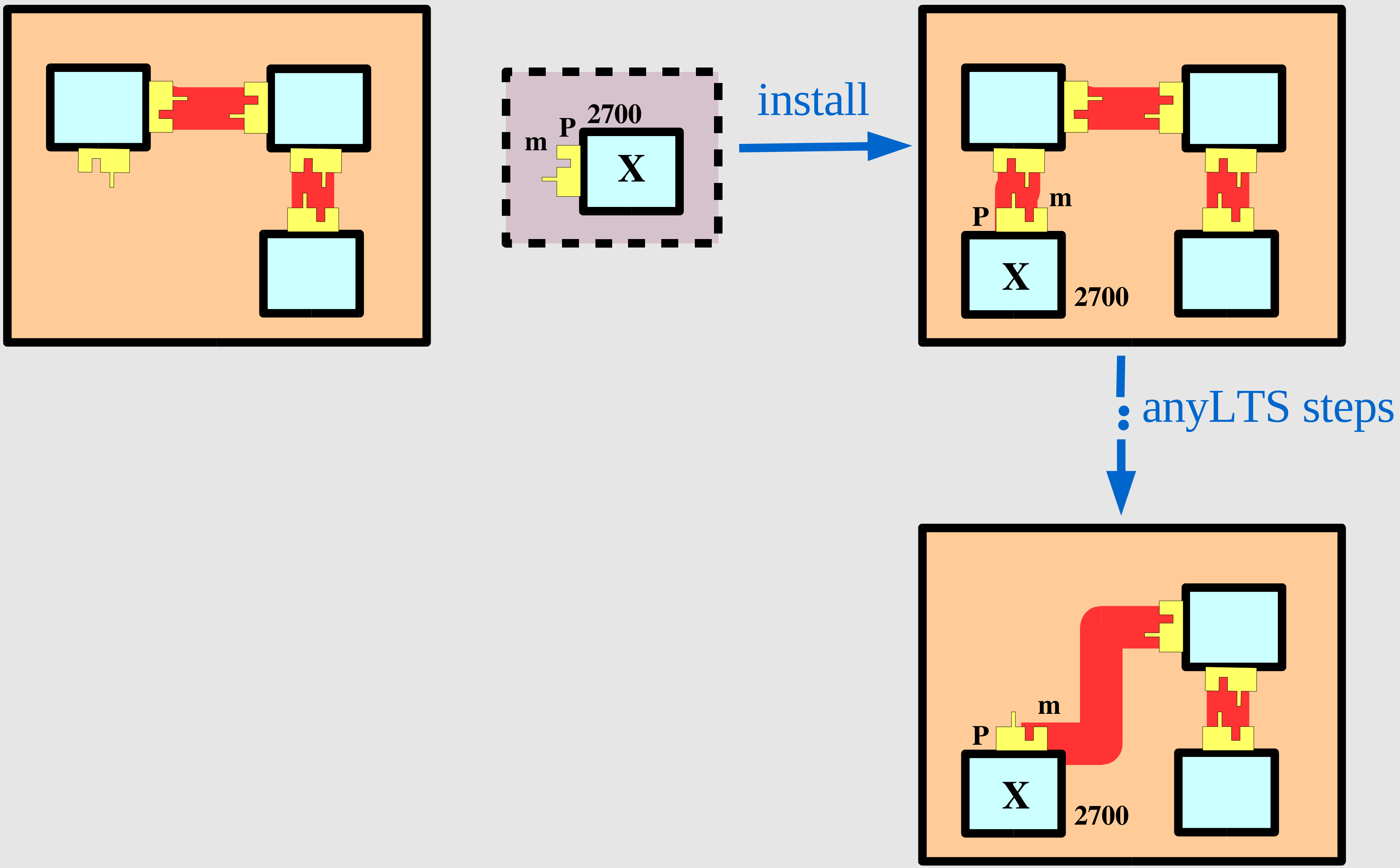
On Any Deployment Site



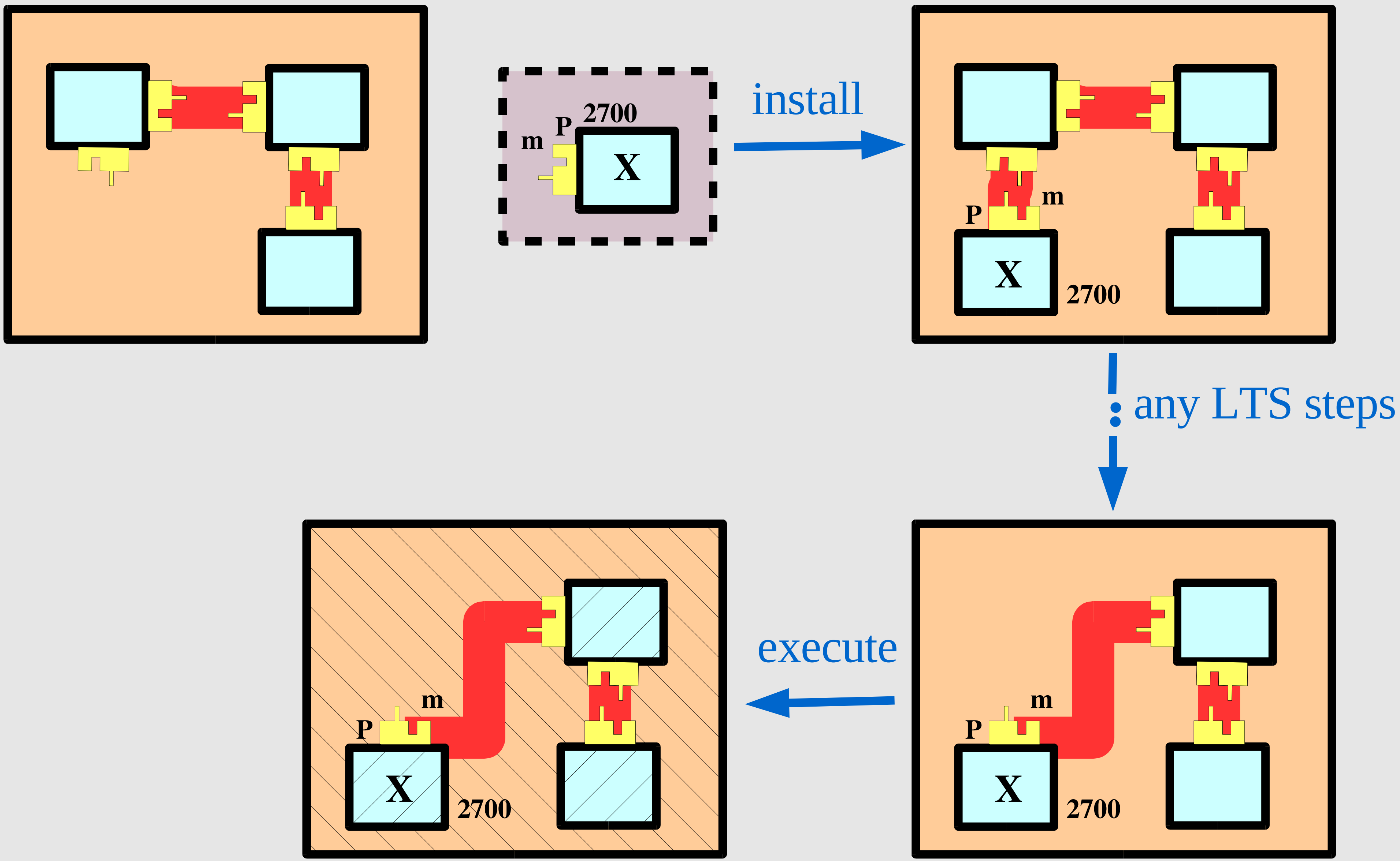
On Any Deployment Site



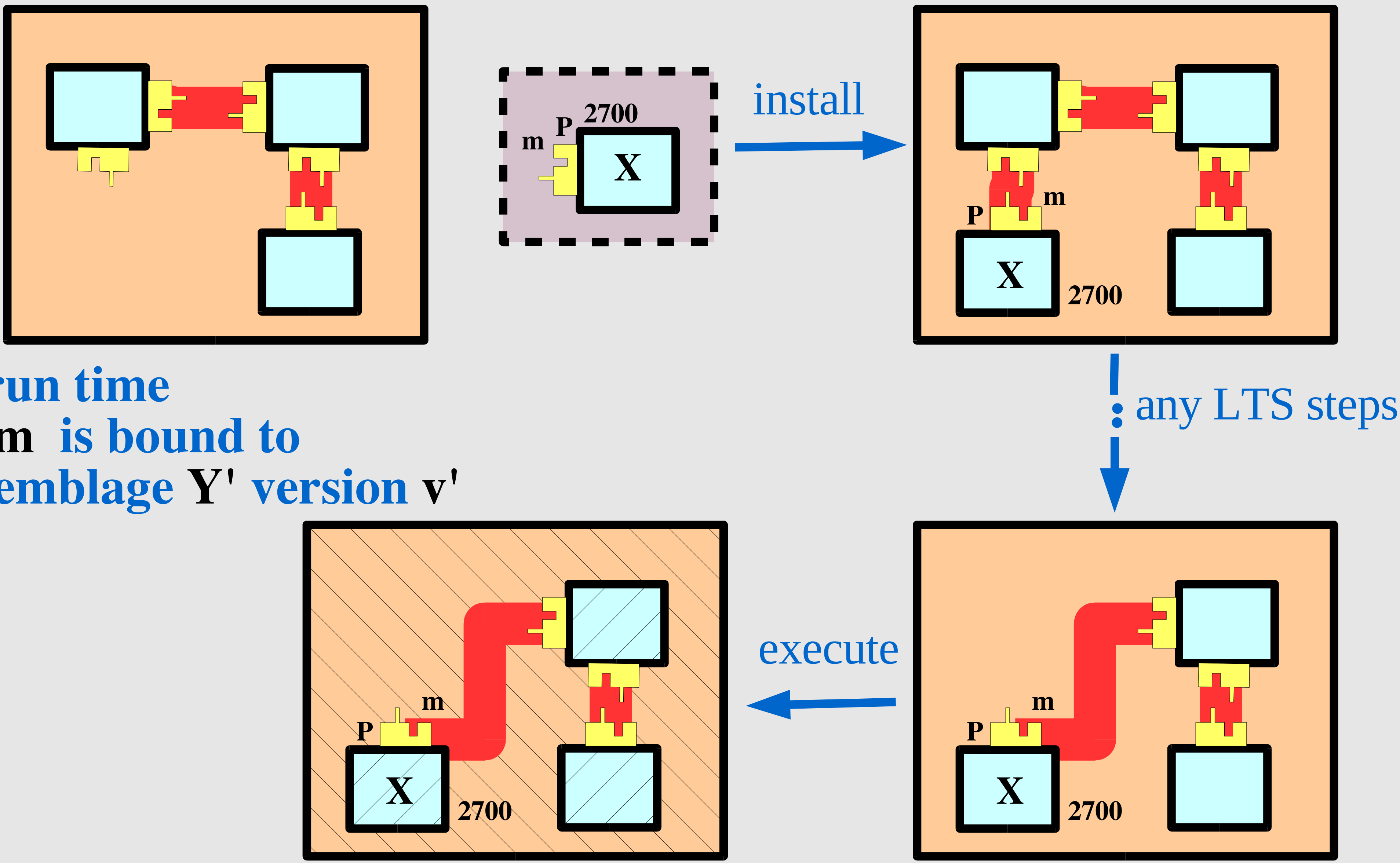
On Any Deployment Site



On Any Deployment Site



On Any Deployment Site



at run time
P::m is bound to
assemblage Y' version v'

Theorem on Version Compatibility

- $Y = Y'$
- $v = v'$ or v' is a subversion of v

Future Work

- Keep the platform-independent spirit, with more expressiveness gains
  - security in deployment
  - distributed deployment (e.g. sensor network applications)
- A closer look at Java deployment
  - an effort to map back to the real world

Related Work

- Many real-world systems
- Formal treatment is rare
  - [Buckley, CD'05]: formalized name-binding of CLI Assemblies
    - platform-specific
    - no modeling of deployment lifecycle
    - no invariant properties proved

Related Work: Real-world Systems

CLI Assemblies

JSR 277

InstallShield

RPM

OSGi

Dpkg

Application Buildbox

EJB Manifests

Portage

Bazaar

CORBA D&C

RubyGems

CTAN

CPAN

Related Work

- Many real-world systems
- Formal treatment is rare
  - [Buckley, CD'05]: formalized name-binding of CLI Assemblies
    - platform-specific
    - no modeling of deployment lifecycle
    - no invariant properties proved

A Retrospective

- For deployment systems designers:
  - platform-independent communication
  - foster next-generation deployment systems
- For deployment system users:
  - tools with well-defined user interfaces
  - tools with provably correct properties
- For module system researchers:
  - a foundational study of when and where of linking