Multiagent Vision for Distributed Systems

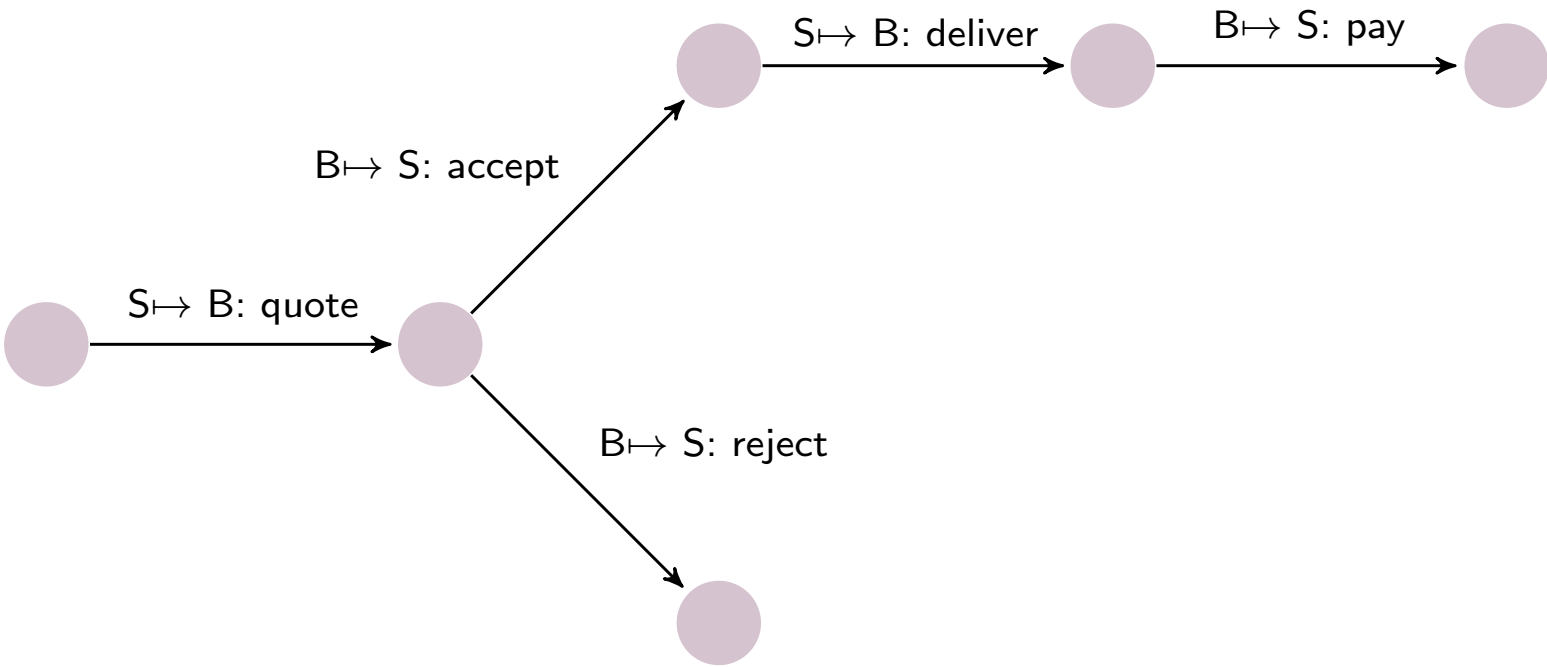
Amit K. Chopra (Lancaster)
Samuel H. Christie V (Lancaster and NC State)
Munindar P. Singh (NC State)

@Imperial College, London

May 6, 2021

Communications: Meaning Matters, Not Relative Order

MAS tenet since Yolum and Singh, 2002



- Choreographies are inflexible
  - *pay* before *accept*? *pay* before *deliver*?
- Specify commitments instead
  - *quote* creates commitment from S to B that *deliver* if *pay*

Distributed Systems Research: Opposite Take

Ignore meaning and provide ordering guarantees in application's communication infrastructure (TCP, message queues, service meshes, etc.)

