

Marketplace

Layered pub/sub networks in Racket

tonyg@ccs.neu.edu, 29 Sep 2013

An operating system is a collection of things that don't fit into a language. There shouldn't be one.

– *Dan Ingalls, "Design Principles Behind Smalltalk", 1981*

VMs: A Browser Application

JavaScript

VMs: A Browser Application

Javascript
Per-page JSVM instance

VMs: A Browser Application

Javascript
Per-page JSVM instance
Browser process

VMs: A Browser Application

Javascript
Per-page JSVM instance
Browser process
Linux kernel

VMs: A Browser Application

Javascript
Per-page JS VM instance
Browser process
Linux kernel
Virtual hardware

VMs: A Browser Application

Javascript
Per-page JS VM instance
Browser process
Linux kernel
Virtual hardware
OS X process

VMs: A Browser Application

Javascript
Per-page JS VM instance
Browser process
Linux kernel
Virtual hardware
OS X process
OS X kernel

VMs: A Browser Application

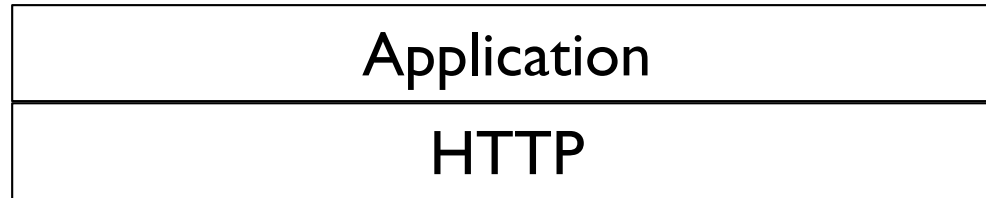
Javascript
Per-page JS VM instance
Browser process
Linux kernel
Virtual hardware
OS X process
OS X kernel
Hardware

Networks: A Typical Packet



Application

Networks: A Typical Packet



Networks: A Typical Packet

Application
HTTP
TCP

Networks: A Typical Packet

Application
HTTP
TCP
IP

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP
GTP

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP
GTP
UDP

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP
GTP
UDP
IP

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP
GTP
UDP
IP
MPLS

Networks: A Typical Packet

Application
HTTP
TCP
IP
IPsec
IP
GTP
UDP
IP
MPLS
MPLS

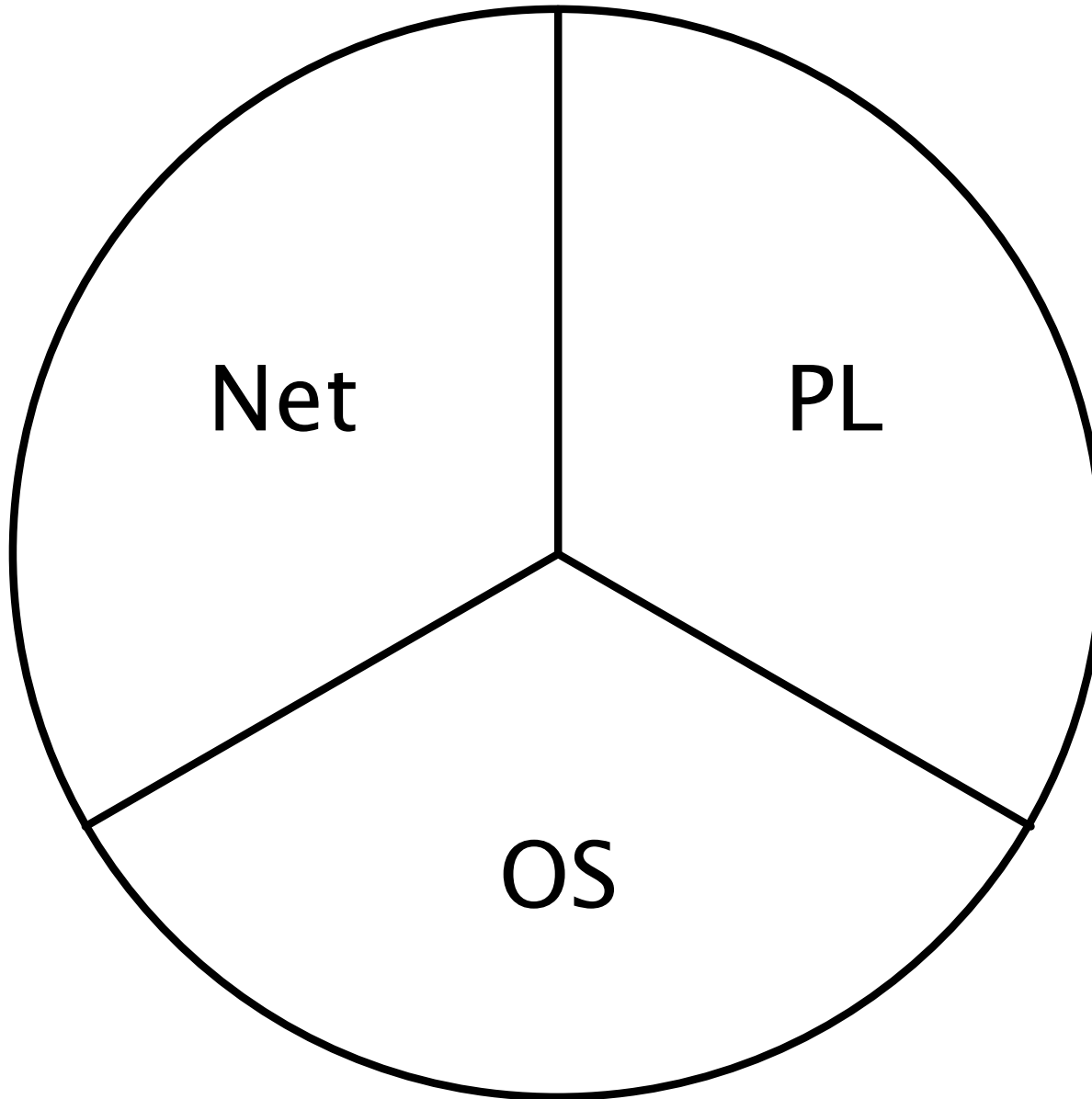
Networks: A Typical Packet

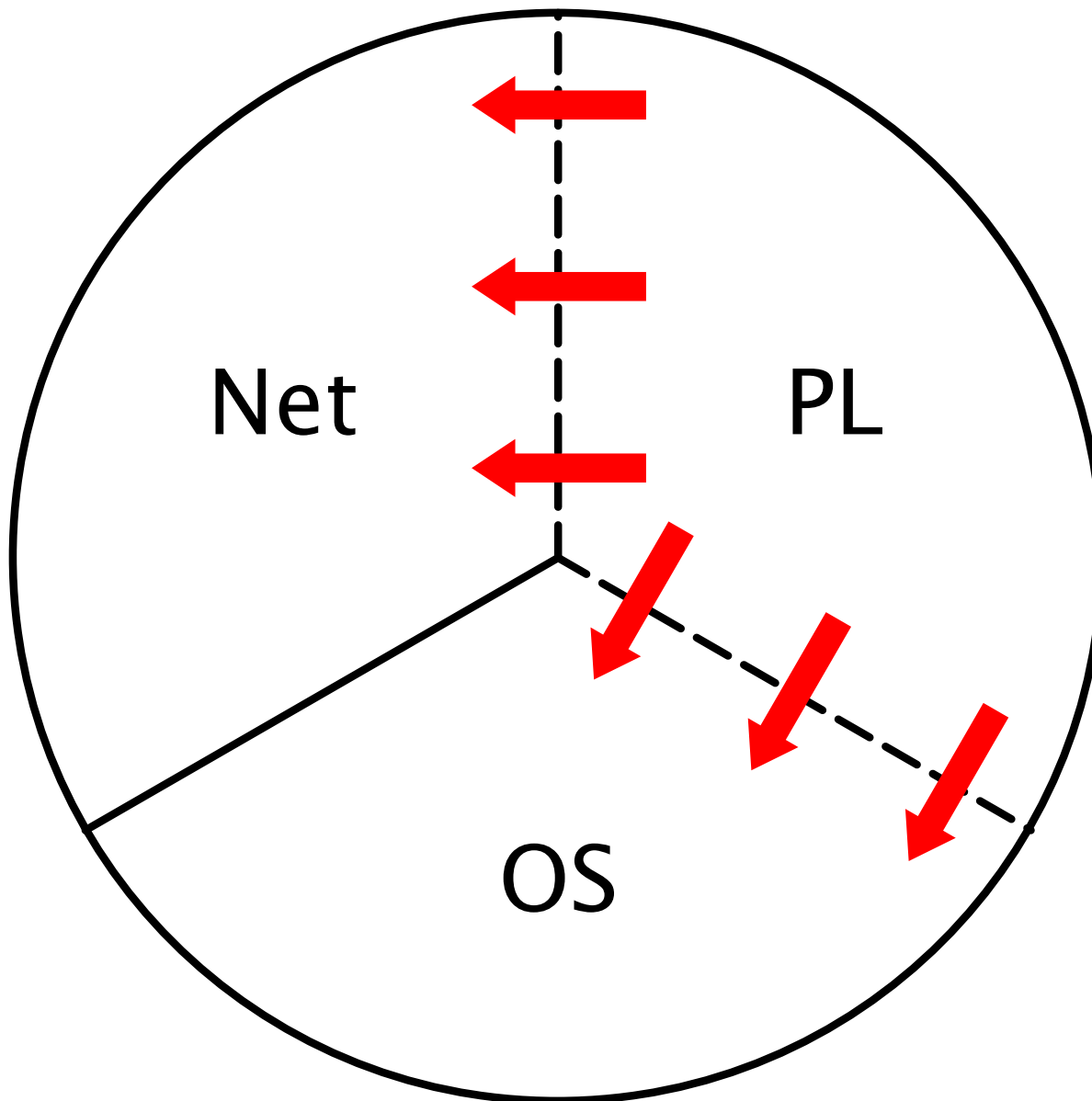
Application
HTTP
TCP
IP
IPsec
IP
GTP
UDP
IP
MPLS
MPLS
Ethernet

