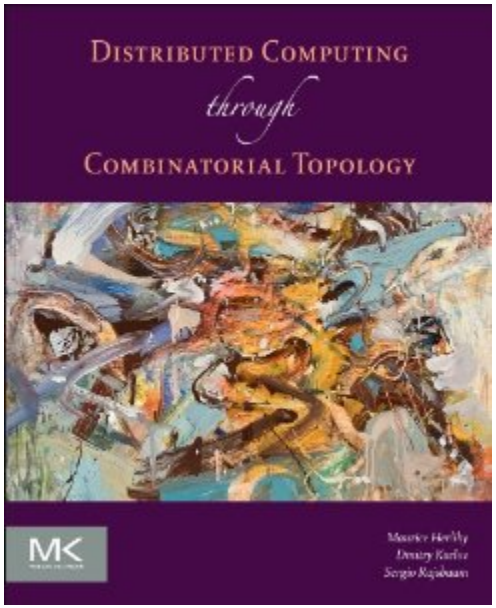
Byzantine-Resilient Colorless Computaton



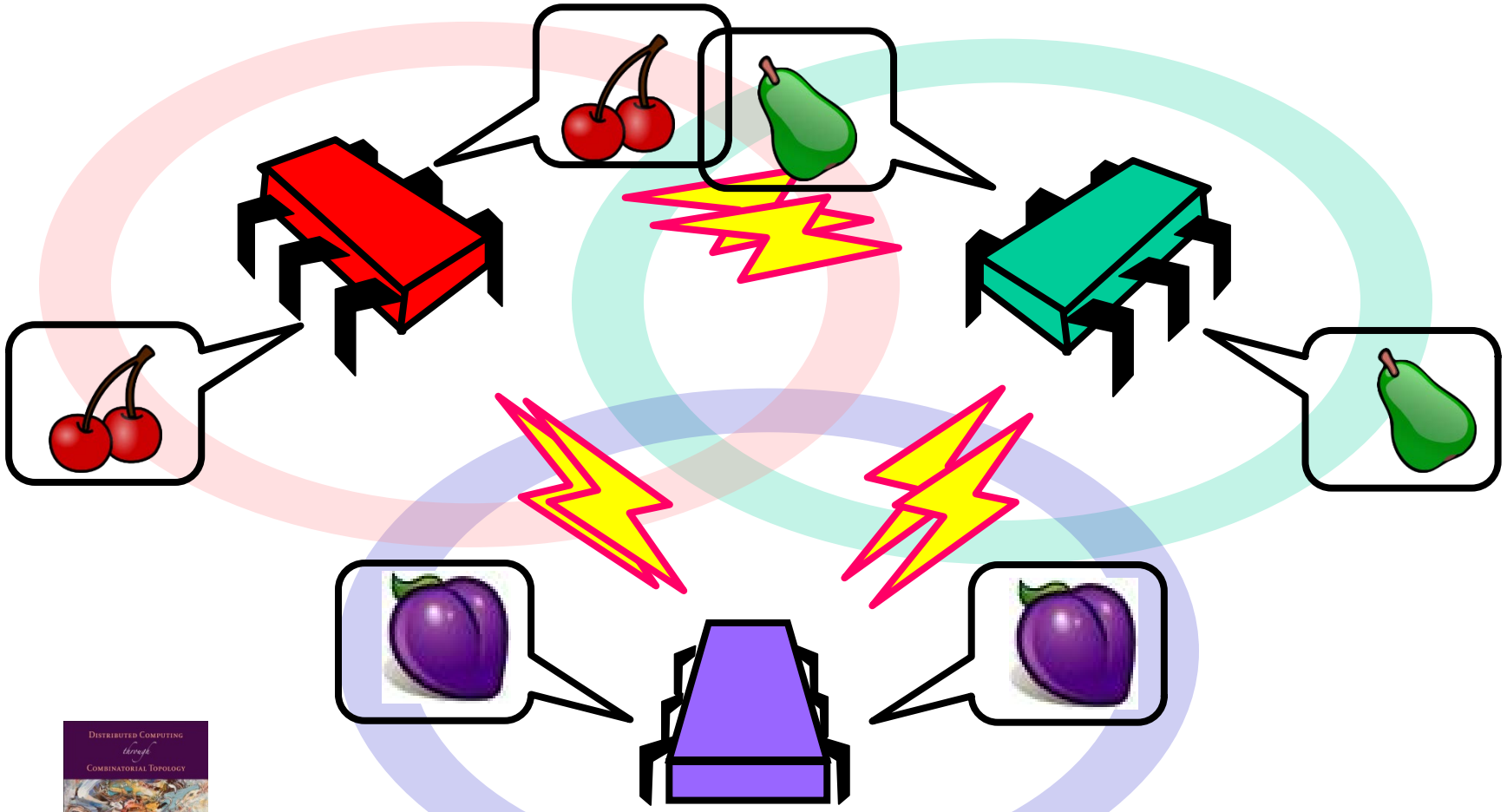
Companion slides for

Distributed Computing

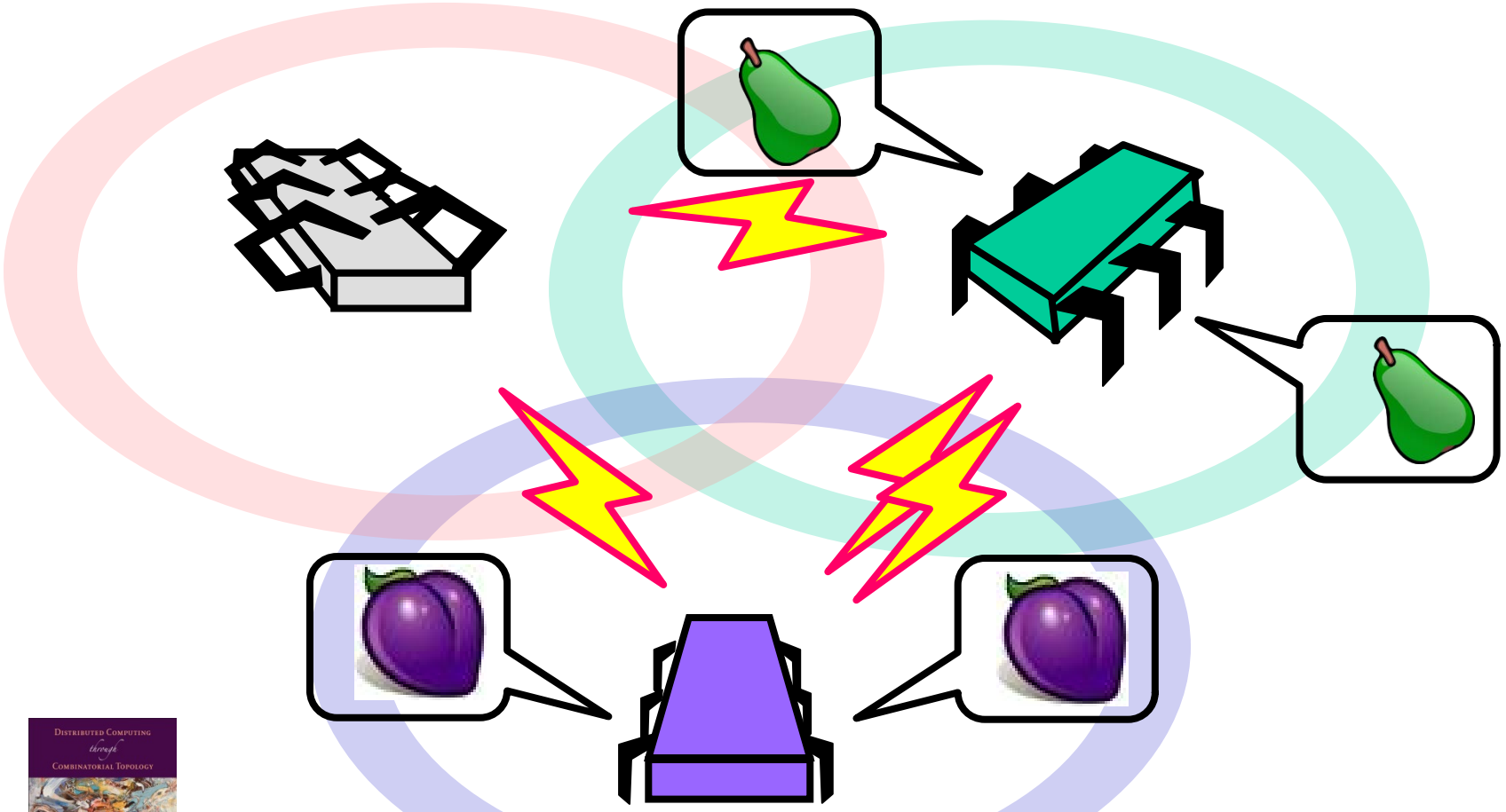