- Motivated from ease of programming
- Violates the *end-to-end argument* (E2EA, Saltzer et al. 1981)
  - Guarantees may be inadequate for application needs
  - Guarantees may interfere with application needs

Inadequacy of Delivery Guarantees

An agent wants to know if other agents have "processed" messages

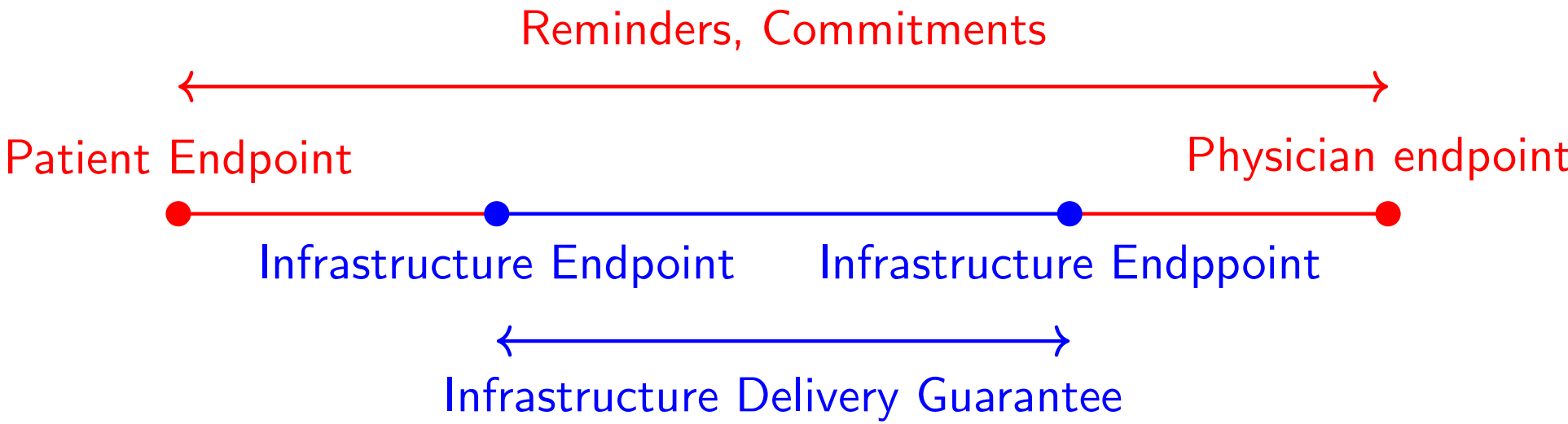
- Medical prescription scenario: Patient, Physician, Pharmacy
  - "Fault": Physician may not have taken up Patient's complaint

- **Reminder Requirement**: Patient may remind Physician by retransmitting complaint until acked by Physician

- **Commitment Requirement**: Physician and Pharmacy act in a timely manner

Redundancy of Delivery Guarantees

Hits performance

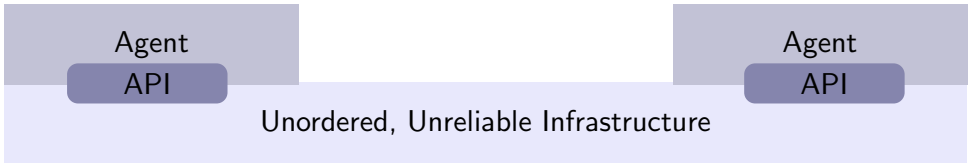


Antiflexibility of FIFO Ordering Guarantee

- **Flexibility-Causally-Unrelated**: Pharmacy can process prescriptions in any order
- **Flexibility-Causally-Related**: Pharmacy can process prescription cancellation before receiving the prescription