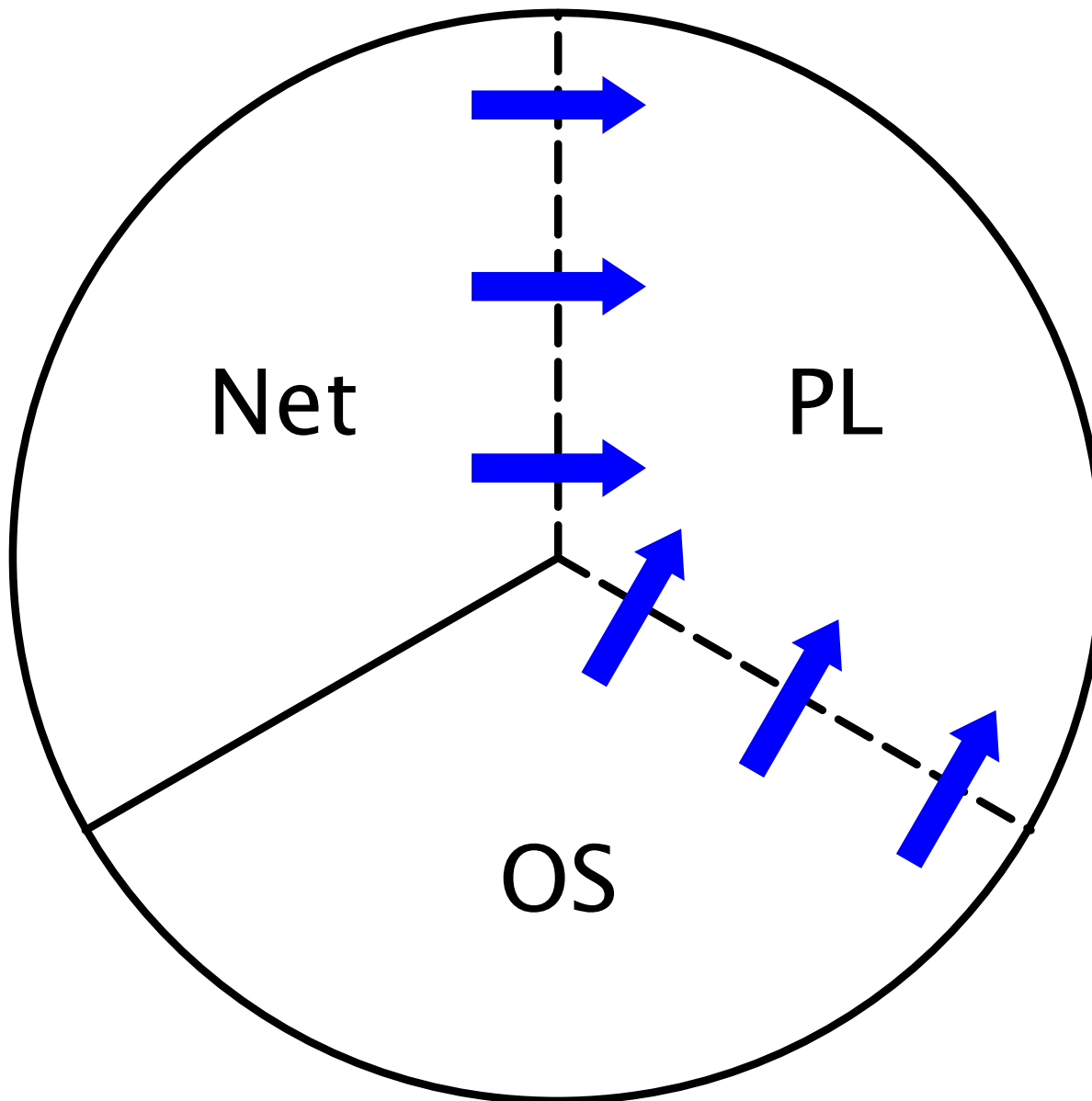
Layers do a lot for you

- Modular system
- One protocol per layer
- One naming system per layer
- (De)multiplex flows

Application
HTTP
TCP
IP
...

