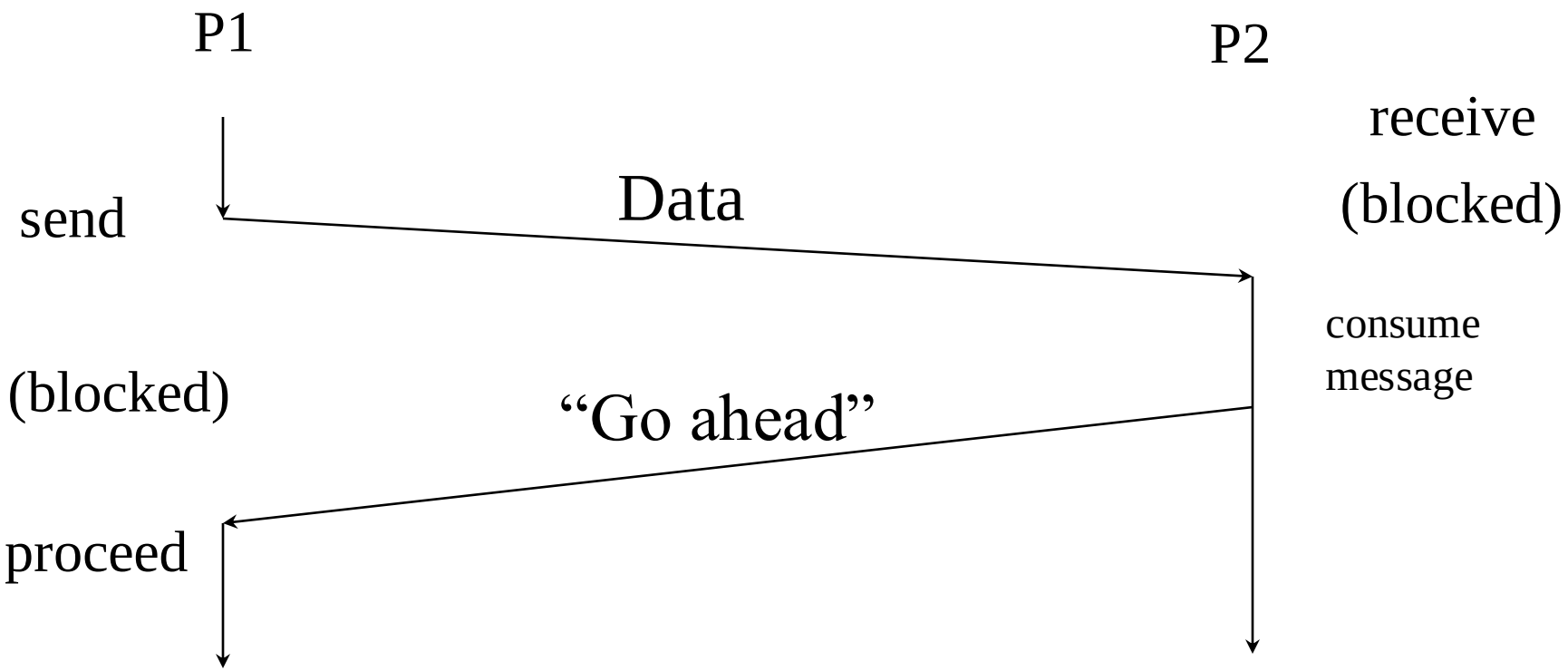
Semantics of Send and Receive

- Can be blocking ("synchronous") or nonblocking ("asynchronous")
  - remember:
    - procedure call is synchronous
    - thread fork is asynchronous
  - Send, Receive both have synchronous and asynchronous implementations
    - Channels were defined as asynchronous send, synchronous receive

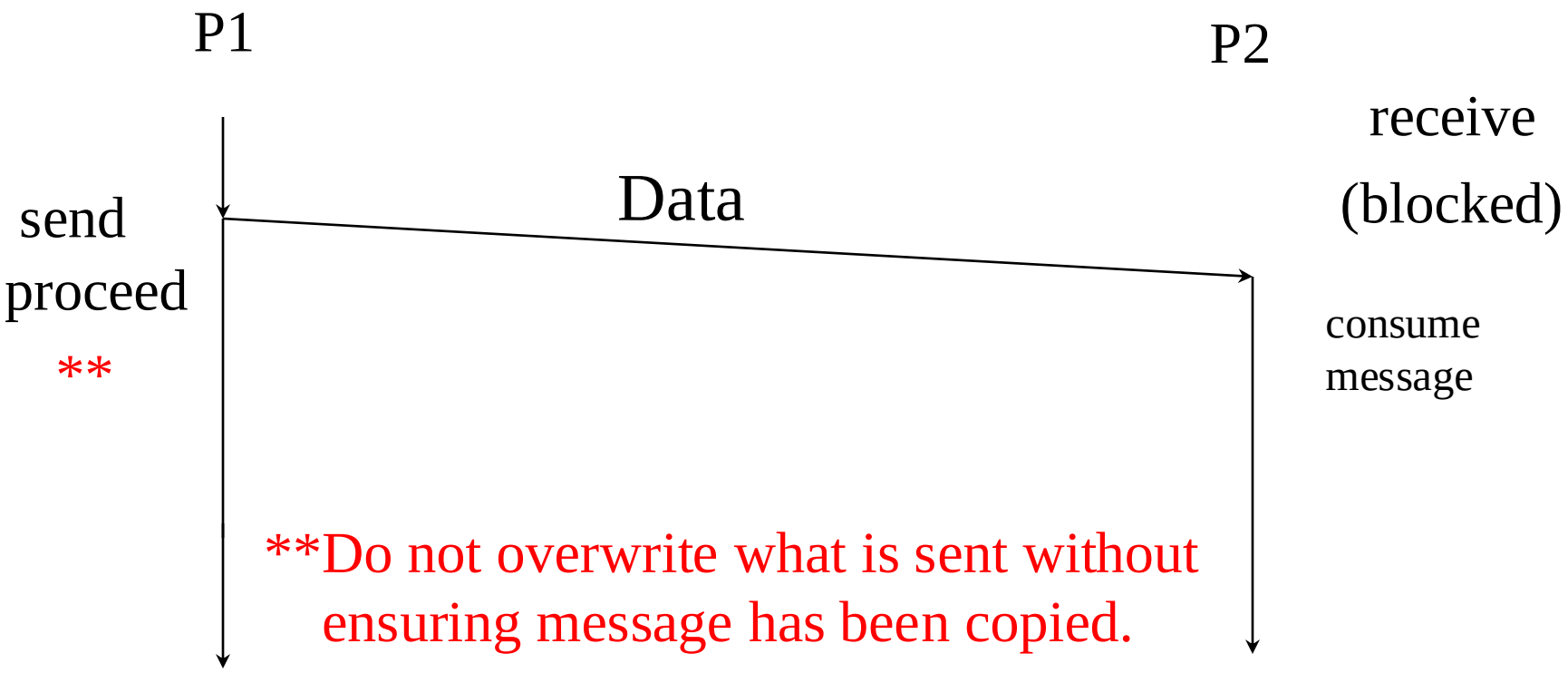
Here, we will consider all four semantic combinations of send/receive