The Grand Dilemma of Distributed Systems

Helpful API but cannot realize flexibilities or performance

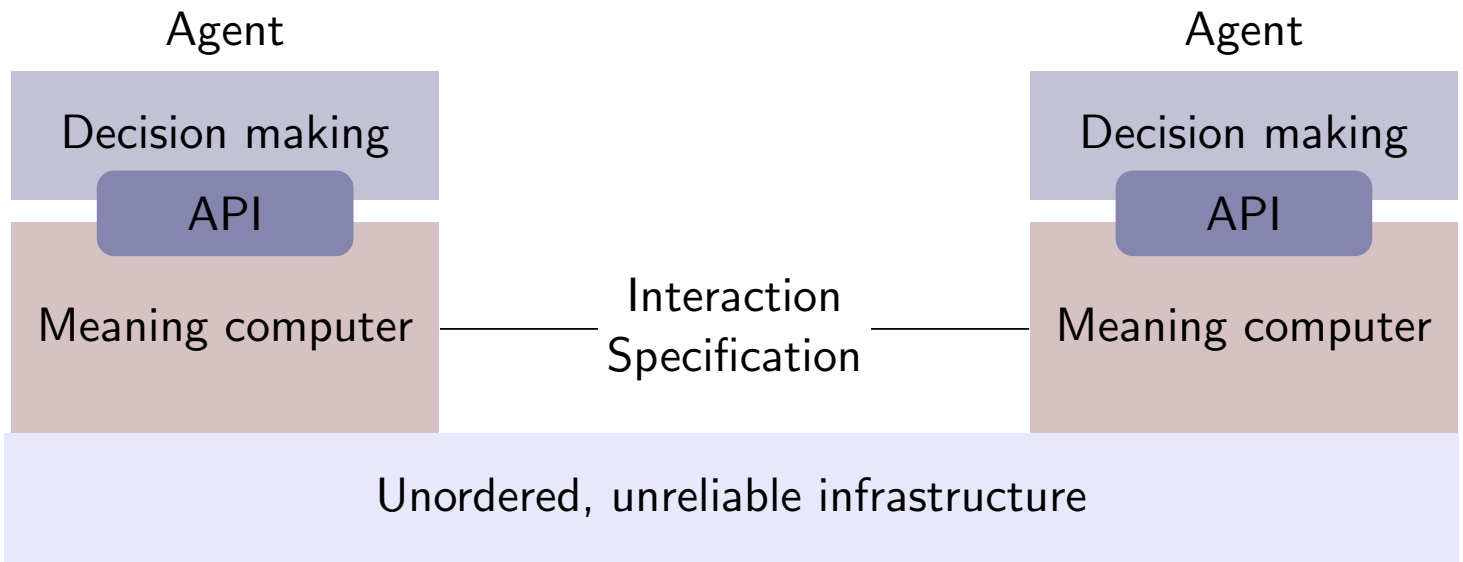


Enables realizing flexibilities and performance but only in an hoc way

- In either case, no support for implementing Reminders and Commitments

MAS Breakthrough: Information protocols (Singh, 2011)

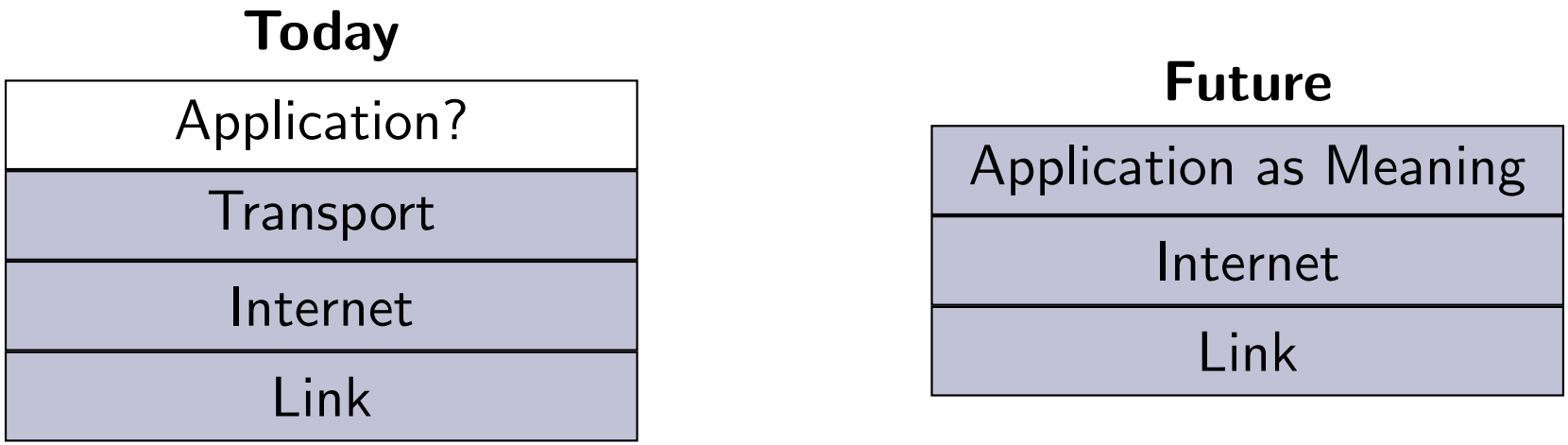
Enables building flexible, robust, and high-performance distributed applications



