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



Future										
Application as Meaning										
Internet										
Link										

- 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/v8avB28S1fM?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

 $\mathsf{M} \, \mapsto \, \mathsf{L} \colon \, \mathsf{RequestLabel} \left[\, \mathsf{out} \ \mathsf{olD} \ \mathsf{key} \, , \ \mathsf{out} \ \mathsf{address} \, \right]$

 $M \mapsto W$: RequestWrapping[in olD key, out ilD key, out item]

 $W \mapsto P$: Wrapped[in olD key, in ilD key, in item, out wrapping]

 $L \mapsto P$: Labeled [in olD key, in address, out label]

 $P \mapsto M$: Packed[in olD key, in ilD key, in item, in wrapping, in label, out status]

・ 同 ト ・ ヨ ト ・ ヨ ト …

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Red	questLabel	M:Re	questWr	apping				M:Packed				
olD	address	olD	iID	item	olD	iID	item	wrapping	label	status		
L:Req	uestLabel		L:Labe	eled								
olD	address	olD	addre	ss labe	el							
W:Re	questWrappir	g		W:Wrap	ped		_					
oID	iID iter	n c	olD i	ID iter	n w	rapping						
	P:Labeled			P:V	Vrappec					P:Packed		
oID	address	label	oID	iID	item	wrapp	ing	olD ilD	item	wrapping	label	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]
```

M:RequestLabel	M:RequestWrapping		M:Packed		
oID address 1 UK	olD ilD item	olD ilD iten	n wrapping	label status	
L:RequestLabel	L:Labeled				
oID address	oID address lab	el			
W:RequestWrappir	ng W:Wra	pped			
olD ilD iter	m oID iID ite	m wrapping			
P:Labeled	P:	Wrapped		P:Packed	
oID address	label oID iID	item wrapping	olD ilD	item wrapping	g label status
			1		

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:RequestLabel	M:RequestWrapping		M:Packed		
olD address 1 UK 2 US	olD ilD item	olD ilD iten	n wrapping	label status	
L:RequestLabel	L:Labeled				
oID address	oID address lab	el			
W:RequestWrappir	ng W:Wra	pped			
olD ilD iter	m oID iID ite	m wrapping			
P:Labeled	P:	Wrapped		P:Packed	
oID address	label oID iID	item wrapping	olD ilD	item wrapping	g label status
					NATE TO

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

olD address olD ilD item olD ilD item wrapping label status 1 UK 3 a1 fig × olD ilD item wrapping label status 2 US olD a1 fig × olD ilD item wrapping label olD address olD address label olD address label W:RequestWrapping W:Wrapped olD ilD item wrapping P:Labeled olD ilD item wrapping P:Packed olD address label olD ilD item wrapping 0L address label olD ilD item wrapping	M:RequestLabel	M:RequestWrapping		M:	Packed				
olD address label W:RequestWrapping W:Wrapped olD ilD olD ilD ilD item wrapping P:Wrapped	1 UK		olD ilD	item	wrapping	label	status		
W:RequestWrapping W:Wrapped oID iID item wrapping P:Labeled P:Wrapped P:Packed	L:RequestLabel	L:Labeled	_						
olD ilD item wrapping P:Labeled P:Wrapped P:Packed	oID address	olD address label							
P:Labeled P:Wrapped P:Packed	W:RequestWrappir	ng W:Wrapp	oed						
	olD ilD iter	m olD ilD item	wrapping						
oID address label oID iID item wrapping oID iID item wrapping label status	P:Labeled	P:W	rapped			P	:Packed		
	oID address	label oID iID	item wrappir	ng olD) iID	item	wrapping	label	status

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Reque	stLabel	M:Re	questWr	apping				M:Packed				
olD a 1 2	address UK US	olD 2 2	iID a1 a2	item fig jam	olD	iID	item	wrapping	label	status		