Picture of Synchronous Send and Synchronous Receive



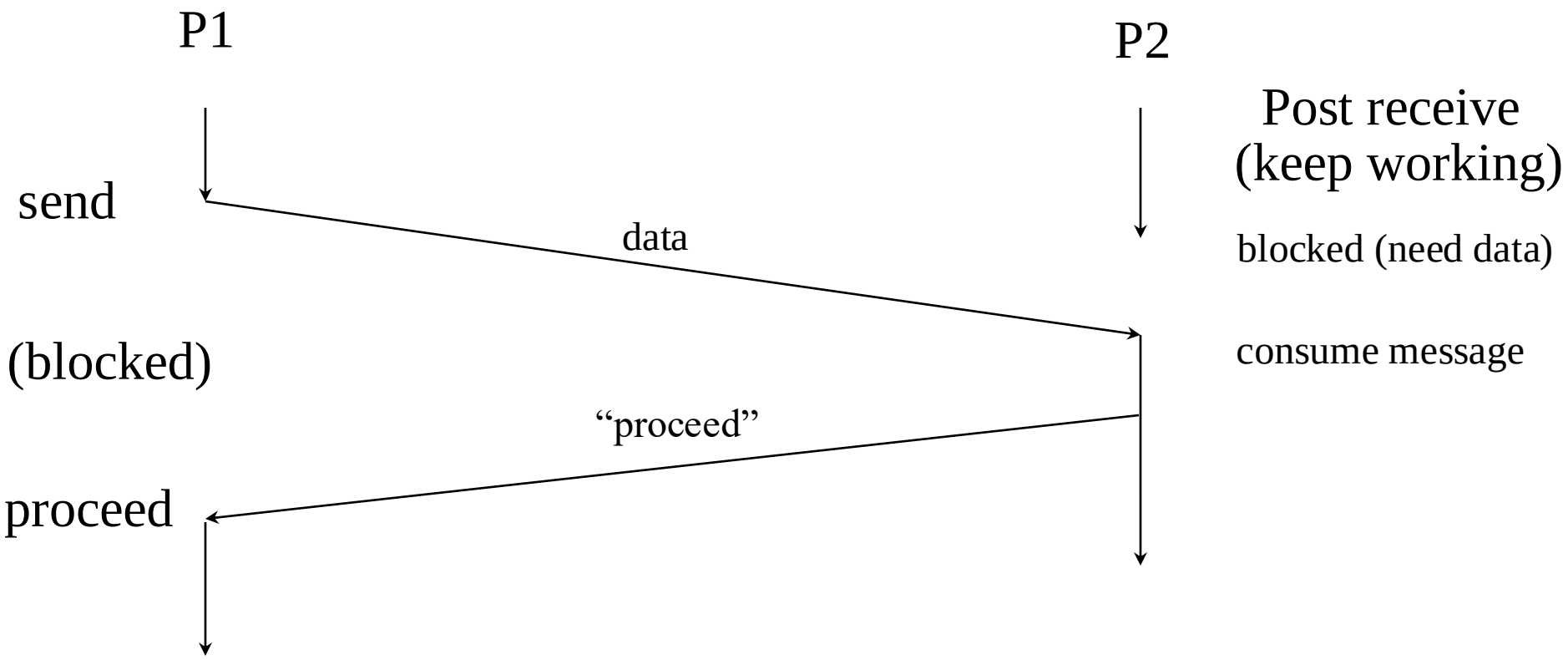
Send blocked if invoked before the receive.

Picture of Asynchronous Send, Synchronous Receive



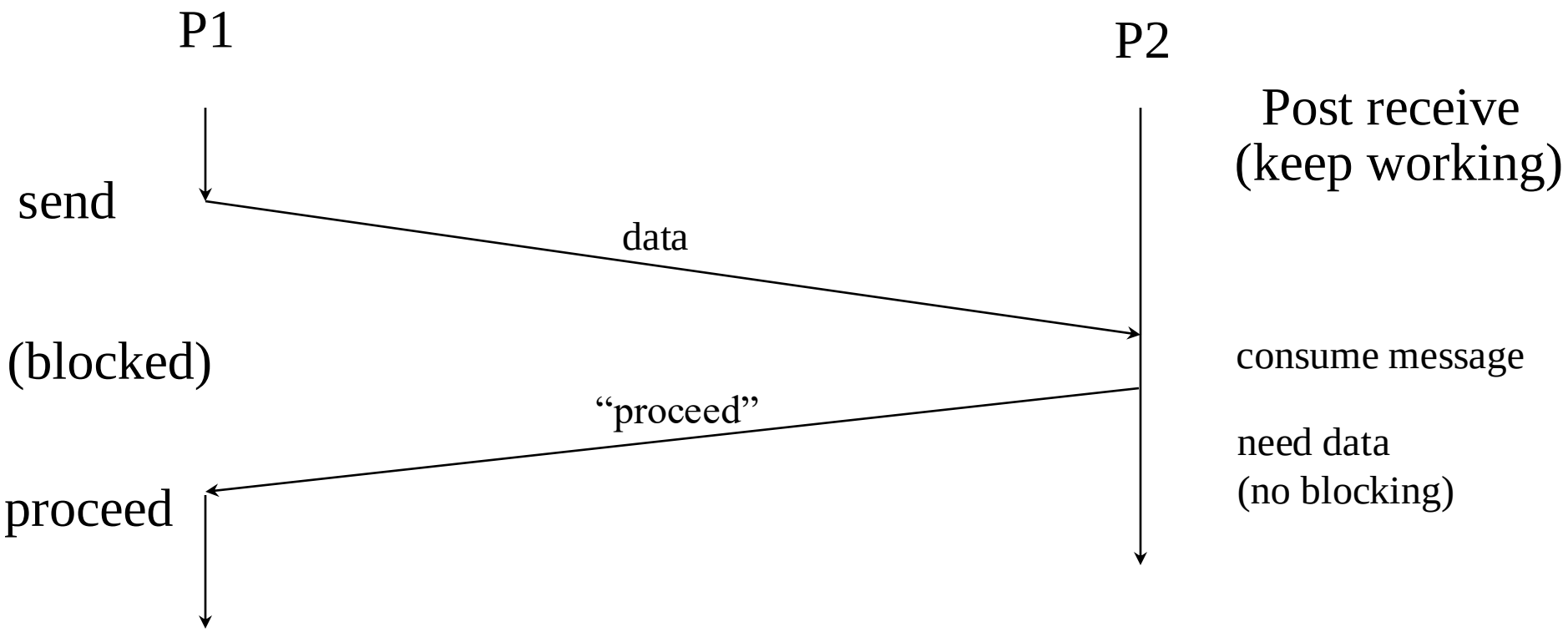
Question: What happens if message arrives before the receive?

Picture of Synchronous Send, Asynchronous Receive



But, message may arrive before the block on the receiver (next slide)

Picture of Synchronous Send, Asynchronous Receive



What does (Asynchronous Send, Asynchronous Receive) look like?

Implementation of Asynchronous Send, Blocking Receive

- Implementation must keep track of all channels
  - one buffer and one semaphore per channel at receiver
- On Send(channel, userSpecifiedData)
  - copy userSpecifiedData into sender-side buffer
  - (buffer eventually put onto network)
- On Receive(channel, userSpecifiedData)
  - P(thisQueue); copy buffer into userSpecifiedData
- On incoming message (specifies channel)
  - copy message into receiver-side buffer; V(thisQueue)

Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)

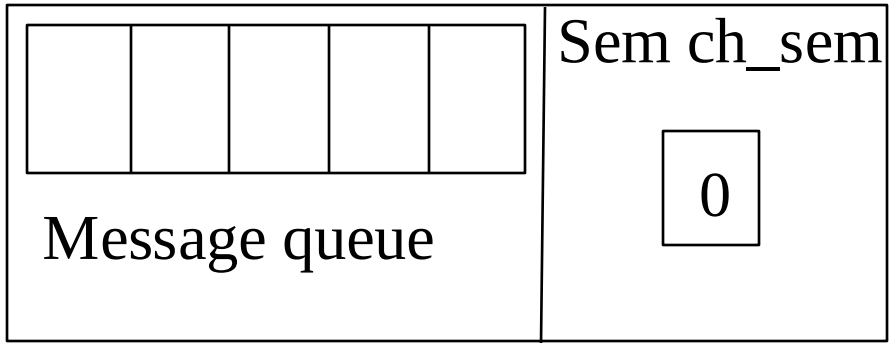
Sender (IP a.b.c.d)

Receiver (IP e.f.g.h)

Send(ch, x)