- Specify information causality, not message ordering
  - Constraints on message emission
  - No constraints on message reception
  - Messages may be retransmitted
- Commitments layered on top

Application-Level Abstractions

Big hole in distributed systems research but MAS specialty



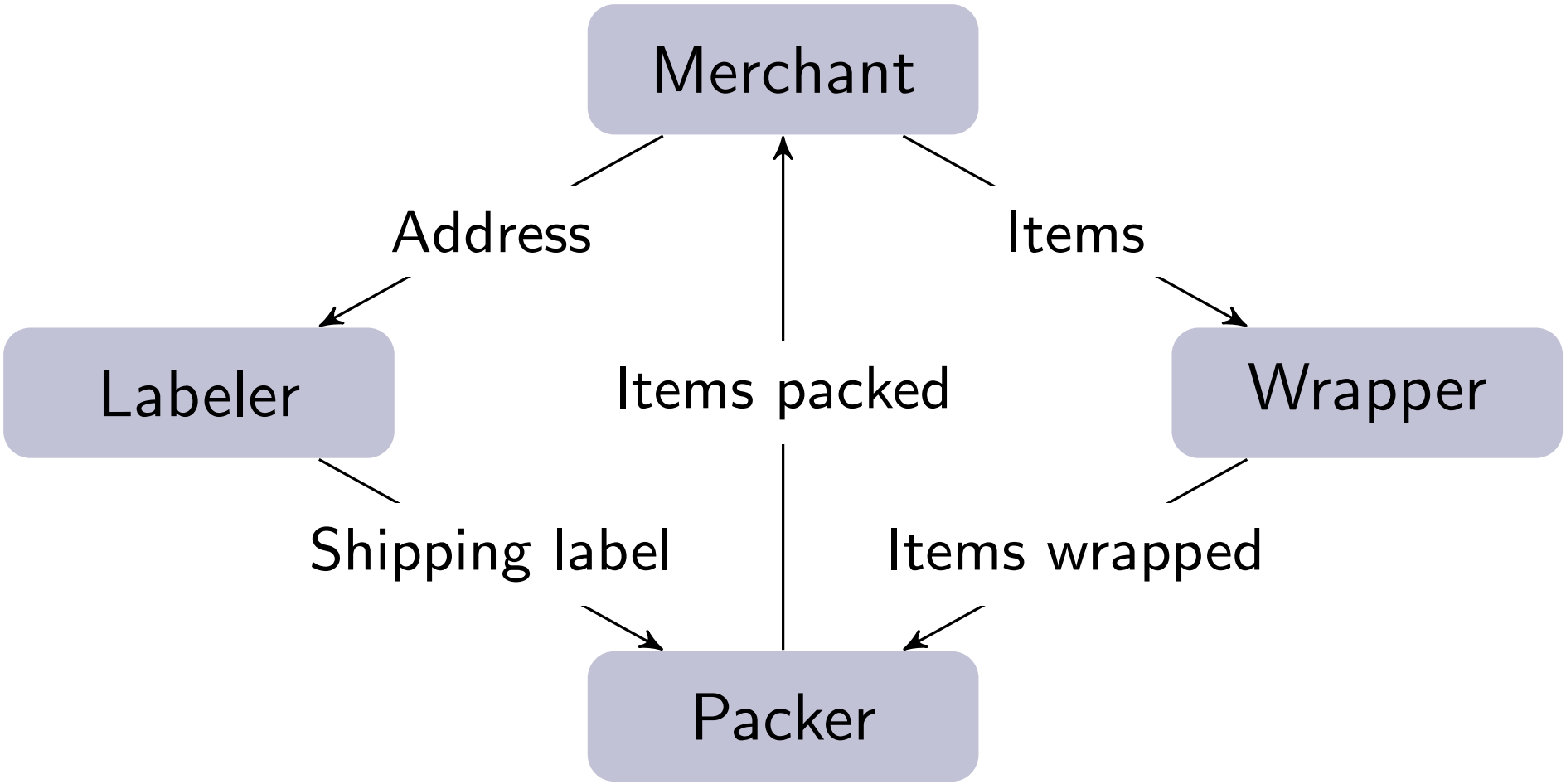
- Meaning-based MAS approaches
  - Protocols, commitments, norms, etc.
  - Overturn conventional distributed systems wisdom
  - Novel foundation for distributed systems