L:Reques	stLabel		L:Labe	led								
olD a	address	olD	addre	s labo	el							
W:Reque	stWrappin	g		W:Wrap	ped		_					
olD i	ID iten	n c	olD il	D iter	n wi	rapping						
P	:Labeled			P:\	Vrapped					P:Packed		
olD a	address	label	oID	iID	item	wrapp	ing	olD ilD	item	wrapping	label	status

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

olD address olD ilD item olD ilD item wrapping label status 1 UK 2 a1 fig olD ilD ilD item wrapping label status L:RequestLabel L:Labeled olD address label label status 0ID address label olD address label label W:RequestWrapping W:Wrapped olD ilD item wrapping 2 a2 jam olD ilD item wrapping P:Labeled P:Wrapped P:Packed P:Packed olD address label olD ilD item wrapping	M:RequestLabel	M:RequestWrapping		M:Packed			
olD address label 1 UK olD address label W:RequestWrapping W:Wrapped olD ilD item wrapping 2 a2 jam olD item wrapping P:Labeled P:Wrapped P:Packed	1 UK	2 a1 fig	olD ilD ite	em wrapping	label status		
1 UK W:RequestWrapping W:Wrapped olD ilD item 2 a2 jam P:Labeled P:Wrapped	L:RequestLabel	L:Labeled					
olD ilD item 2 a2 jam P:Labeled P:Wrapped		oID address lab	el				
2 a2 jam P:Labeled P:Wrapped P:Packed	W:RequestWrappin	ng W:Wrap	oped				
			n wrapping				
oID address label oID iID item wrapping oID iID item wrapping label status	P:Labeled	P:\	Vrapped		P:Packed		
	olD address	label oID iID	item wrapping	olD ilD	item wrapping	; label	status

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Re	questLabel	M:Re	questWra	apping				M:Packed				
olD 1 2	address UK US	olD 2 2	iID a1 a2	item fig jam	olD	iID	item	wrapping	label	status		
L:Rec	luestLabel		L:Lab	eled								
olD 1	address UK	olD 1	addres India		oel 4 ×							
W:Re	questWrappi	ng		W:Wrap	ped							
olD 2	iID ite a2 jai		bID il	D iter	n w	rapping						
	P:Labeled			P:V	Vrappe	d				P:Packed		
oID	address	label	oID	iID	item	wrappi	ng	oID iID	item	wrapping	label	status

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

oID address oID iID item 1 UK 2 a1 fig 2 US 2 a2 jam	status
L:RequestLabel L:Labeled	
oID address label 1 UK 1 UK 1234	
W:RequestWrapping W:Wrapped	
oID iID item oID iID item wrapping 2 a2 jam	
P:Labeled P:Wrapped P:	Packed
olD address label olD ilD item wrapping olD ilD item	vrapping label status

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Rec	questLabel	M:Re	equestWrap	ping				M:Packed				
olD 1 2	address UK US	olD 2 2	al	tem fig jam	olD	iID	item	wrapping	label	status		
L:Req	uestLabel		L:Labele	d								
olD 1	address UK	olD 1	address UK	labe 1234								
W:Red	questWrappi	ıg	Ň	N:Wrap	ped							
olD 2	iID ite a2 jar		oID iID 2 a2	iten jam		rapping silk						
	P:Labeled		_	P:W	/rappe	d				P:Packed		
olD	address	label	oID	iID	item	wrappi	ng	olD ilD	item	wrapping	label	status
									< □ ▶ <		<	5 PQC

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Red	questLabel	M:Re	questWrap	ping				M:Packed				
olD 1 2	address UK US	olD 2 2	al	item fig jam	olD	iID	item	wrapping	label	status		
L:Req	uestLabel		L:Labele	d								
olD 1	address UK	olD 1	address UK	labo 123								
W:Red	questWrappi	ng	,	W:Wrap	ped							
olD 2	iID ite a2 ja		olD ilD 2 a2			vrapping silk						
	P:Labeled			P:\	Vrappe	ed				P:Packed		
olD 1	address UK	label 1234	oID 2	iID a2	item jam	wrappii silk	ng	oID iID	item	wrapping	label	status
									4 m h - 4		4 E 5	3