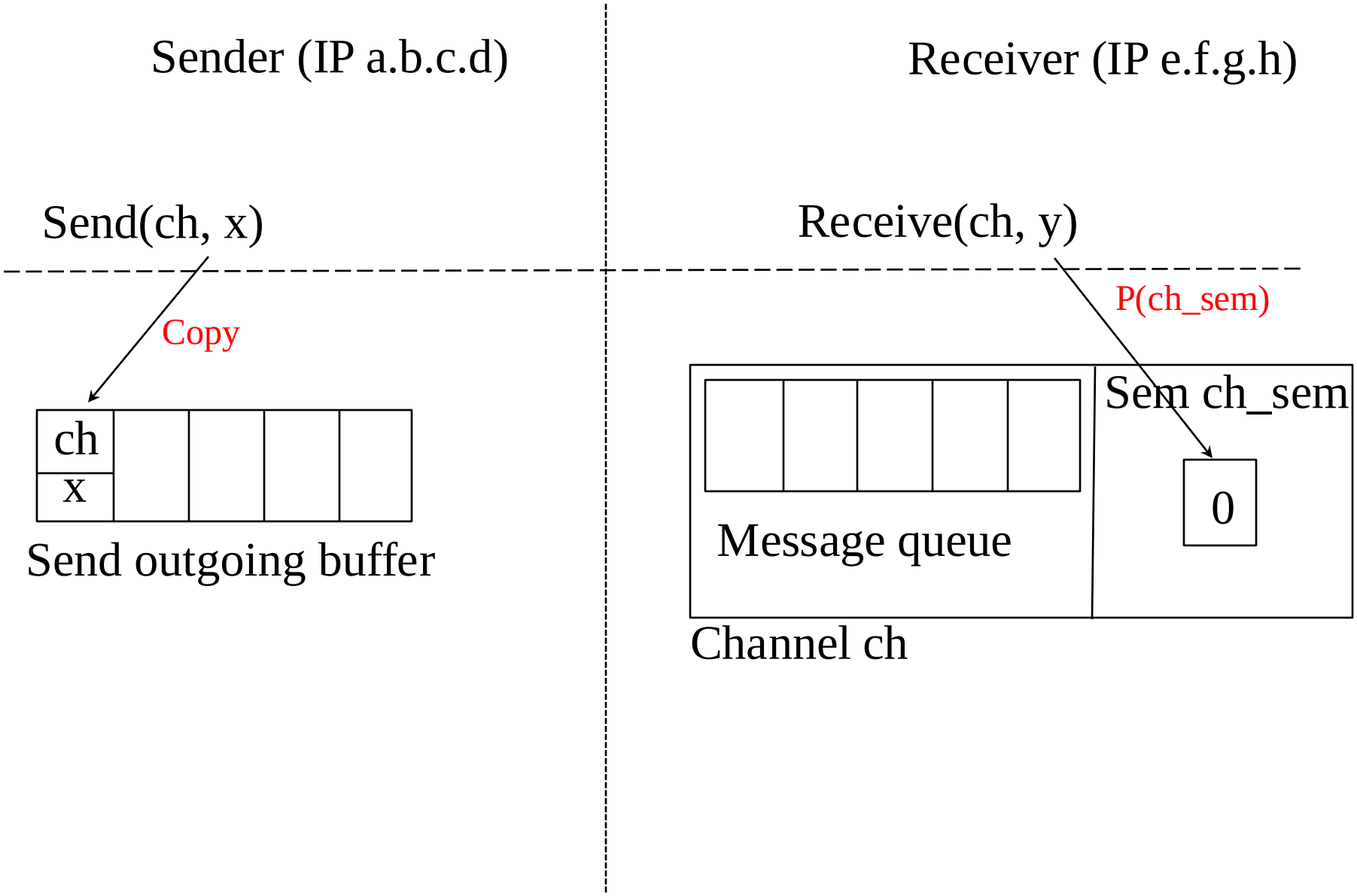
Receive(ch, y)