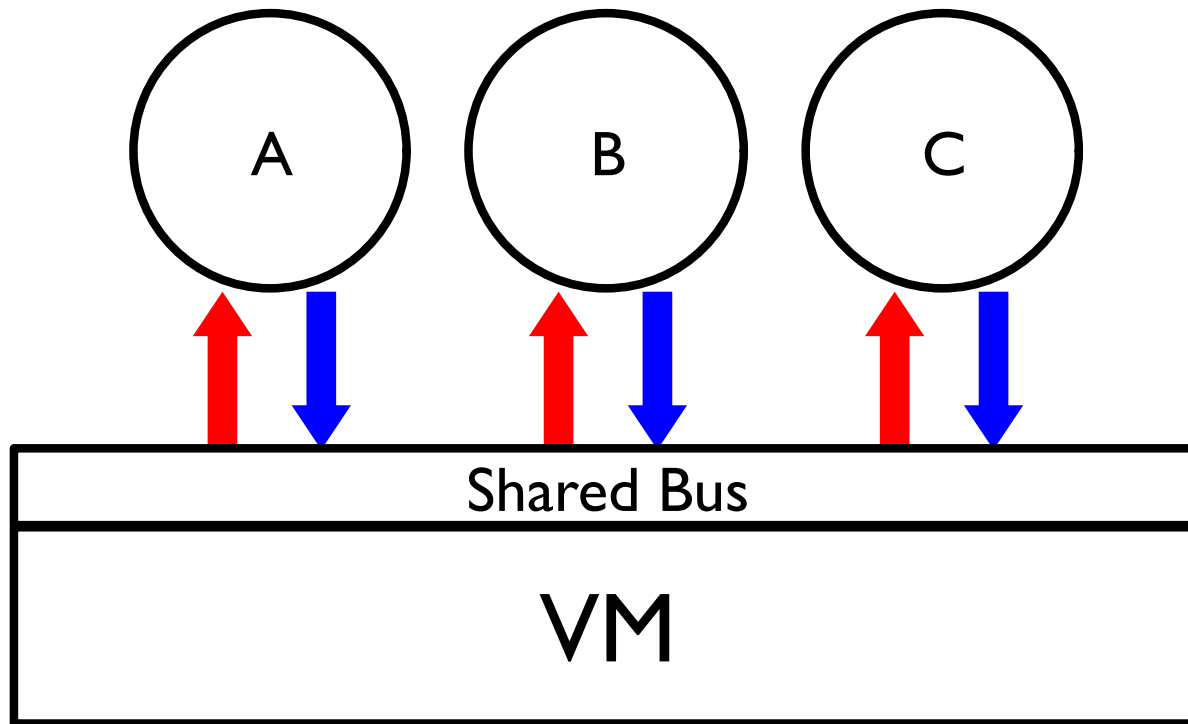




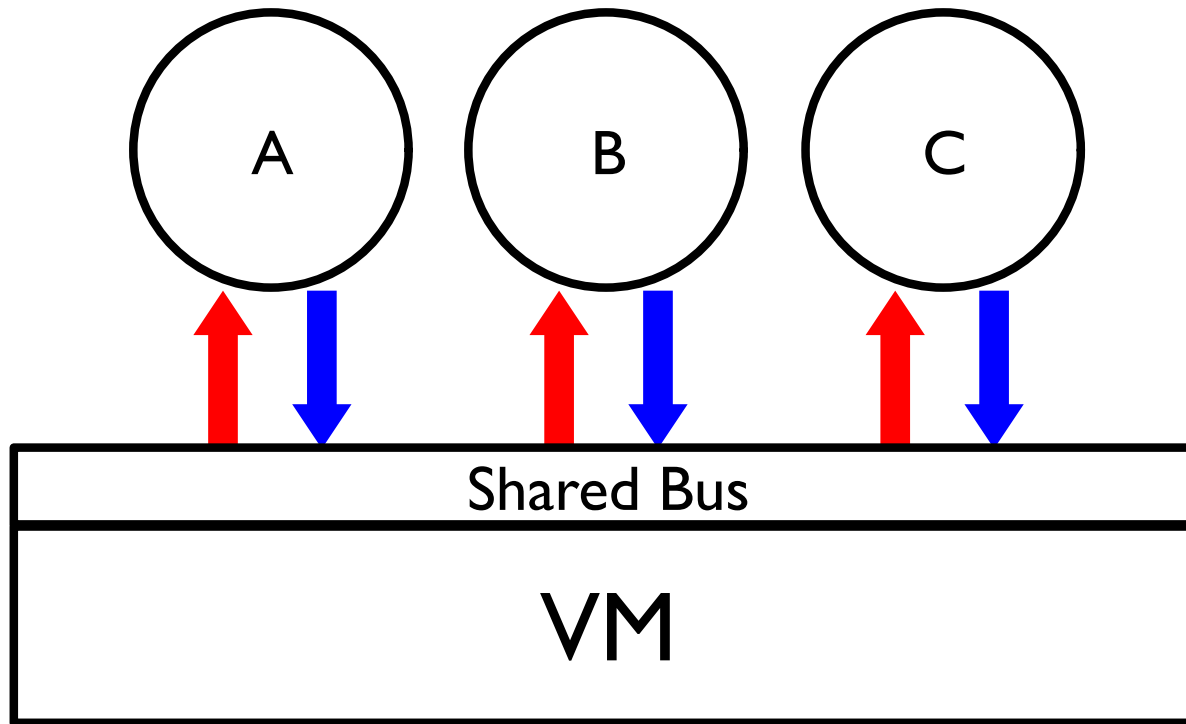


actor : Event × State → [Action] × State

actor : Event \times State \rightarrow [Action] \times State



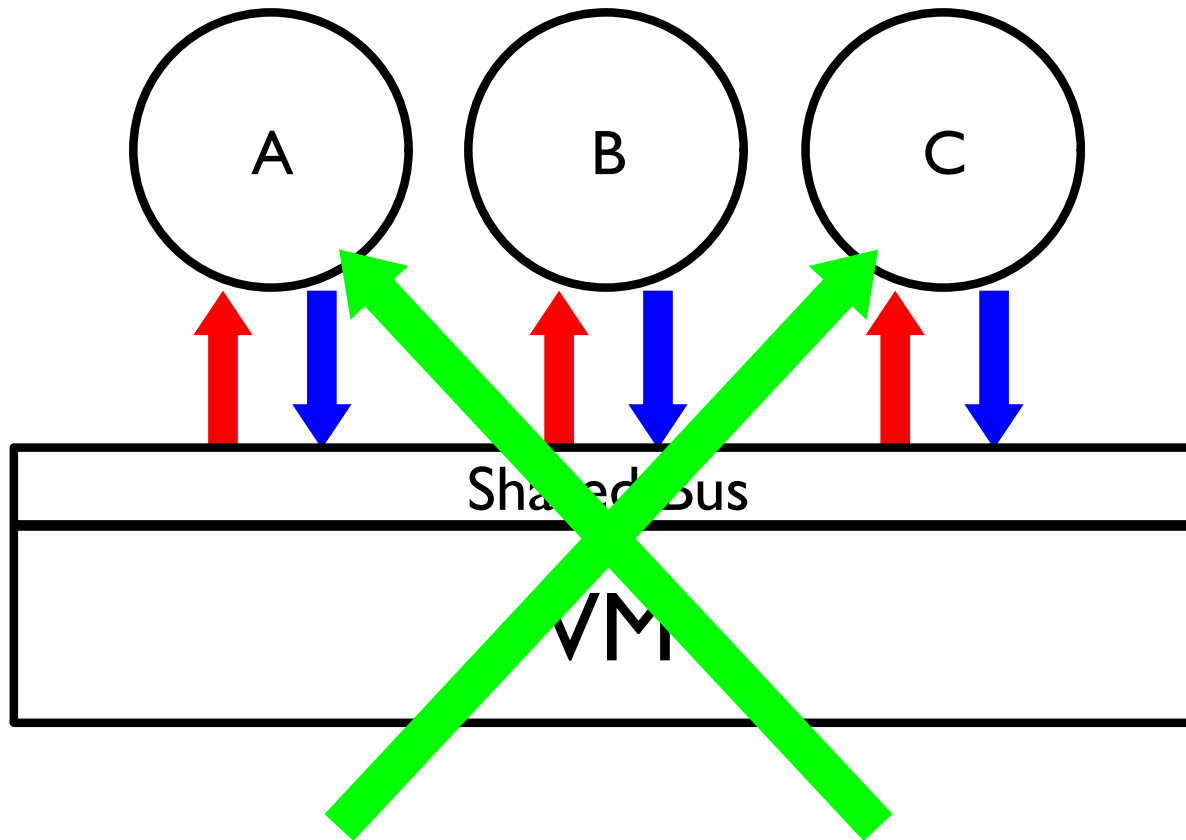
actor : Event × State → [Action] × State



```
(subscriber  
  `(price gold , ?))
```

```
(publisher  
  `(price gold , ?))
```

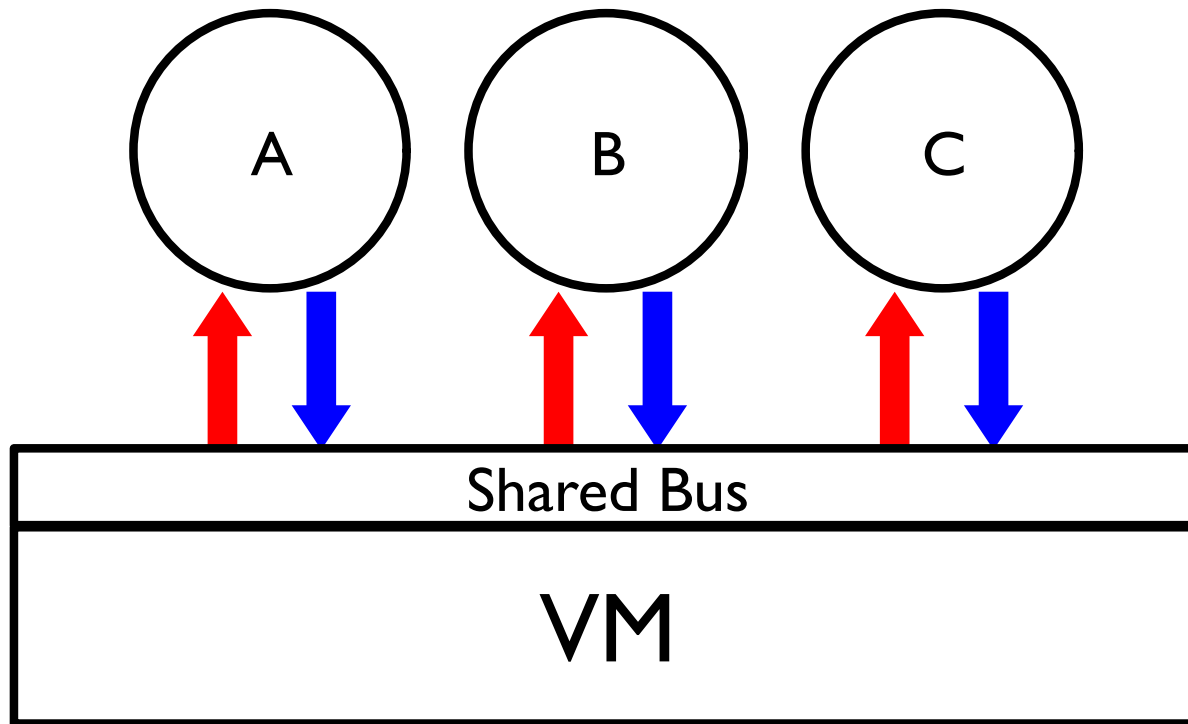
actor : Event × State → [Action] × State



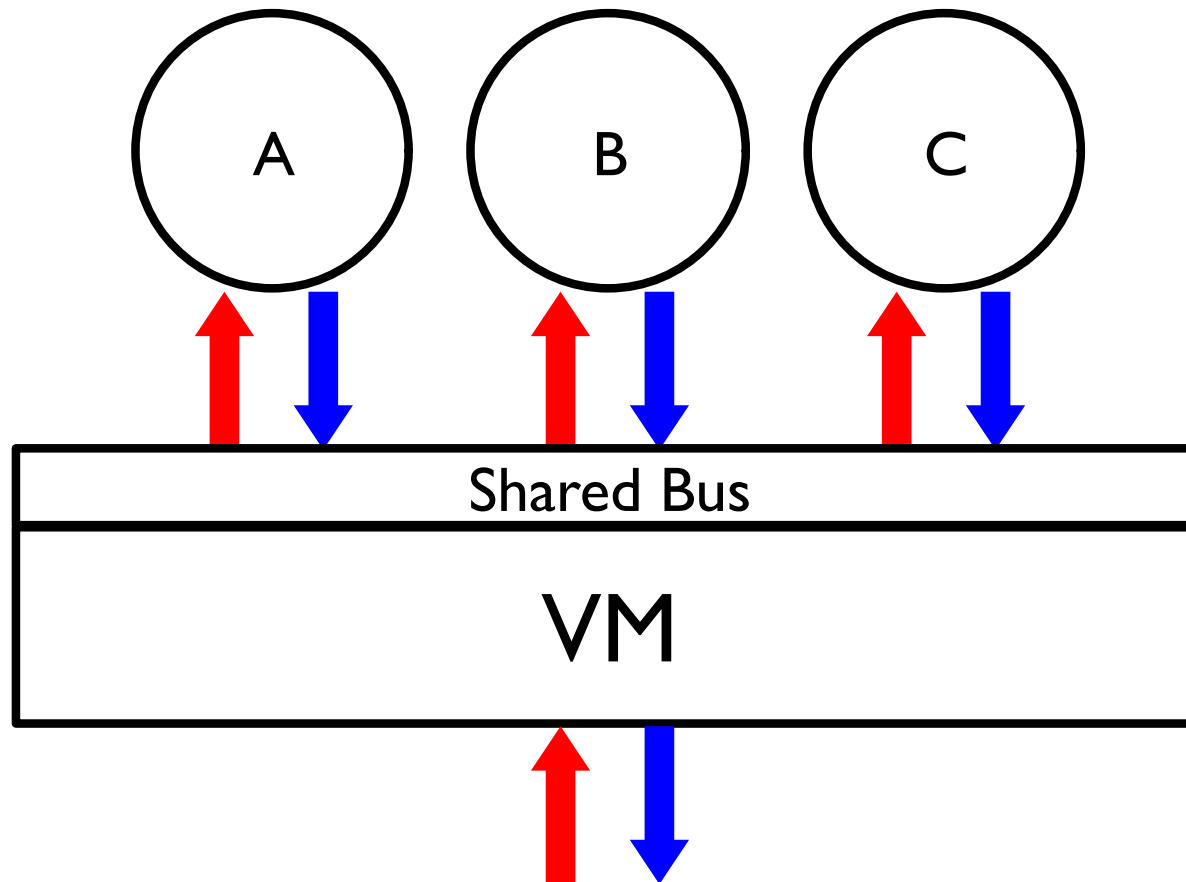
```
(subscriber  
  `(price gold , ?))
```

```
(publisher  
  `(price gold , ?))
```

