Storing and Accessing Live Mashup Content in the Cloud

Krzysztof Ostrowski, Ken Birman

Cornell University

{krzys|ken}@cs.cornell.edu

Agenda

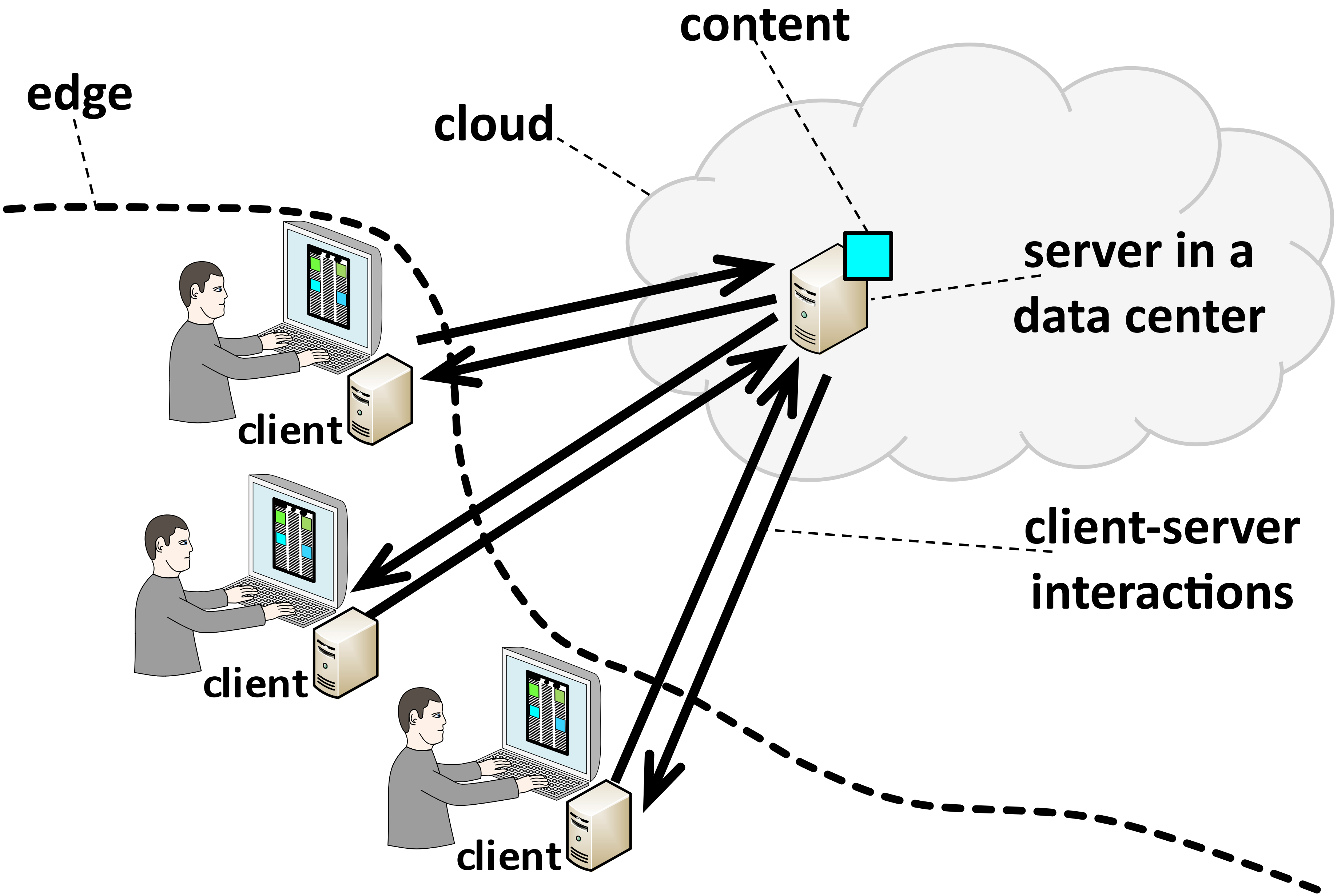
Agenda

1. A new, versatile storage abstraction:
   Checkpointed Channel (CC)

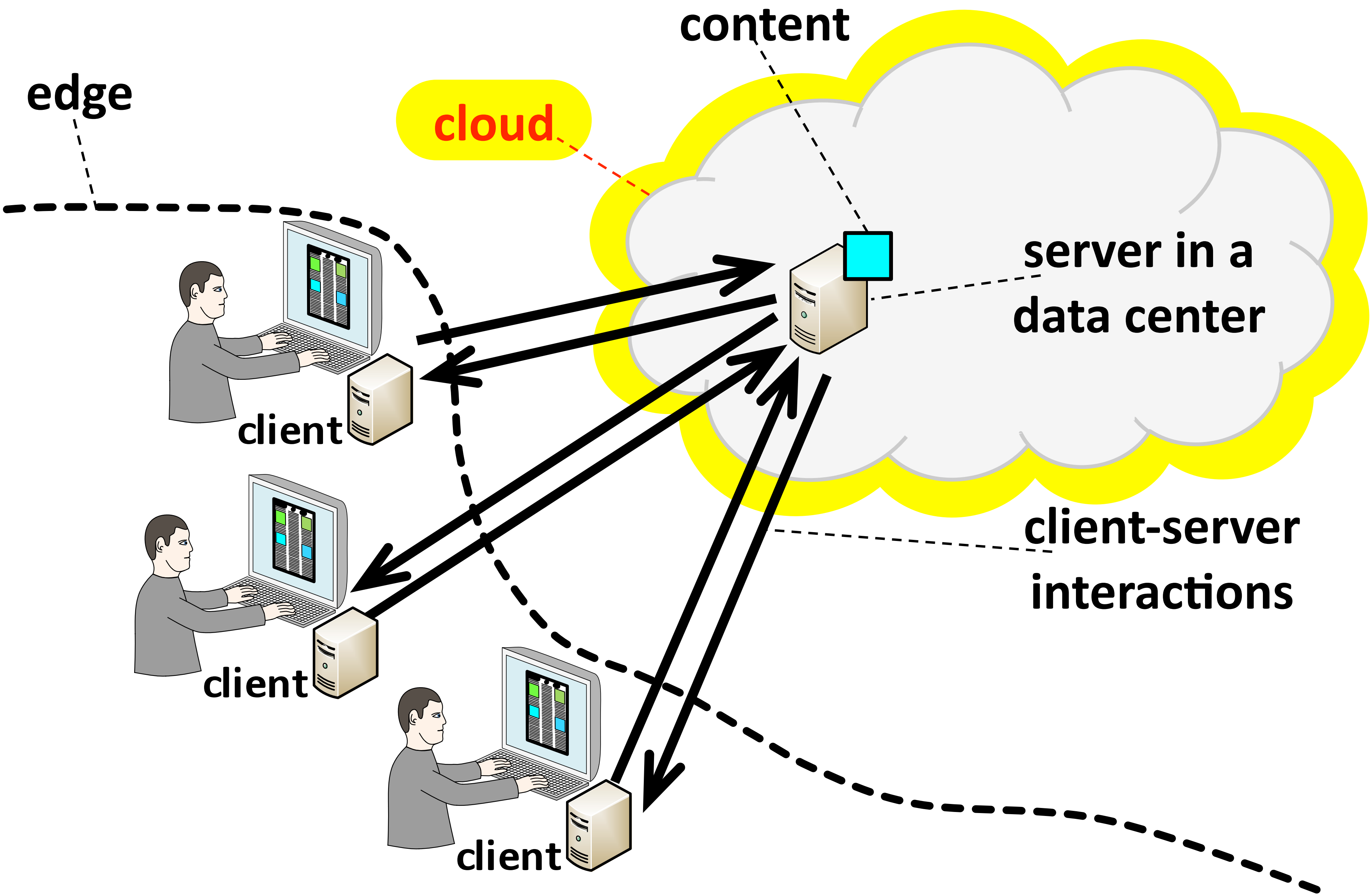
2. A new web application architecture:
   A web of (hyperlinked) CCs

Introduction

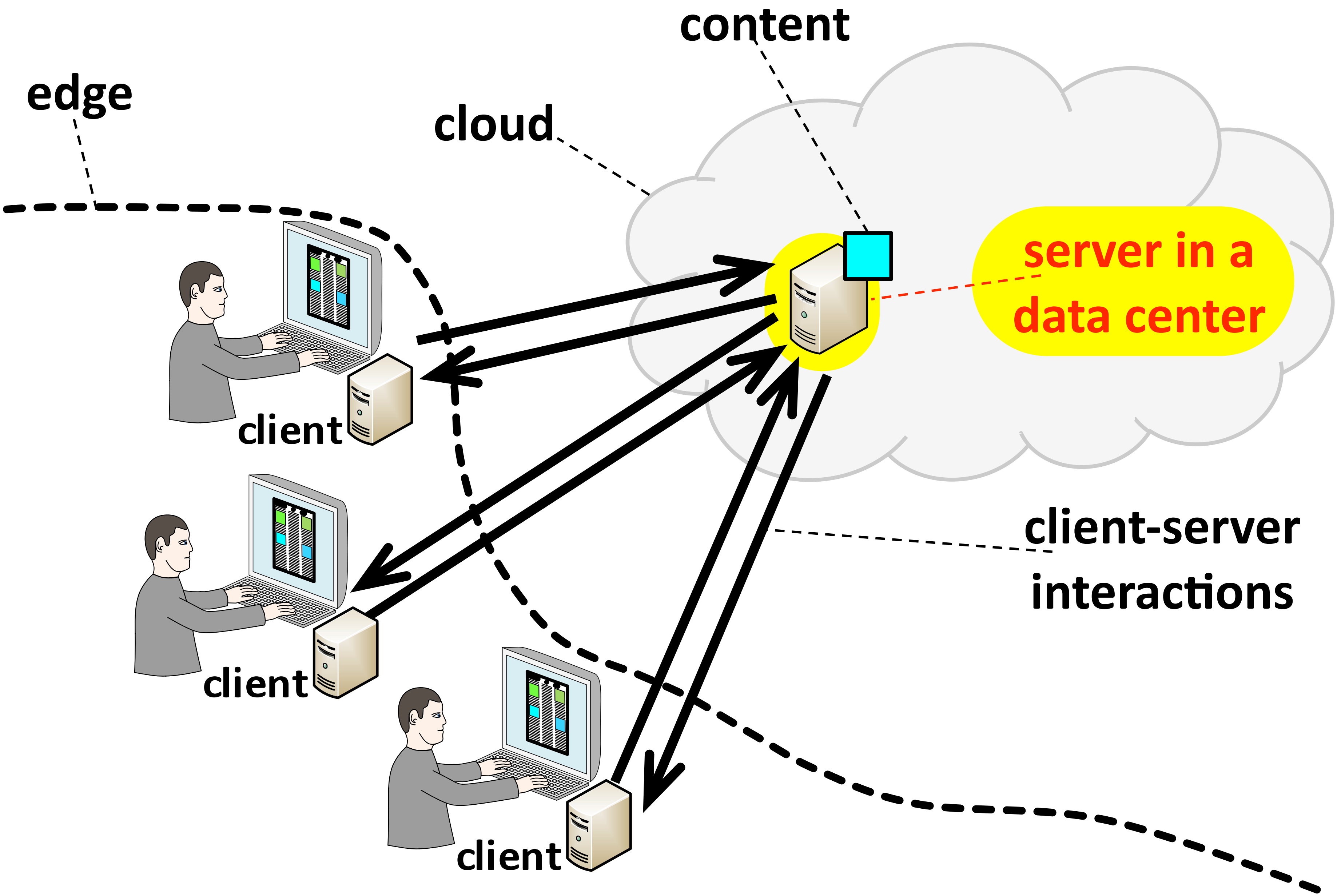
Storing Content in the Cloud



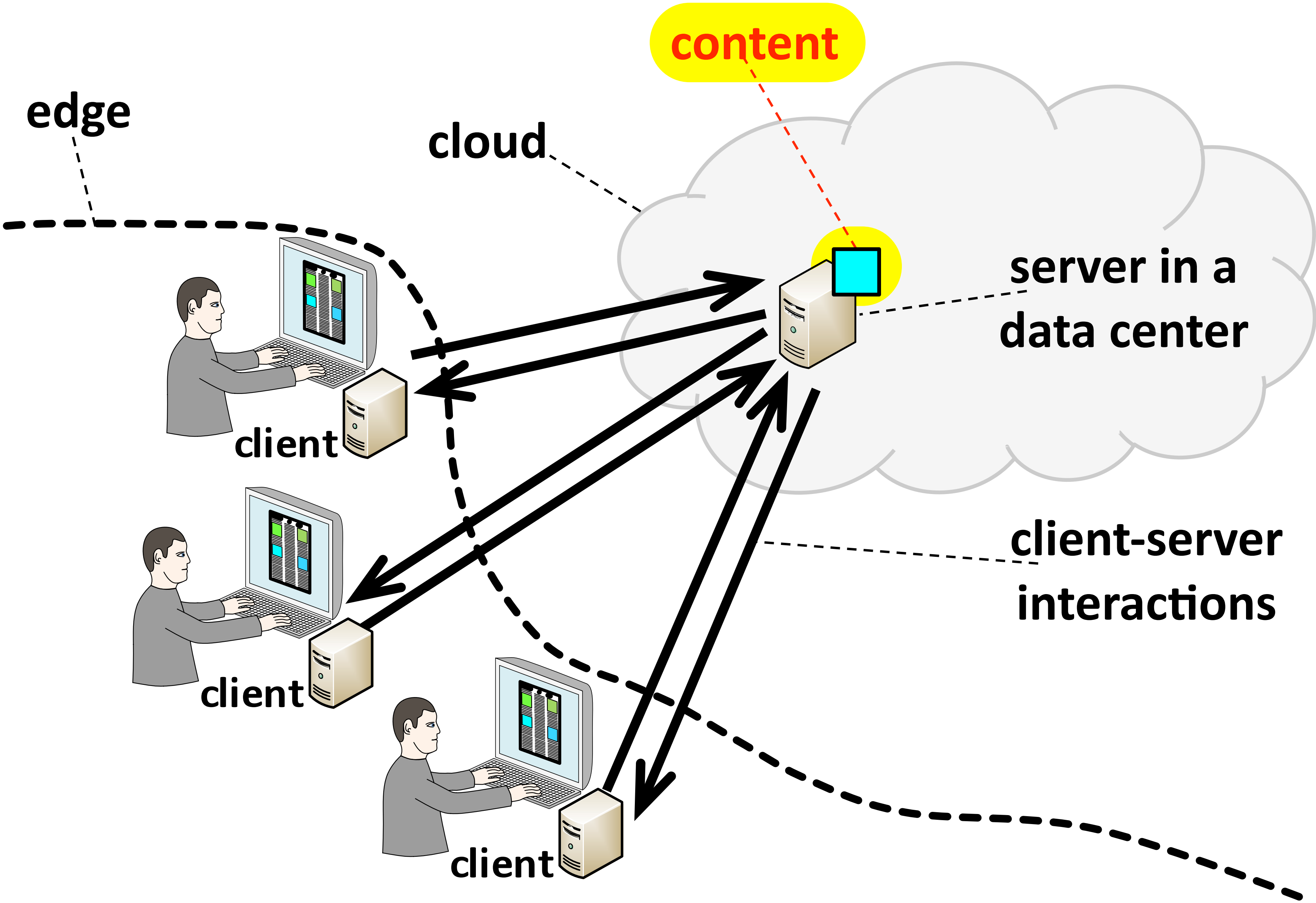
Storing Content in the Cloud



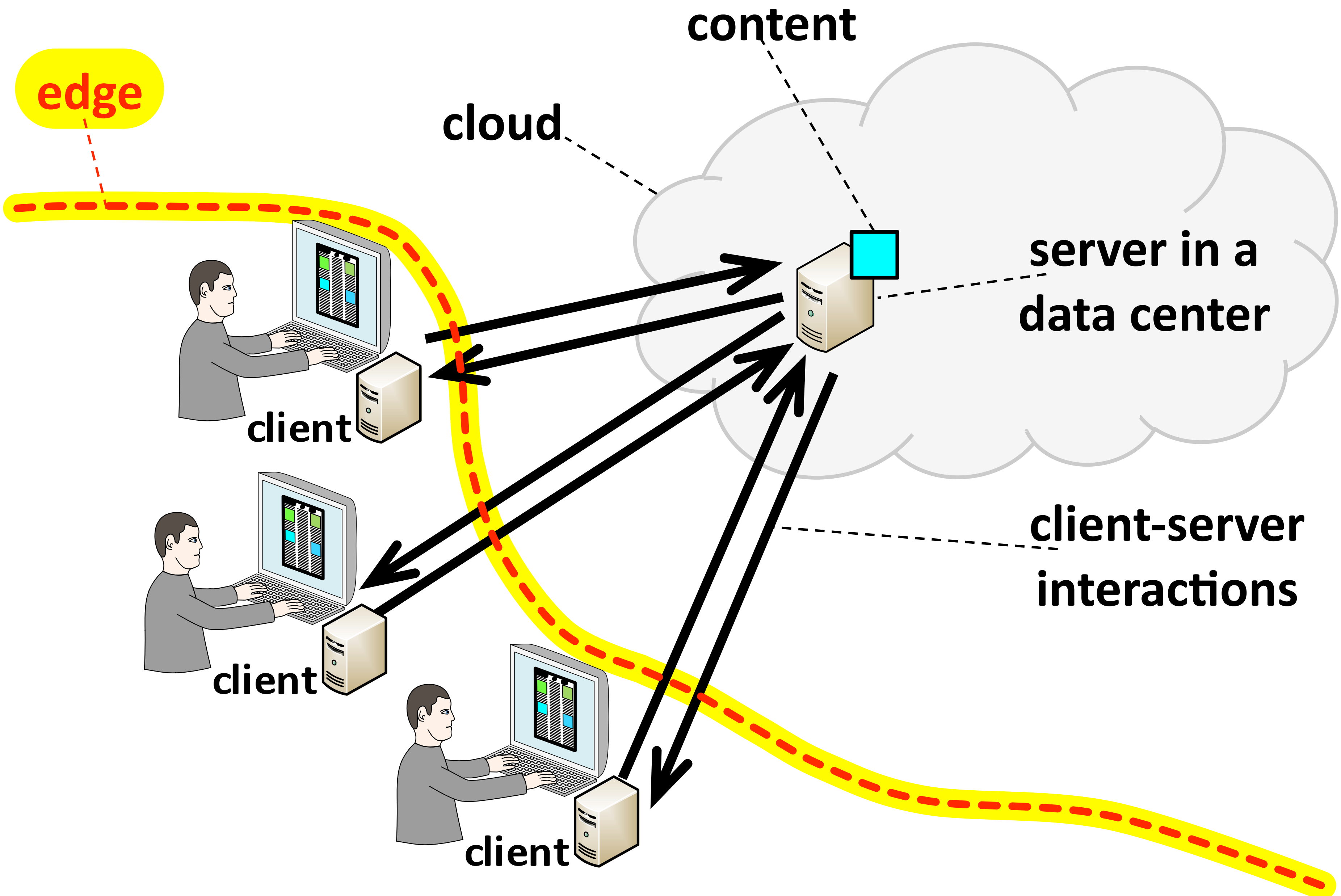
Storing Content in the Cloud



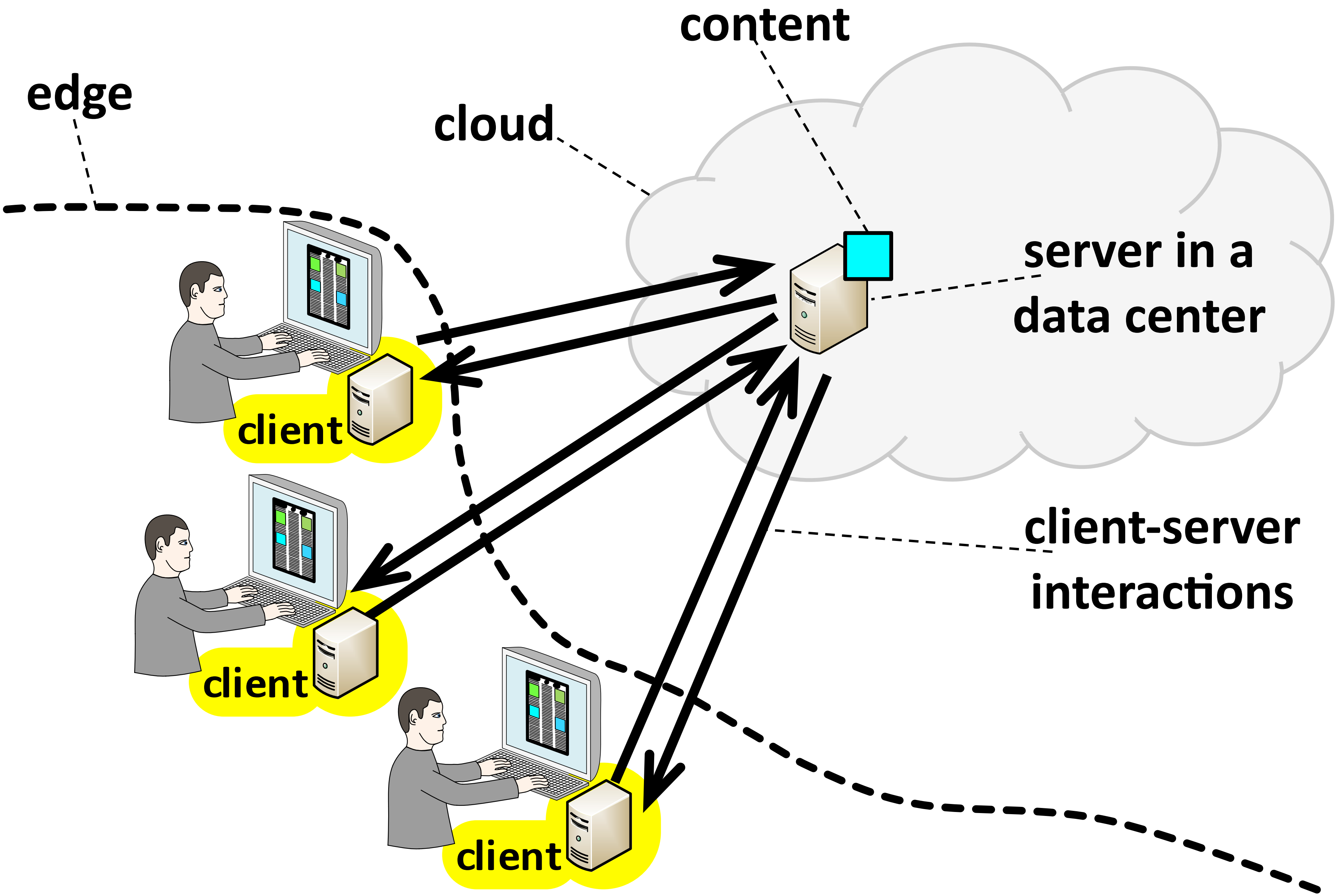
Storing Content in the Cloud



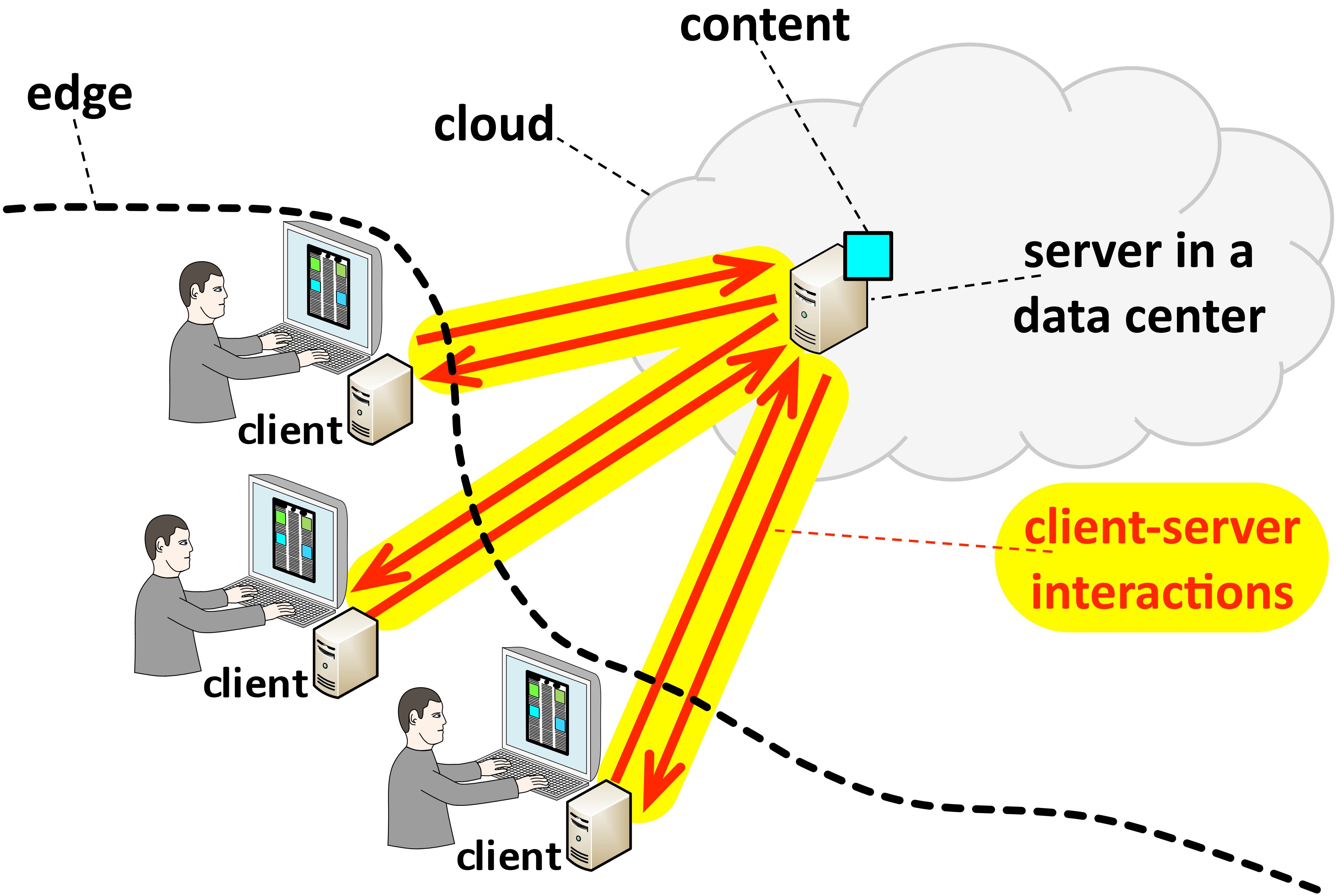
Storing Content in the Cloud



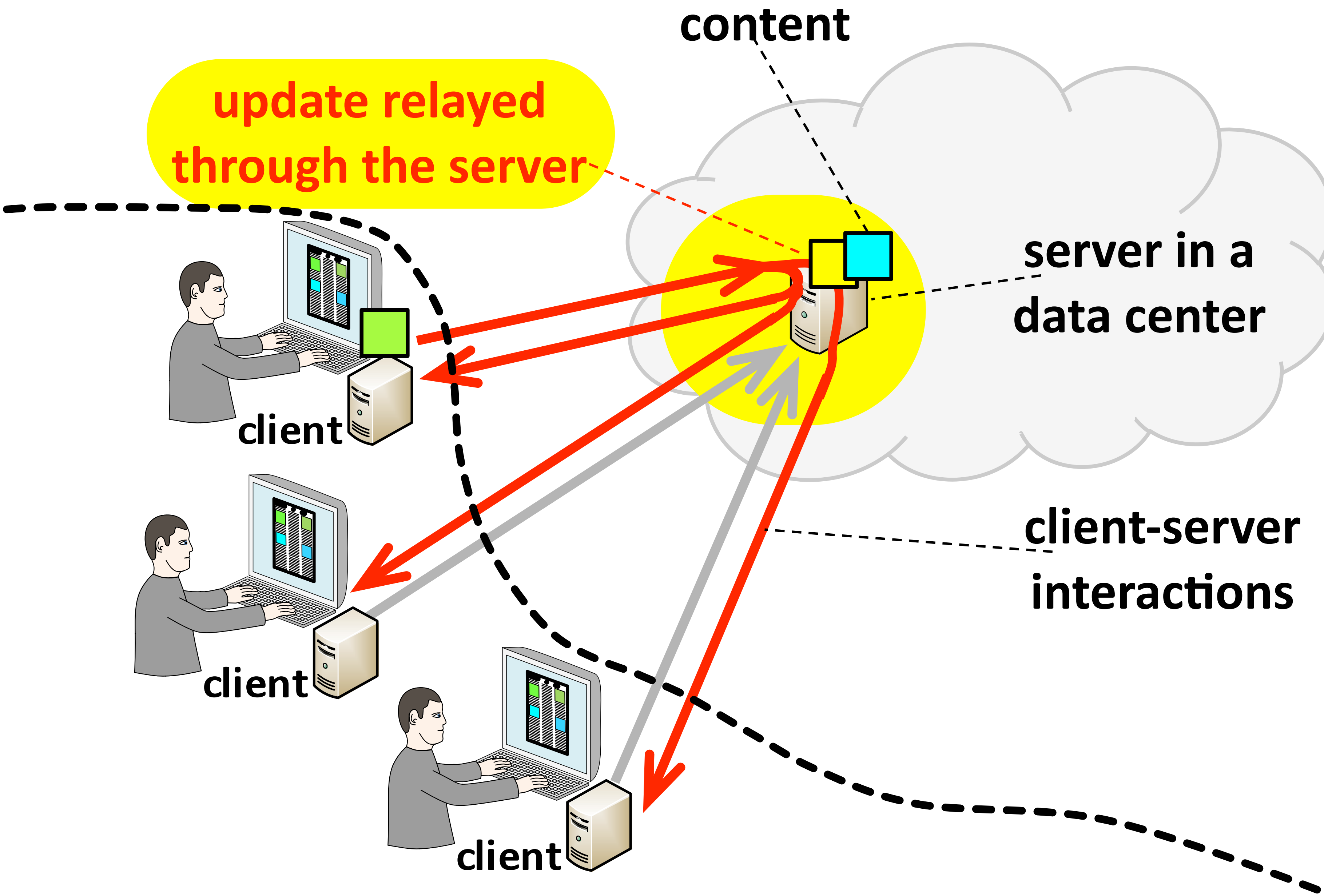
Storing Content in the Cloud



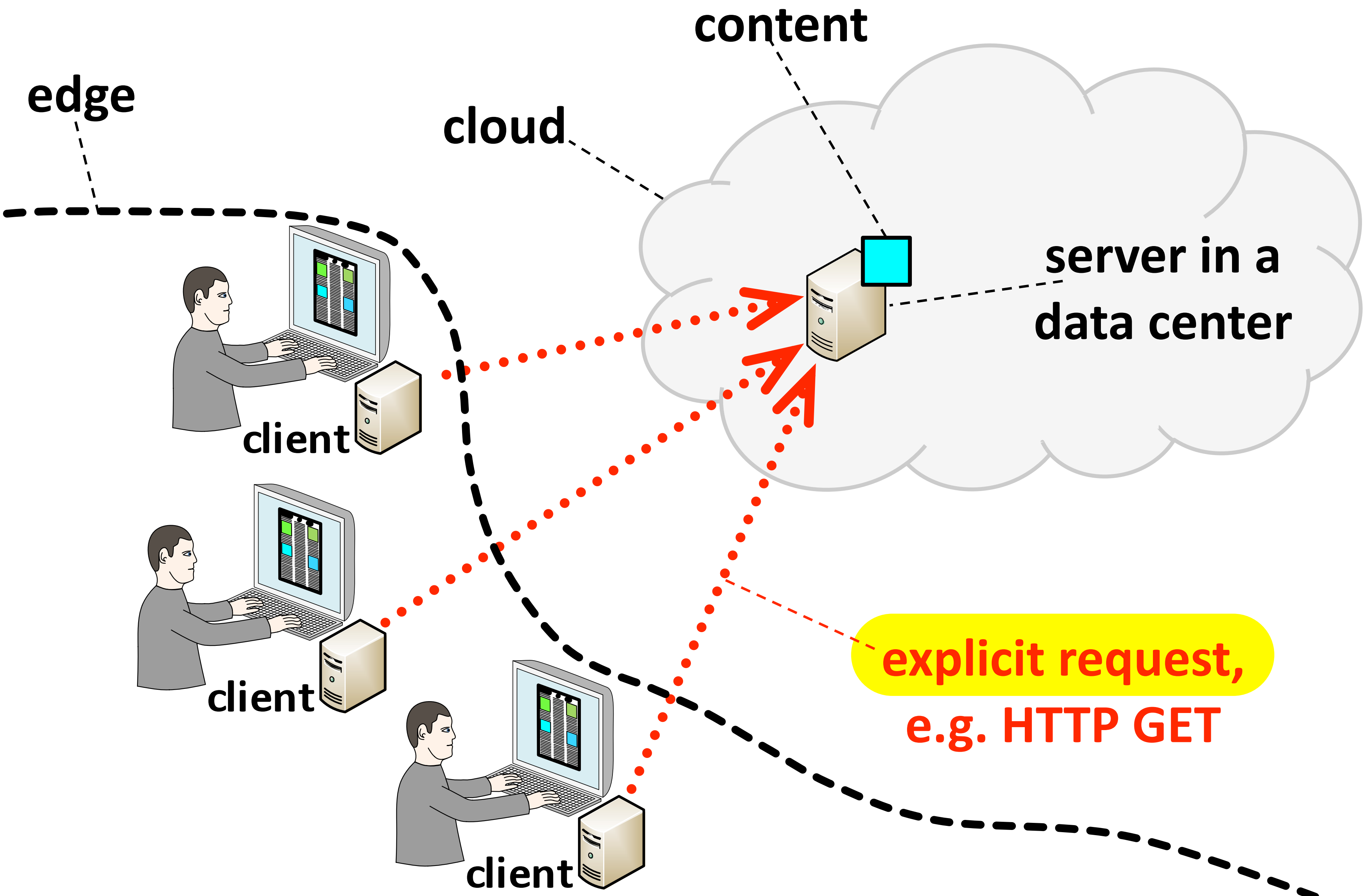
Storing Content in the Cloud



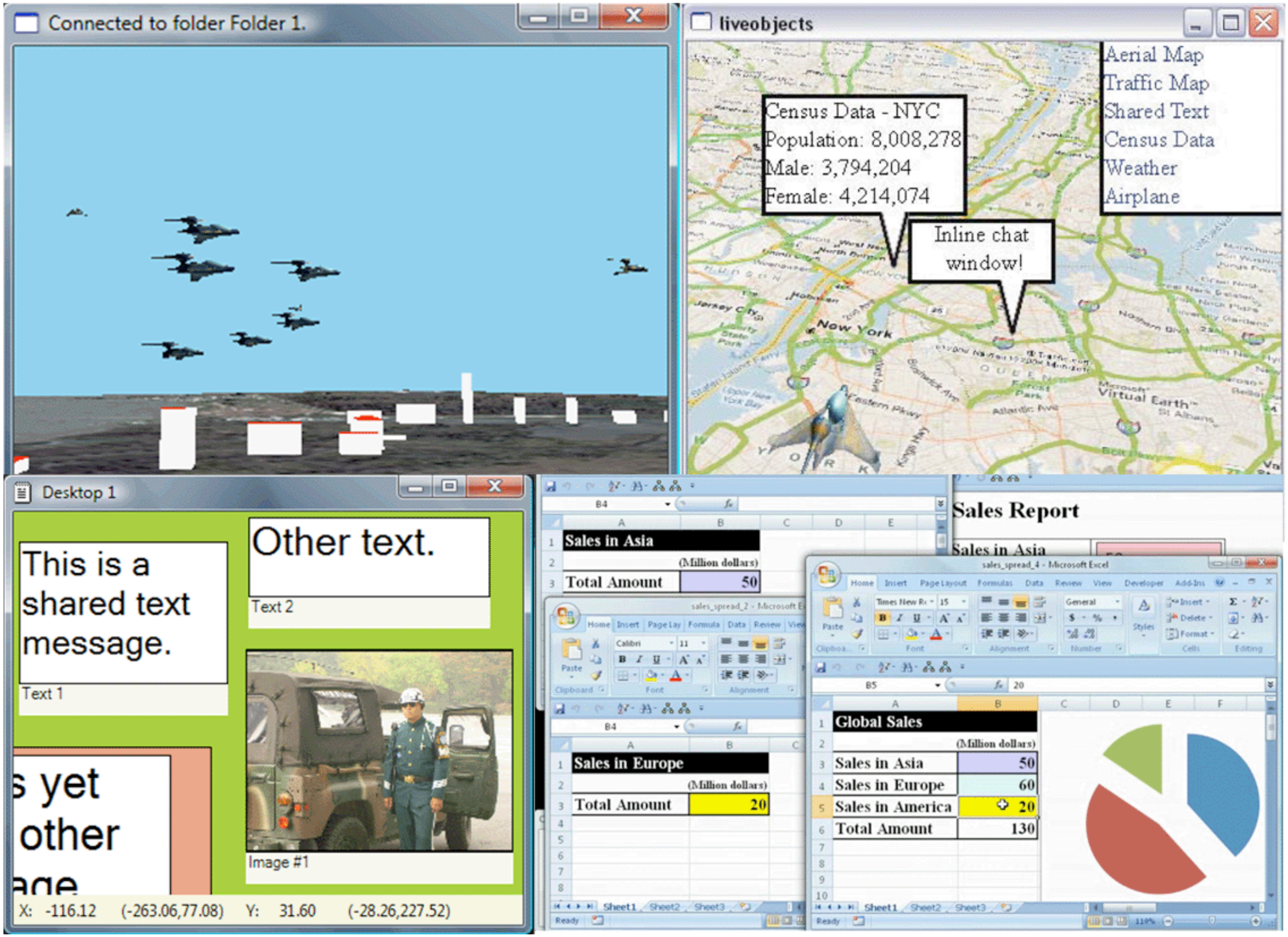
Storing Content in the Cloud



Storing Content in the Cloud



http://liveobjects.cs.cornell.edu



Connected to folder Folder 1.

liveobjects

Census Data - NYC
Population: 8,008,278
Male: 3,794,204
Female: 4,214,074

Inline chat window!

Desktop 1

This is a shared text message.

Text 1

Other text.

Text 2

s yet other age

Image #1

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)

Sales Report

Sales in Asia

B4

	A	B
1	Sales in Asia	
2		(Million dollars)
3	Total Amount	50

sales_spread_2 - Microsoft E

B4

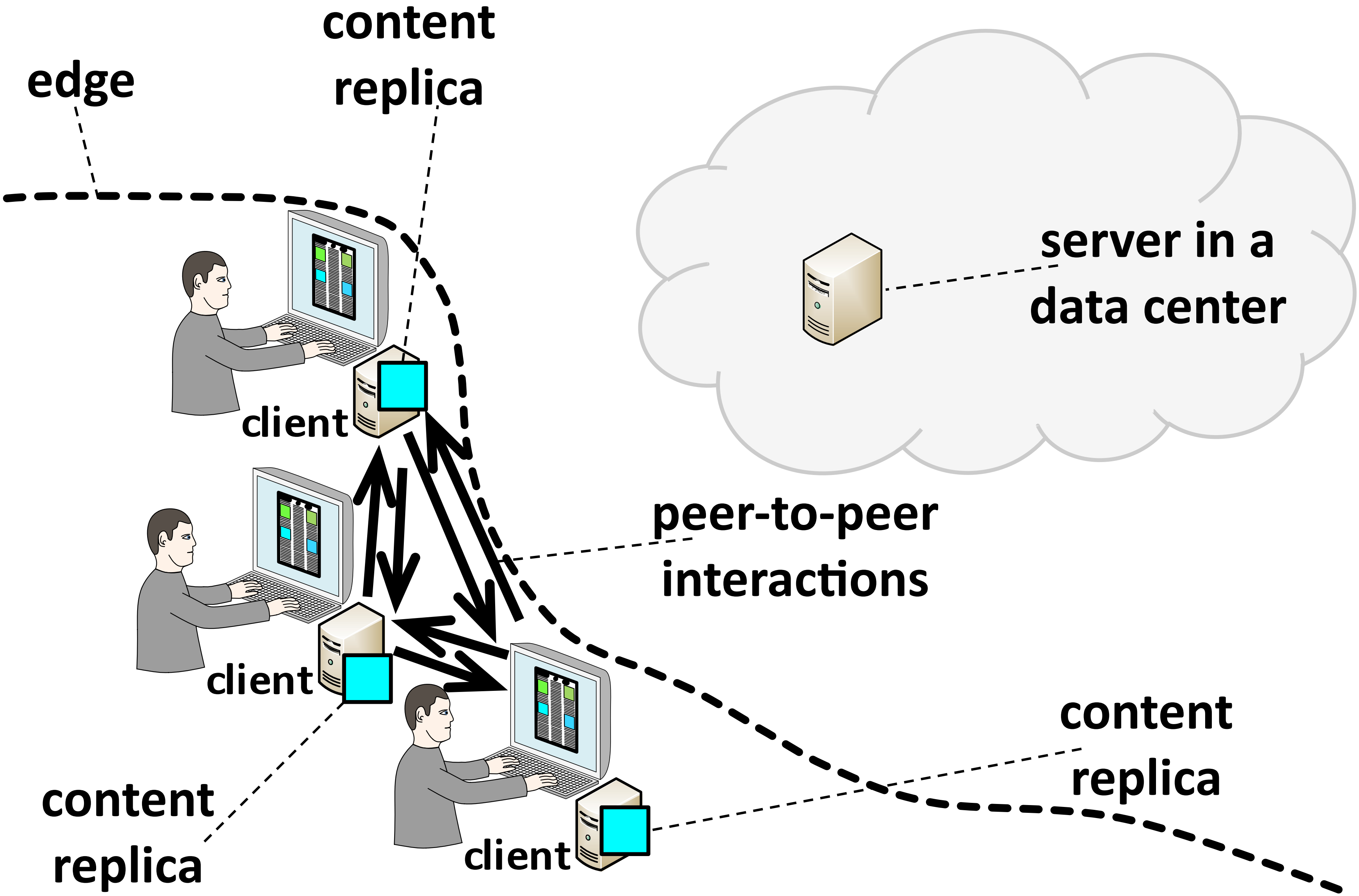
	A	B
1	Sales in Europe	
2		(Million dollars)
3	Total Amount	20

sales_spread_4 - Microsoft Excel

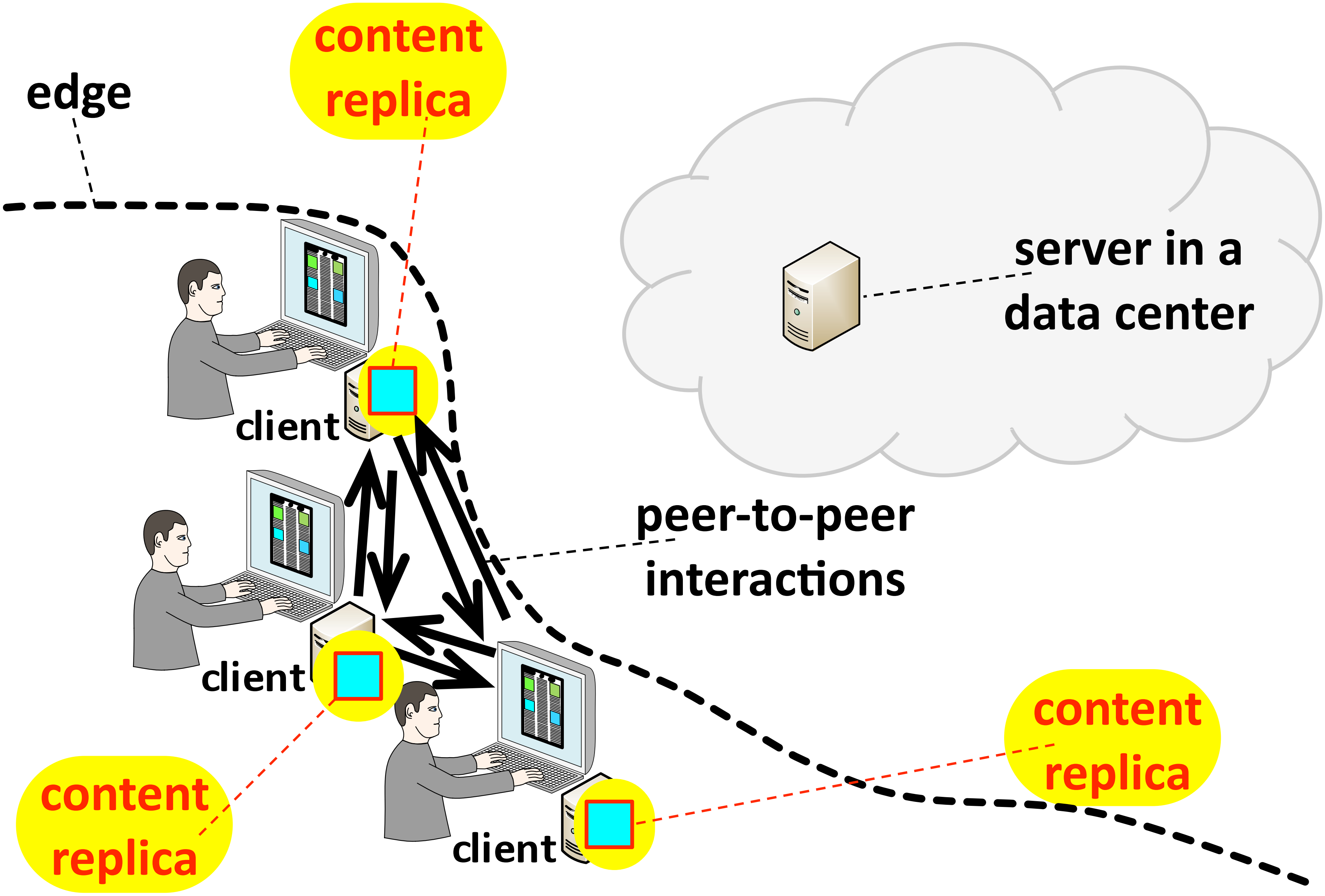
B5 fx 20

	A	B
1	Global Sales	
2		(Million dollars)
3	Sales in Asia	50
4	Sales in Europe	60
5	Sales in America	20
6	Total Amount	130

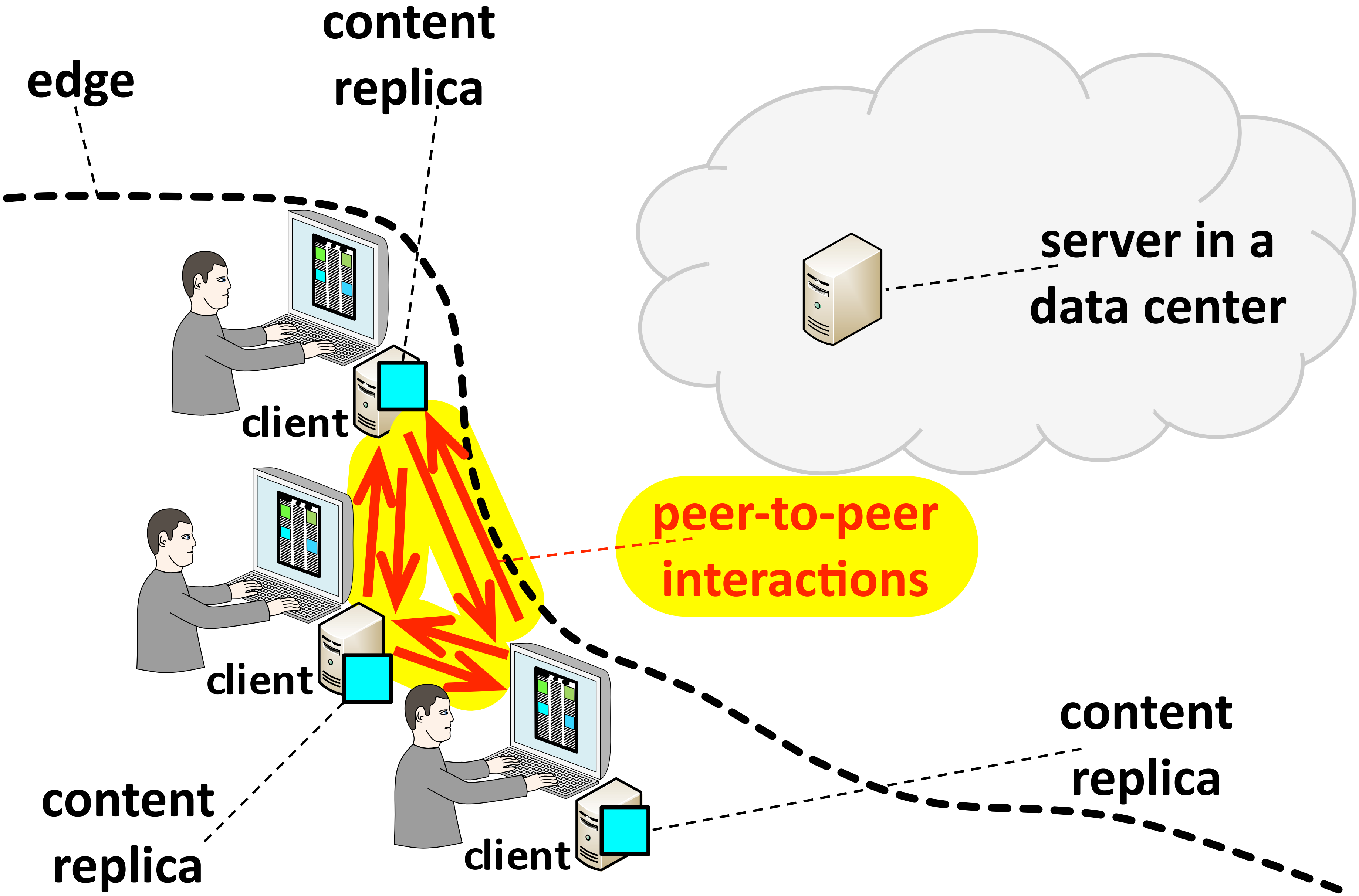
Storing Content at the Edge



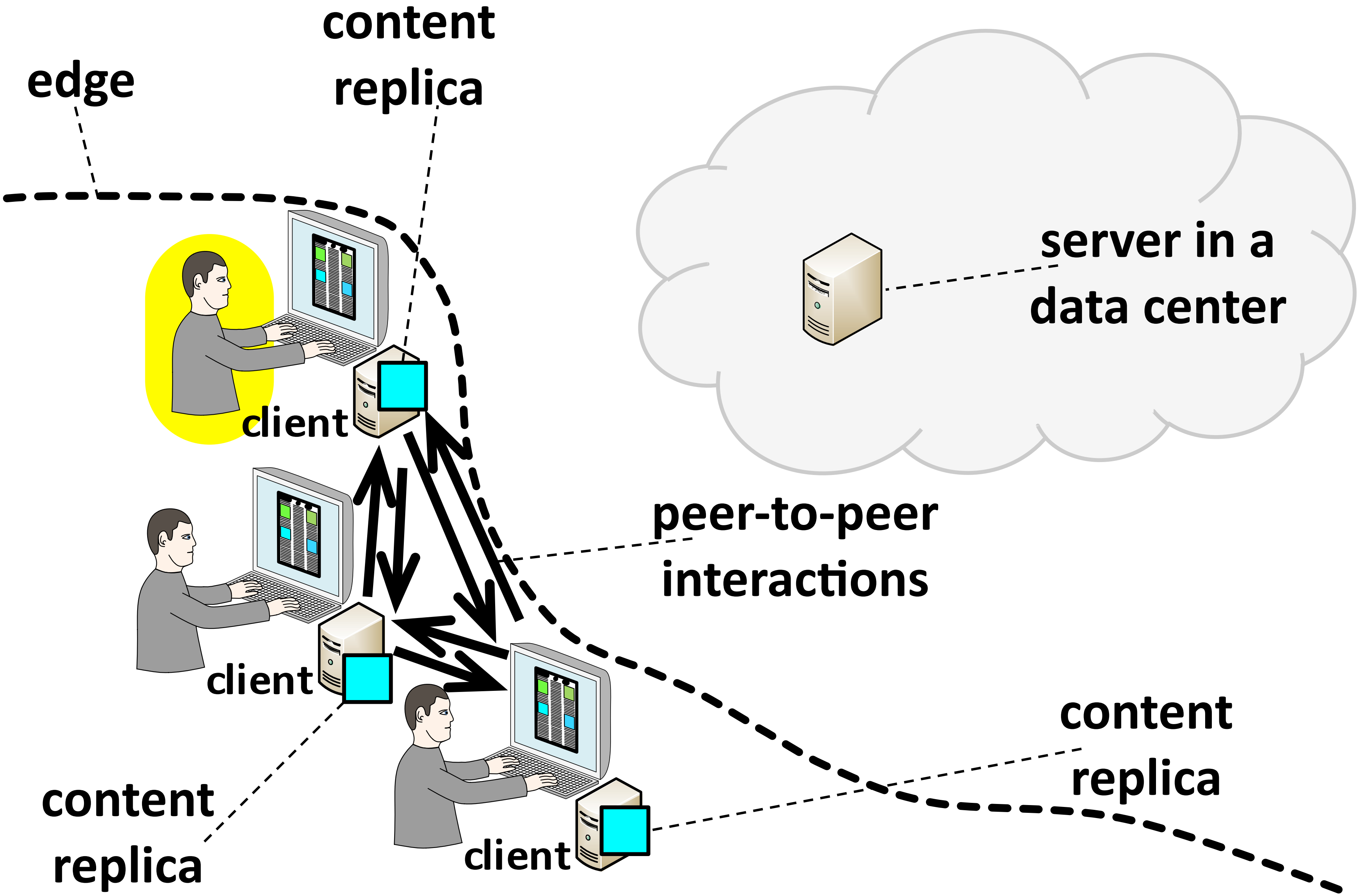
Storing Content at the Edge



Storing Content at the Edge



Storing Content at the Edge



edge

content replica

server in a data center

client

client

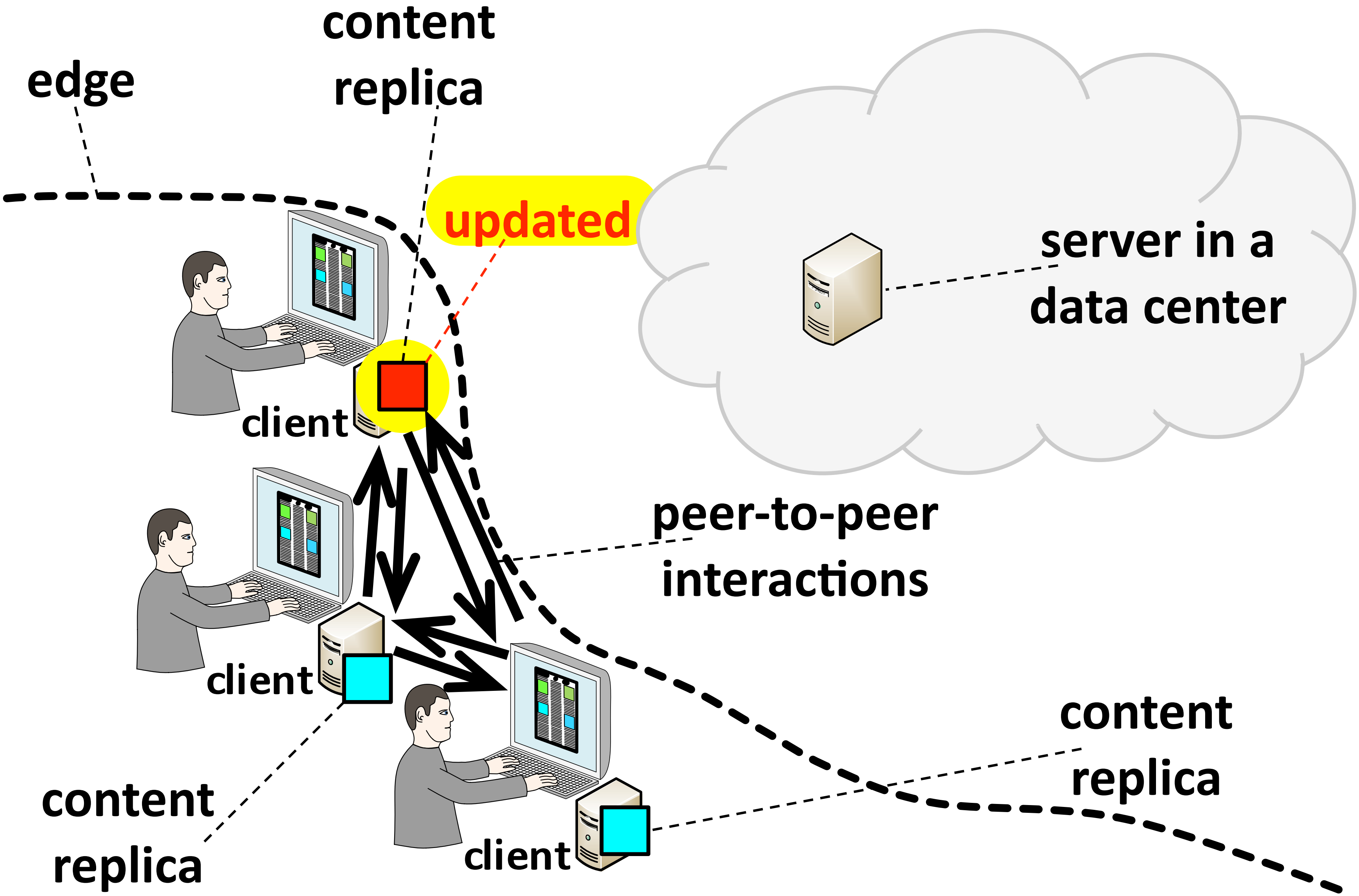
client

peer-to-peer interactions

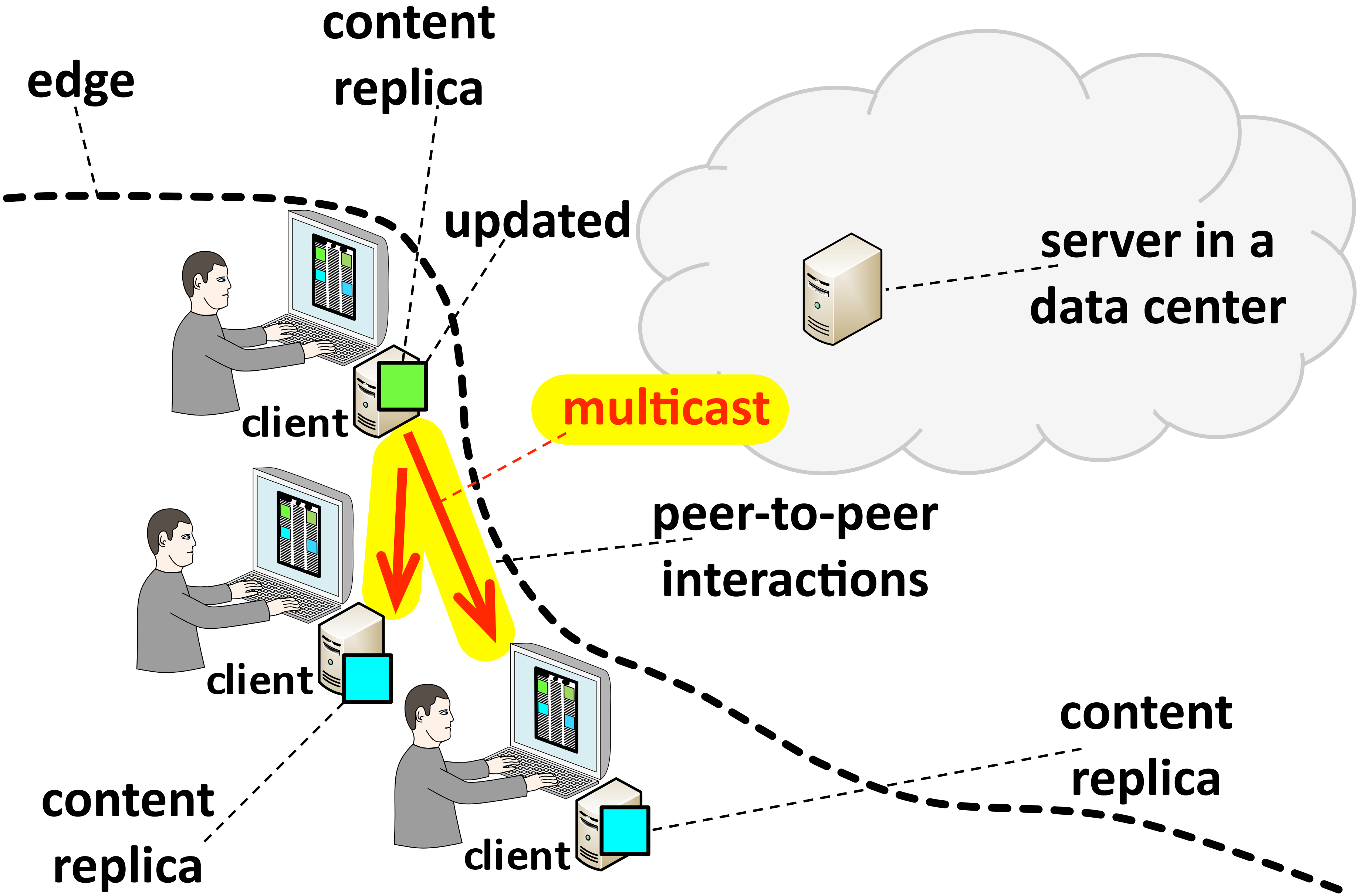
content replica

content replica

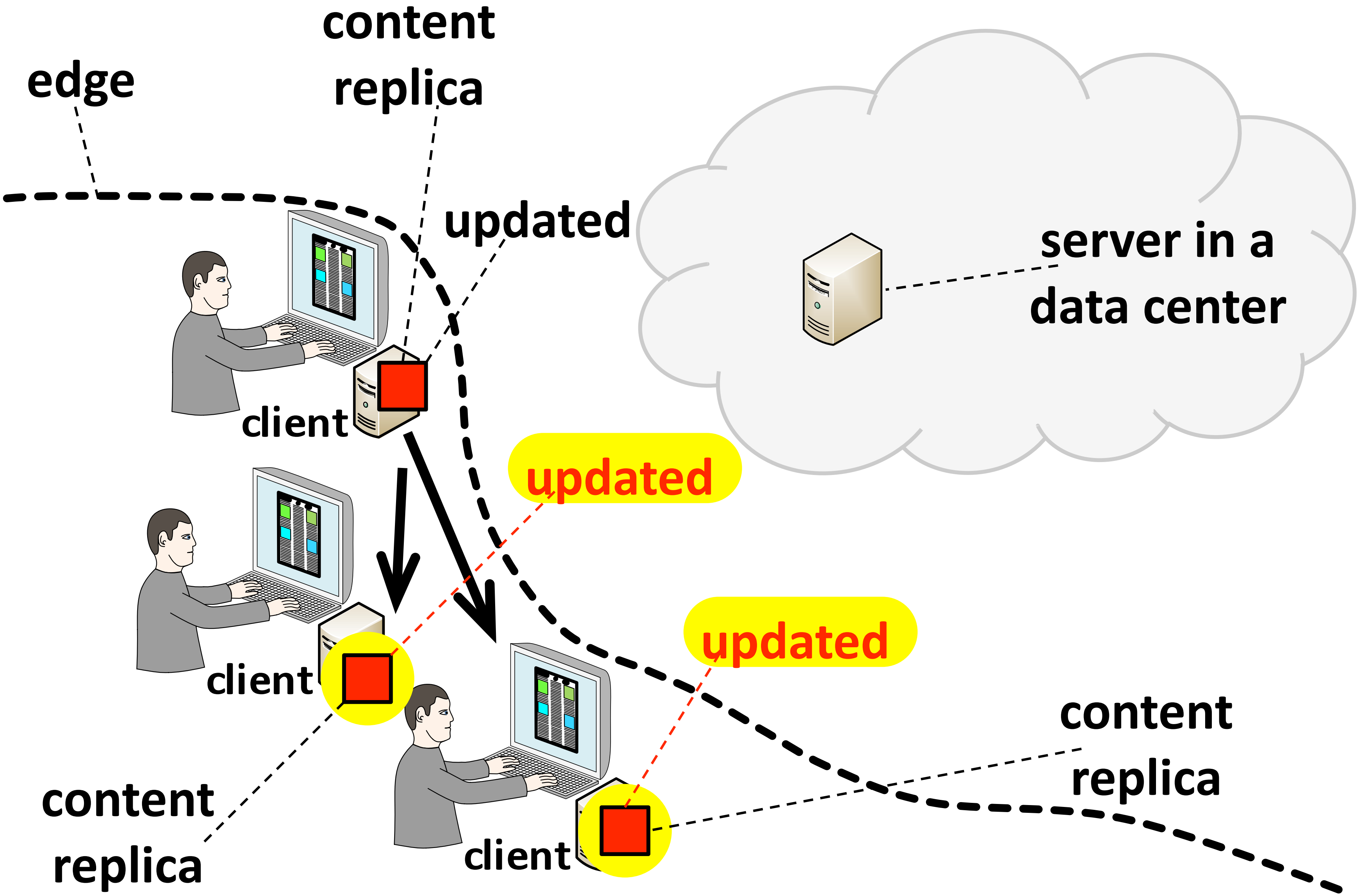
Storing Content at the Edge



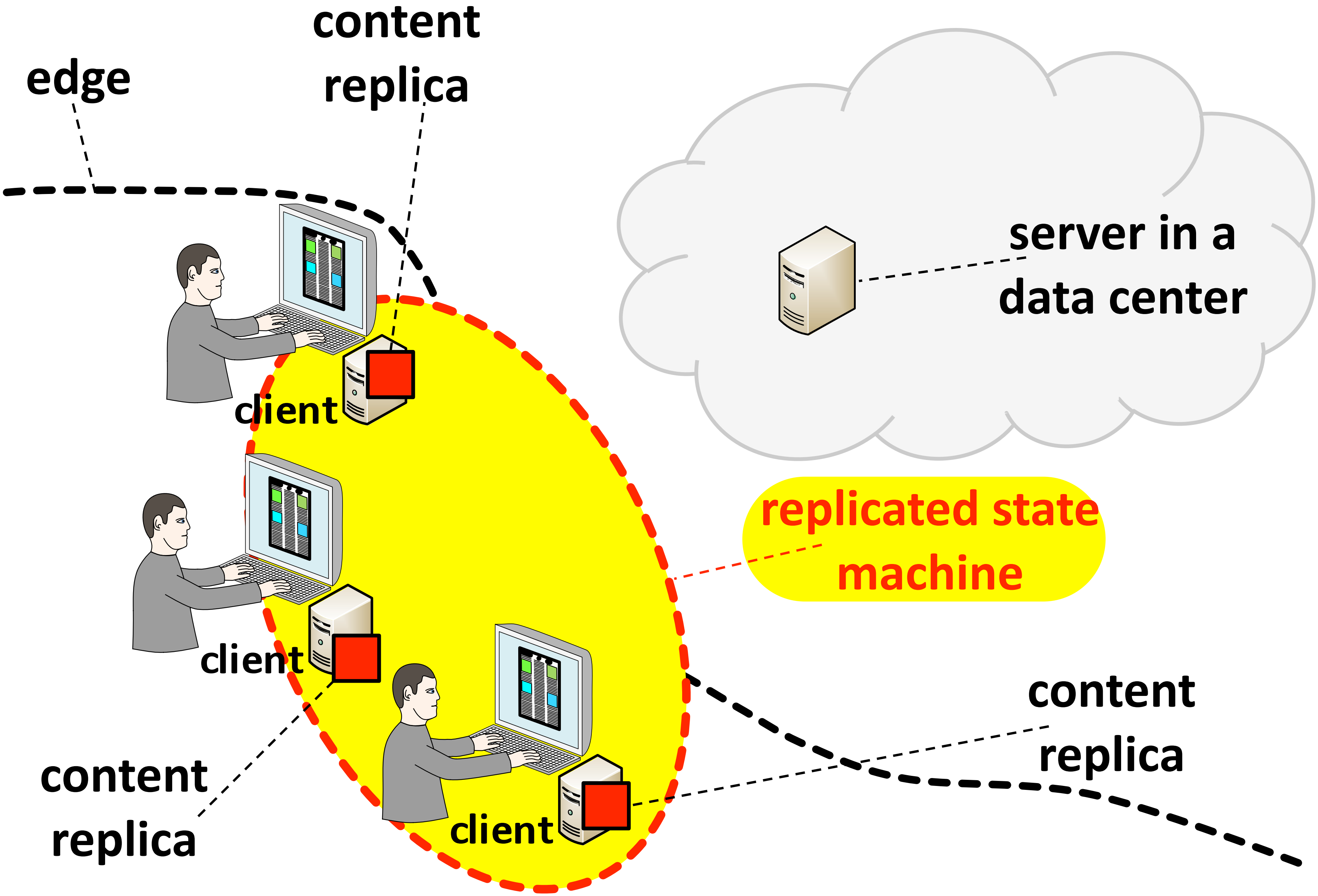
Storing Content at the Edge



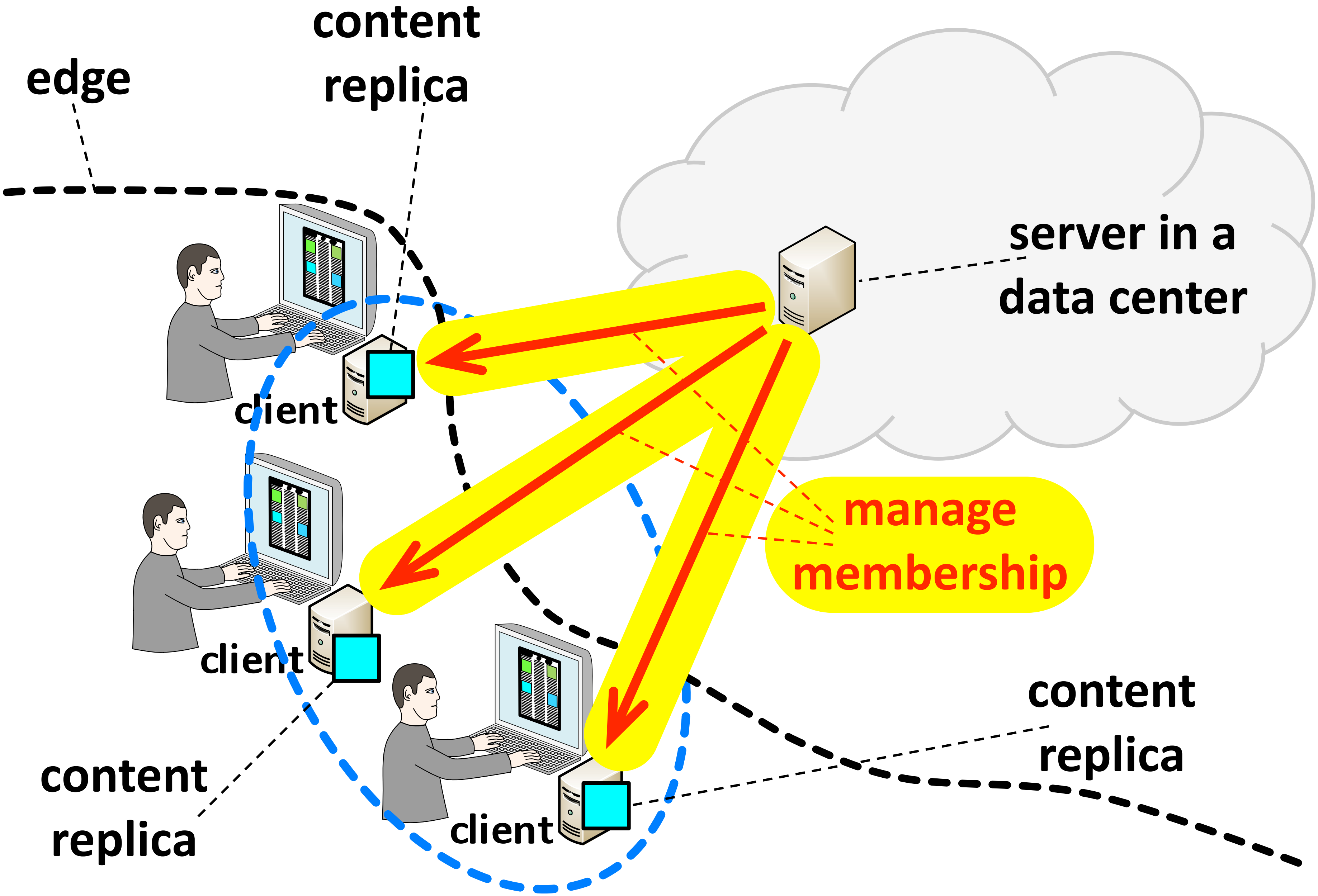
Storing Content at the Edge



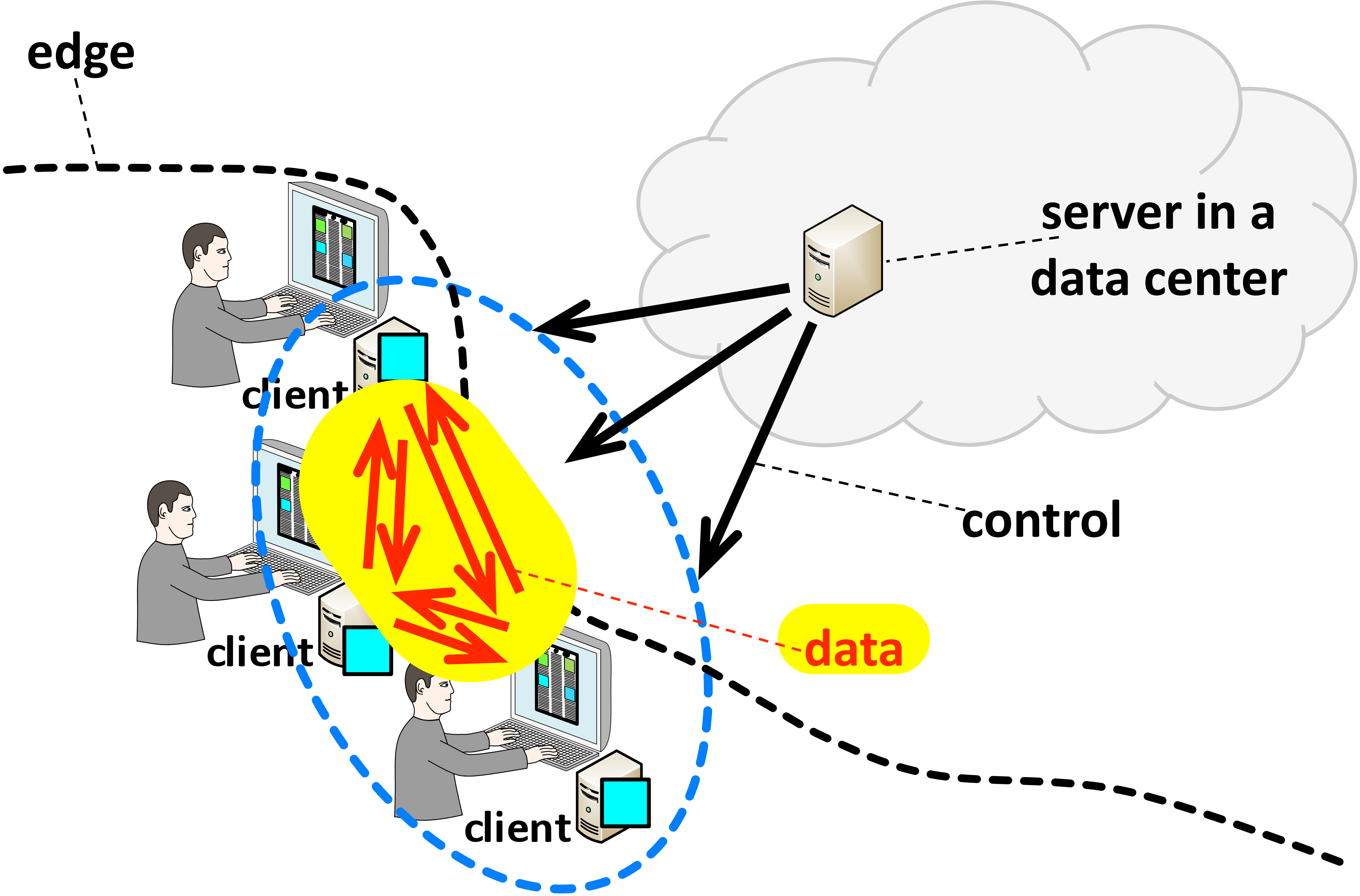
Storing Content at the Edge



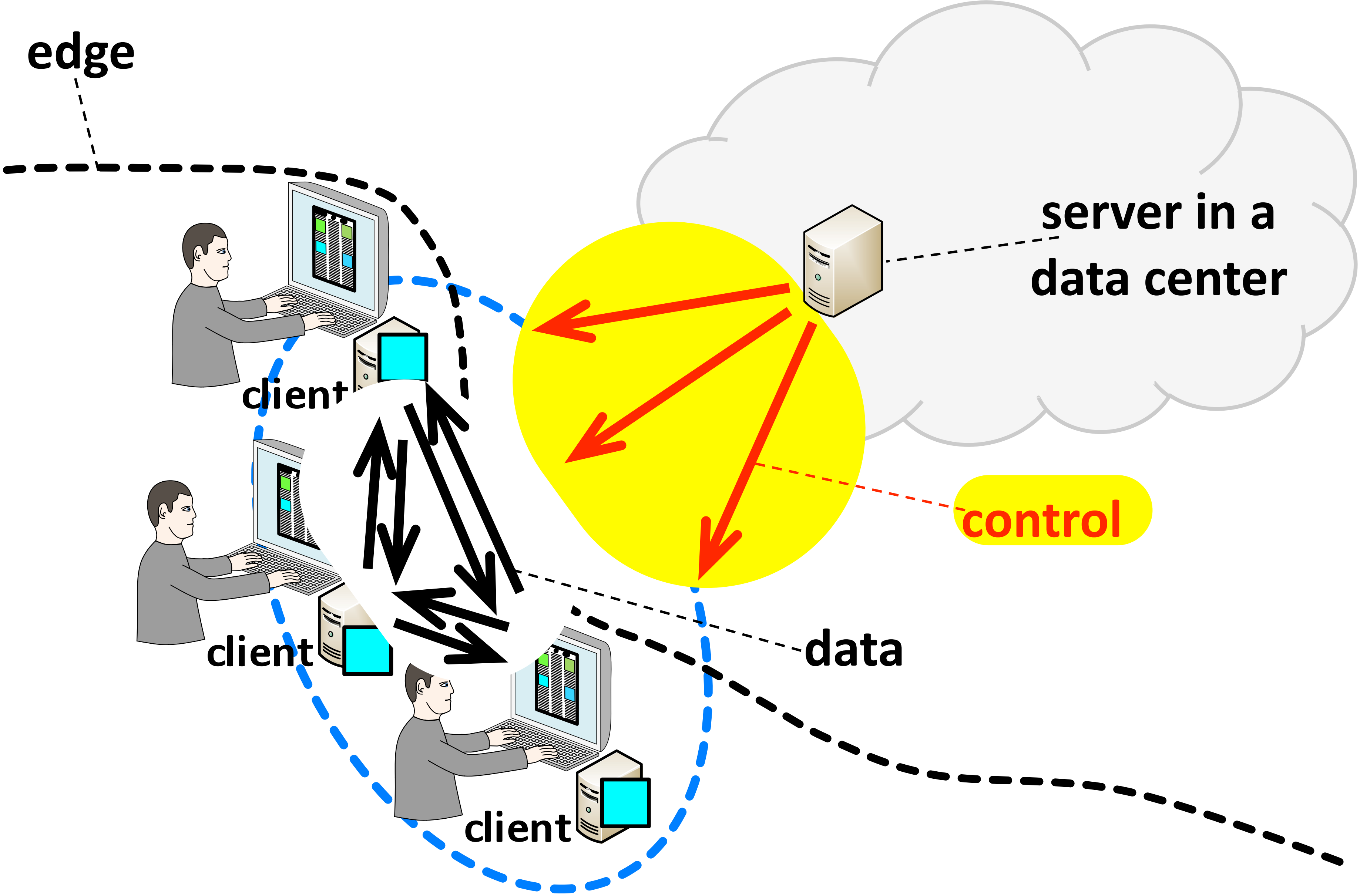
Storing Content at the Edge



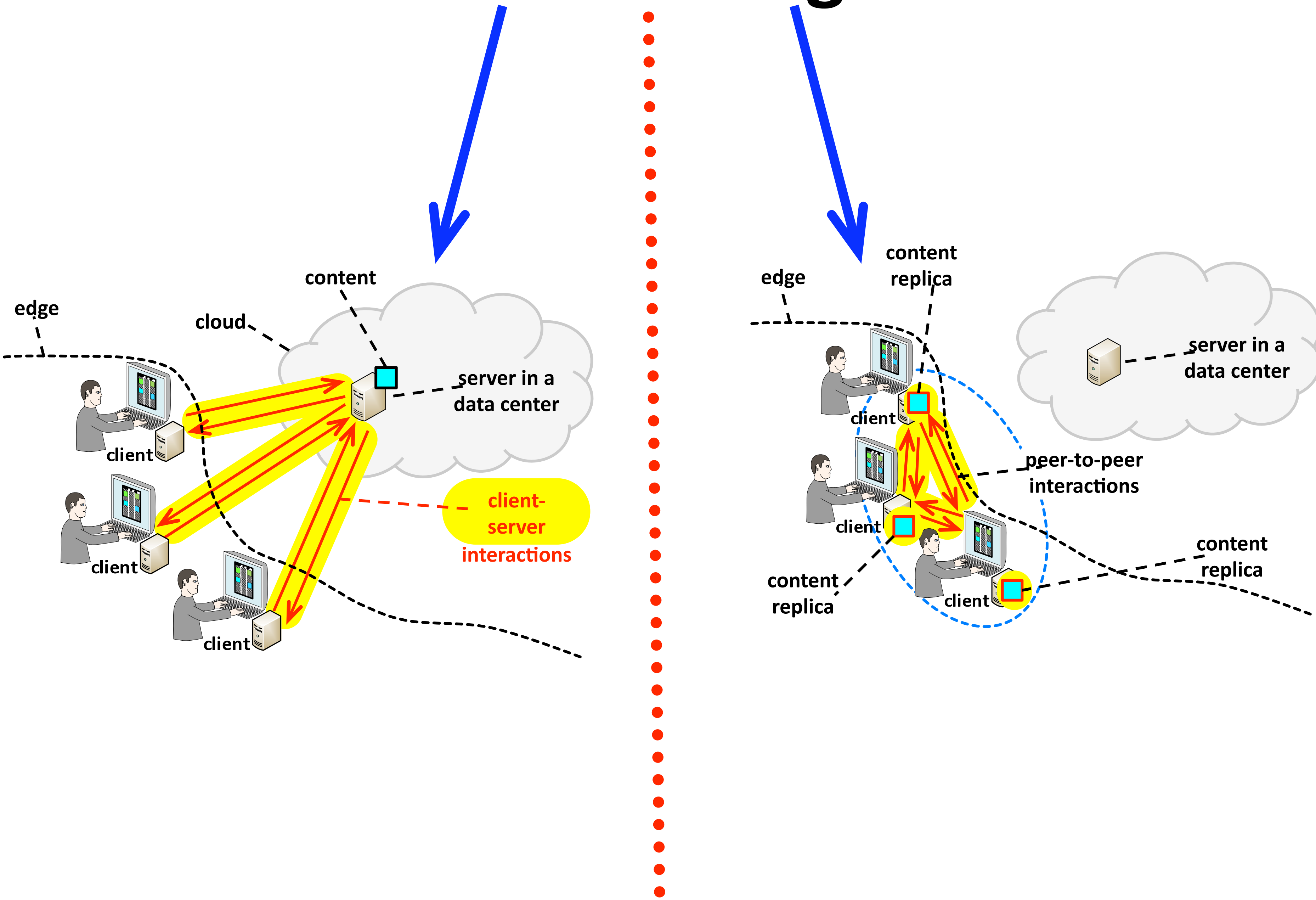
Storing Content at the Edge



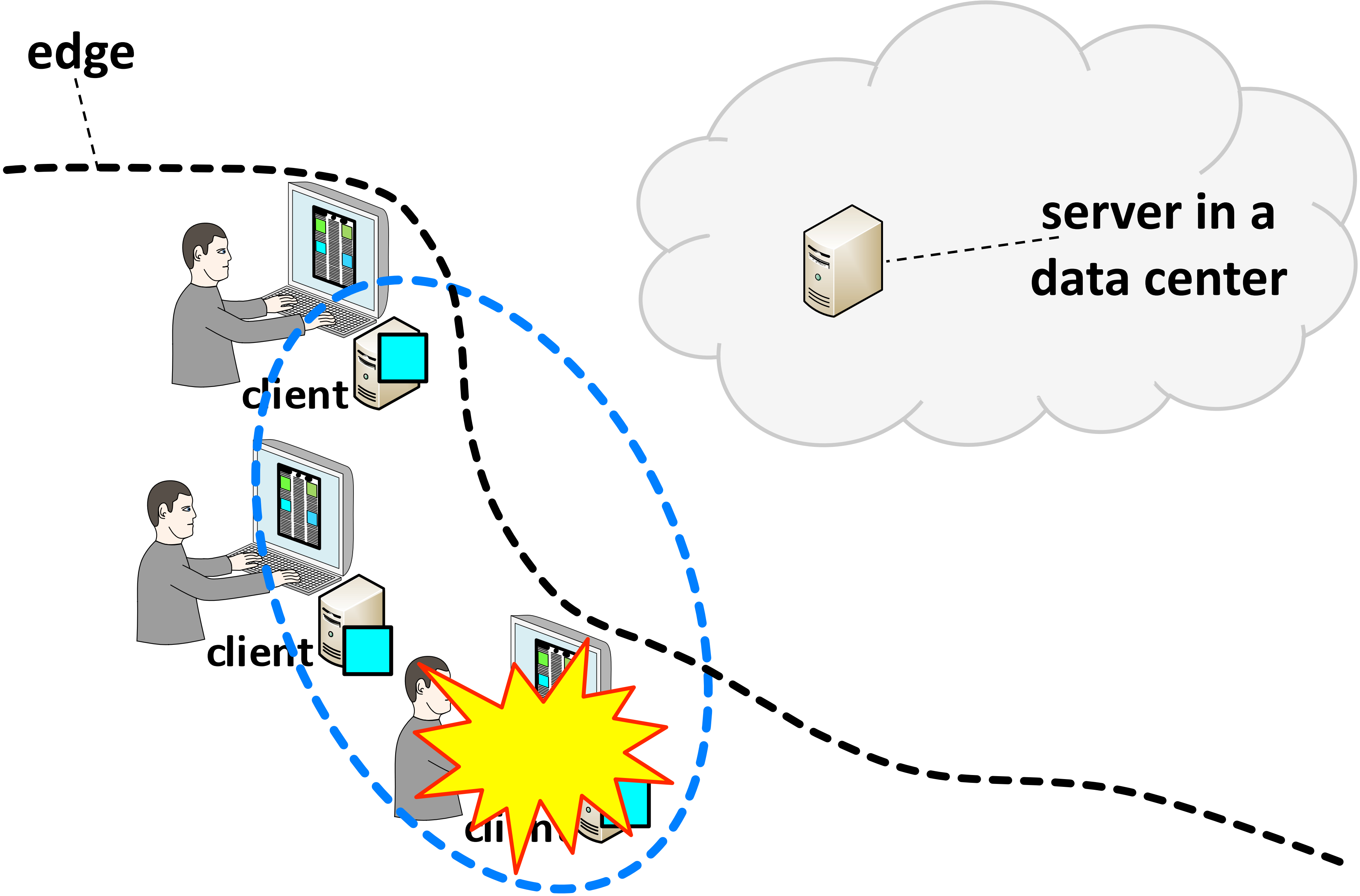
Storing Content at the Edge



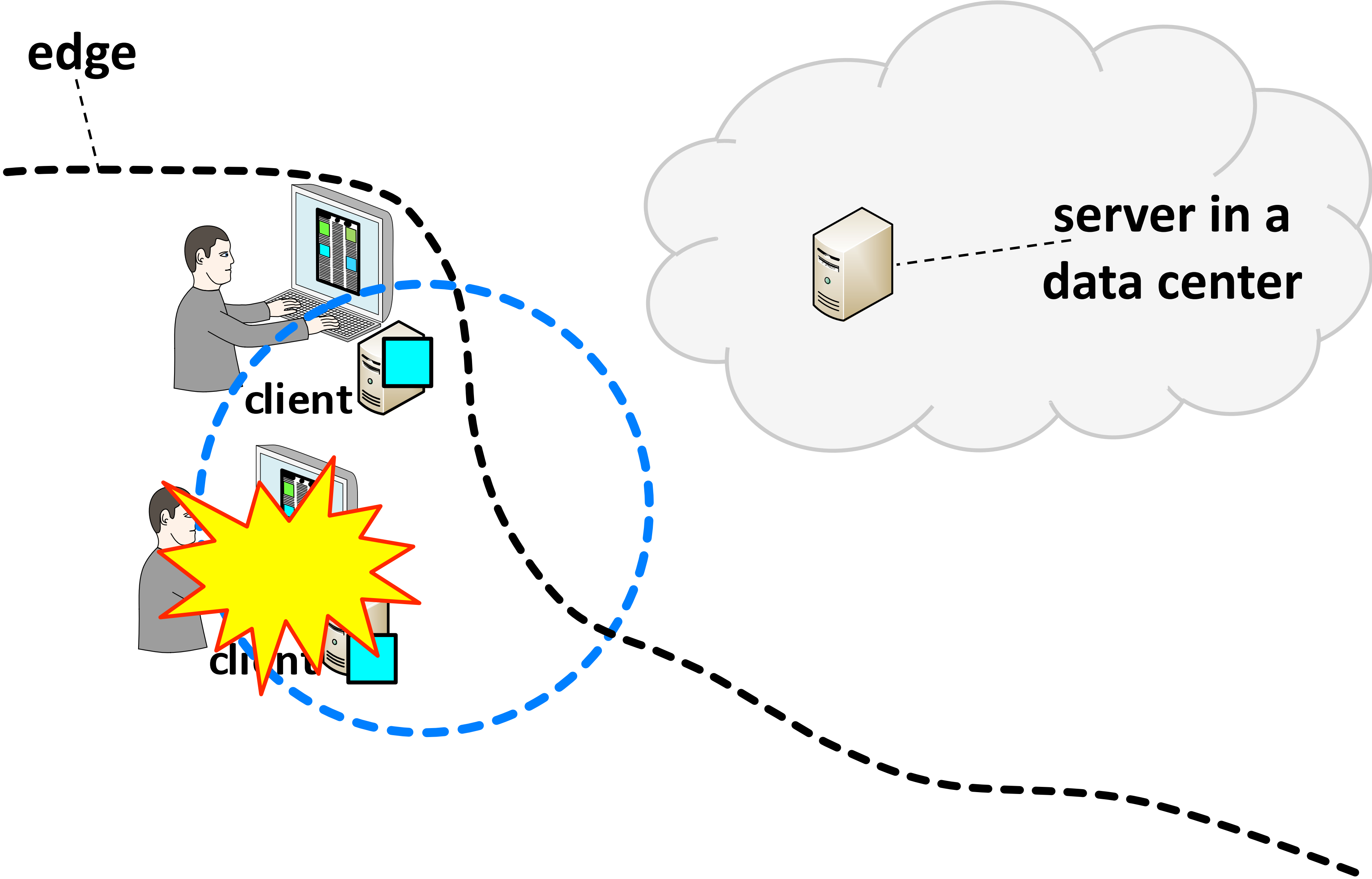
Cloud vs. Edge?