Send outgoing buffer

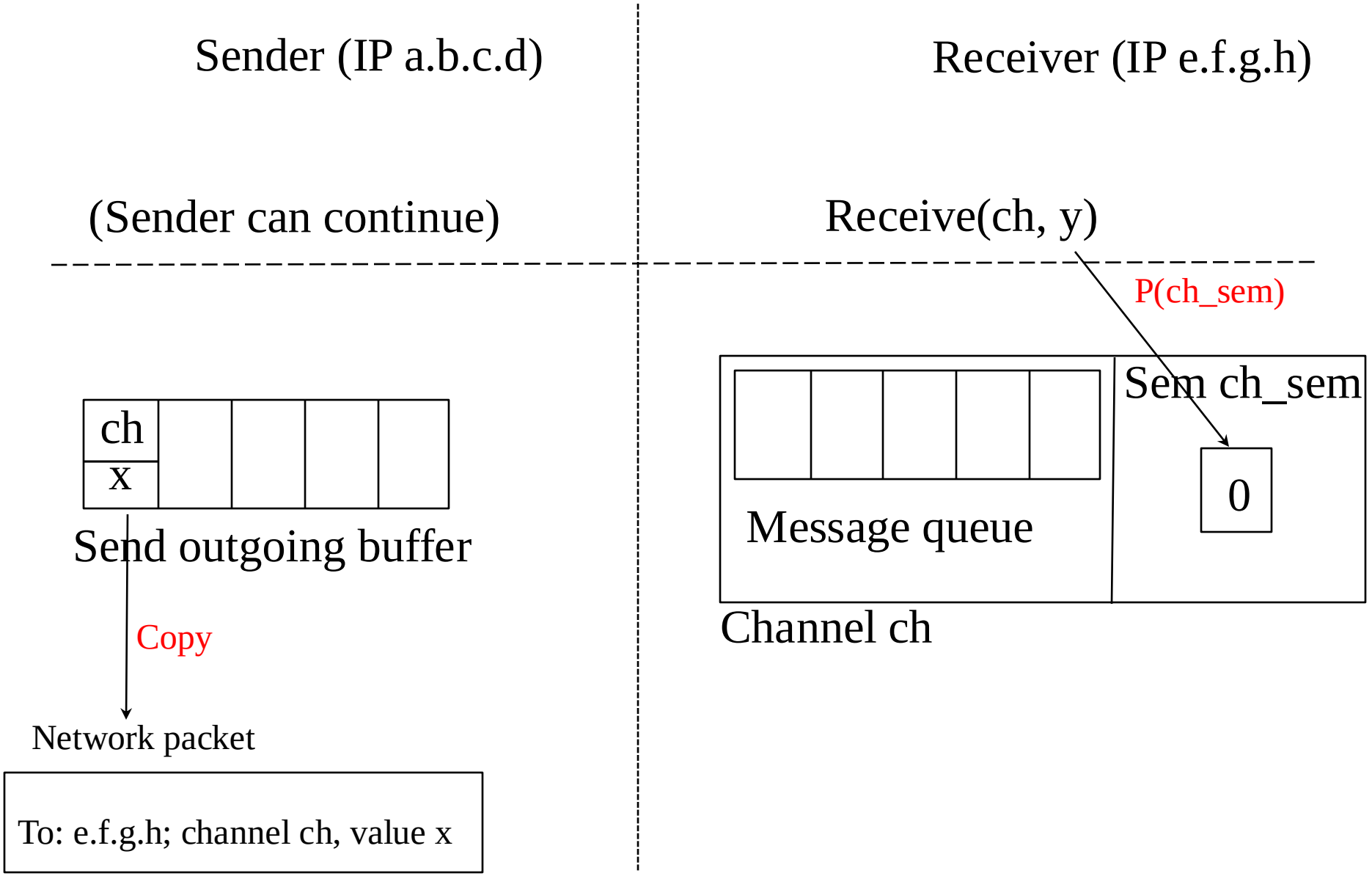


Channel ch

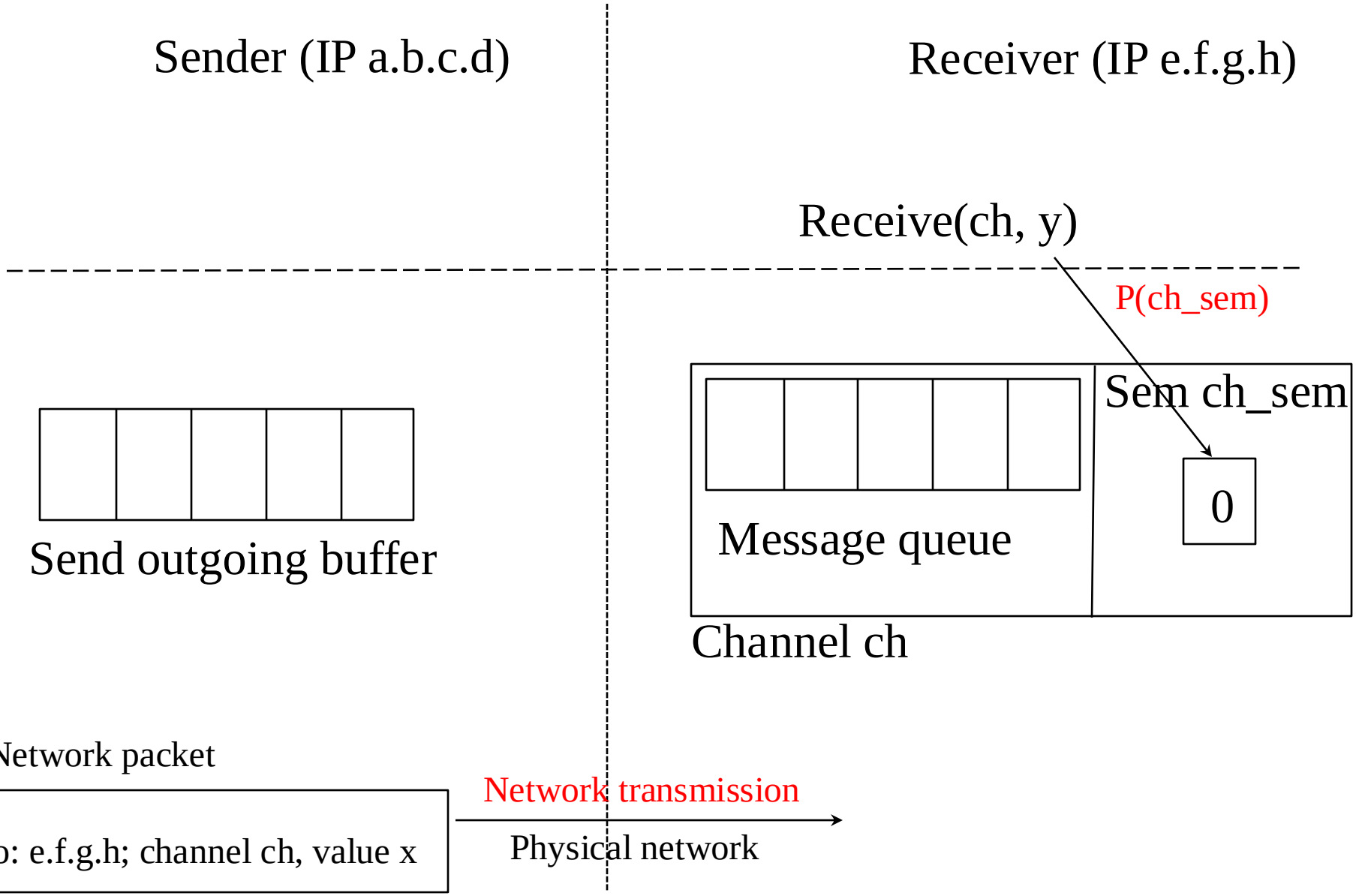
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



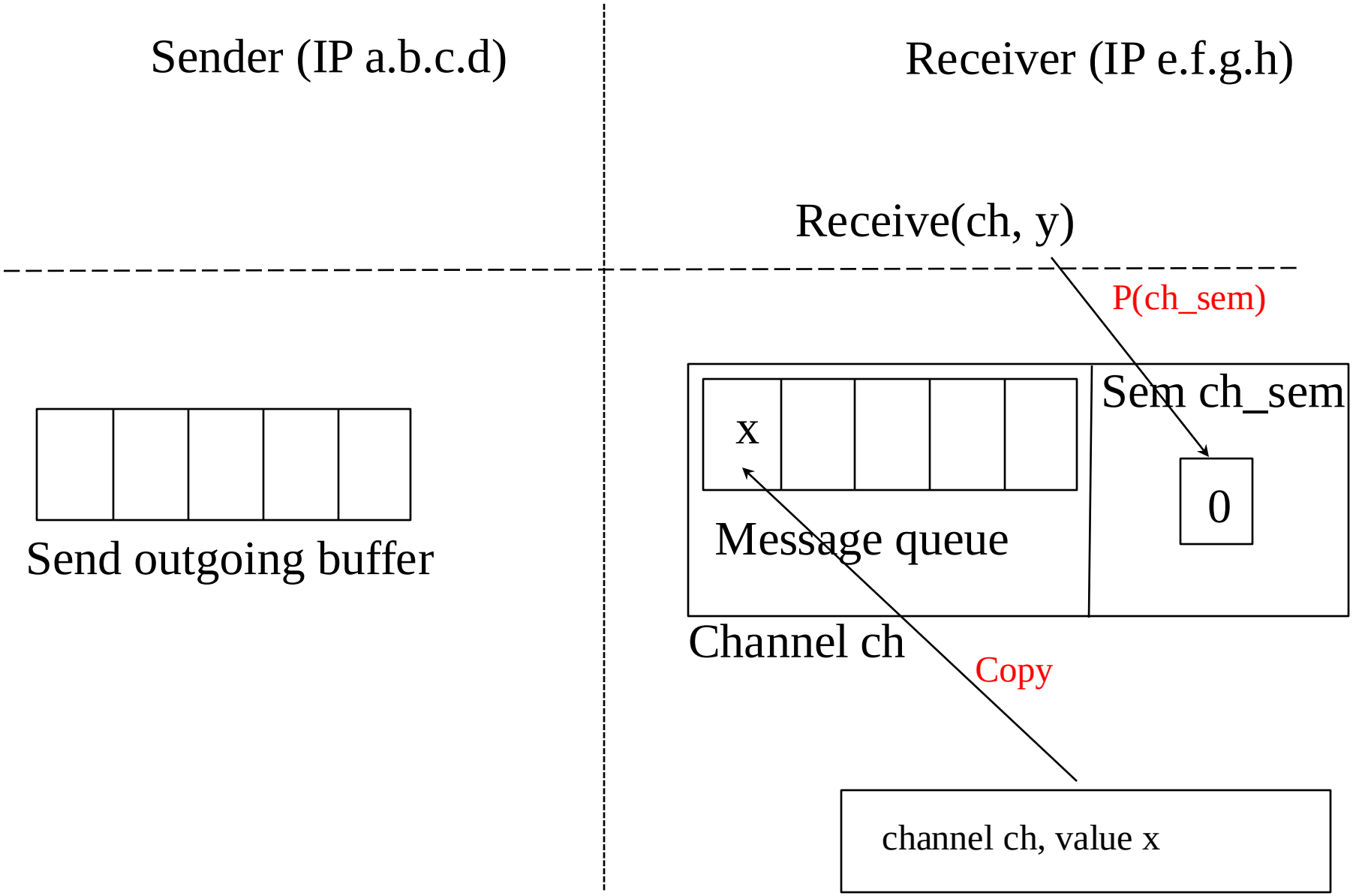
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