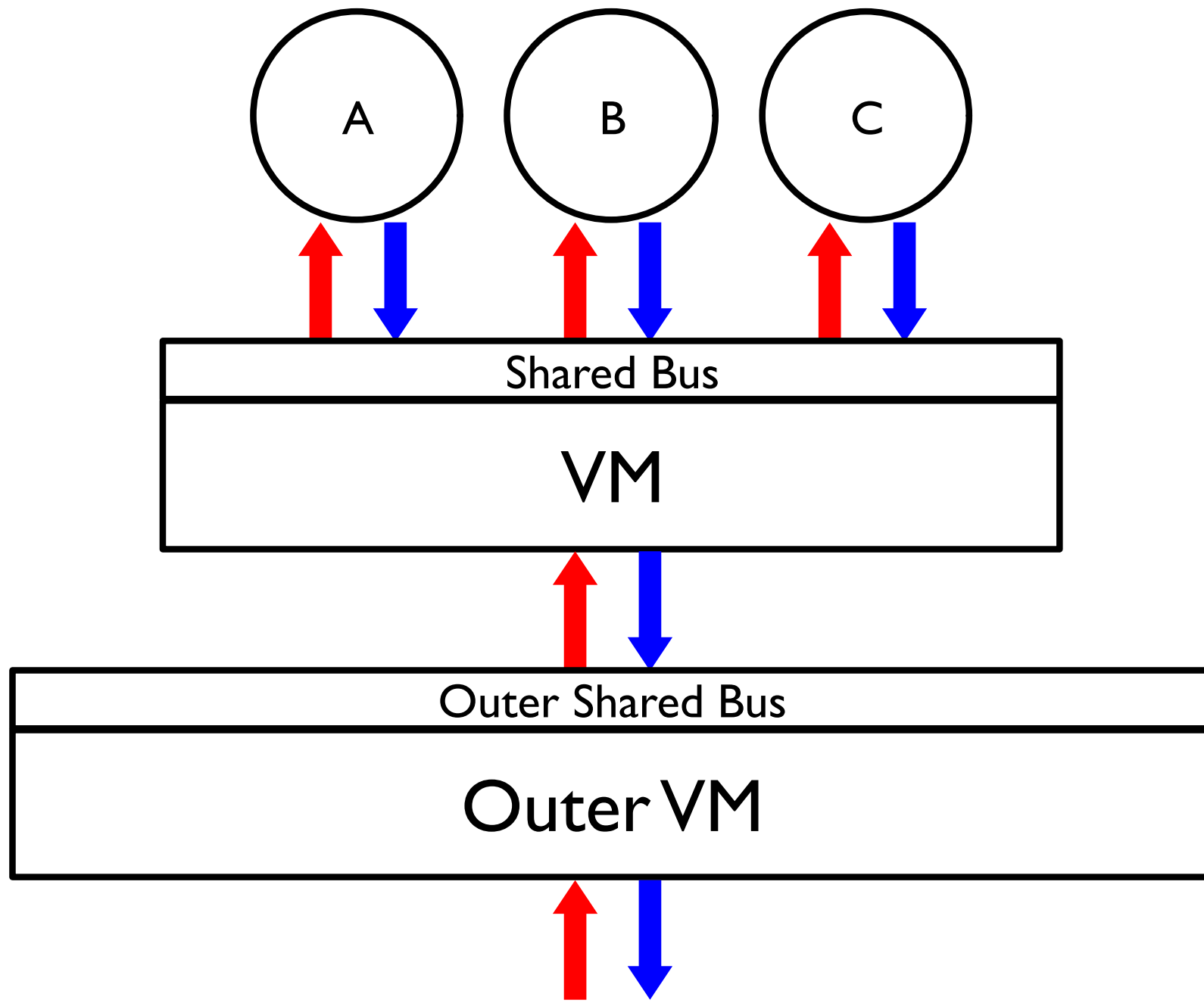
actor : Event \times State \rightarrow [Action] \times State



$vm : \text{Event} \times \text{State} \rightarrow [\text{Action}] \times \text{State}$



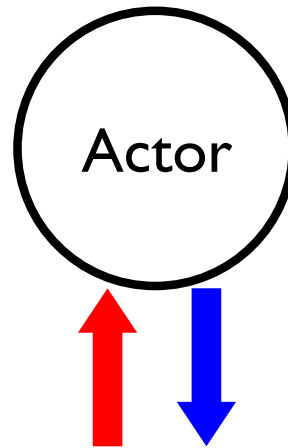
$vm : \text{Event} \times \text{State} \rightarrow [\text{Action}] \times \text{State}$



actor : Event × State → [Action] × State

Events

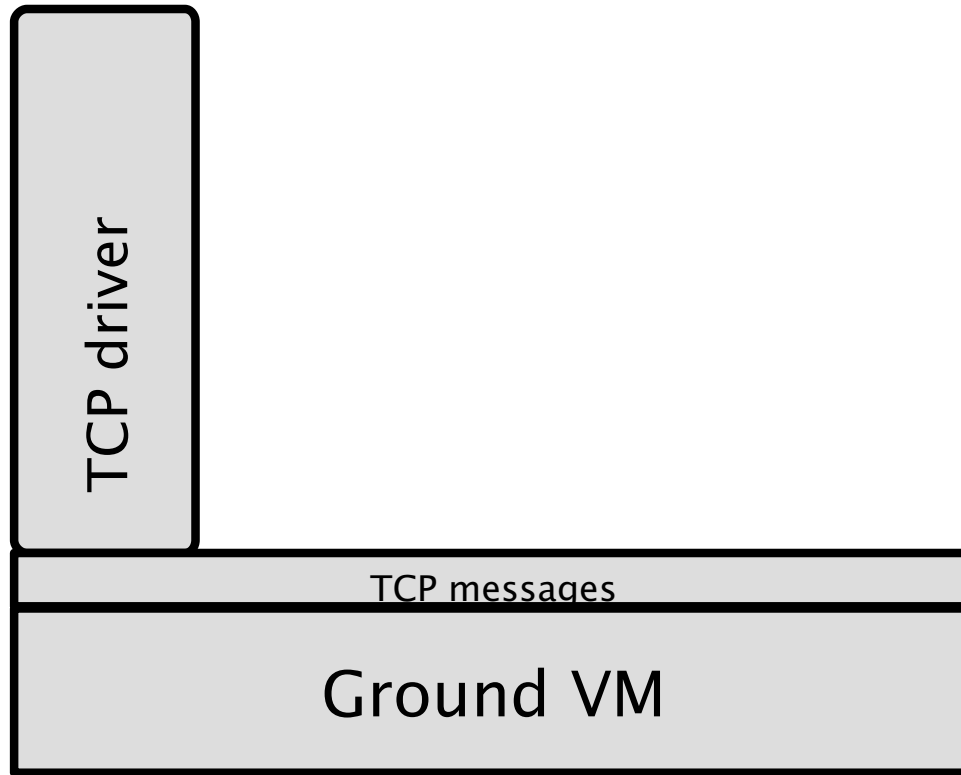
- message delivery
- presence notification
- absence notification



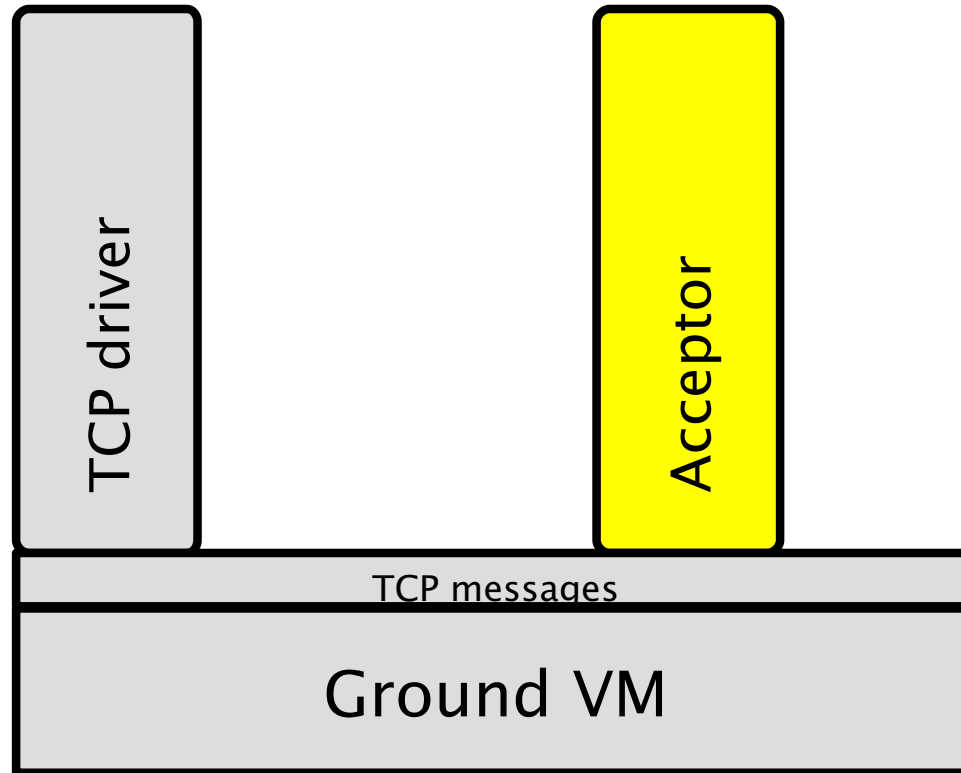
Actions

- send message
- spawn actor
- quit
- add/delete subscriber
- add/delete publisher
- observe subscribers
- observe publishers

```
$ raco pkg install marketplace
```