Additional References

- Challenges for Distributed Systems (2015 SOSP History Day)
  - David Clark: `https://youtu.be/v8avB28SlfM?t=1525`
  - Ken Birman: `https://youtu.be/4tN_mJcMOYI?t=488`
- Our contributions
  - PoT (IEEE Computer): `https://www.lancaster.ac.uk/staff/chopraak/pdfs/pot.pdf`
  - Bungie (IEEE Computer): `https://www.lancaster.ac.uk/staff/chopraak/pdfs/Bungie.pdf`
  - Mandrake (ongoing)

Logistics Scenario

Several items in a customer's order that may be wrapped and packed independently to create a shipment



The *Logistics* Protocol

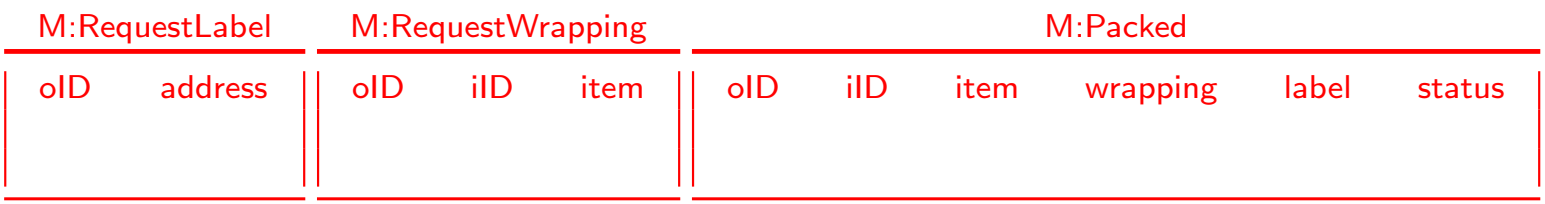
Notice the composite key ⟨oID,iID⟩

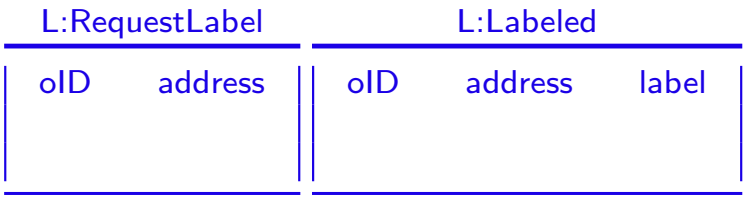
```
Logistics {

  role M, W, L, P  /*Merchant, Wrapper, Labeler, Packer*/
  parameter out oID key, out iID key, out item, out status

  M ↦ L: RequestLabel[out oID key, out address]

  M ↦ W: RequestWrapping[in oID key, out iID key, out item]

  W ↦ P: Wrapped[in oID key, in iID key, in item,
                 out wrapping]

  L ↦ P: Labeled[in oID key, in address, out label]

  P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
                in label, out status]
}
```

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```





Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK

M:RequestWrapping

oID	iID	item

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address

L:Labeled

oID	address	label

W:RequestWrapping

oID	iID	item

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address

L:Labeled

oID	address	label

W:RequestWrapping

oID	iID	item

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
3	a1	fig ×

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address

L:Labeled

oID	address	label

W:RequestWrapping

oID	iID	item

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address

L:Labeled

oID	address	label

W:RequestWrapping

oID	iID	item

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label
1	India	1234 ×

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label
1	UK	1234

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label
1	UK	1234

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Labeled

oID	address	label

P:Wrapped

oID	iID	item	wrapping

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label
1	UK	1234

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Labeled

oID	address	label
1	UK	1234

P:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK

L:Labeled

oID	address	label
1	UK	1234

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Labeled

oID	address	label
1	UK	1234

P:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Packed

oID	iID	item	wrapping	label	status
2	a2	jam	silk	1234	ok ×

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK
2	US

L:Labeled

oID	address	label
1	UK	1234
2	US	abcd

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Labeled

oID	address	label
1	UK	1234
2	US	abcd

P:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Packed

oID	iID	item	wrapping	label	status

Enactments of *Logistics*

Agents act based upon *local state* (set of messages sent and received)

```
M ↦ L: RequestLabel[out oID key, out address]
M ↦ W: RequestWrapping[in oID key, out iID key, out item]
W ↦ P: Wrapped[in oID key, in iID key, in item,
               out wrapping]
L ↦ P: Labeled[in oID key, in address, out label]
P ↦ M: Packed[in oID key, in iID key, in item, in wrapping,
              in label, out status]
```