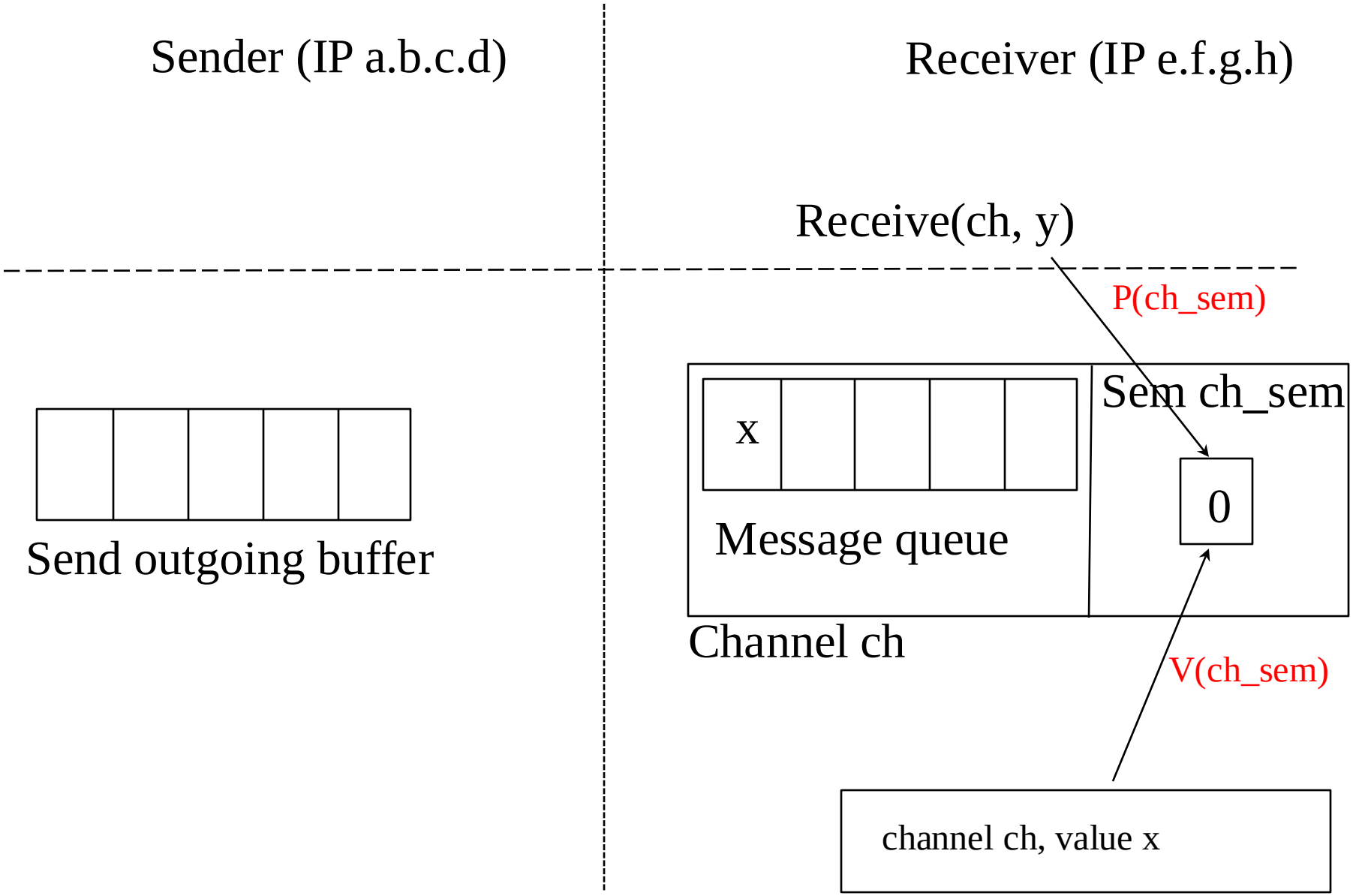
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



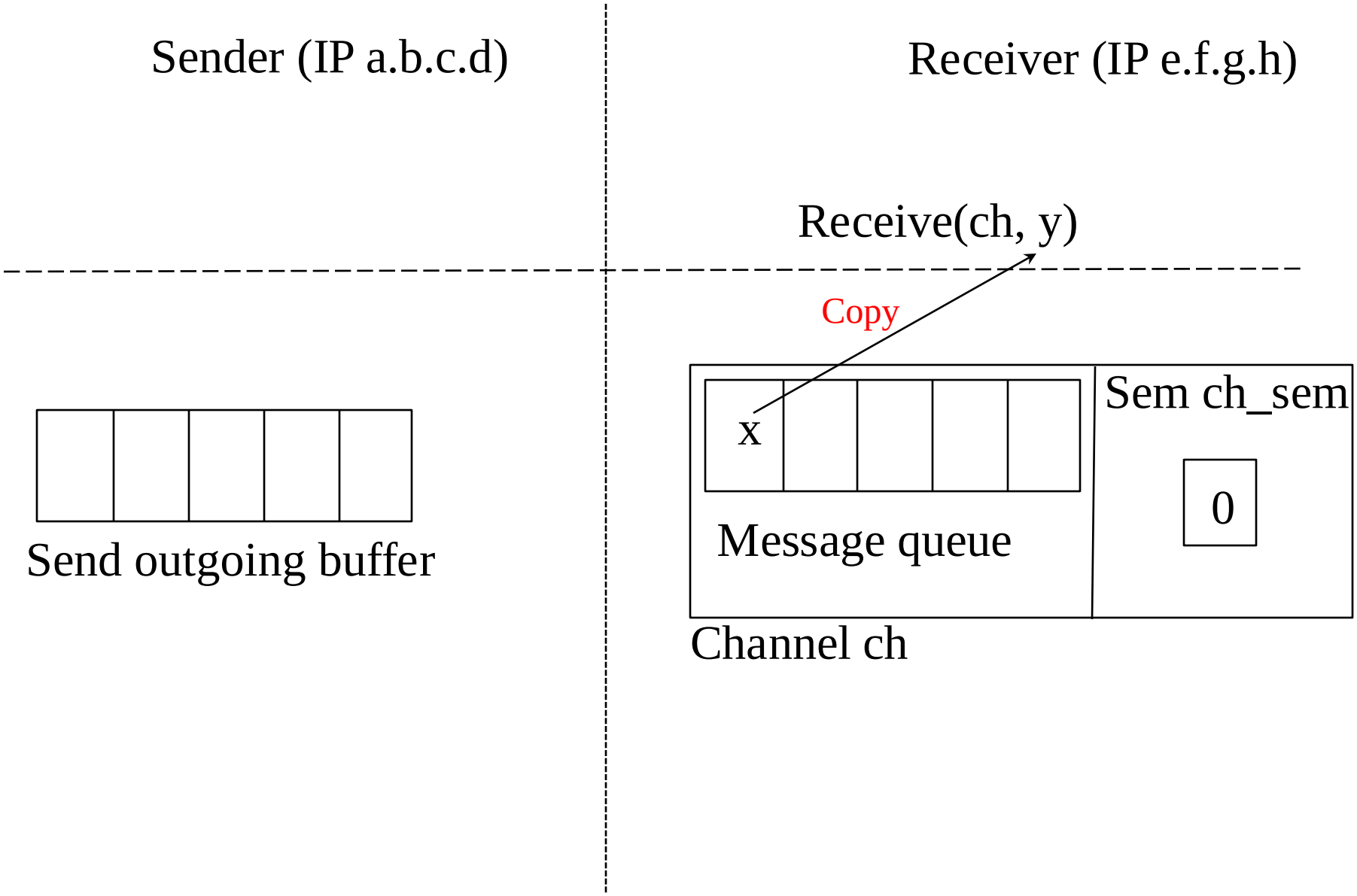
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)