`#lang marketplace`

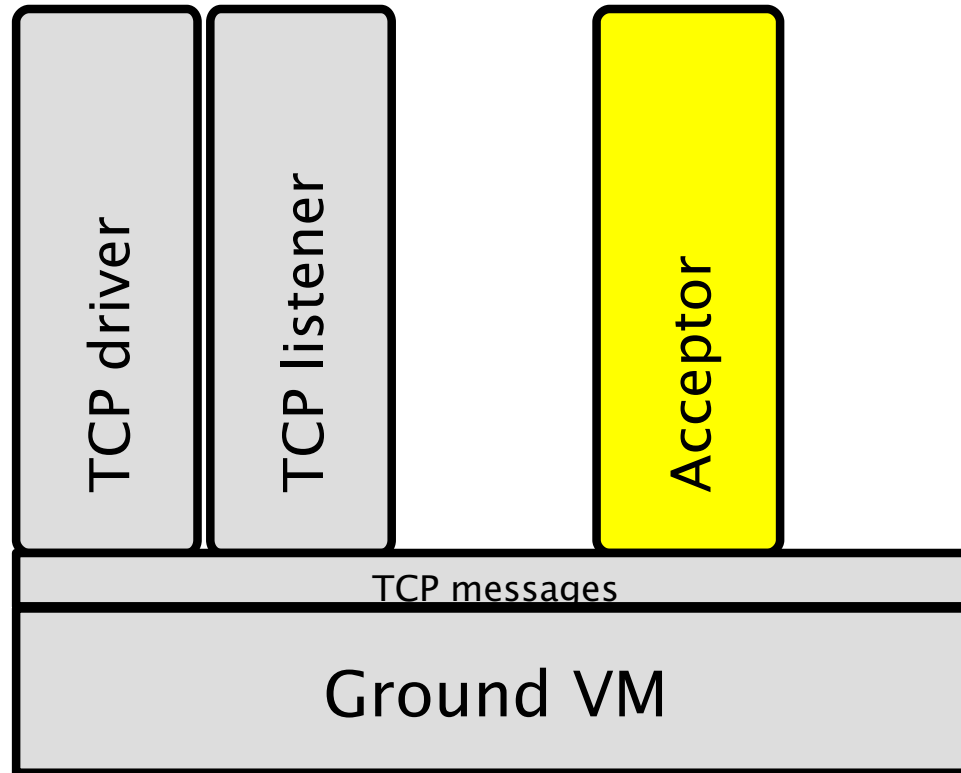


```
#lang marketplace
```

```
(observe-publishers
```

```
  (tcp-channel ? (tcp-listener 5999) ?)
```

```
  ...)
```

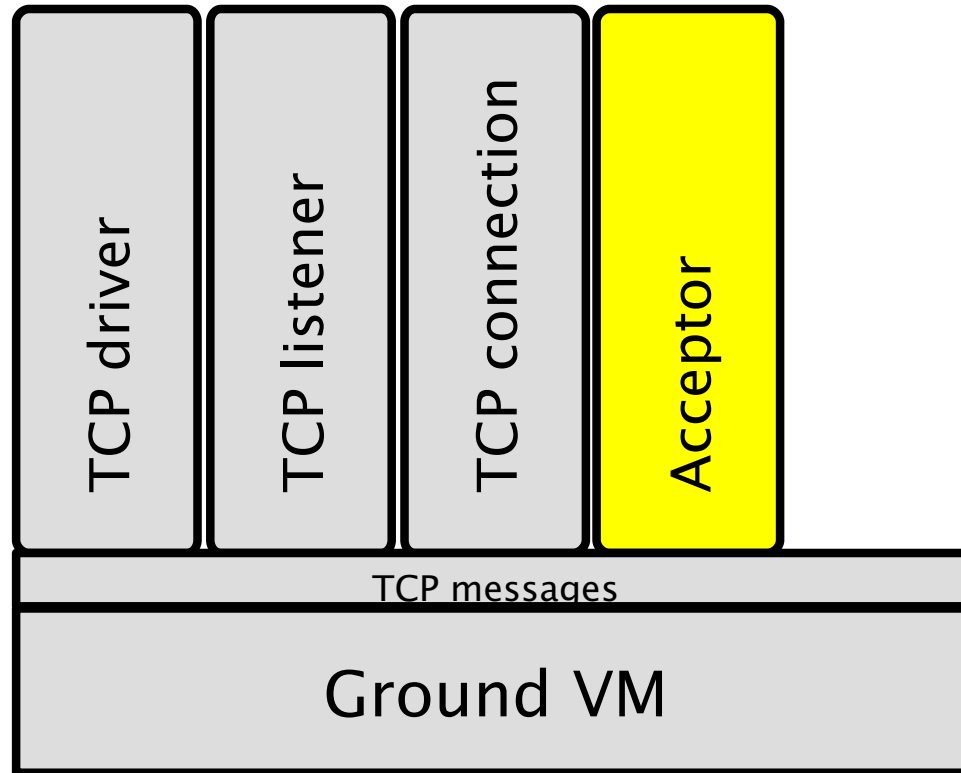


```
#lang marketplace
```

```
(observe-publishers
```

```
  (tcp-channel ? (tcp-listener 5999) ?)
```

```
  ...)
```

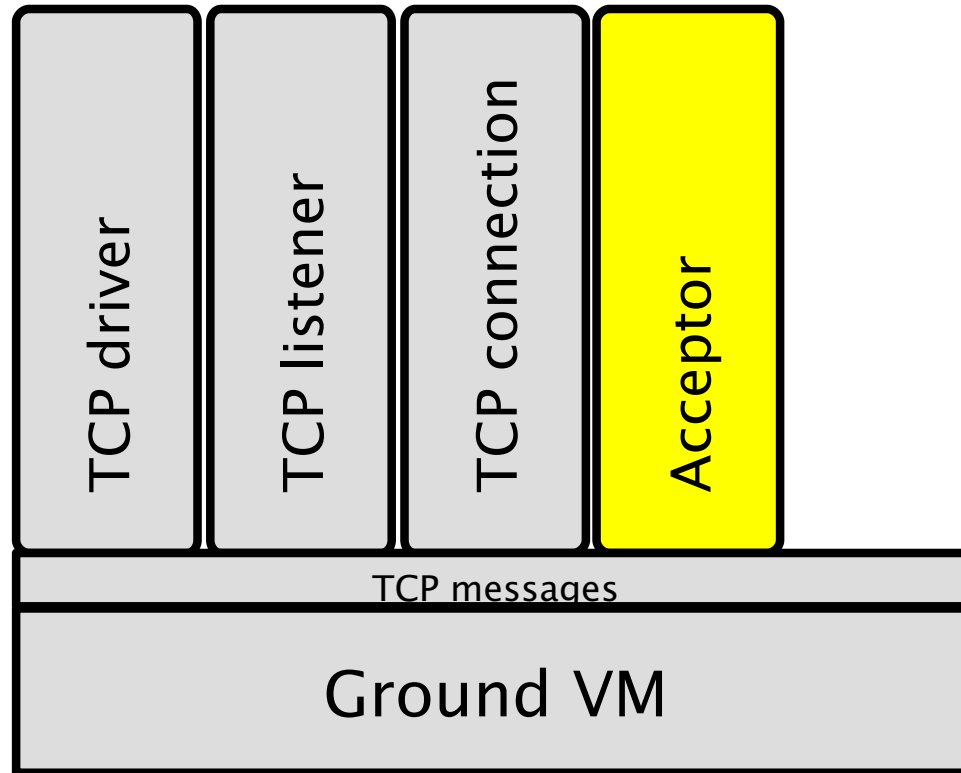


```
#lang marketplace
```

```
(observe-publishers
```

```
  (tcp-channel ? (tcp-listener 5999) ?)
```

```
  ...)
```



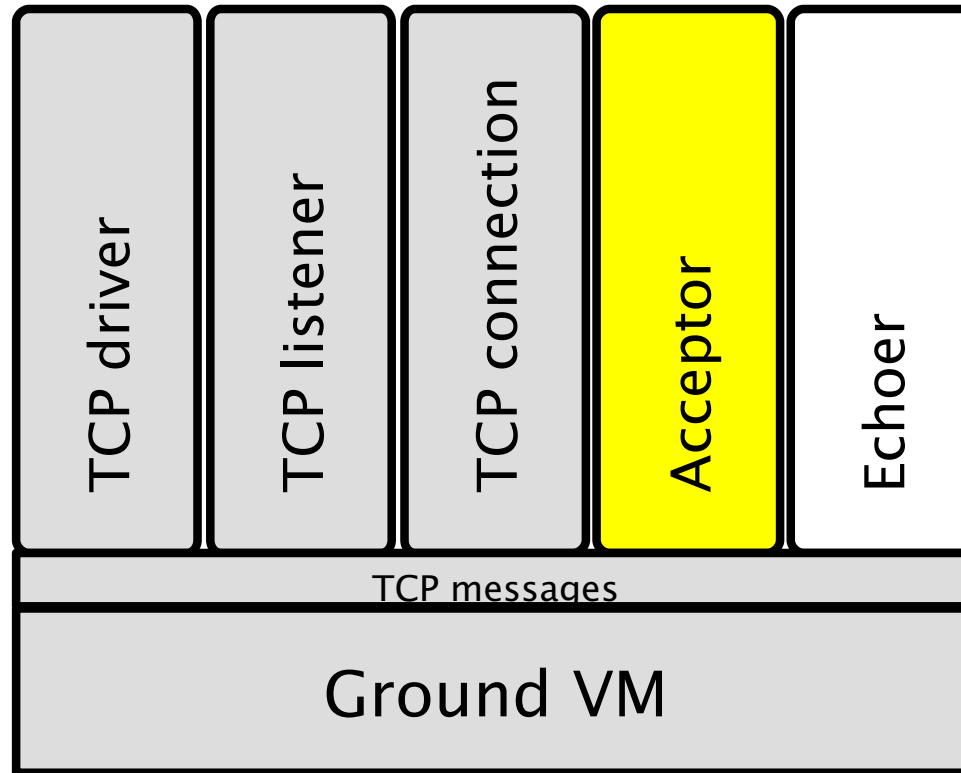
```
#lang marketplace
```

```
(observe-publishers
```

```
  (tcp-channel ? (tcp-listener 5999) ?)
```

```
  (match-conversation (tcp-channel from to _)
```

```
    (on-presence ...)))
```