M:RequestLabel

oID	address
1	UK
2	US

M:RequestWrapping

oID	iID	item
2	a1	fig
2	a2	jam

M:Packed

oID	iID	item	wrapping	label	status

L:RequestLabel

oID	address
1	UK
2	US

L:Labeled

oID	address	label
1	UK	1234
2	US	abcd

W:RequestWrapping

oID	iID	item
2	a2	jam

W:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Labeled

oID	address	label
1	UK	1234
2	US	abcd

P:Wrapped

oID	iID	item	wrapping
2	a2	jam	silk

P:Packed

oID	iID	item	wrapping	label	status
2	a2	jam	silk	abcd	ok

Programming Interface Idea

Packer's reactors for handling Labeled and Wrapped messages. Asynchrony and correlation abstracted away!

```
react(Labeled l) {
  List<Packed> pList = getEnabledPacked(l.oID)

  for Packed p in pList
      p.setStatus('EasyPeasy!')
      adapter.send(p)
}

react(Wrapped w) {
    Packed p = getEnabledPacked(w.oID, w.iID)
    p.setStatus('EasyPeasy!')
    adapter.send(p)
}
```

Ultimate Programming Interface Idea

```
enabled(Packed p)
    if(p.item is perishable)
        p.status = ``easyPeasy'';
        adapter.send(p)
```

Failure of Expectations

Merchant may consider it important to receive *Packed* messages for items in certain orders (say oID 2)

M:RequestLabel

oID	address
1	UK
2	USA

M:RequestWrapping

oID	iID	item
1	a1	fig
2	a2	pears
2	a3	jam

M:Packed

oID	iID	item	wrapping	label	status

W:RequestWrapping

oID	iID	item
2	a2	pears

W:Wrapped

oID	iID	item	wrapping

- W hasn't yet reacted to RequestWrapping(2, a2, pears)
- RequestWrapping(2, a3, jam) is lost
- But loss versus no progress by W are indistinguishable to M
- The protocol could support acknowledgments (by W) and retransmissions (by M)

Acknowledging Message Receipt

Application-level: Requires a response from the receiving agent, which a network-level ack would be insufficient for

```
@acknowledge
M ↦ W: RequestWrapping[in oID key, out iID key, out item]

/*Annotation means the following message is added to the
    protocol*/

W ↦ M: RequestWrappingAck[in oID key, in iID key, out ackID
    key]
```

Sending Reminders ("Retransmission")

Application-level: Regardless of whether the original message is lost or simply because the other agent hasn't yet responded to it

```
@remind
M ↦ W: RequestWrapping[in oID key, out iID key, out item]

/*Annotation means the following message is added to the
    protocol*/

M ↦ W: RequestWrappingReminder[in oID key, in iID key, in
    item, out remID key]
```

Message Reception Order Does Not Cause Faults

Contrasts with current approaches

- Example: Customer sends an AcceptOffer to Merchant and then sends an Instruction to its Bank to pay. Merchant receives Payment from Bank before it receives AcceptOffer.
- Traditional approaches
  - Implement the Merchant to receive AcceptOffer before Payment. Then reception of Payment before AcceptOffer is a fault.
  - Use causal delivery infrastructure. Then Payment is blocked from reception until AcceptOffer is received. Again, undesirable.
- In Mandrake, messages can be received in any order

Let's Raise Our Game...to the Application-Level

- Approaches for supporting *application meaning* the central challenge of distributed systems research
- IOP (Mandrake) is an approach for doing application semantics
  - Model application meaning via application-specific protocol
  - Focus on message meaning, not ordering
- Fault tolerance is application-level; must be reflected in the protocol
- Empower developers via programming models based on protocols
  - Don't try to hide distribution from developer