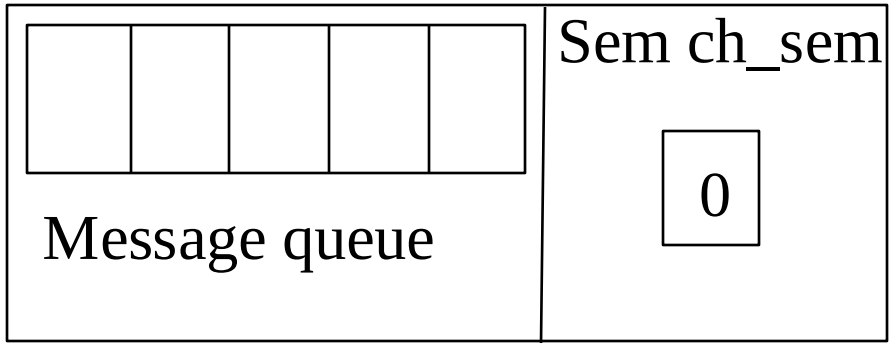
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)

Sender (IP a.b.c.d)

Receiver (IP e.f.g.h)

Receive(ch, y) y is now == x

Send outgoing buffer



Channel ch

Picture of Message Passing Implementation
(Asynchronous Send, Synchronous Receive)
Data at Receiver Before Receive

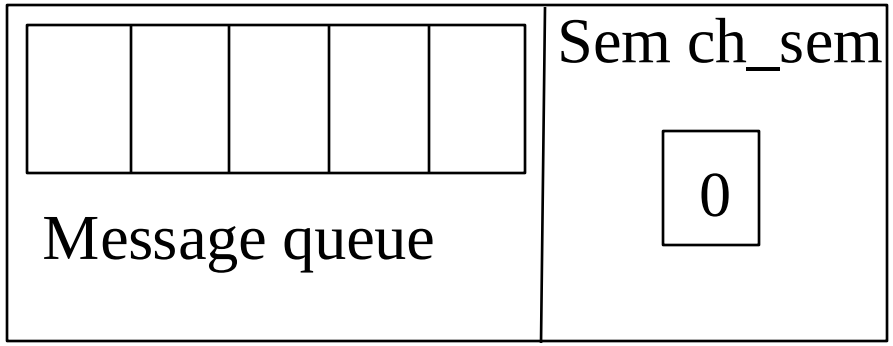
Sender (IP a.b.c.d)

Receiver (IP e.f.g.h)

Send(ch, x)

(Nothing happening here yet)

Send outgoing buffer



Channel ch

Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Data at Receiver Before Receive

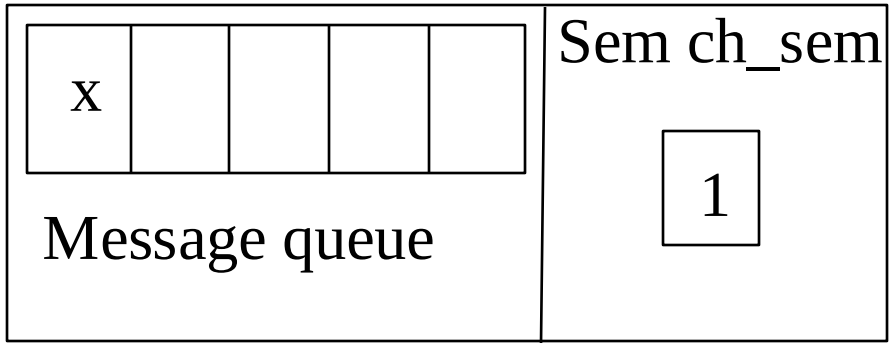
Sender (IP a.b.c.d)

Receiver (IP e.f.g.h)

(Sender can continue)

(Nothing happening here yet)