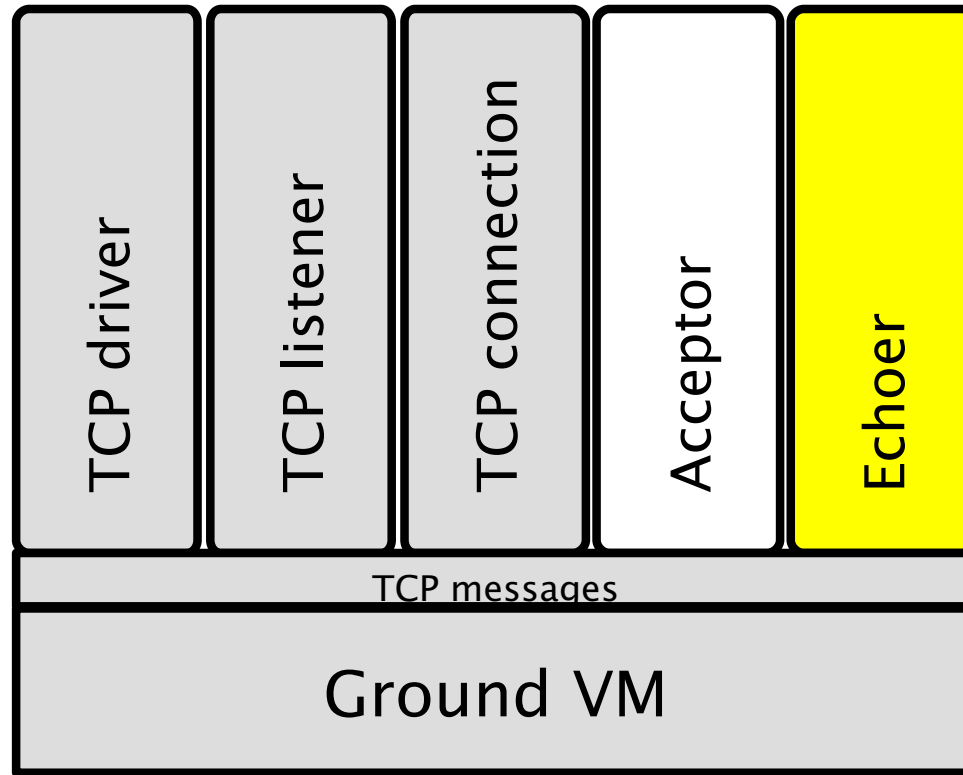
```
#lang marketplace
```

```
(observe-publishers
```

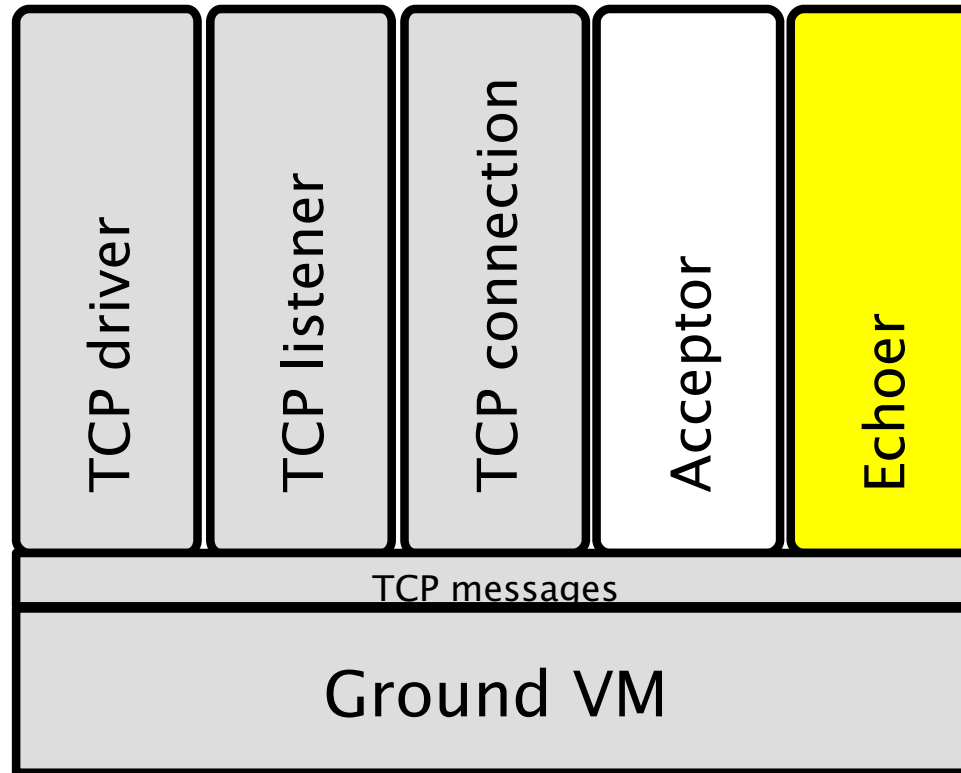
```
  (tcp-channel ? (tcp-listener 5999) ?)
```

```
  (match-conversation (tcp-channel from to _)
```

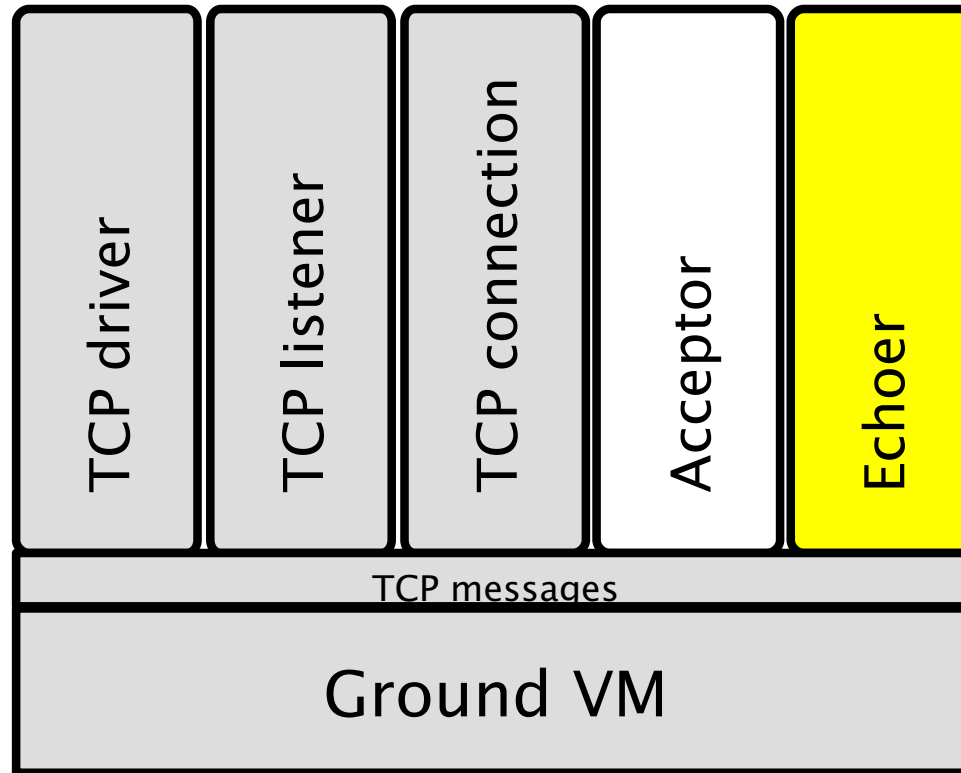
```
    (on-presence (spawn (echoer from to))))))
```