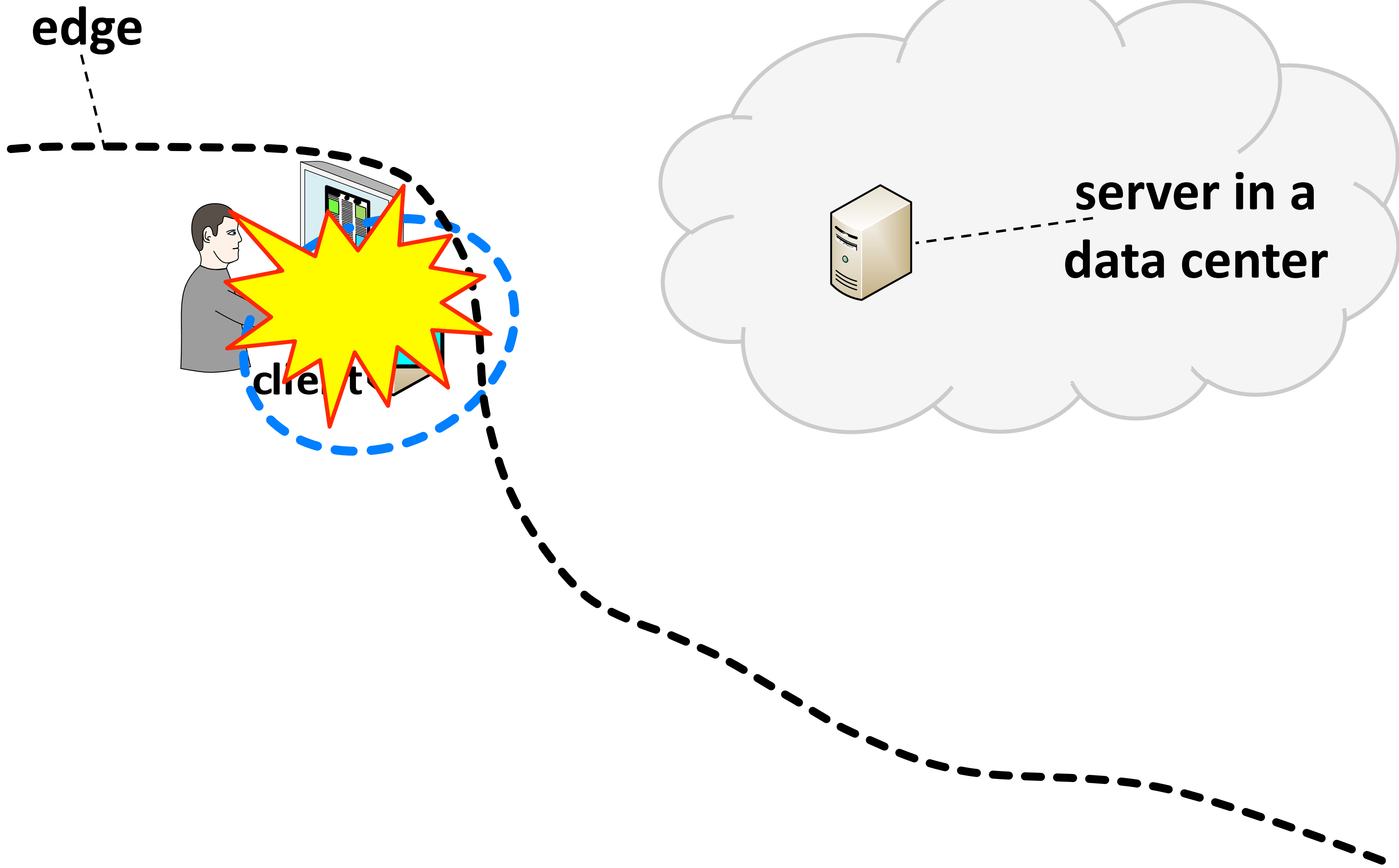
Cloud vs. Edge?



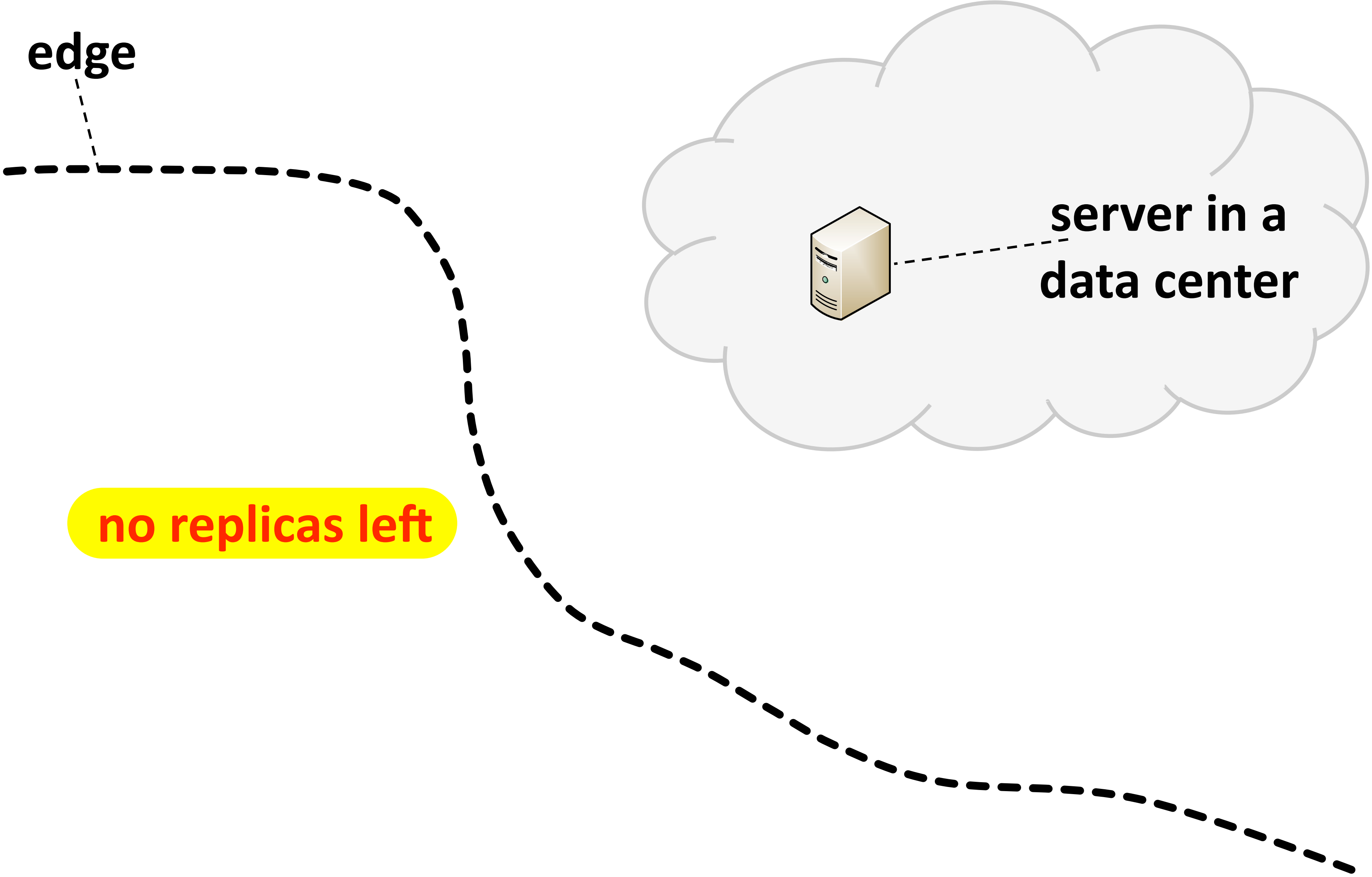
Cloud vs. Edge?



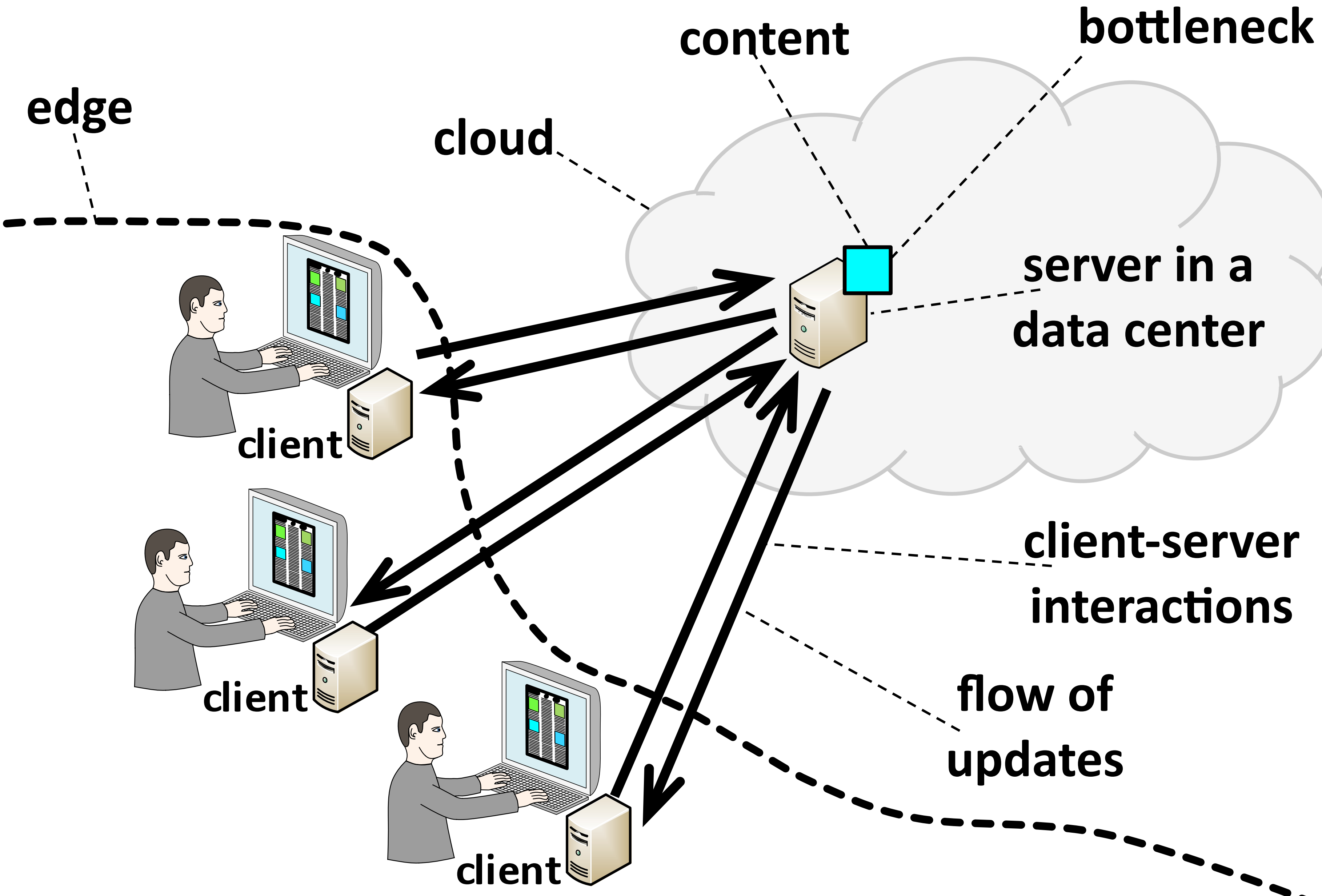
Cloud vs. Edge?



Cloud vs. Edge?



Cloud vs. Edge?



content

bottleneck

edge

cloud

server in a data center

client

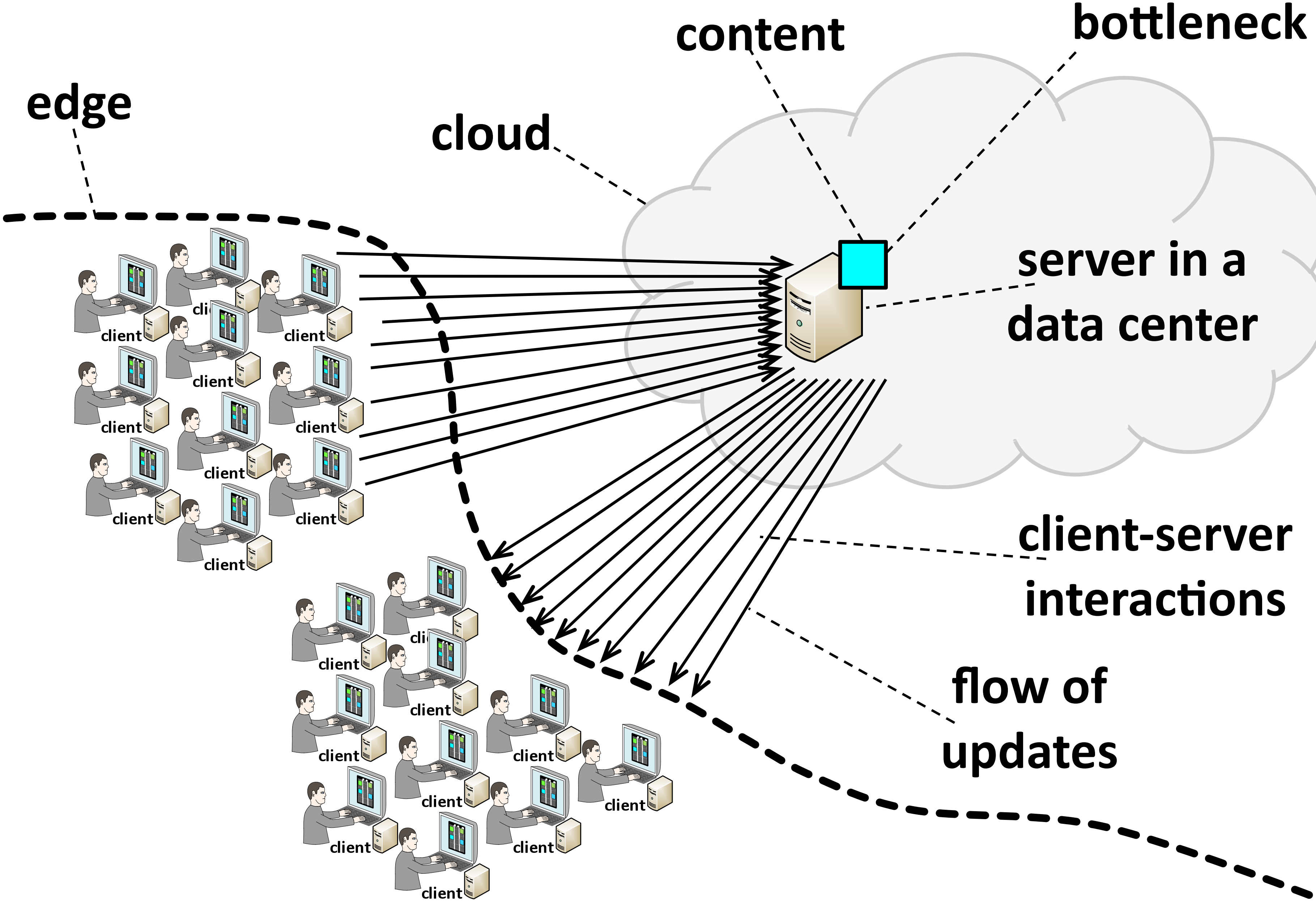
client

client

client-server interactions

flow of updates

Cloud vs. Edge?



edge

cloud

content

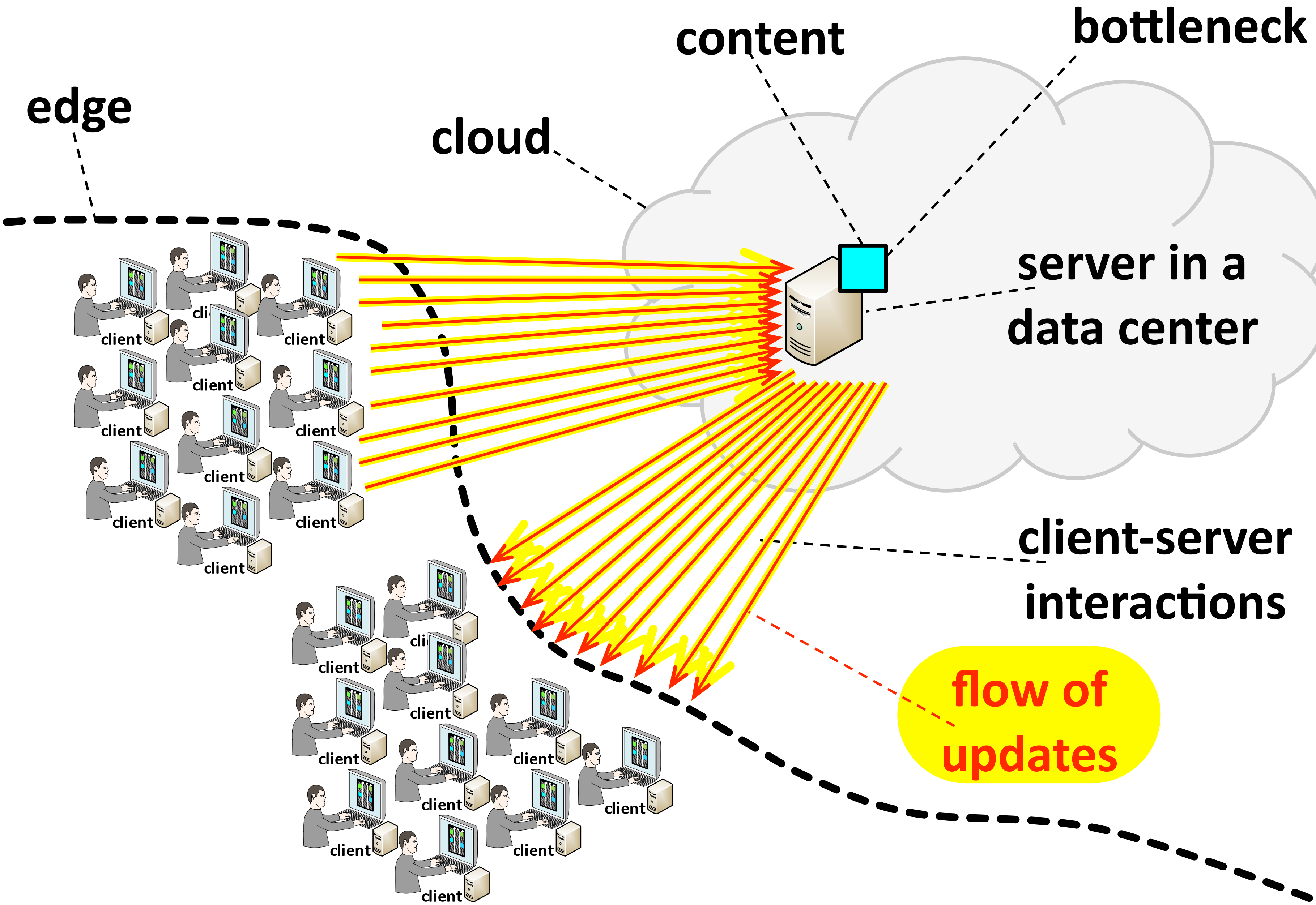
bottleneck

server in a data center

client-server interactions

flow of updates

Cloud vs. Edge?



edge

cloud

content

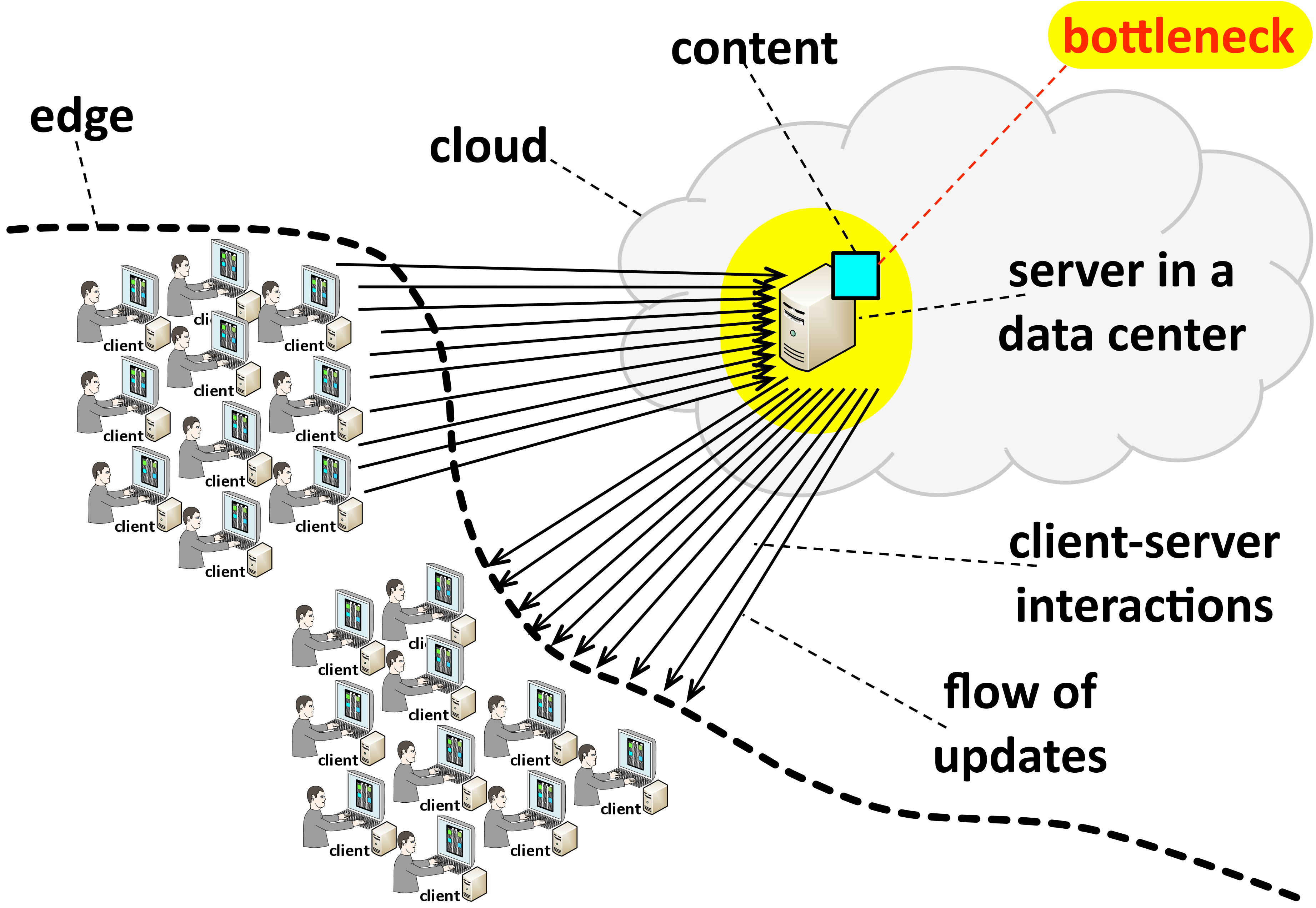
bottleneck

server in a data center

client-server interactions

flow of updates

Cloud vs. Edge?



content

bottleneck

edge

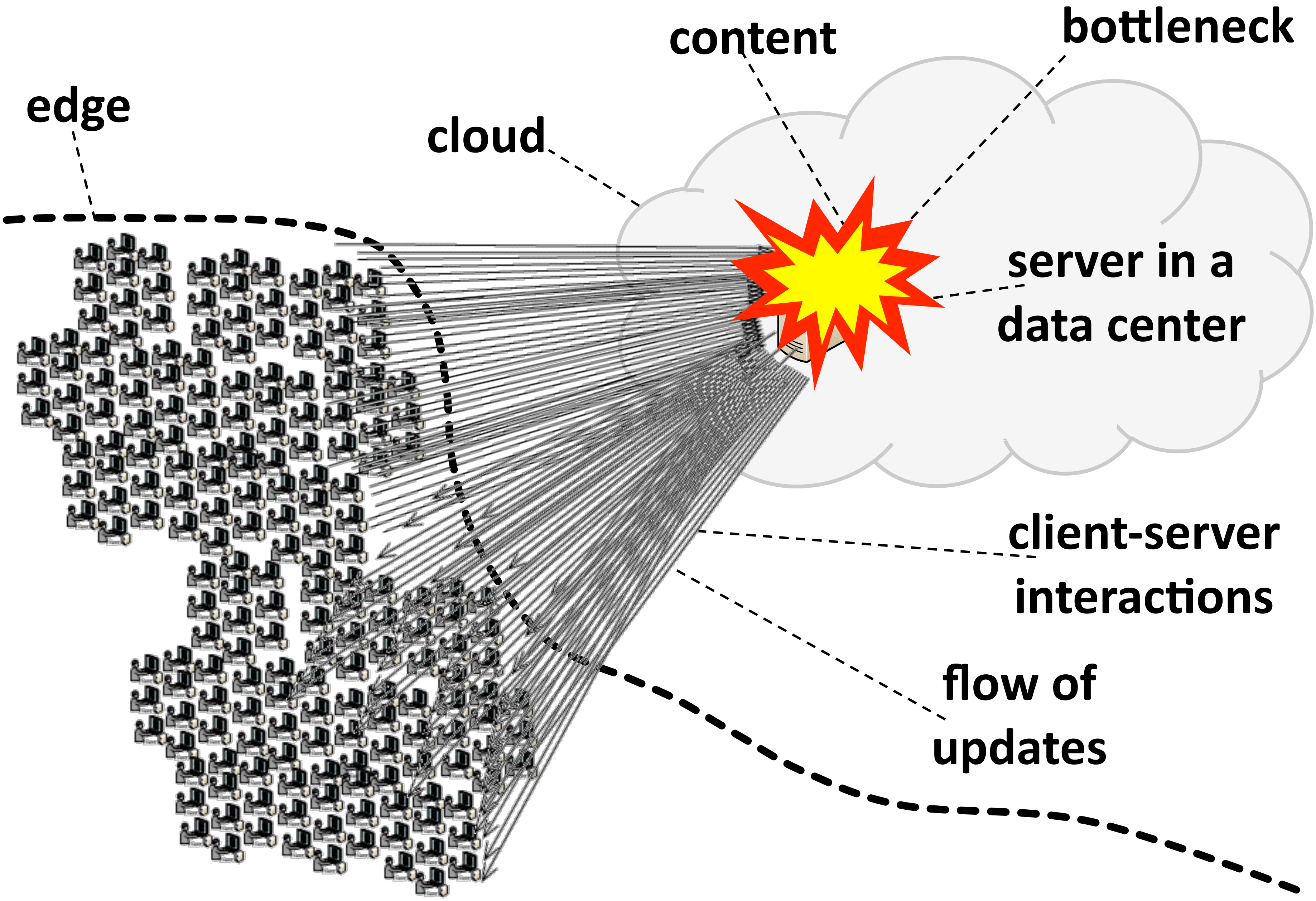
cloud

server in a data center

client-server interactions

flow of updates

Cloud vs. Edge?

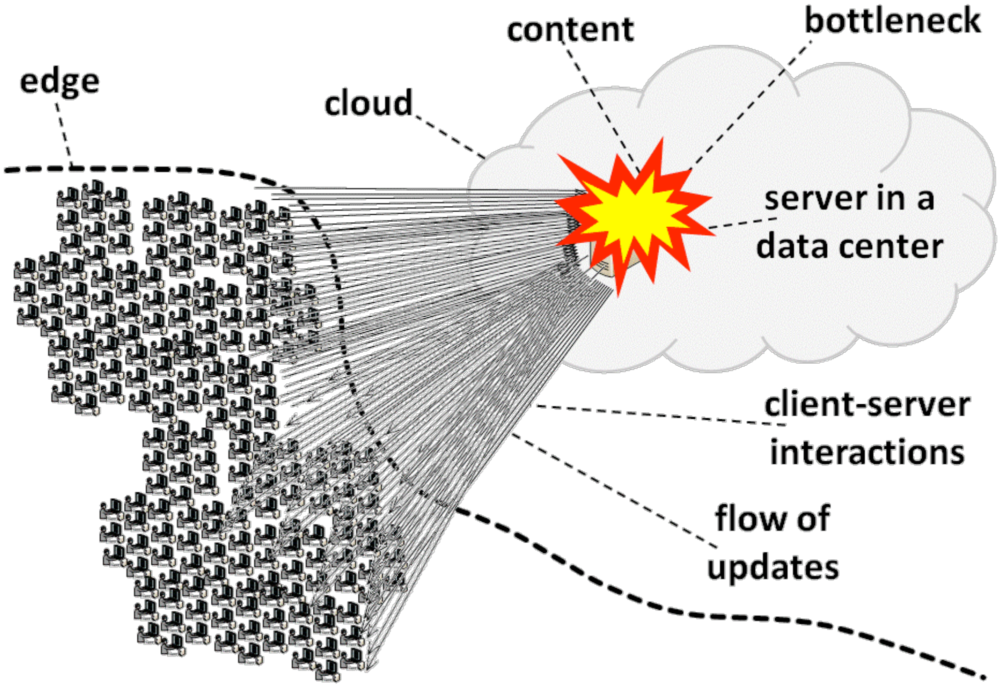


Cloud vs. Edge?

Server capacity:

- SecondLife: ~40 clients/server

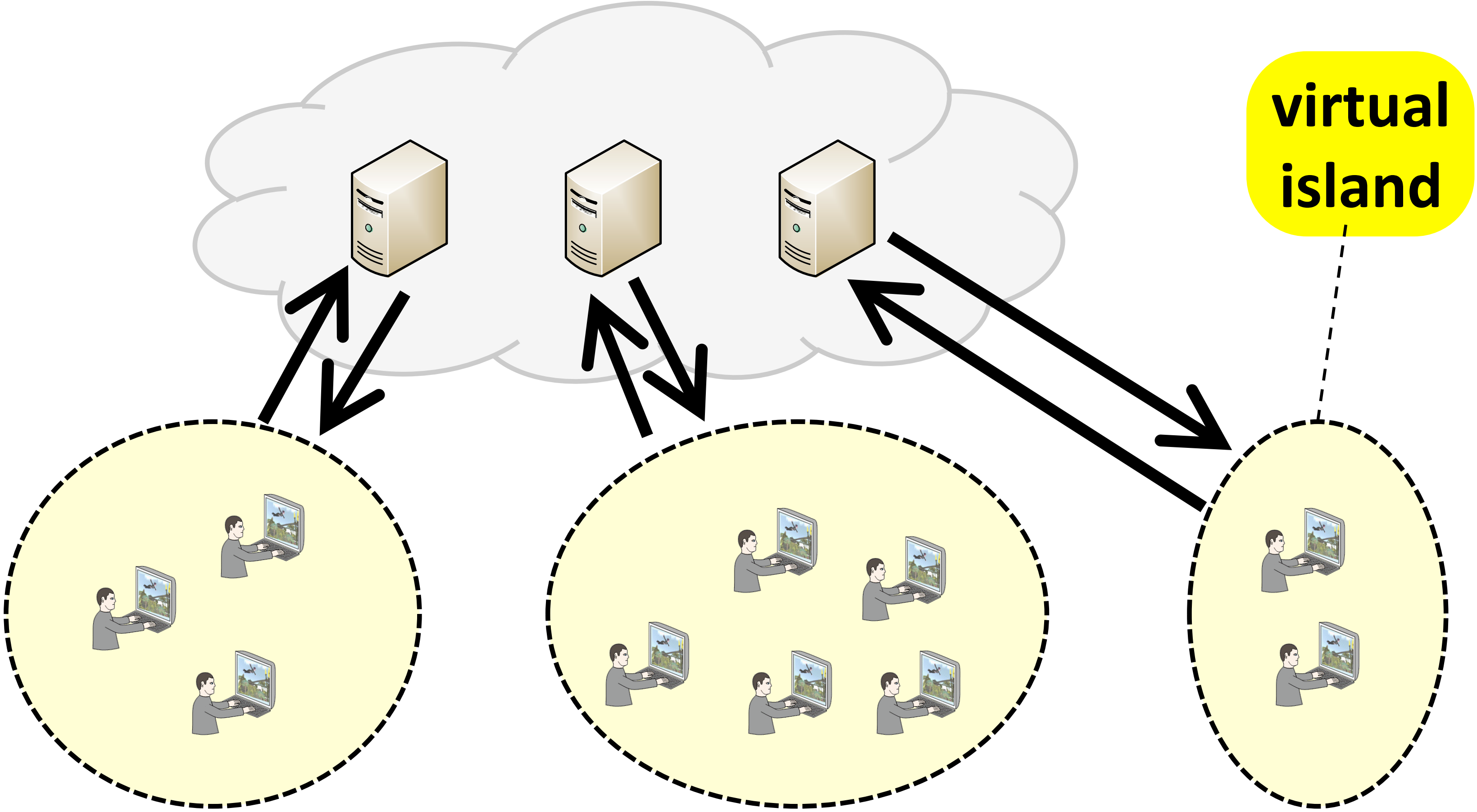
Second Life and the New Generation of Virtual Worlds.
S. Kumar et al. 2008. IEEE Computer, 41(9):46-53



Cloud vs. Edge?

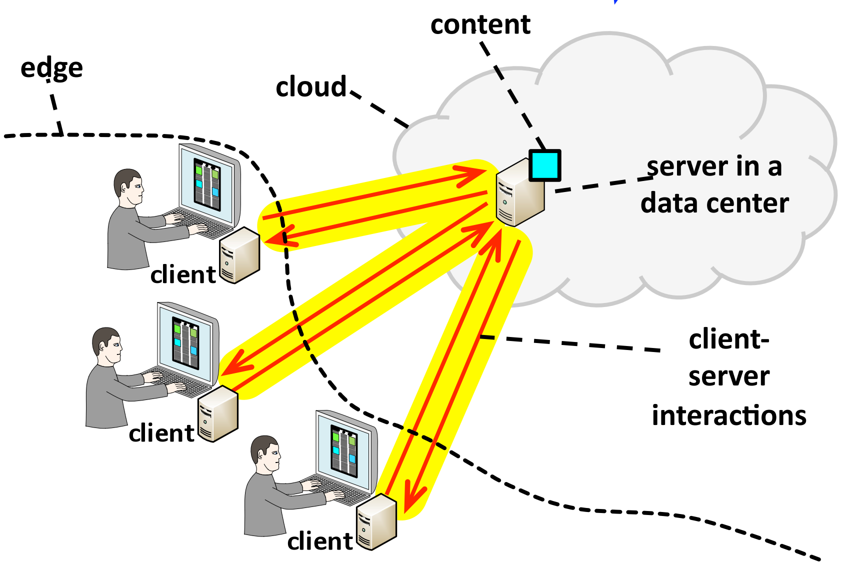
Server capacity:

- **SecondLife: ~40 clients/server**

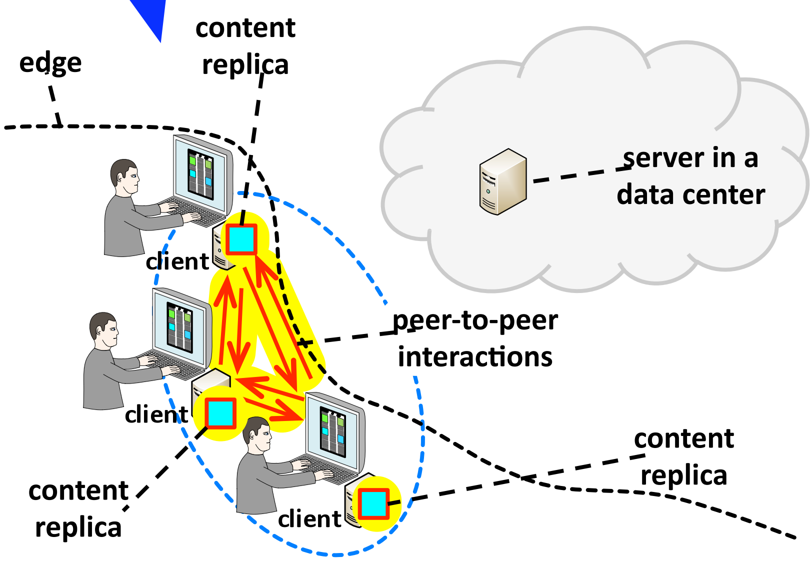


Cloud ♥ Edge?

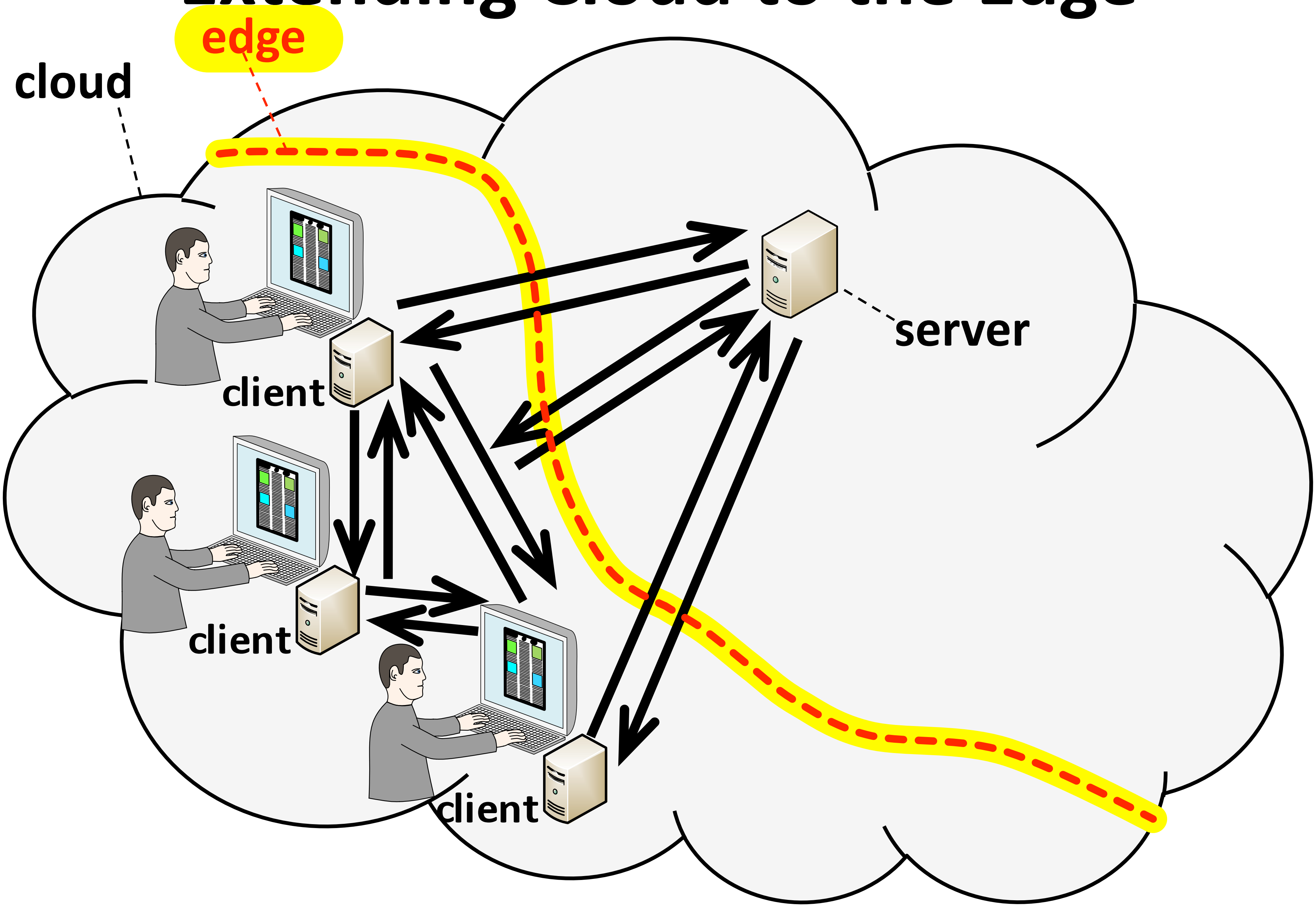
persistence and scalability



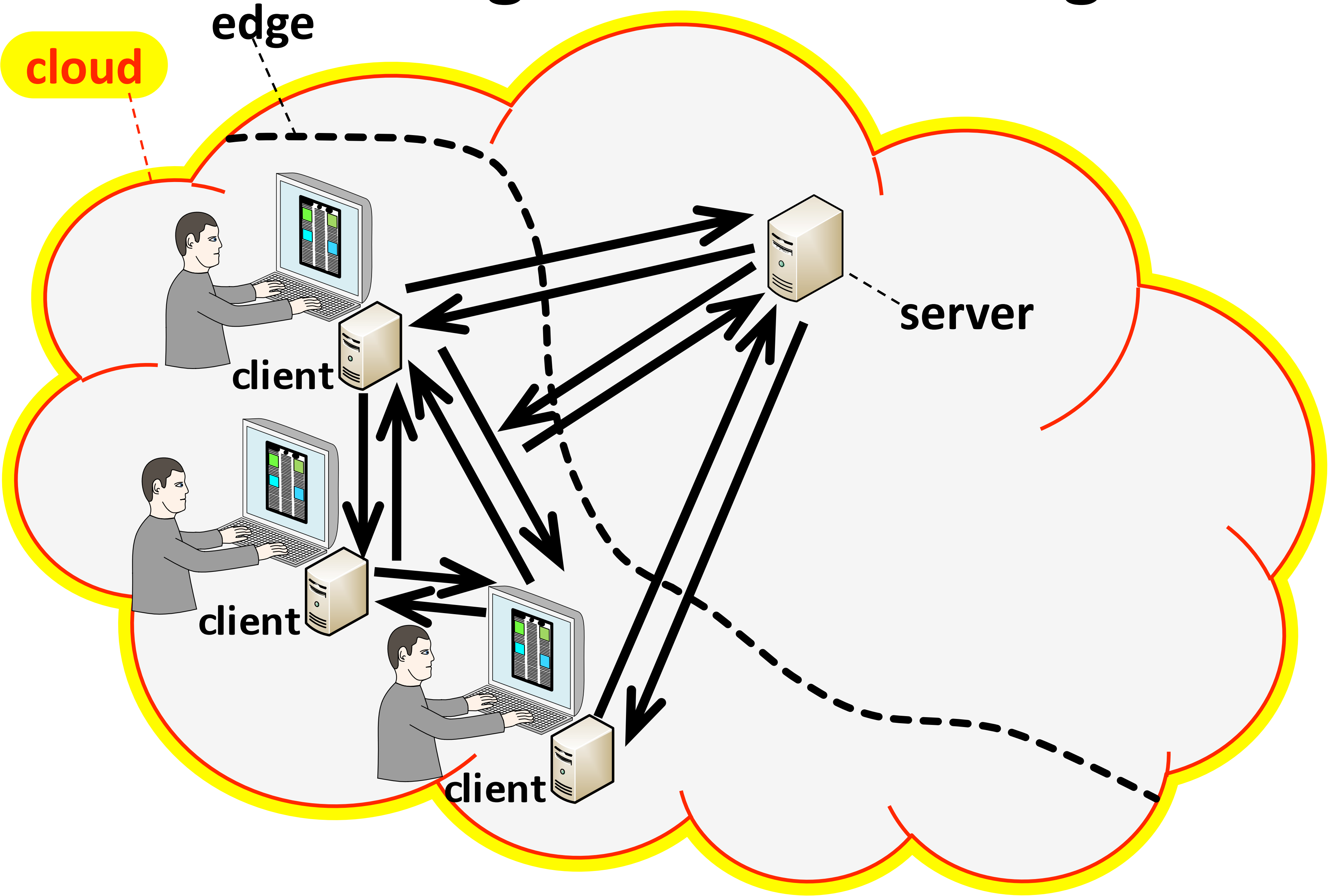
+



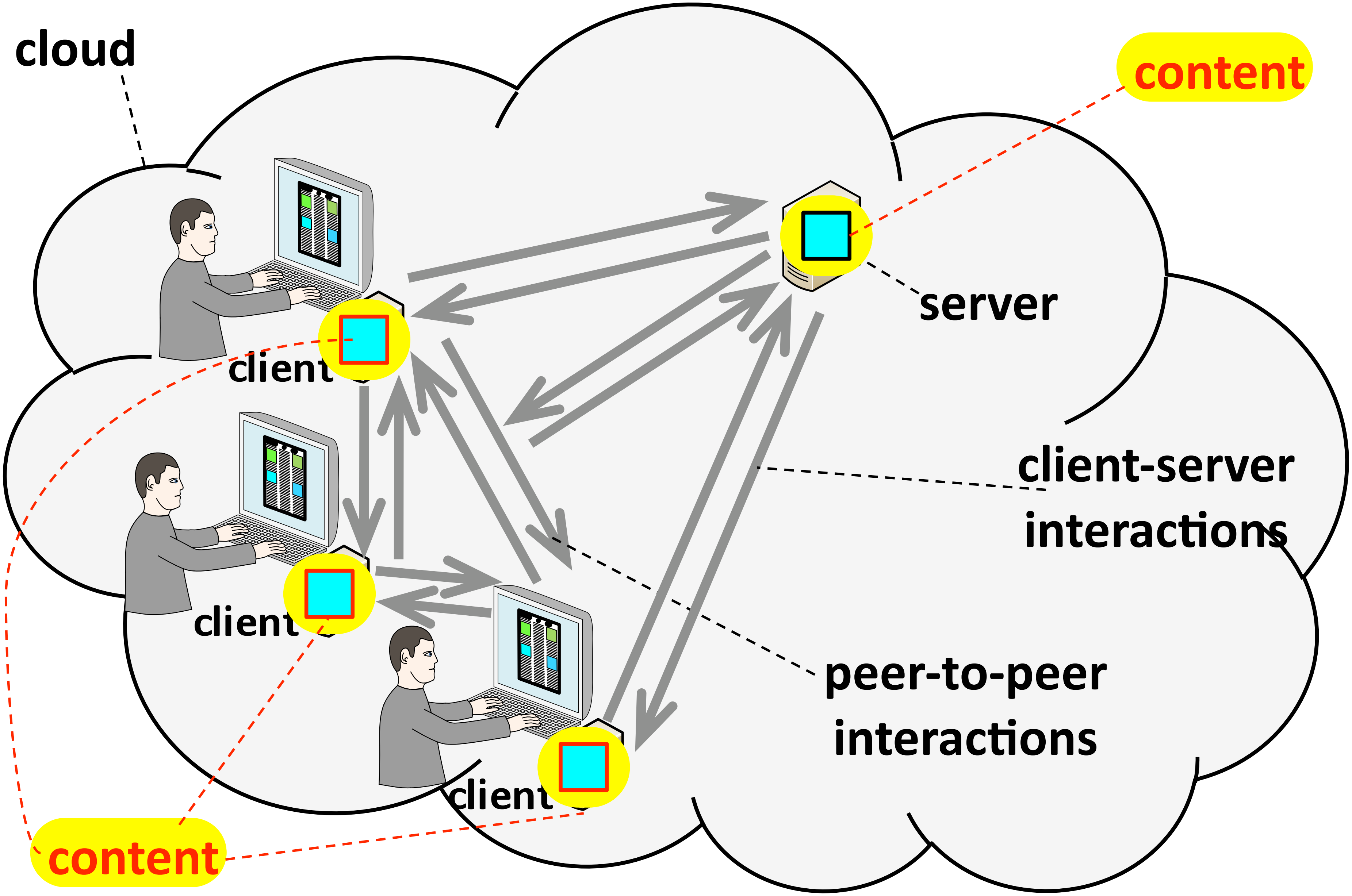
Extending Cloud to the Edge



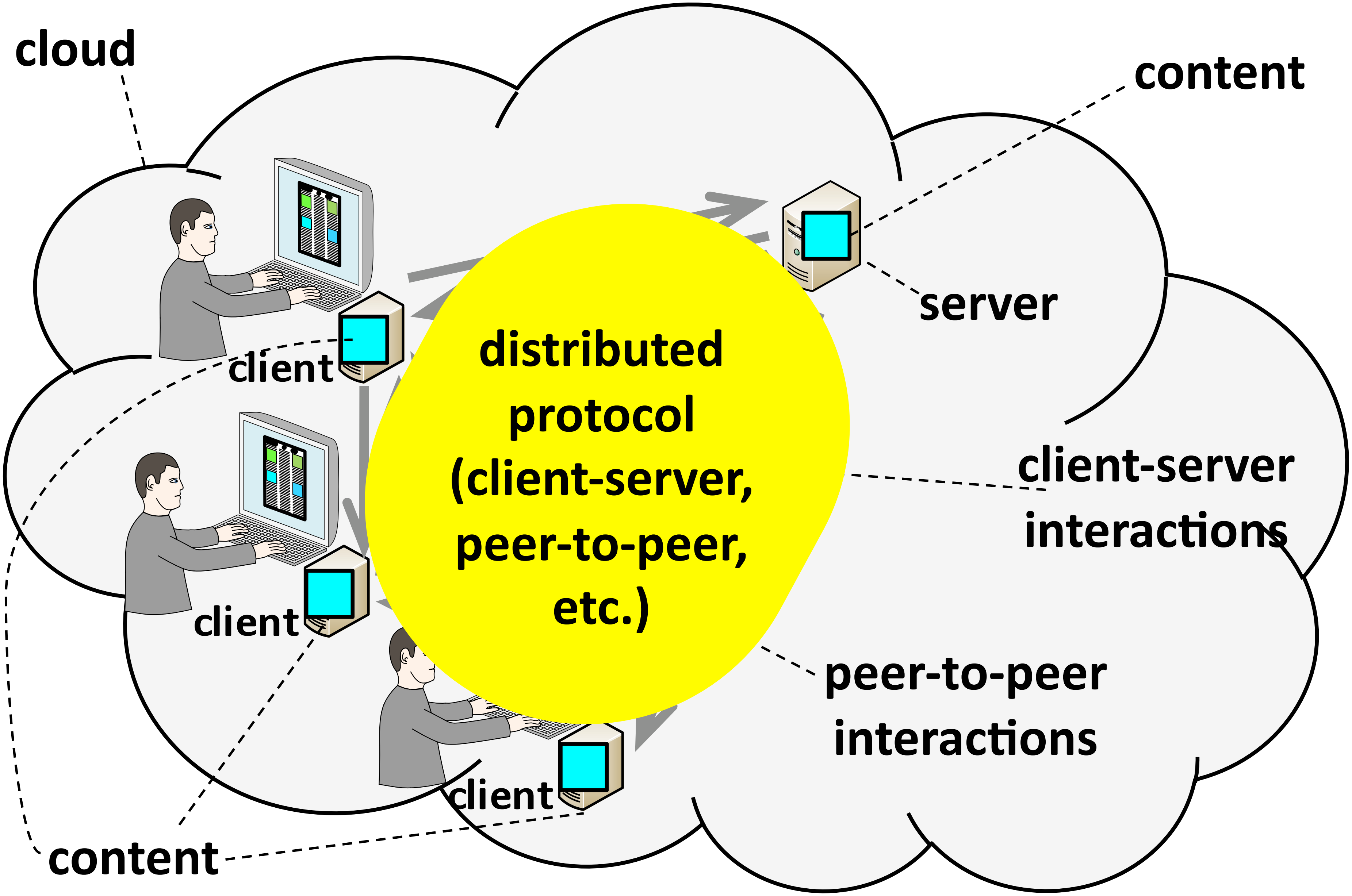
Extending Cloud to the Edge



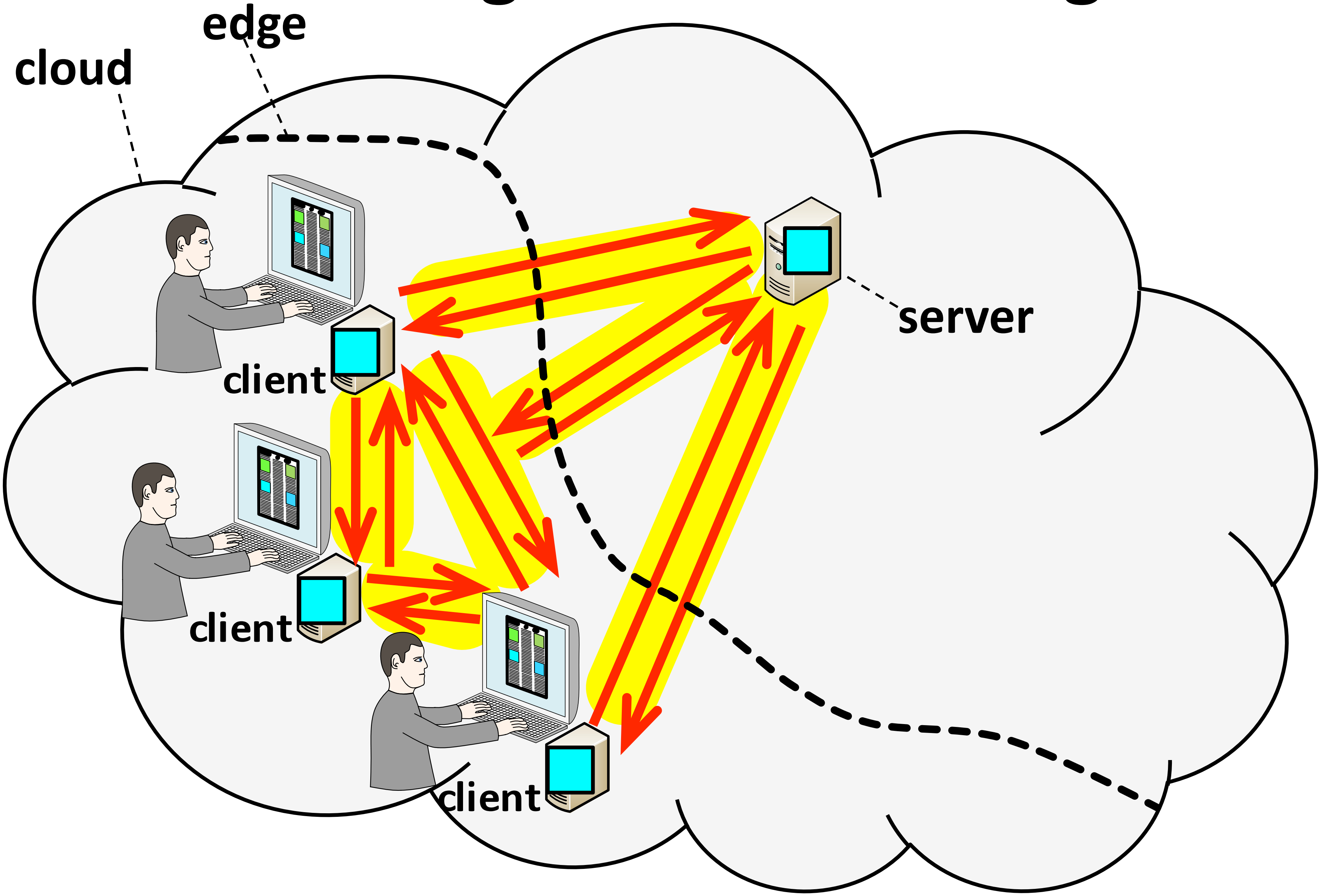
Extending Cloud to the Edge



Extending Cloud to the Edge


cloud
content
server
client
client
client
distributed protocol (client-server, peer-to-peer, etc.)
client-server interactions
peer-to-peer interactions
content

Extending Cloud to the Edge



Edge is Larger than the Cloud

There are:

- 40+ times more PCs than servers (billions today, to double in five years)
- 2000+ client PCs per Google server
- 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized

Edge is Larger than the Cloud

There are:

- **40+ times more PCs than servers (billions today, to double in five years)**
- **2000+ client PCs per Google server**
- **40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft**

=> a lot of computational power and resources at the edge that are greatly underutilized

Edge is Larger than the Cloud

There are:

- 40+ times more PCs than servers (billions today, to double in five years)
- 2000+ client PCs per Google server
- 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized

Edge is Larger than the Cloud

There are:

- 40+ times more PCs than servers (billions today, to double in five years)
- 2000+ client PCs per Google server
- 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized

Edge is Larger than the Cloud

There are:

- 40+ times more PCs than servers (billions today, to double in five years)
- 2000+ client PCs per Google server
- 40,000+ times more client PCs purchased each year by home users than new servers purchased each year by Microsoft

=> a lot of computational power and resources at the edge that are greatly underutilized

New Storage Abstraction

Unify different content access models:

- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams

New Storage Abstraction

Unify different content access models:

- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams

New Storage Abstraction

Unify different content access models:

- **Centralized vs. replicated**
  - **Centralized: web service, database**
  - **Replicated: distributed P2P replication protocols**
- **Persistent vs. temporary**
  - **Persistent: server in a data center**
  - **Temporary: collaboration session formed ad hoc**
- **Server-side vs. client-side**
- **Shared public vs. shared private**
  - **Public: web service or collaboration session**
  - **Private: session state, cookies, etc.**
- **Stateful vs. stateless**
  - **Stateful: variables, files, objects, database tables**
  - **Stateless: video or notification streams**

New Storage Abstraction

Unify different content access models:

- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams

New Storage Abstraction

Unify different content access models:

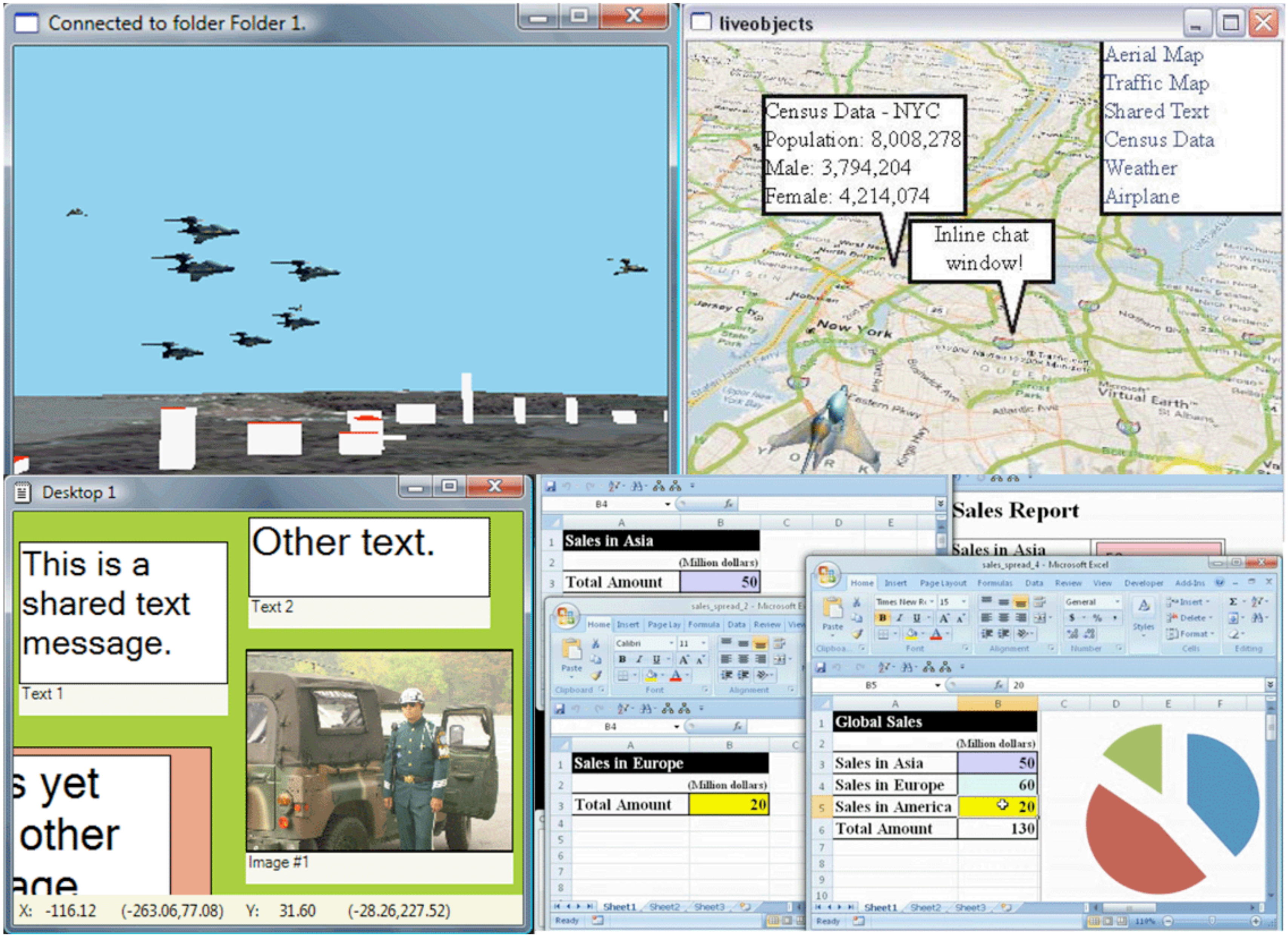
- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams

New Storage Abstraction

Unify different content access models:

- Centralized vs. replicated
  - Centralized: web service, database
  - Replicated: distributed P2P replication protocols
- Persistent vs. temporary
  - Persistent: server in a data center
  - Temporary: collaboration session formed ad hoc
- Server-side vs. client-side
- Shared public vs. shared private
  - Public: web service or collaboration session
  - Private: session state, cookies, etc.
- Stateful vs. stateless
  - Stateful: variables, files, objects, database tables
  - Stateless: video or notification streams

http://liveobjects.cs.cornell.edu



Census Data - NYC
Population: 8,008,278
Male: 3,794,204
Female: 4,214,074

Inline chat window!

This is a shared text message.