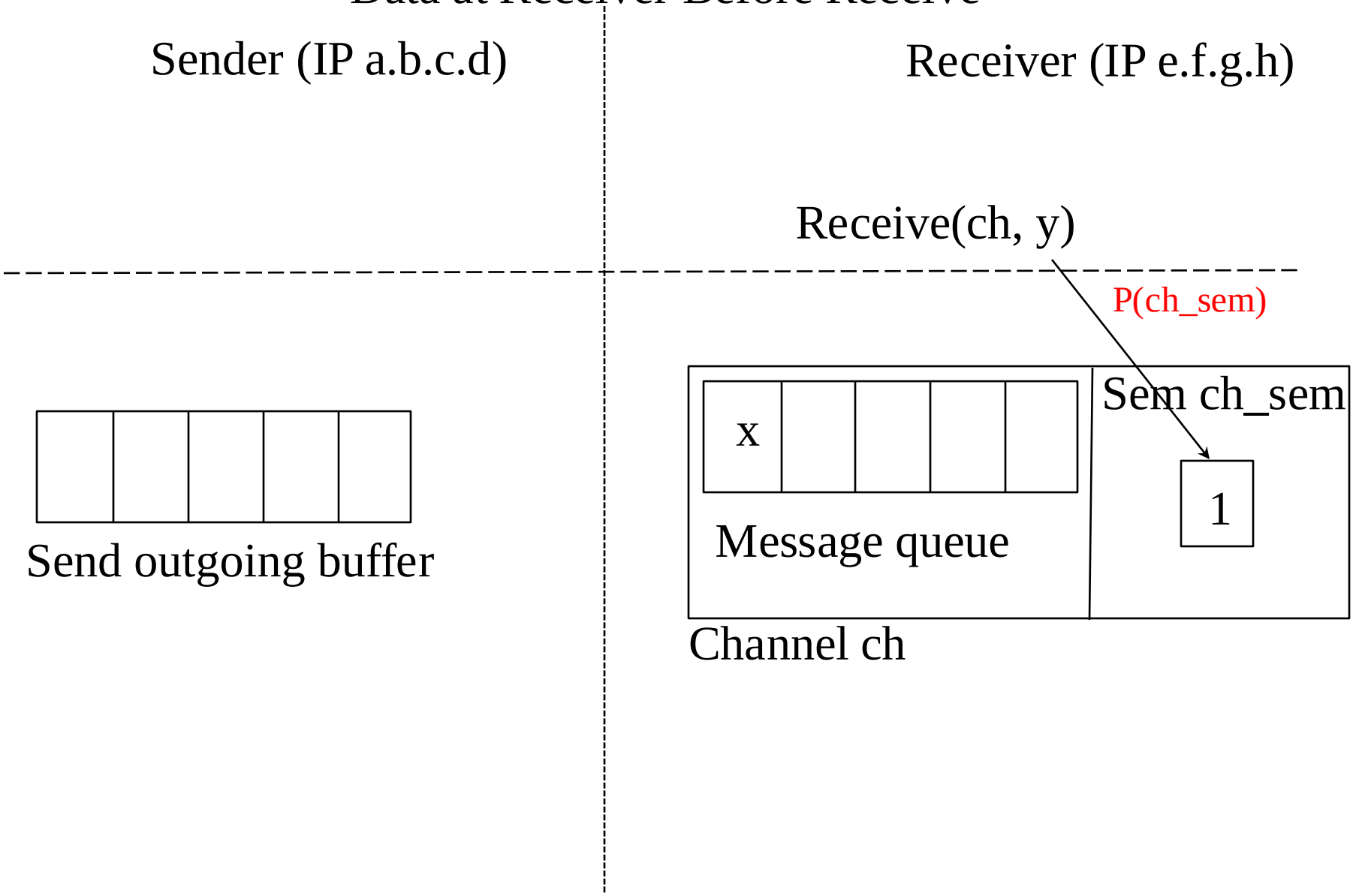
Send outgoing buffer



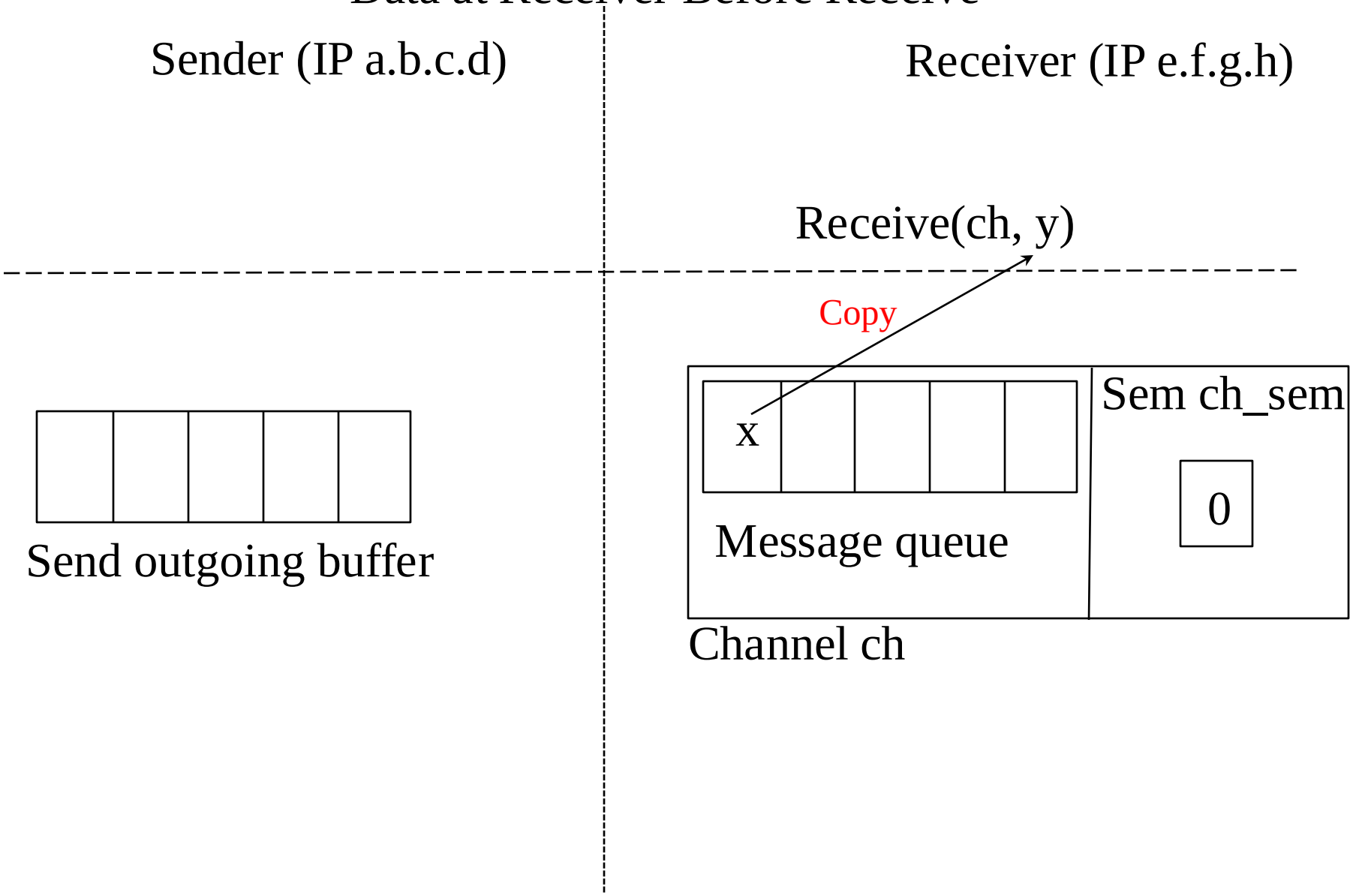
Channel ch

Skipped many steps where message transmitted over network and arrived

Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Data at Receiver Before Receive



Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Data at Receiver Before Receive



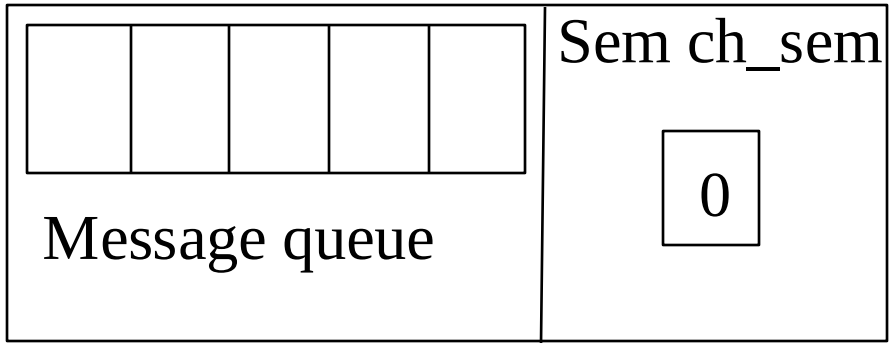
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive)

Sender (IP a.b.c.d)

Receiver (IP e.f.g.h)