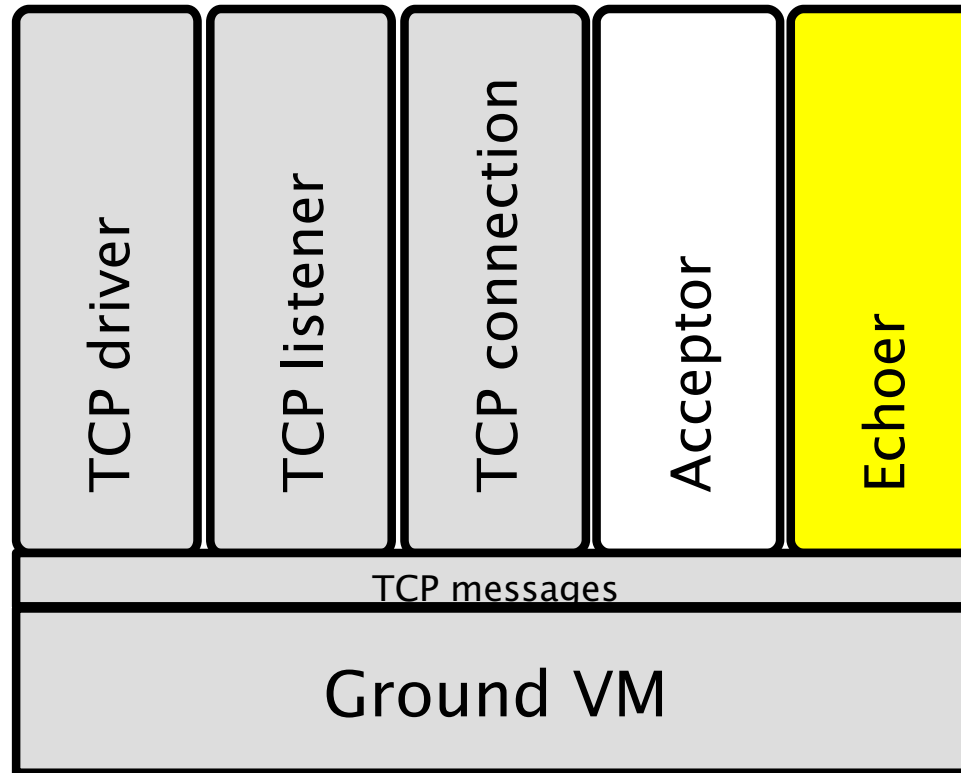
```
(define (echoer from to)  
  ...)
```



```
(define (echoer from to)
  (transition 'no-state-in-particular
    ...))
```



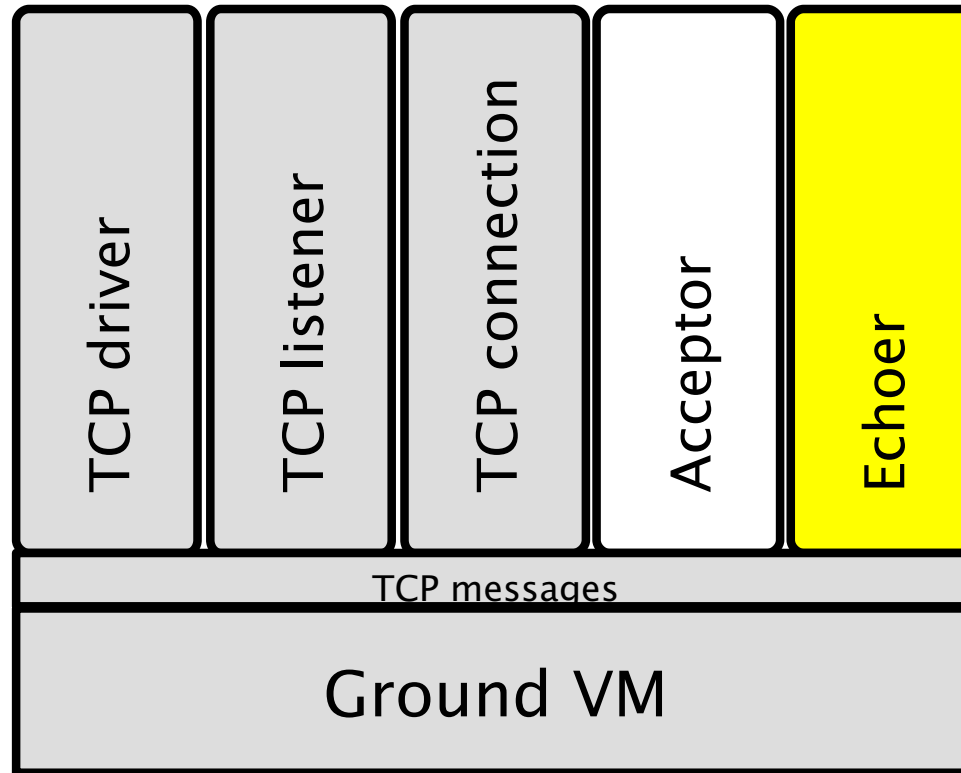
```
(define (echoer from to)
  (transition 'no-state-in-particular
    (publisher (tcp-channel to from ?))
    (subscriber (tcp-channel from to ?)
      ...)))
```



```

(define (echoer from to)
  (transition 'no-state-in-particular
    (publisher (tcp-channel to from ?))
    (subscriber (tcp-channel from to ?)
      (on-message
        [(tcp-channel _ _ data)
         (send-message (tcp-channel to from data))]
        ...))))

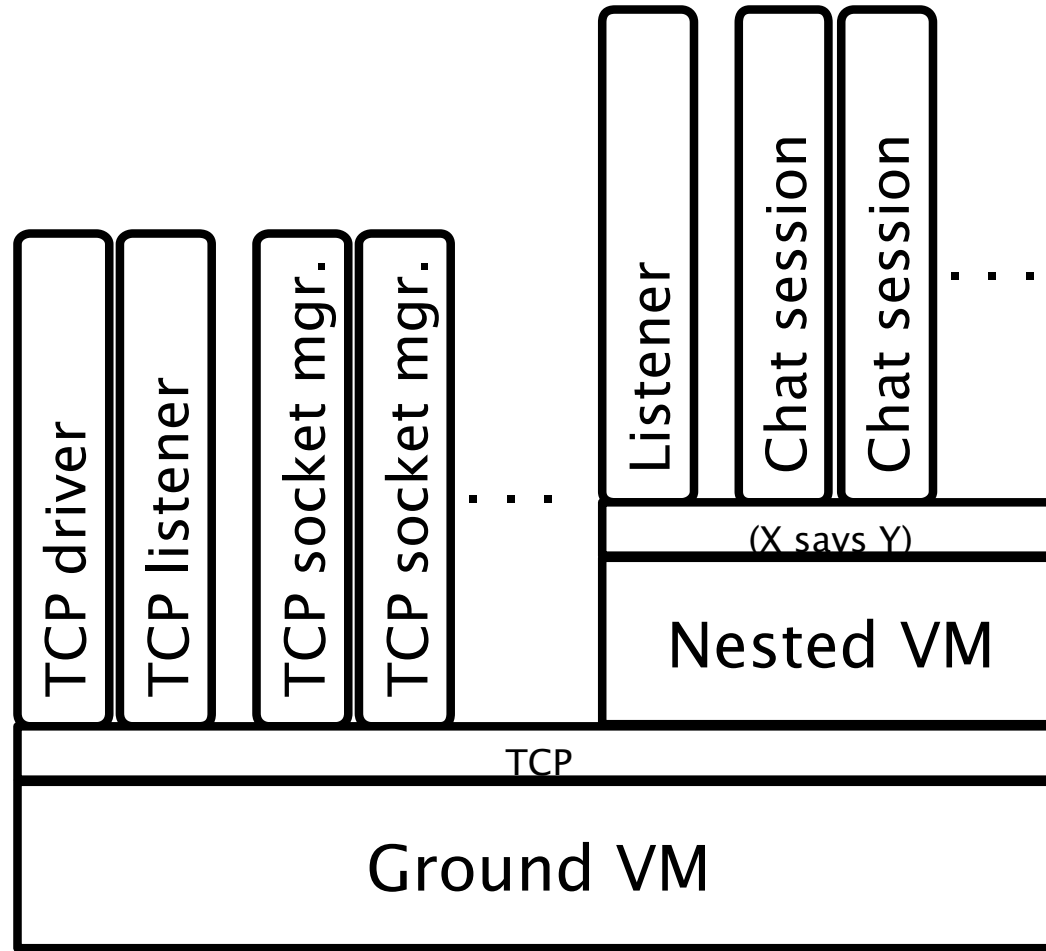
```