```
M → L: RequestLabel[out olD key, out address]
M → W: RequestWrapping[in olD key, out ilD key, out item]
W → P: Wrapped[in olD key, in ilD key, in item,
out wrapping]
L → P: Labeled[in olD key, in address, out label]
P → M: Packed[in olD key, in ilD key, in item, in wrapping,
in label, out status]
```

M:Red	questLabel	M:Re	equestWrap	ping				M:Packed					
olD 1 2	address UK US	olD 2 2	iID a1 a2	item fig jam	olD	ilD i	tem	wrappi	ng I	abel	status		
L:Req	uestLabel		L:Labele	d									
olD 1	address UK	olD 1	address UK	labe 1234									
W:Ree	questWrappi	ıg		W:Wrap	ped								
olD 2	iID ite a2 jar		olD ilD 2 a2	iten jarr		rapping silk							
	P:Labeled			P:V	Vrappe	d					P:Packed		
olD 1	address UK	label 1234	oID 2	iID a2	item jam	wrapping silk	g			tem jam	wrapping silk	label 1234	status ok 🗙
									< □] ▶ ∢	日本・日本	<= ≣⇒	1 9 9 9 P

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

M → L: RequestLabel[out olD key, out address] M → W: RequestWrapping[in olD key, out ilD key, out item] W → P: Wrapped[in olD key, in ilD key, in item, out wrapping] L → P: Labeled[in olD key, in address, out label] P → M: Packed[in olD key, in ilD key, in item, in wrapping, in label, out status]

M:RequestLabel M:Requ			Reques	stWrap	oing		M:Packed							
olD 1 2	address UK US	olE 2 2	a	1	tem fig am	olD) iID	item	wra	apping	label	status		
L:RequestLabel			L:	Labelec	I									
olD address olD 1 UK 1 2 US 2		i a	ddress UK US	lab 123 abo	34									
W:Re	questWrapp	oing		V	V:Wra	pped		_						
olD 2		em am	olD 2	iID a2	ite jai		wrapping silk							
			P:	Wrapp	ed					P:Packed				
olD 1 2	address UK US	labe 1234 abco	+ -	oID 2	iID a2	item jam			oID	iID	item	wrapping		

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

M → L: RequestLabel[out olD key, out address] M → W: RequestWrapping[in olD key, out ilD key, out item] W → P: Wrapped[in olD key, in ilD key, in item, out wrapping] L → P: Labeled[in olD key, in address, out label] P → M: Packed[in olD key, in ilD key, in item, in wrapping, in label, out status]

M:RequestLabel M:F			M:RequestWrapping			M:Packed							
olD 1 2	address UK US	olD 2 2	a1	item fig jam	olD	iID	item	wrap	ping	label	status		
L:RequestLabel			L:Labele	d									
olD 1 2	address UK US	olD 1 2	address UK US	labe 123 abc	4								
W:Red	W:RequestWrapping W:Wrap				ped								
olD 2	iID ite a2 jar		olD ilD 2 a2	iter jan		rapping silk							
P:Labeled P					Vrappe	d					P:Packed		
olD 1 2	address UK US	label 1234 abcd	olD 2	iID a2	item jam	wrappir silk	ng	oID 2	iID a2	item jam	wrapping silk	label abcd	status ok
										• • •		< ∃ >	500 E

Programming Interface Idea

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

```
react(Labeled I) {
  List < Packed > pList = getEnabledPacked(1.olD)
  for Packed p in pList
      p.setStatus('EasyPeasy!')
      adapter.send(p)
}
react(Wrapped w) {
   Packed p = getEnabledPacked(w.olD, 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:Red	questLabel	M:Red	questW	/rapping	M:Packed							
olD 1 2	address UK USA	olD 1 2 2	iID a1 a2 a3	item fig pears jam	olD	iID	item	wrapping	label	status		
W:Re	W:RequestWrapping W:Wrap						_					
olD 2	iID ite a2 pea		olD ilD iter		m wr	apping						

- 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

<code>@acknowledge M \mapsto W: RequestWrapping[in olD key, out ilD key, out item]</code>

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

 $W\mapsto M\colon RequestWrappingAck[in olD key, in ilD key, out acklD key]$

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

 $\mathsf{M} \mapsto \mathsf{W}: \ \mathsf{RequestWrapping} [\text{ in olD key, out ilD key, out item }]$

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

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