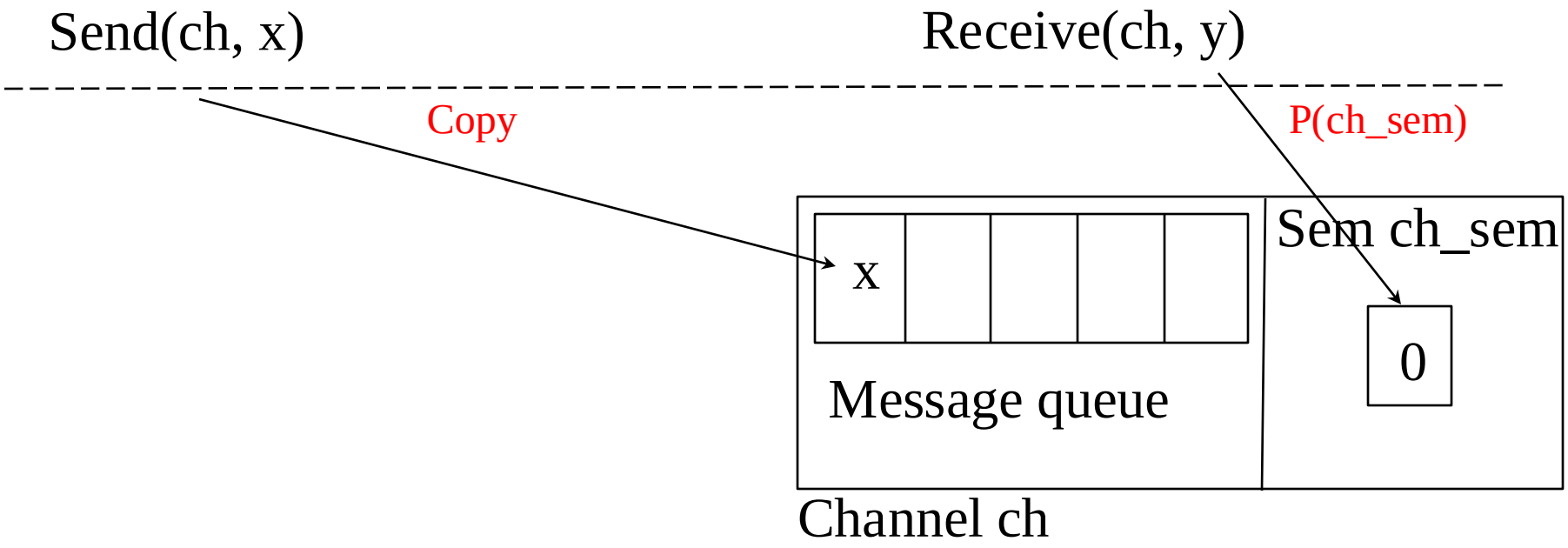
Receive(ch, y) y is now == x

Send outgoing buffer

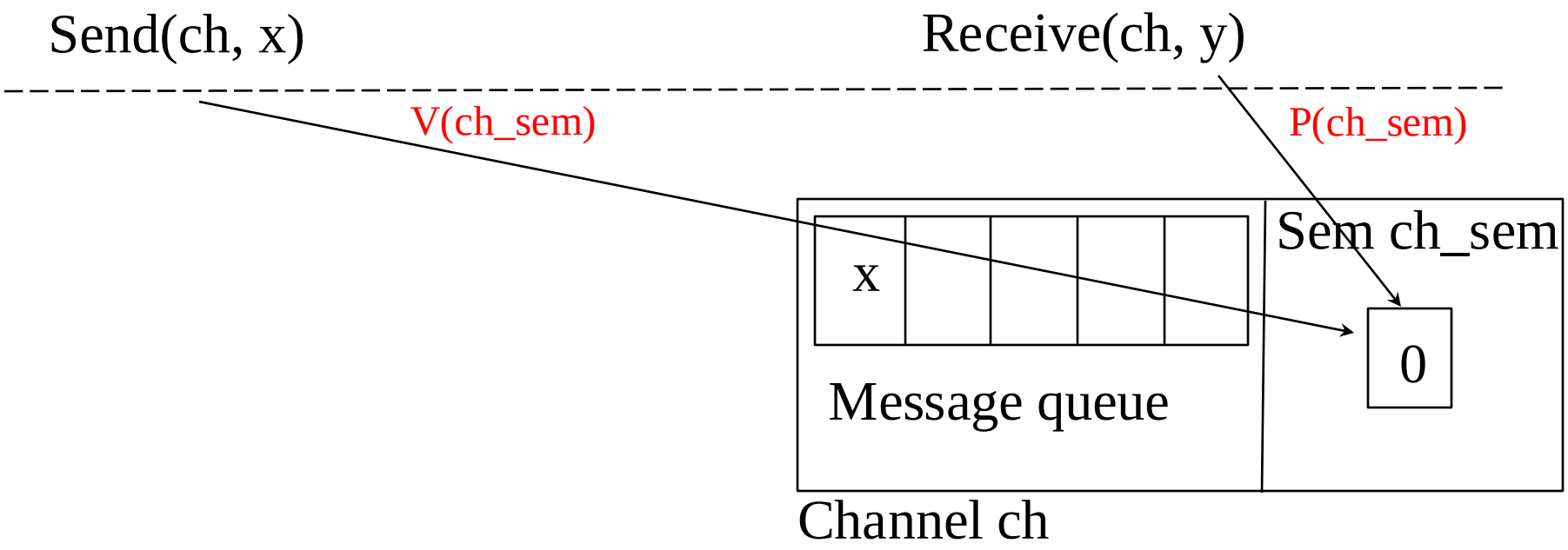


Channel ch

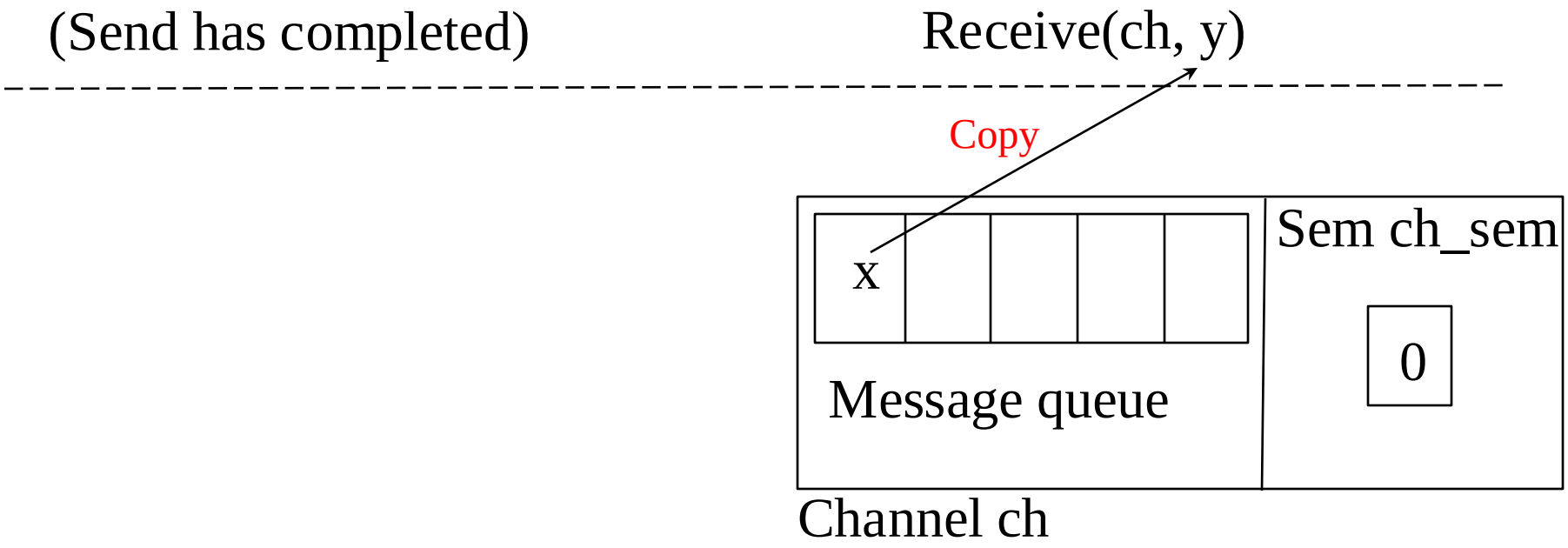
Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Sender and Receiver on same machine



Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Sender and Receiver on same machine

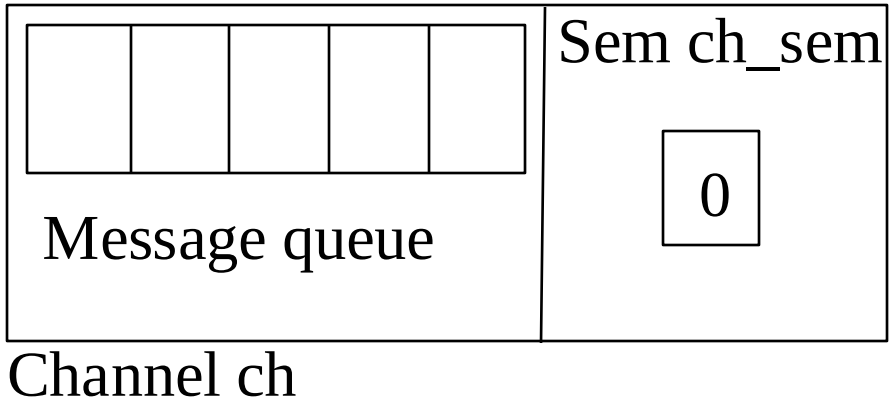


Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Sender and Receiver on same machine



Picture of Message Passing Implementation (Asynchronous Send, Synchronous Receive) Sender and Receiver on same machine

Receive(ch, y) y is now == x



Tradeoffs in Message Passing

- Advantages of synchronous send
  - won't overwrite message, less buffering
- Advantages of asynchronous send
  - can continue after send (can do other work)
  - but what if the send buffer is full? Block? Fail?
- Advantages of synchronous receive
  - know data is available when receive completes, avoid polling
- Advantages of asynchronous receive
  - can result in fewer copies (buffer posted in advance)

Sieve of Eratosthenes: general idea to find primes <= N

1. Add the number 2 to the set of primes
2. Create a list of consecutive odd integers
3. Let $p = 3$, the first prime number (besides 2)
4. Count up from p and cross out all multiples of p
5. Add p to the set of primes
6. Set p = the smallest number not crossed out
7. If $p <= N$, goto step 4; else quit

Sieve of Eratosthenes Using Synchronous Message Passing

```
process Sieve[1] {
    for j = 3 to N by 2
        Sieve[2]!j
}
```

```
process Sieve[i = 2 to Max] {
    Sieve[i-1]?p
    print "found a prime", p
    while (Sieve[i-1]?num) {
        if (num mod p != 0)
            Sieve[i+1]!num
    }
}
```

- Uses CSP notation: ? (receive) and ! (send)
- Terminates in deadlock, but this could be fixed
- Max must be large enough to guarantee sufficient primes generated