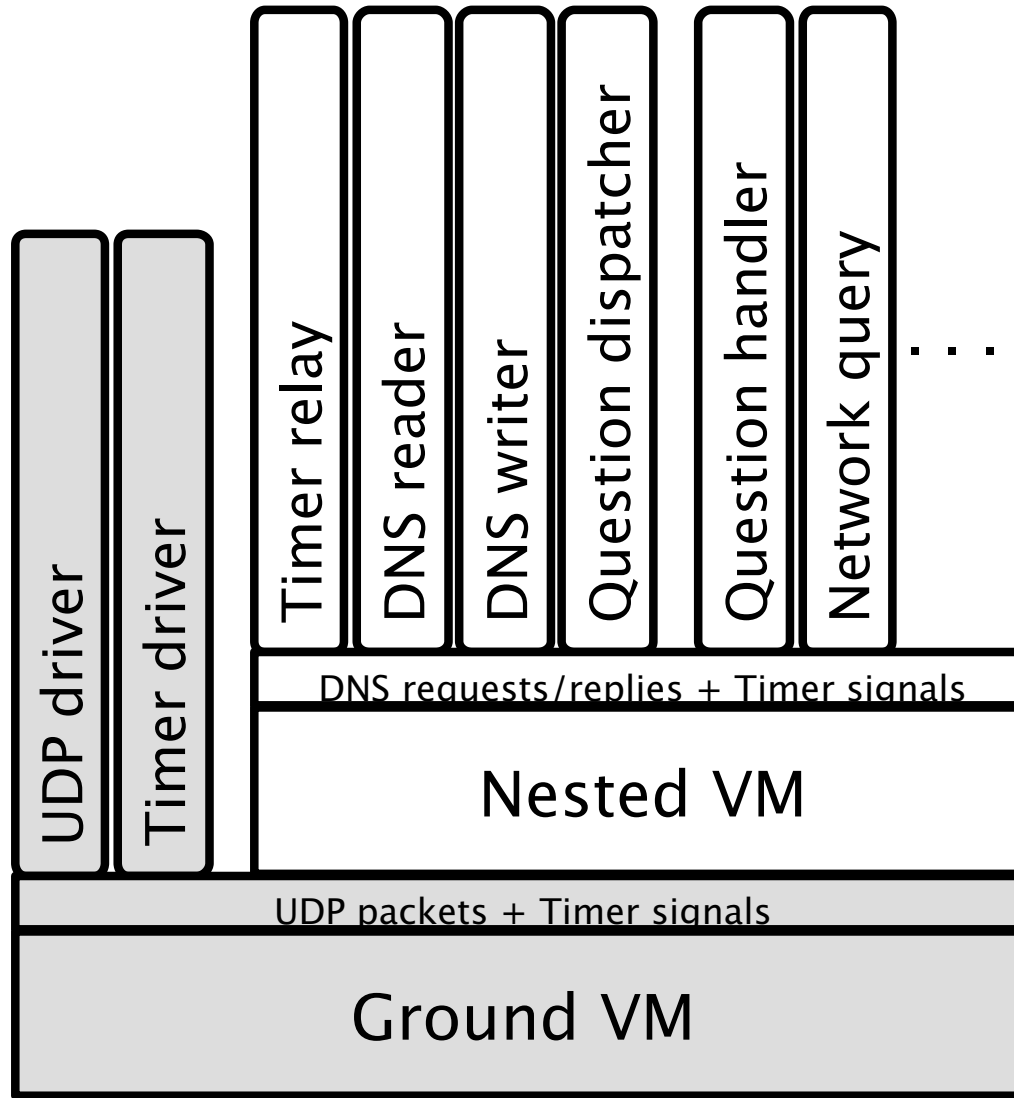
```

(define (echoer from to)
  (transition 'no-state-in-particular
    (publisher (tcp-channel to from ?))
    (subscriber (tcp-channel from to ?)
      (on-message
        [(tcp-channel _ _ data)
         (send-message (tcp-channel to from data))]))
    (on-absence (quit))))

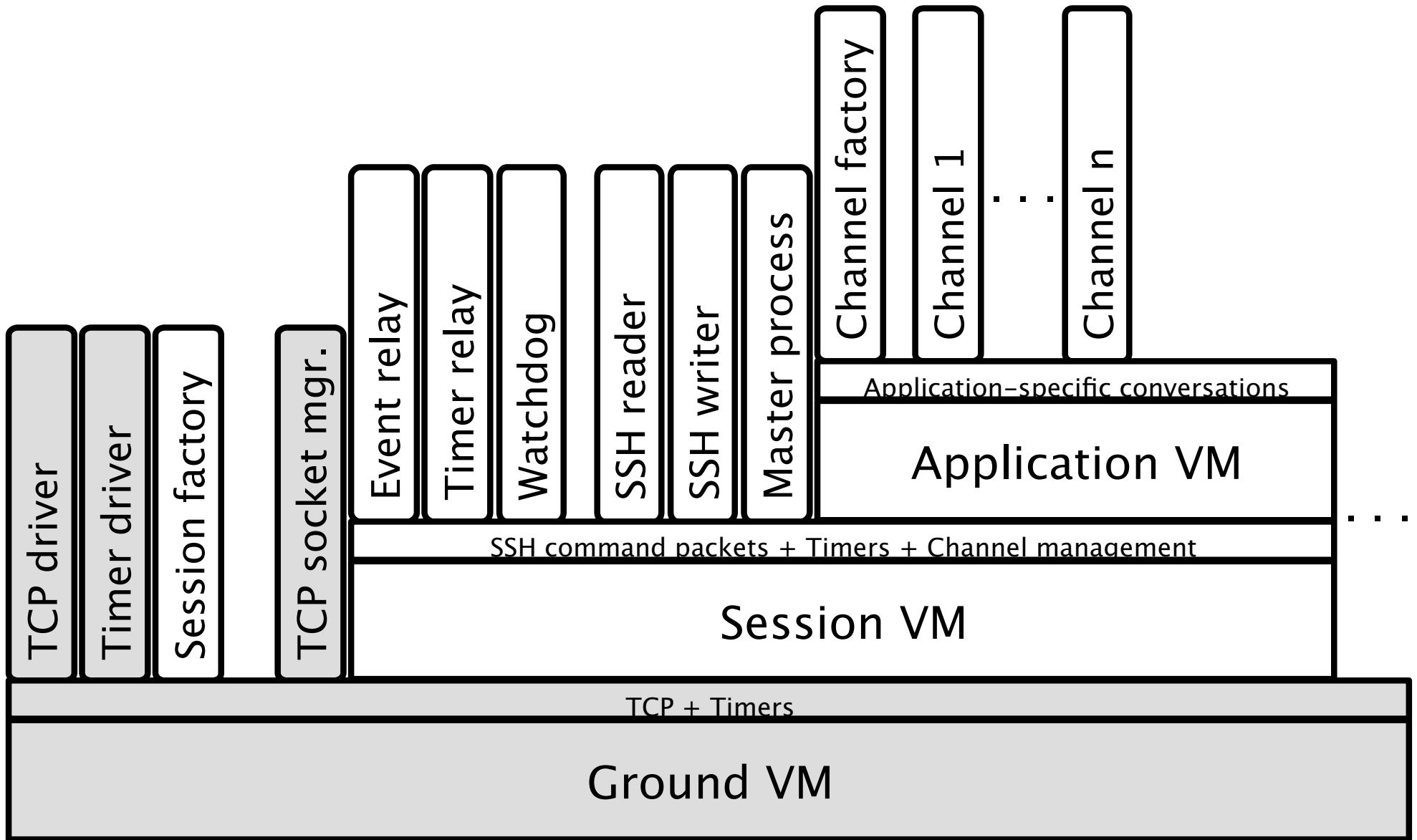
```



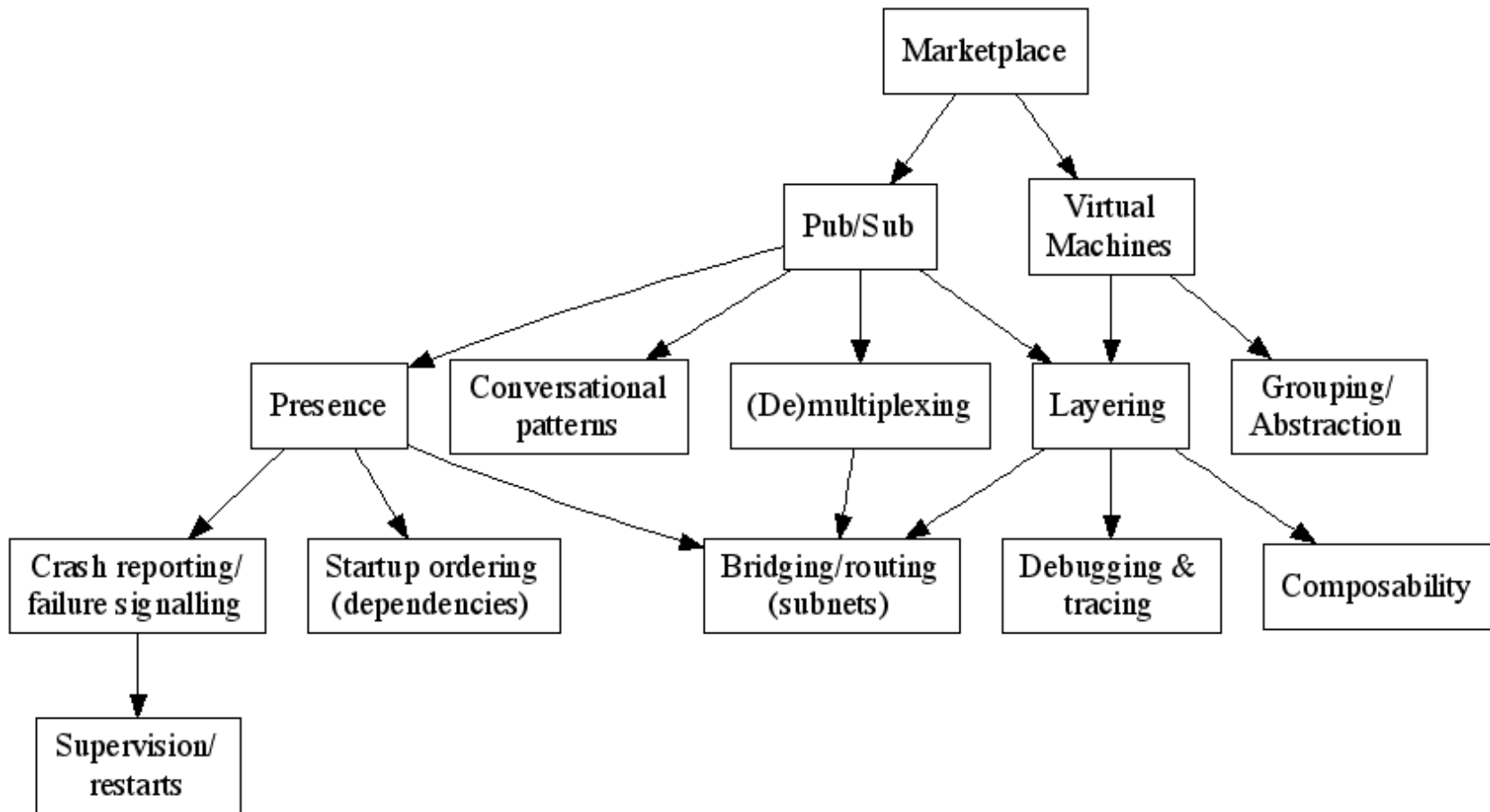
github.com/tonyg/marketplace



github.com/tonyg/marketplace-dns



github.com/tonyg/marketplace-ssh



www.ccs.neu.edu/home/tonyg/marketplace

github.com/tonyg/marketplace {-dns, -ssh}