Other text.

s yet other age

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)

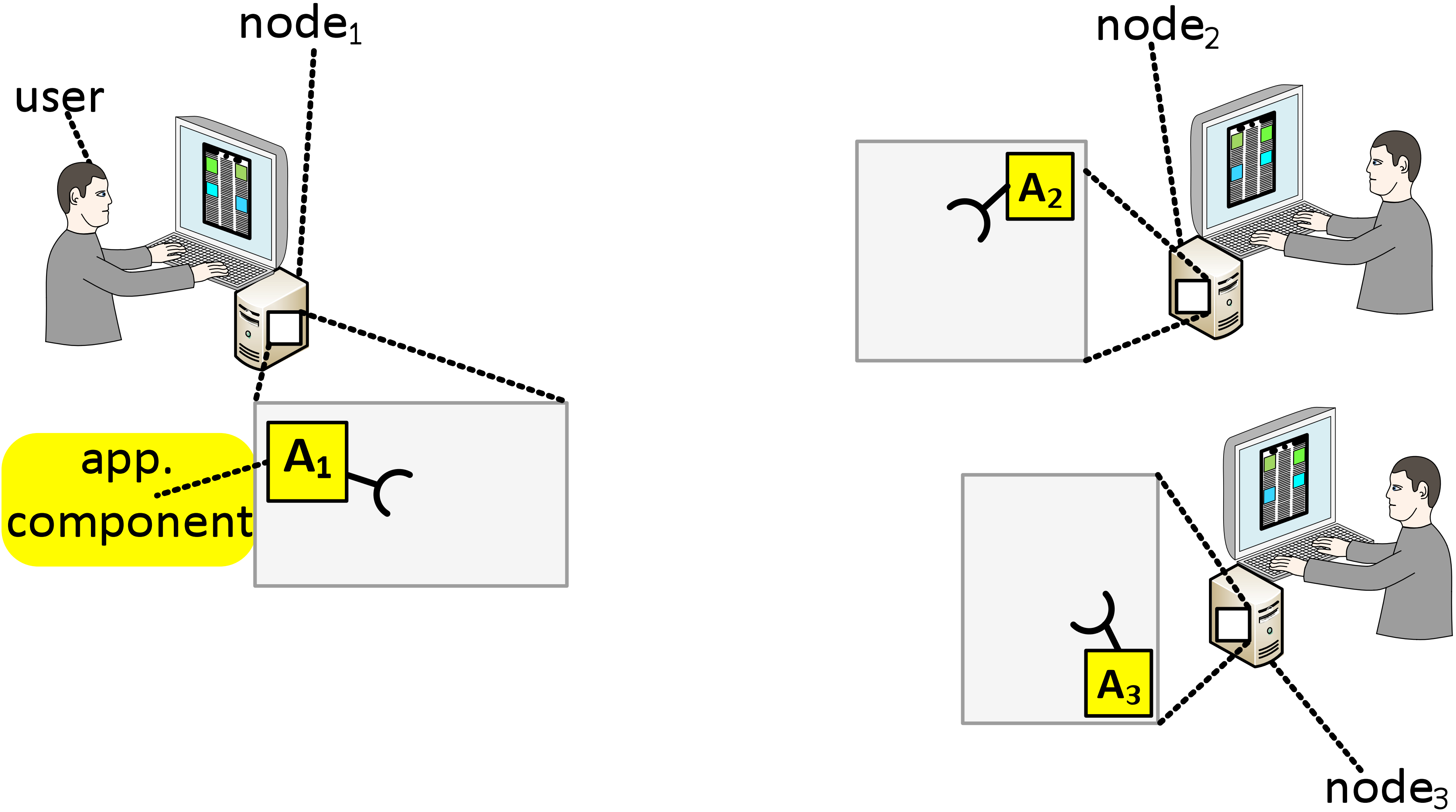
	A	B
1	Sales in Asia	
2		(Million dollars)
3	Total Amount	50

	A	B
1	Sales in Europe	
2		(Million dollars)
3	Total Amount	20

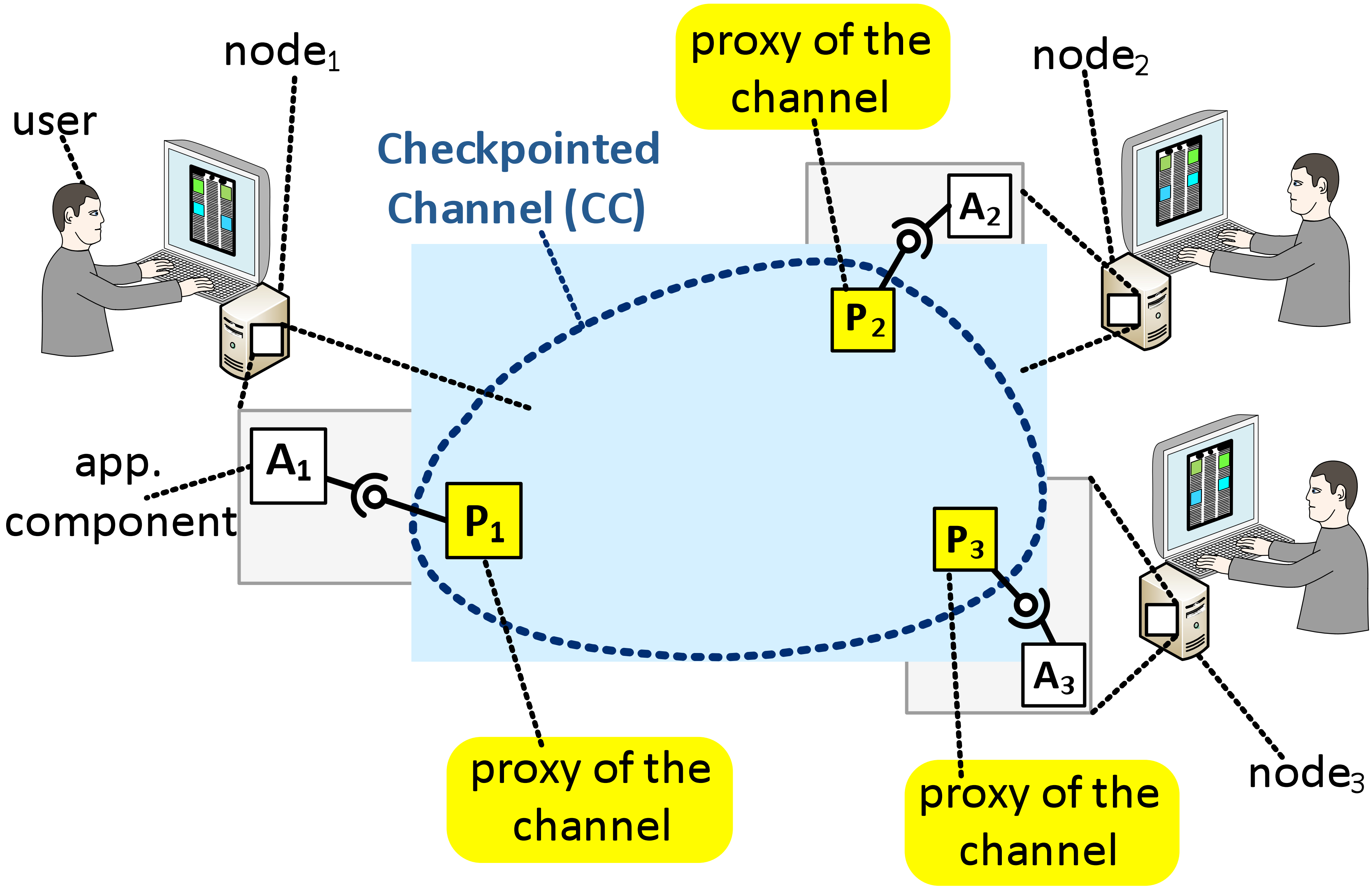
	A	B
1	Global Sales	
2		(Million dollars)
3	Sales in Asia	50
4	Sales in Europe	60
5	Sales in America	20
6	Total Amount	130

Checkpointed Channels (CC)

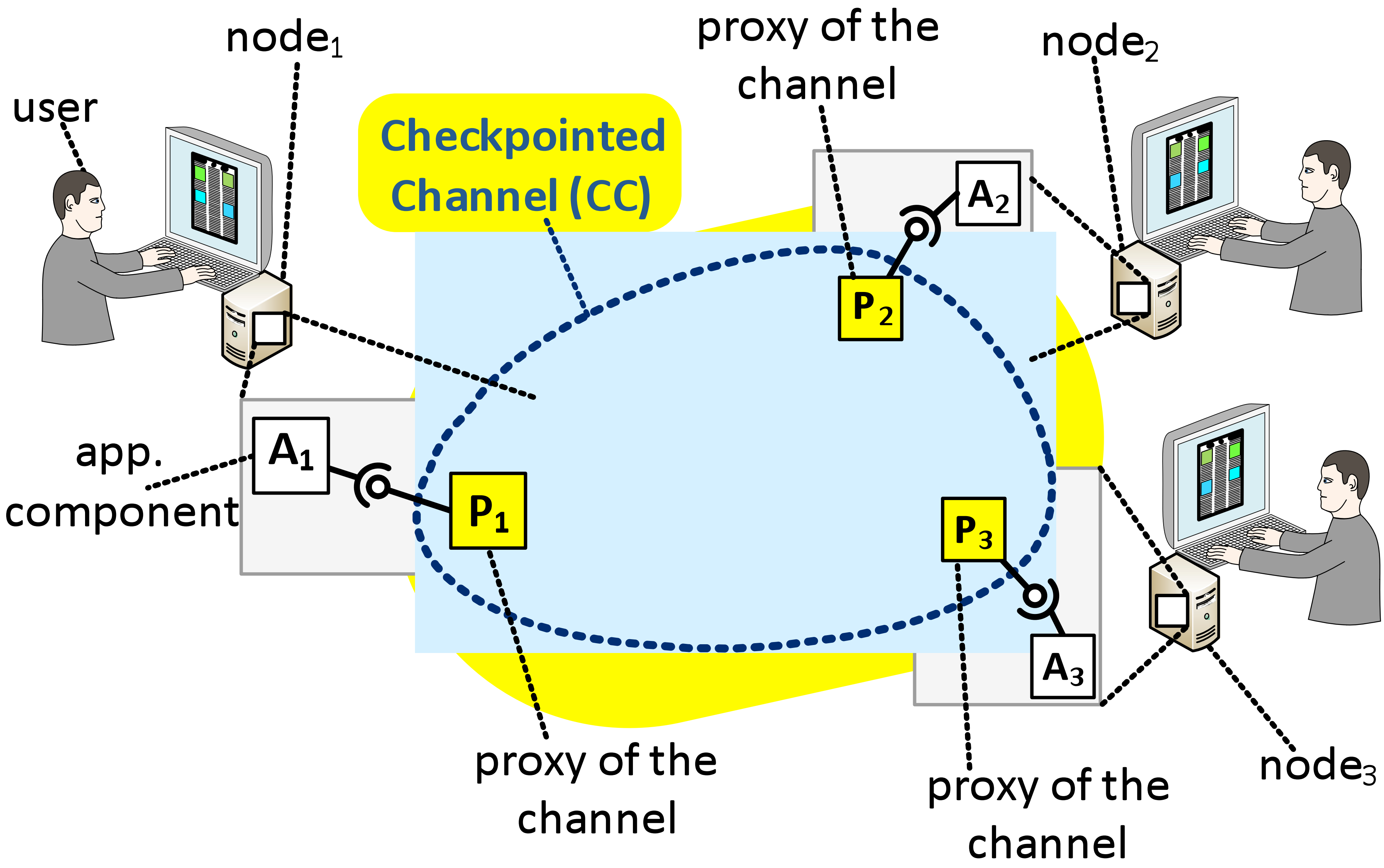
Architecture



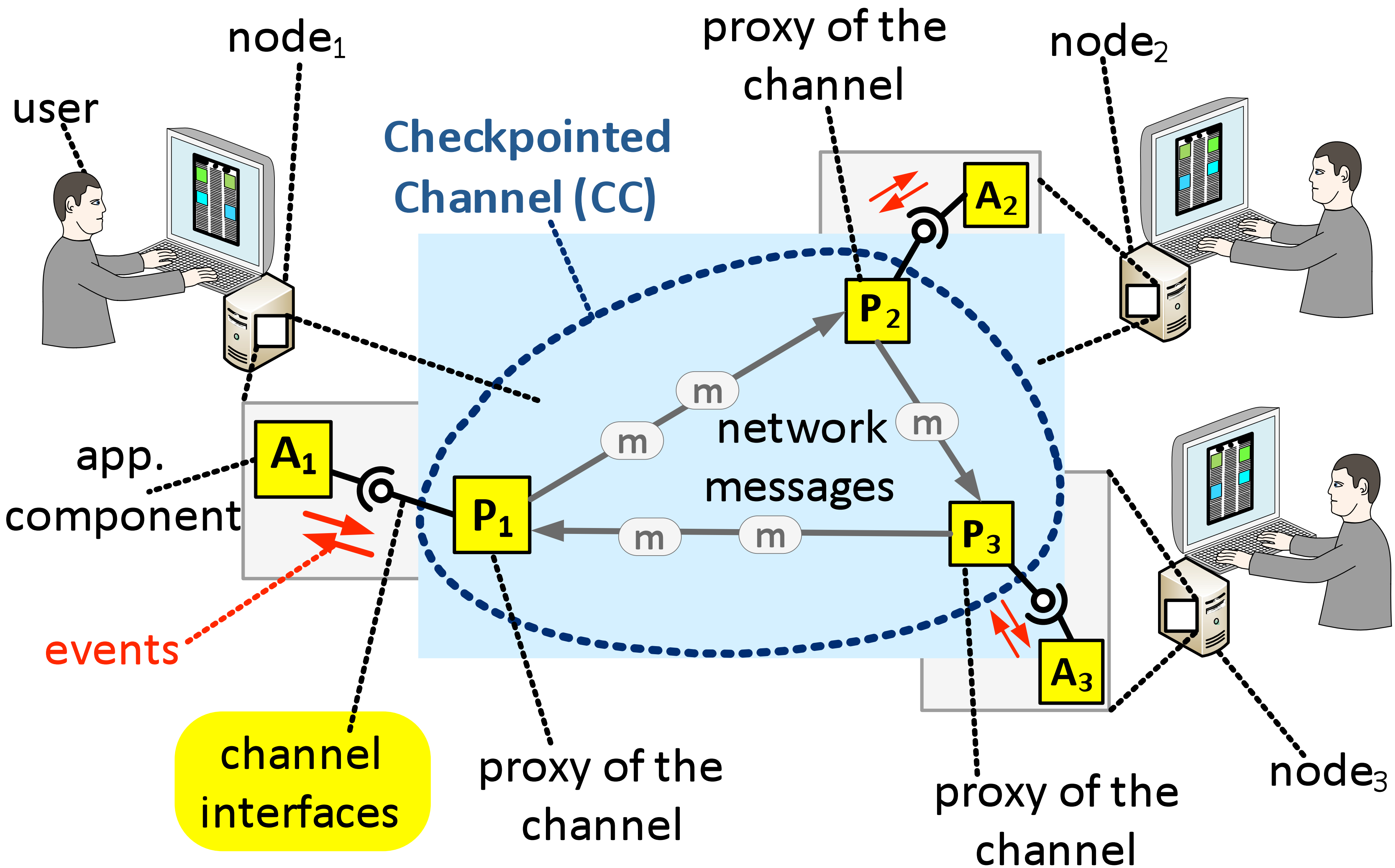
Architecture



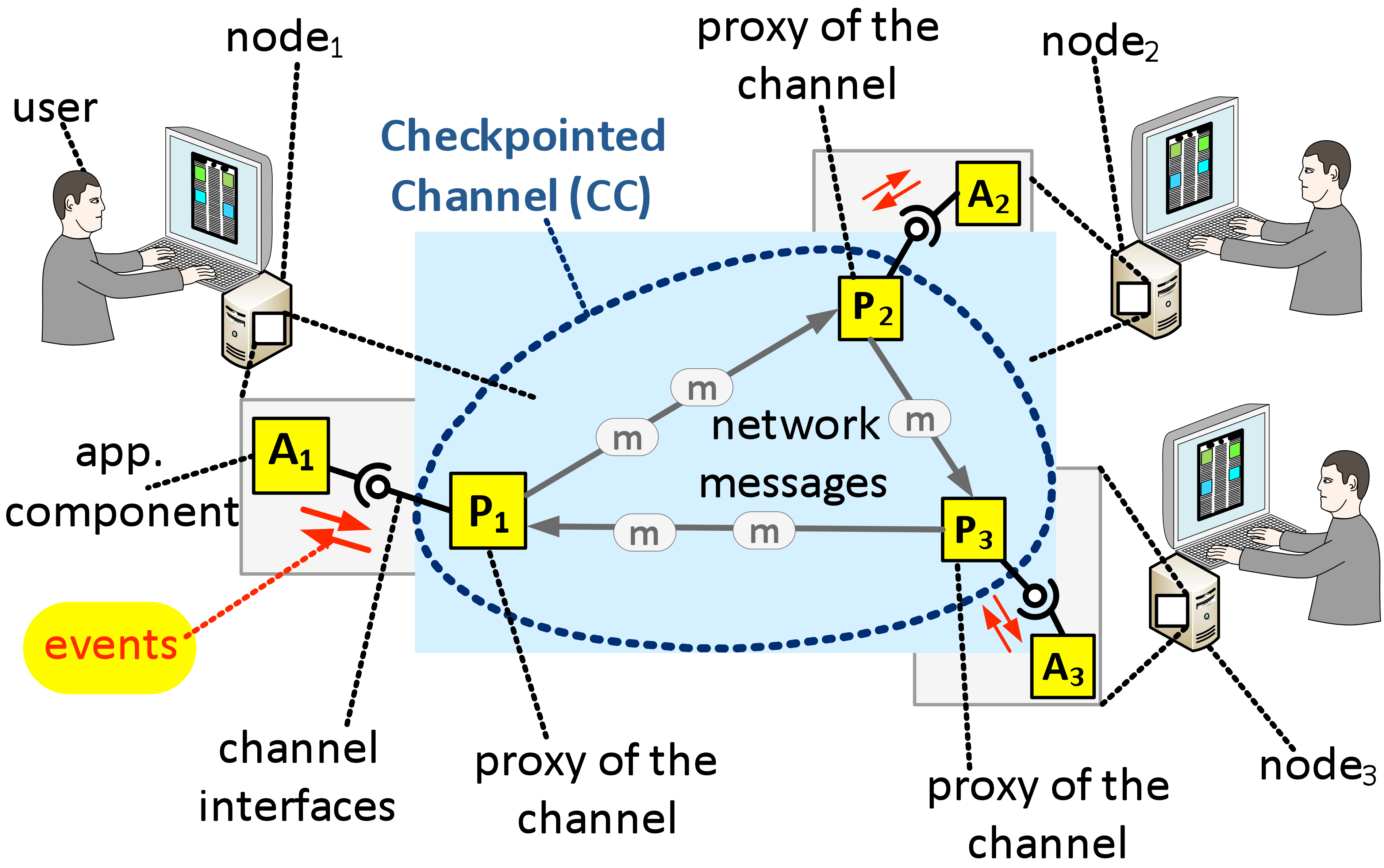
Architecture



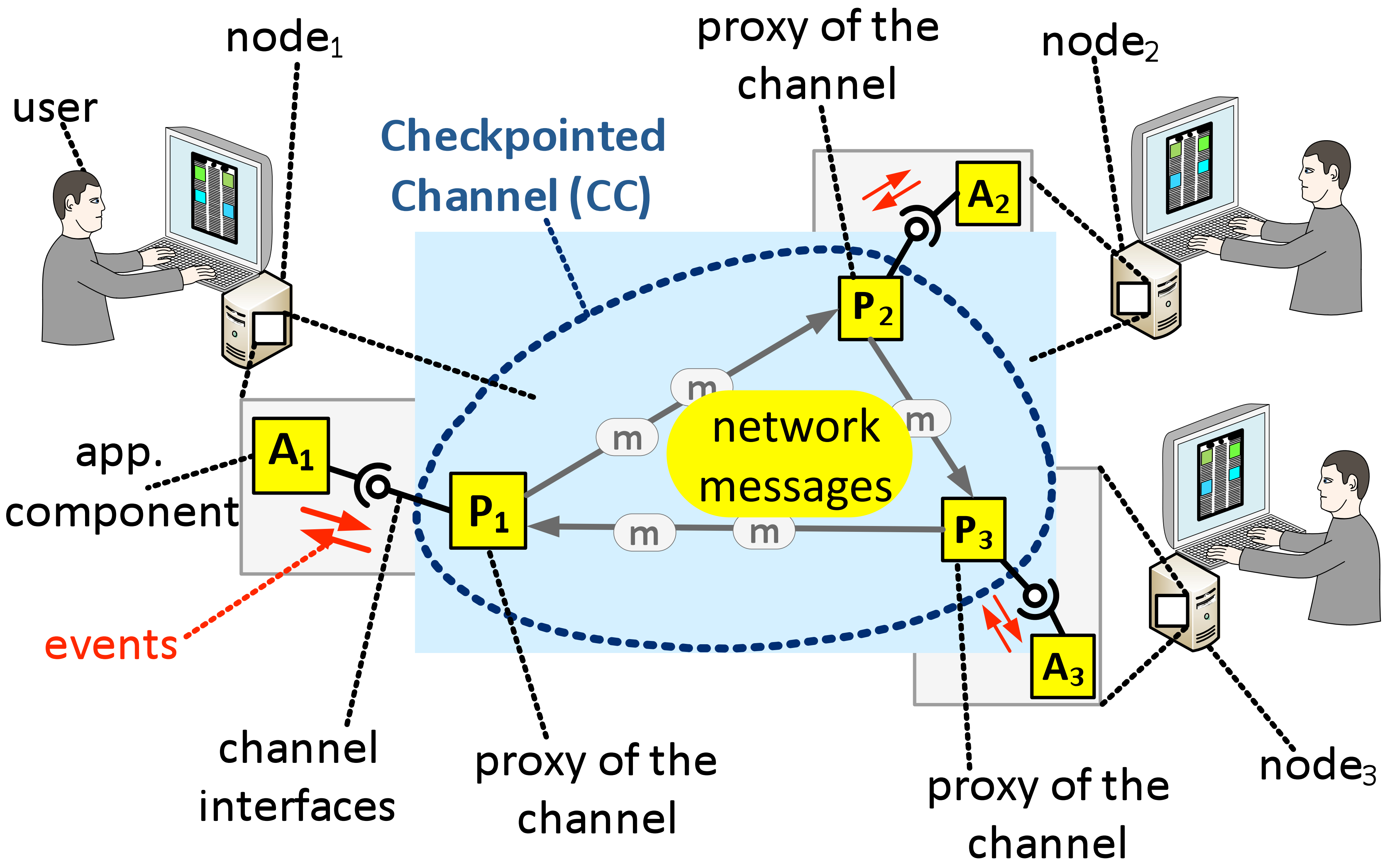
Architecture



Architecture

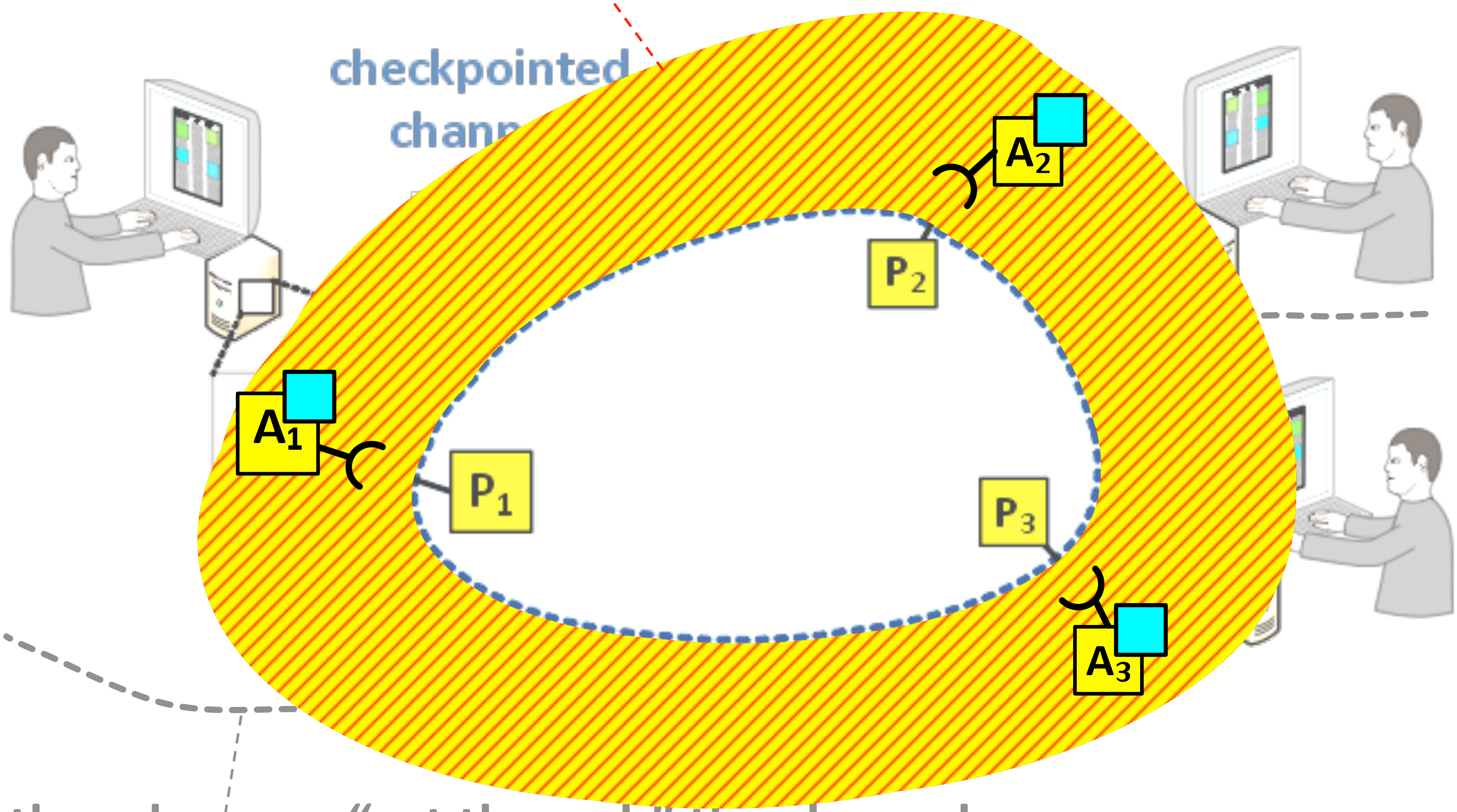


Architecture



content physically resides outside the channel…

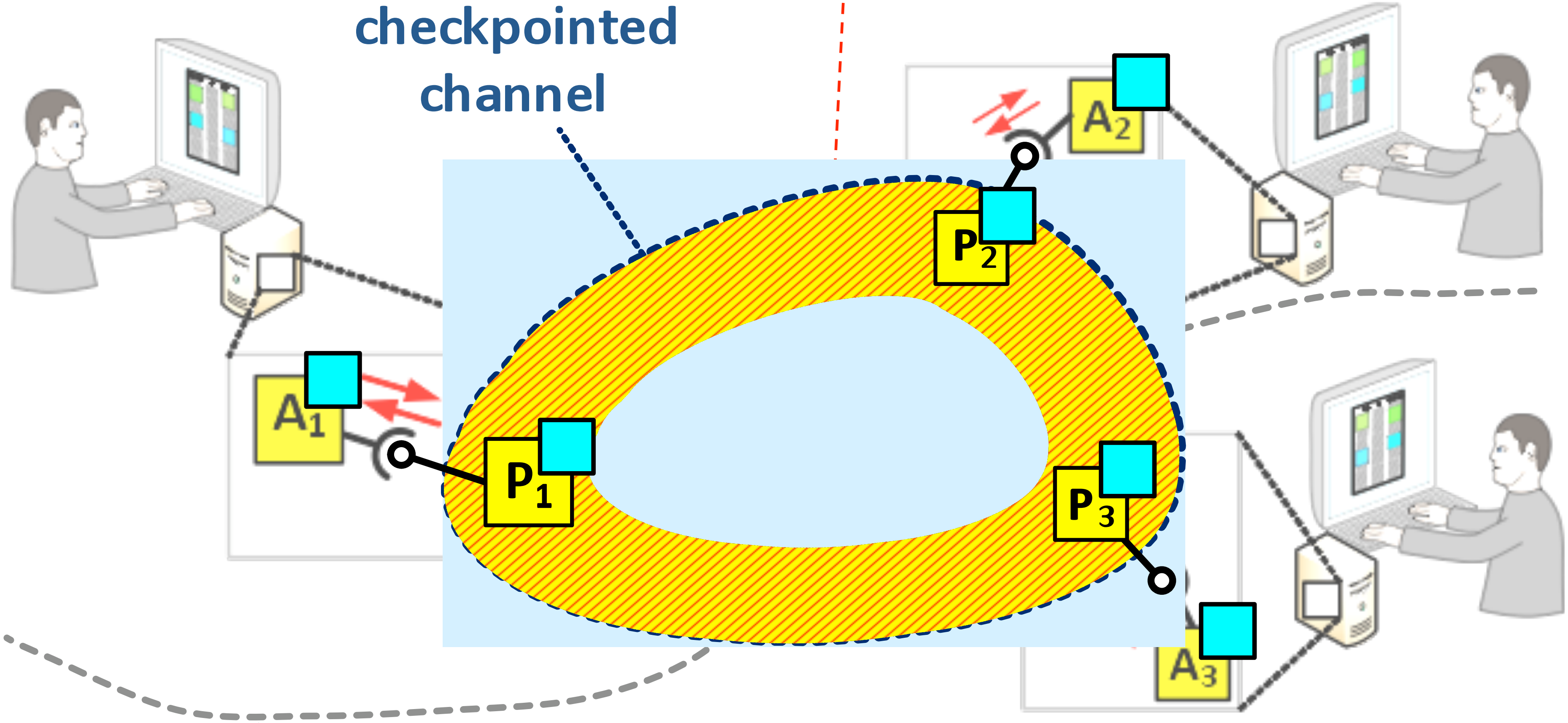
…but it could also be cached in it



the edge may “cut through” the channel

content physically resides outside the channel…

…but it could also be cached in it

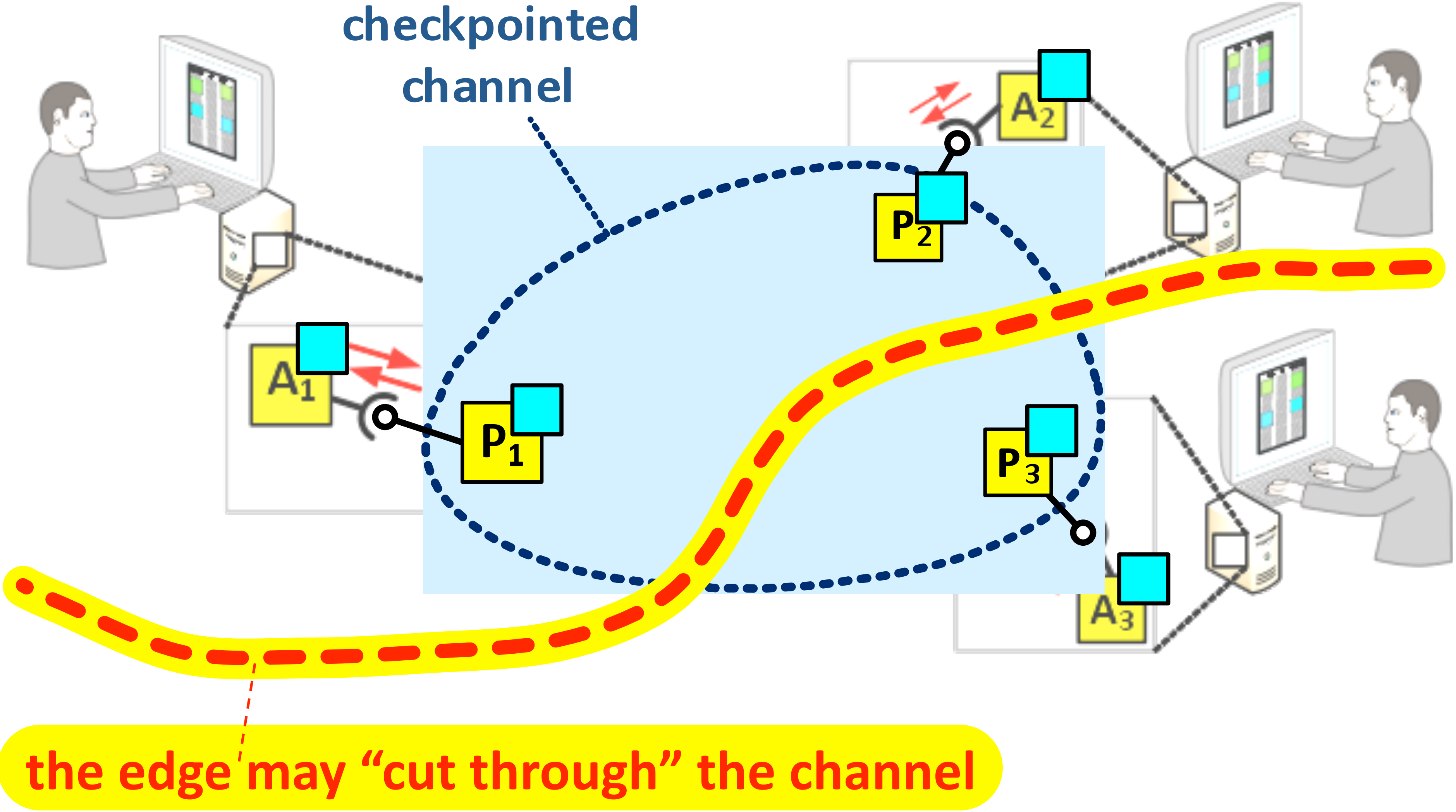


checkpointed channel

the edge may “cut through” the channel

content physically resides outside the channel…

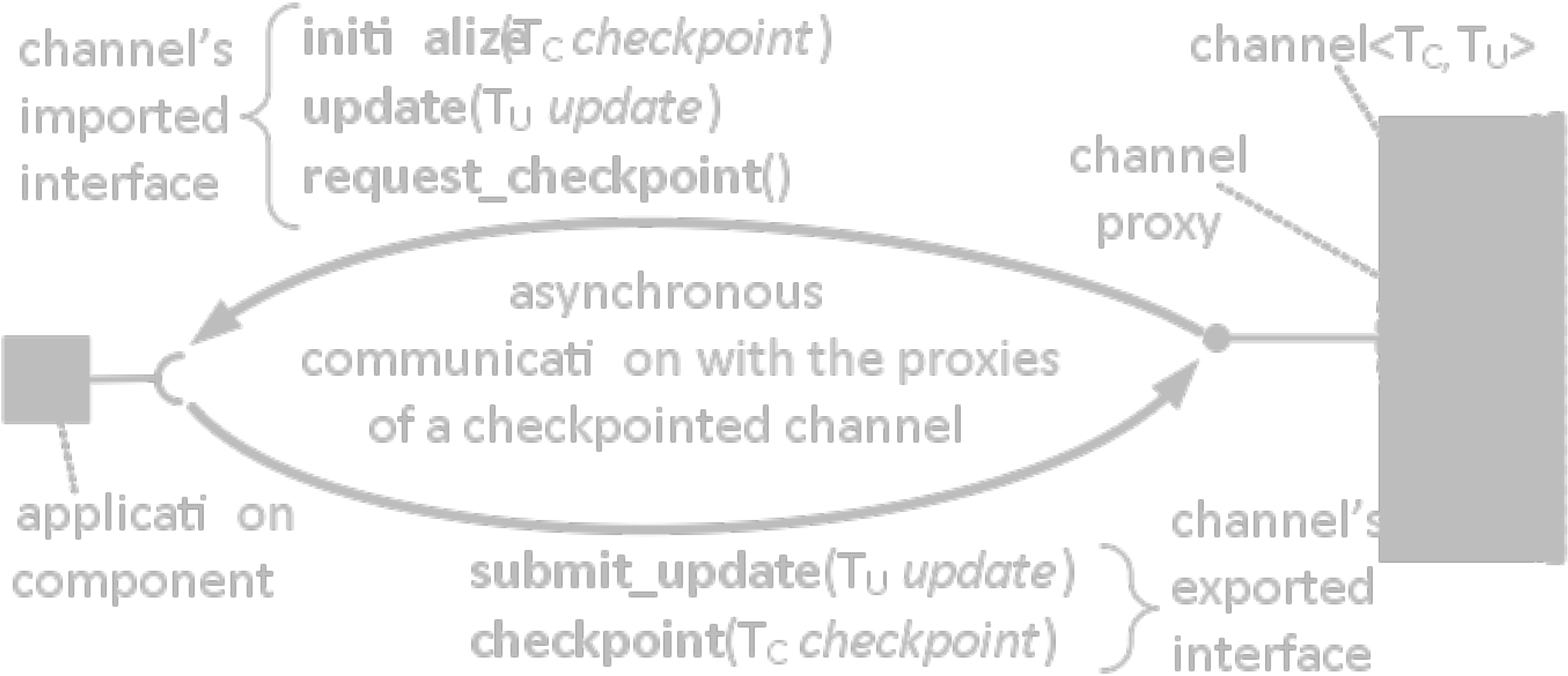
…but it could also be cached in it



Interfaces

Checkpointed Communication Channel (CC)

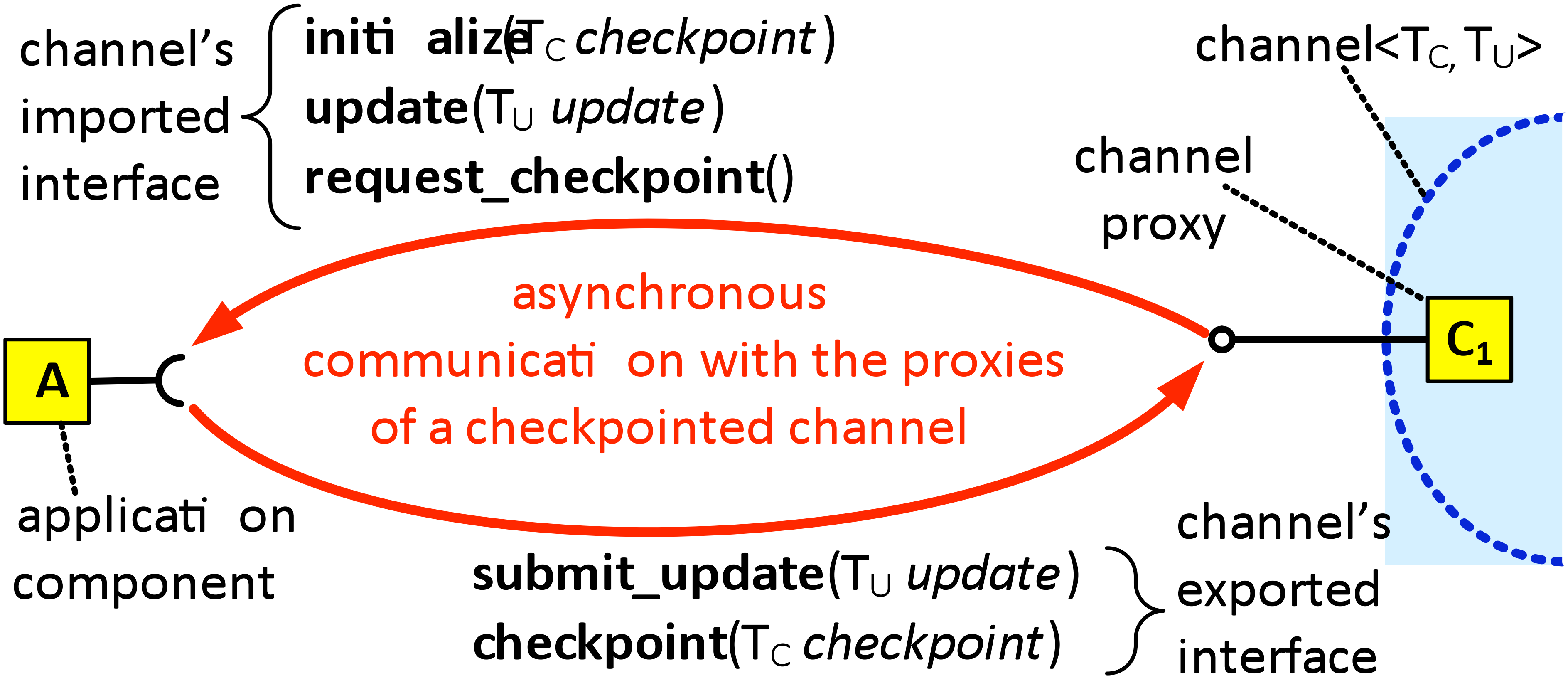
a writable stream of checkpoints and updates


channel's imported interface
initialize(T_C *checkpoint*)
update(T_U *update*)
request_checkpoint()
channel<T_C, T_U>
channel proxy
asynchronous communication with the proxies of a checkpointed channel
application component
submit_update(T_U *update*)
checkpoint(T_C *checkpoint*)
channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

- **initialize**(T_C *checkpoint*)
- **update**(T_U *update*)
- **request_checkpoint**()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

application component

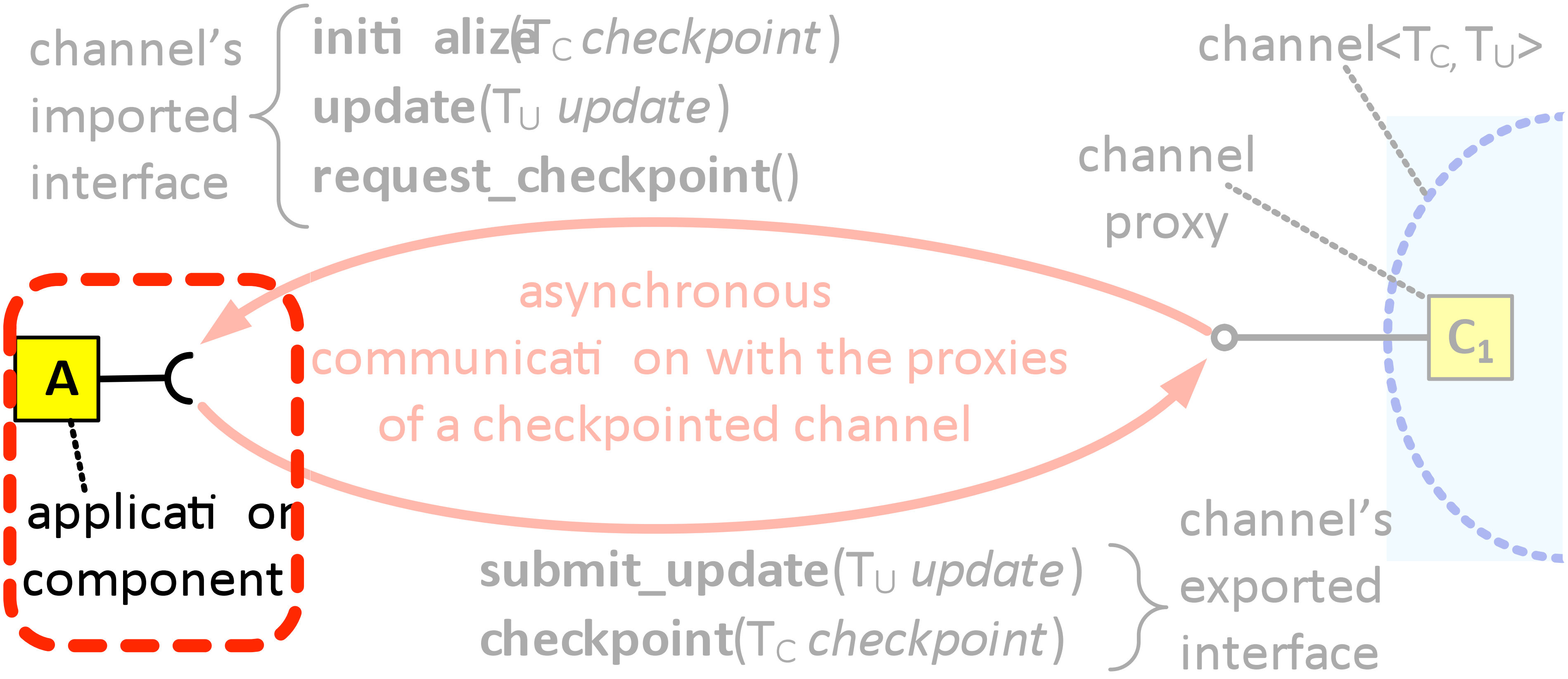
- **submit_update**(T_U *update*)
- **checkpoint**(T_C *checkpoint*)

channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

application component

submit_update(T_U *update*)

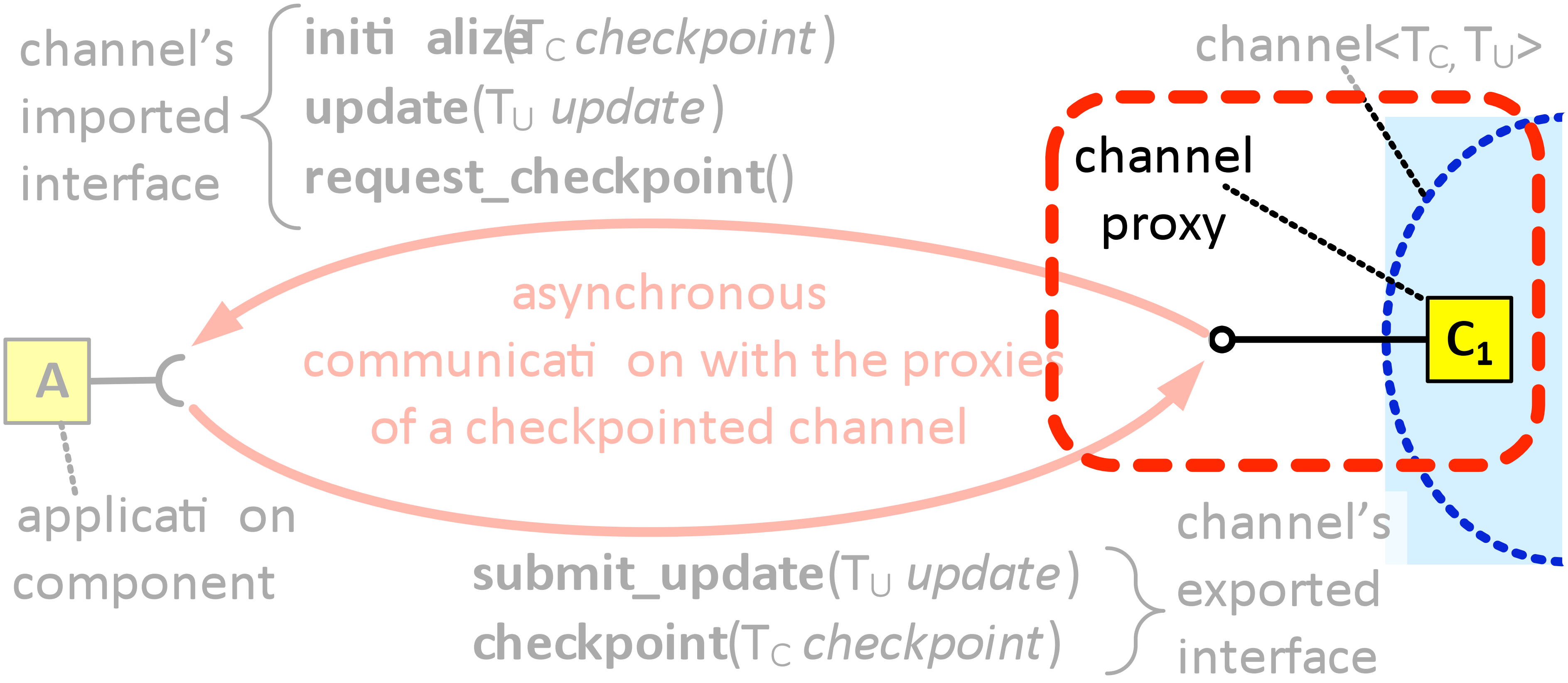
checkpoint(T_C *checkpoint*)

channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

C_1

A

asynchronous communication with the proxies of a checkpointed channel

application component

submit_update(T_U *update*)

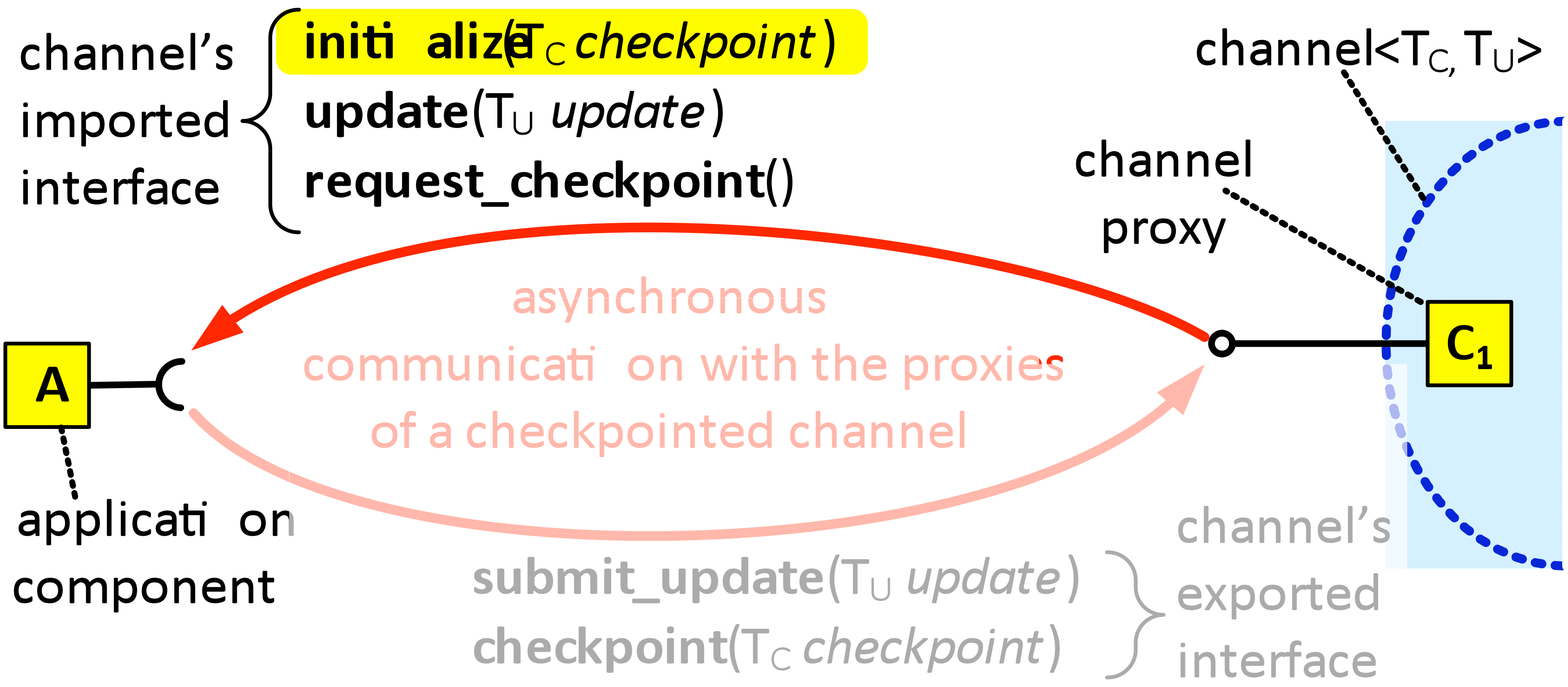
checkpoint(T_C *checkpoint*)

channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

application component

submit_update(T_U *update*)

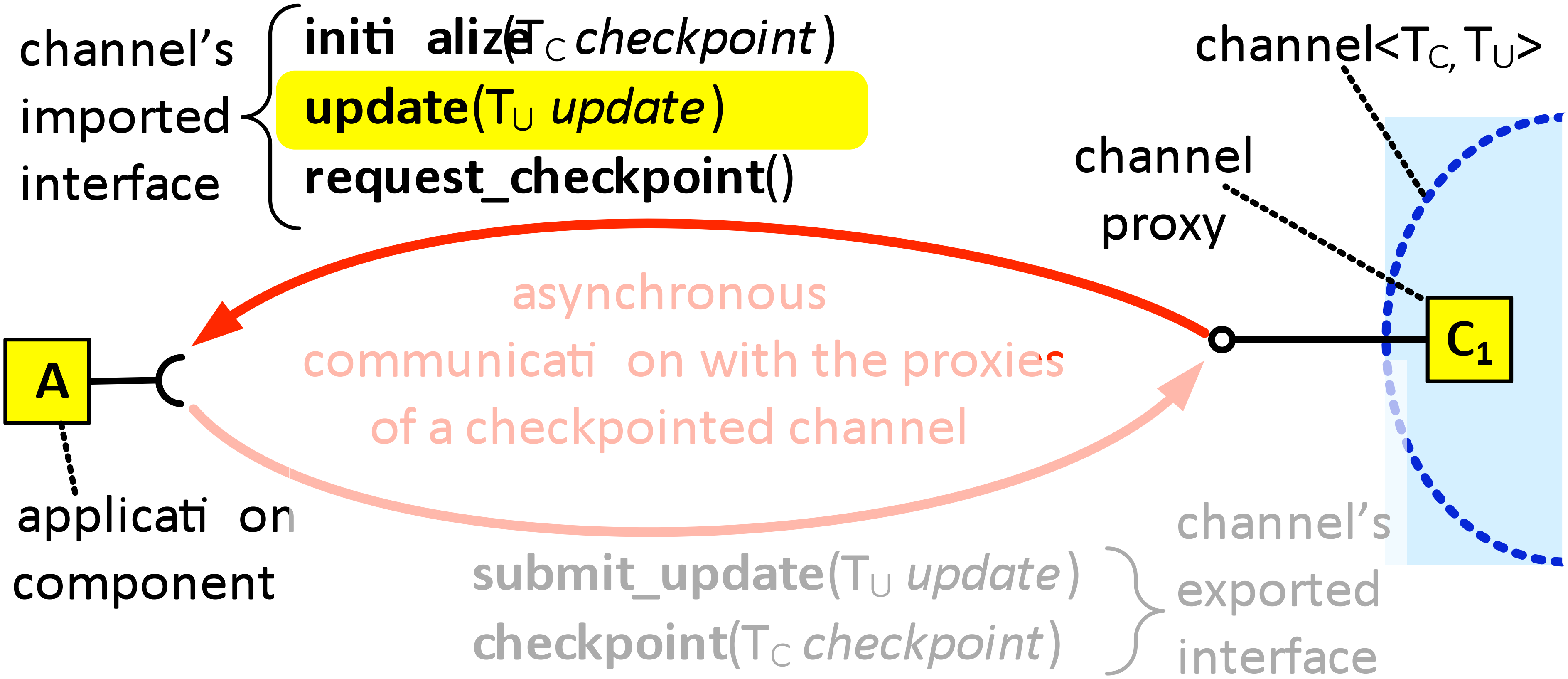
checkpoint(T_C *checkpoint*)

channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

C_1

A

application component

asynchronous communication with the proxies of a checkpointed channel

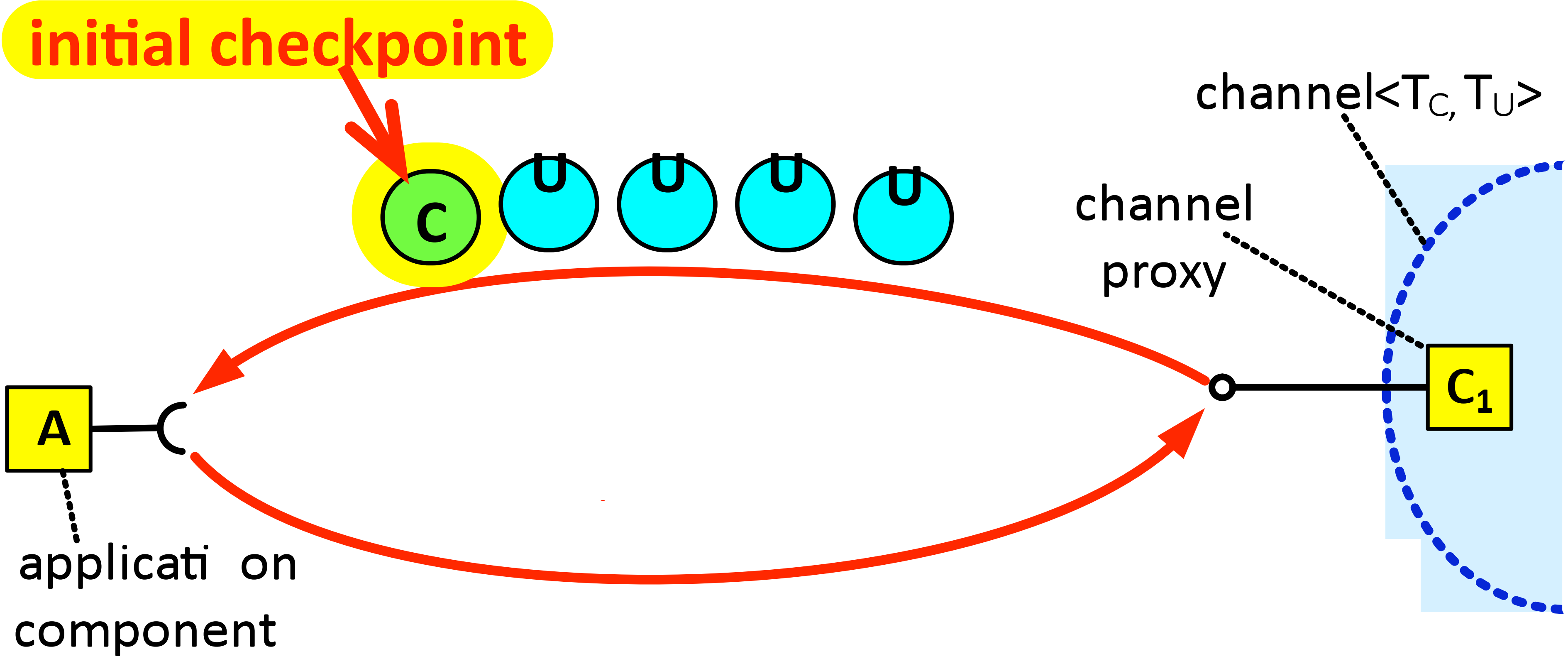
submit_update(T_U *update*)

checkpoint(T_C *checkpoint*)

channel's exported interface

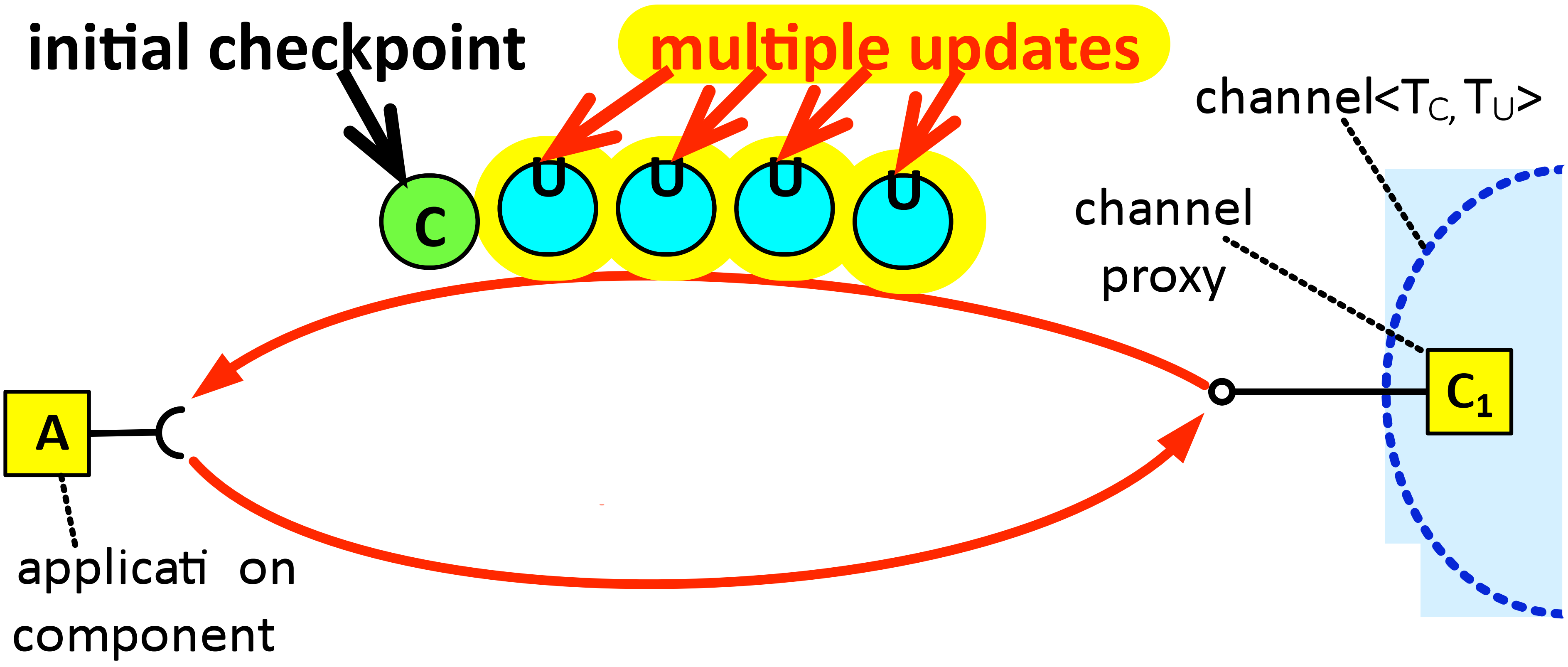
Dynamics (local)

Checkpointed Communication Channel (CC)



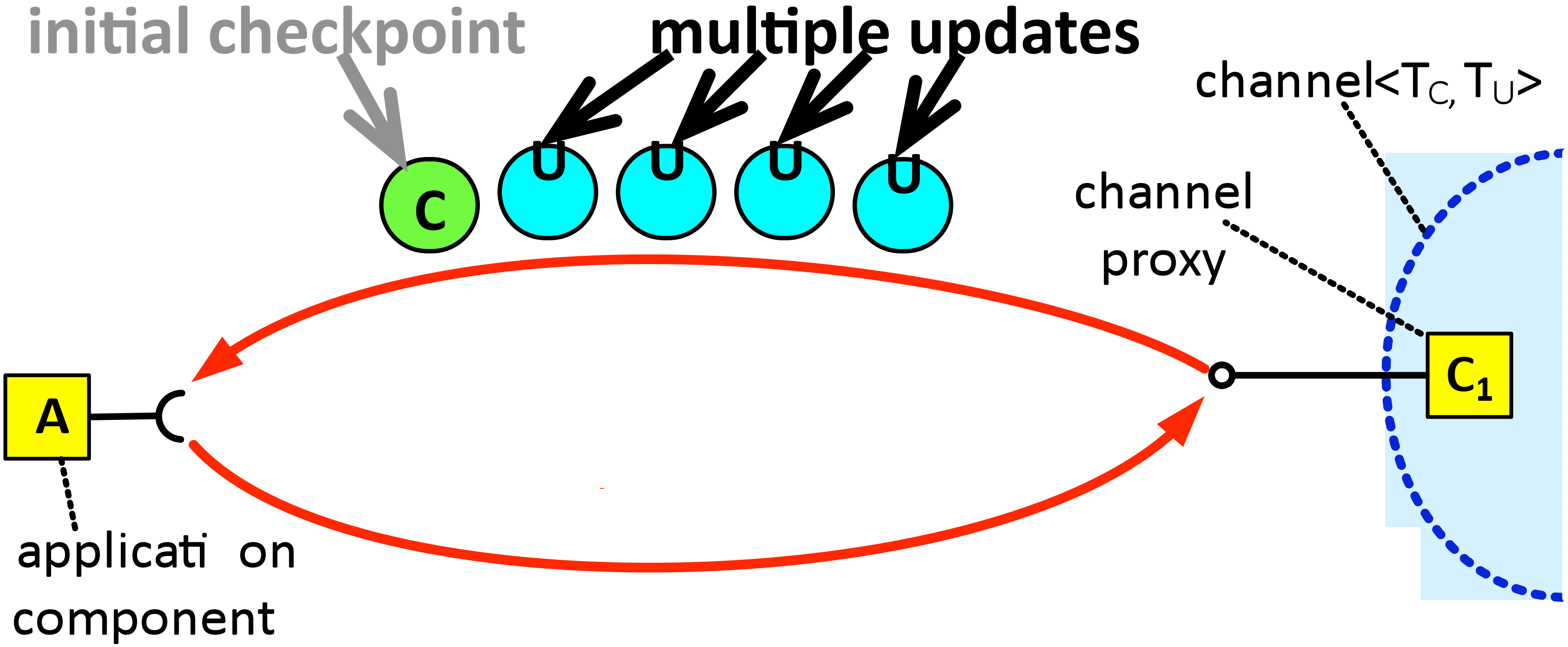
Dynamics (local)

Checkpointed Communication Channel (CC)



Dynamics (local)

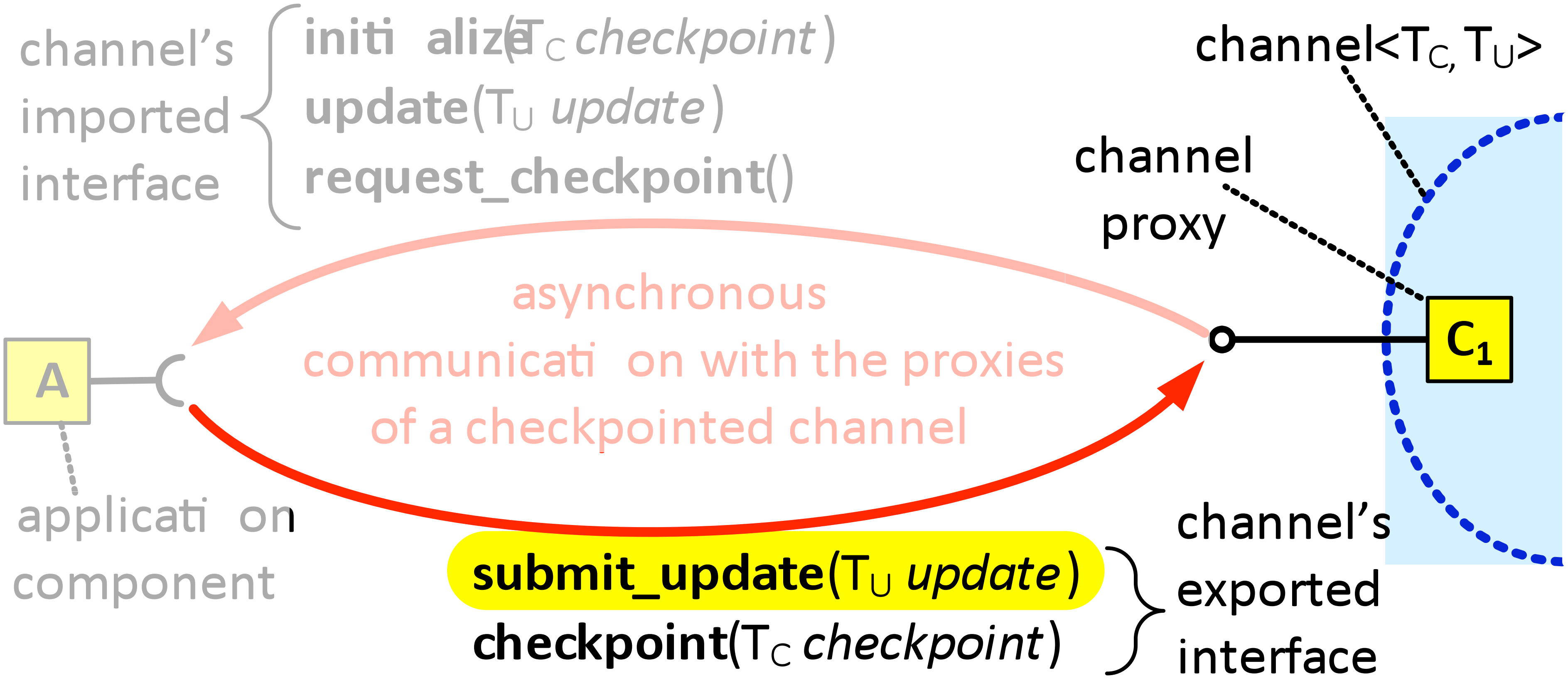
Checkpointed Communication Channel (CC)



Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

A

C_1

application component

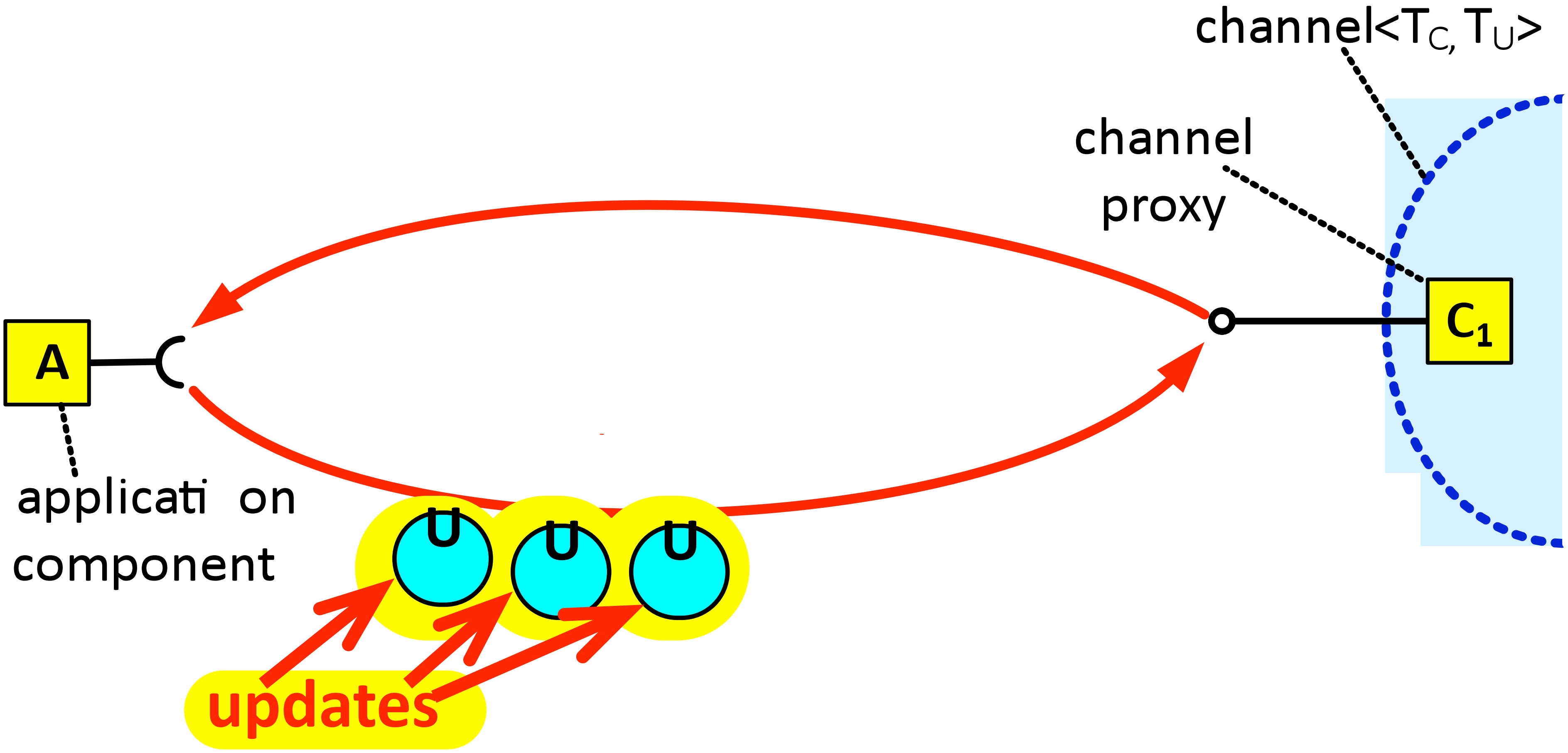
submit_update(T_U *update*)

checkpoint(T_C *checkpoint*)

channel's exported interface

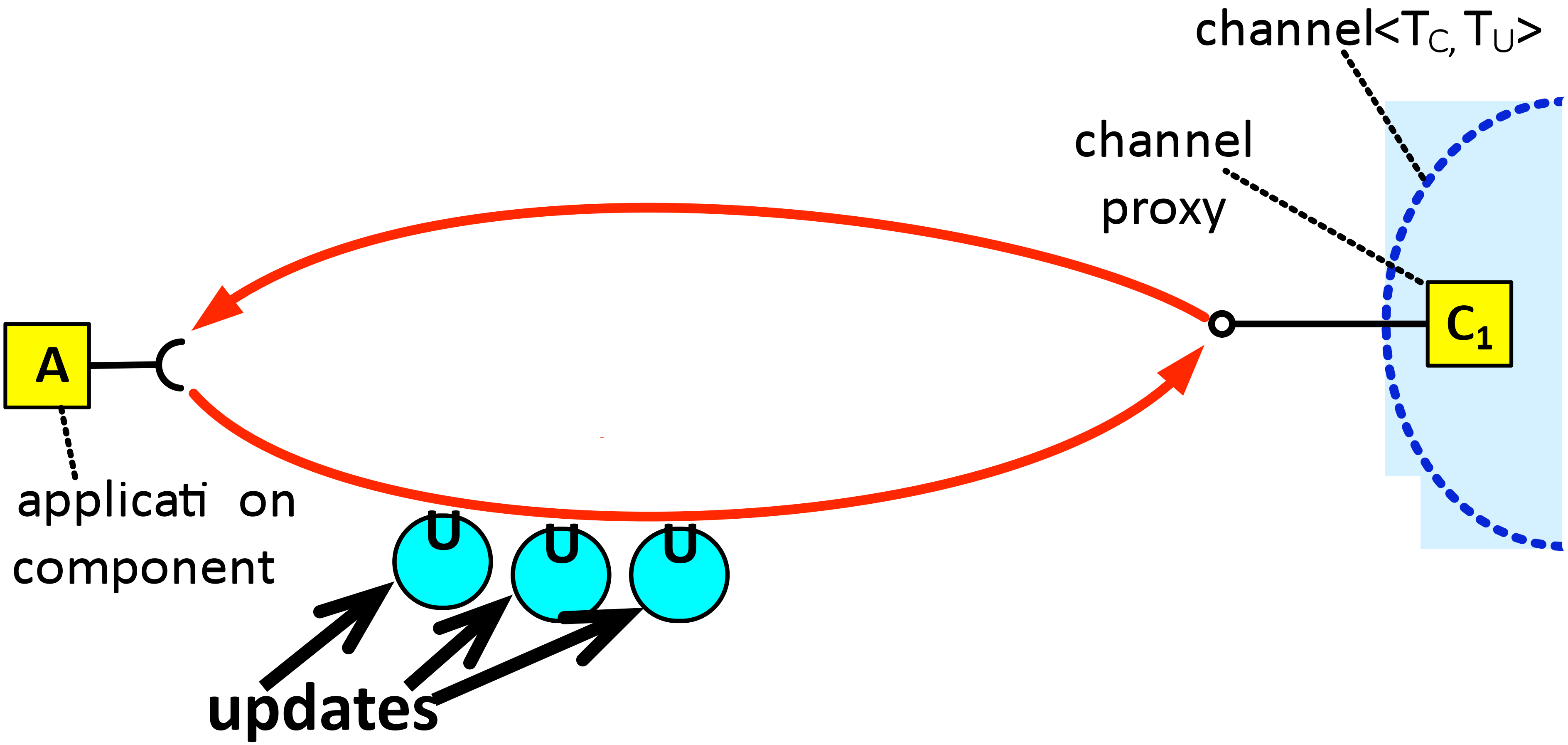
Dynamics (local)

Checkpointed Communication Channel (CC)



Dynamics (local)

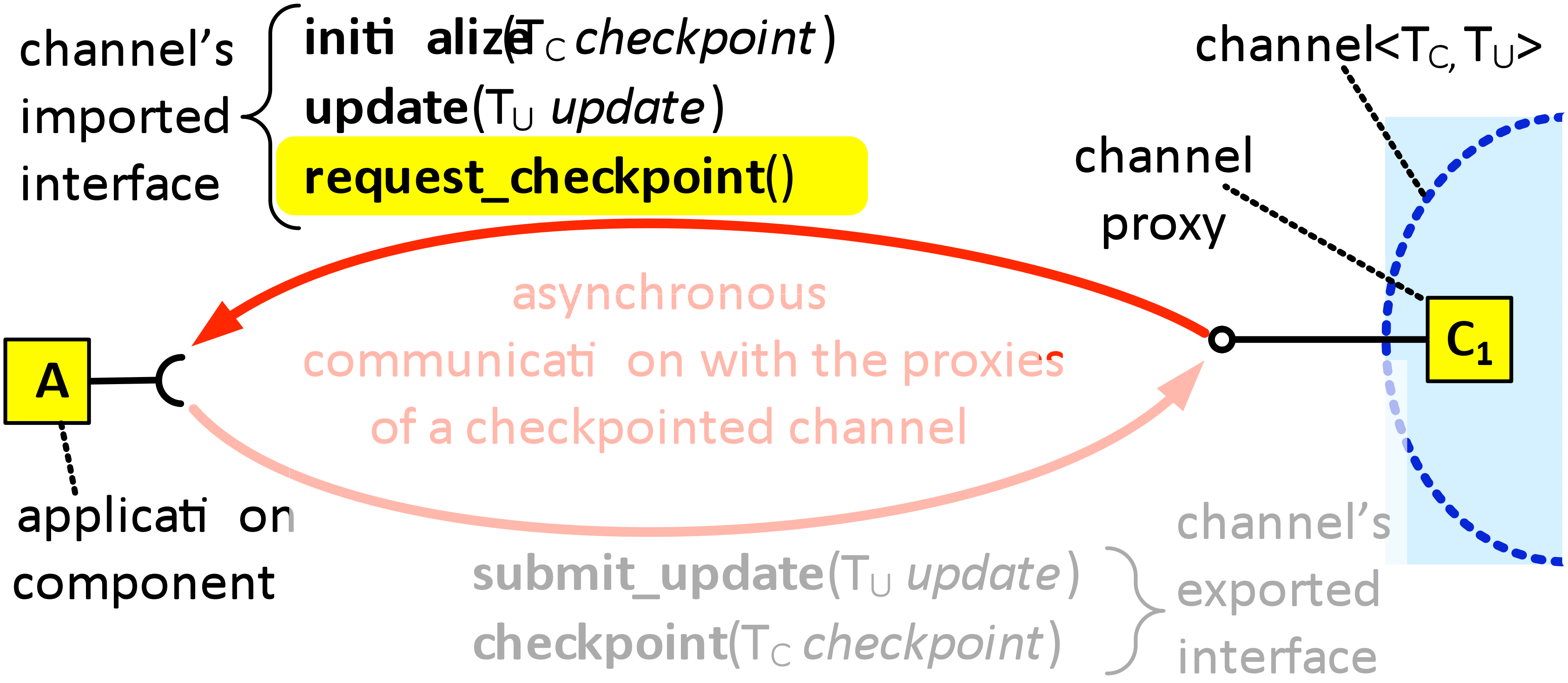
Checkpointed Communication Channel (CC)



Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

application component

submit_update(T_U *update*)

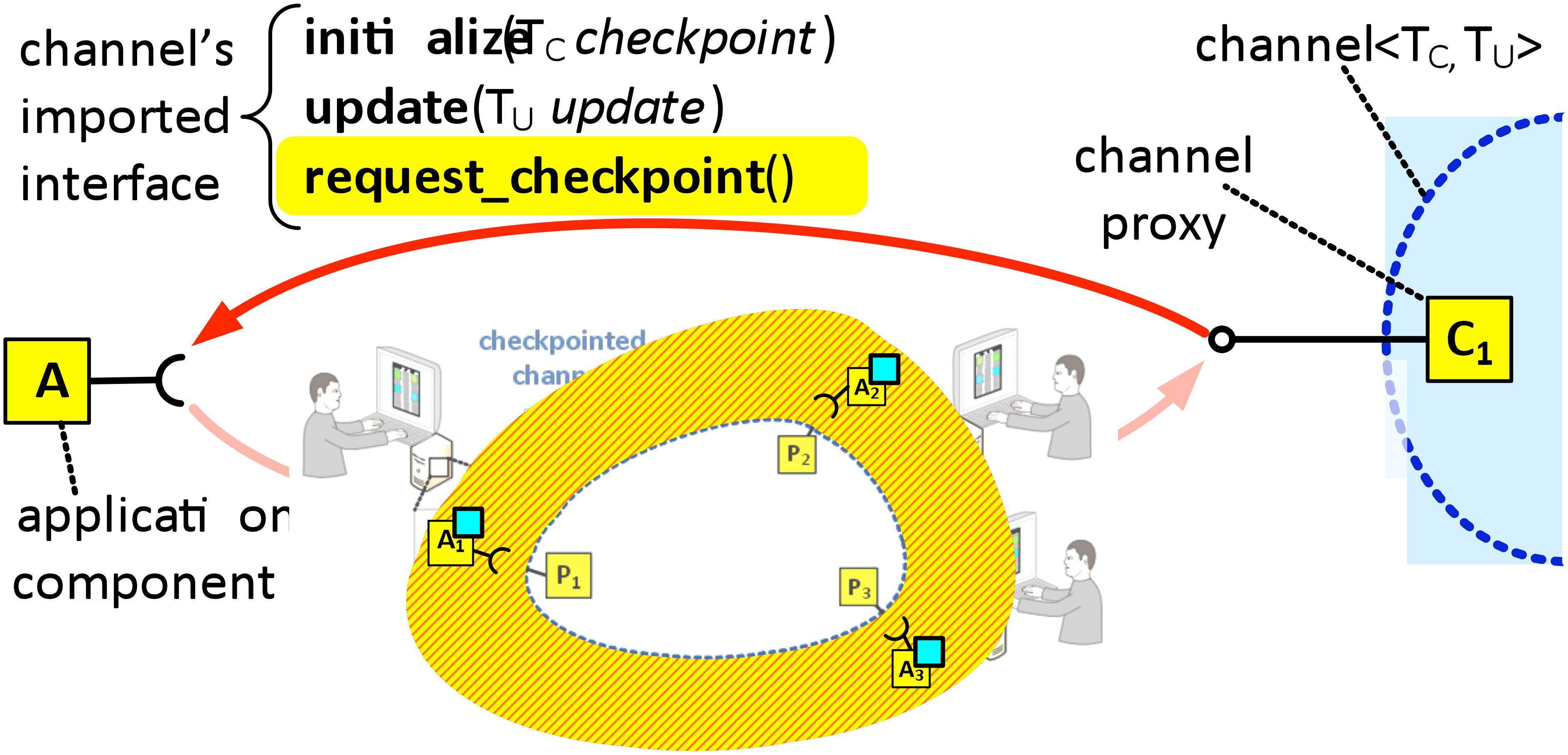
checkpoint(T_C *checkpoint*)

channel's exported interface

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

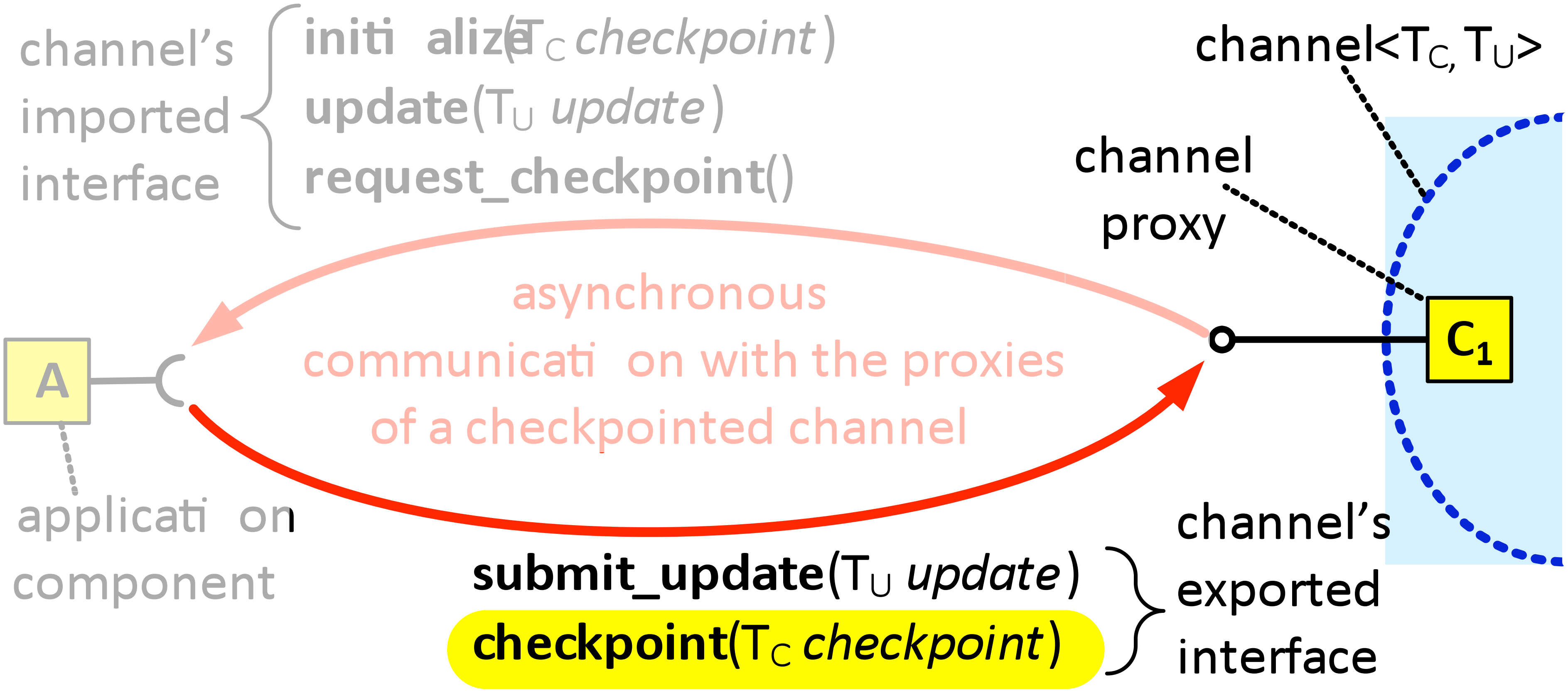
channel proxy

application component

Interfaces

Checkpointed Communication Channel (CC)

a writable stream of checkpoints and updates



channel's imported interface

initialize(T_C *checkpoint*)

update(T_U *update*)

request_checkpoint()

channel<T_C, T_U>

channel proxy

asynchronous communication with the proxies of a checkpointed channel

application component

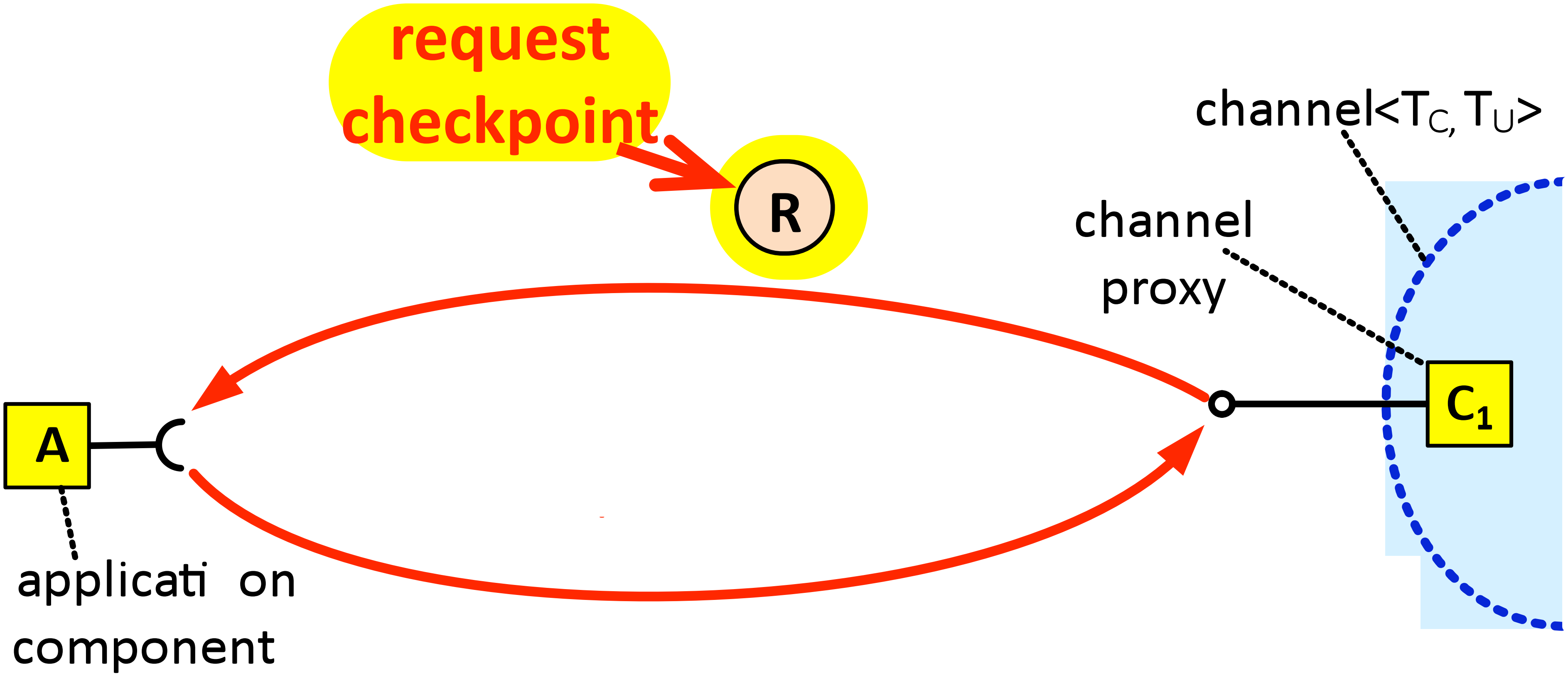
submit_update(T_U *update*)

checkpoint(T_C *checkpoint*)

channel's exported interface

Dynamics (local)

Checkpointed Communication Channel (CC)



request checkpoint

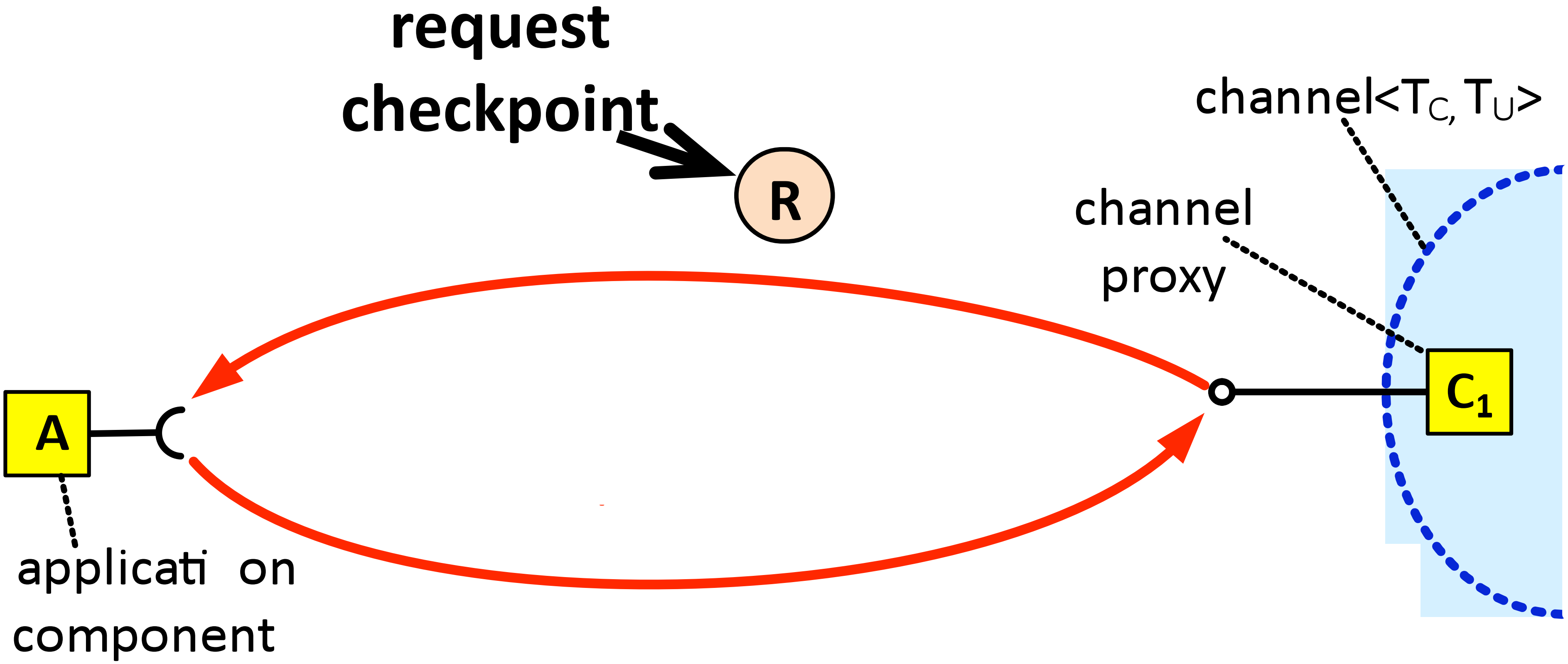
channel<T_C, T_U>

channel proxy

application component

Dynamics (local)

Checkpointed Communication Channel (CC)



request checkpoint

R

channel<T_C, T_U>

channel proxy

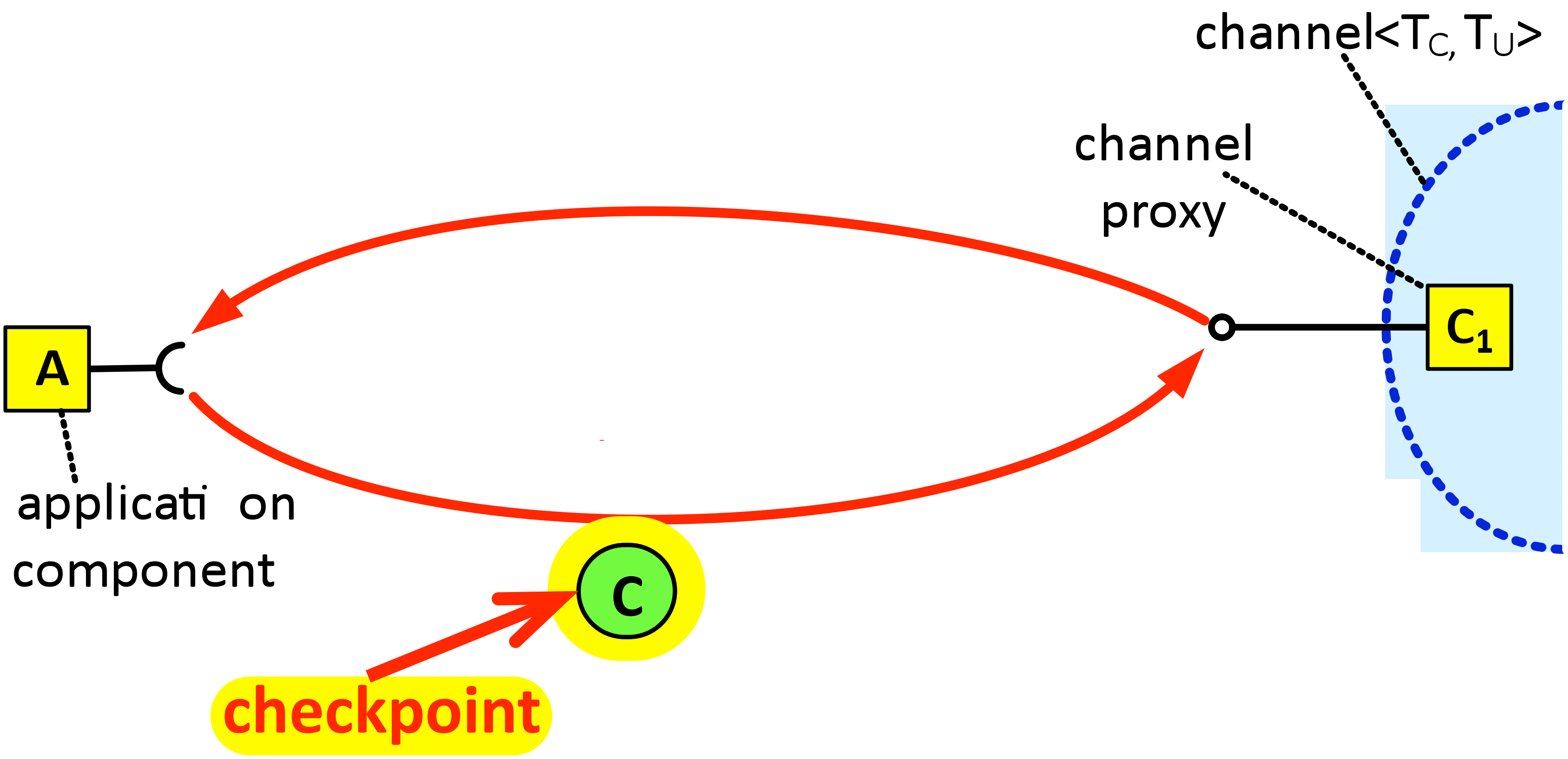
A

C_1

application component

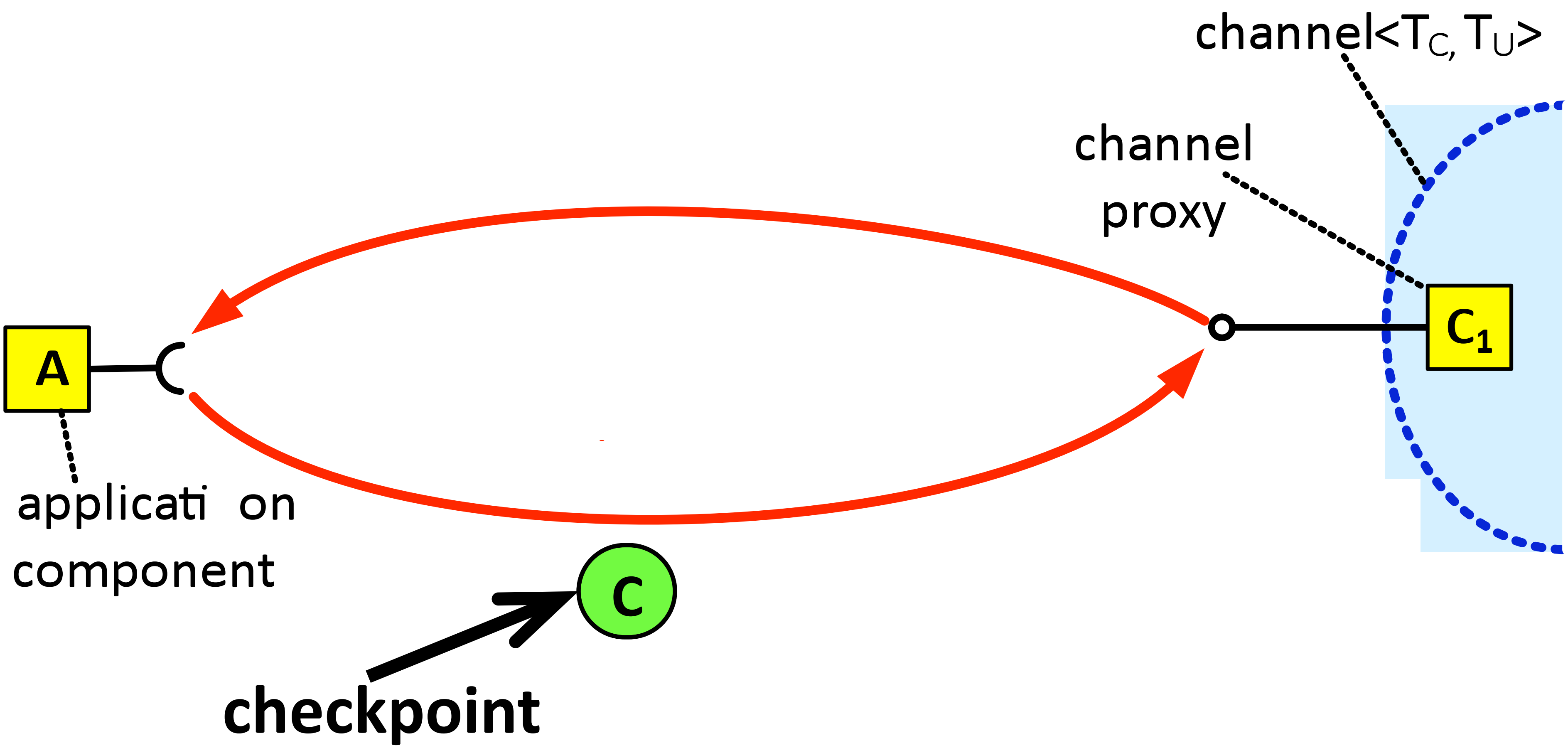
Dynamics (local)

Checkpointed Communication Channel (CC)

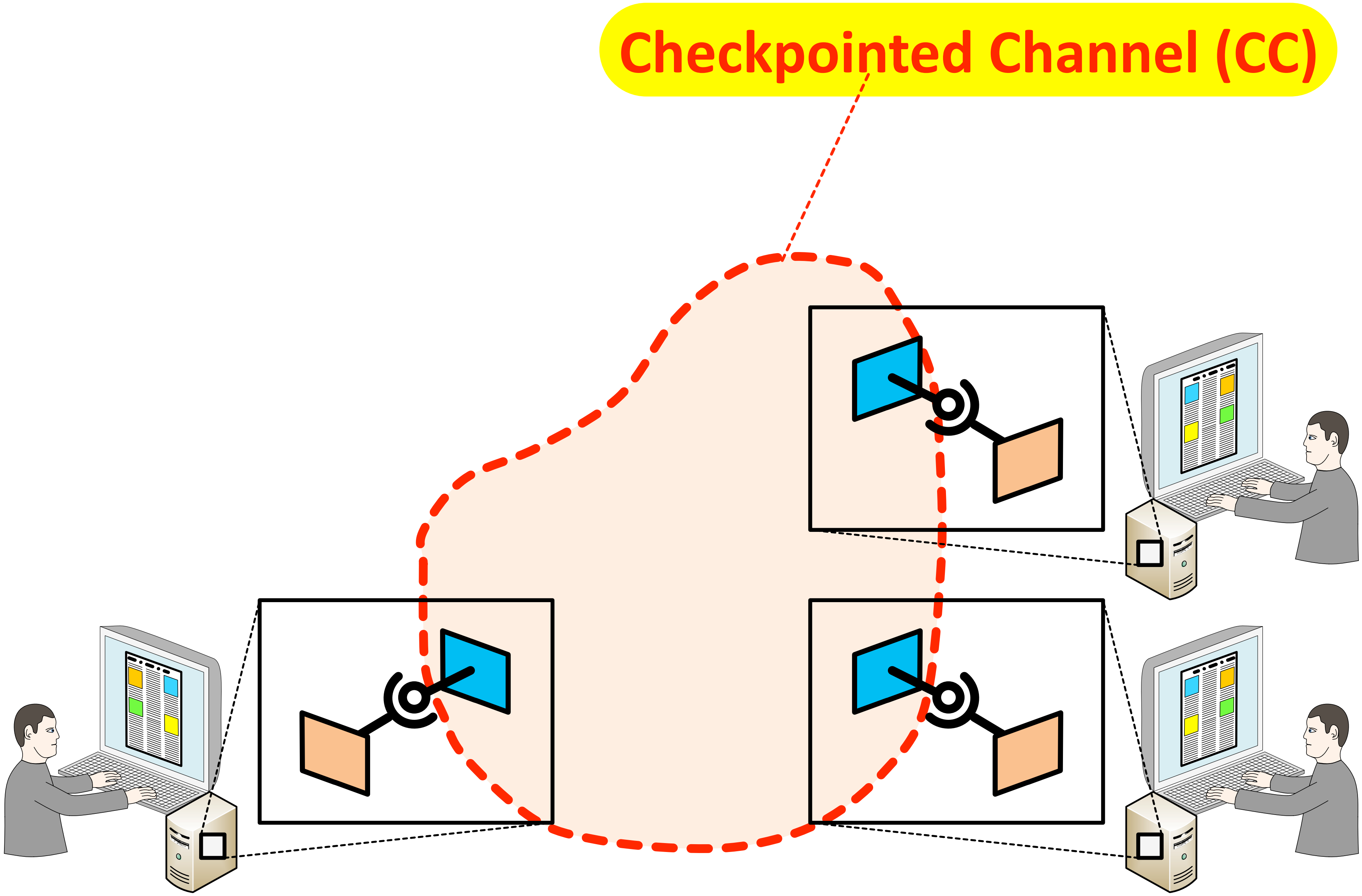


Dynamics (local)

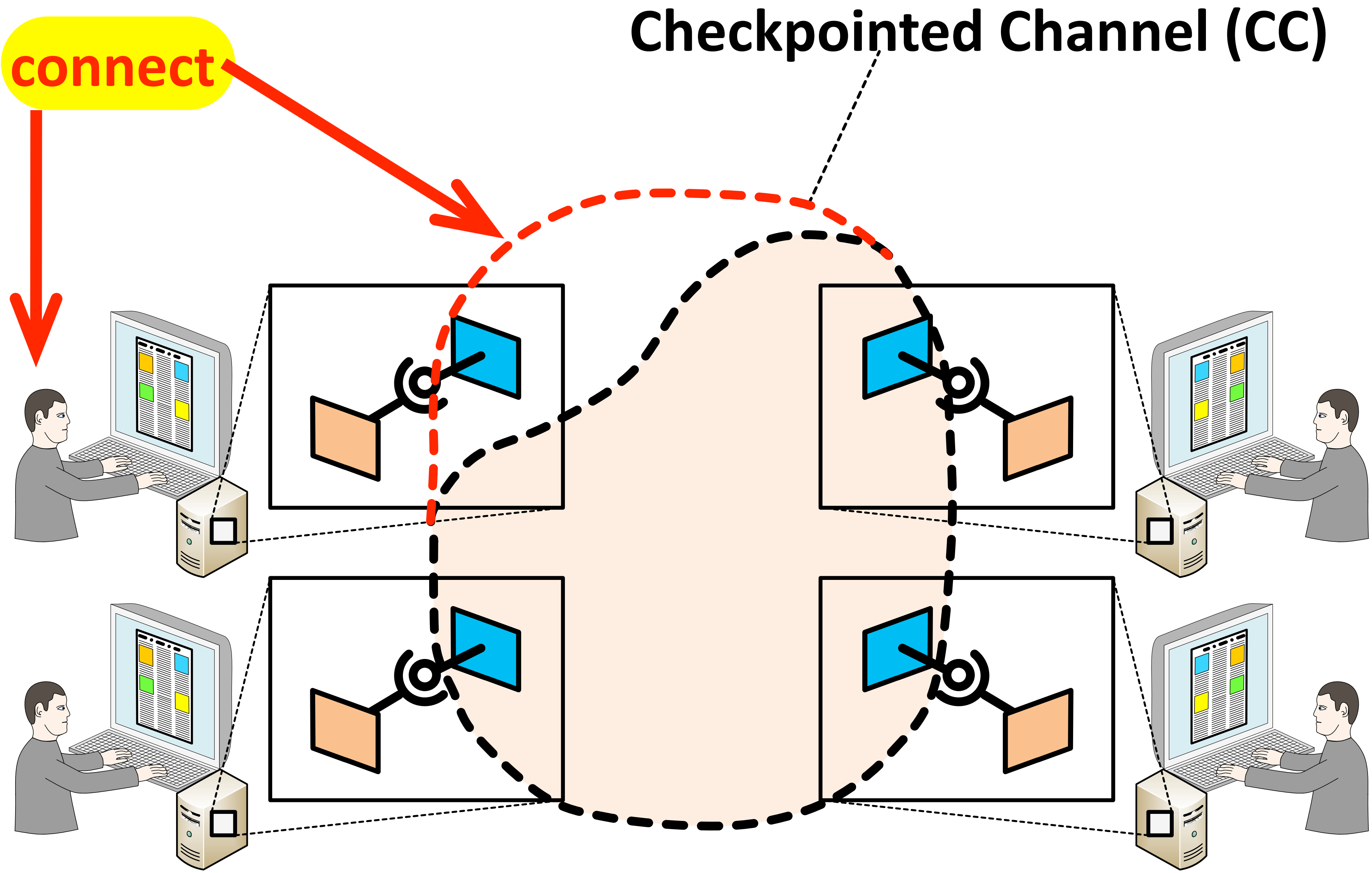
Checkpointed Communication Channel (CC)



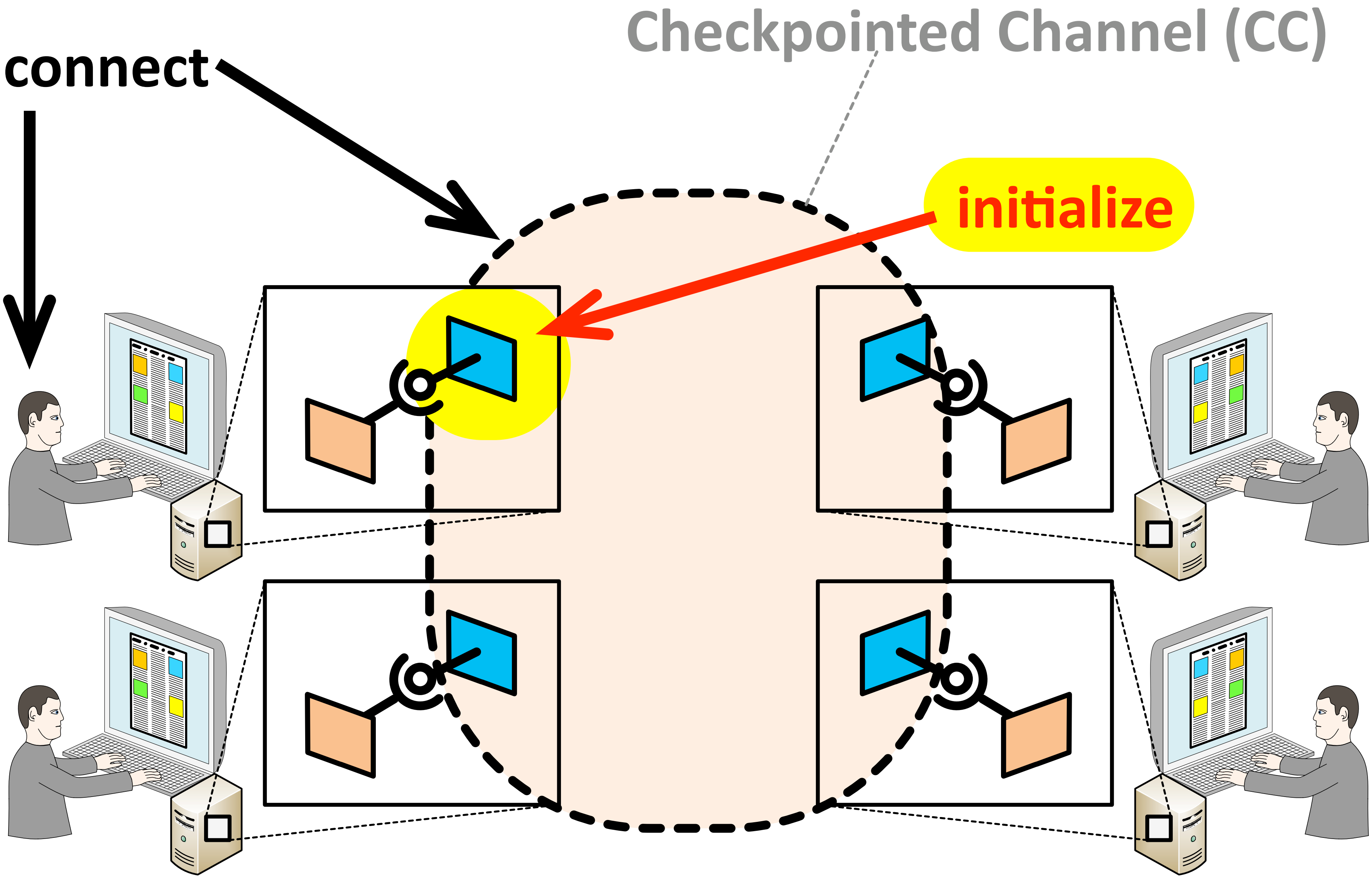
Dynamics (global)



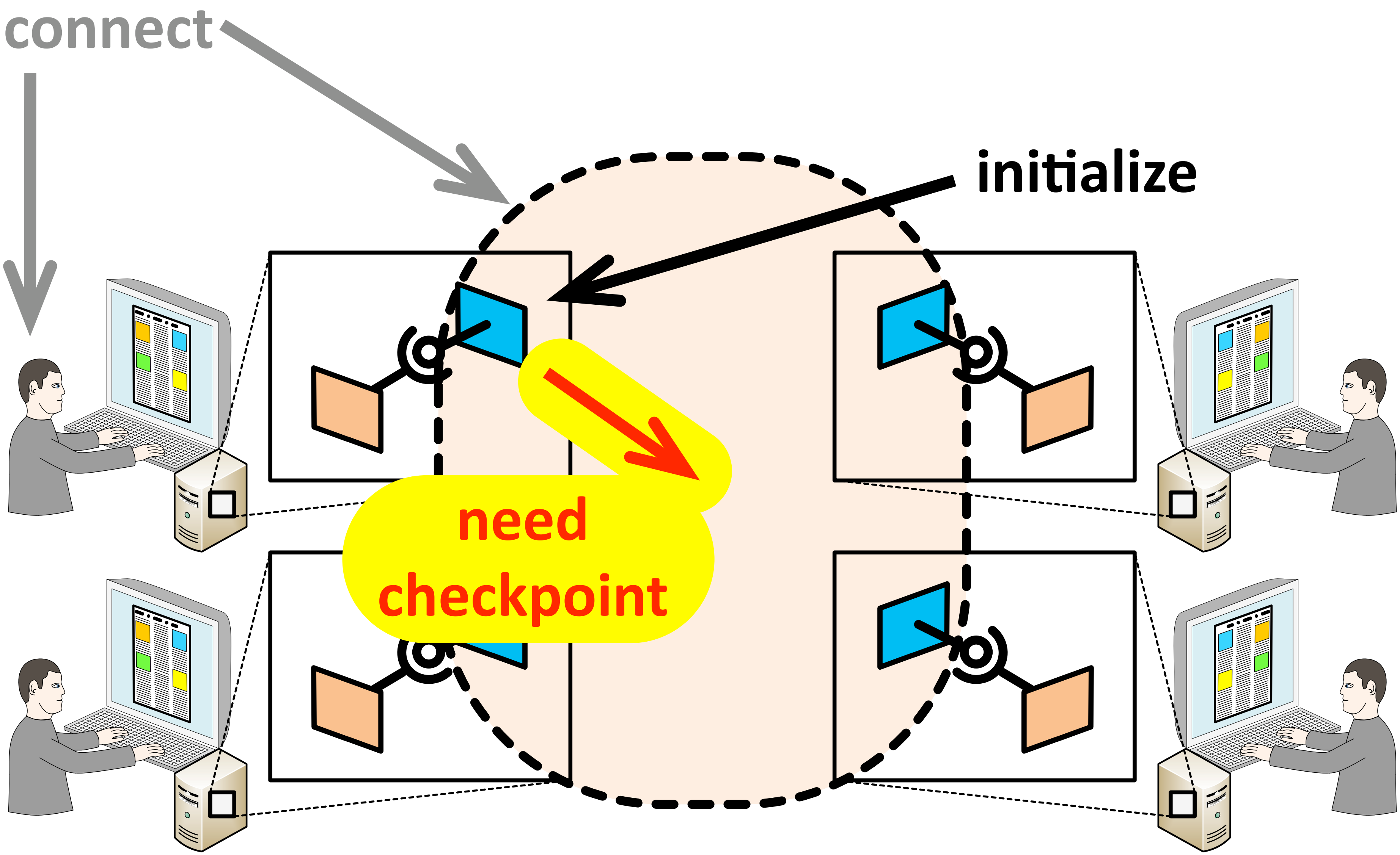
Dynamics (global)



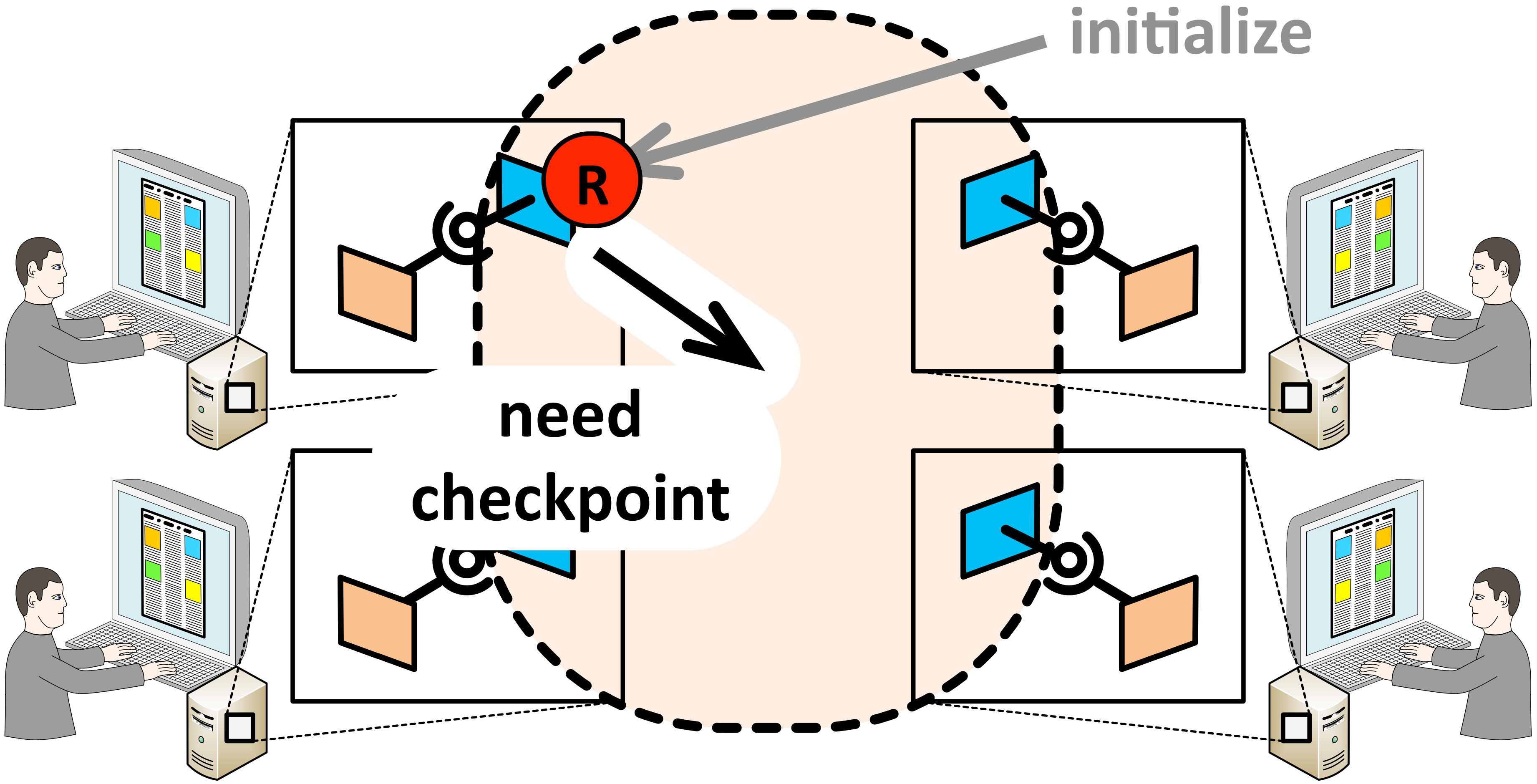
Dynamics (global)



Dynamics (global)

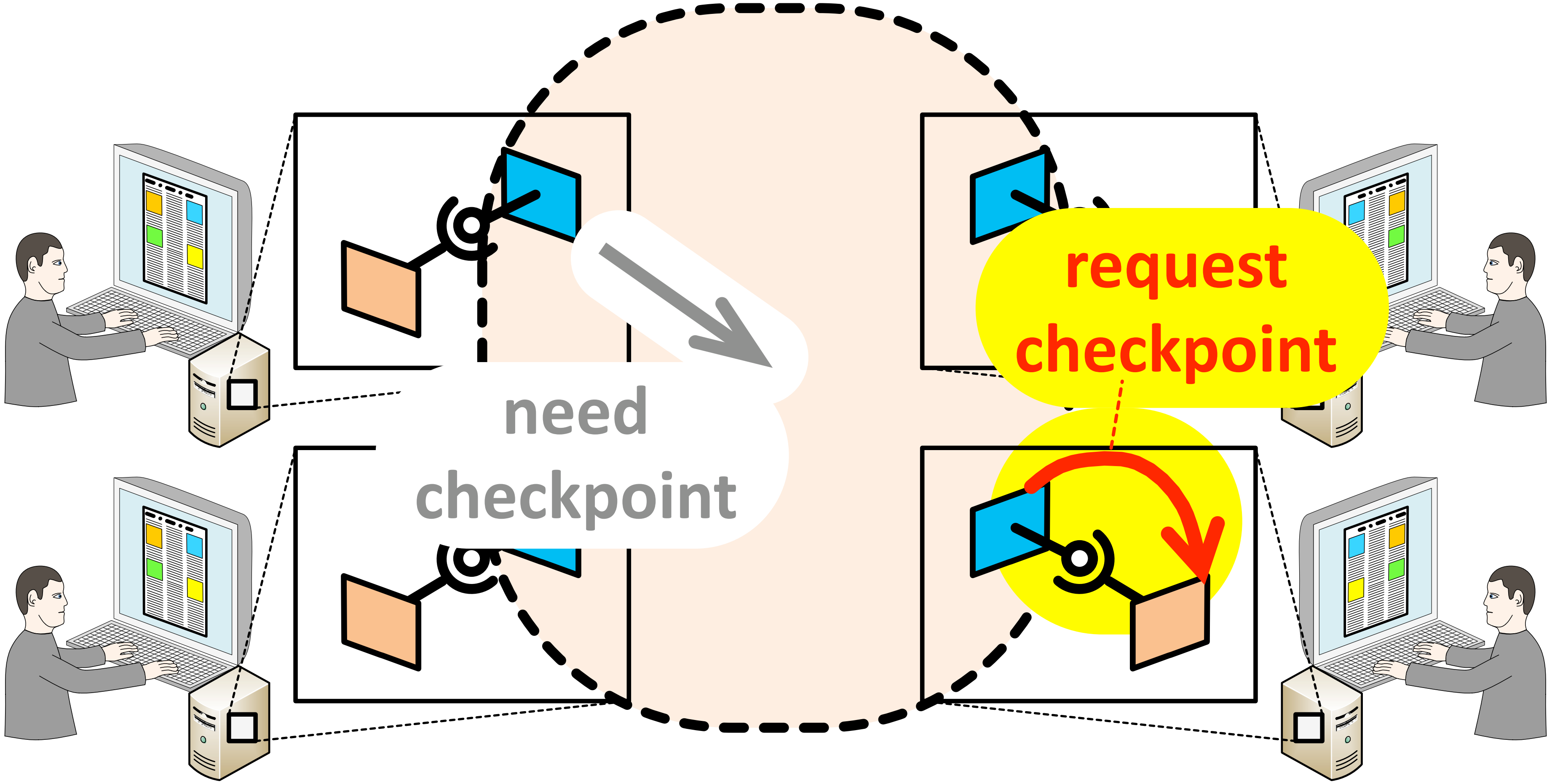


Dynamics (global)

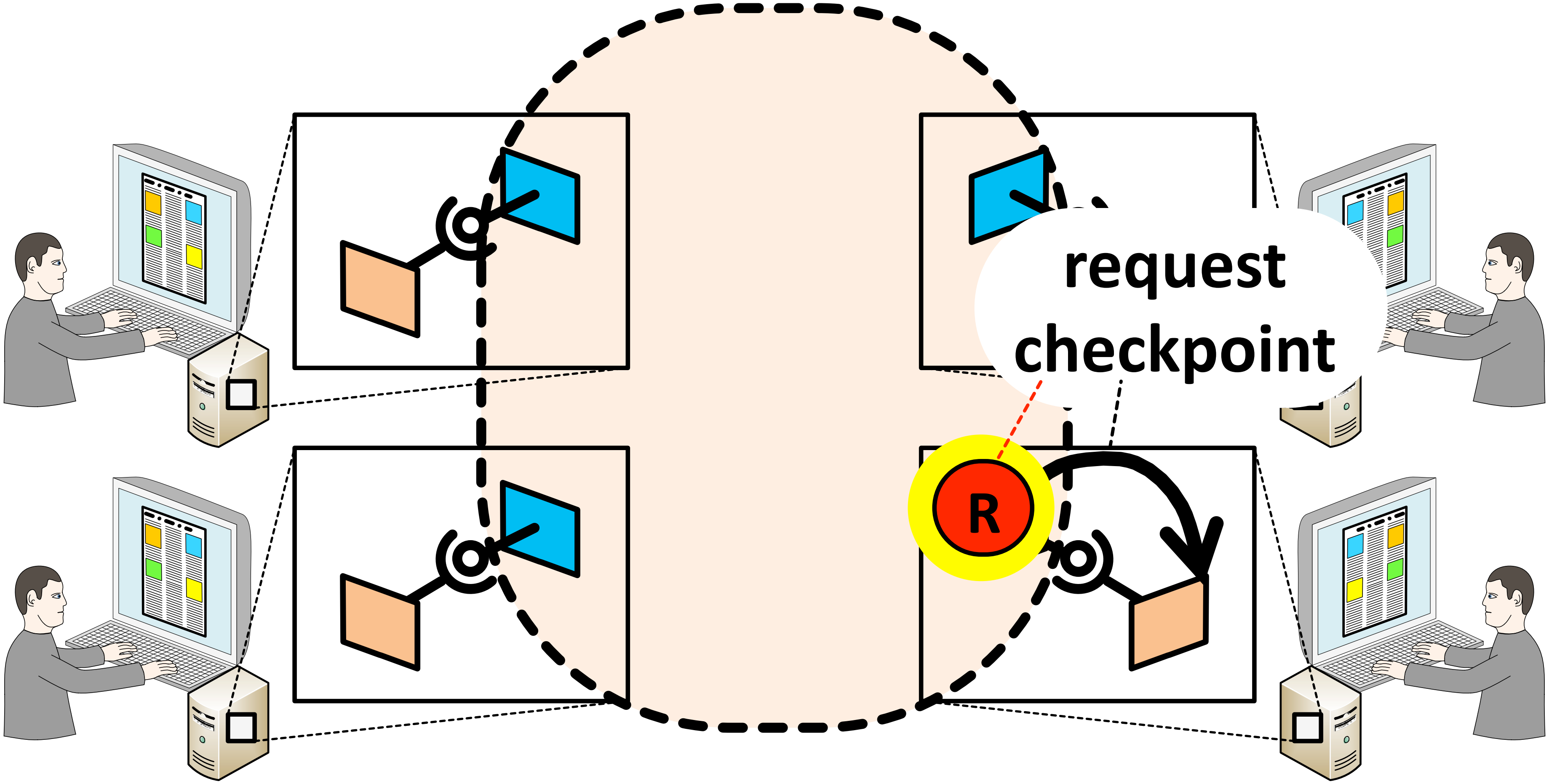


need checkpoint

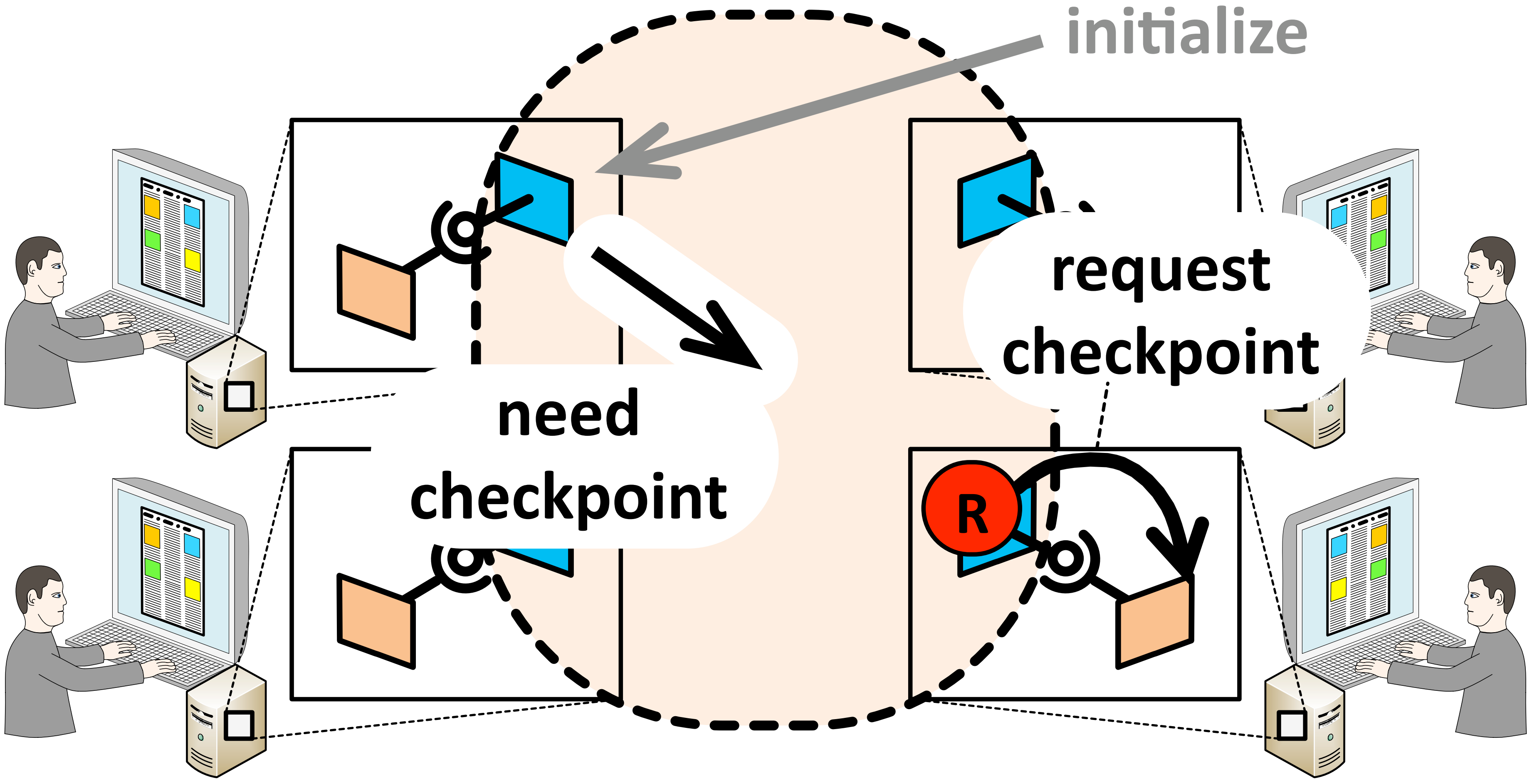
Dynamics (global)



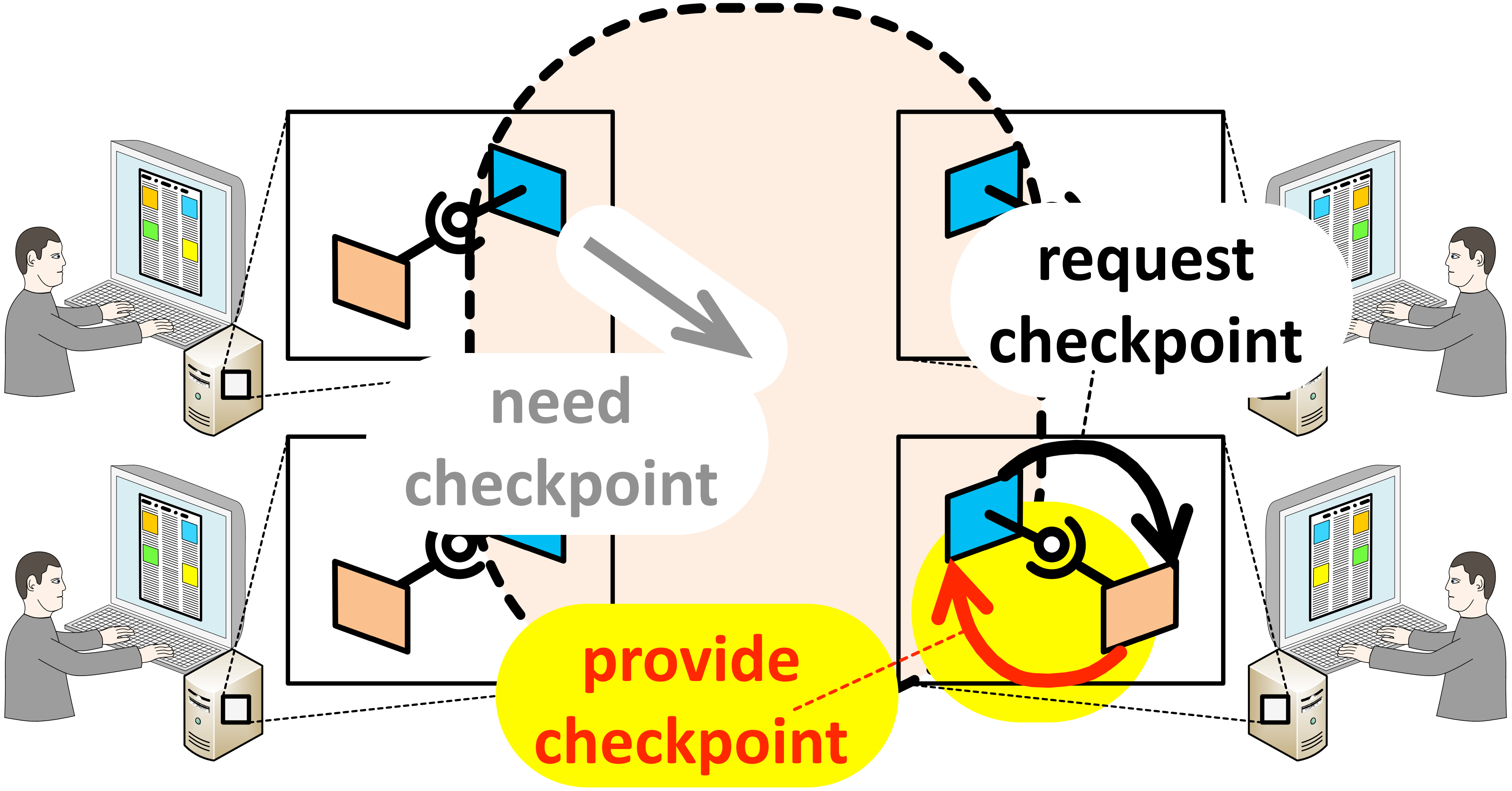
Dynamics (global)



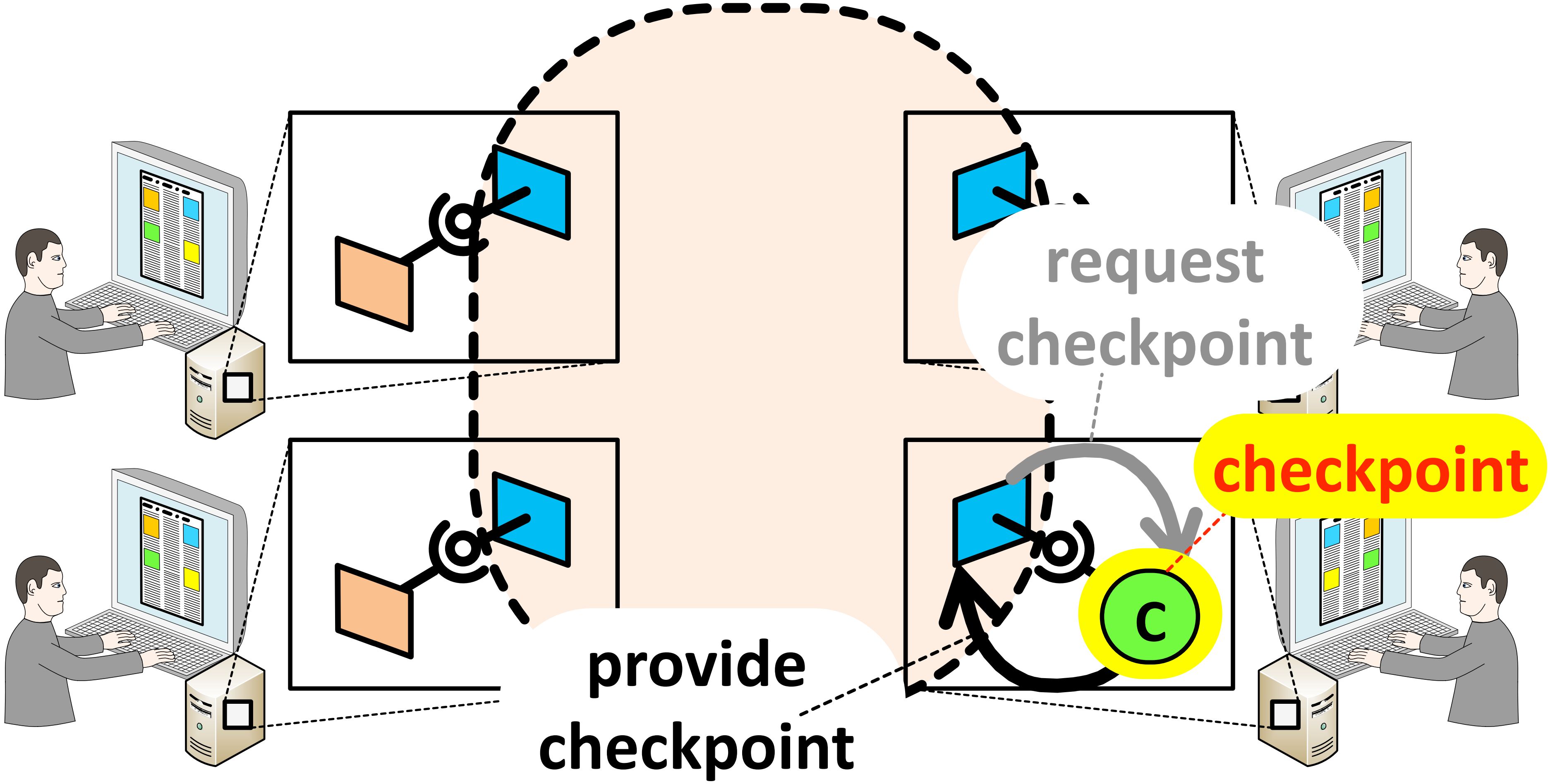
Dynamics (global)



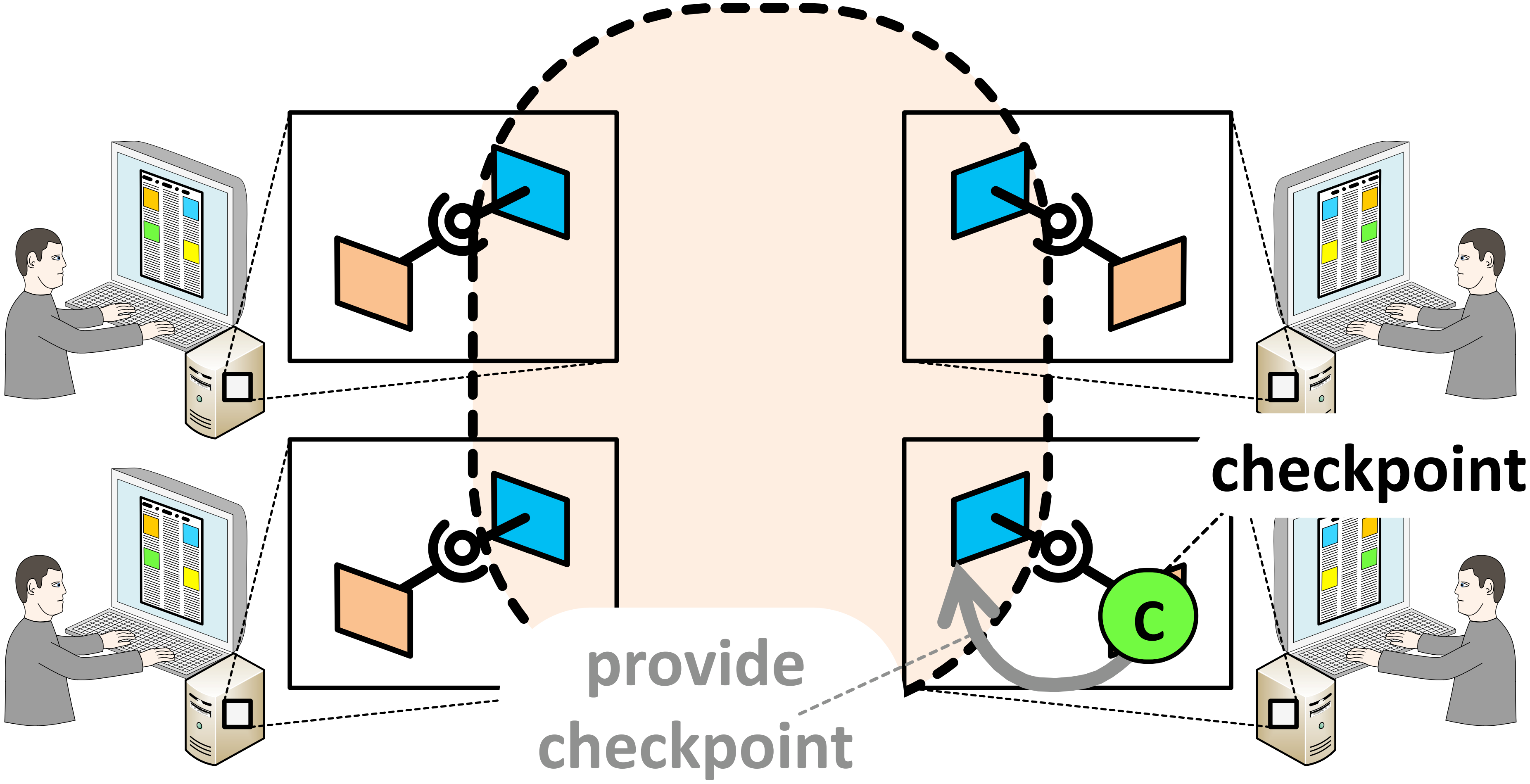
Dynamics (global)



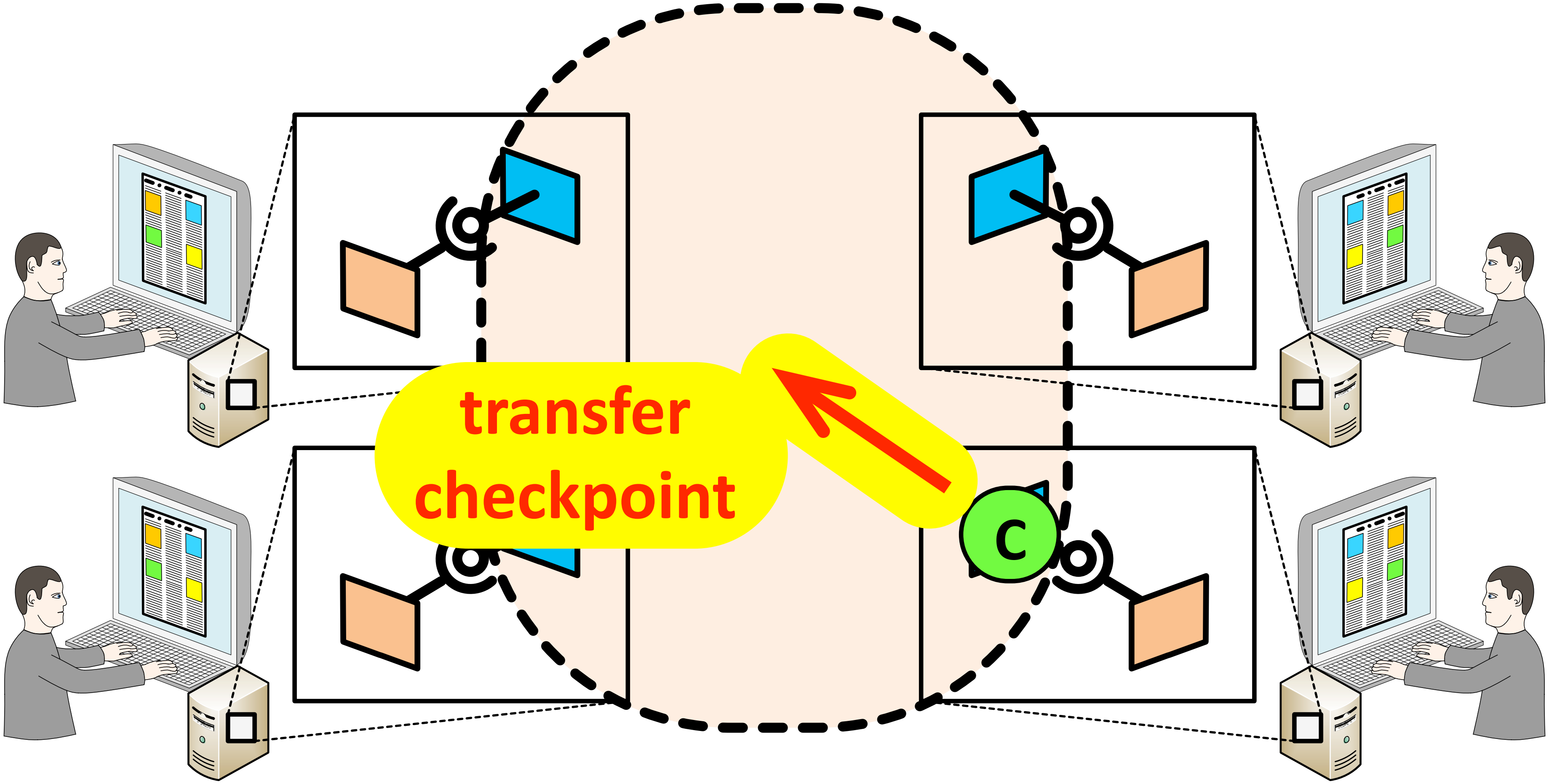
Dynamics (global)



Dynamics (global)

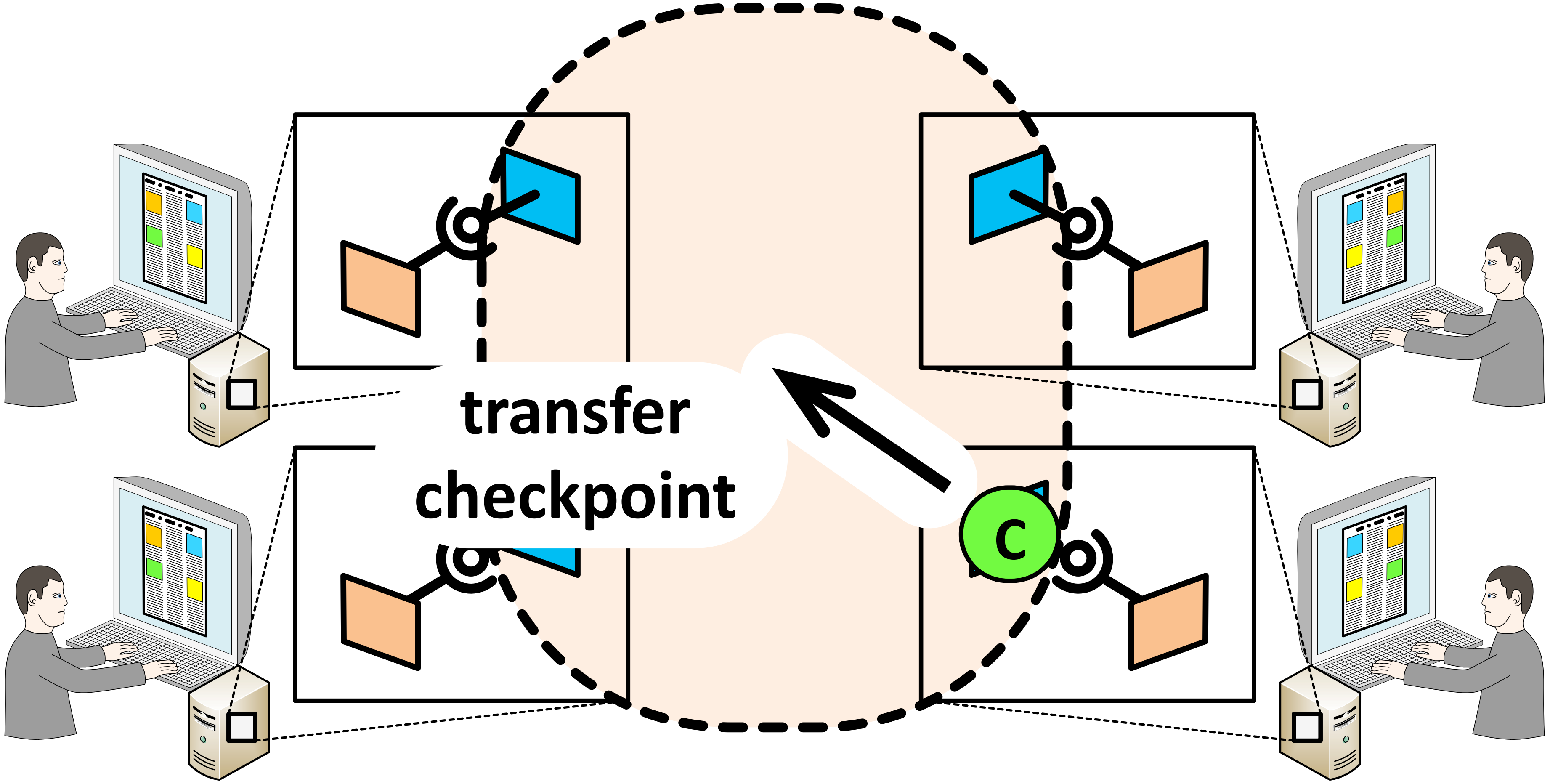


Dynamics (global)

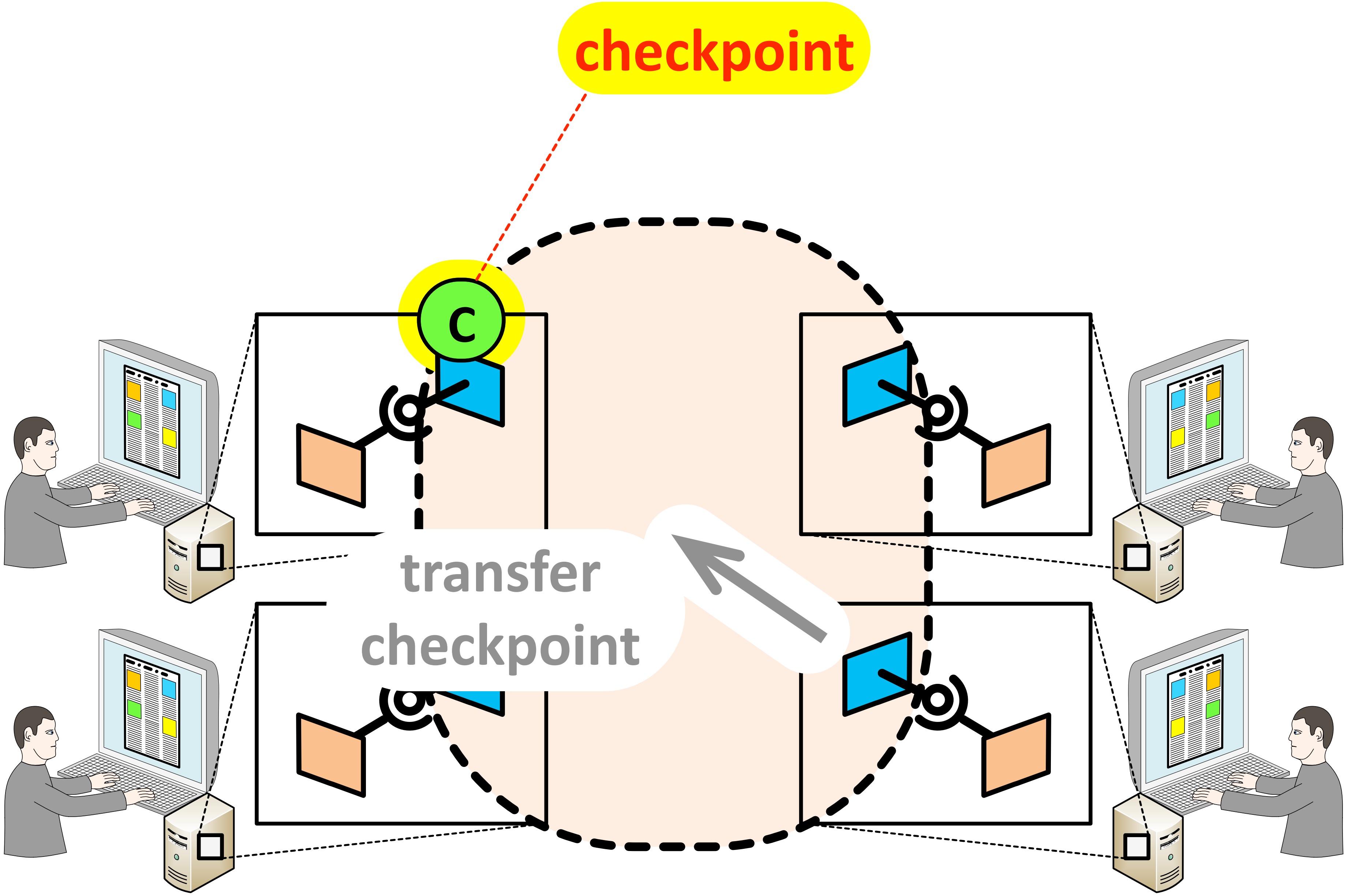


transfer checkpoint

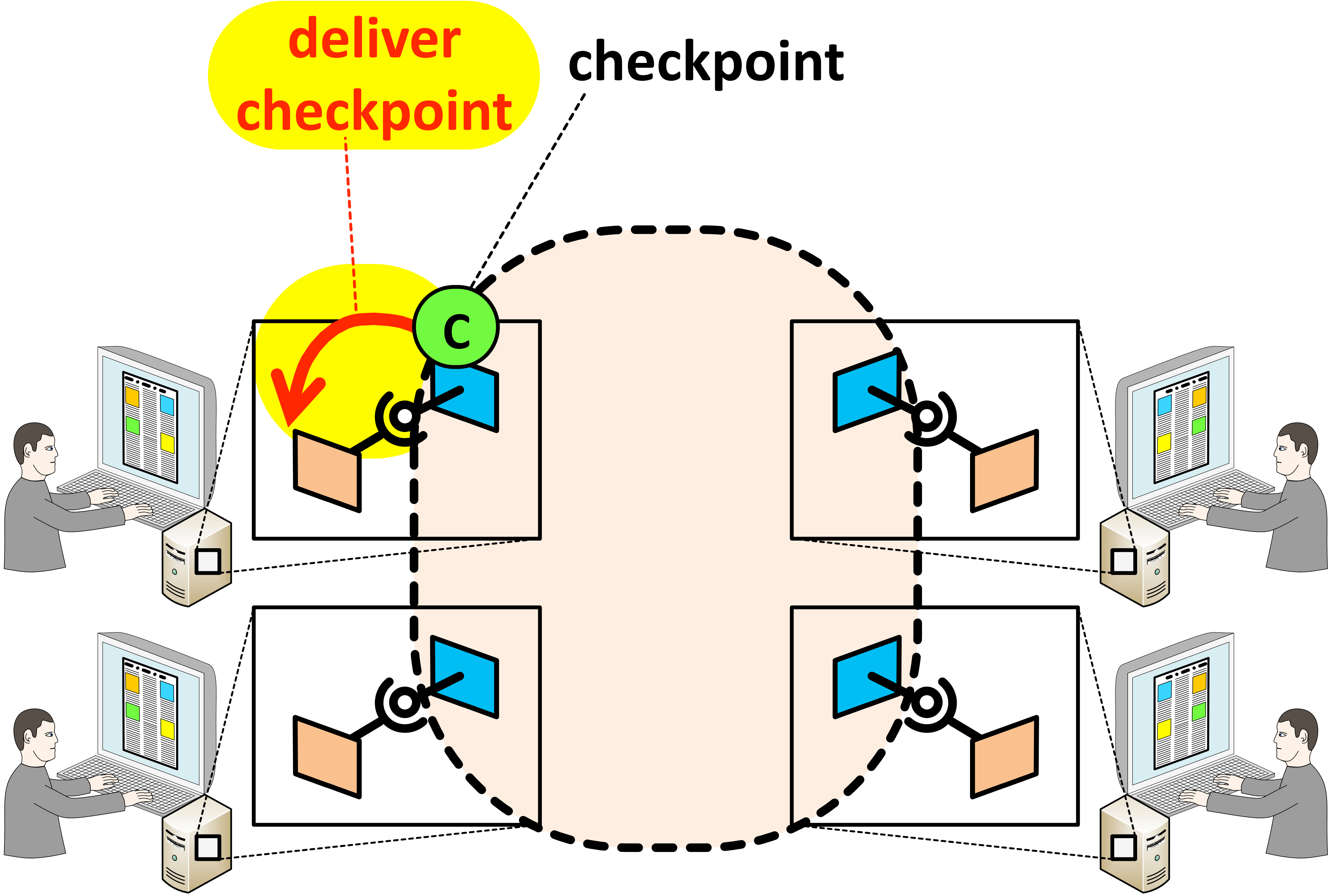
Dynamics (global)



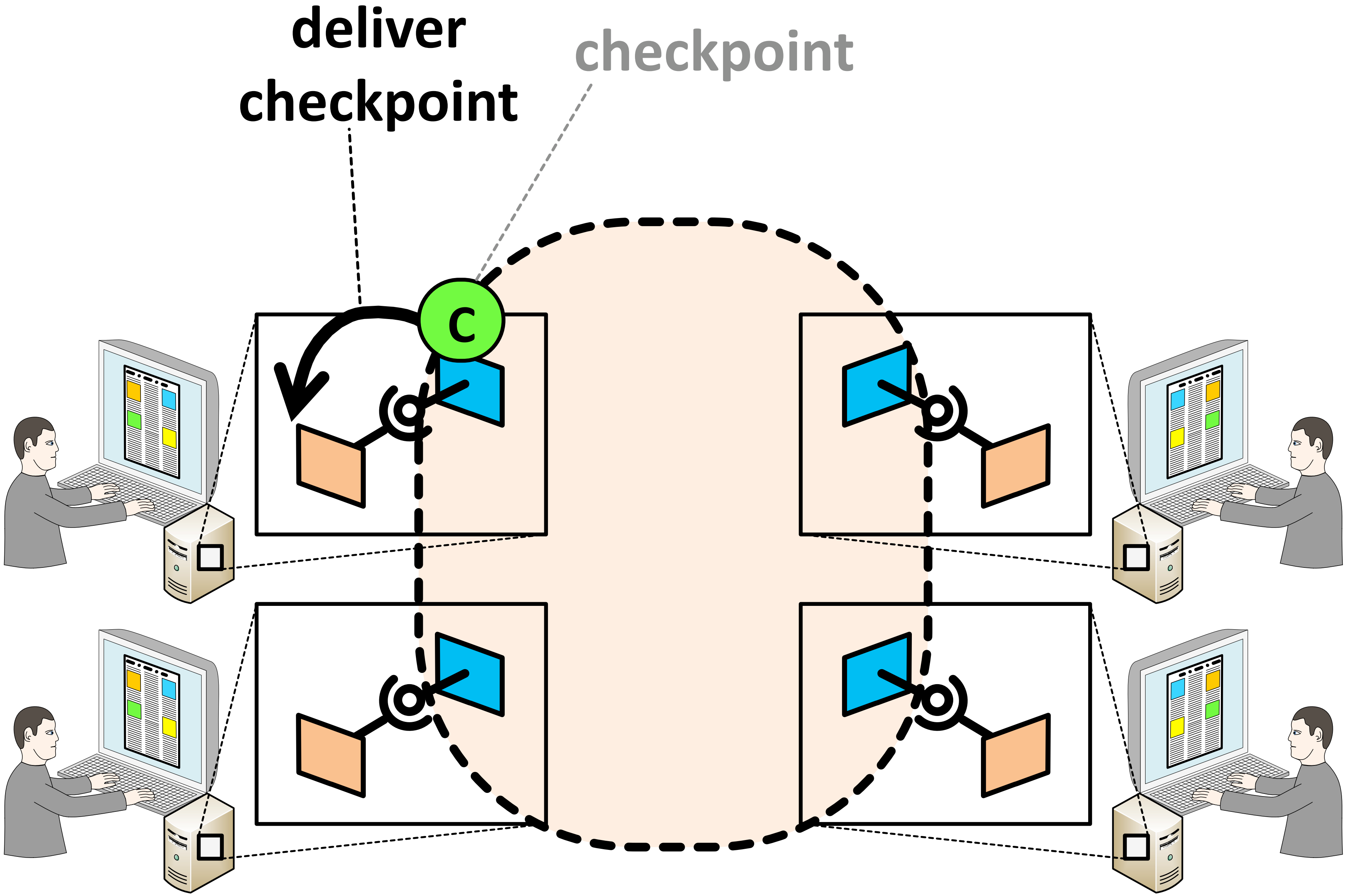
Dynamics (global)



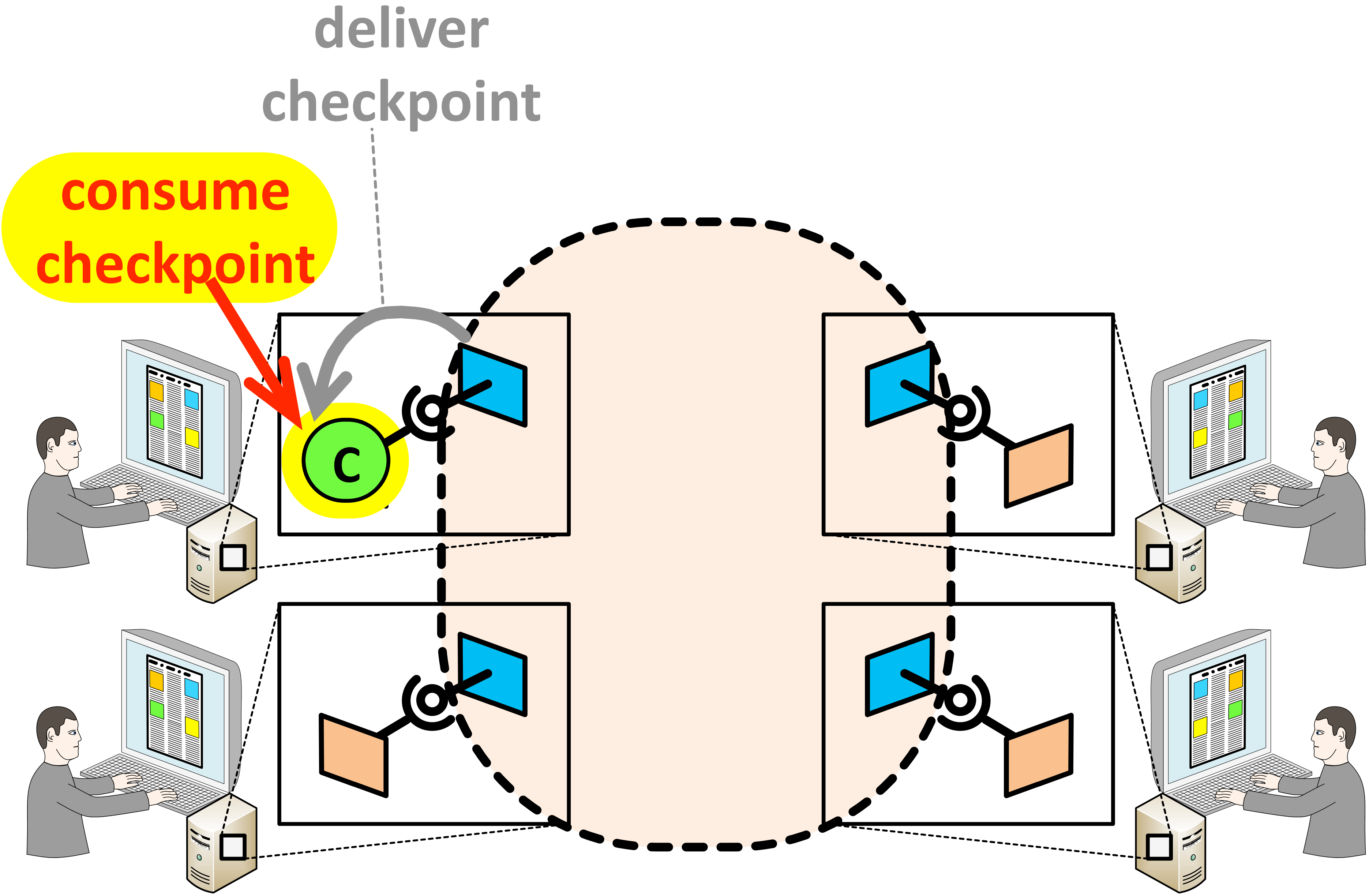
Dynamics (global)



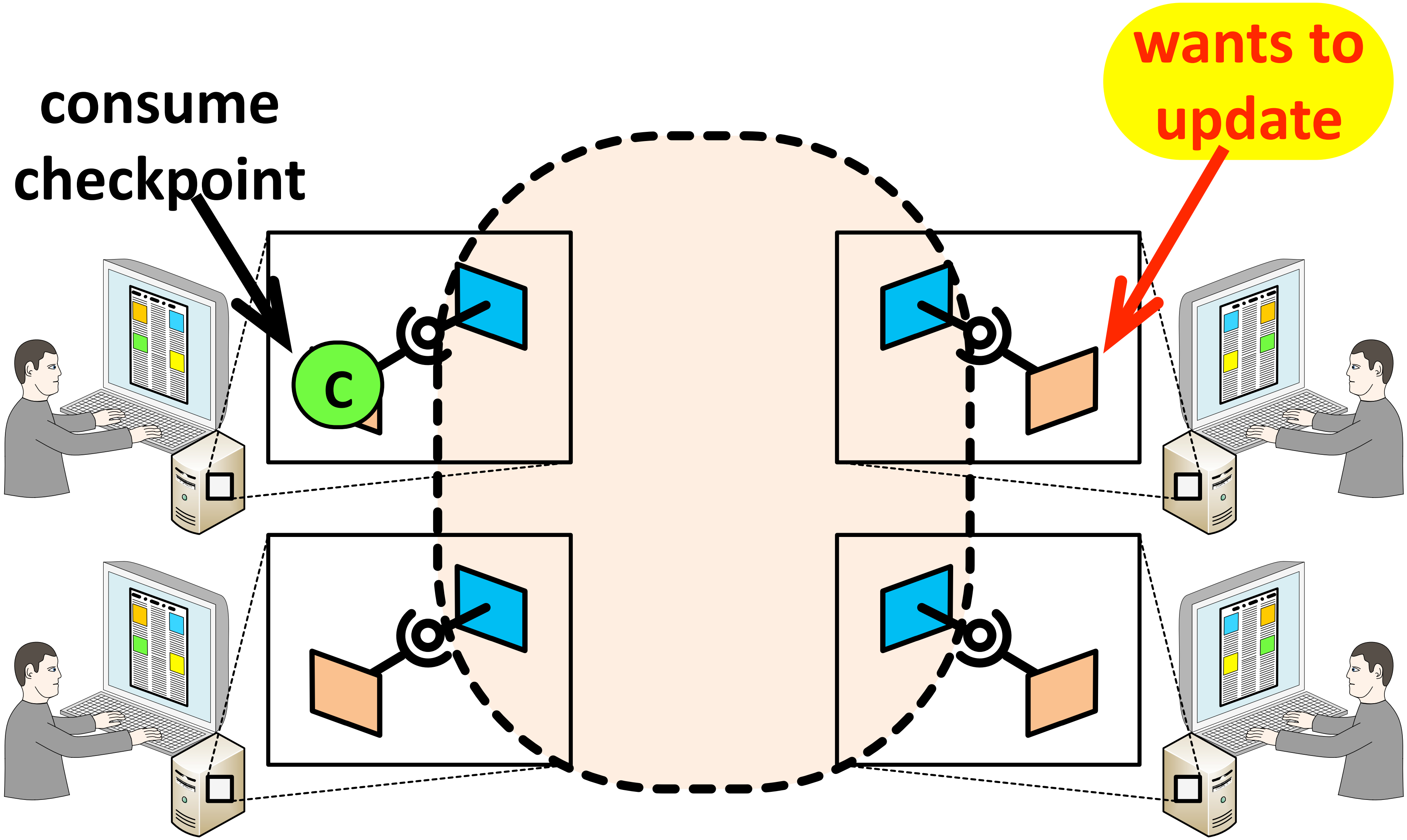
Dynamics (global)



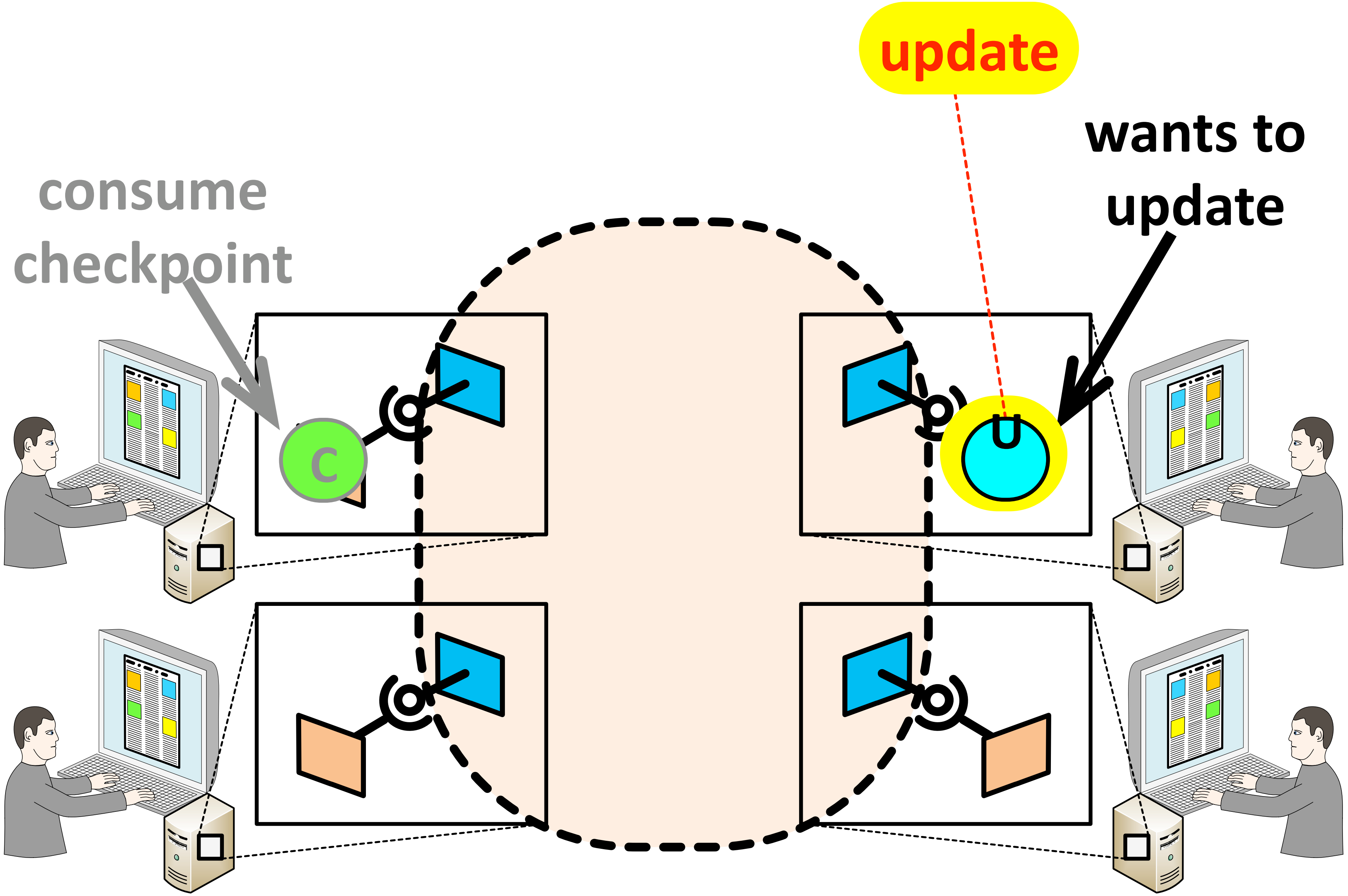
Dynamics (global)



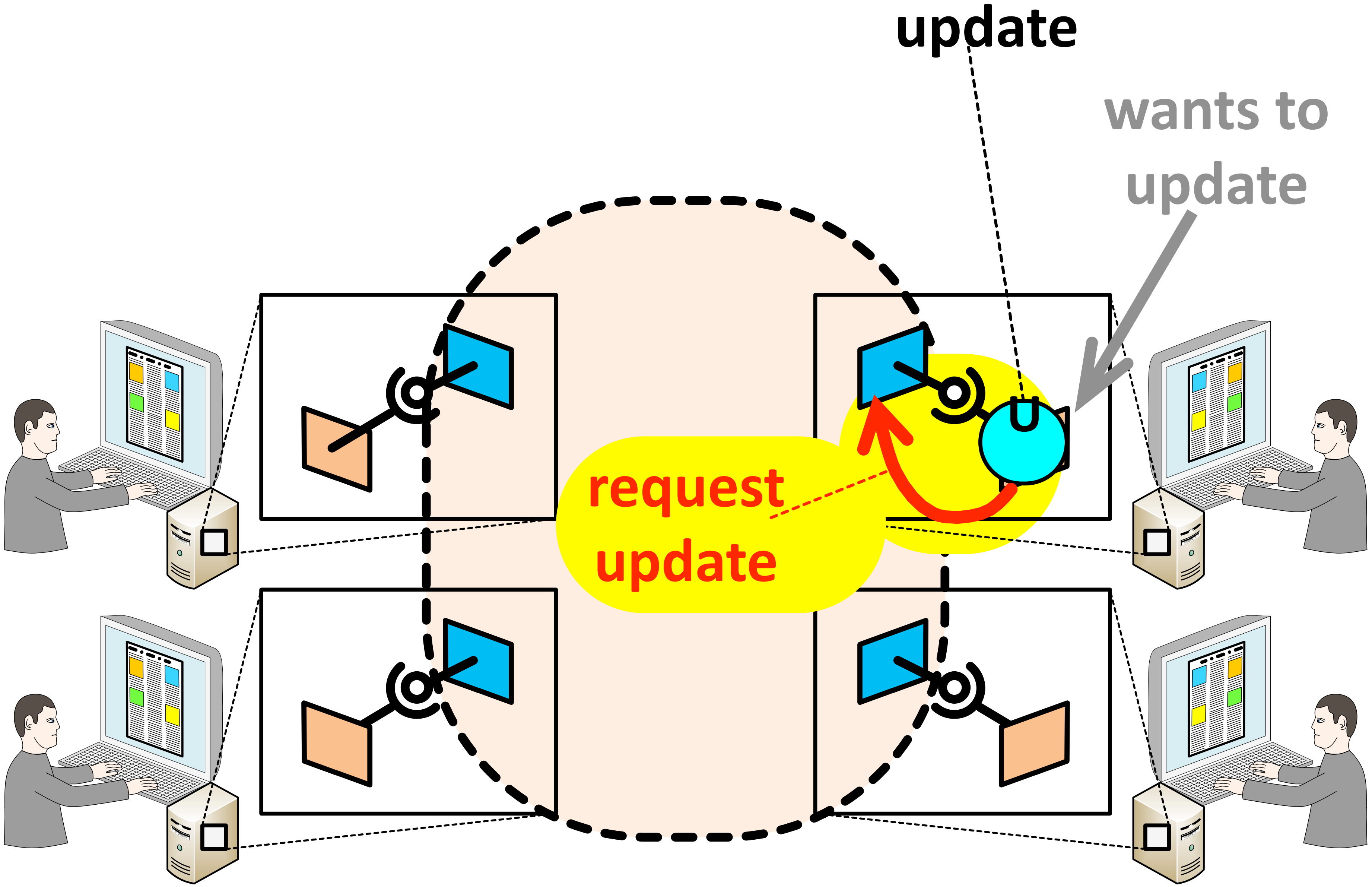
Dynamics (global)



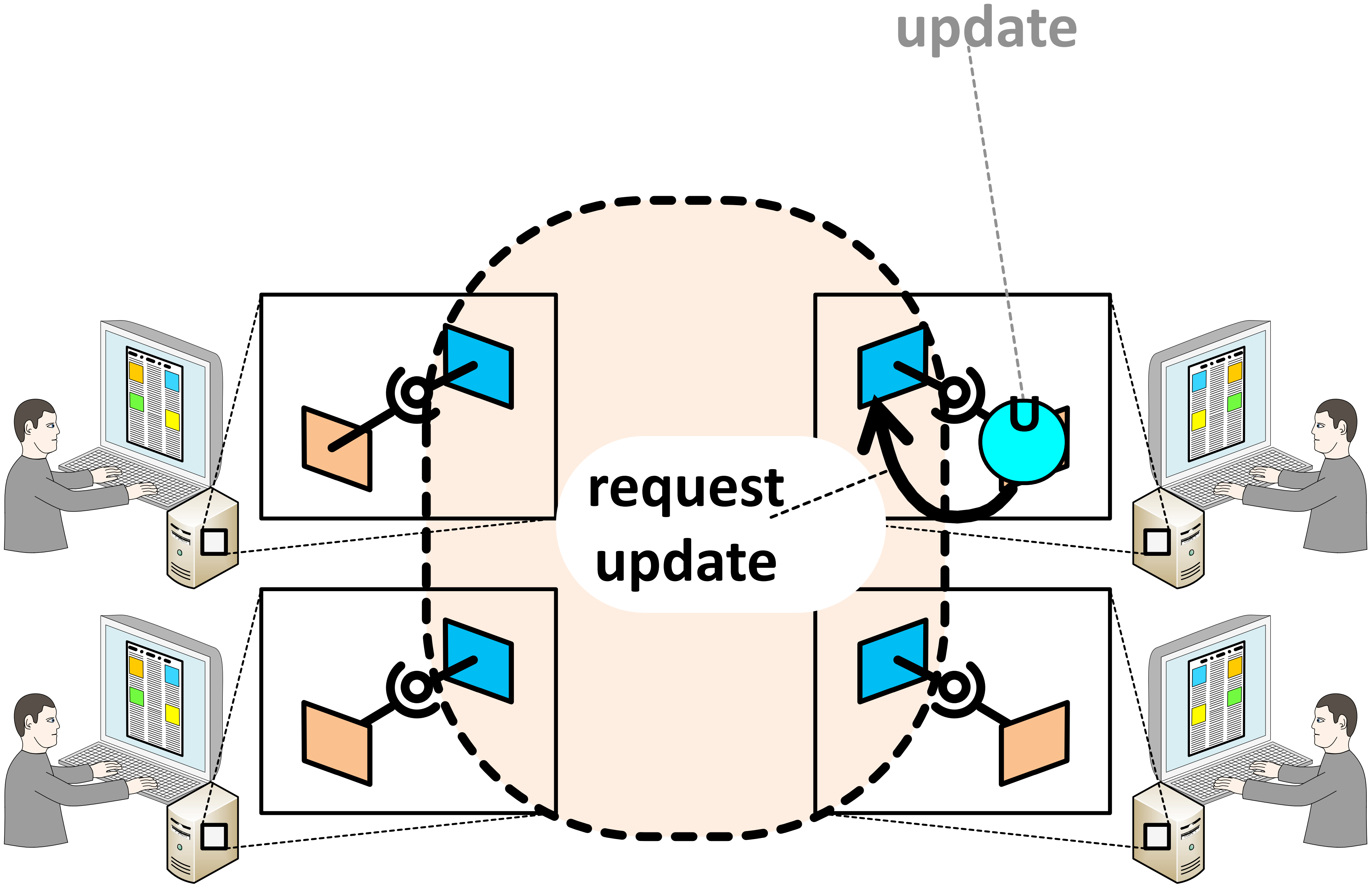
Dynamics (global)



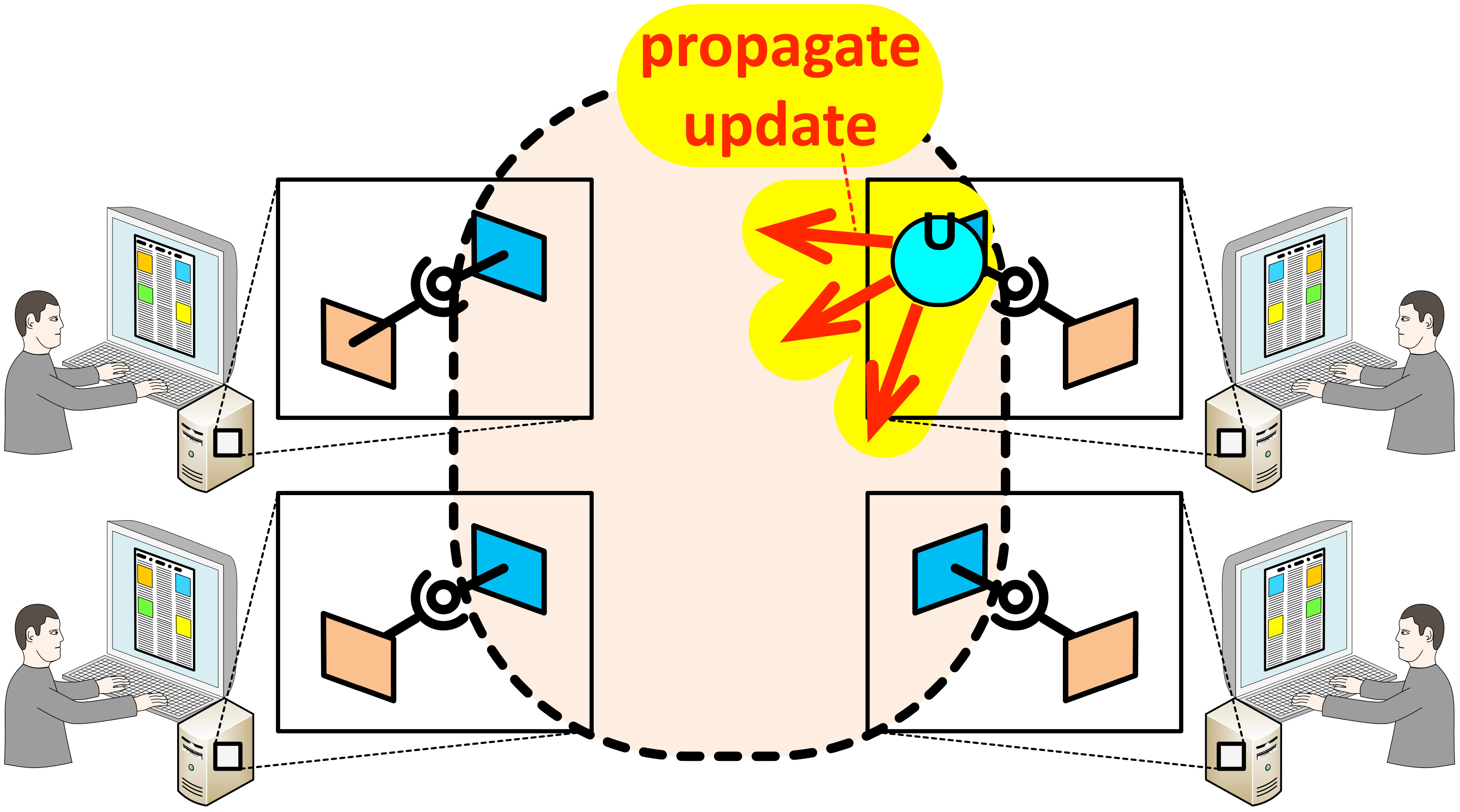
Dynamics (global)



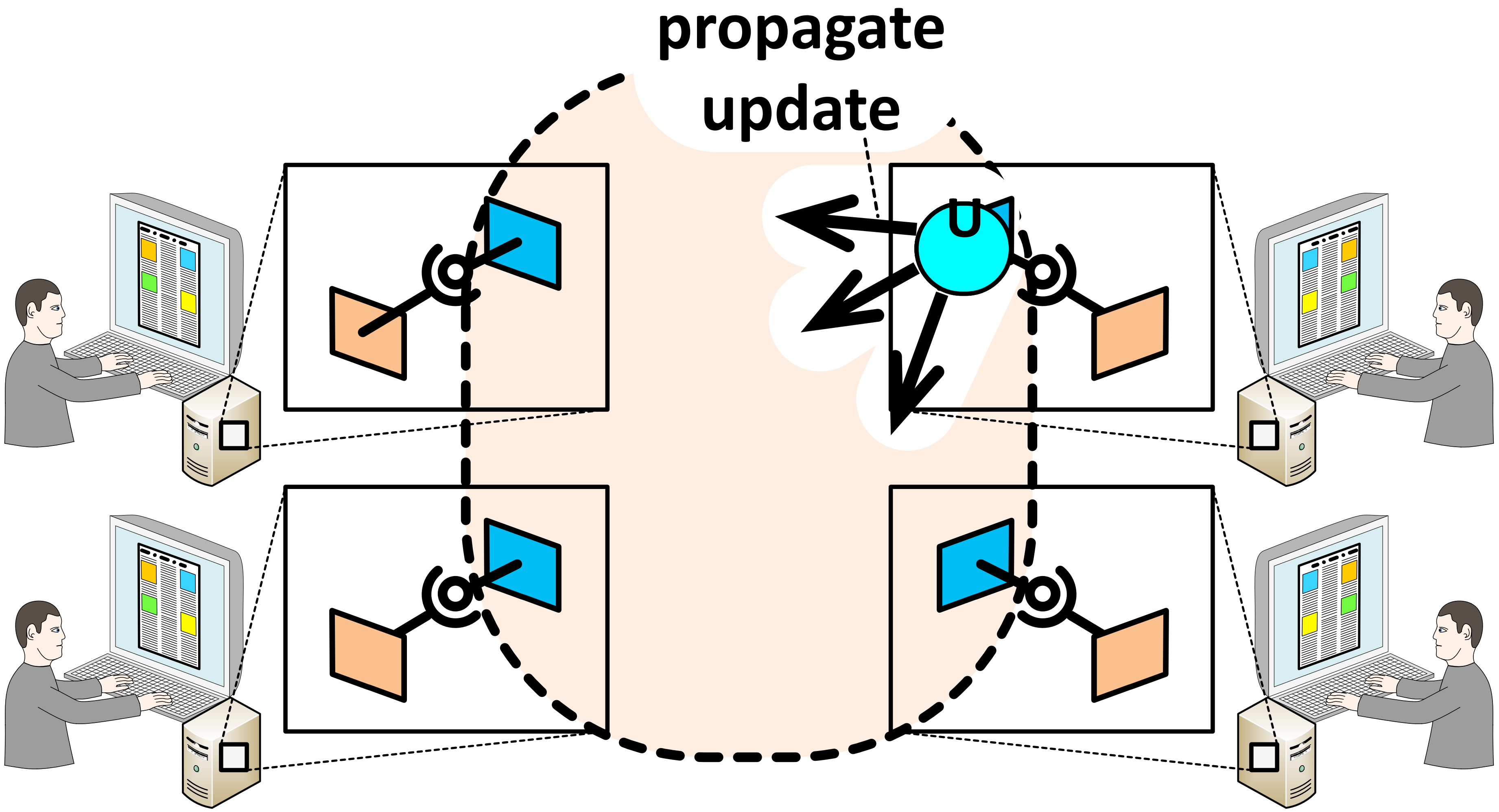
Dynamics (global)



Dynamics (global)

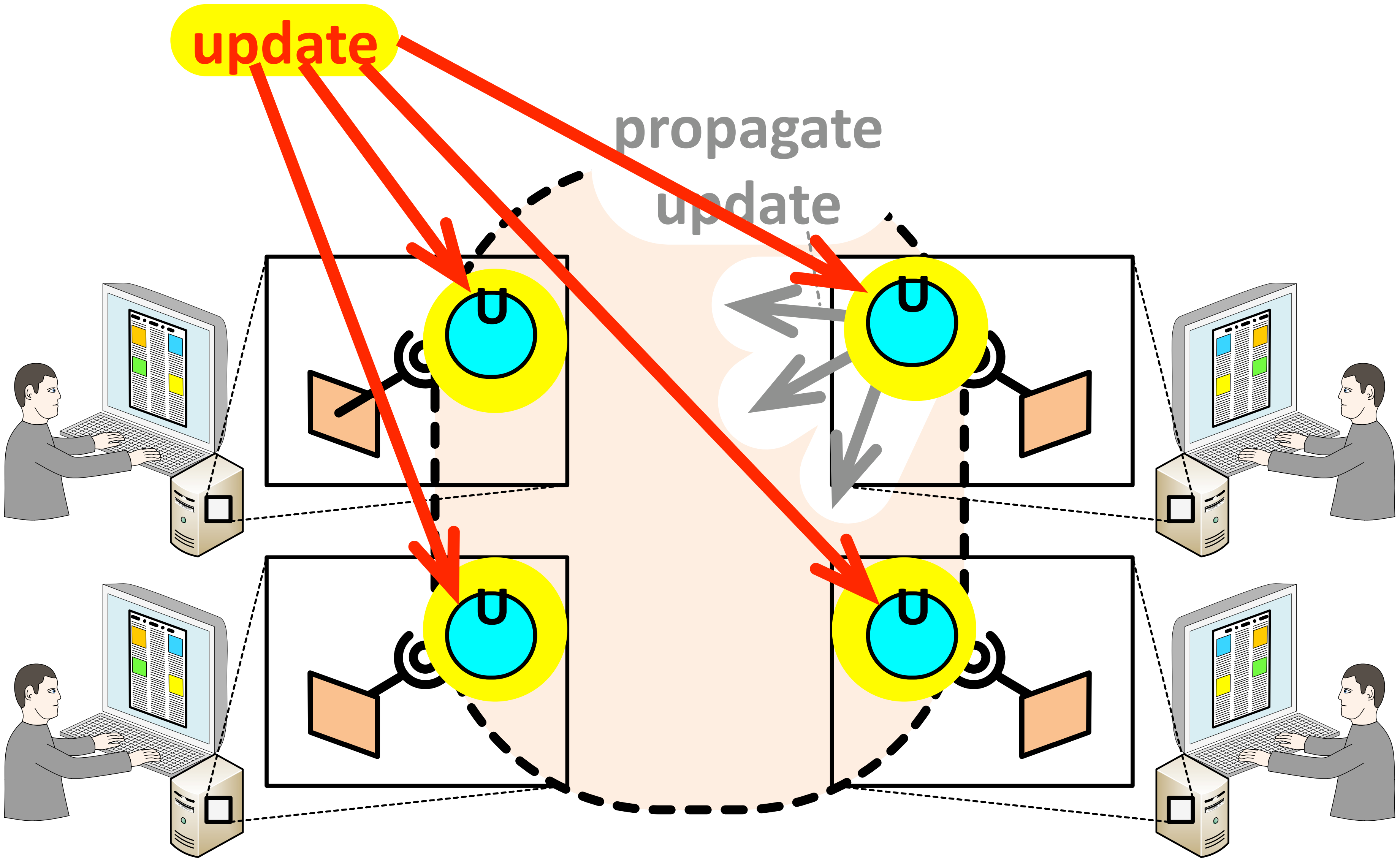


Dynamics (global)

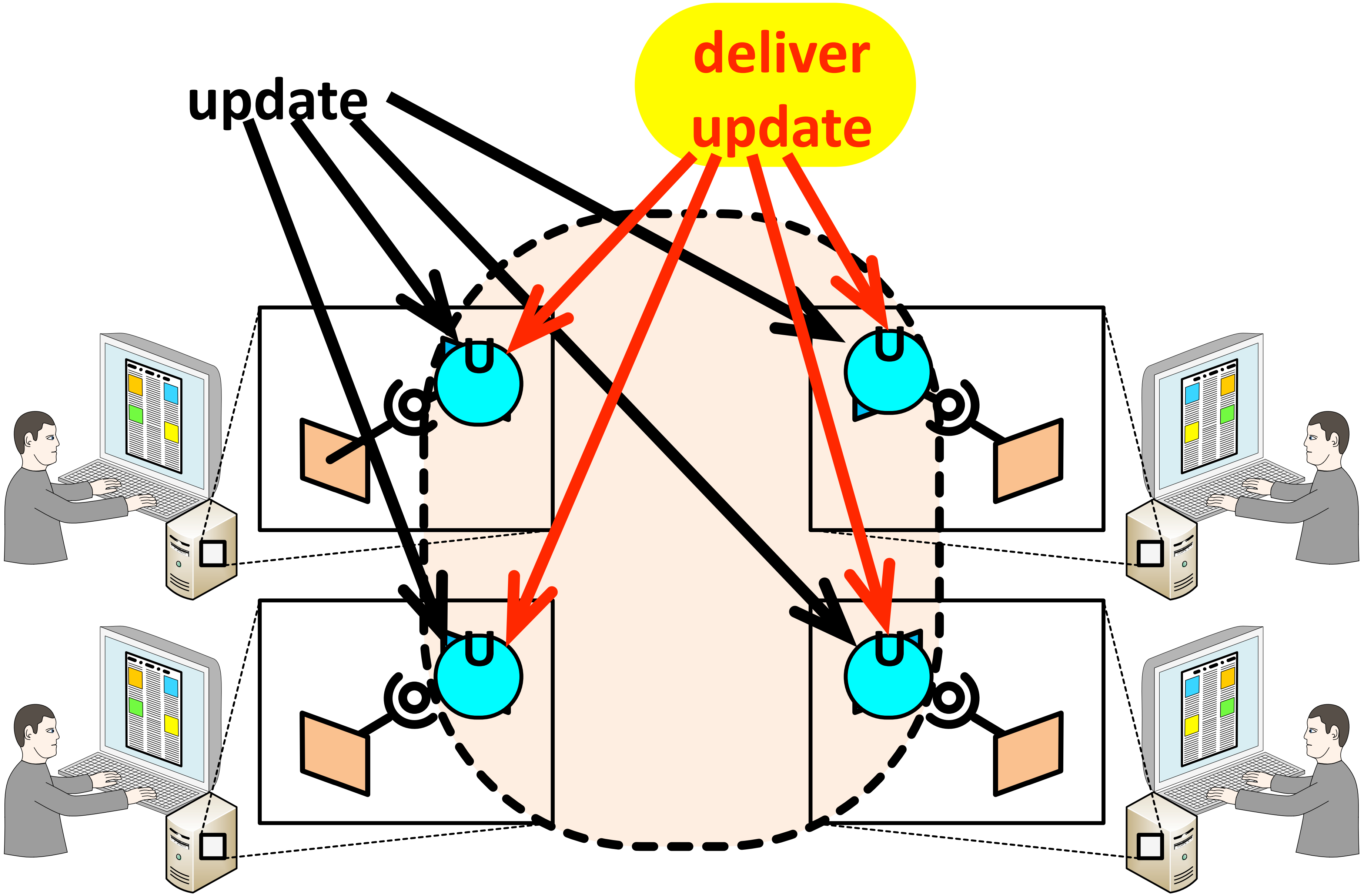


propagate update

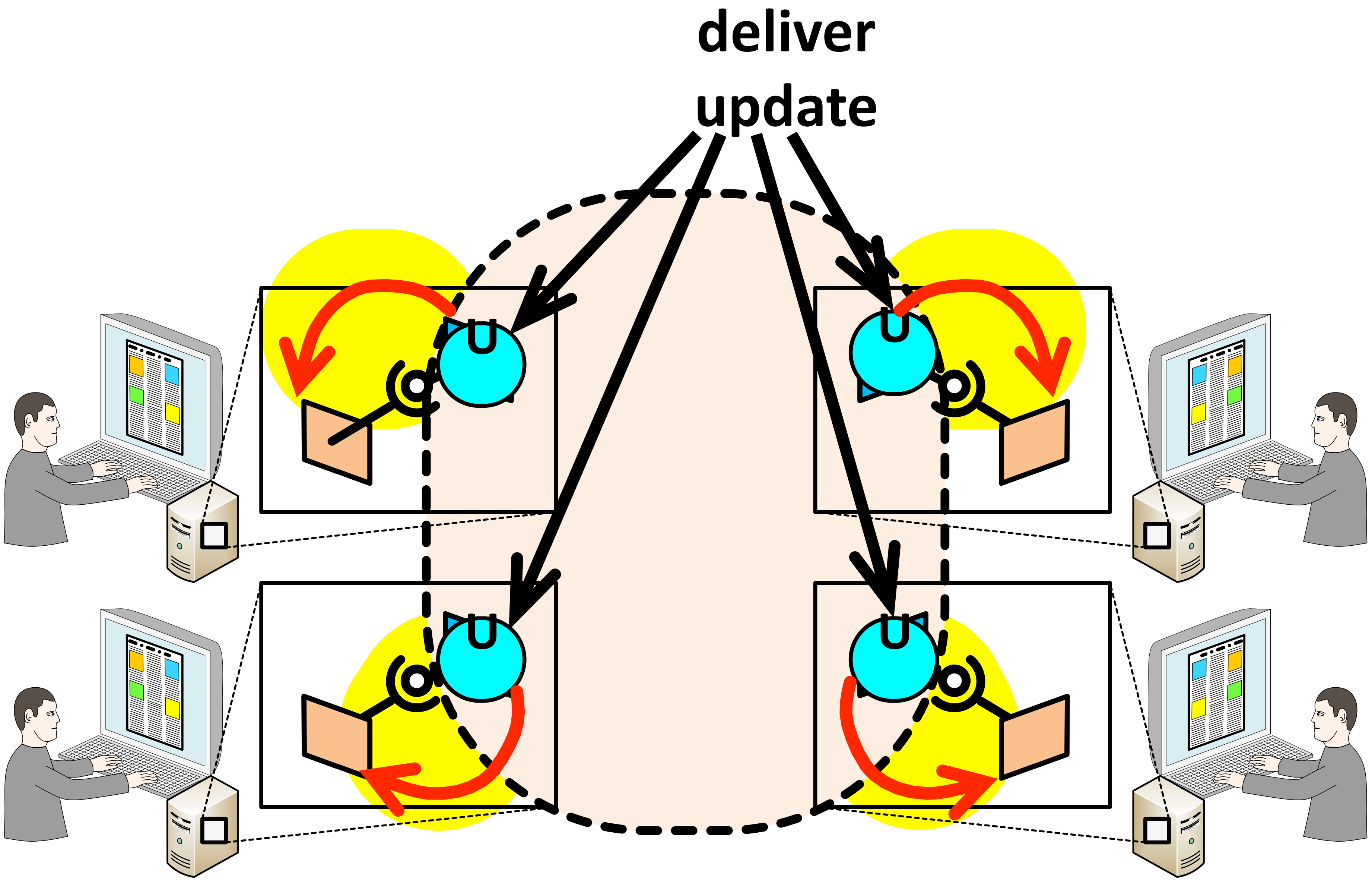
Dynamics (global)



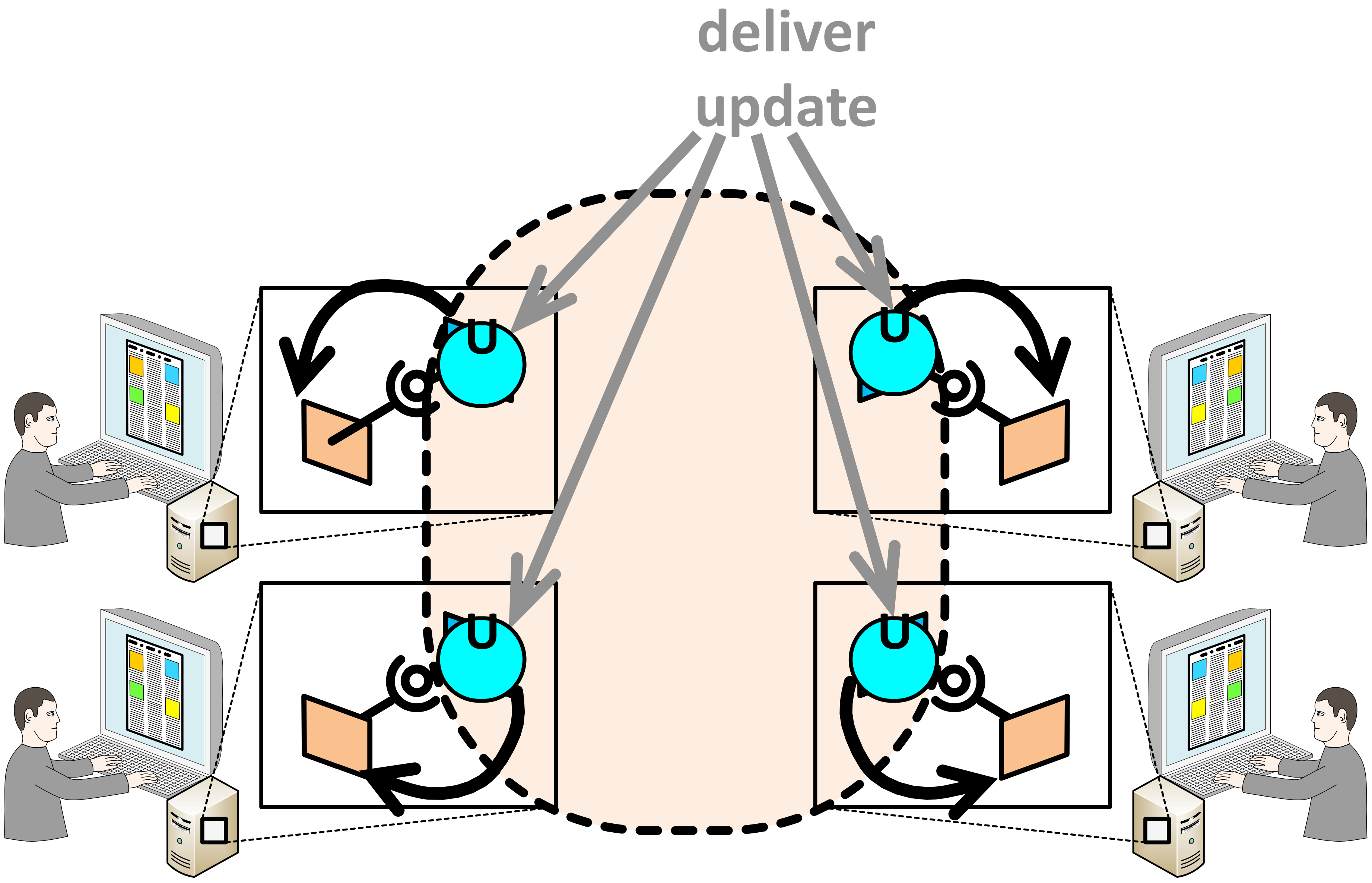
Dynamics (global)



Dynamics (global)

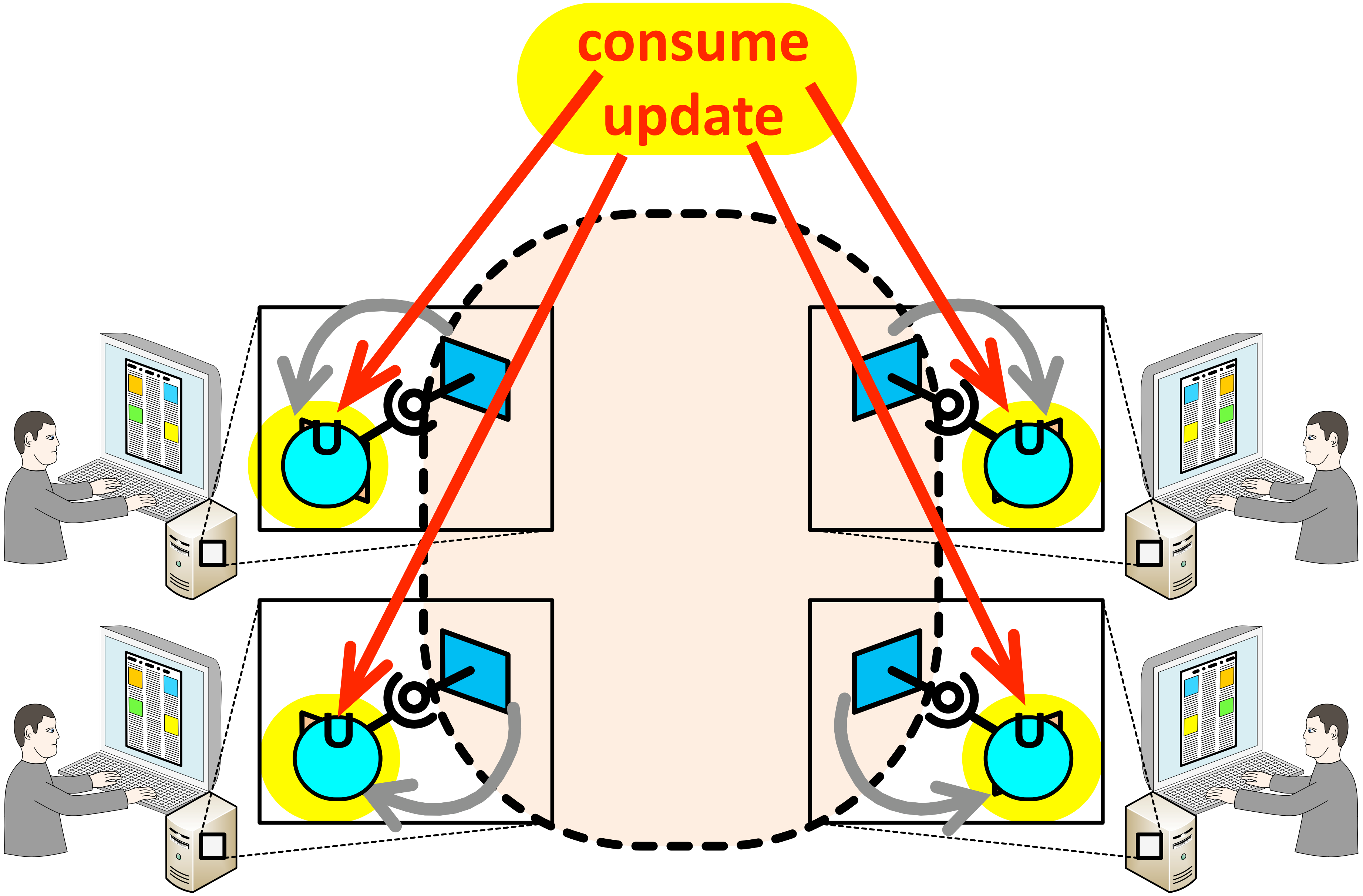


Dynamics (global)

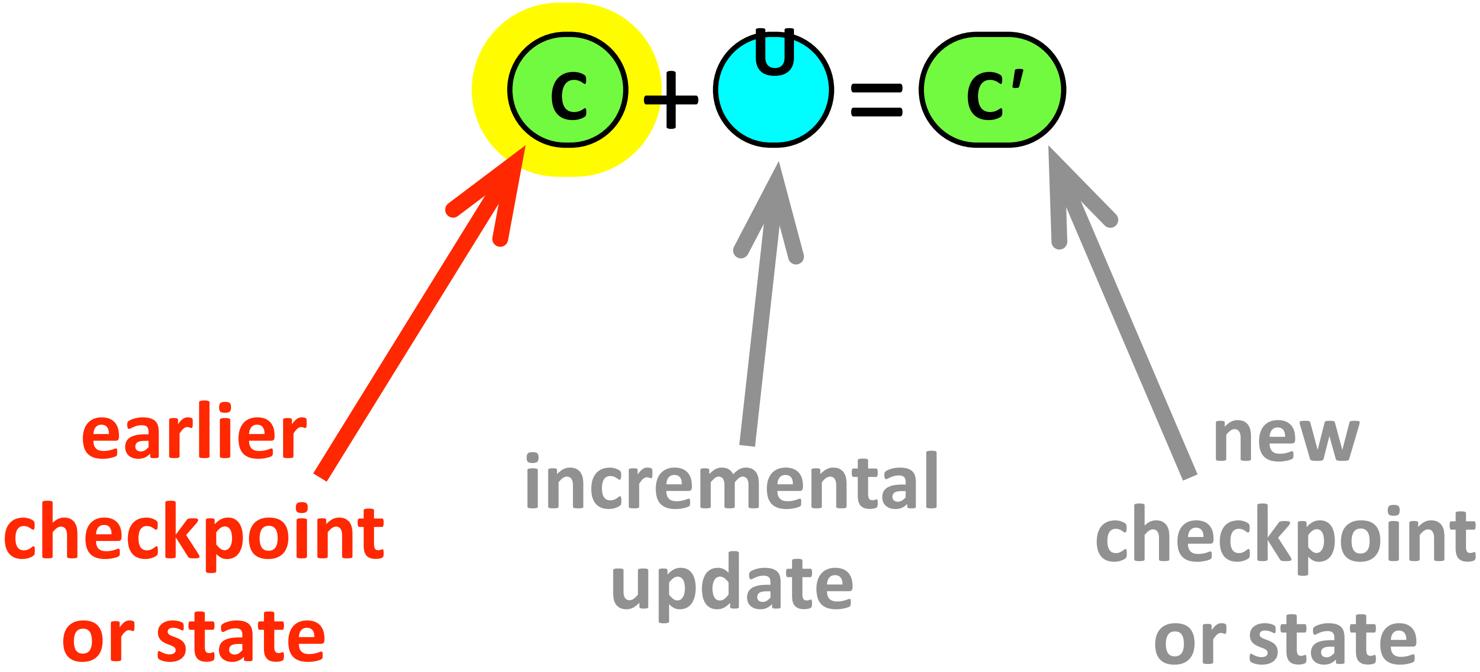


deliver update

Dynamics (global)



Semantics



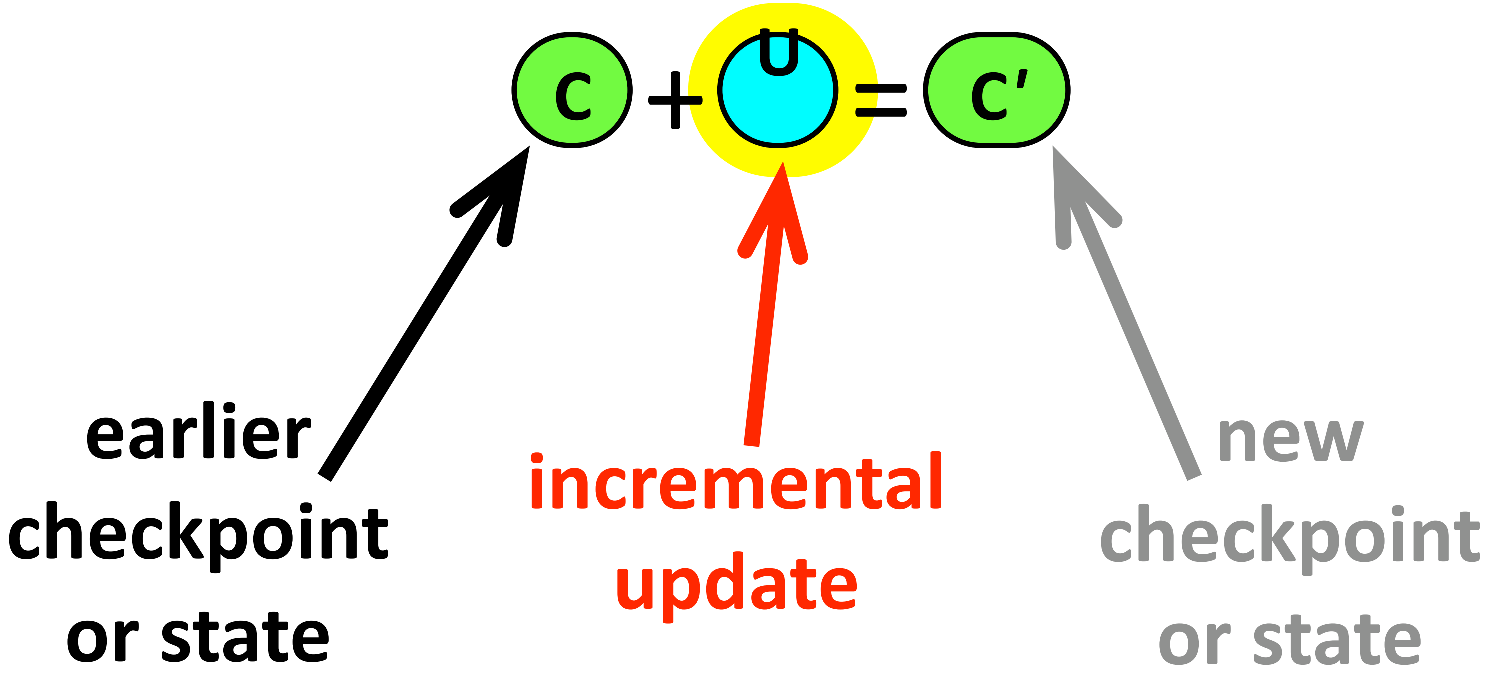
$c + u = c'$

earlier checkpoint or state

incremental update

new checkpoint or state

Semantics

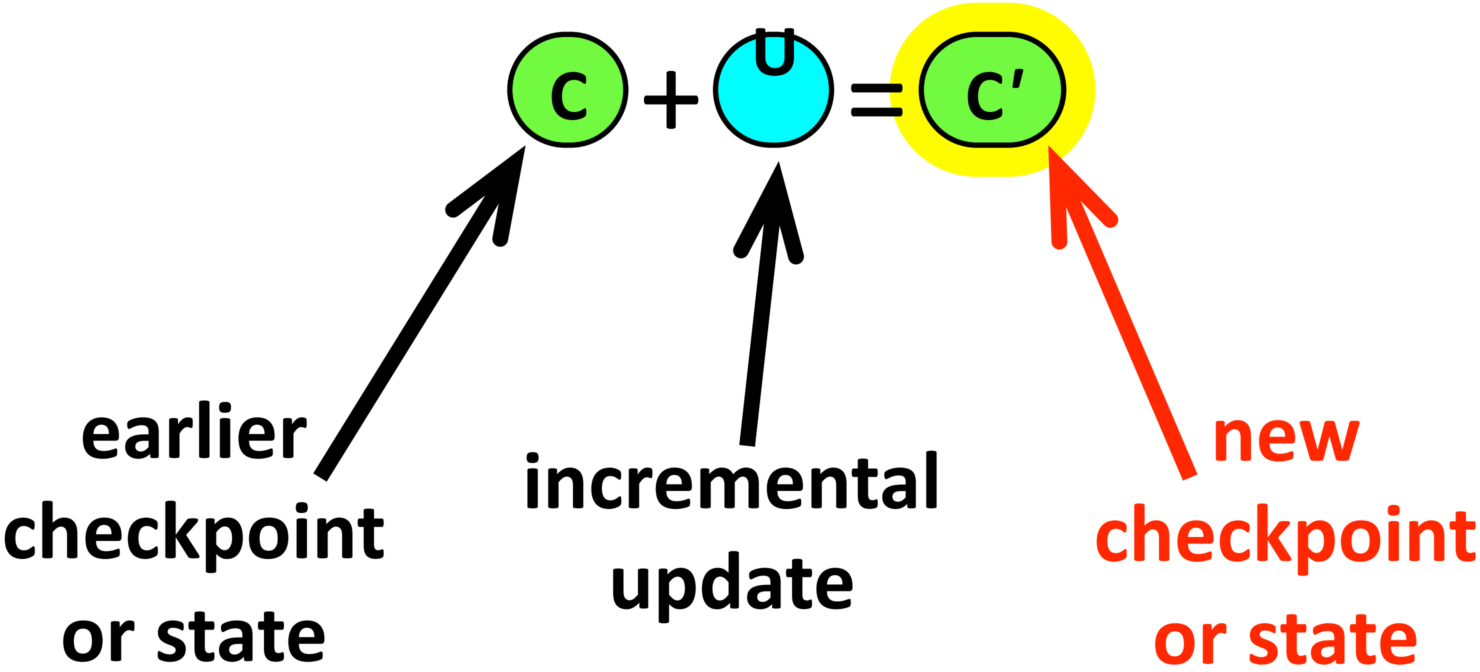


earlier checkpoint or state

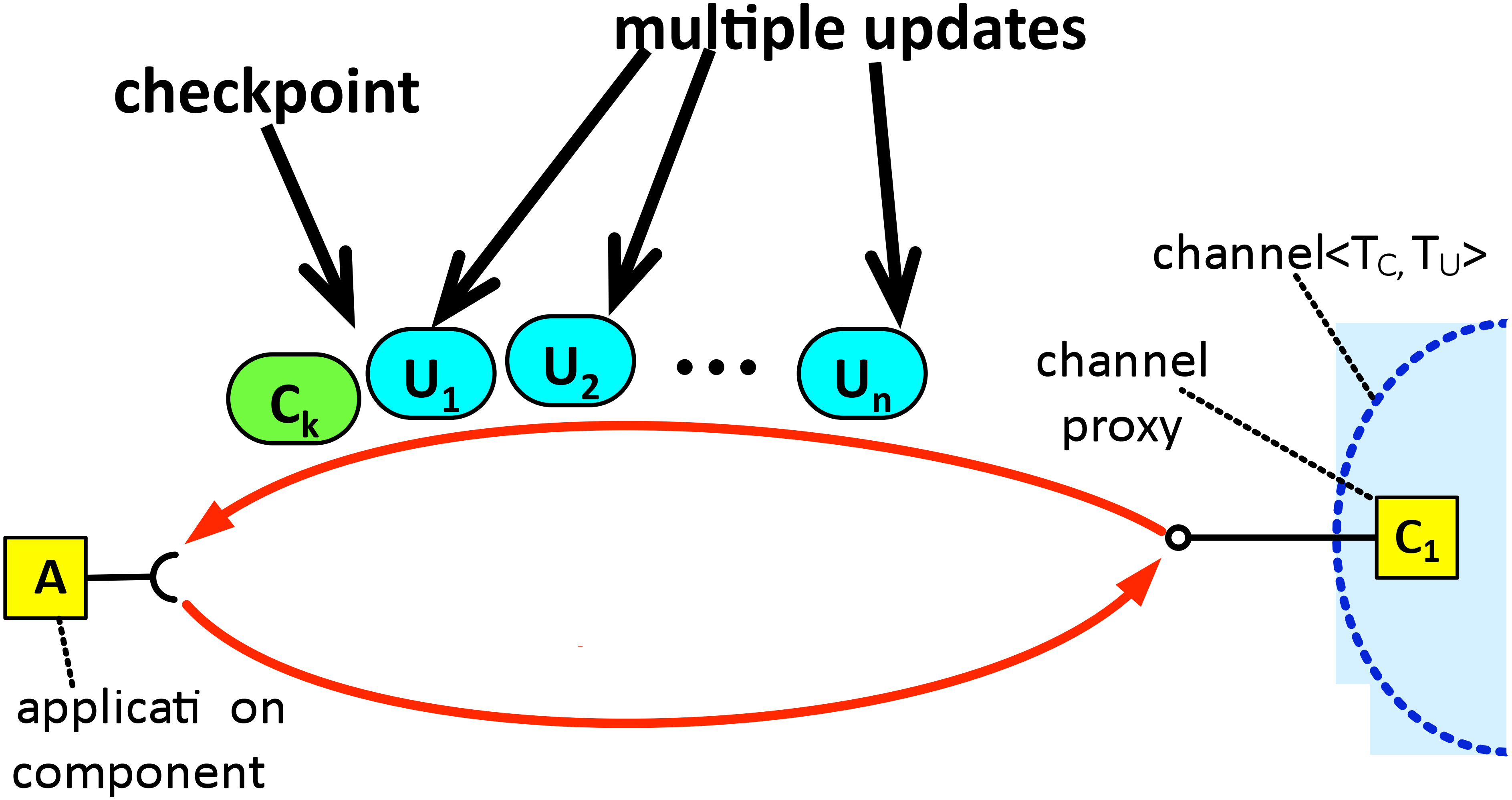
incremental update

new checkpoint or state

Semantics


C
+
U
=
C′
earlier
checkpoint
or state
incremental
update
new
checkpoint
or state

Semantics



checkpoint

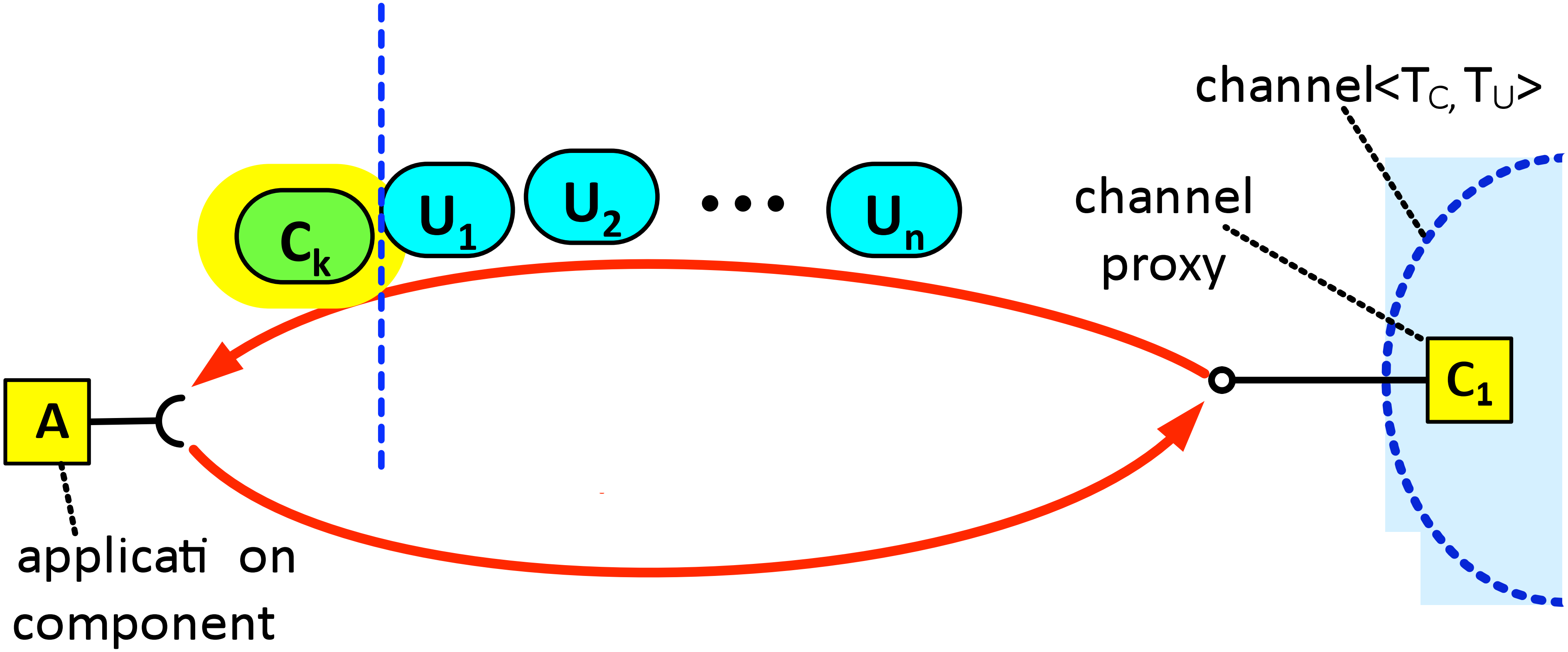
multiple updates

channel<T_C, T_U>

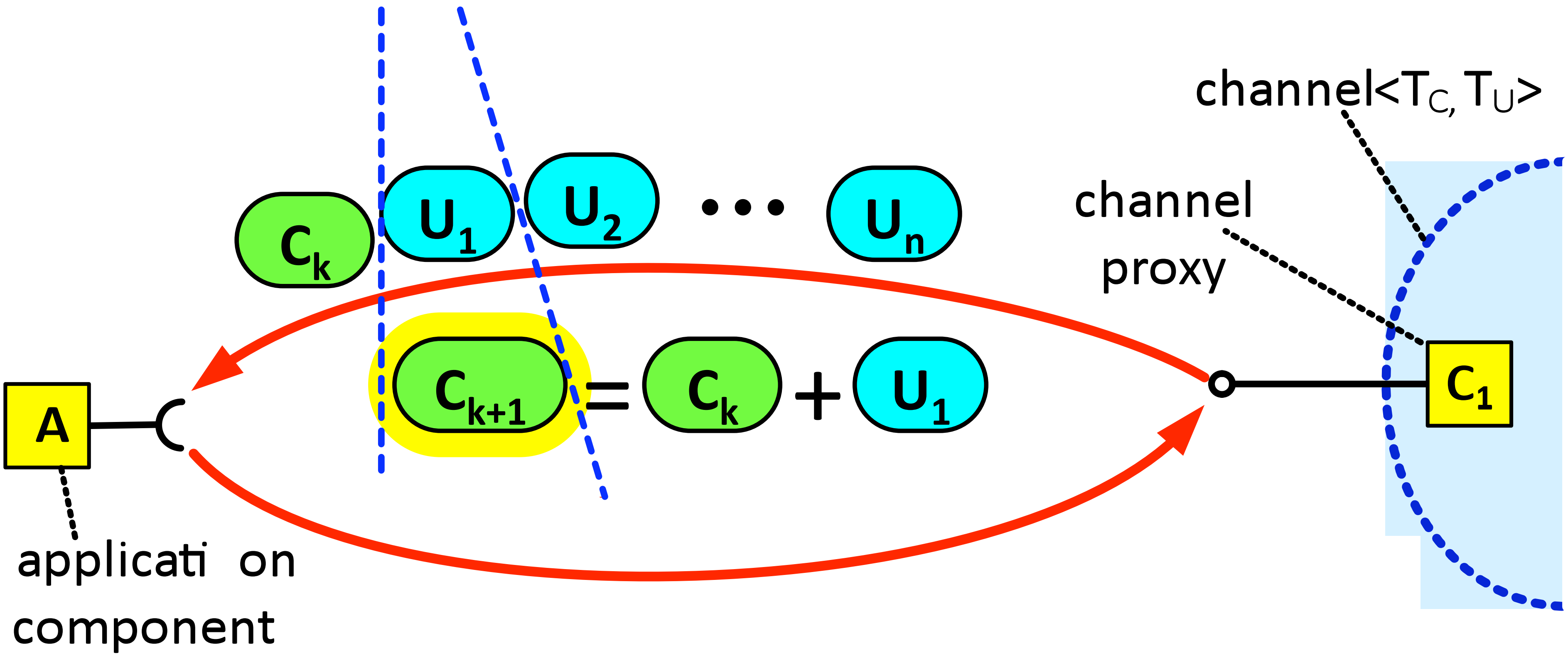
channel proxy

application component

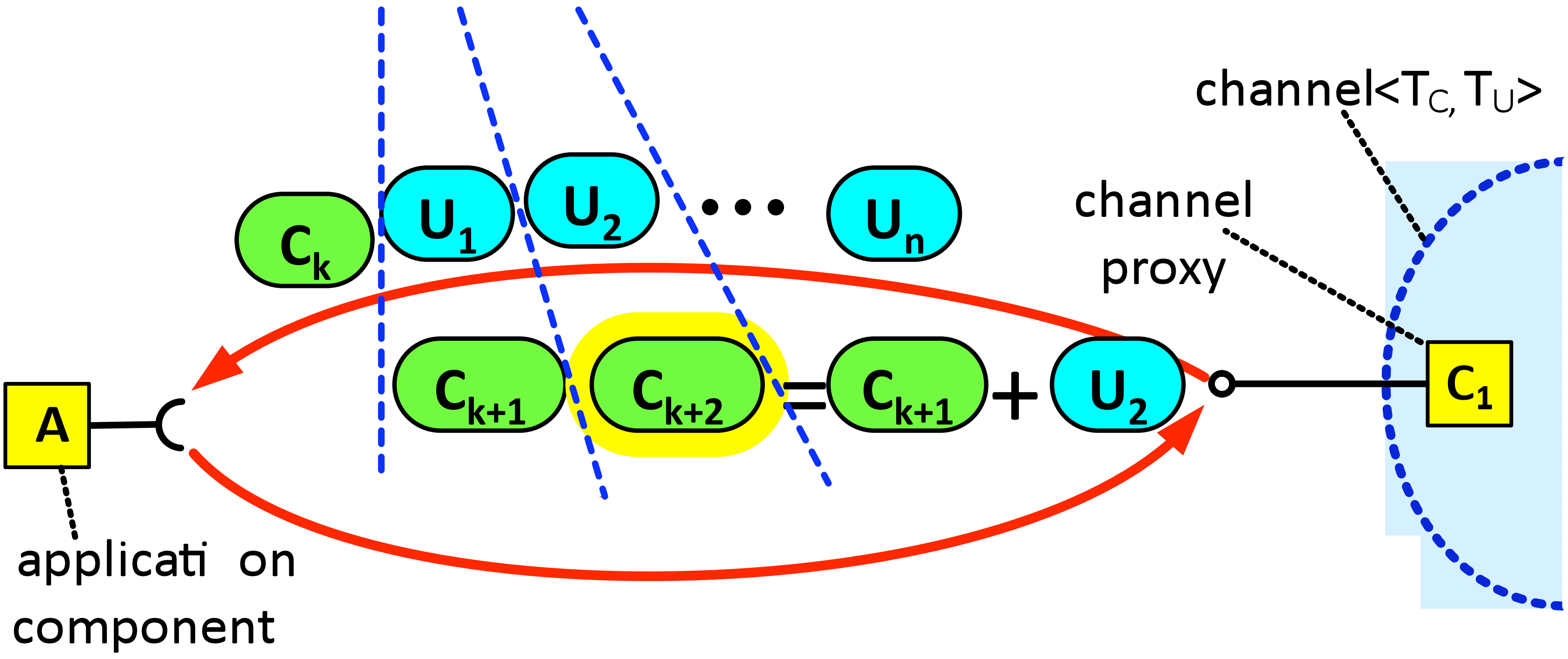
Semantics


channel<T_C, T_U>
channel
proxy
C_k
U_1
U_2
• • •
U_n
A
C_1
applicati on
component

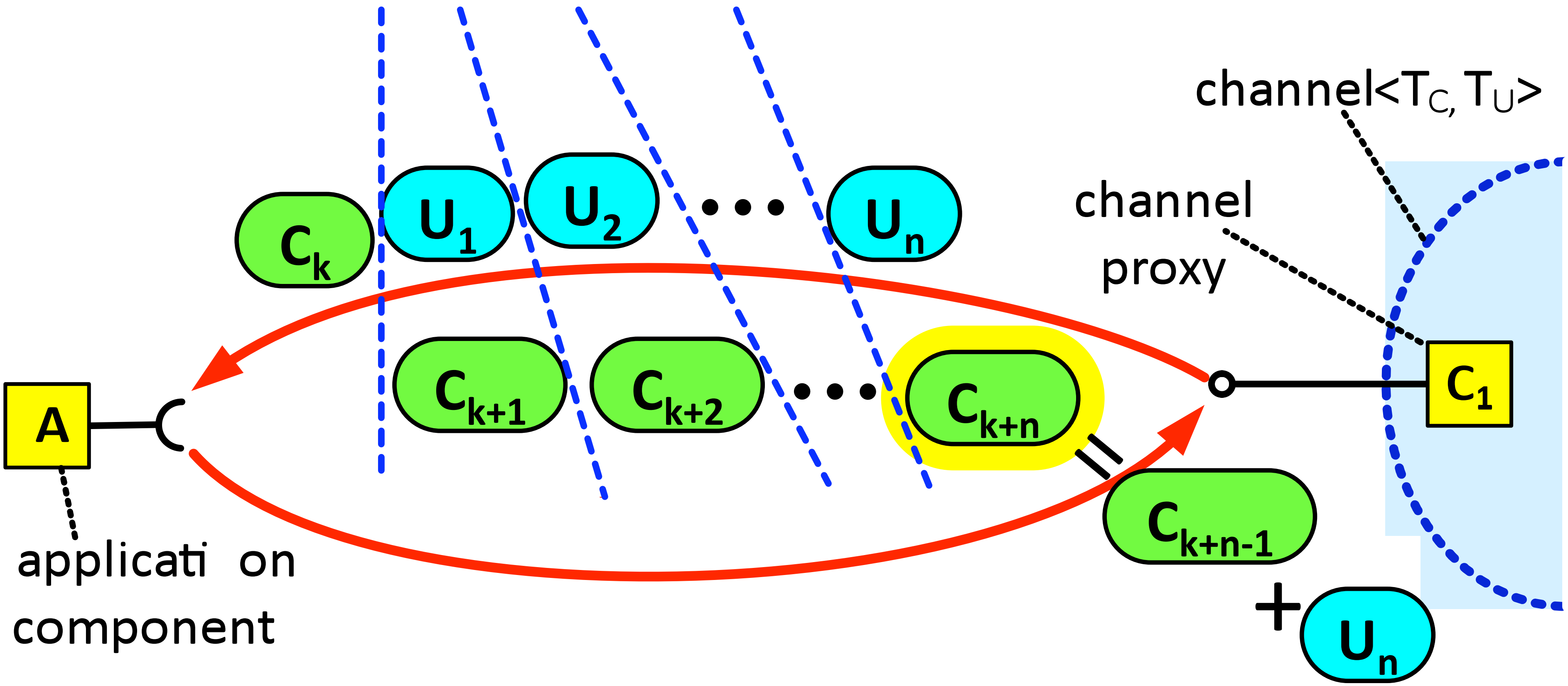
Semantics


channel<T_C, T_U>
channel proxy
C_1
A
applicati on component
C_k
U_1
U_2
•••
U_n
$C_{k+1} = C_k + U_1$

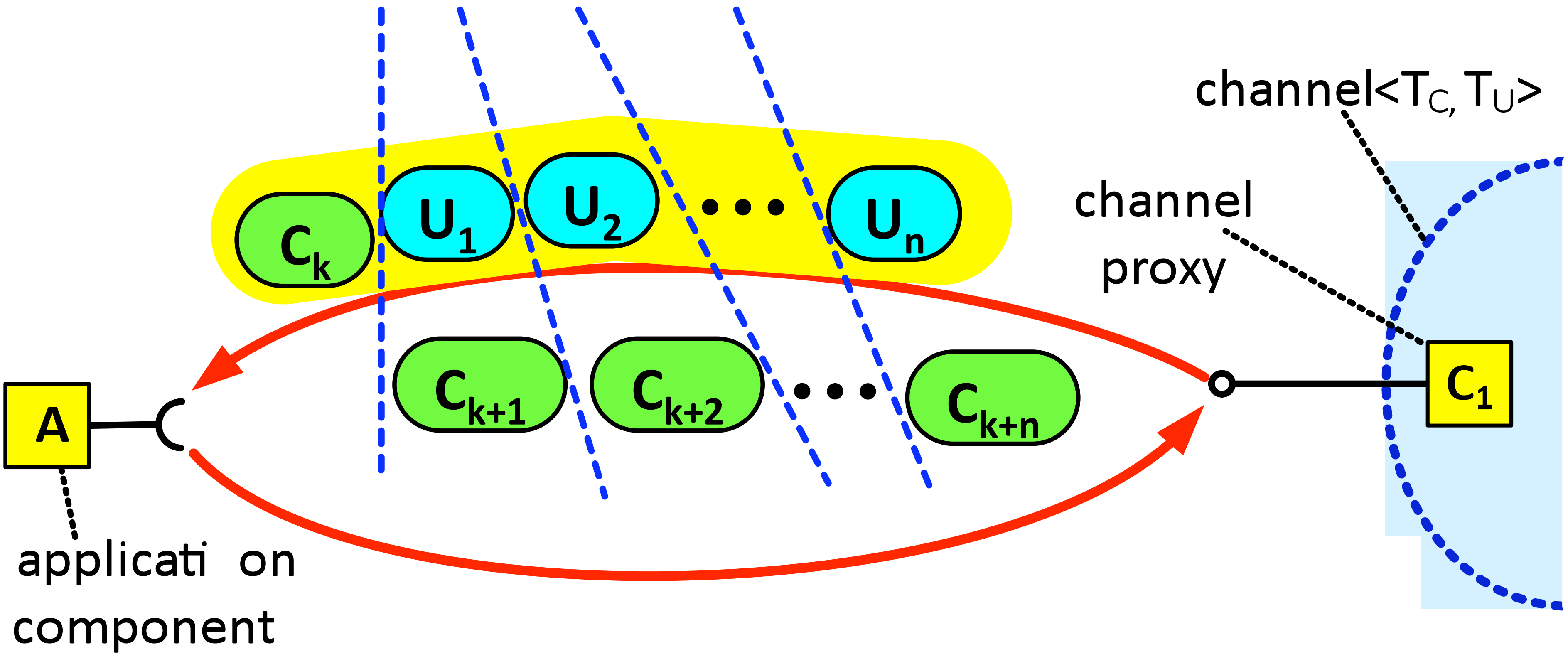
Semantics


channel<T_C, T_U>
channel
proxy
C_k
U_1
U_2
• • •
U_n
A
C_{k+1}
C_{k+2}
=
C_{k+1}
+
U_2
C_1
applicati on
component

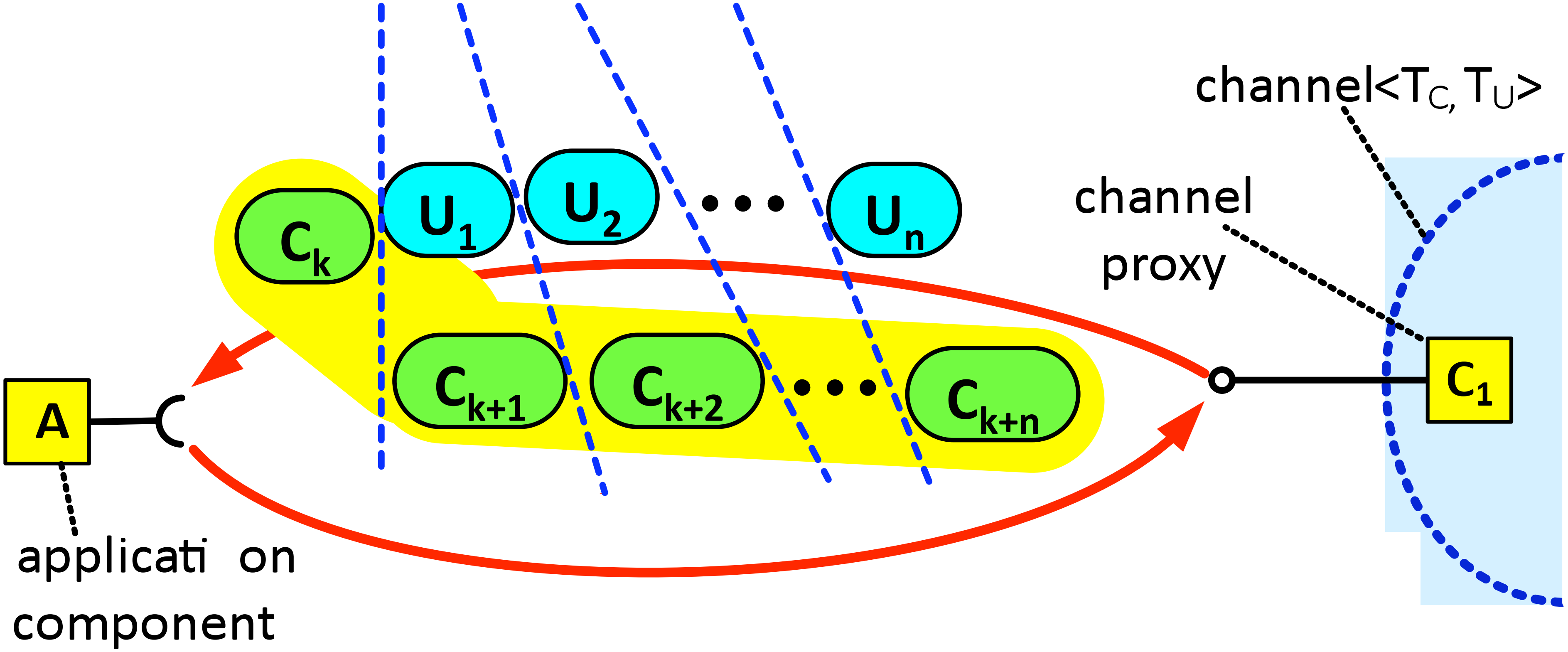
Semantics


channel<T_C, T_U>
channel
proxy
C_k
U_1
U_2
• • •
U_n
A
C_{k+1}
C_{k+2}
• • •
C_{k+n}
C_1
C_{k+n-1}
+
U_n
applicati on
component

Semantics

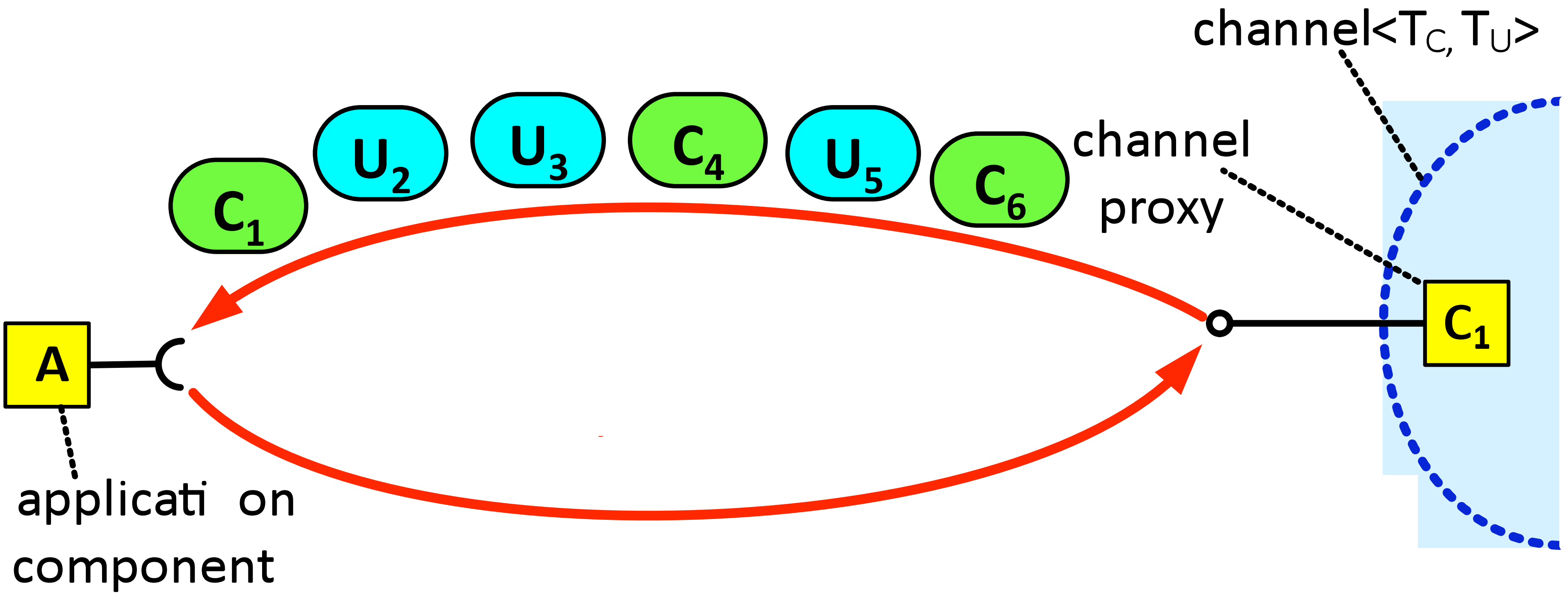

C_k
U_1
U_2
• • •
U_n
C_{k+1}
C_{k+2}
• • •
C_{k+n}
A
applicati on component
channel<T_C, T_U>
channel proxy
C_1

Semantics



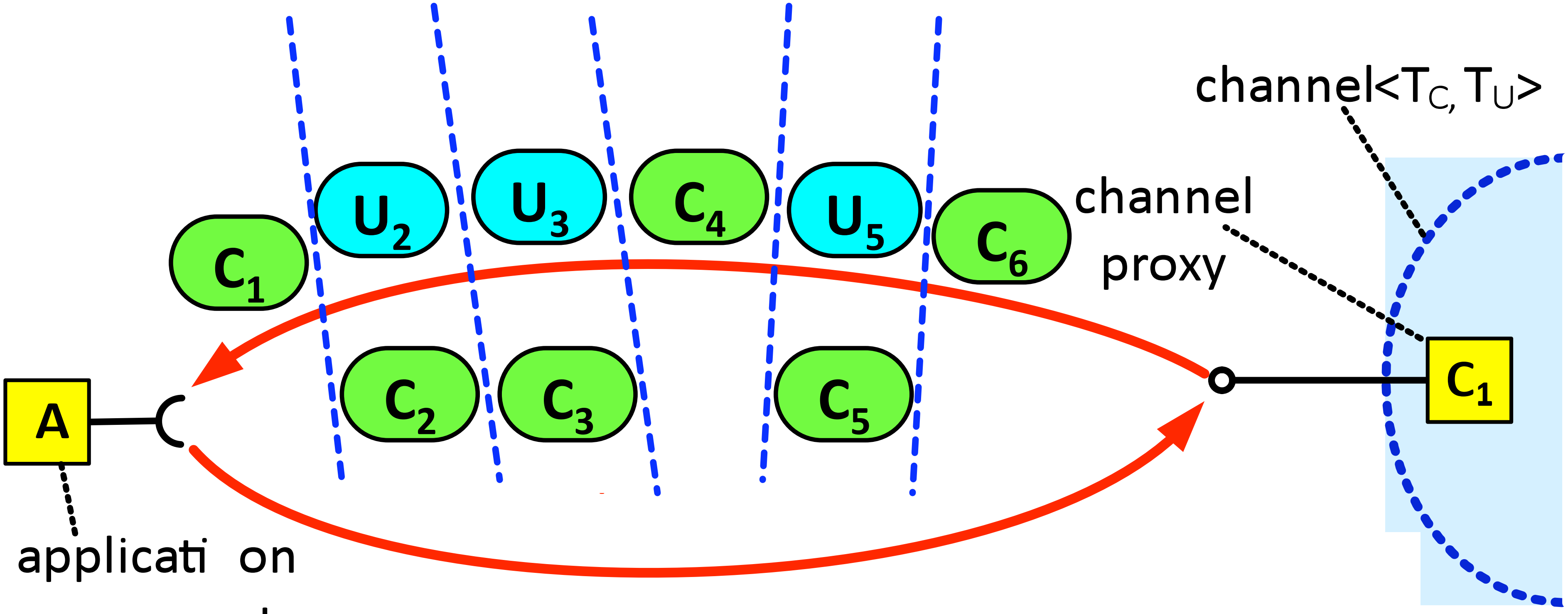
channel<T_C, T_U>
channel
proxy
U1
U2
• • •
Un
Ck
Ck+1
Ck+2
• • •
Ck+n
A
C1
applicati on
component

Semantics



channel<T_C, T_U>
C_1
U_2
U_3
C_4
U_5
C_6
channel proxy
A
C_1
application component

Semantics

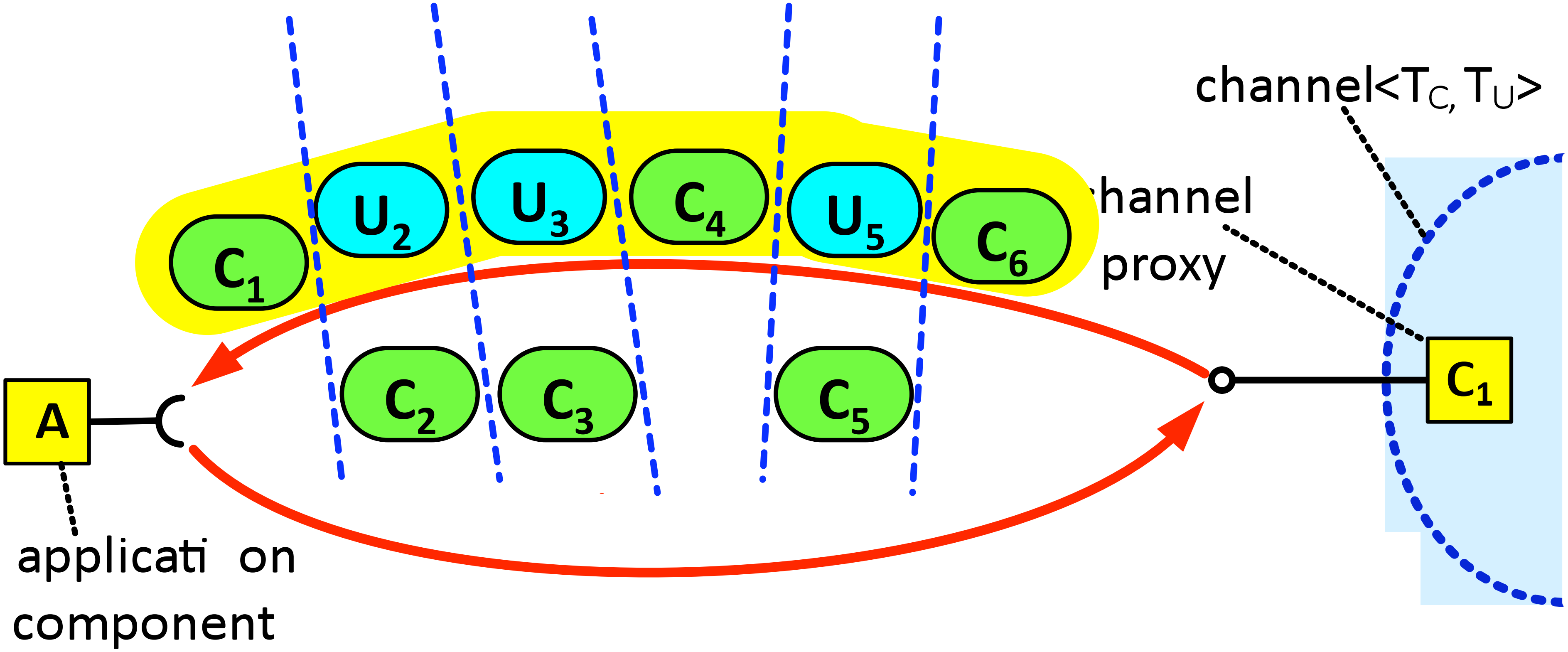


channel<T_C, T_U>

channel proxy

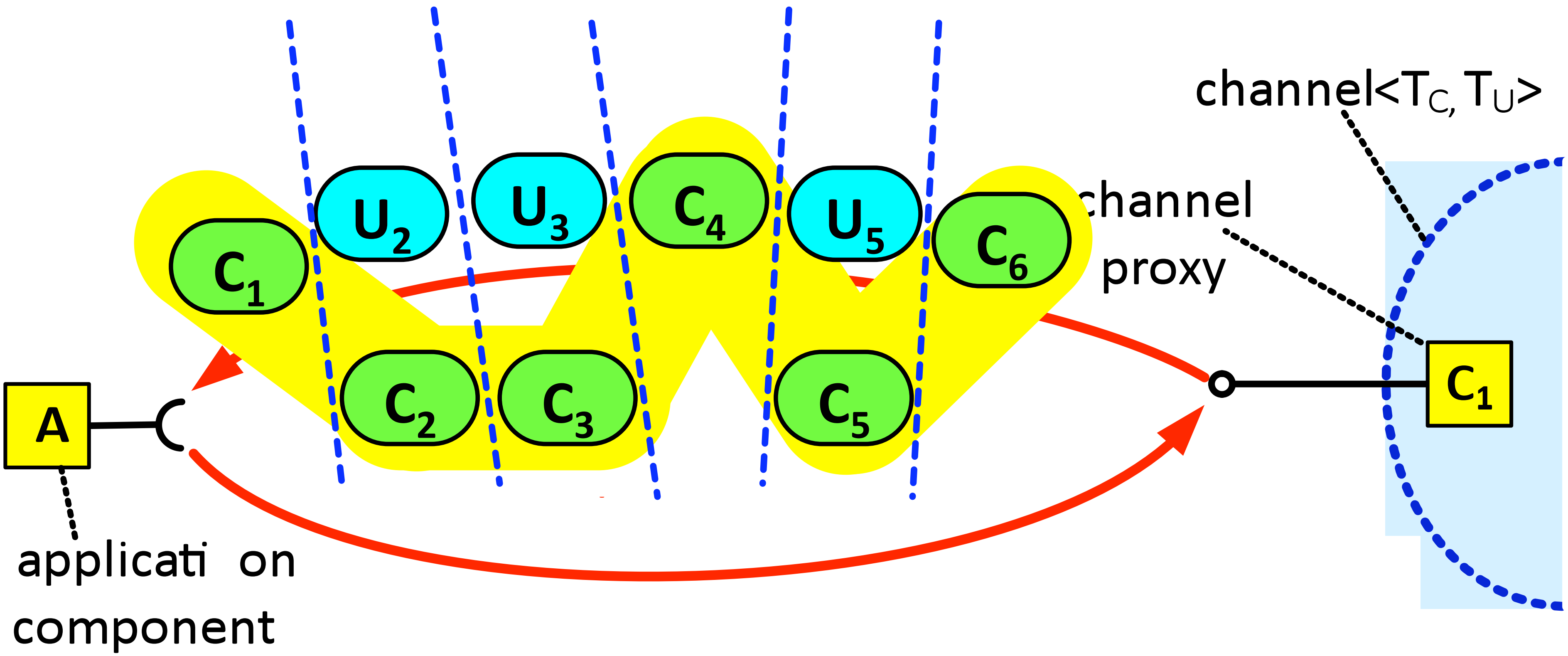
application component

Semantics



channel<T_C, T_U>
C_1
U_2
U_3
C_4
U_5
C_6
channel
proxy
C_2
C_3
C_5
A
C_1
application
component

Semantics

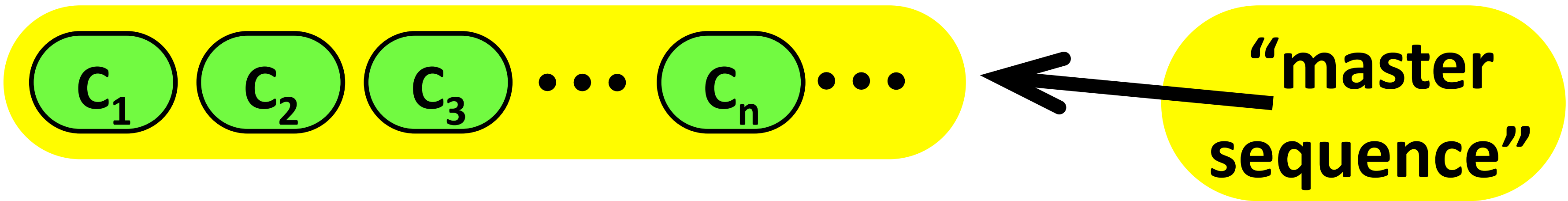


channel<T_C, T_U>

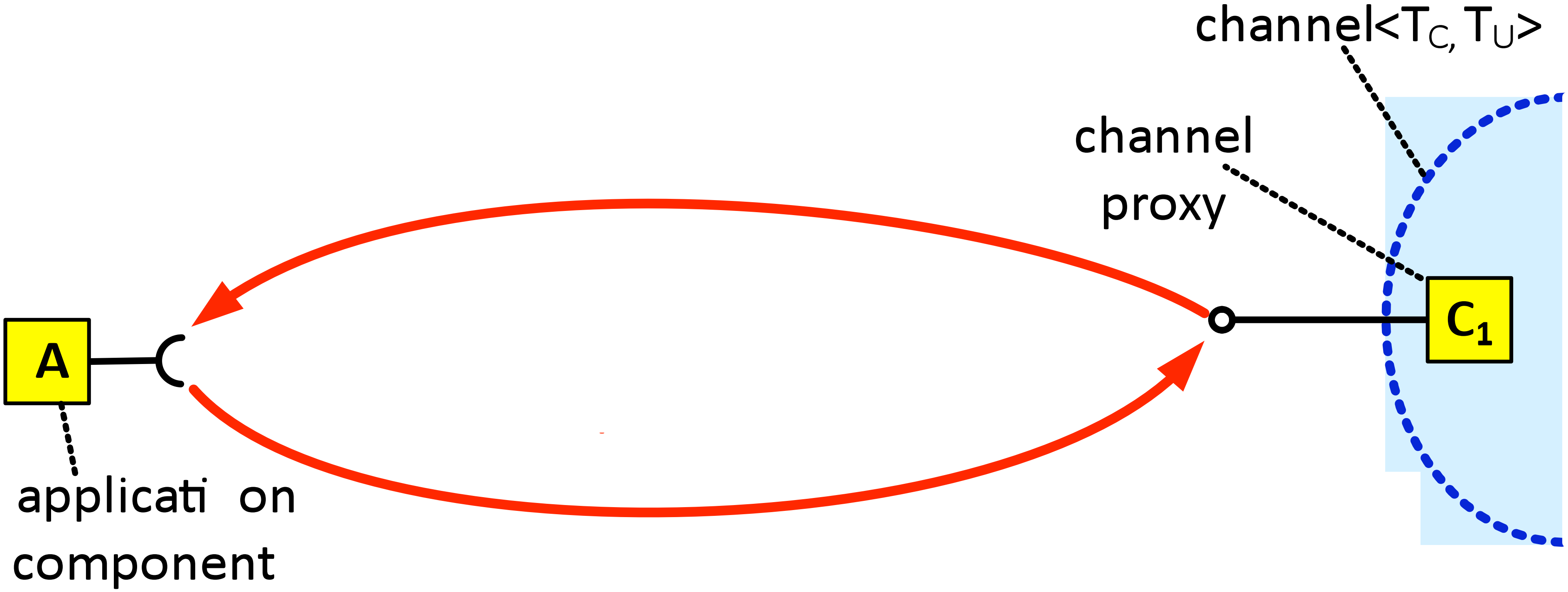
channel proxy

application component

Semantics



“master sequence”

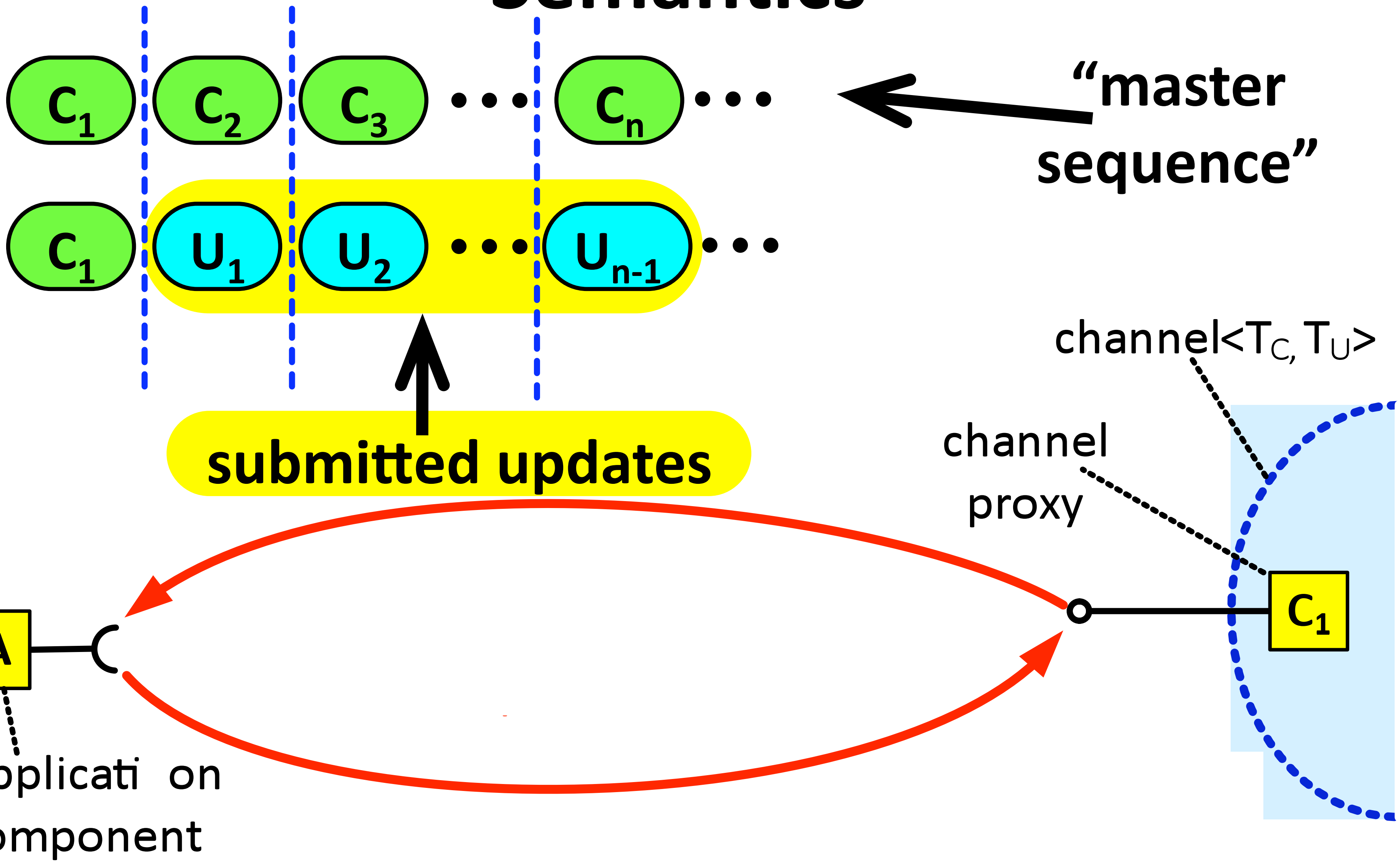


channel<T_C, T_U>

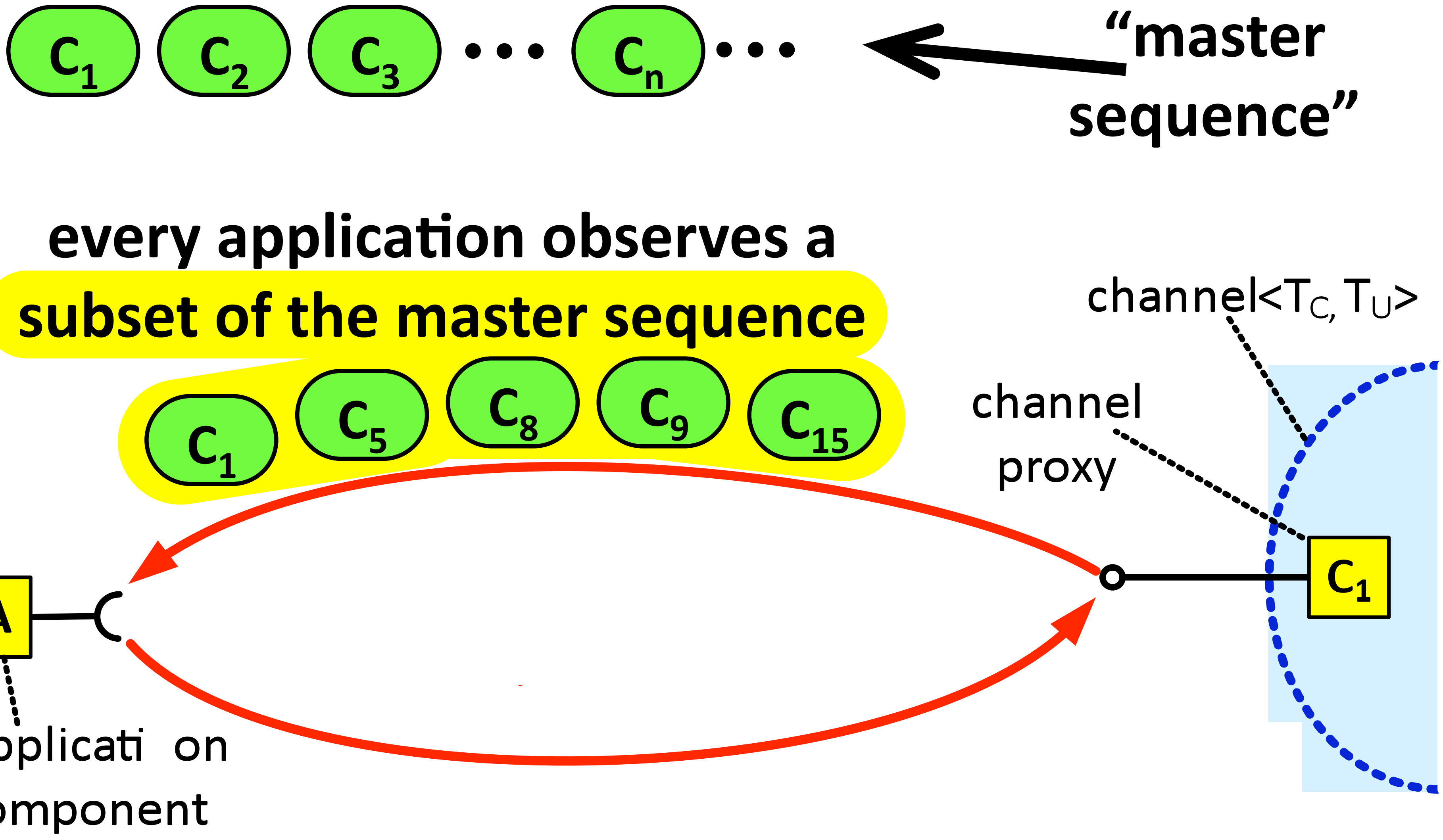
channel proxy

application component

Semantics


C_1 C_2 C_3 • • • C_n • • •
C_1 U_1 U_2 • • • U_{n-1} • • •
"master sequence"
submitted updates
channel<T_C, T_U>
channel proxy
C_1
A
application component

Semantics



“master sequence”

every application observes a **subset of the master sequence**

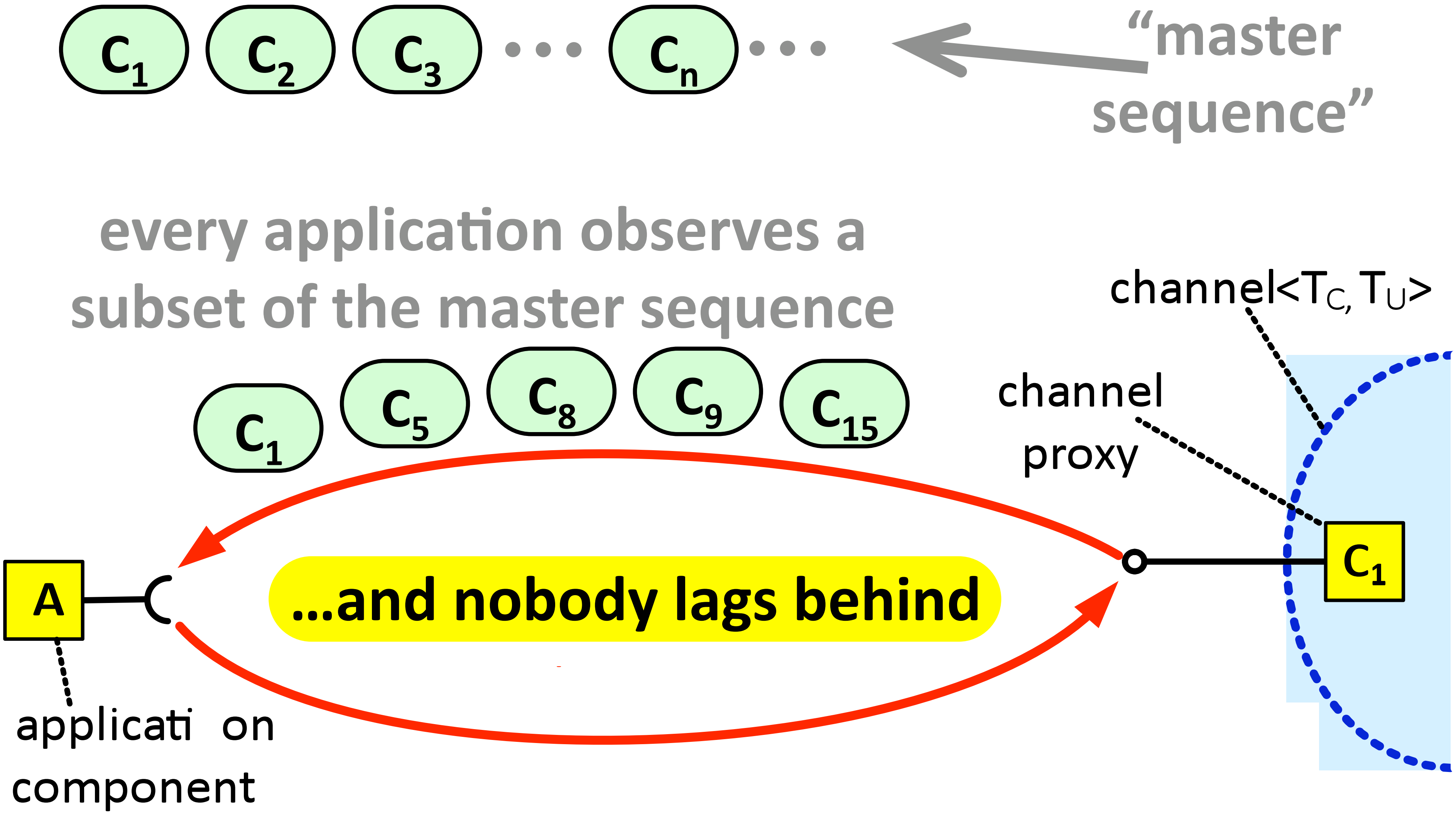
channel$<T_C, T_U>$

channel proxy

application component

C_1 C_2 C_3 • • • C_n • • •
C_1 C_5 C_8 C_9 C_{15}
A
C_1

Semantics



“master sequence”

every application observes a subset of the master sequence

...and nobody lags behind

Types

Checkpointed Communication Channel (CC) can be classified based on:

- the type of checkpoints,
- the type of updates,
- (and many other factors we won’t dicuss)

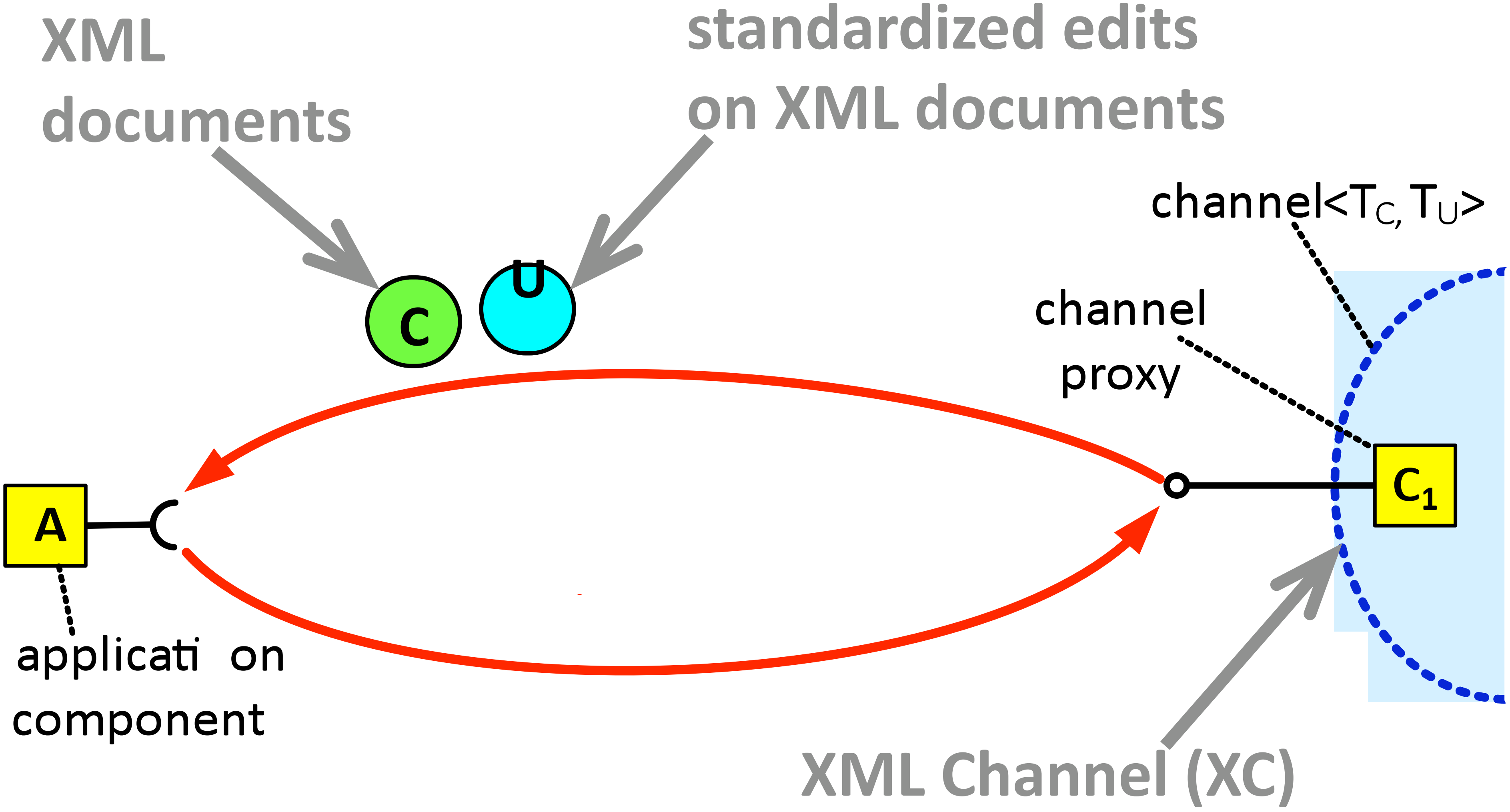
Types

Checkpointed Communication Channel (CC)

can be classified based on:

- the type of checkpoints,
- the type of updates,
- (and many other factors we won't dicuss)

Types



XML documents

standardized edits on XML documents

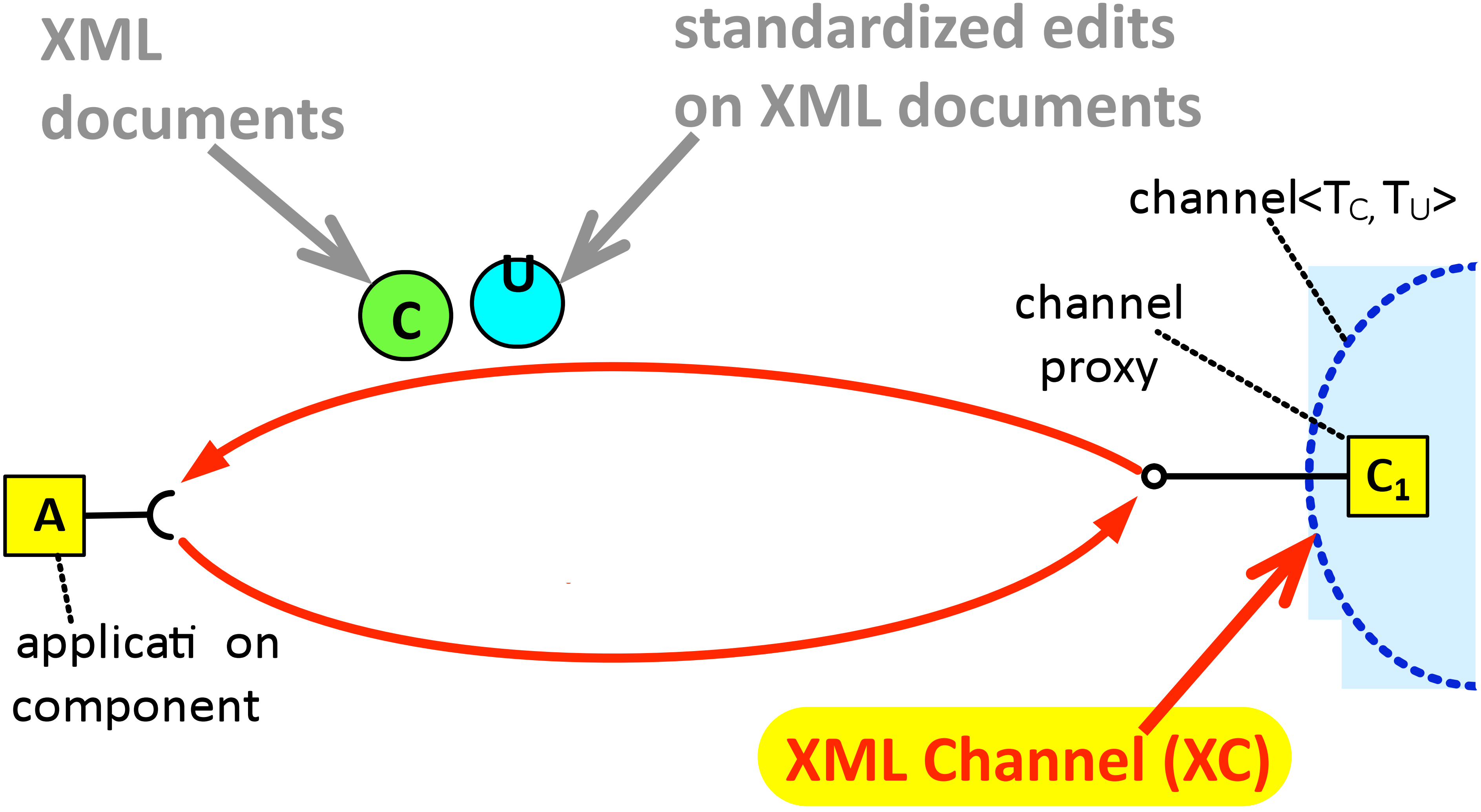
channel<T_C, T_U>

channel proxy

application component

XML Channel (XC)

Types



XML documents

standardized edits on XML documents

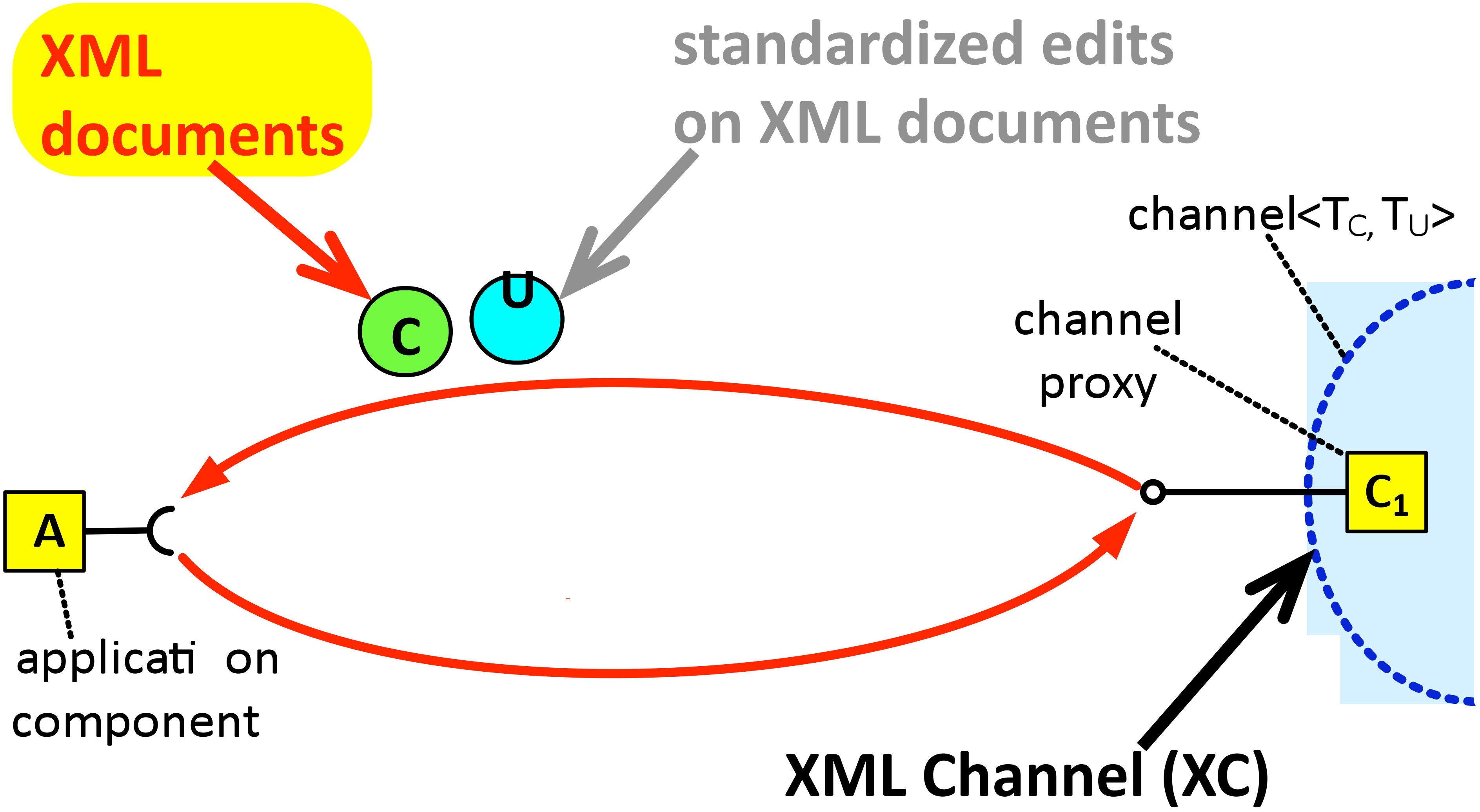
channel$<T_C, T_U>$

channel proxy

application component

XML Channel (XC)

Types



XML documents

standardized edits on XML documents

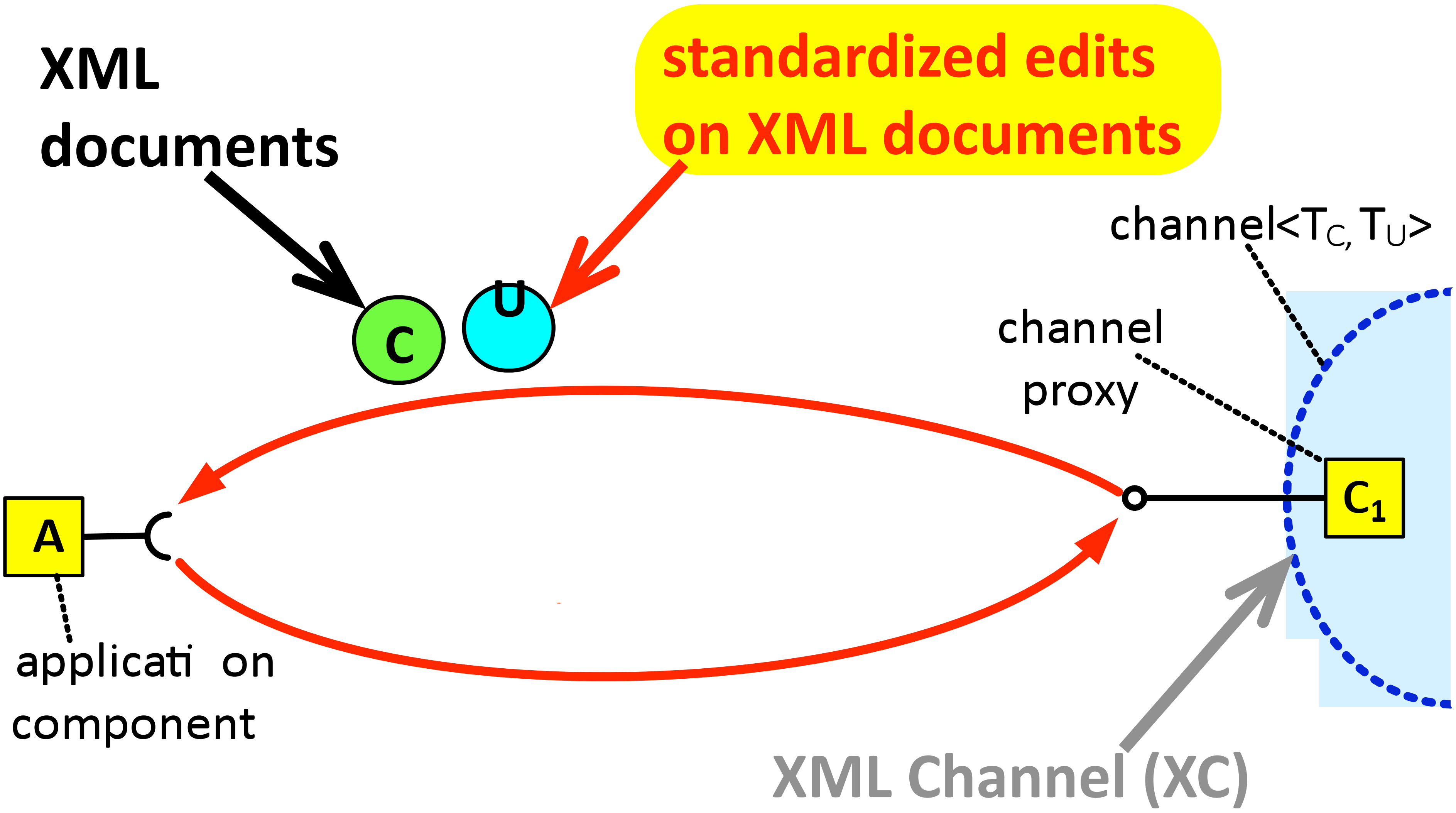
channel$<T_C, T_U>$

channel proxy

application component

XML Channel (XC)

Types



XML documents

standardized edits on XML documents

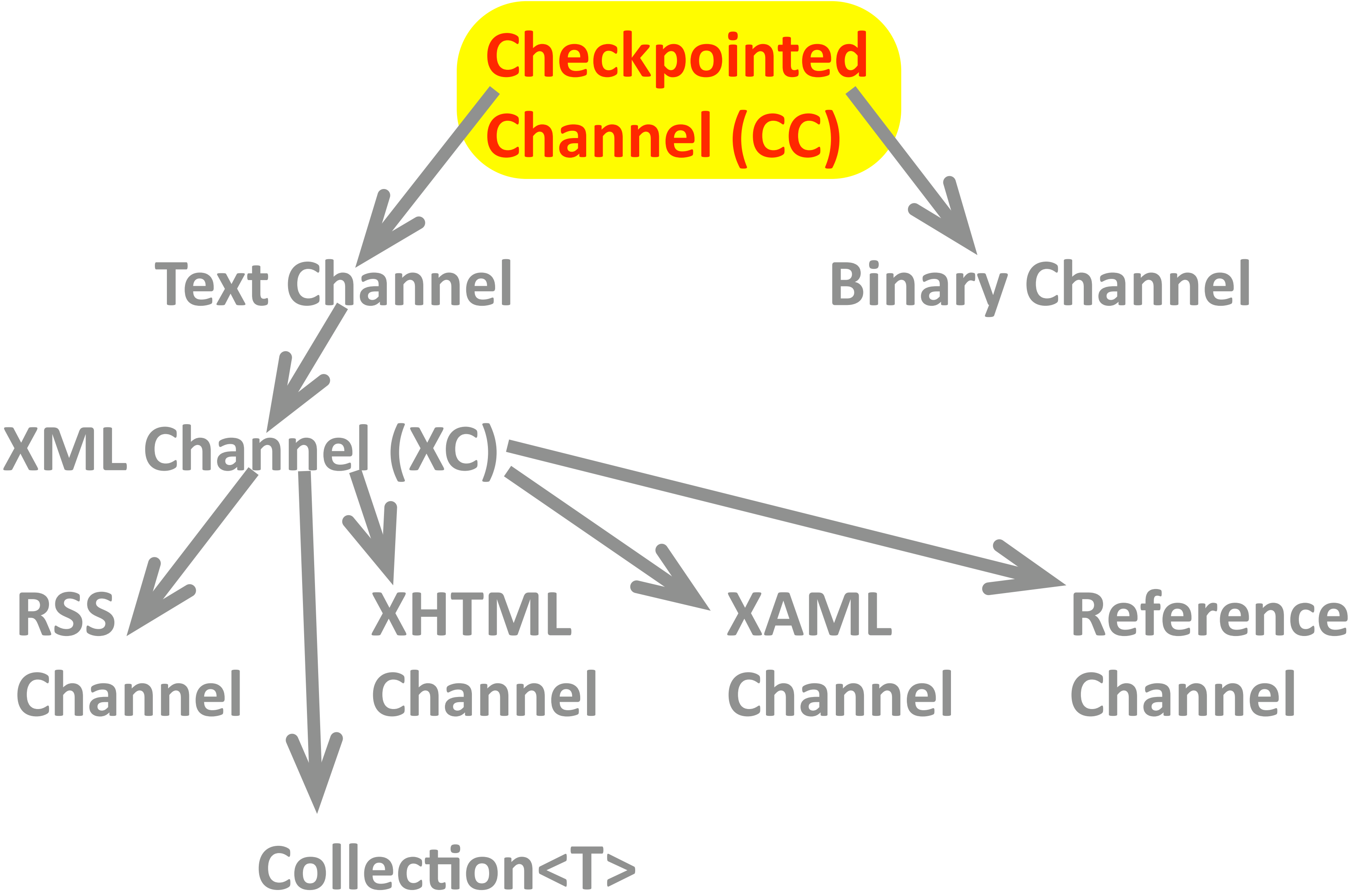
channel<T_C, T_U>

channel proxy

application component

XML Channel (XC)

Types



Checkpointed Channel (CC)

Text Channel

Binary Channel

XML Channel (XC)

RSS Channel

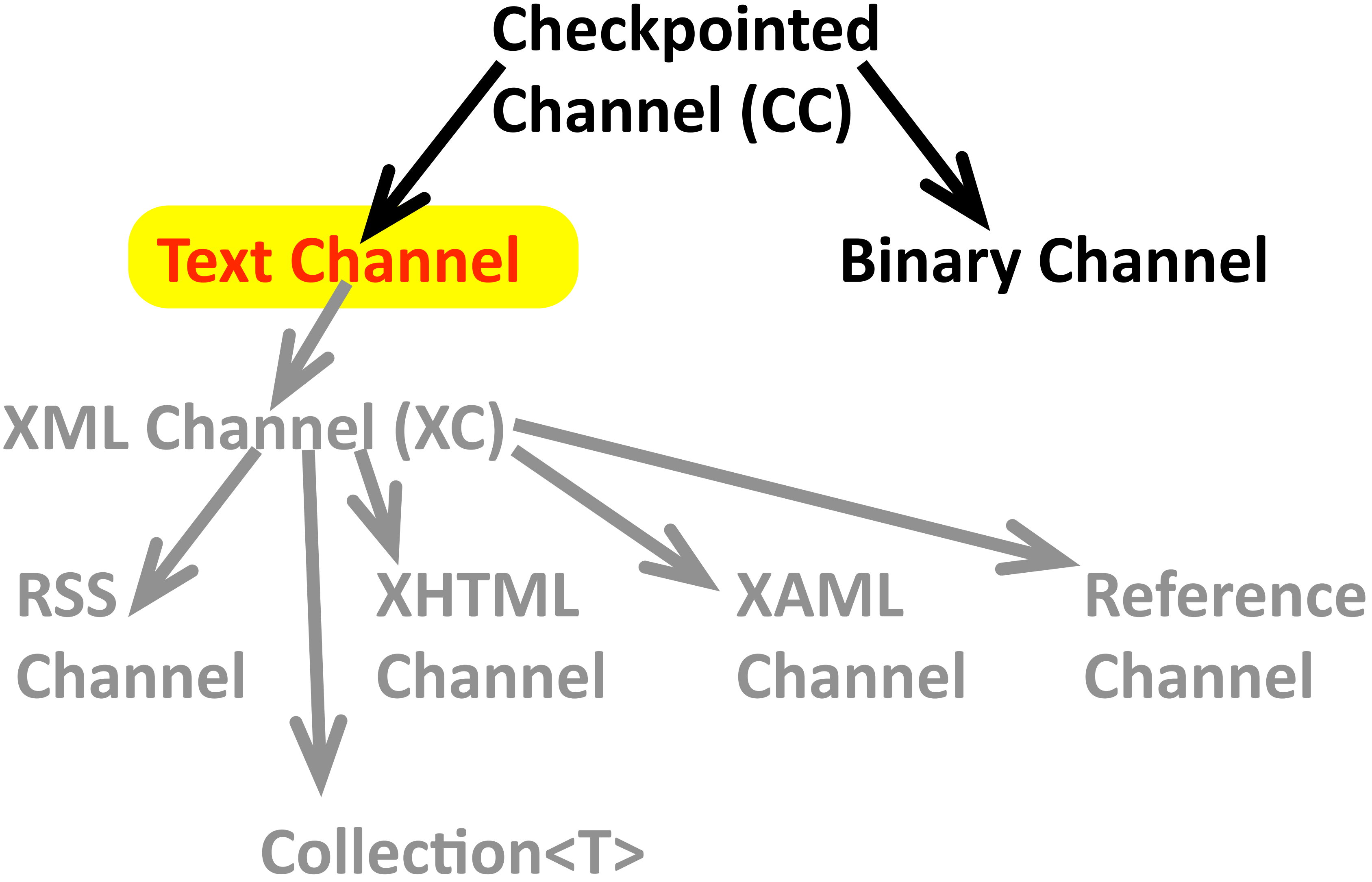
XHTML Channel

XAML Channel

Reference Channel

Collection<T>

Types



Checkpointed Channel (CC)

Text Channel

Binary Channel

XML Channel (XC)

RSS Channel

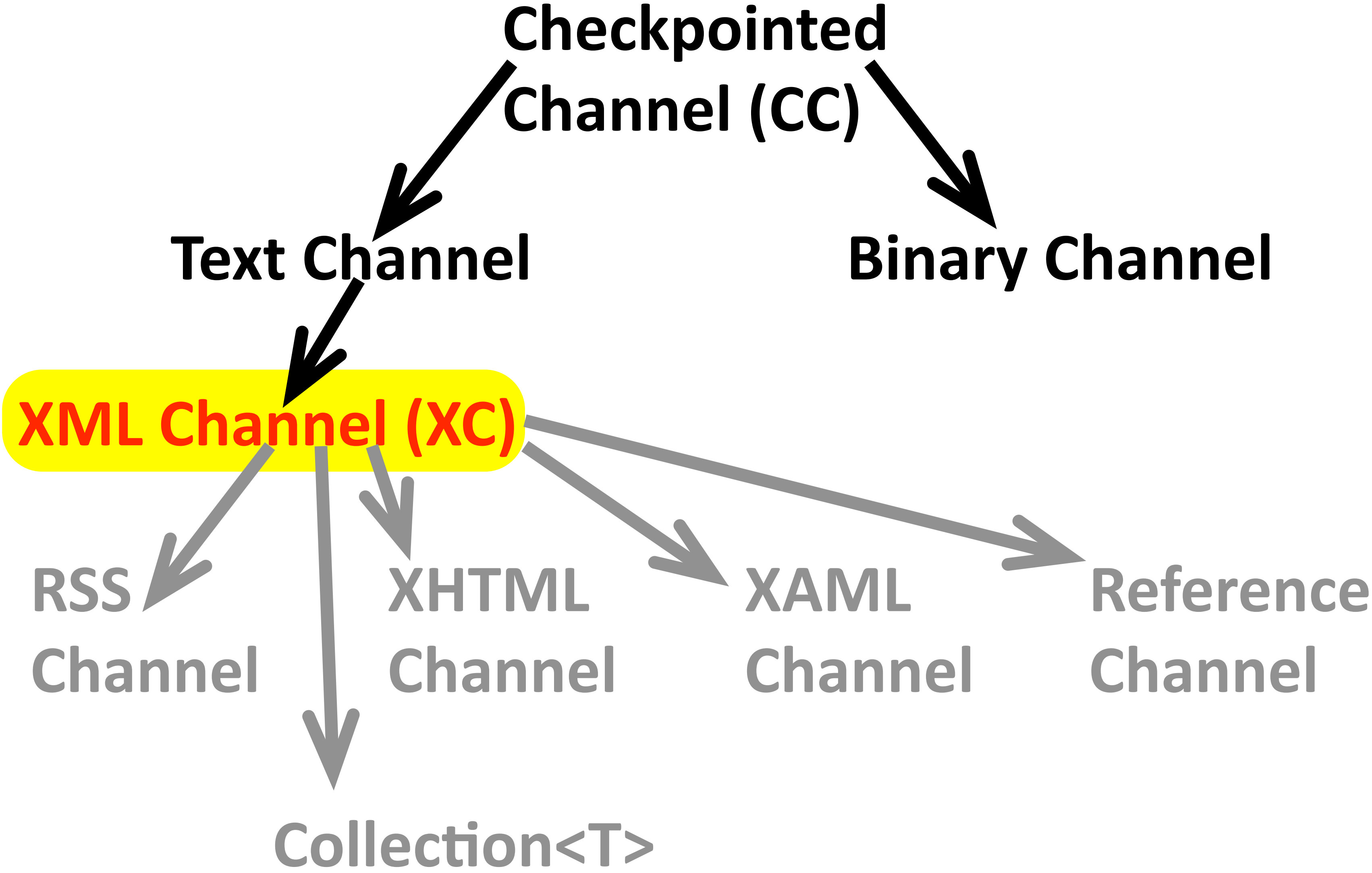
XHTML Channel

XAML Channel

Reference Channel

Collection<T>

Types



Checkpointed Channel (CC)

Text Channel

Binary Channel

XML Channel (XC)

RSS Channel

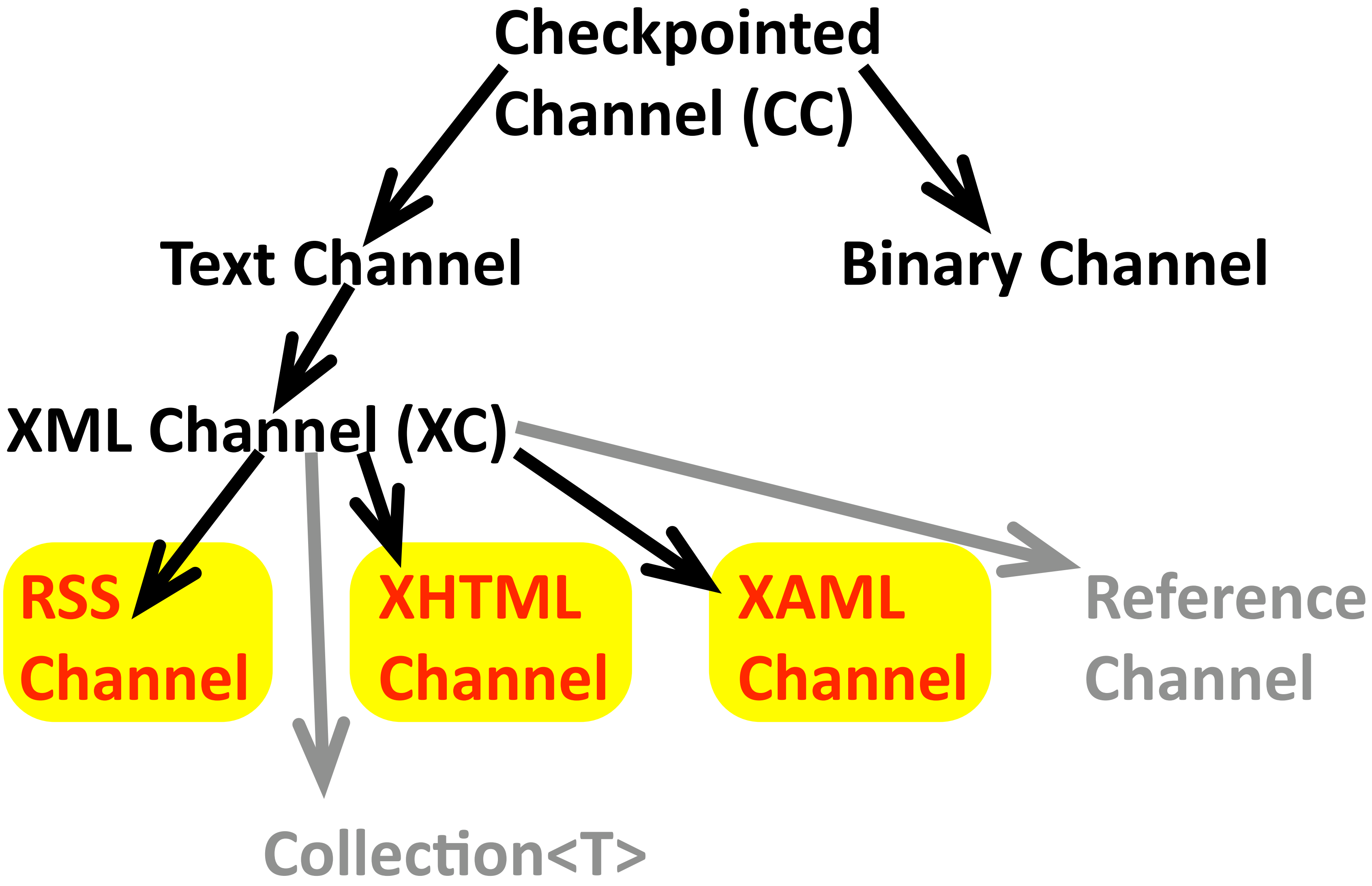
XHTML Channel

XAML Channel

Reference Channel

Collection<T>

Types



Checkpointed Channel (CC)

Text Channel

Binary Channel

XML Channel (XC)

RSS Channel

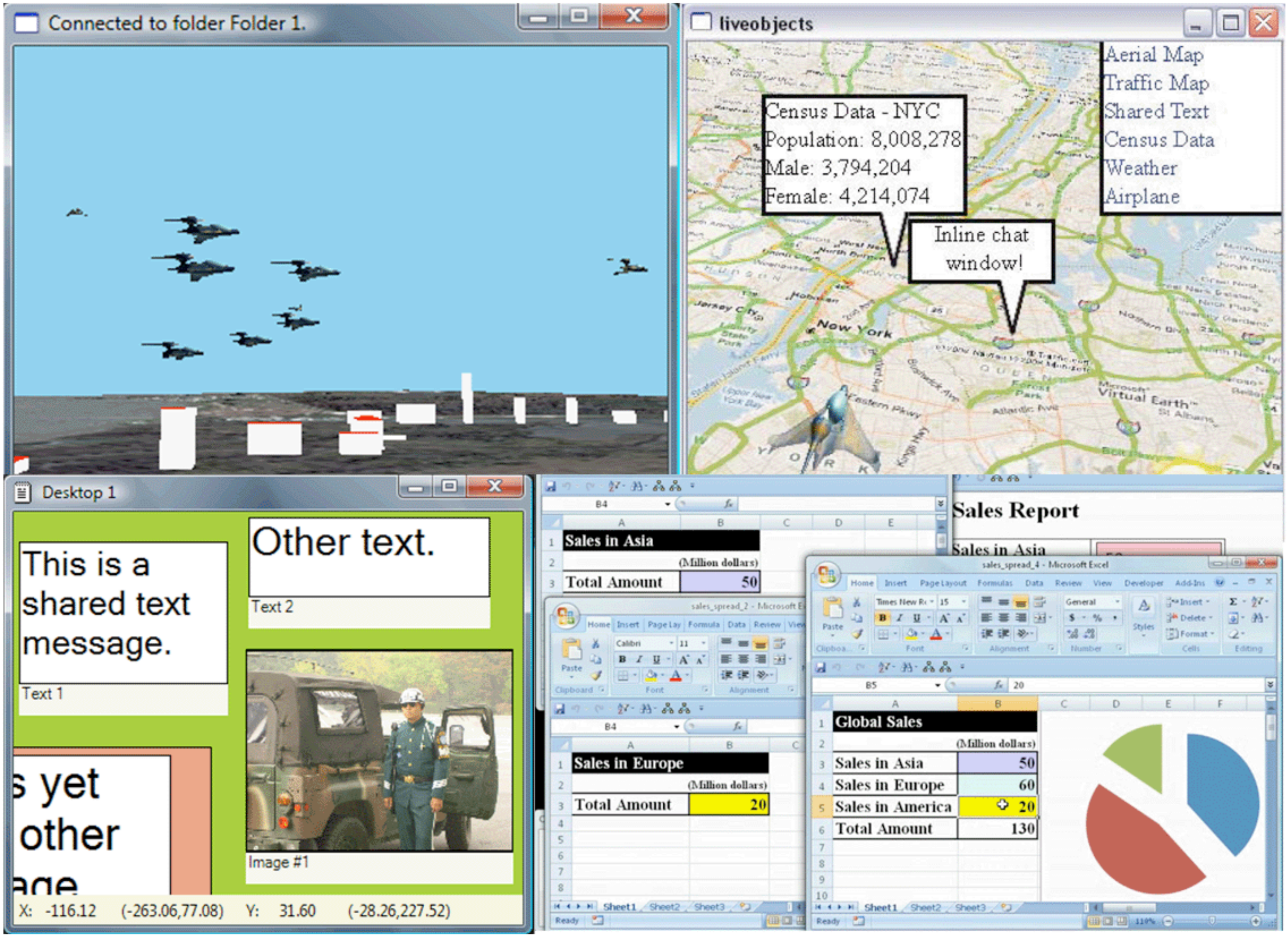
XHTML Channel

XAML Channel

Reference Channel

Collection<T>

http://liveobjects.cs.cornell.edu



Census Data - NYC
Population: 8,008,278
Male: 3,794,204
Female: 4,214,074

Inline chat window!

This is a shared text message.

Text 1

Other text.

Text 2

s yet other age

Image #1

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)

	A	B
1	Sales in Asia	
2		(Million dollars)
3	Total Amount	50

Sales Report

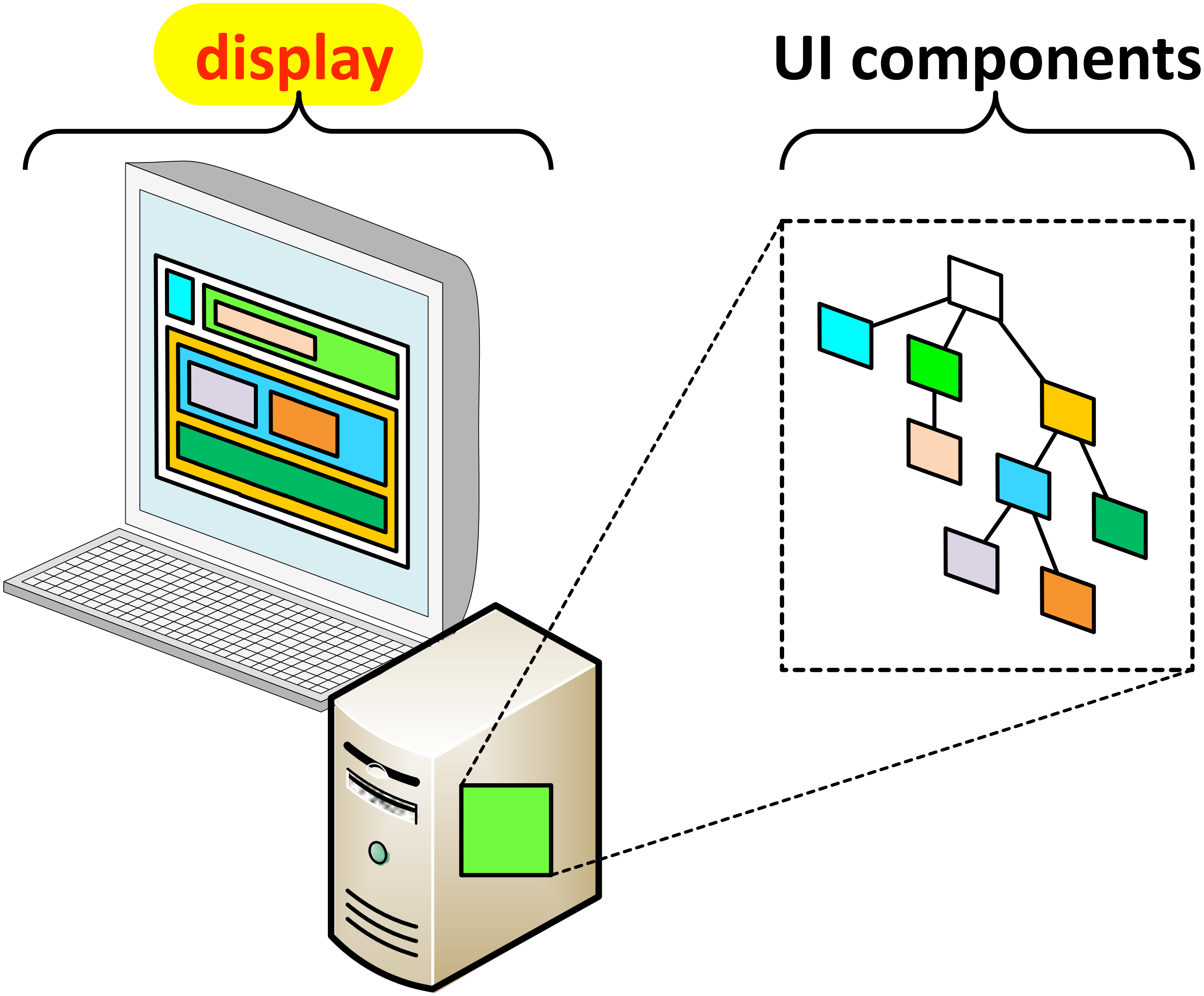
Sales in Asia

	A	B
1	Sales in Europe	
2		(Million dollars)
3	Total Amount	20

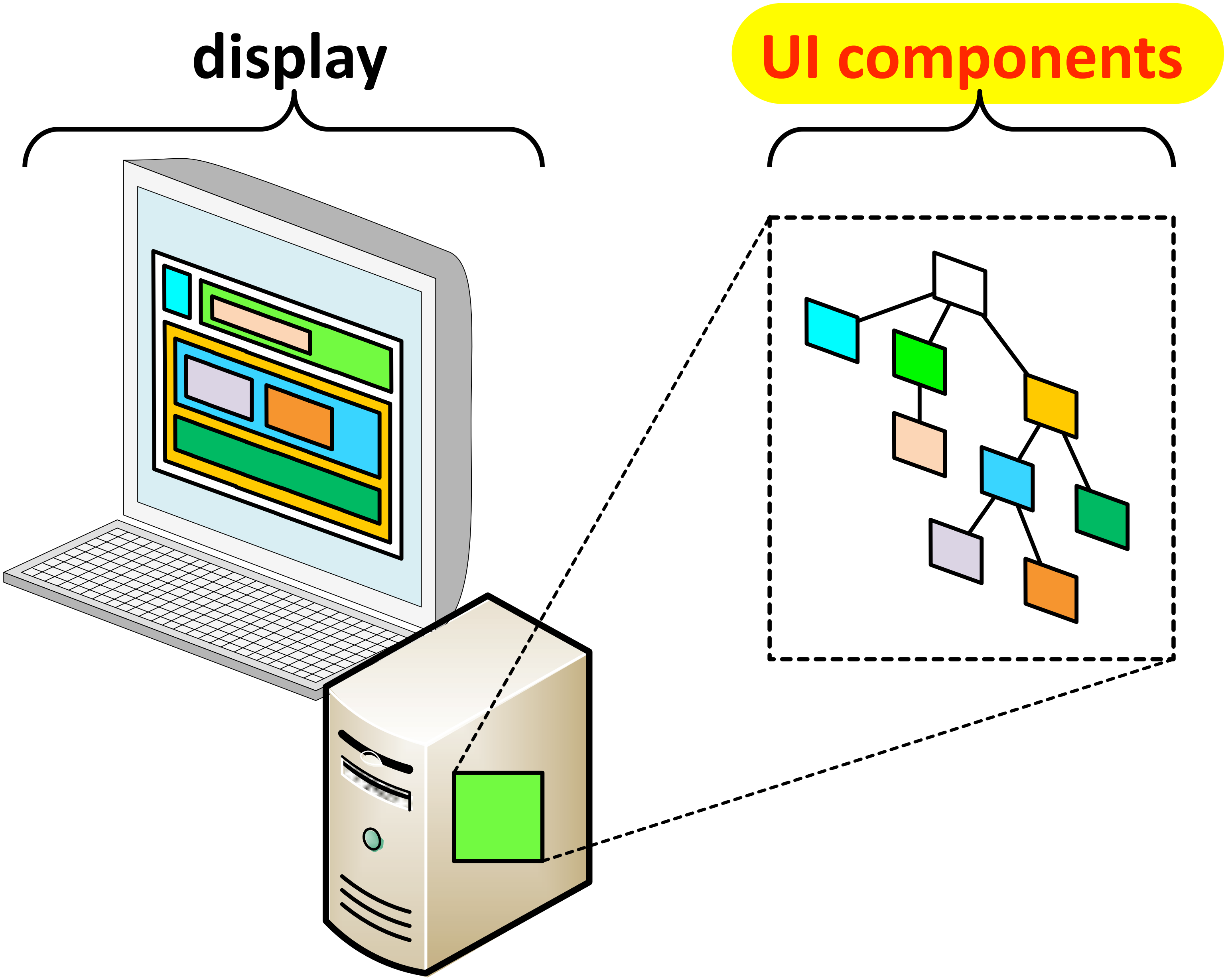
	A	B
1	Global Sales	
2		(Million dollars)
3	Sales in Asia	50
4	Sales in Europe	60
5	Sales in America	20
6	Total Amount	130

Live Distributed Objects (LO)

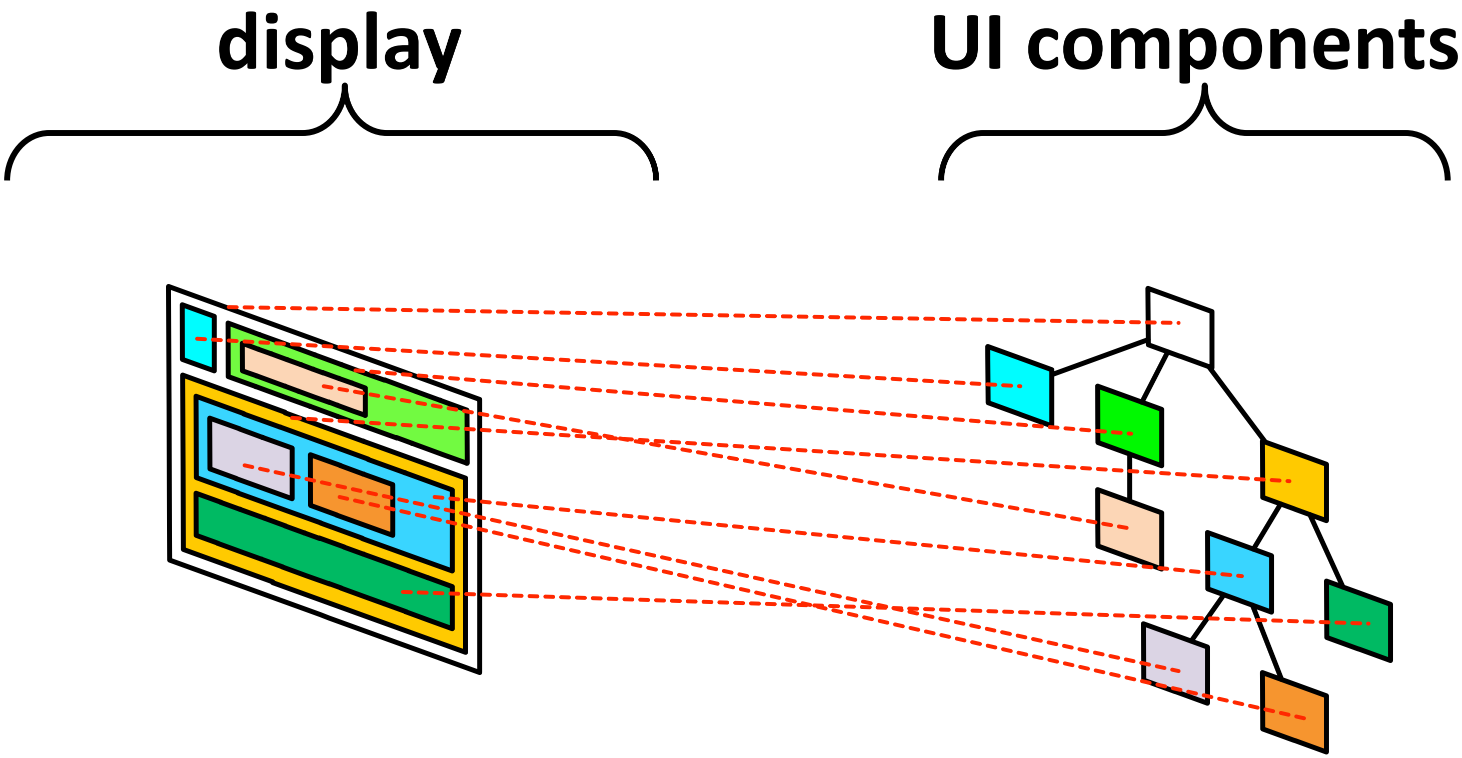
Ordinary Web Applications



Ordinary Web Applications



Ordinary Web Applications



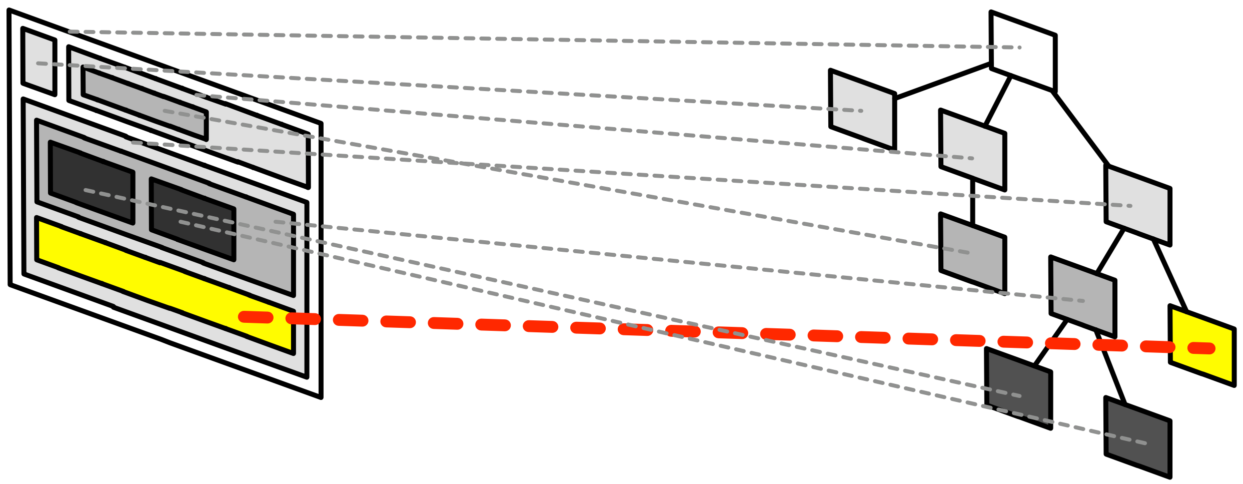
display

UI components

Ordinary Web Applications

display

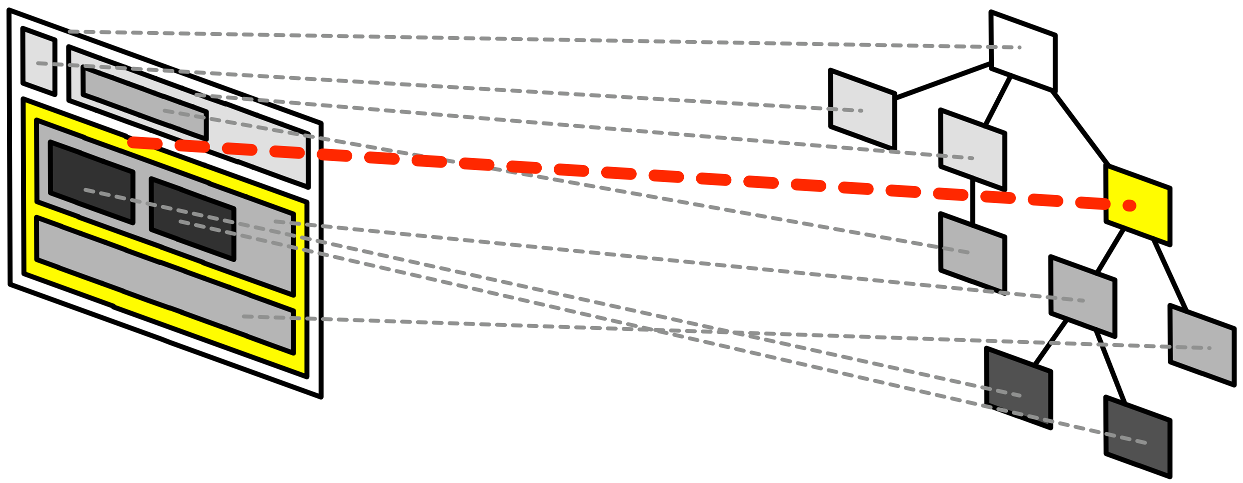
UI components



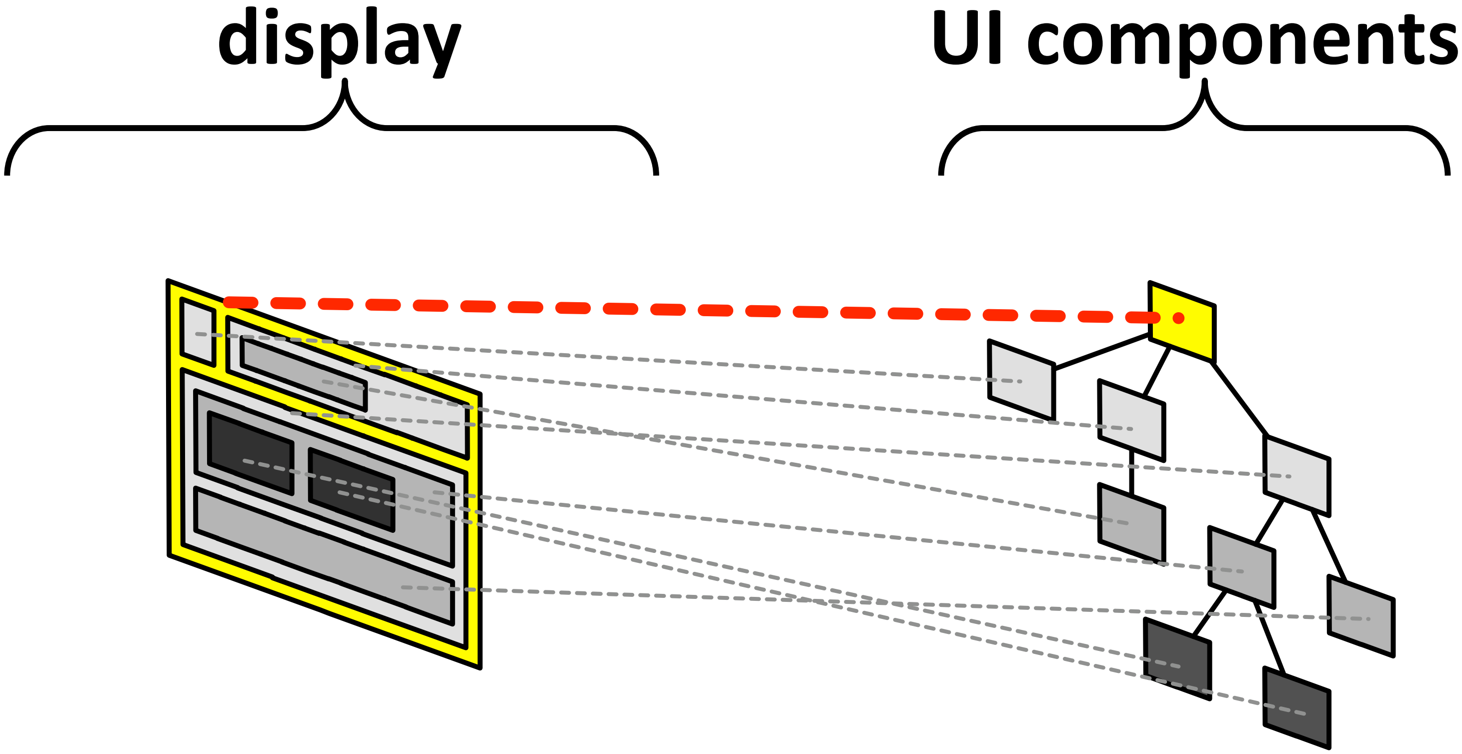
Ordinary Web Applications

display

UI components



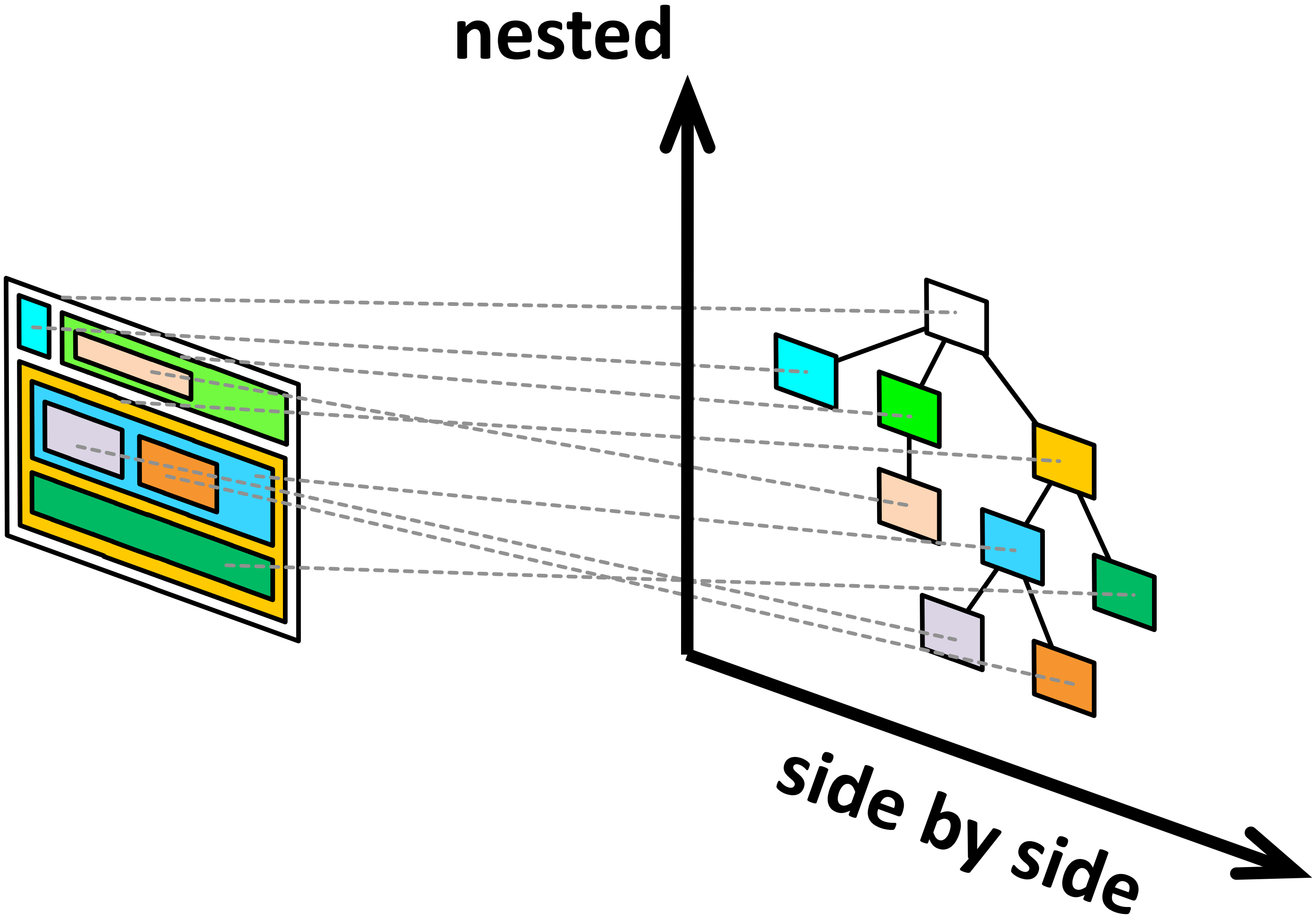
Ordinary Web Applications



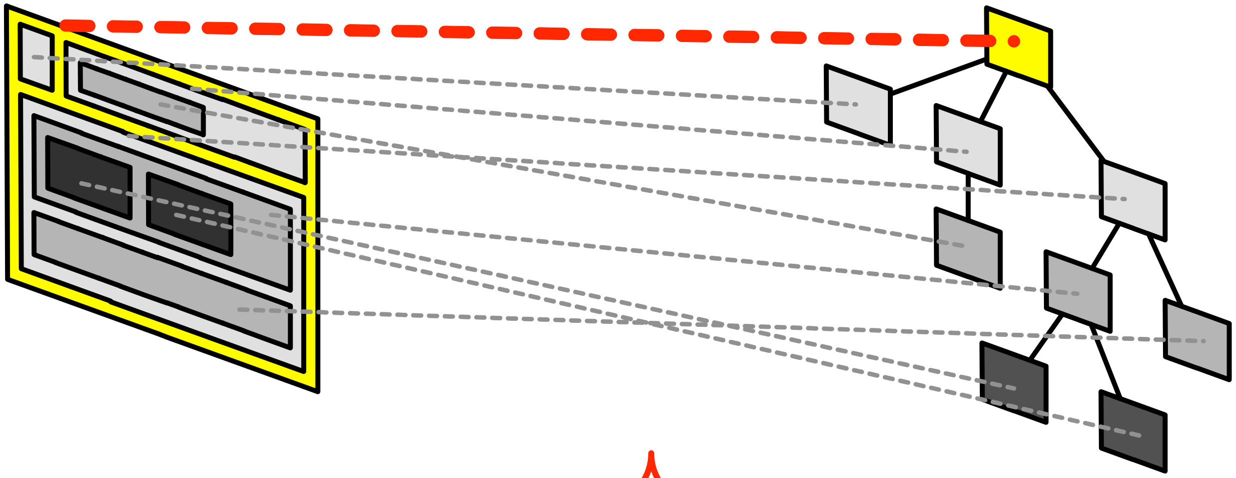
display

UI components

Ordinary Web Applications



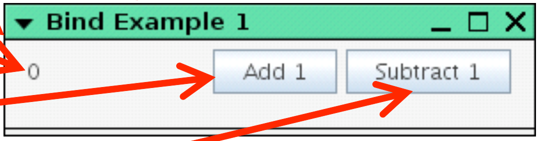
Ordinary Web Applications



JavaFX Script

```
import javafx.ui.*;
Frame {
  title: "Bind Example 1"
  width: 300
  height: 75
  content:
    FlowPanel {
      content: [
        Label {
          text: "0"
        },
        Button {
          text: "Add 1"
        },
        Button {
          text: "Subtract 1"
        }]}
  visible: true
}
```

composition

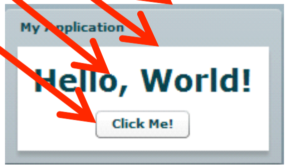


source: http://jfx.wikia.com/wiki/Introduction_to_Binding_in_JavaFX

MXML

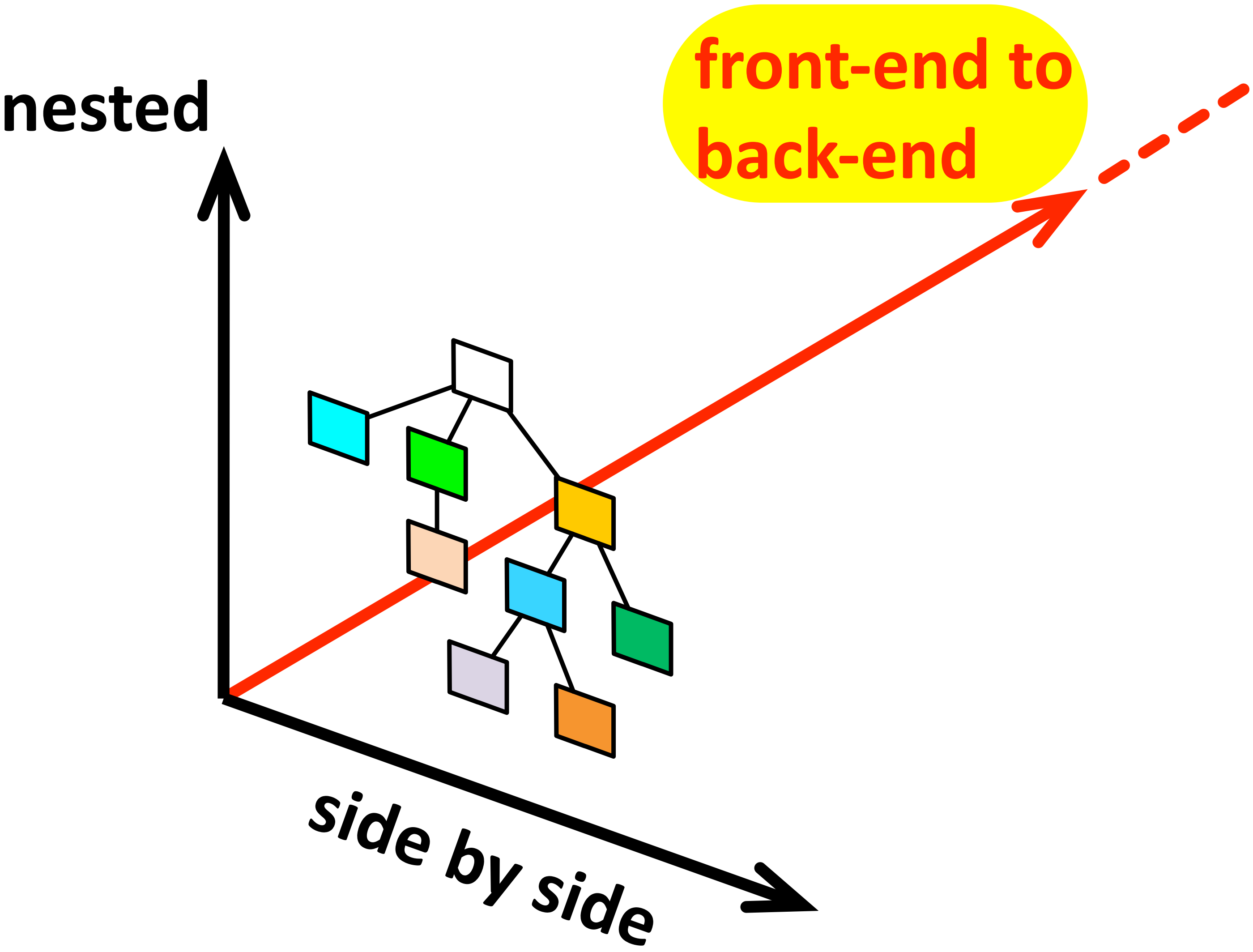
```
<mx:Application width="300" height="200">
  <mx:Panel title="My Application">
    <mx:Label id="myLabel"
      width="180" fontWeight="bold“ fontSize="24"/>
    <mx:Button id="myButton"
      label="Click Me!" click="clickHandler(event);" />
  </mx:Panel>
  <mx:Script>
    <![CDATA[
      import flash.events.MouseEvent;
      private function clickHandler(event:MouseEvent):void {
        myLabel.text = "Hello, World!";
      }
    ]]>
  </mx:Script>
</mx:Application>
```

composition

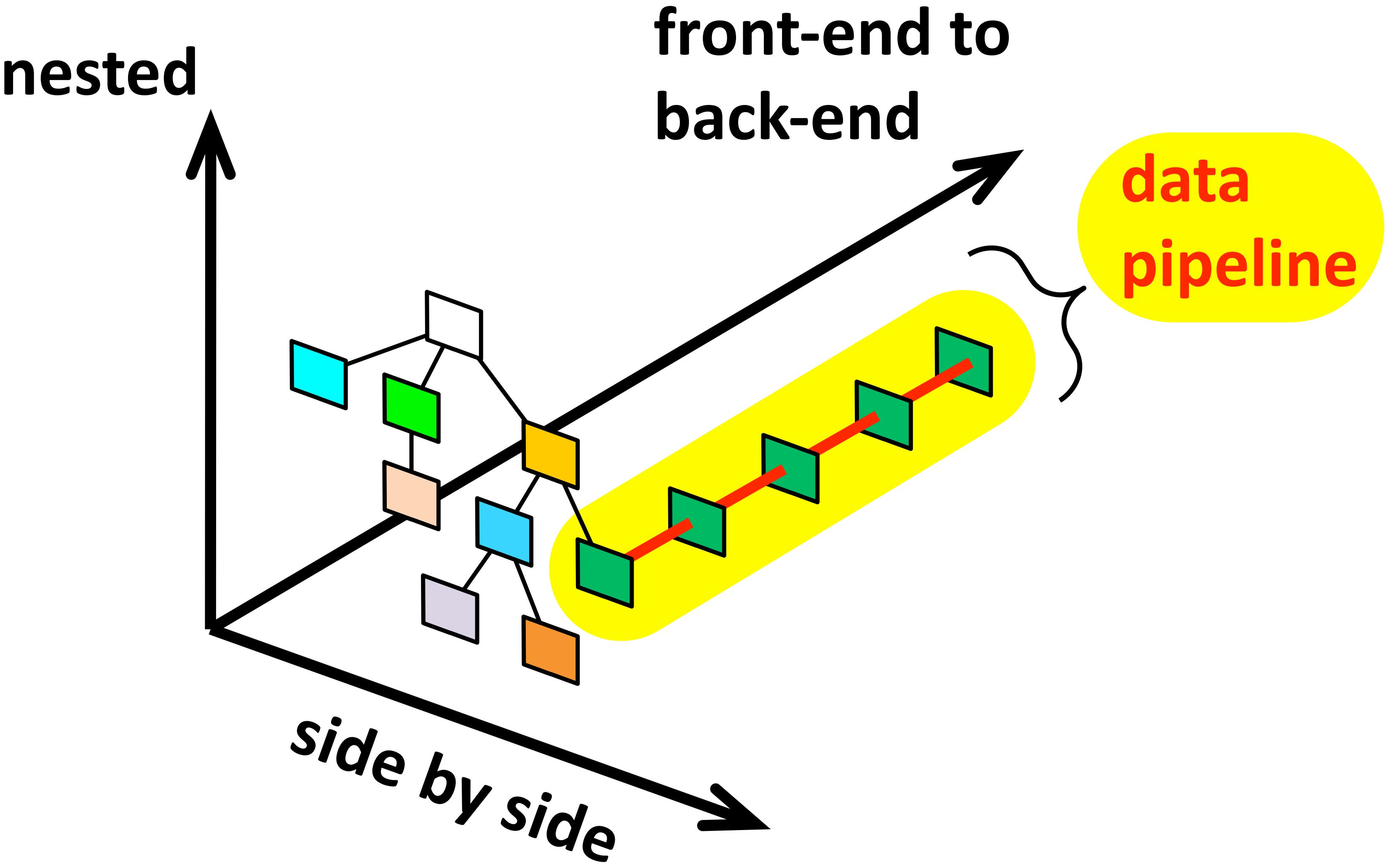


source: http://www.adobe.com/devnet/flex/quickstart/handling_events/

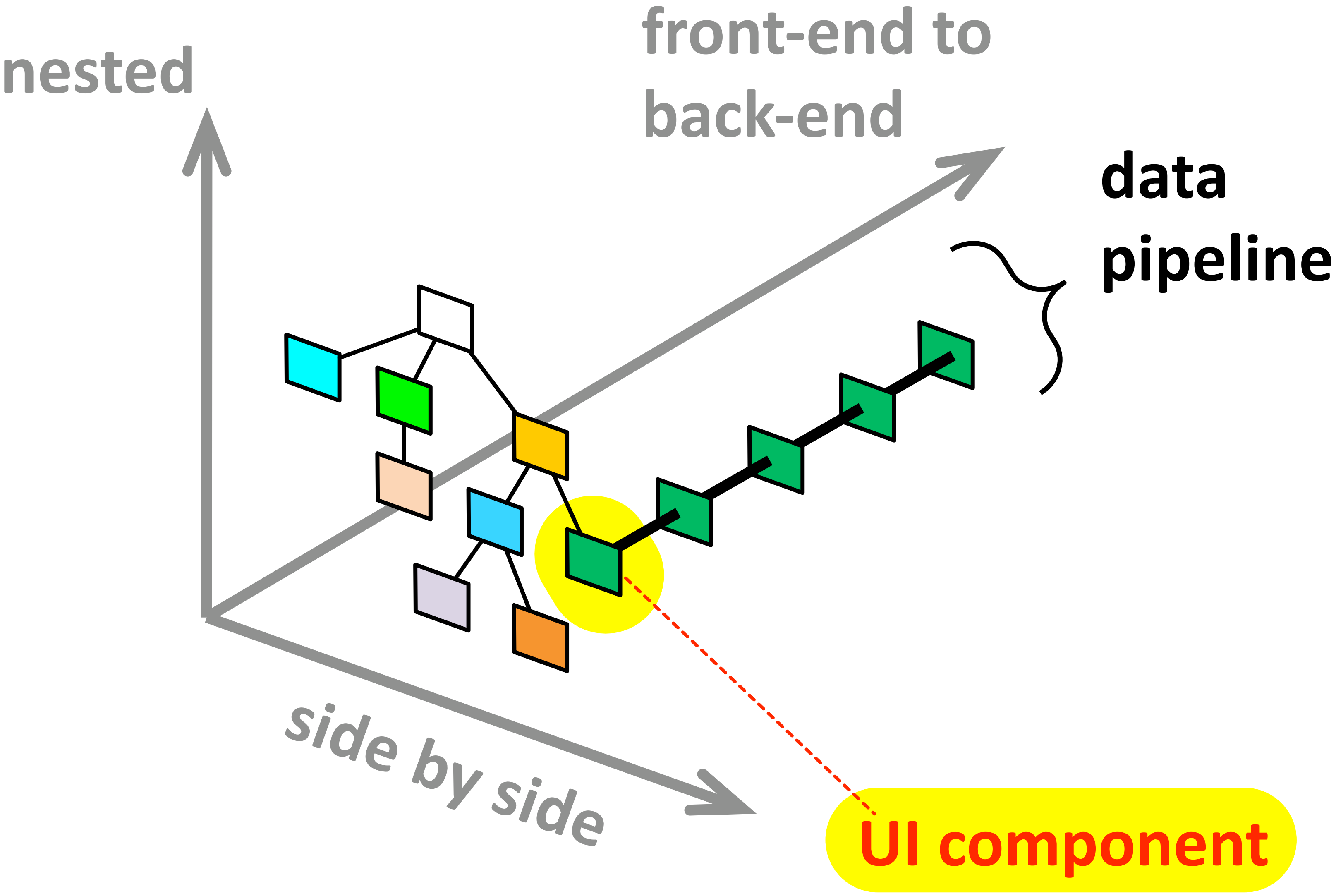
Live Objects Applications



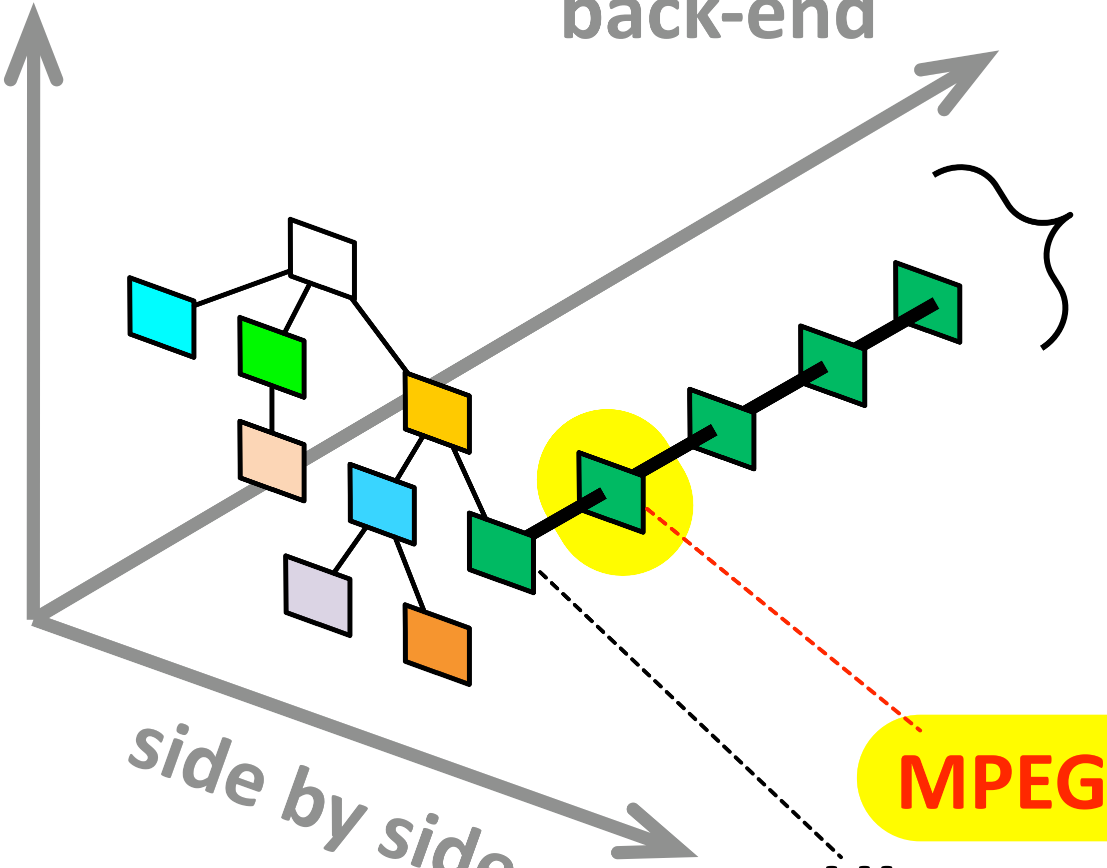
Live Objects Applications



Live Objects Applications



Live Objects Applications



nested

front-end to back-end

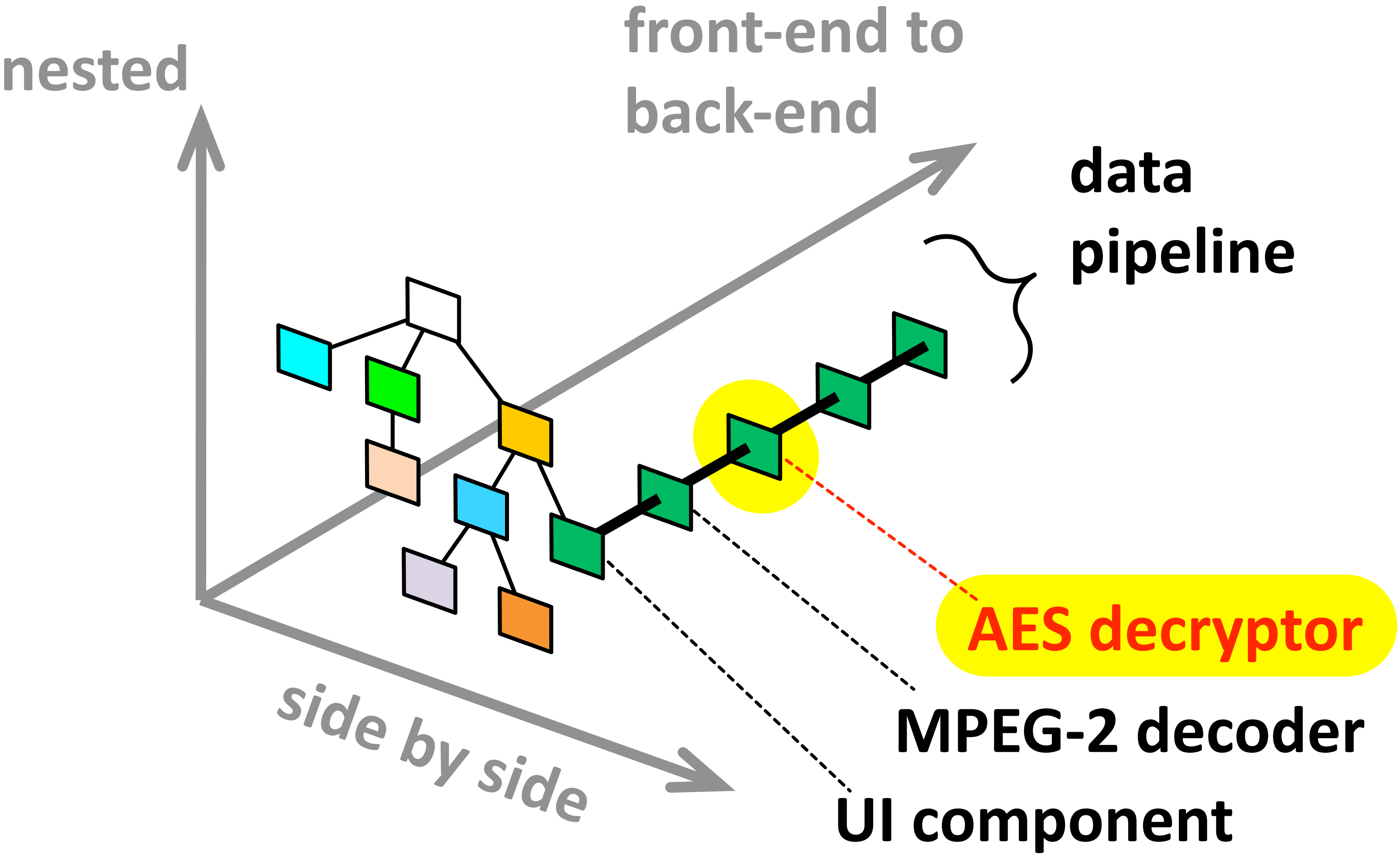
side by side

data pipeline

MPEG-2 decoder

UI component

Live Objects Applications



nested

front-end to back-end

data pipeline

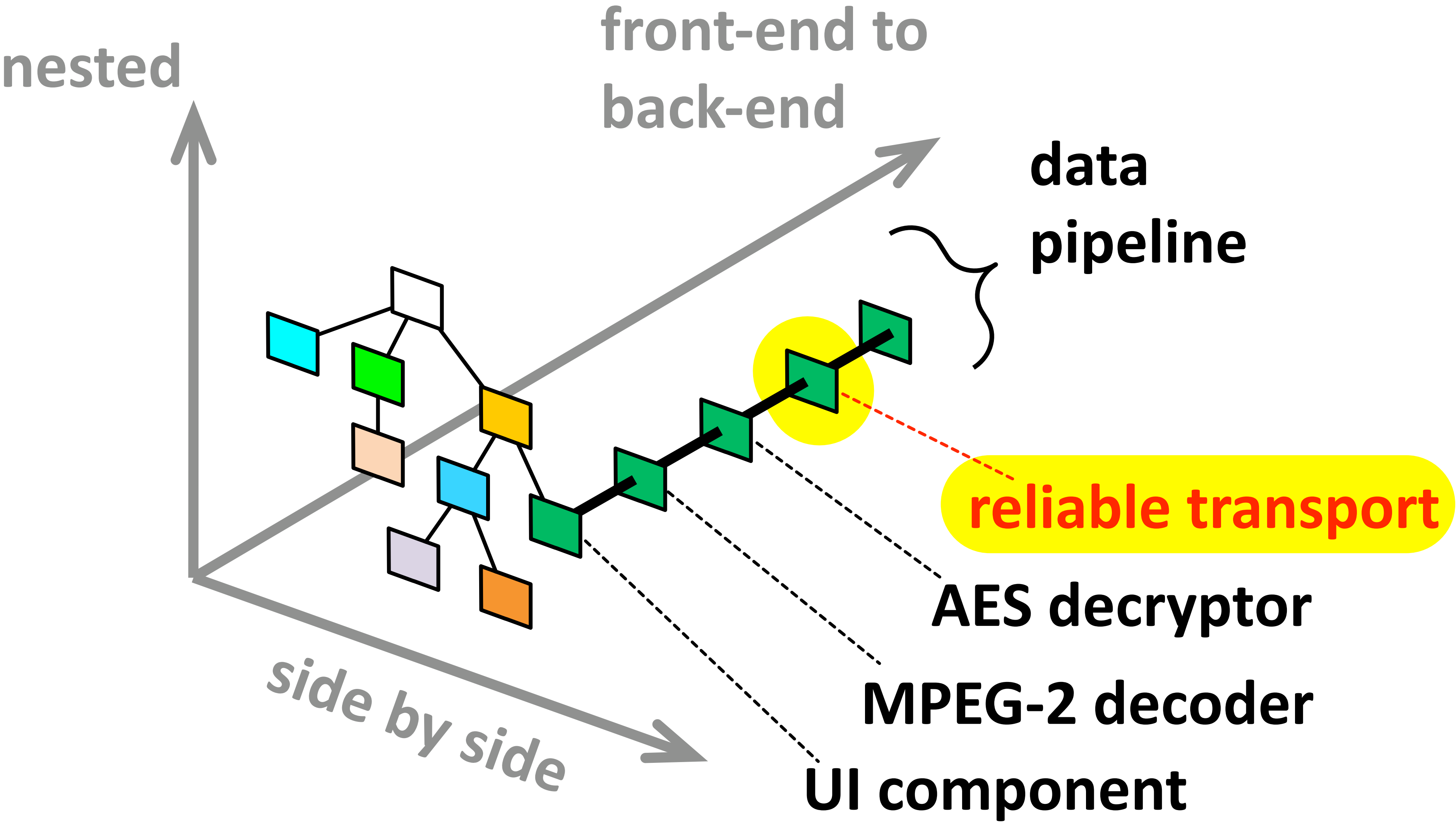
AES decryptor

MPEG-2 decoder

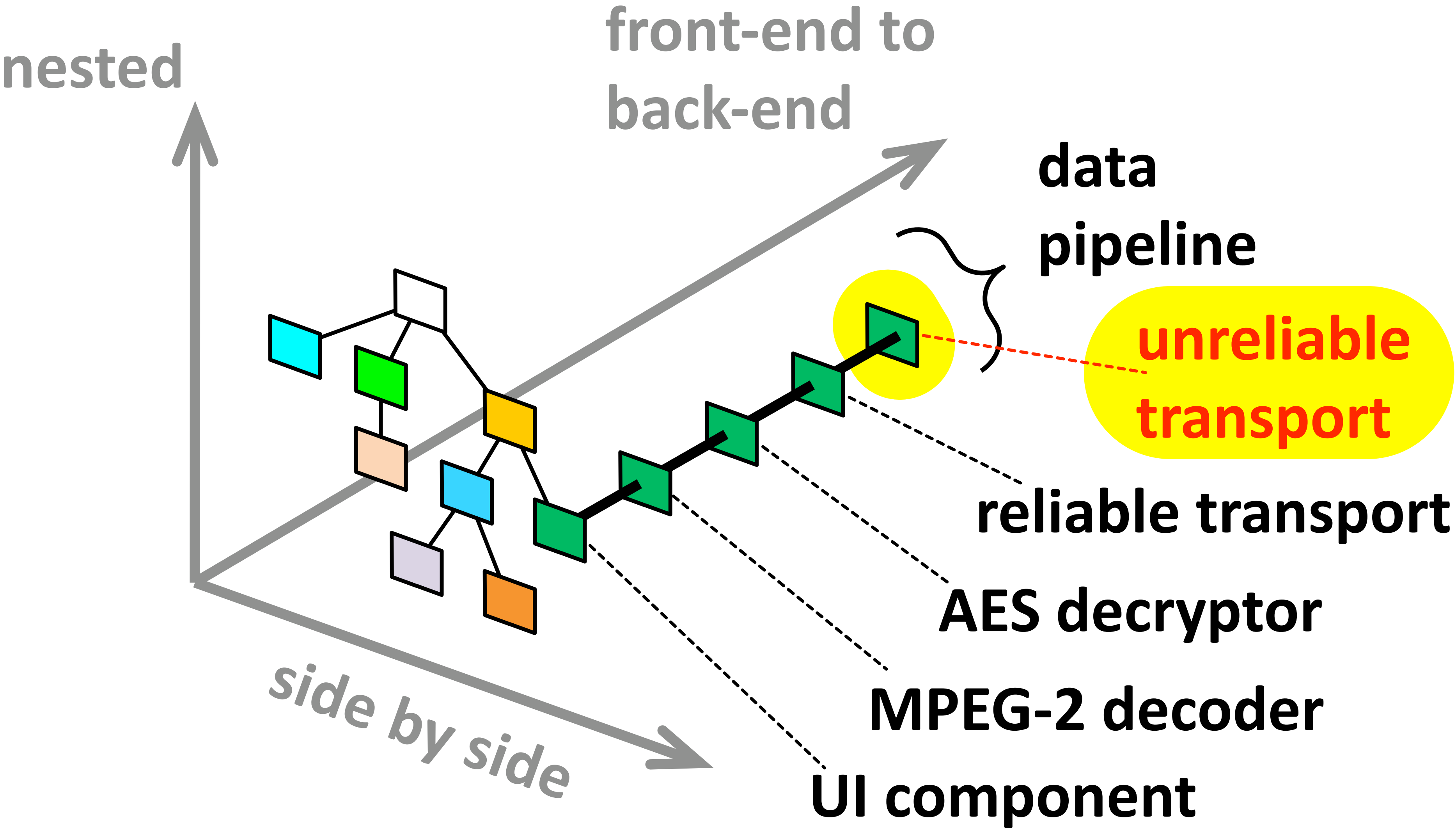
UI component

side by side

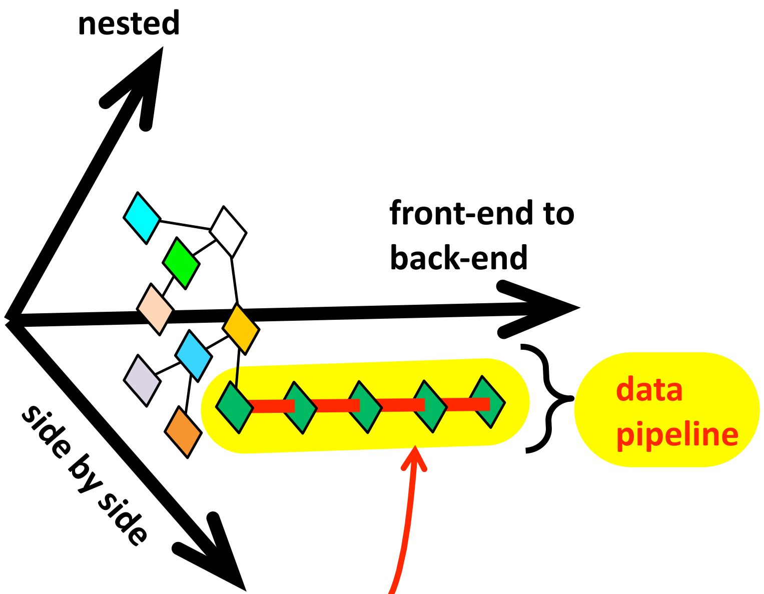
Live Objects Applications



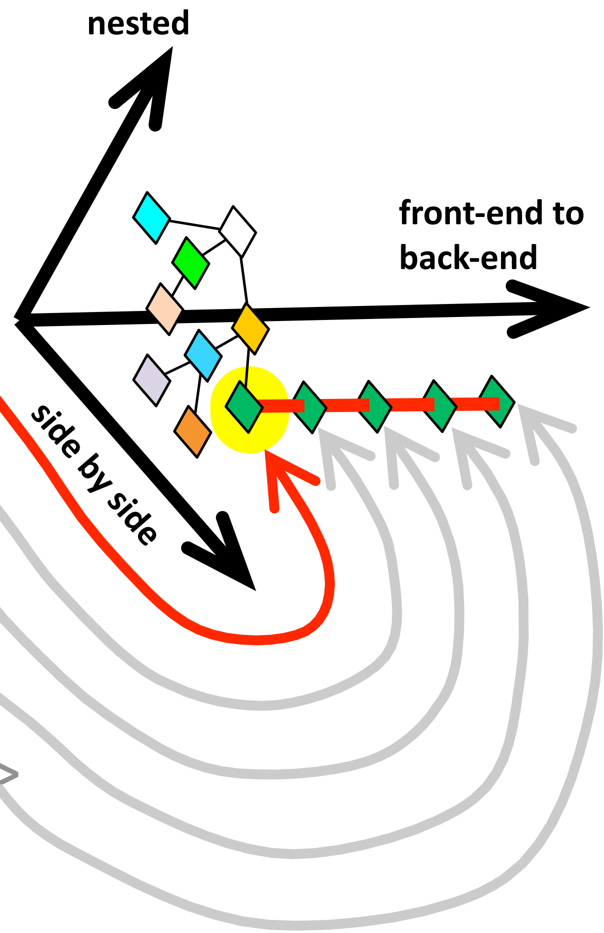
Live Objects Applications



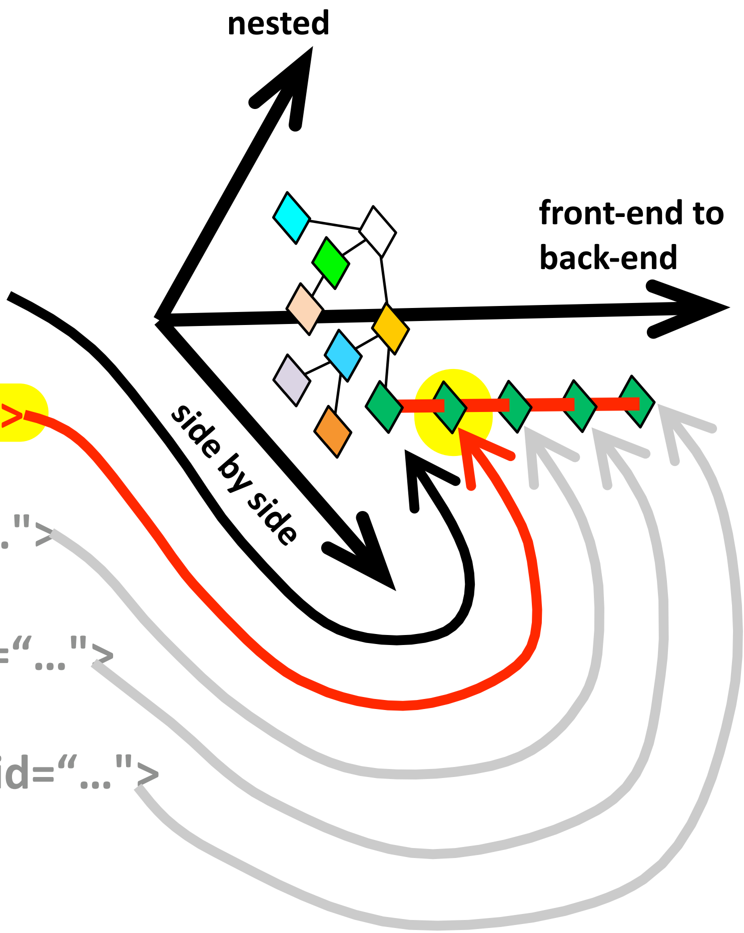
```
<Object xsi:type="ReferenceObject" id=“…”>
 <Parameter id=“…">
  <Value xsi:type="ReferenceObject" id=“…">
   <Parameter id=“…">
    <Value xsi:type="ReferenceObject" id=“…">
     <Parameter id=“…">
      <Value xsi:type="ReferenceObject" id=“…">
       <Parameter id=“…">
        <Value xsi:type="ReferenceObject" id=“…">
         …
        </Value>
       </Parameter>
      </Value>
     </Parameter>
    </Value>
   </Parameter>
  </Value>
 </Parameter>
</Object>
```


nested
front-end to back-end
side by side
data pipeline

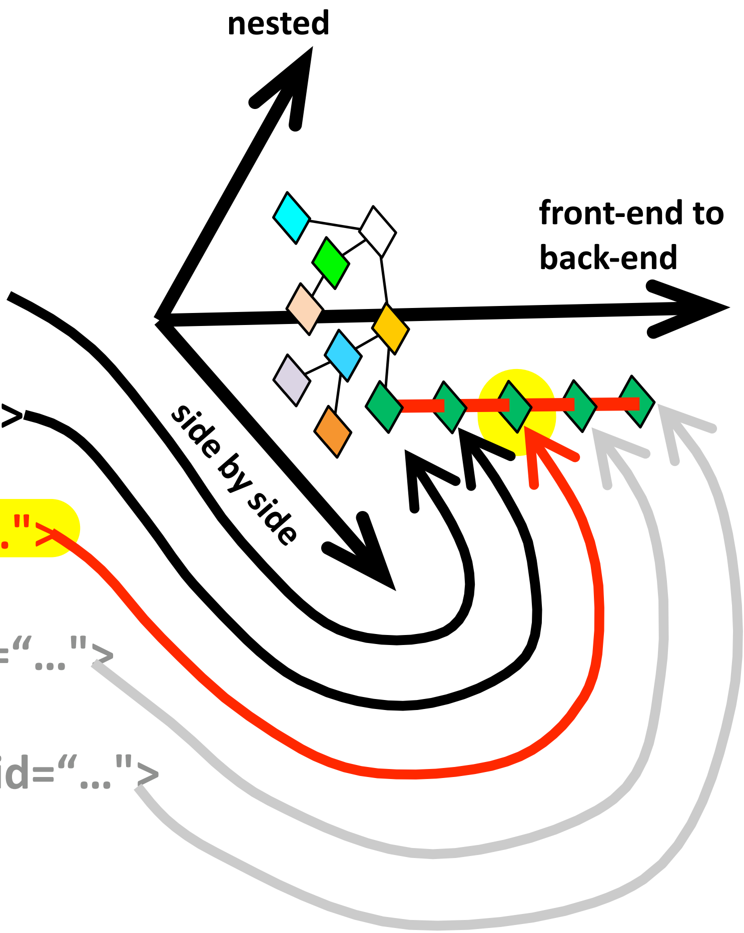
```
<Object xsi:type="ReferenceObject" id=“…”>
  <Parameter id=“…">
    <Value xsi:type="ReferenceObject" id=“…">
      <Parameter id=“…">
        <Value xsi:type="ReferenceObject" id=“…">
          <Parameter id=“…">
            <Value xsi:type="ReferenceObject" id=“…">
              <Parameter id=“…">
                <Value xsi:type="ReferenceObject" id=“…">
                  …
                </Value>
              </Parameter>
            </Value>
          </Parameter>
        </Value>
      </Parameter>
    </Value>
  </Parameter>
</Object>
```


nested
front-end to back-end
side by side

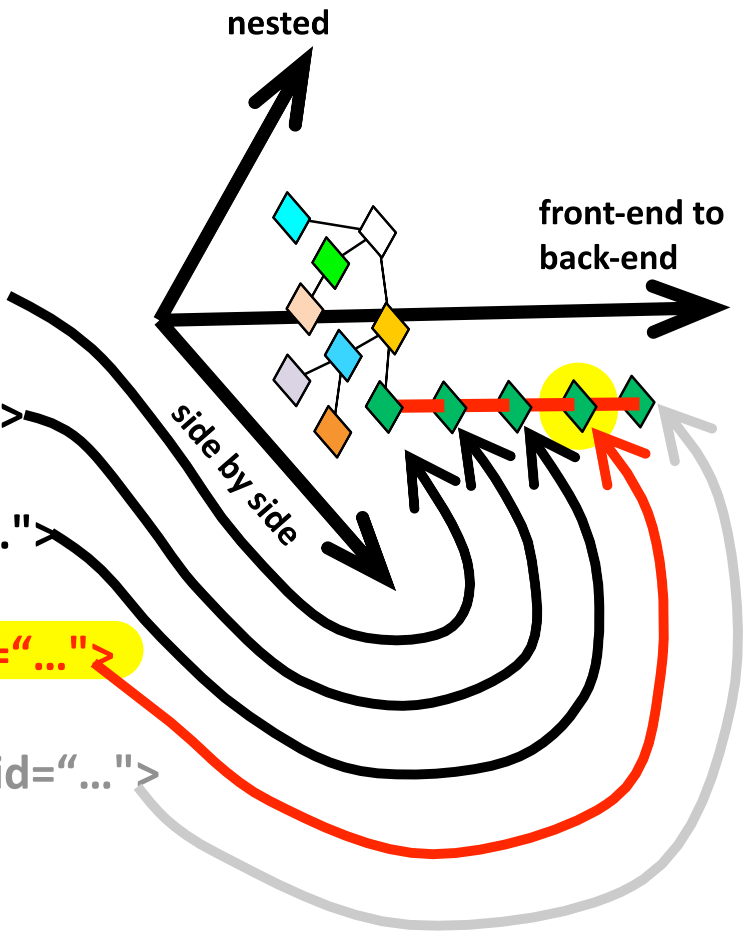
```
<Object xsi:type="ReferenceObject" id="…”>
 <Parameter id="…">
  <Value xsi:type="ReferenceObject" id="…">
    <Parameter id="…">
     <Value xsi:type="ReferenceObject" id="…">
       <Parameter id="…">
        <Value xsi:type="ReferenceObject" id="…">
          <Parameter id="…">
           <Value xsi:type="ReferenceObject" id="…">
            …
           </Value>
          </Parameter>
        </Value>
       </Parameter>
     </Value>
    </Parameter>
  </Value>
 </Parameter>
</Object>
```


nested
front-end to back-end
side by side

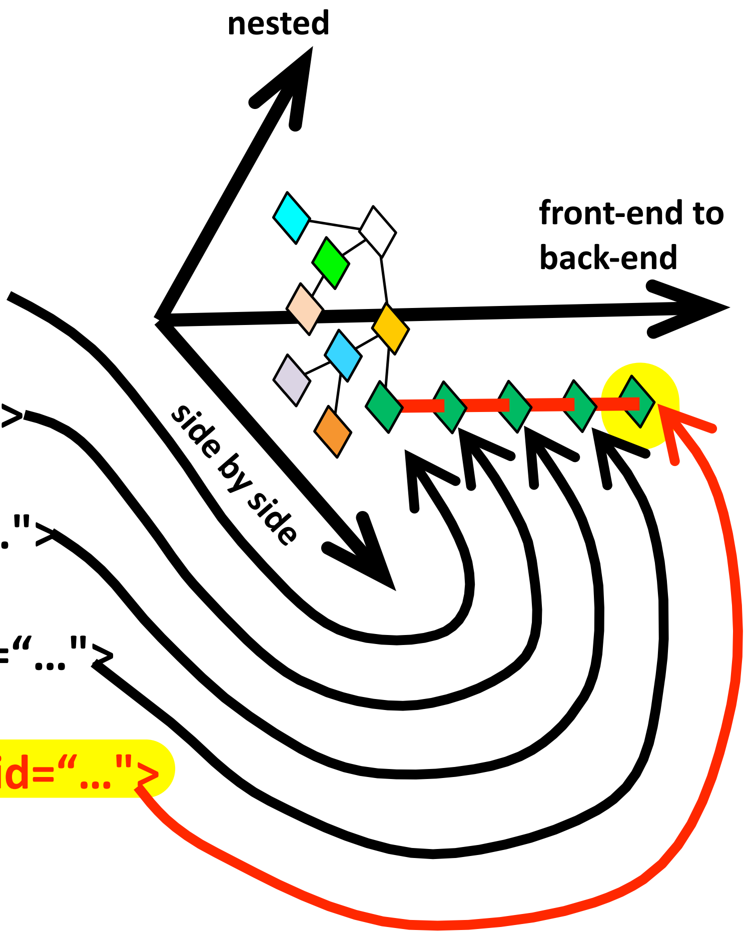
```
<Object xsi:type="ReferenceObject" id=“…”>
 <Parameter id=“…">
  <Value xsi:type="ReferenceObject" id=“…">
   <Parameter id=“…">
   <Value xsi:type="ReferenceObject" id=“…">
     <Parameter id=“…">
      <Value xsi:type="ReferenceObject" id=“…">
       <Parameter id=“…">
        <Value xsi:type="ReferenceObject" id=“…">
         …
        </Value>
       </Parameter>
      </Value>
     </Parameter>
    </Value>
   </Parameter>
  </Value>
 </Parameter>
</Object>
```


nested
front-end to back-end
side by side

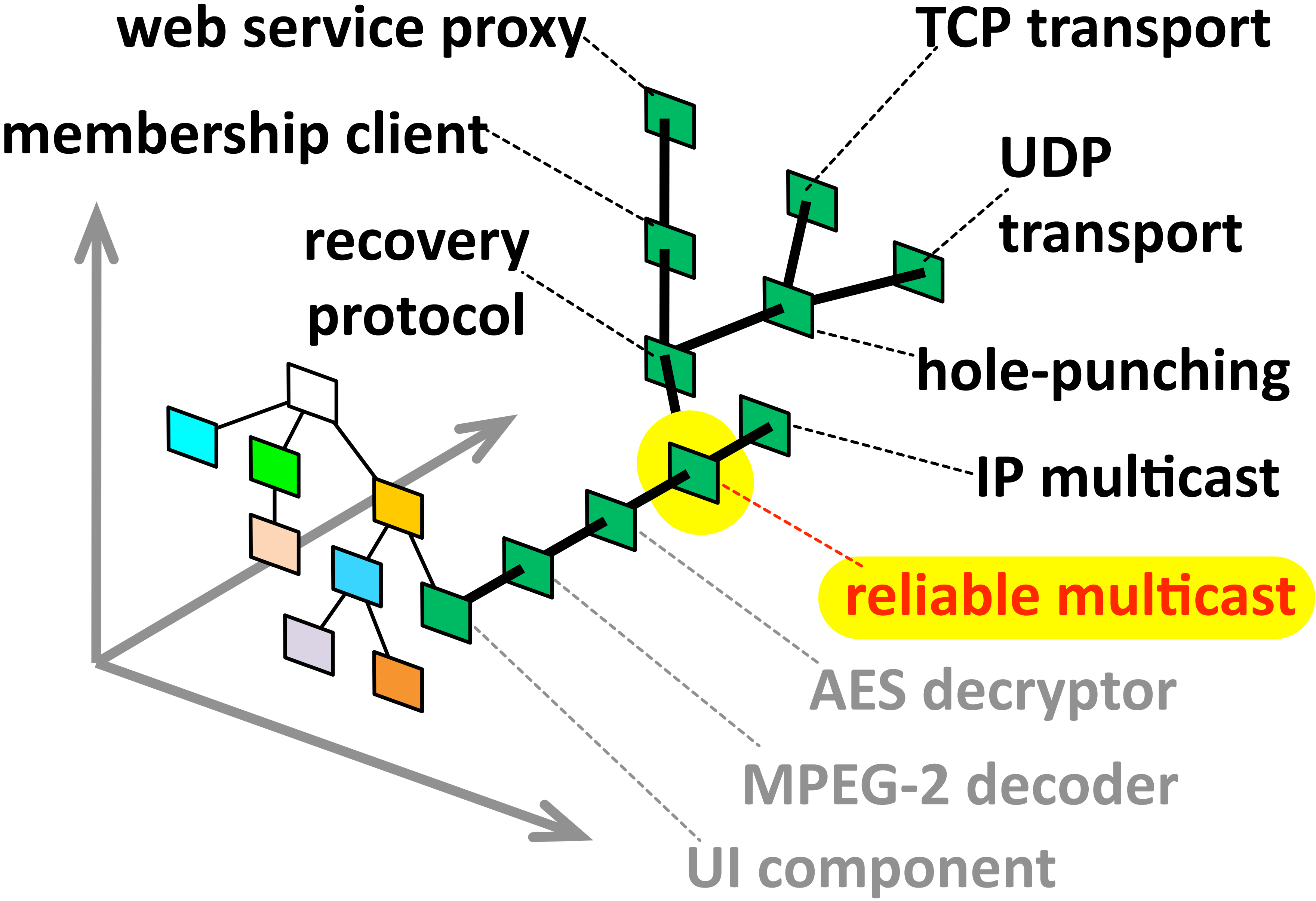
```
<Object xsi:type="ReferenceObject" id=“…”>
 <Parameter id=“…">
  <Value xsi:type="ReferenceObject" id=“…">
   <Parameter id=“…">
    <Value xsi:type="ReferenceObject" id=“…">
     <Parameter id=“…">
      <Value xsi:type="ReferenceObject" id=“…">
       <Parameter id=“…">
        <Value xsi:type="ReferenceObject" id=“…">
         …
        </Value>
       </Parameter>
      </Value>
     </Parameter>
    </Value>
   </Parameter>
  </Value>
 </Parameter>
</Object>
```


nested
front-end to back-end
side by side

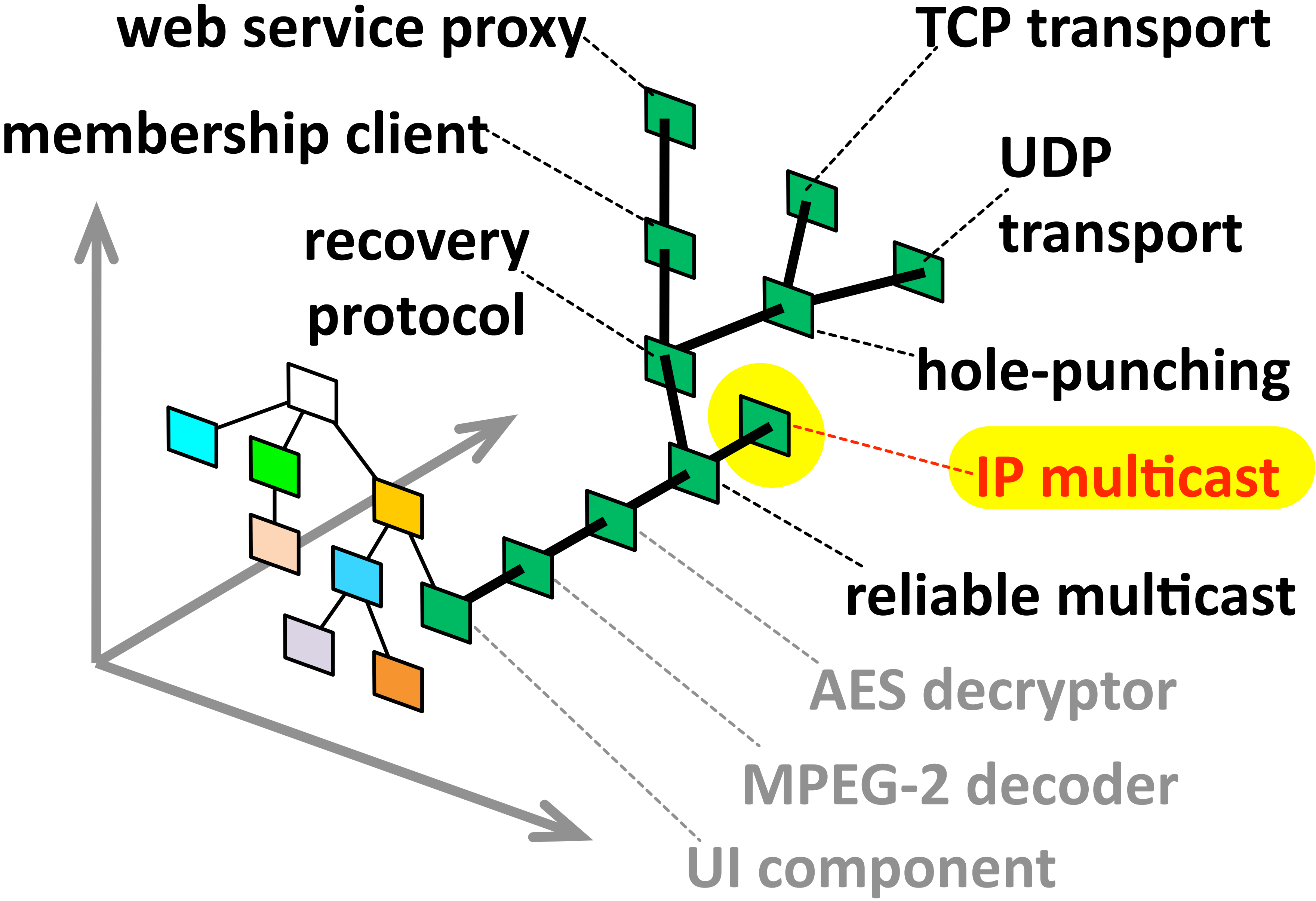
```
<Object xsi:type="ReferenceObject" id=“…”>
 <Parameter id=“…">
  <Value xsi:type="ReferenceObject" id=“…">
   <Parameter id=“…">
    <Value xsi:type="ReferenceObject" id=“…">
     <Parameter id=“…">
      <Value xsi:type="ReferenceObject" id=“…">
       <Parameter id=“…">
       <Value xsi:type="ReferenceObject" id=“…">
         …
       </Value>
       </Parameter>
      </Value>
     </Parameter>
    </Value>
   </Parameter>
  </Value>
 </Parameter>
</Object>
```


nested
front-end to back-end
side by side

Live Objects Applications



Live Objects Applications



web service proxy

TCP transport

membership client

UDP transport

recovery protocol

hole-punching

IP multicast

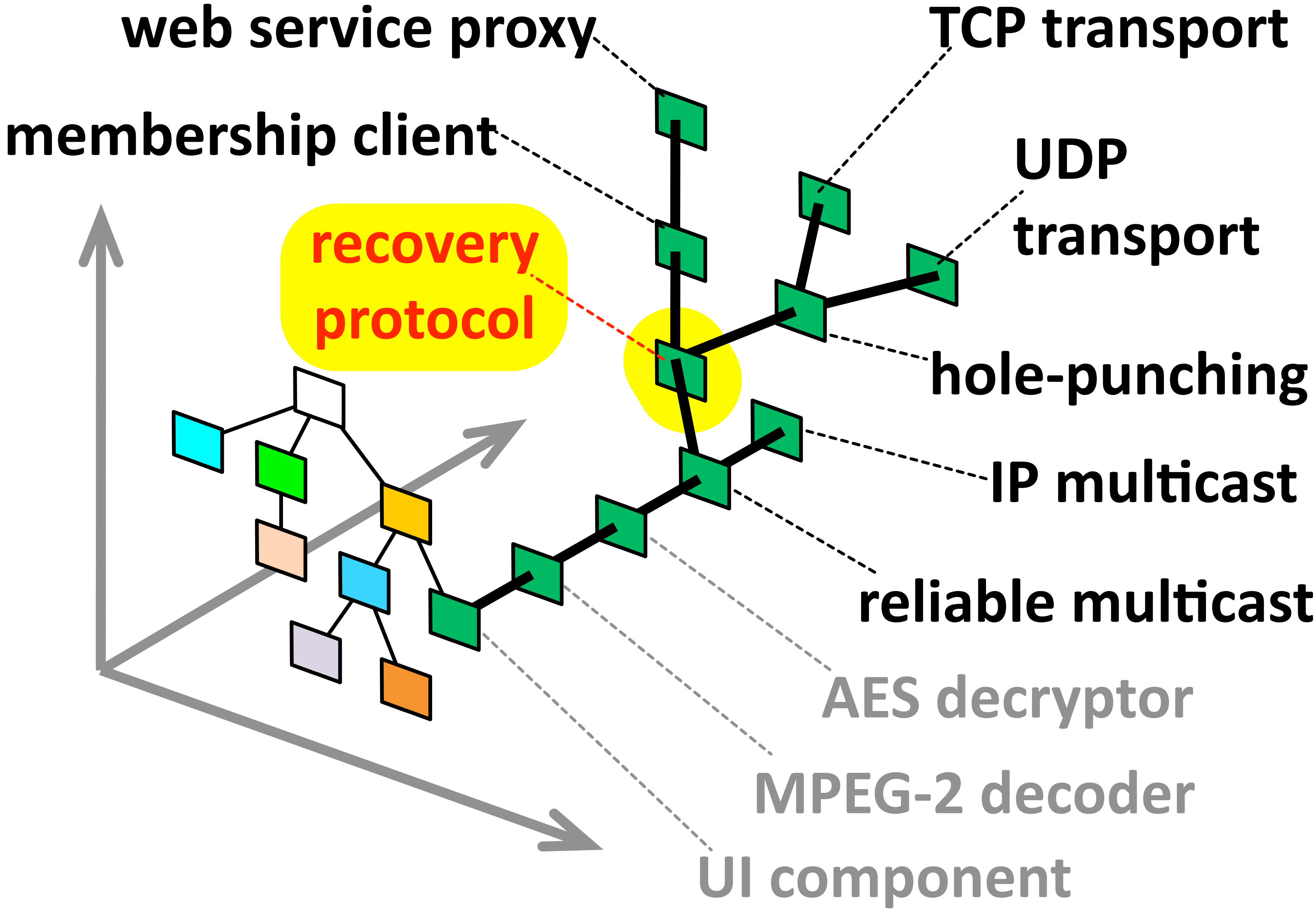
reliable multicast

AES decryptor

MPEG-2 decoder

UI component

Live Objects Applications



web service proxy

TCP transport

membership client

UDP transport

recovery protocol

hole-punching

IP multicast

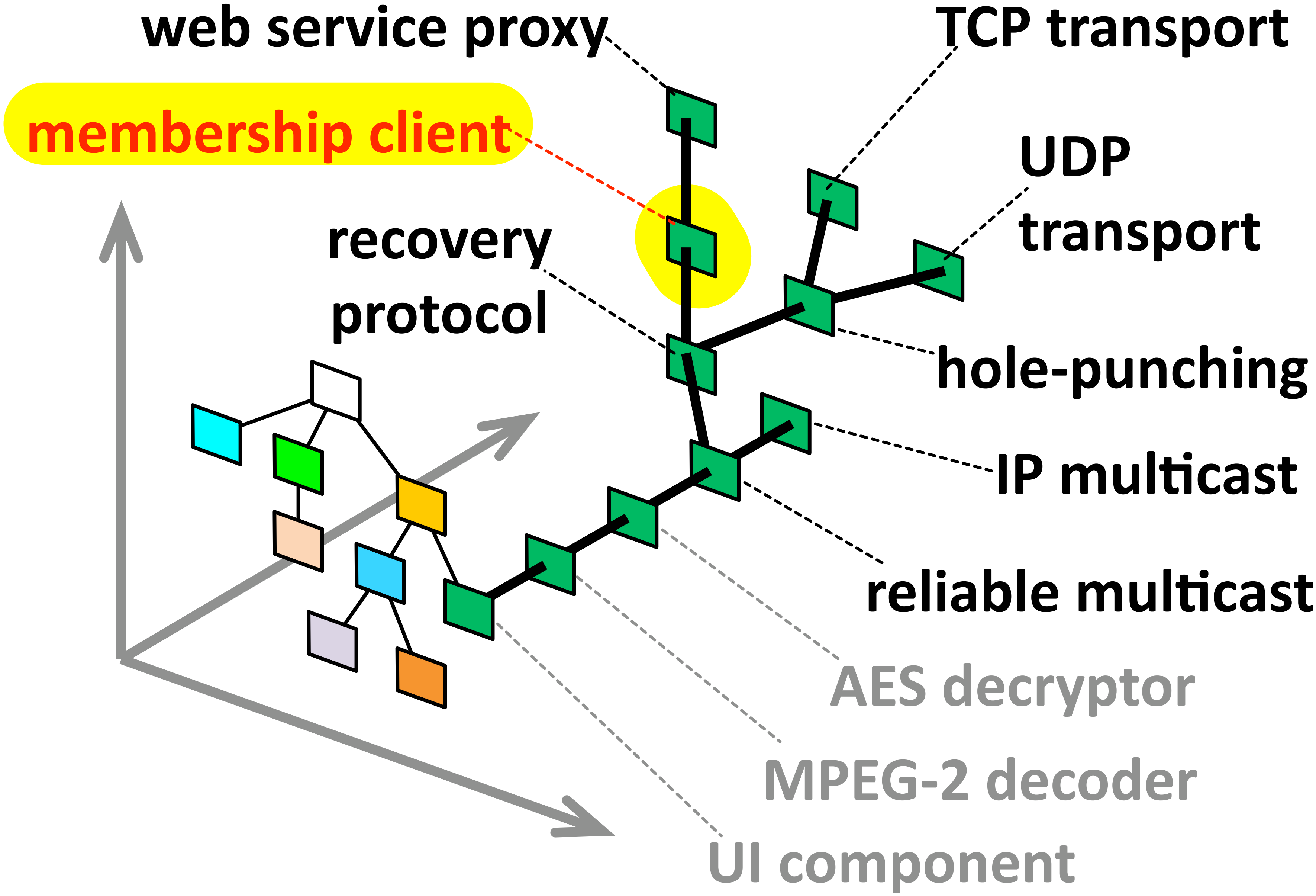
reliable multicast

AES decryptor

MPEG-2 decoder

UI component

Live Objects Applications



web service proxy

TCP transport

membership client

UDP transport

recovery protocol

hole-punching

IP multicast

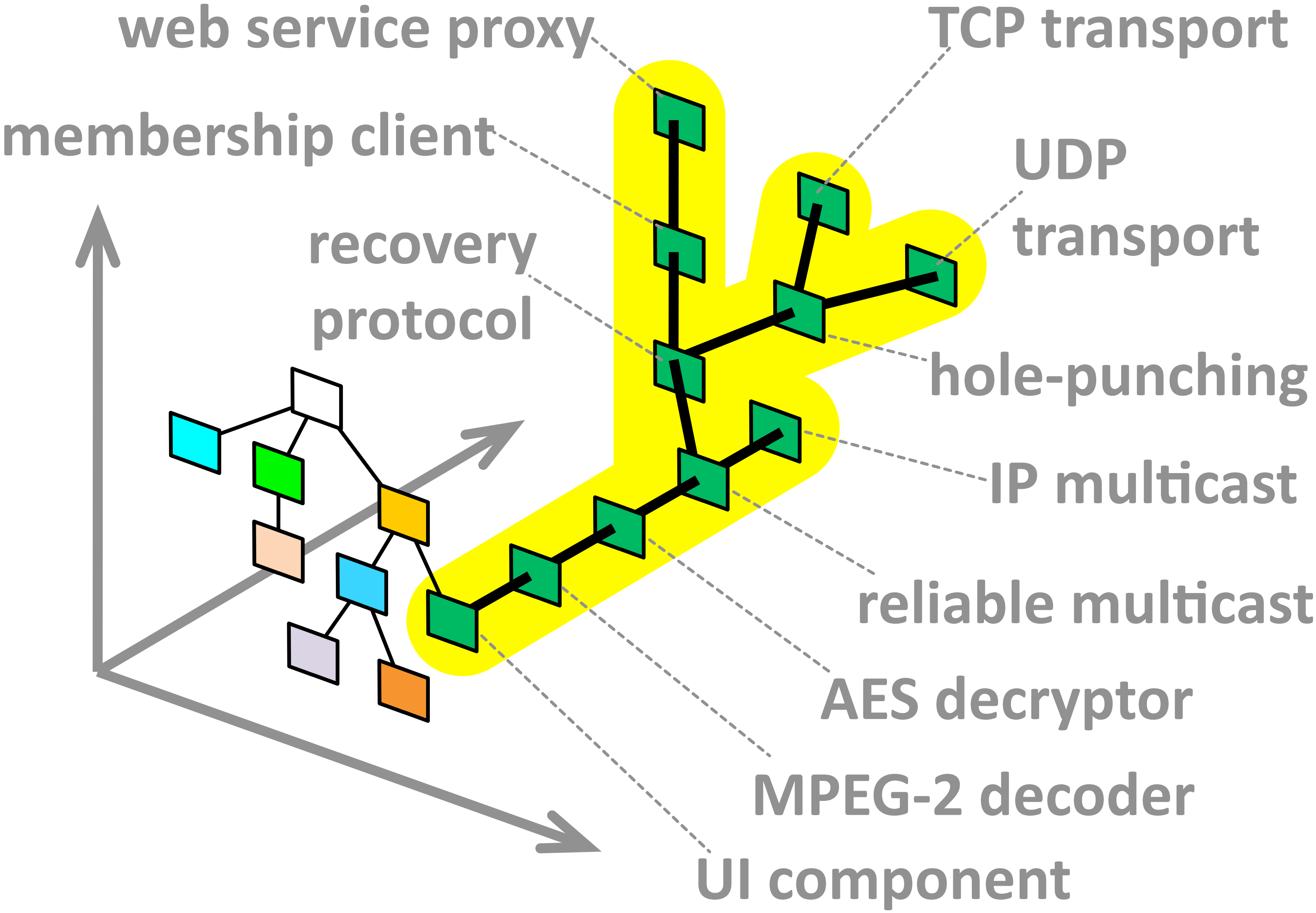
reliable multicast

AES decryptor

MPEG-2 decoder

UI component

Live Objects Applications



web service proxy

TCP transport

membership client

UDP transport

recovery protocol

hole-punching

IP multicast

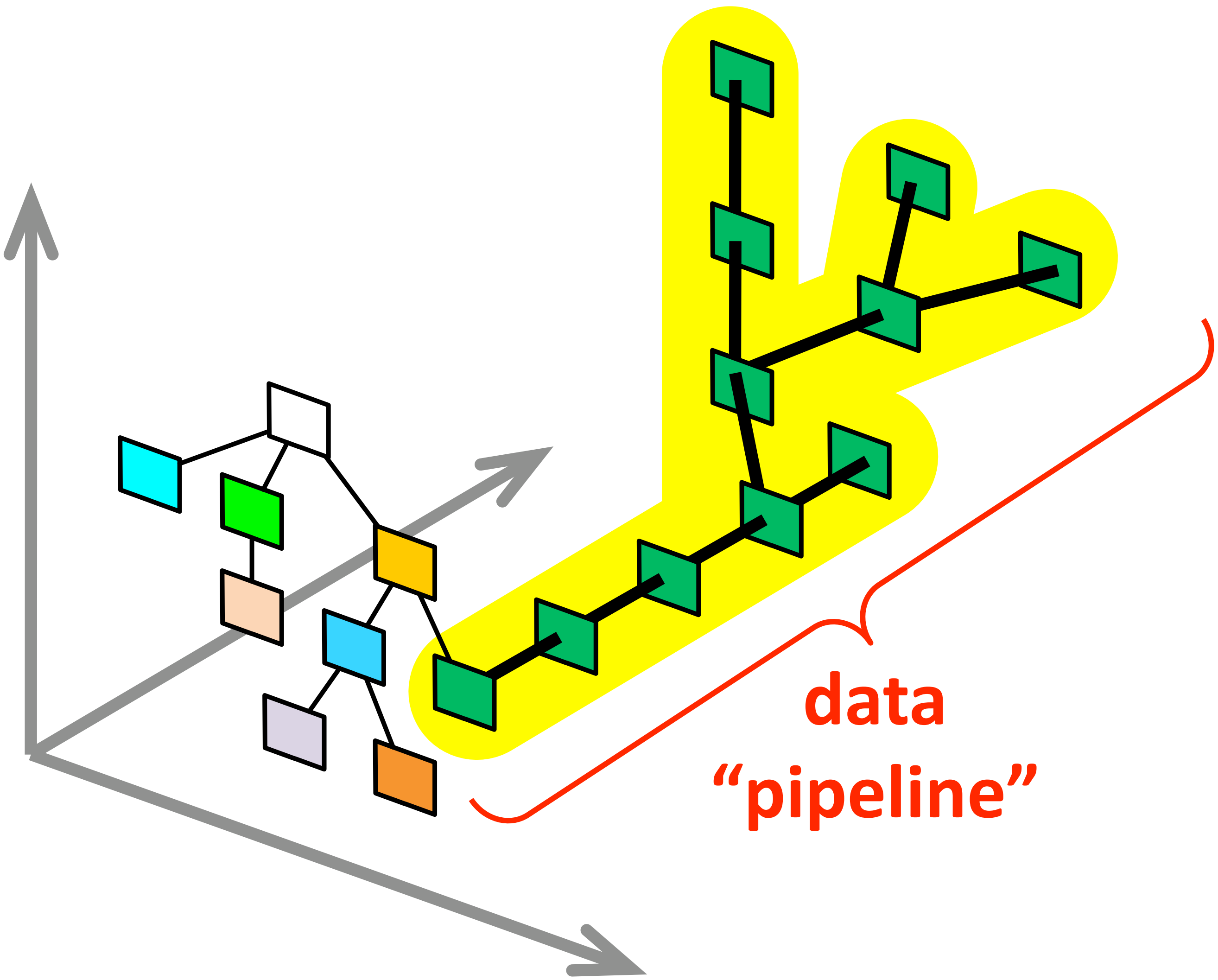
reliable multicast

AES decryptor

MPEG-2 decoder

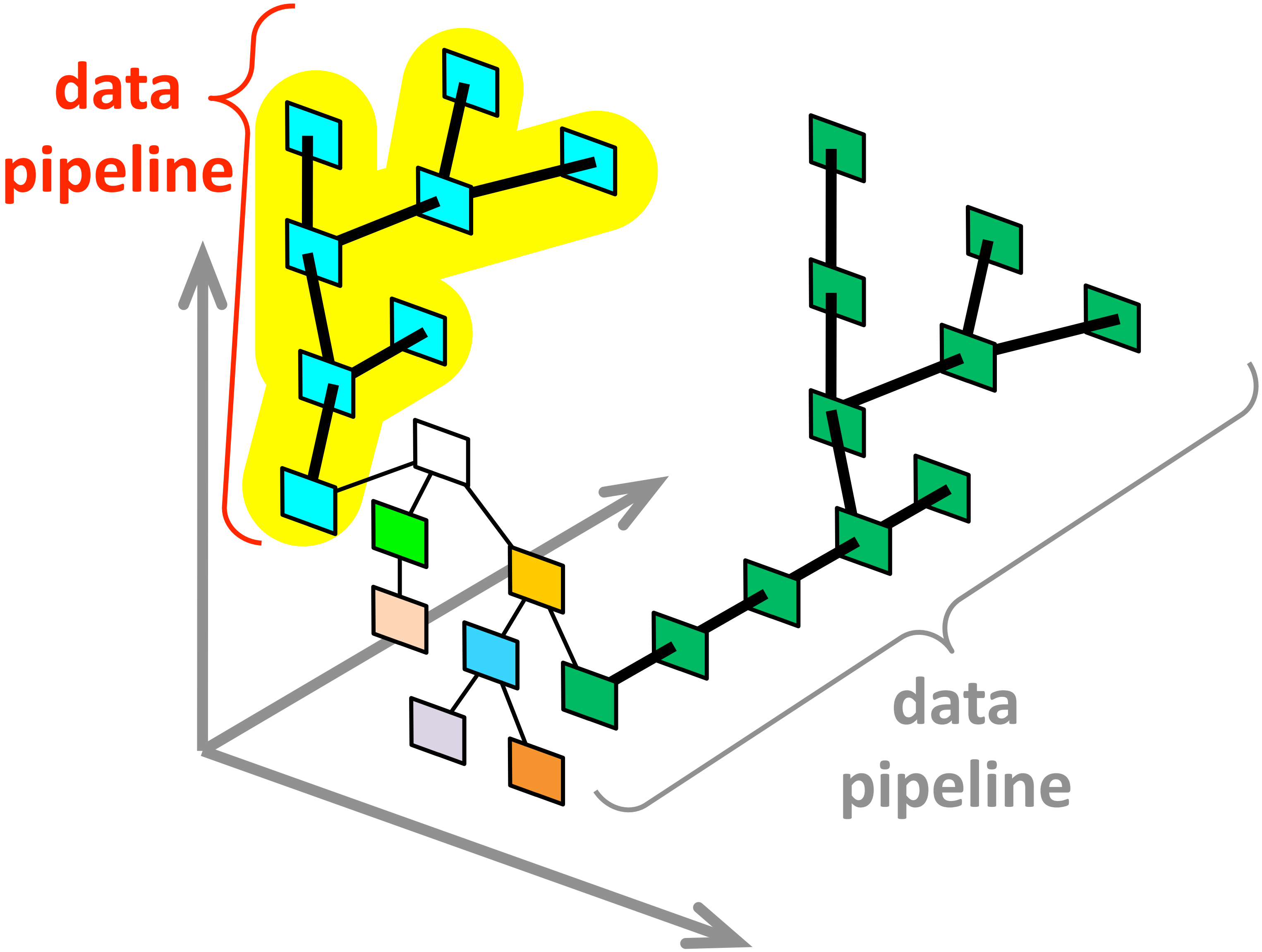
UI component

Live Objects Applications

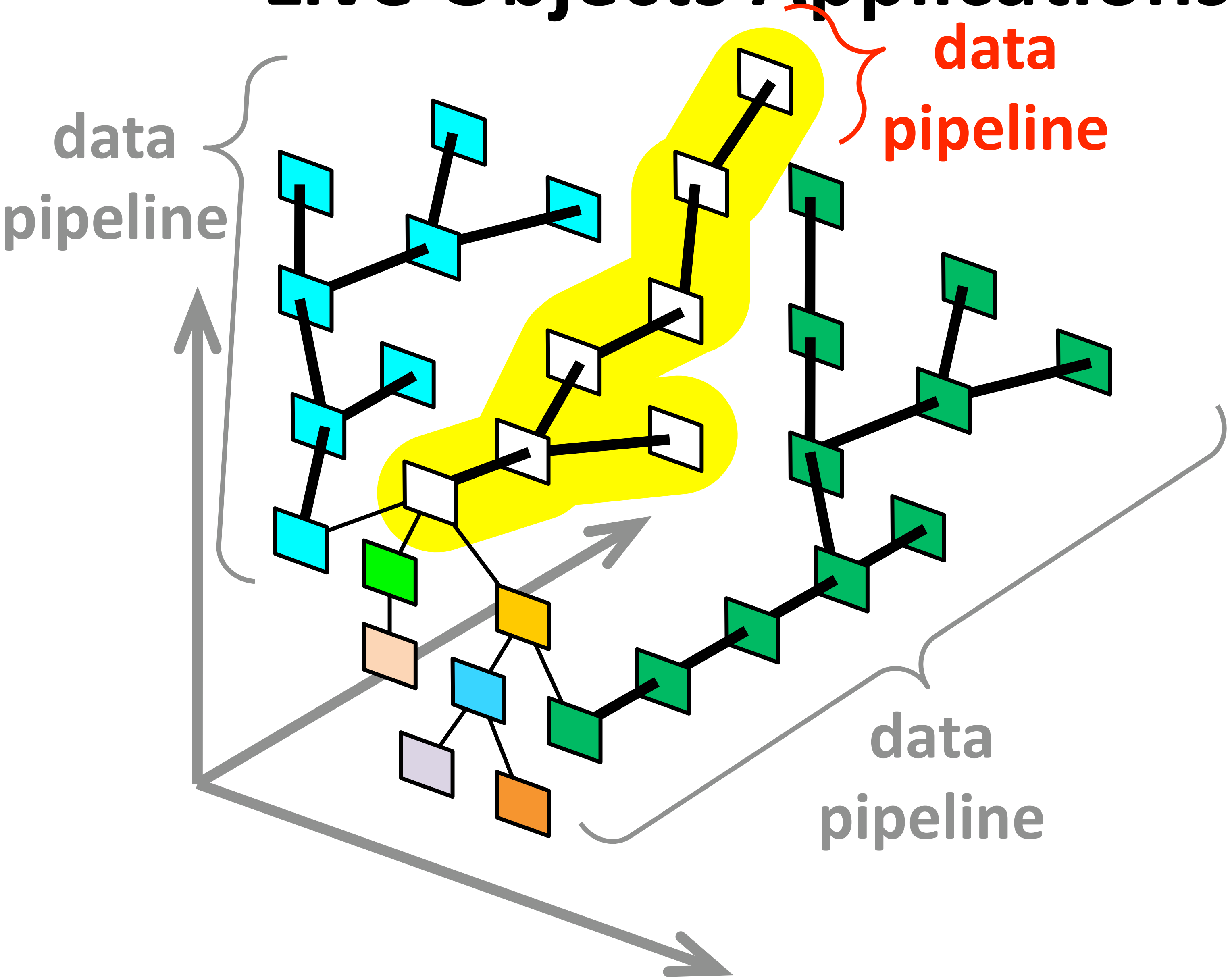


data
“pipeline”

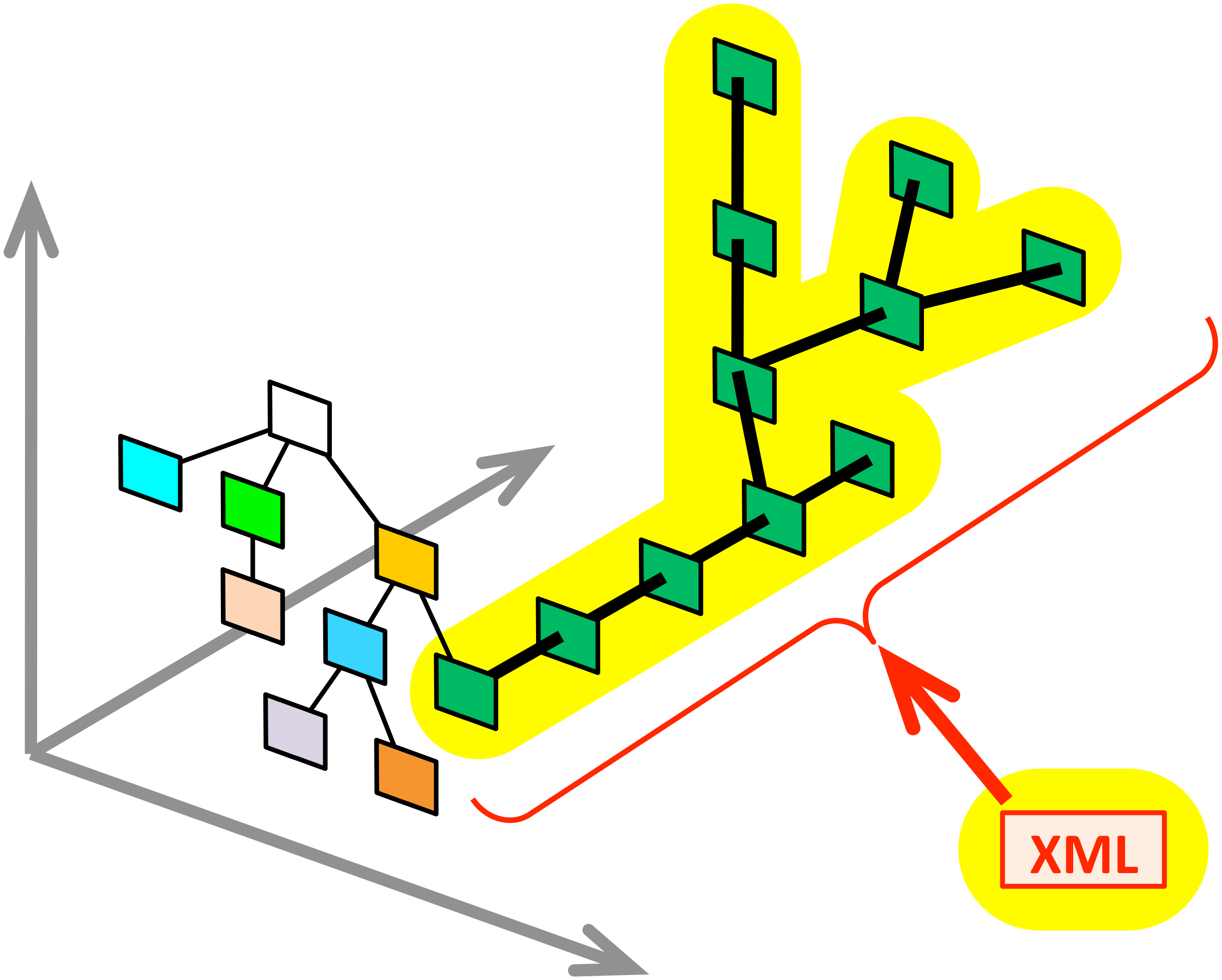
Live Objects Applications



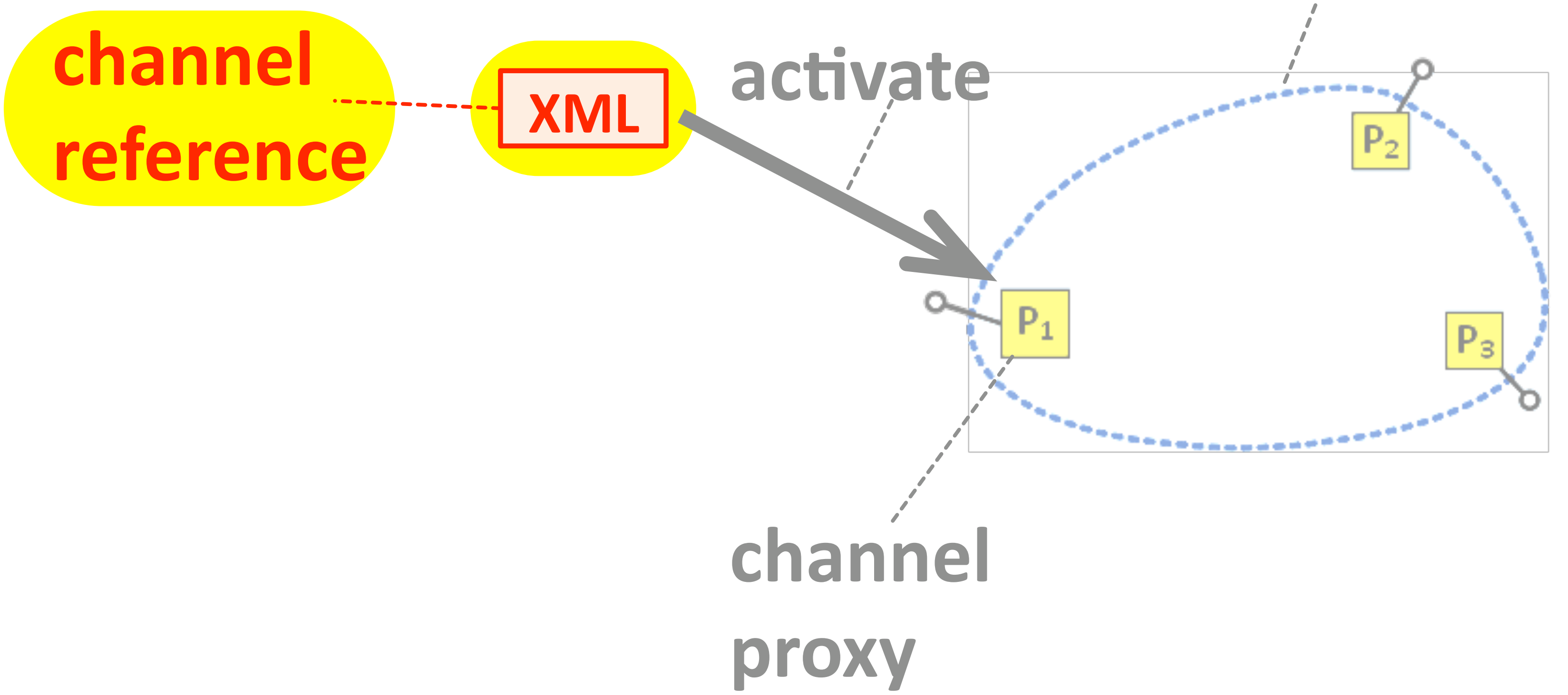
Live Objects Applications



Live Objects Applications



References



channel reference

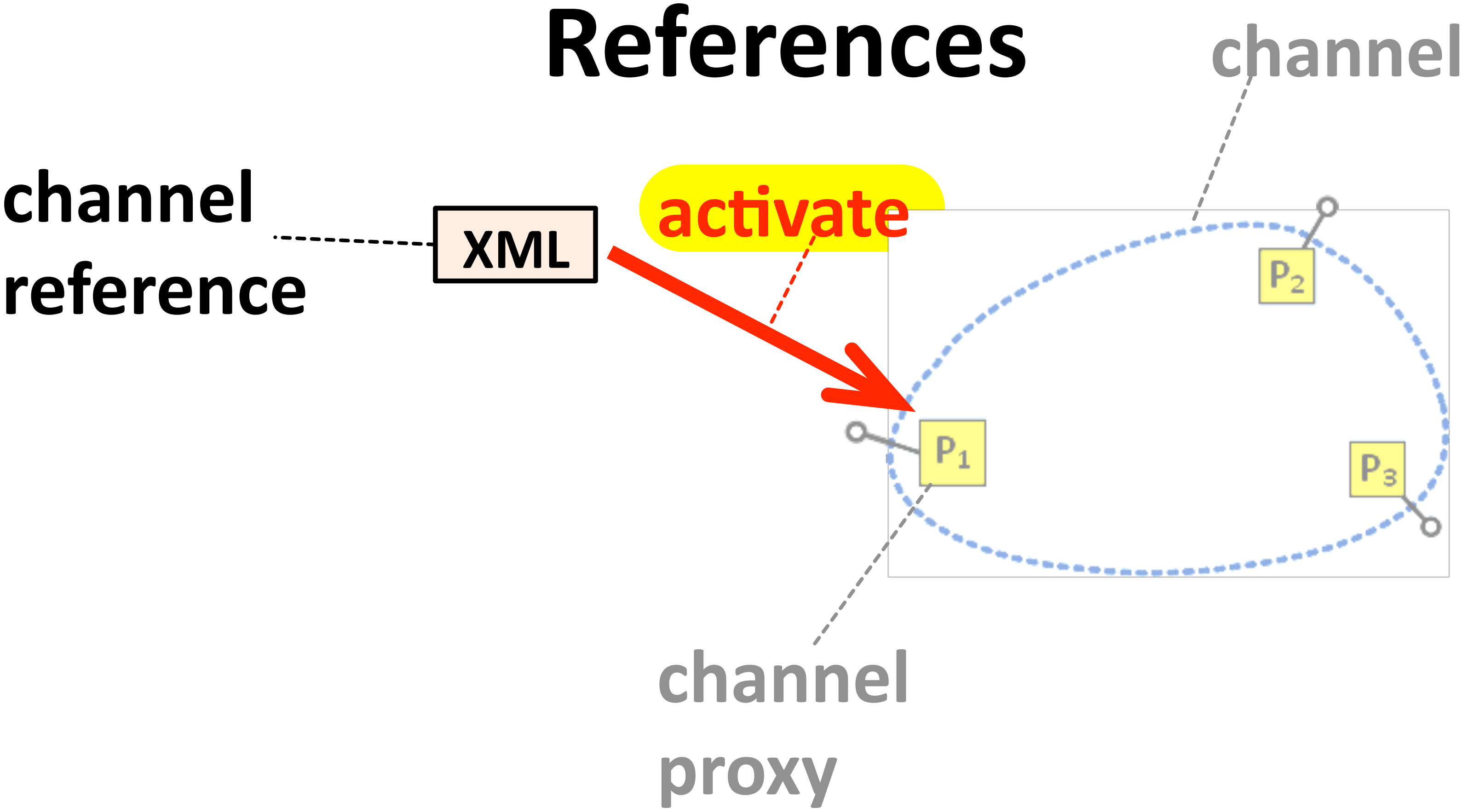
XML

activate

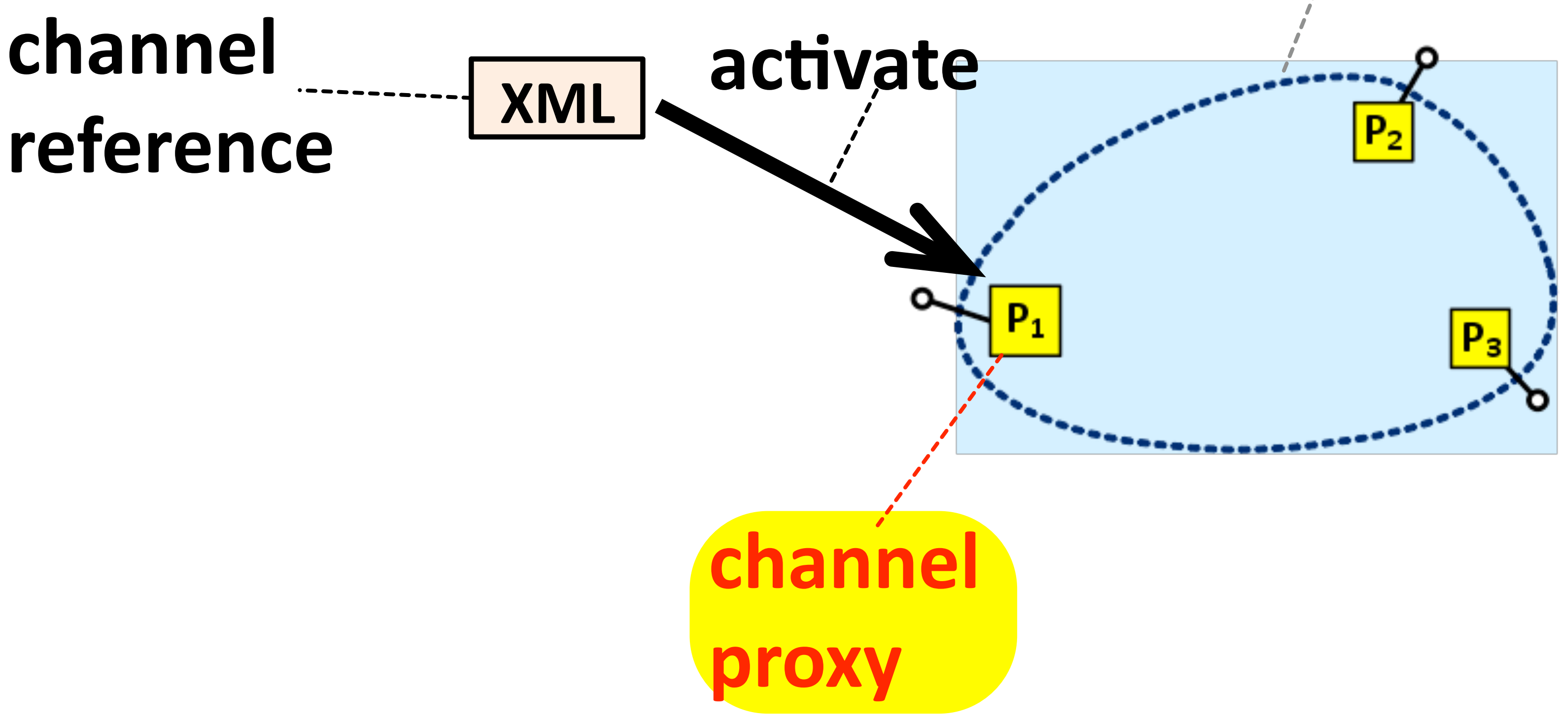
channel

channel proxy

References


channel
channel reference
XML
activate
P_1
P_2
P_3
channel proxy

References



channel

channel reference

XML

activate

P_1

P_2

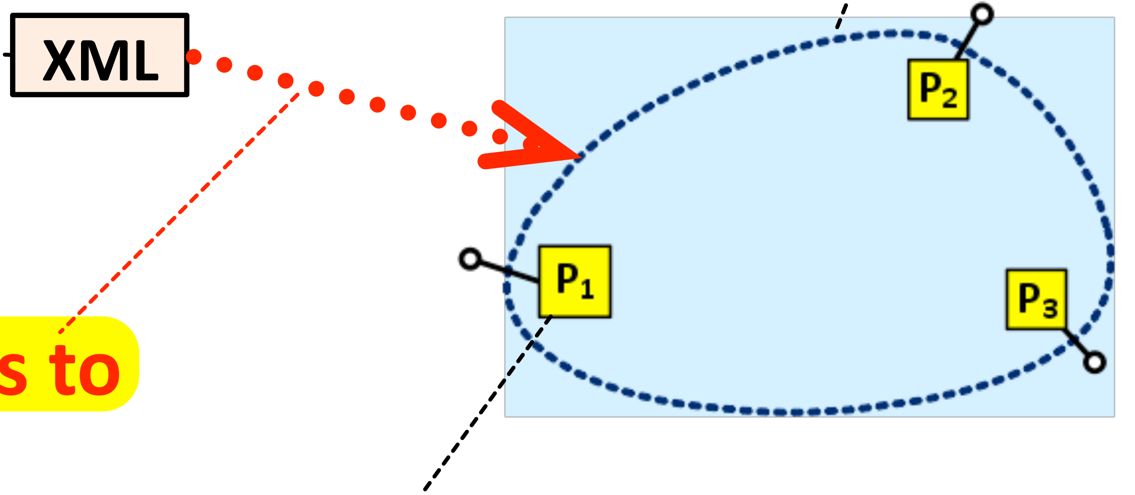
P_3

channel proxy

References

channel

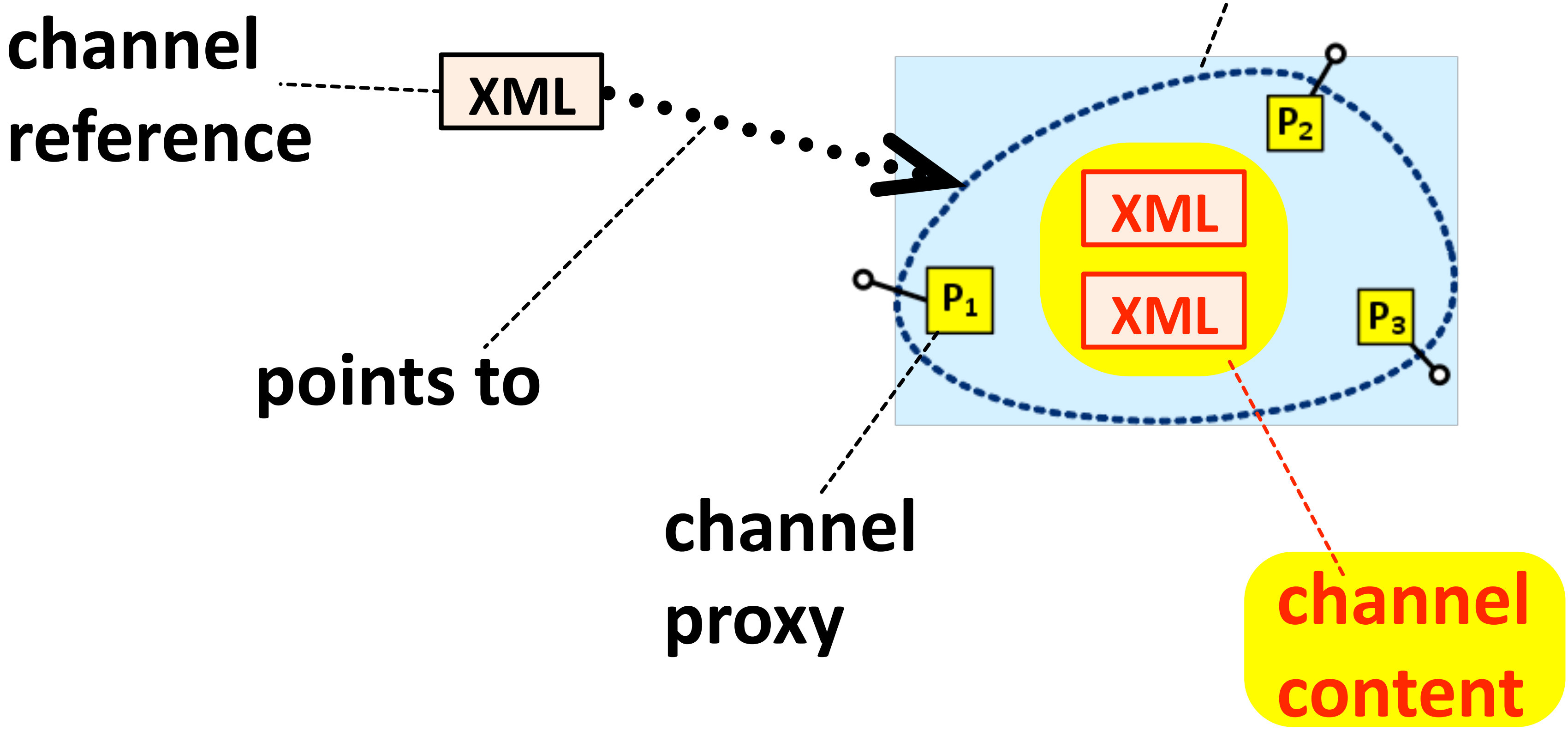
channel reference



points to

channel proxy

References



channel

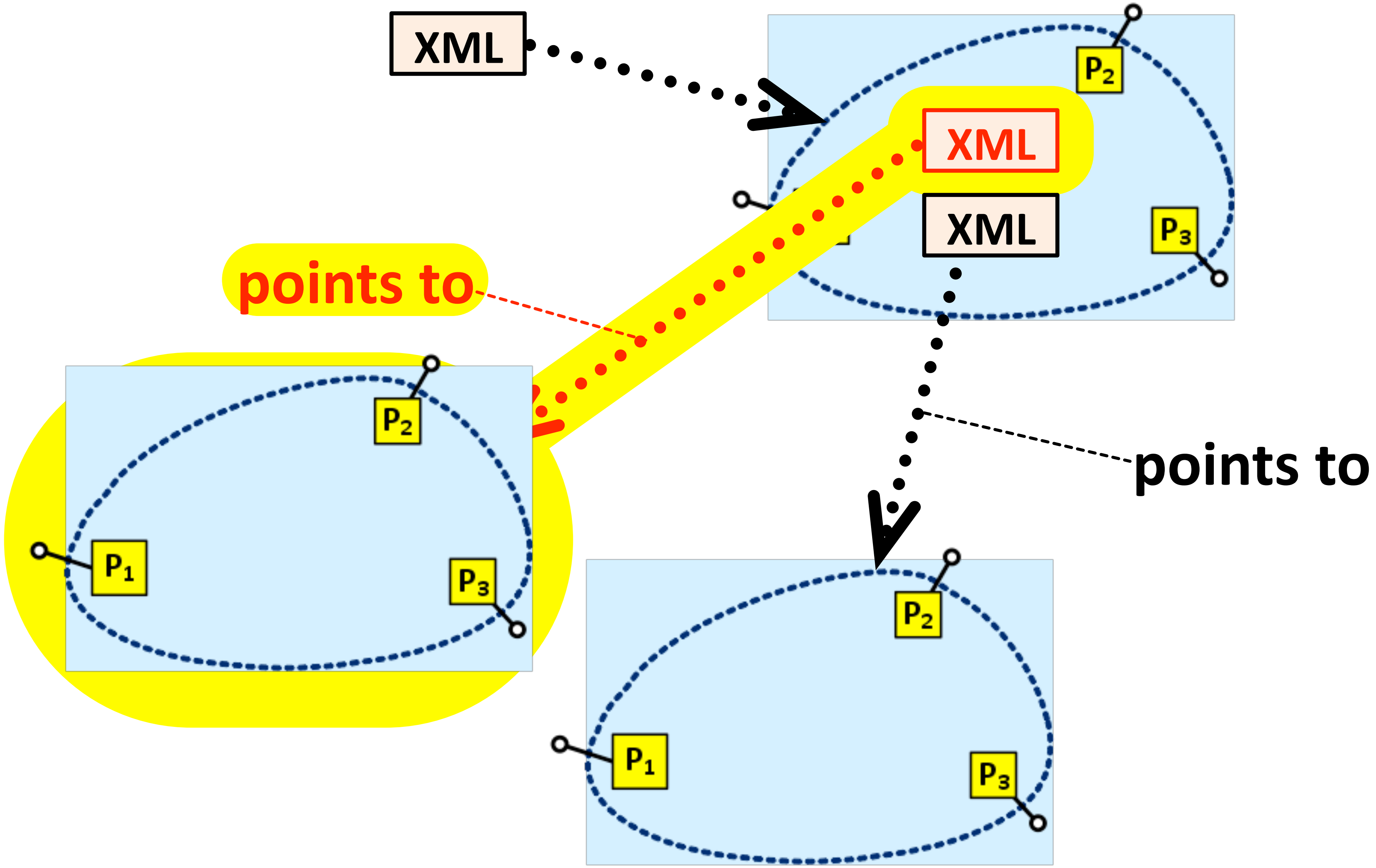
channel reference

points to

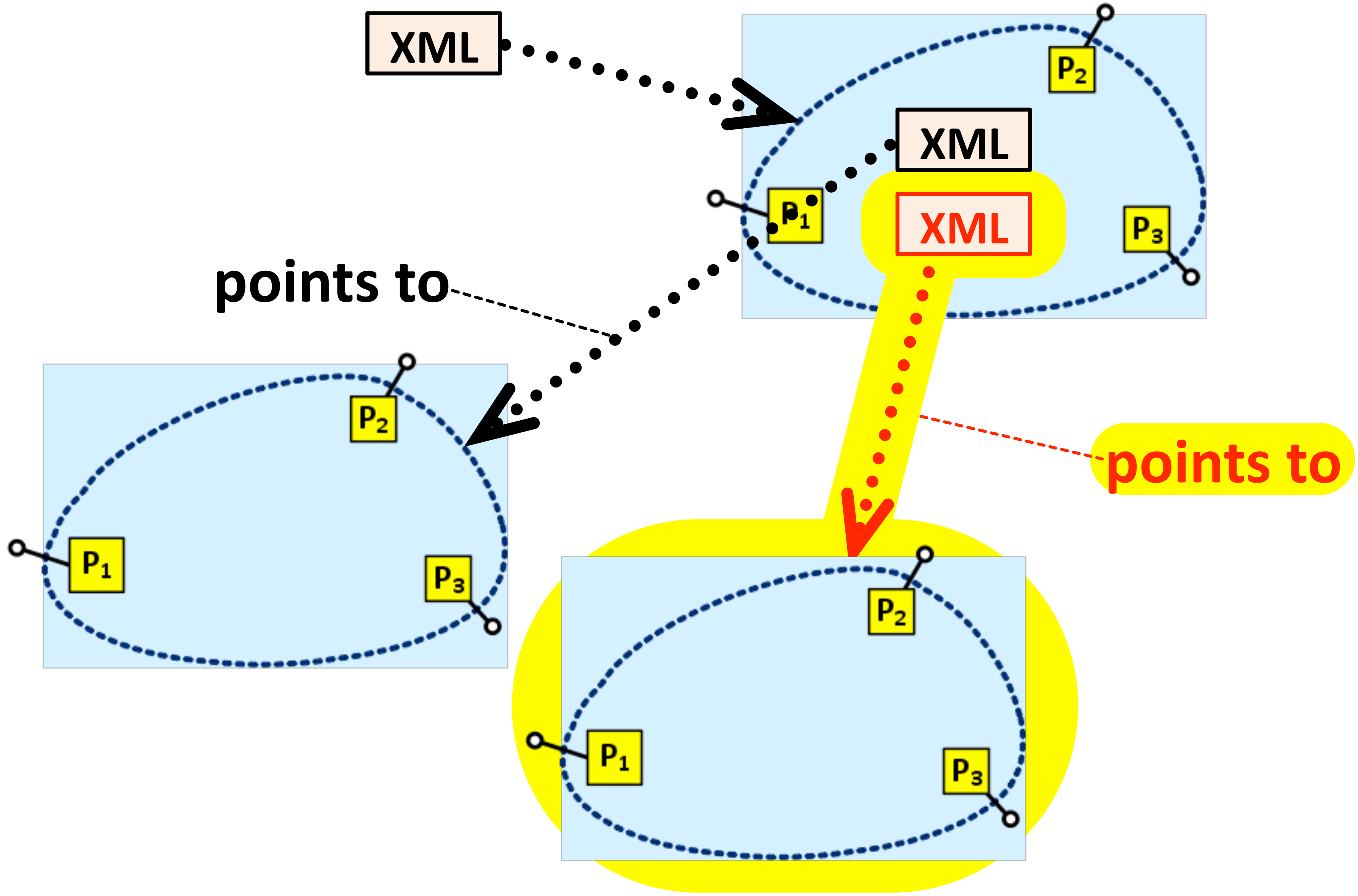
channel proxy

channel content

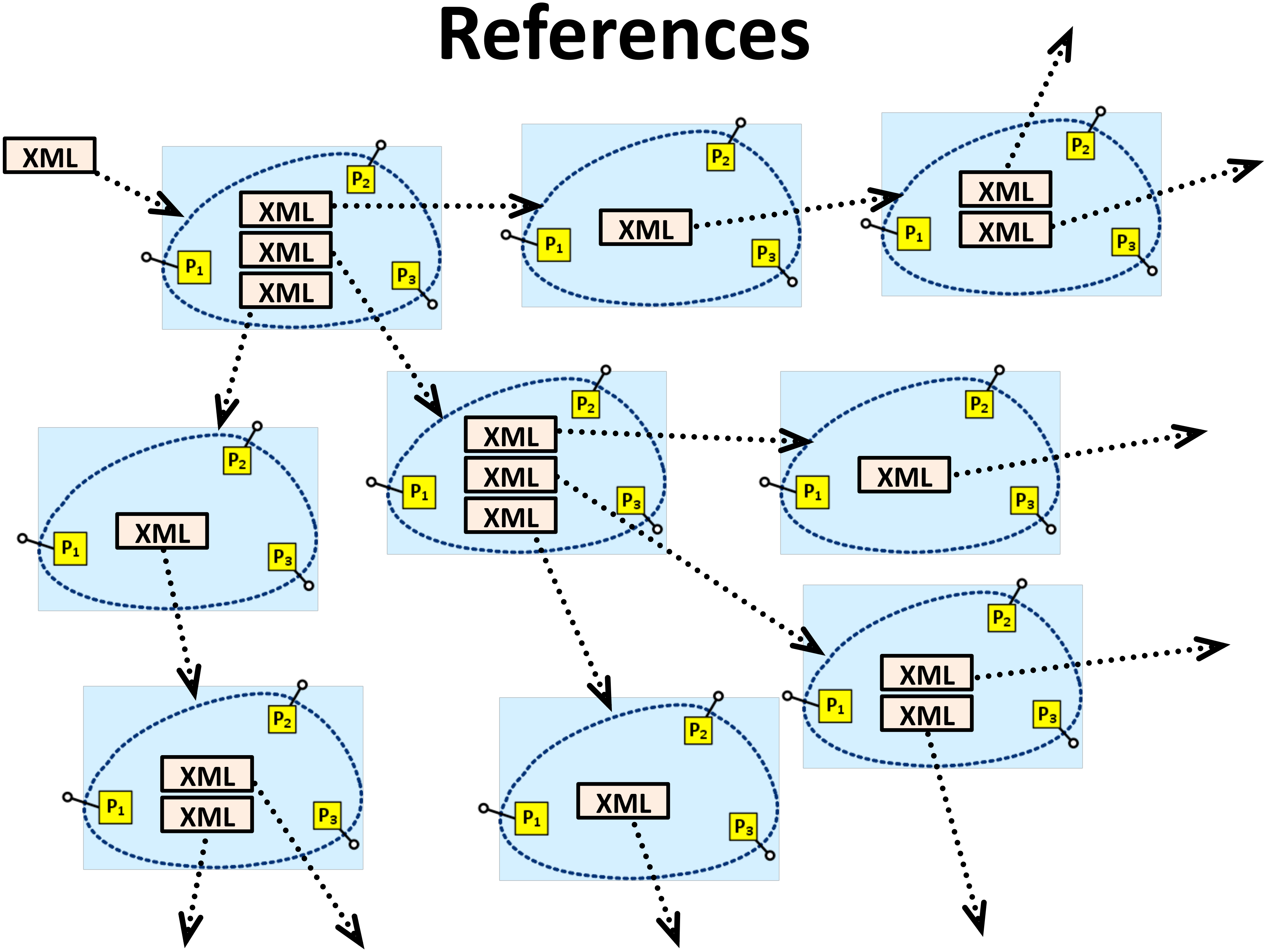
References



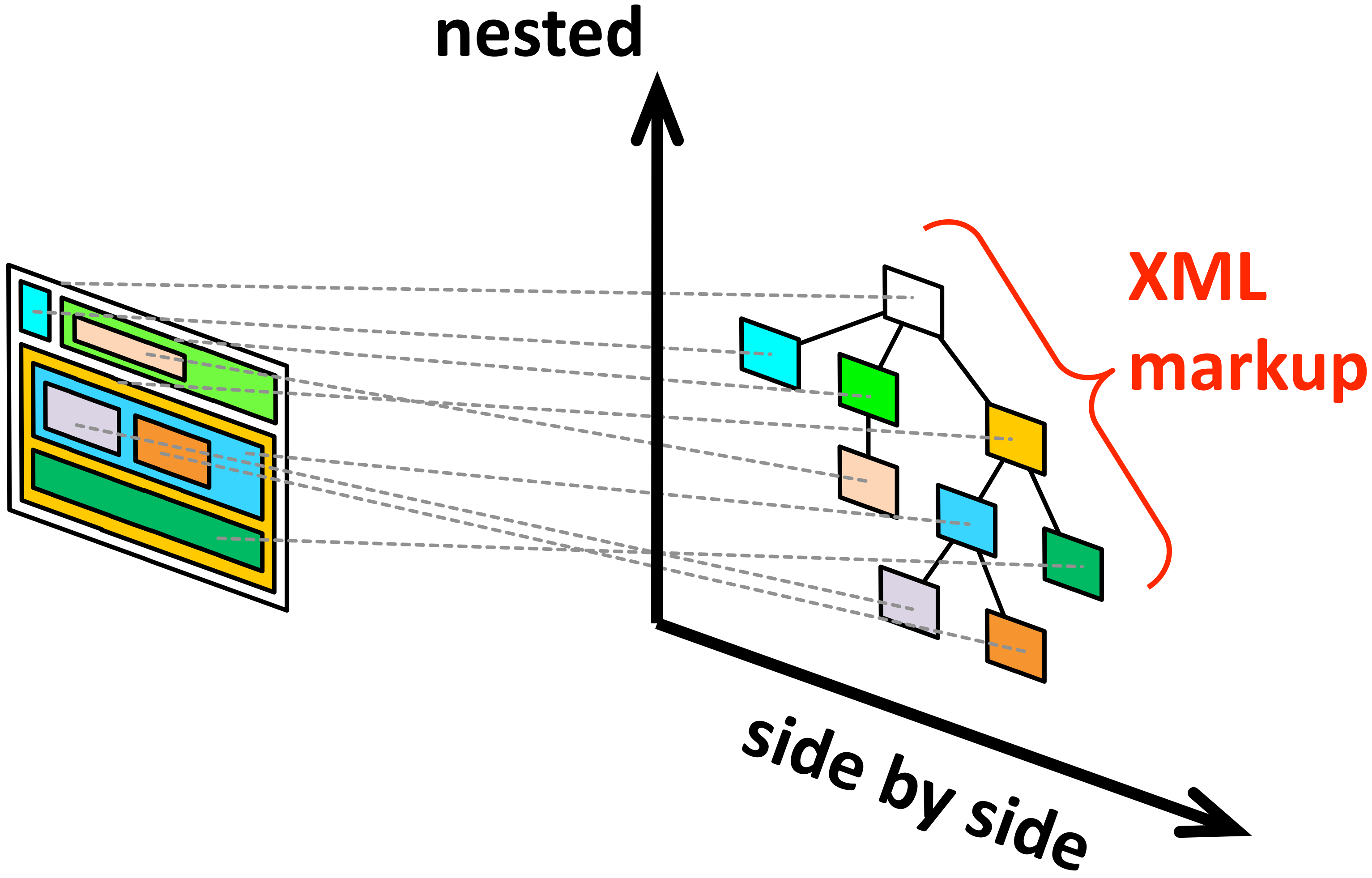
References



References


XML
XML
XML
XML
P1
P2
P3
XML
P1
P2
P3
XML
XML
P1
P2
P3
XML
P1
P2
P3
XML
XML
XML
P1
P2
P3
XML
P1
P2
P3
XML
XML
P1
P2
P3
XML
XML
P1
P2
P3
XML
P1
P2
P3

Ordinary Web Applications

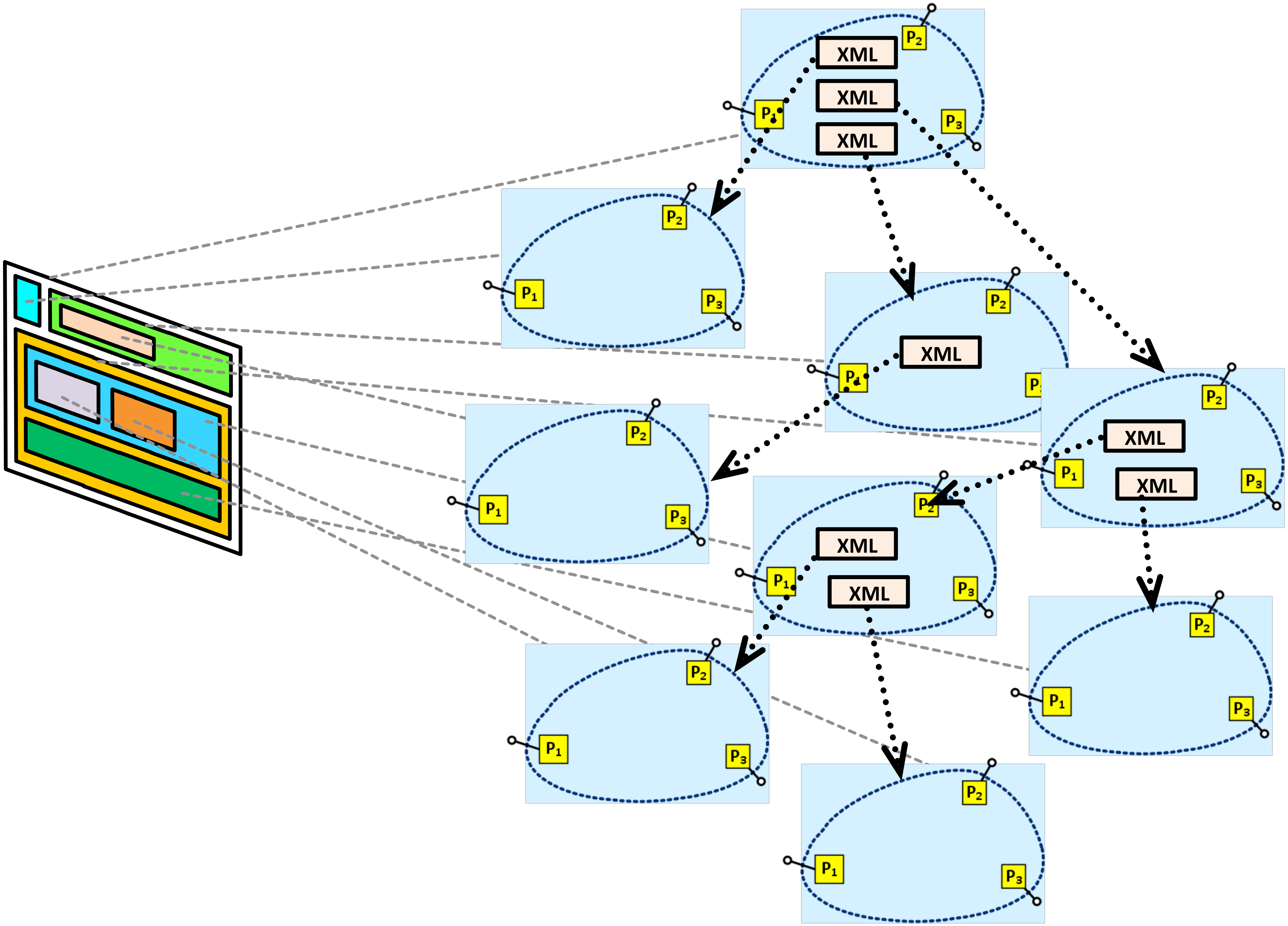


nested

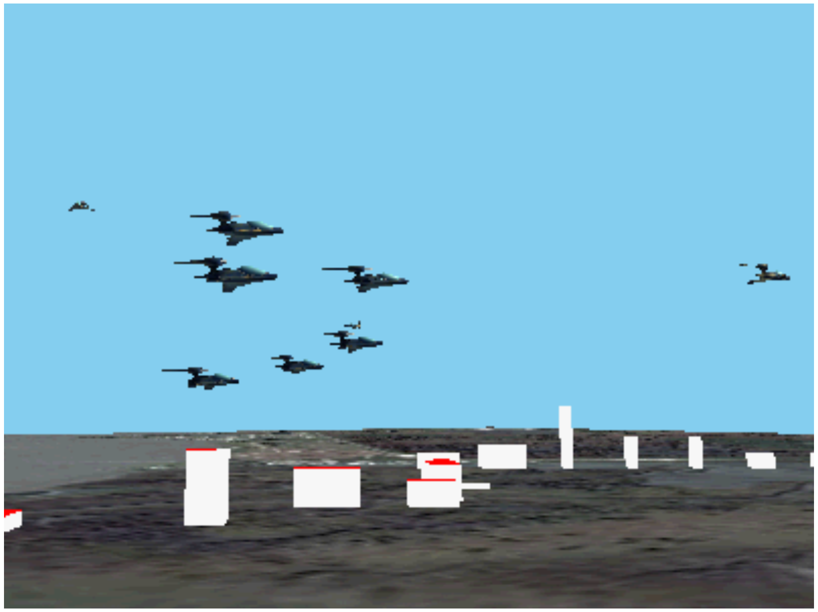
XML markup

side by side

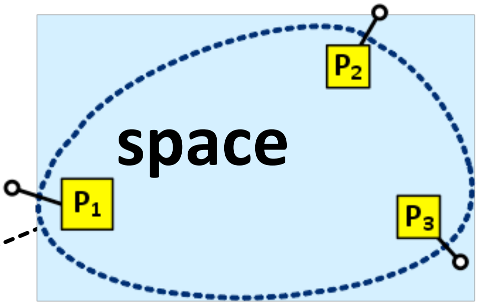
Live Objects Applications



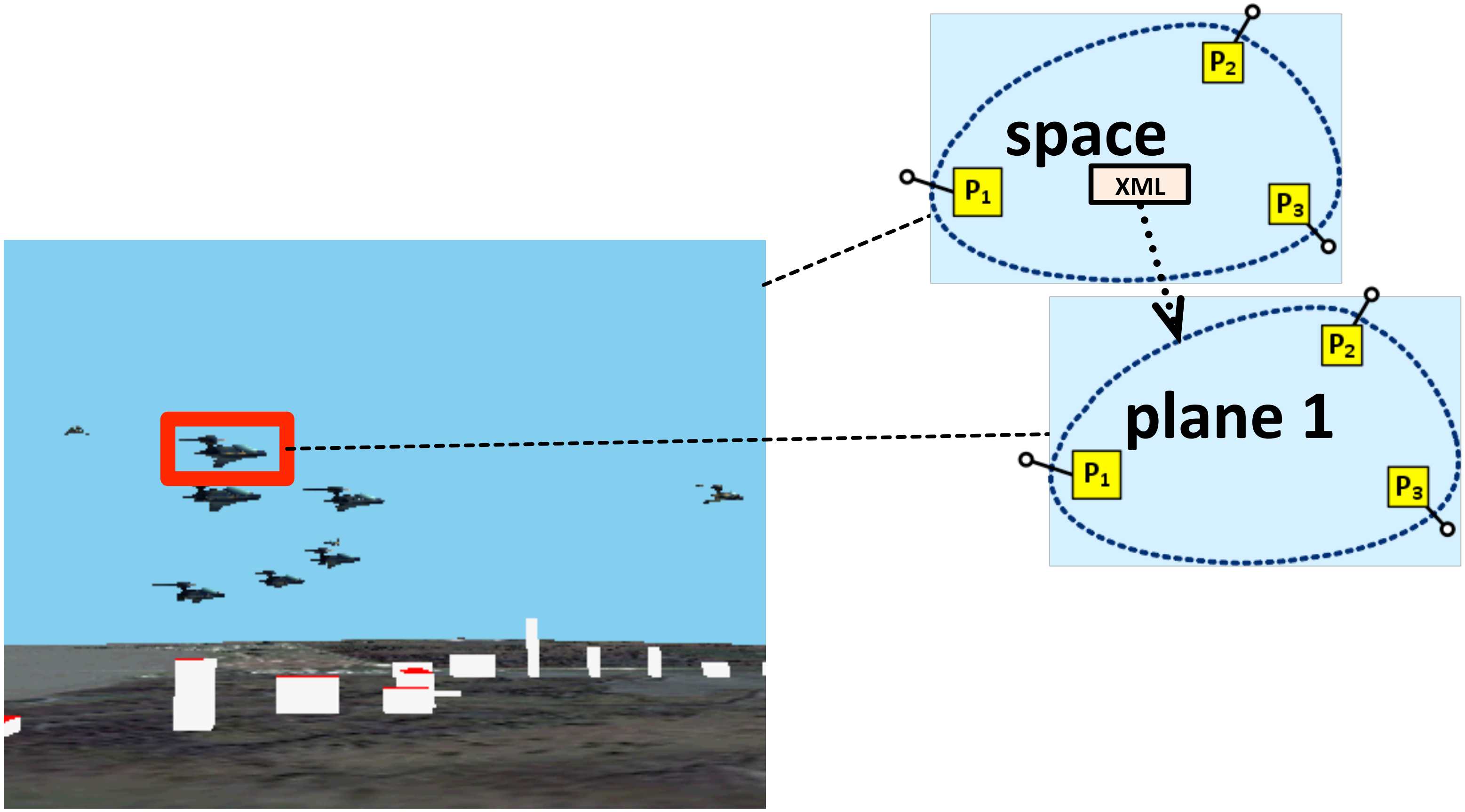
Live Objects Applications



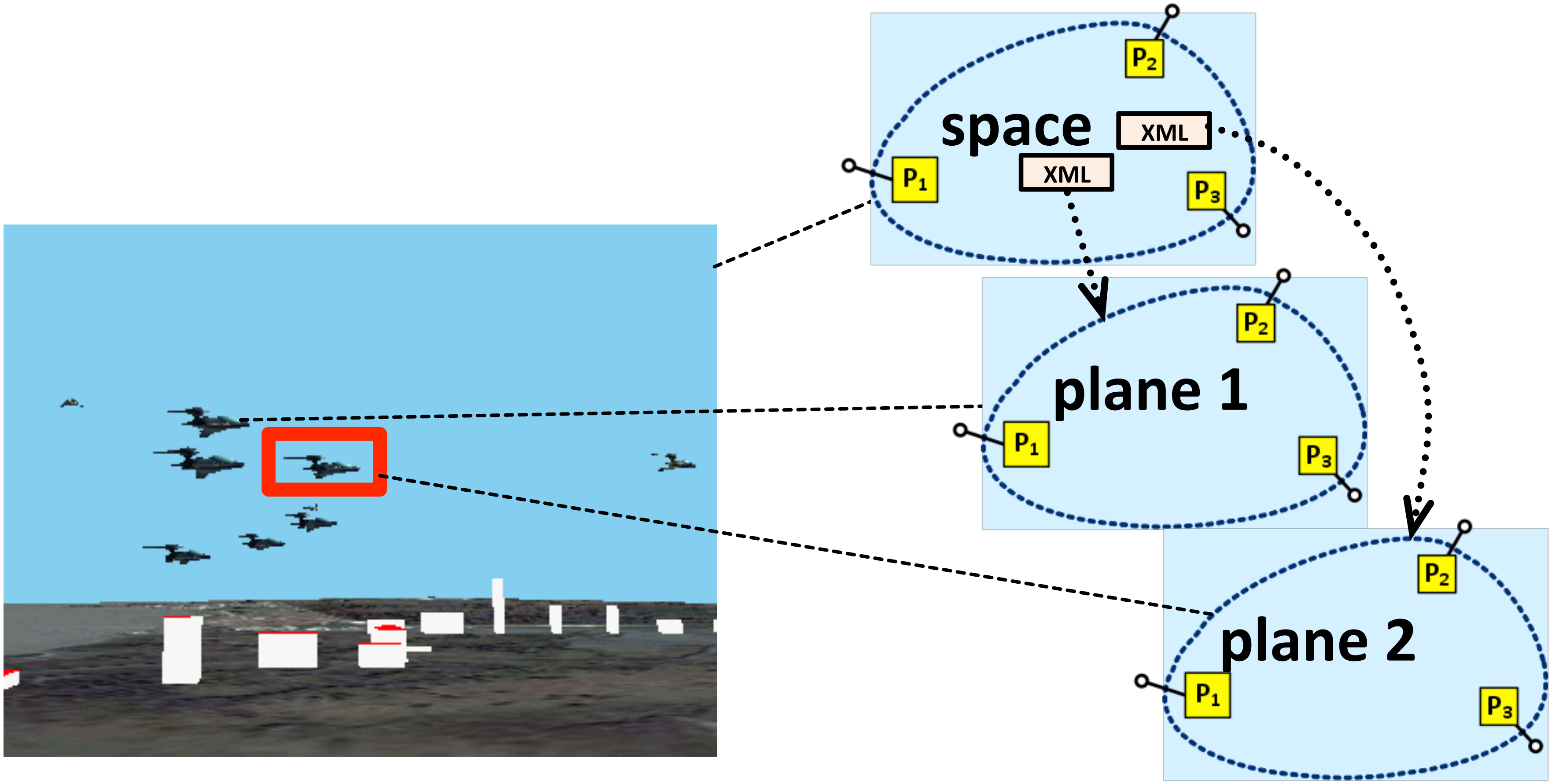
Live Objects Applications



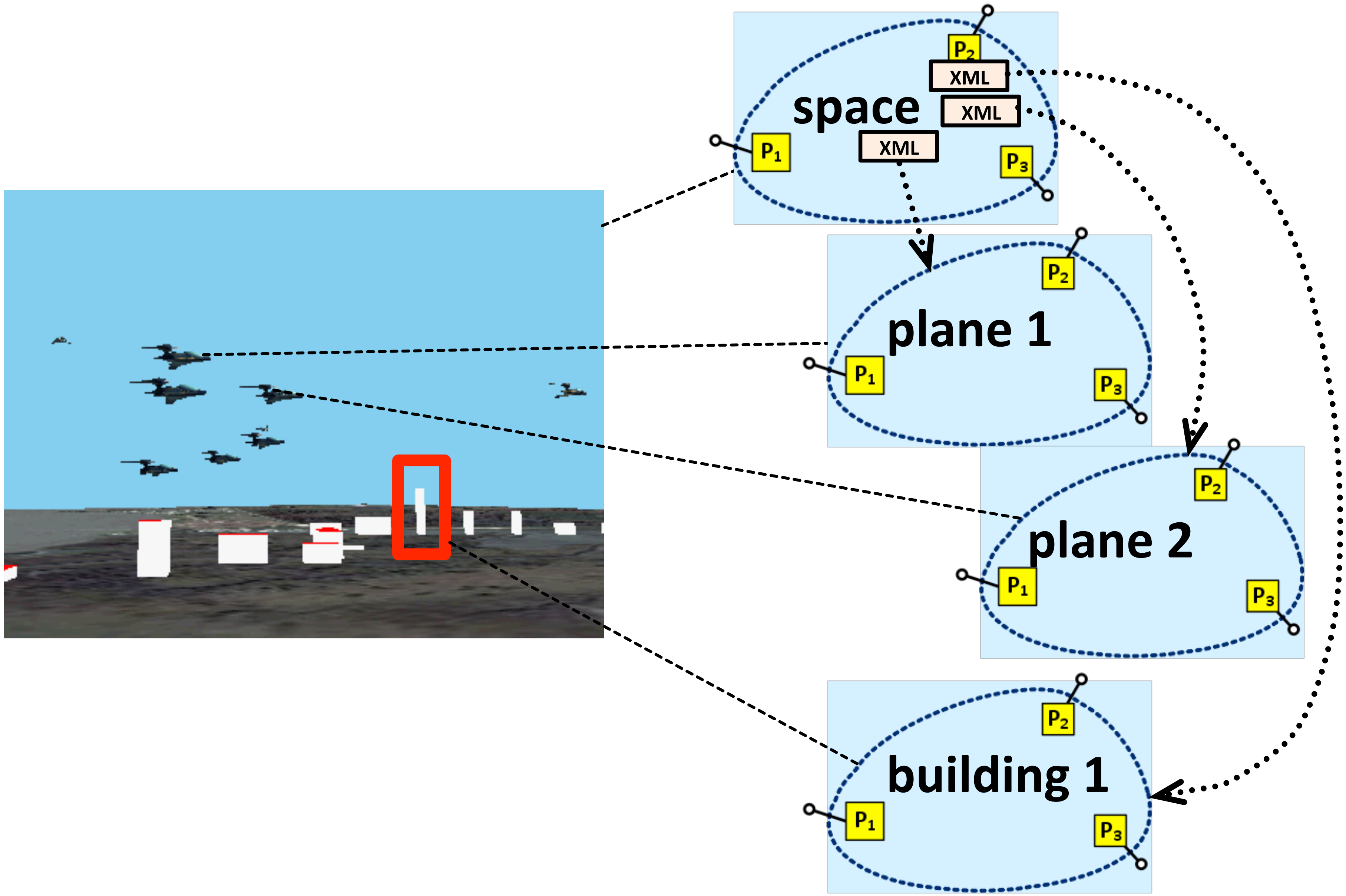
Live Objects Applications



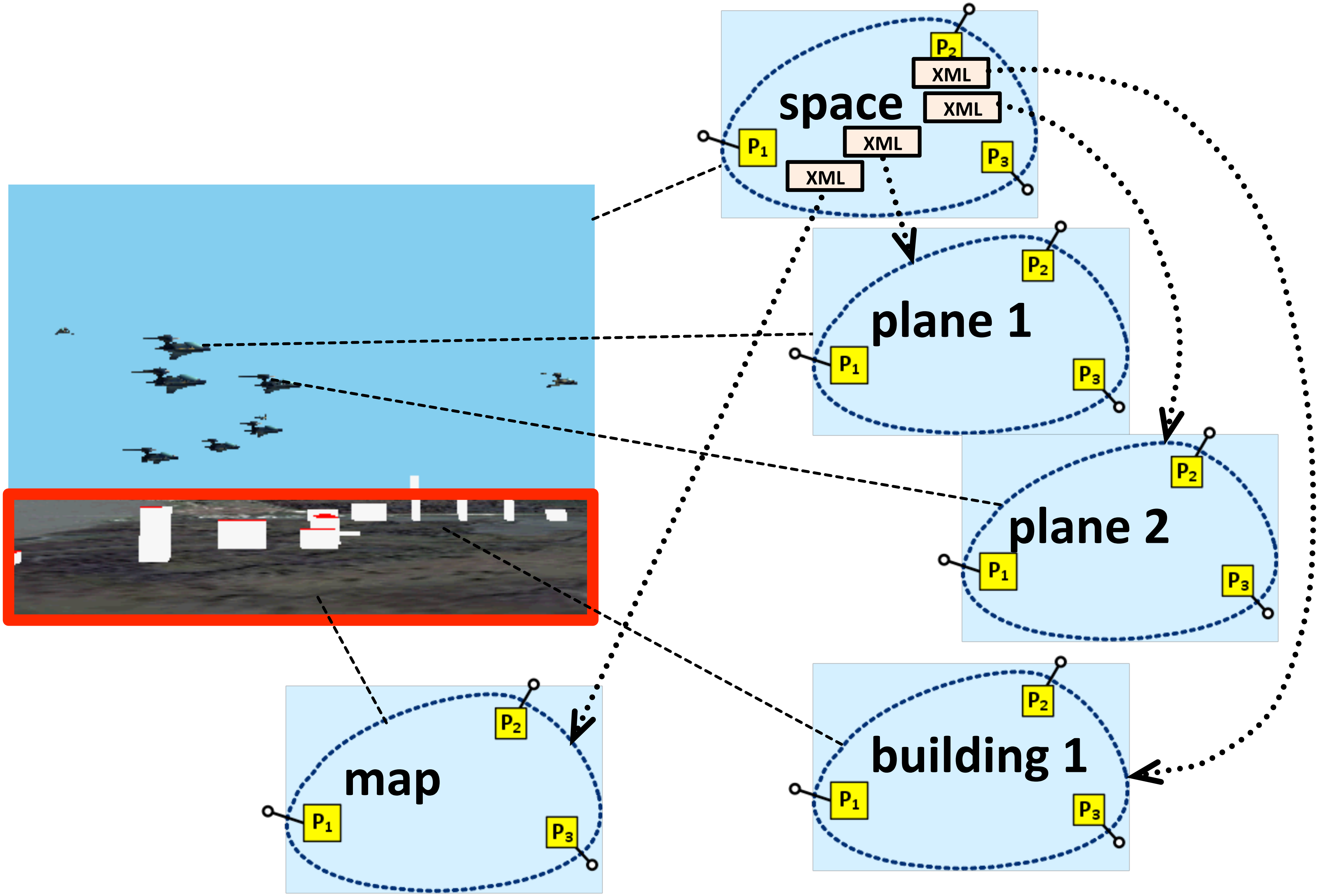
Live Objects Applications



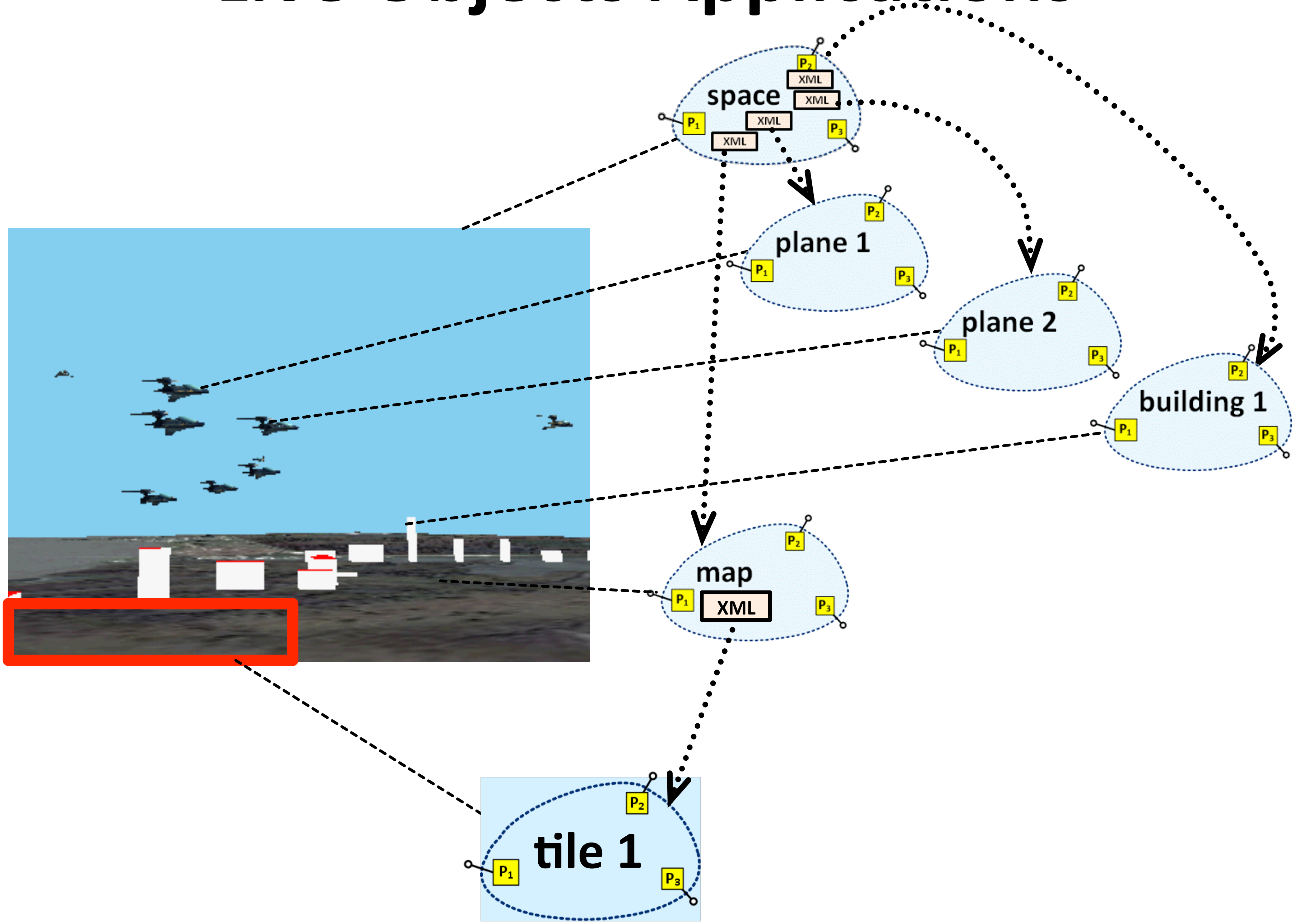
Live Objects Applications



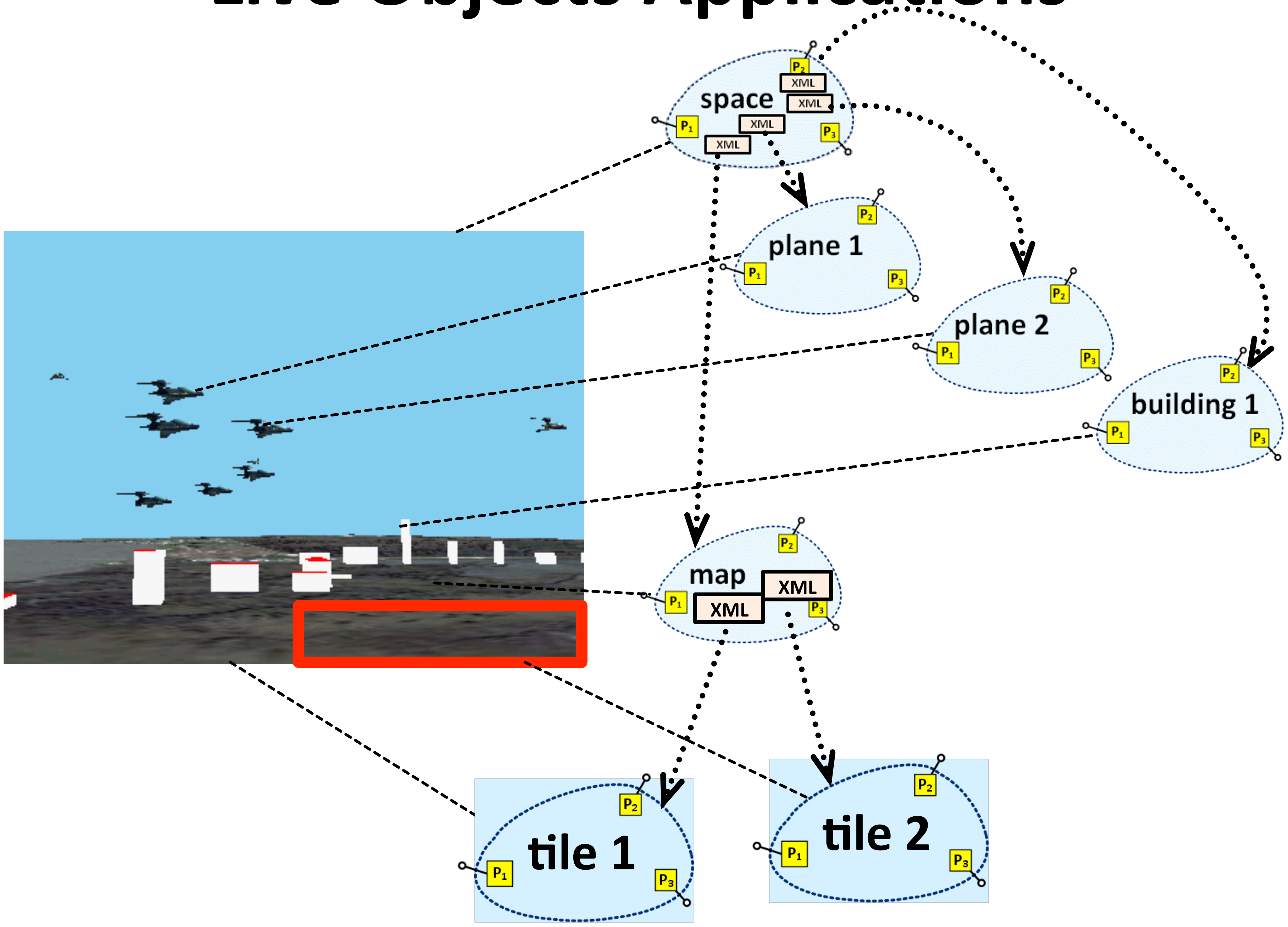
Live Objects Applications



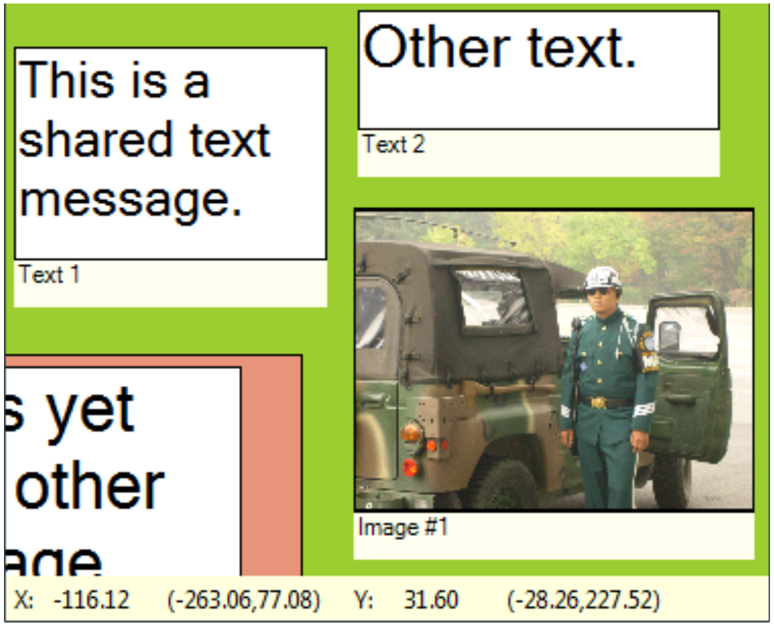
Live Objects Applications



Live Objects Applications



Live Objects Applications



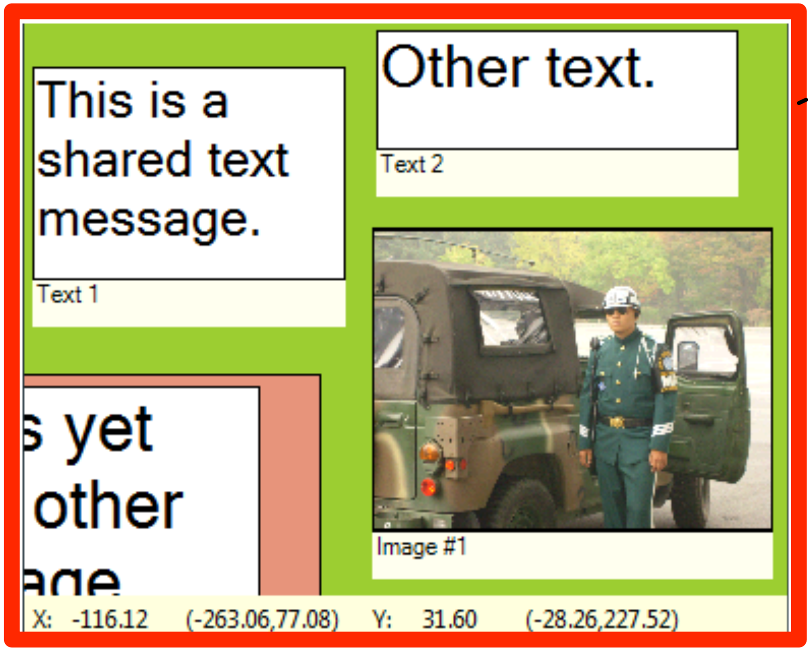
This is a shared text message.

Other text.

s yet
other
age

```
X:   -116.12     (-263.06,77.08)    Y:    31.60      (-28.26,227.52)
```

Live Objects Applications



This is a shared text message.

Text 1

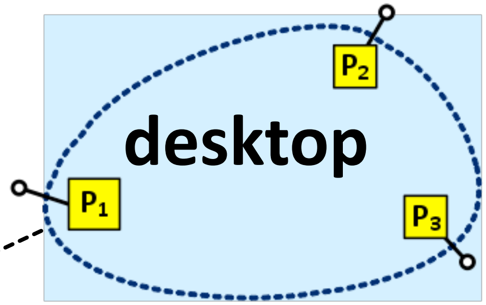
Other text.

Text 2

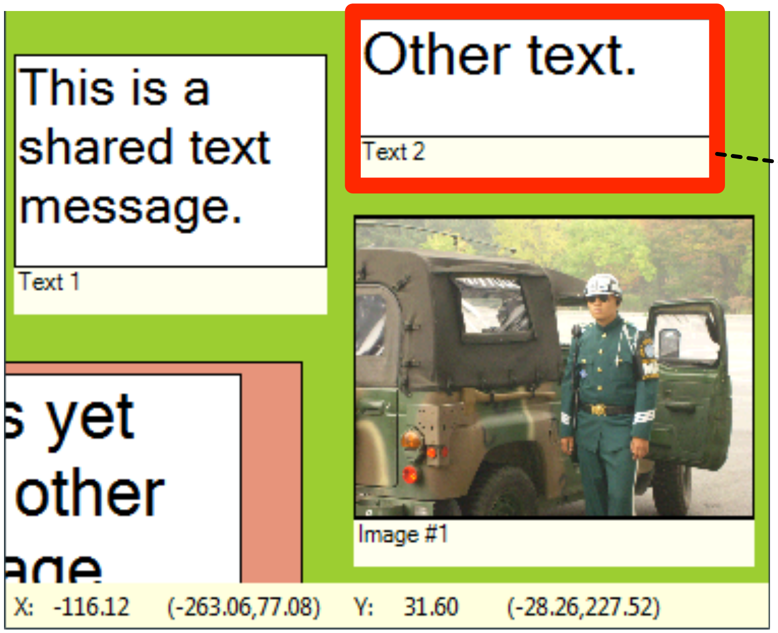
s yet
other
age

Image #1

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)



Live Objects Applications



Other text.

Text 2

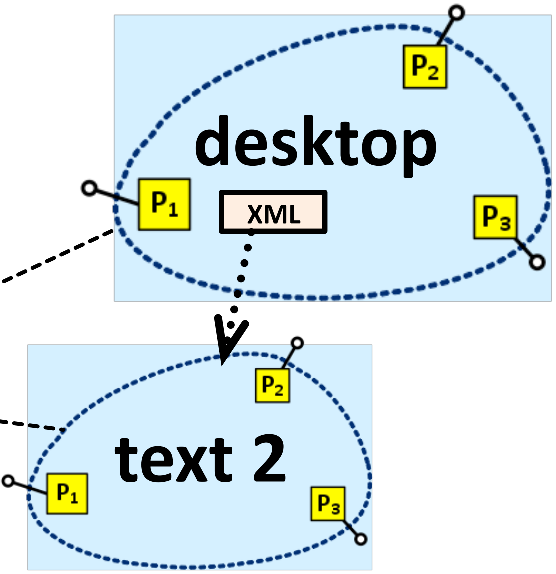
This is a shared text message.

Text 1

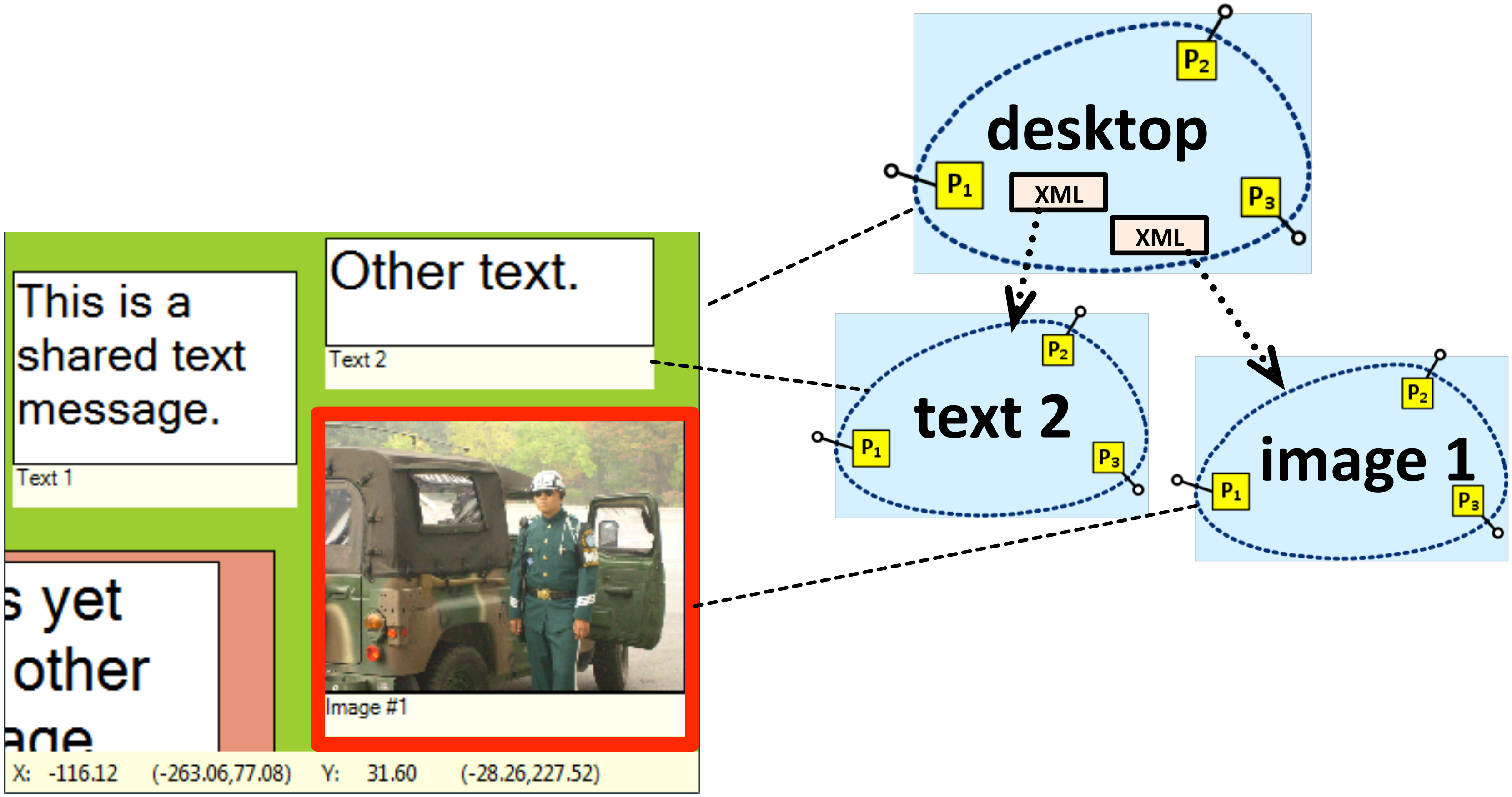
s yet
other
age

Image #1

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)



Live Objects Applications



This is a shared text message.

Text 1

Other text.

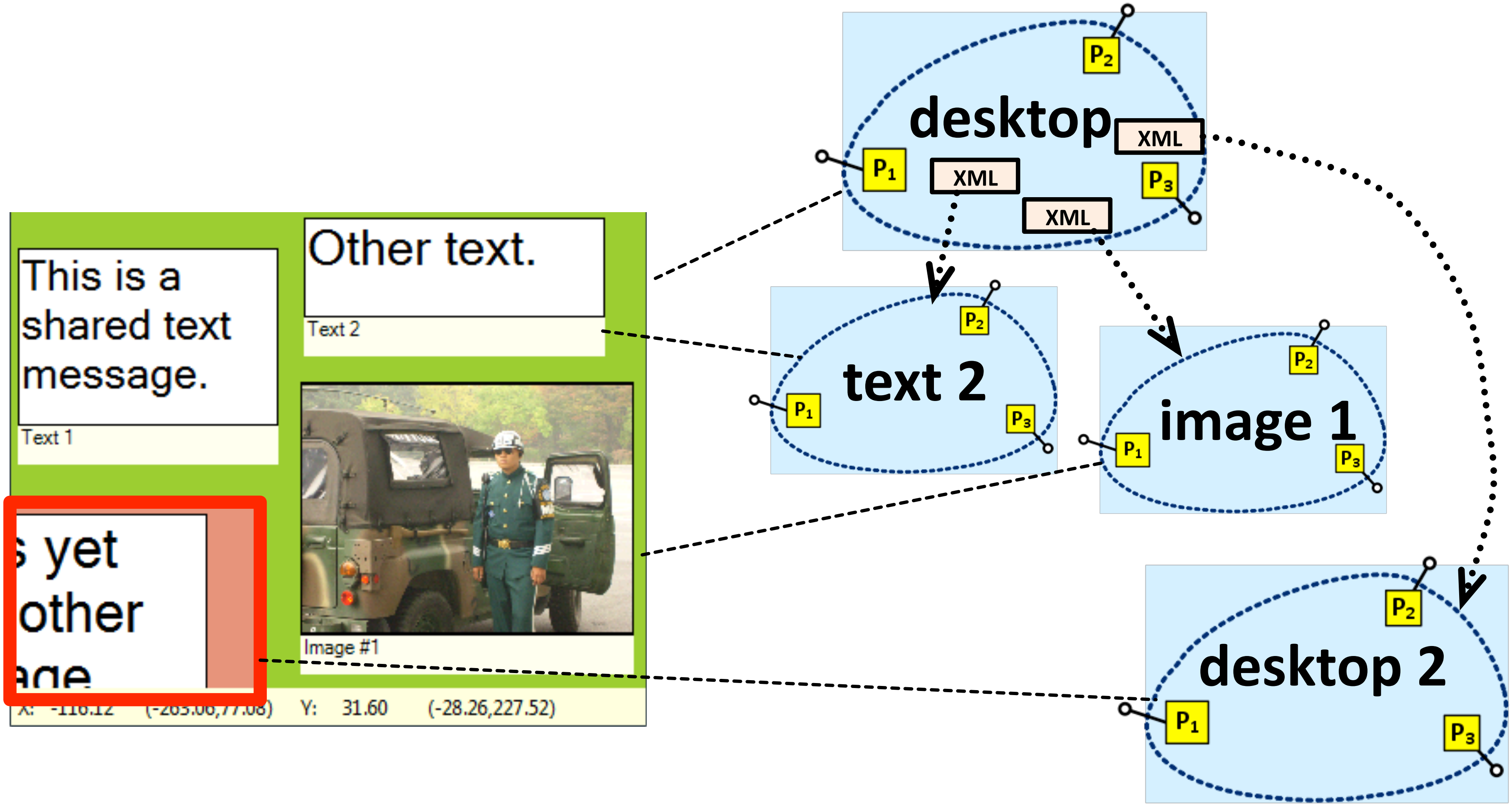
Text 2

s yet other age

Image #1

X: -116.12 (-263.06,77.08) Y: 31.60 (-28.26,227.52)

Live Objects Applications



This is a shared text message.

Text 1

Other text.

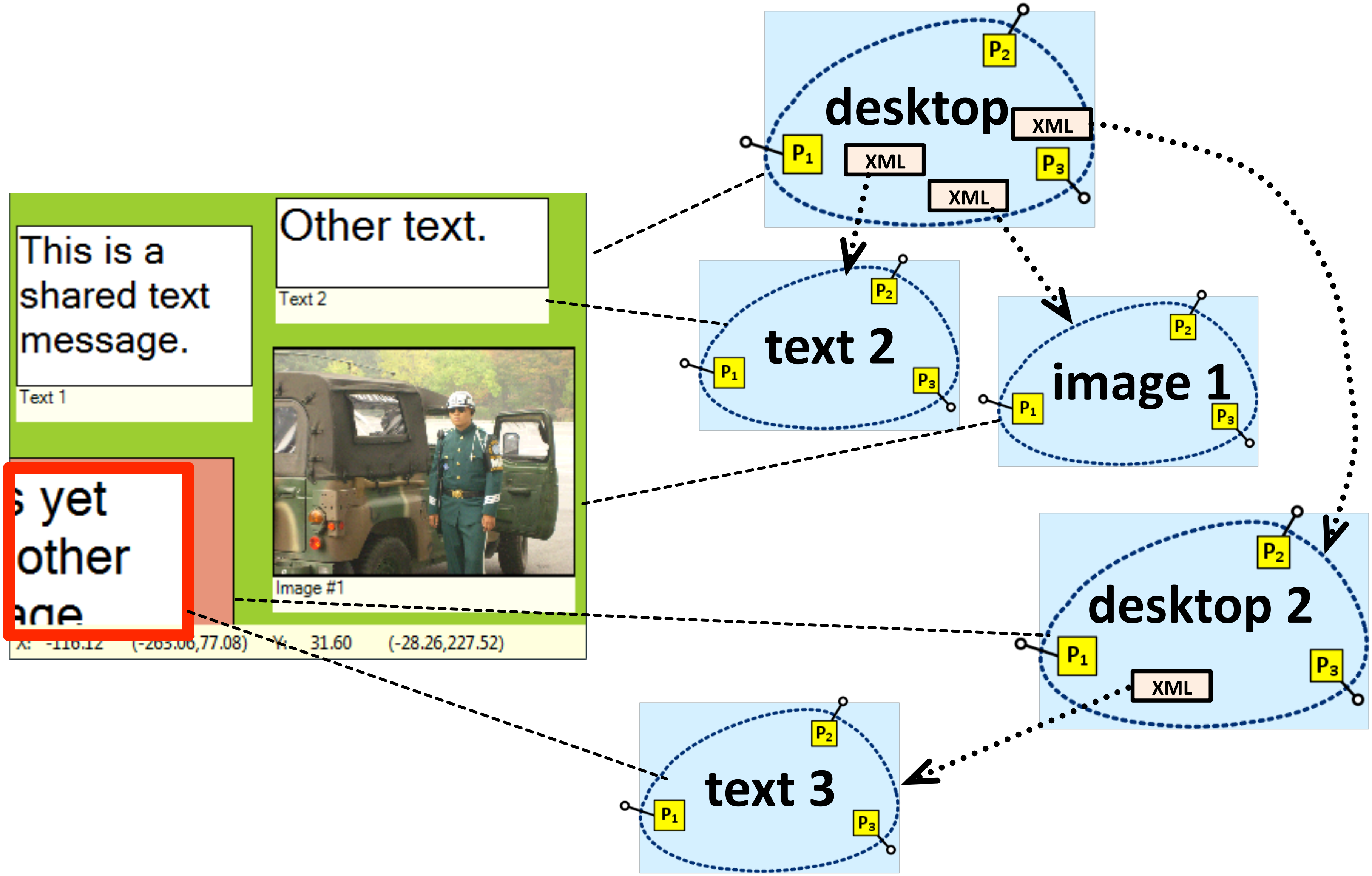
Text 2

s yet
other
age

Image #1

X: -116.12 (-283.06,77.08) Y: 31.60 (-28.26,227.52)

Live Objects Applications



This is a shared text message.

Text 1

Other text.

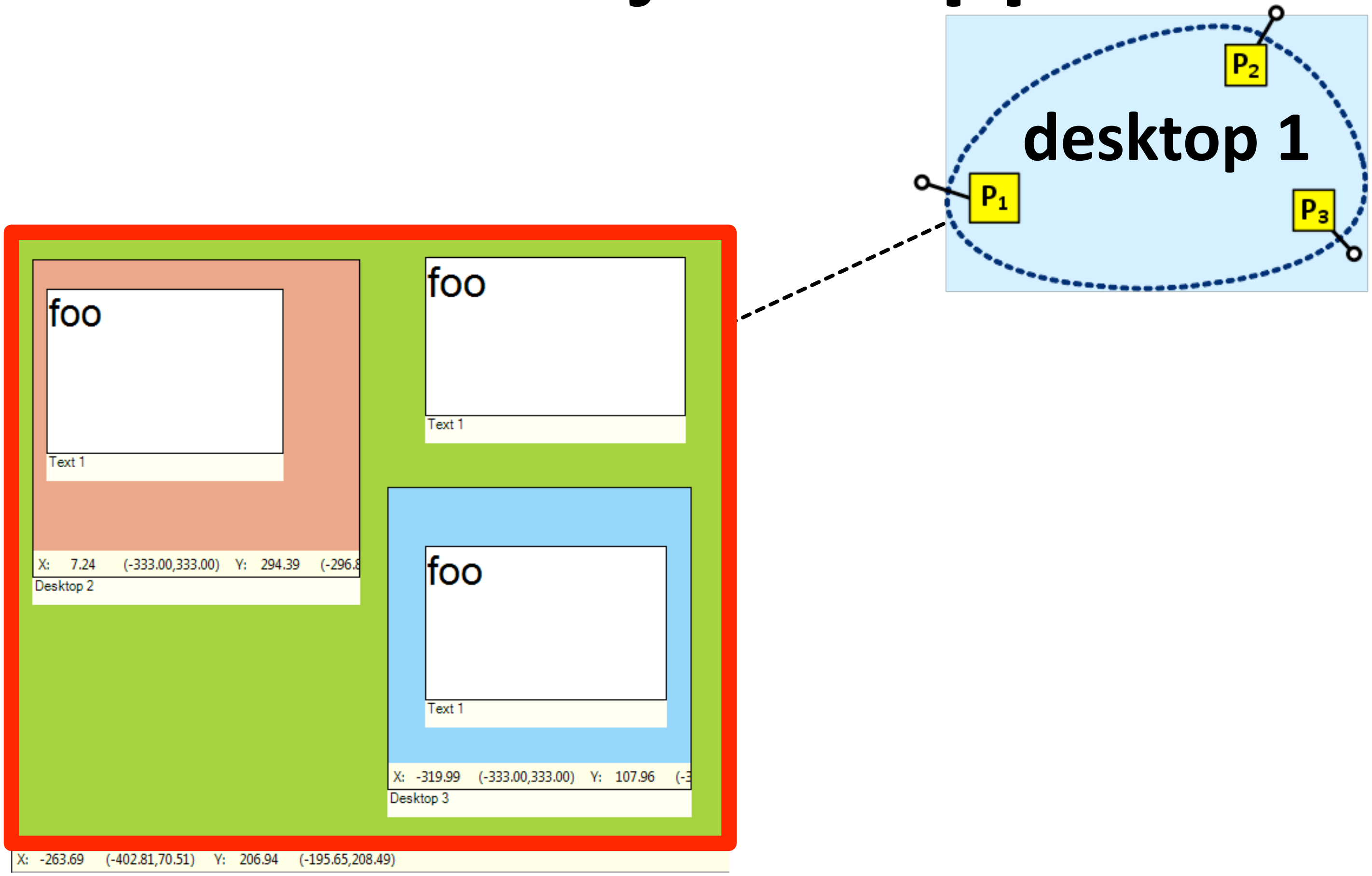
Text 2

Image #1

s yet other age

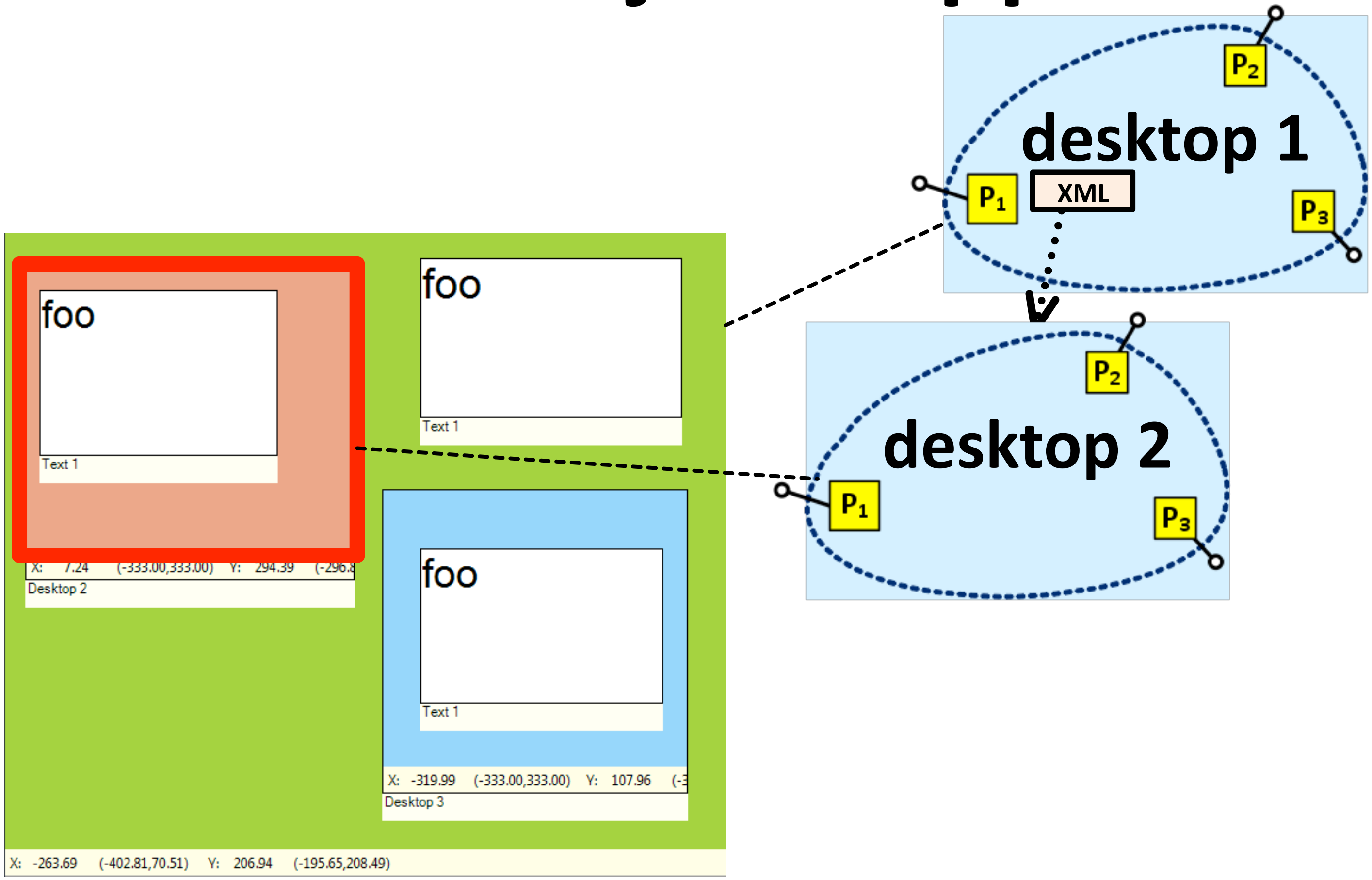
X: -116.12 (-283.06,77.08) Y: 31.60 (-28.26,227.52)

Live Objects Applications

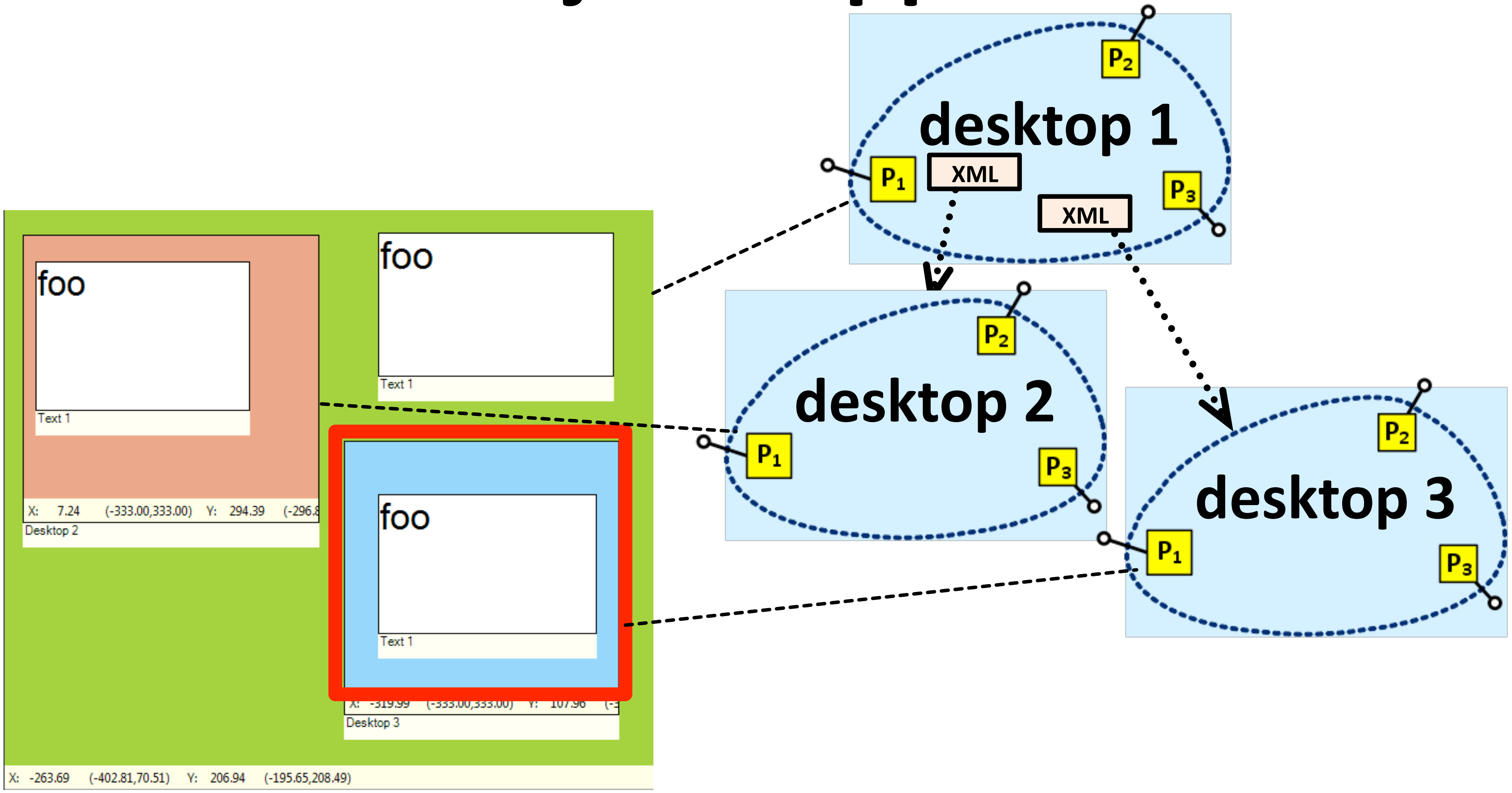


```
X:    7.24     (-333.00,333.00)   Y:   294.39     (-296.8
X:  -319.99    (-333.00,333.00)   Y:   107.96     (-3
X:  -263.69    (-402.81,70.51)    Y:   206.94     (-195.65,208.49)
```

Live Objects Applications


desktop 1
P1
P2
P3
XML
desktop 2
P1
P2
P3
foo
Text 1
X: 7.24 (-333.00,333.00) Y: 294.39 (-296.8
Desktop 2
foo
Text 1
foo
Text 1
X: -319.99 (-333.00,333.00) Y: 107.96 (-3
Desktop 3
X: -263.69 (-402.81,70.51) Y: 206.94 (-195.65,208.49)

Live Objects Applications

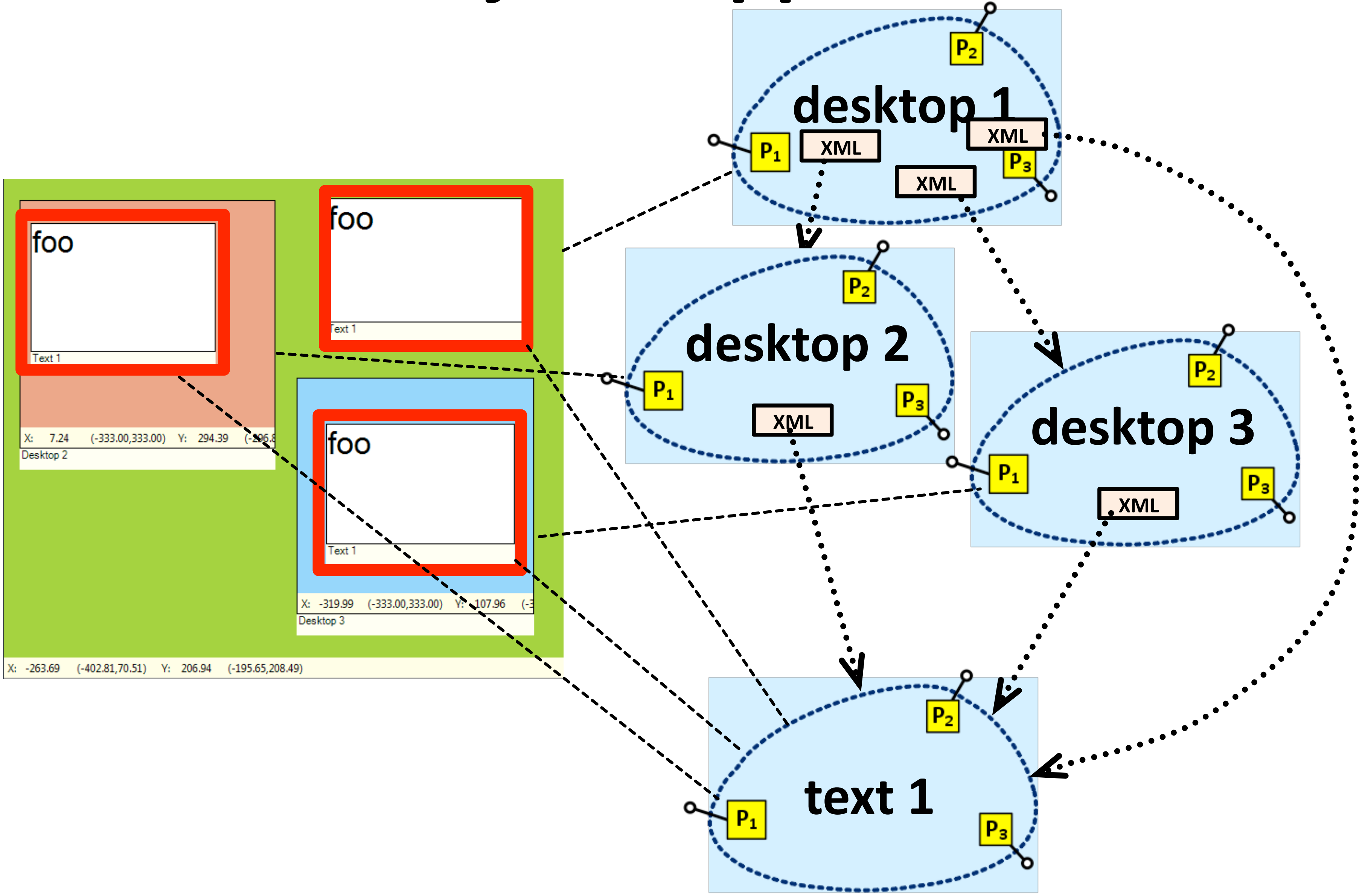


X: 7.24 (-333.00,333.00) Y: 294.39 (-296.8

X: -319.99 (-333.00,333.00) Y: 107.96 (-3

X: -263.69 (-402.81,70.51) Y: 206.94 (-195.65,208.49)

Live Objects Applications



foo

Text 1

X: 7.24 (-333.00,333.00) Y: 294.39 (-296.8

Desktop 2

foo

Text 1

foo

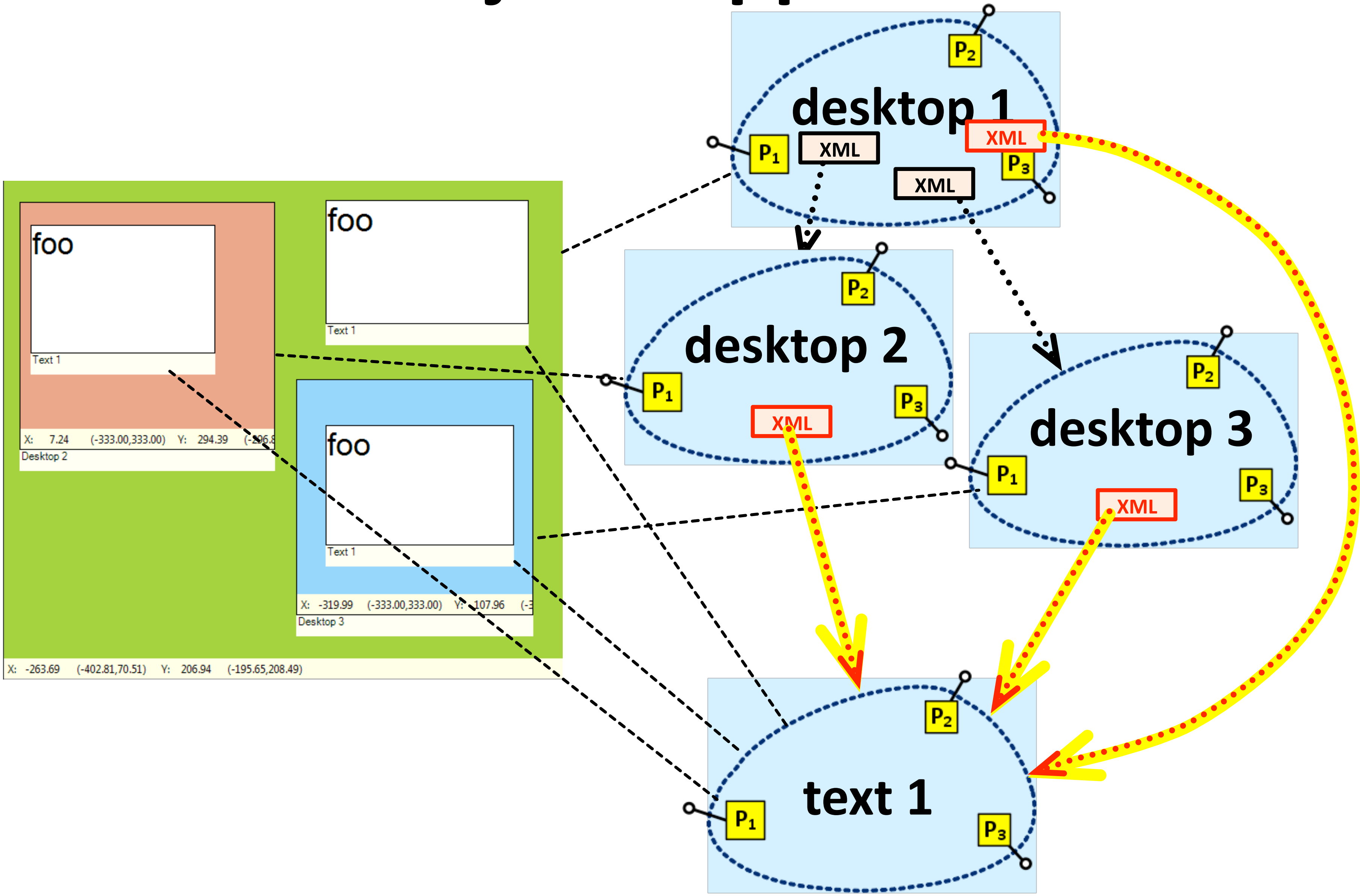
Text 1

X: -319.99 (-333.00,333.00) Y: 107.96 (-3

Desktop 3

X: -263.69 (-402.81,70.51) Y: 206.94 (-195.65,208.49)

Live Objects Applications


foo
Text 1
X: 7.24 (-333.00,333.00) Y: 294.39 (-296.8
Desktop 2
foo
Text 1
foo
Text 1
X: -319.99 (-333.00,333.00) Y: 107.96 (-3
Desktop 3
X: -263.69 (-402.81,70.51) Y: 206.94 (-195.65,208.49)
desktop 1
P1
P2
P3
XML
XML
XML
desktop 2
P1
P2
P3
XML
desktop 3
P1
P2
P3
XML
text 1
P1
P2
P3

Conclusions

Conclusions

1. Cloud and edge technologies can work together…

   …but we need a new storage abstraction.

2. Checkpointed Channels (CCs)

   a) Support a variety of protocols: client-server, P2P, distributed replication, append log files, etc.,

   b) Can be used at any level of the protocol stack, at the fronted and at the backend,

   c) Can efficiently store structured mashup content,

   d) Can be hyperlinked into webs of CCs; these could serve as a basis for a new Web architecture.

Thanks