Through Combinatorial Topology

Maurice Herlihy & Dmitry Kozlov & Sergio Rajsbaum

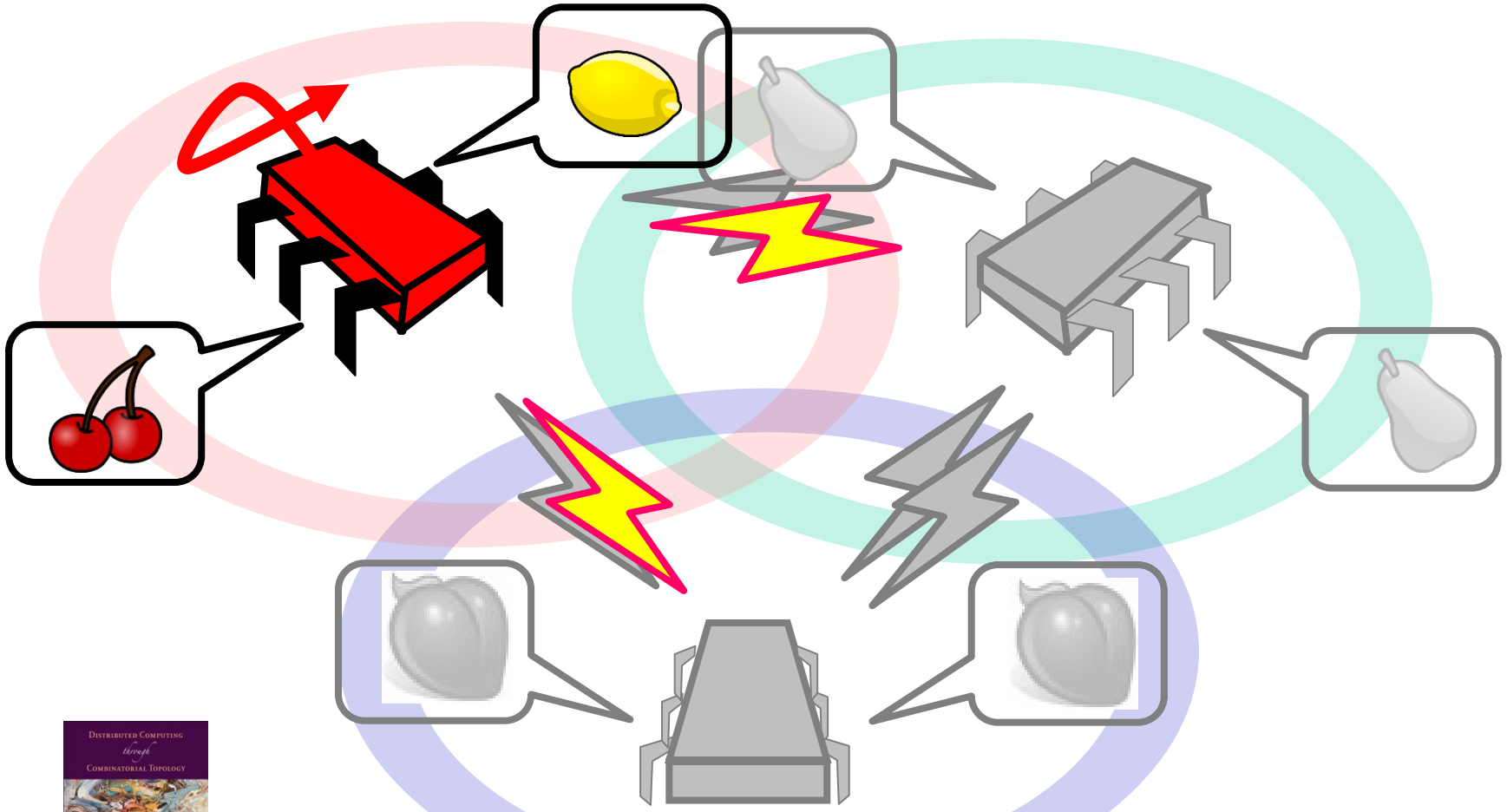
Message-Passing



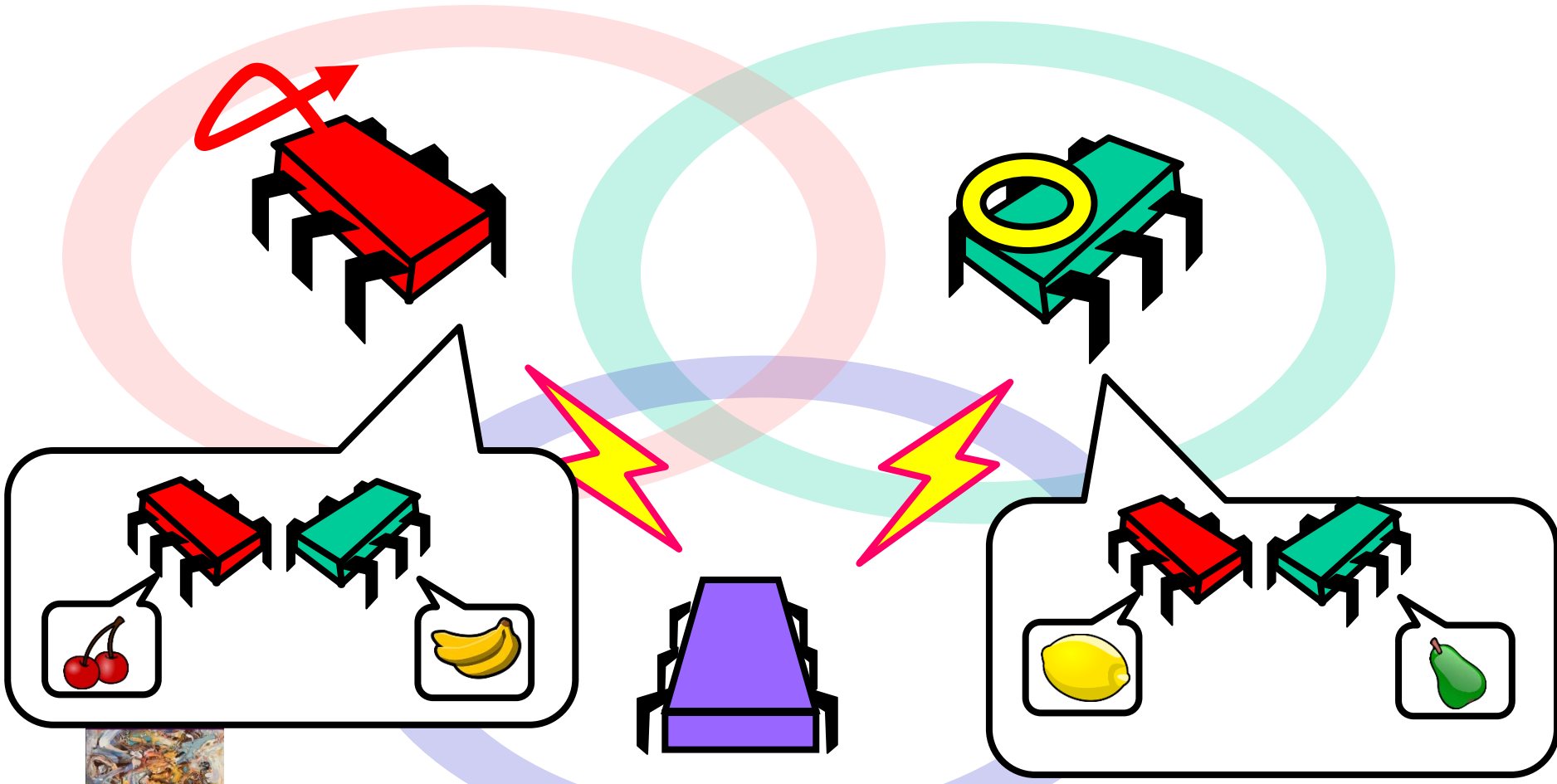
Crash Failures



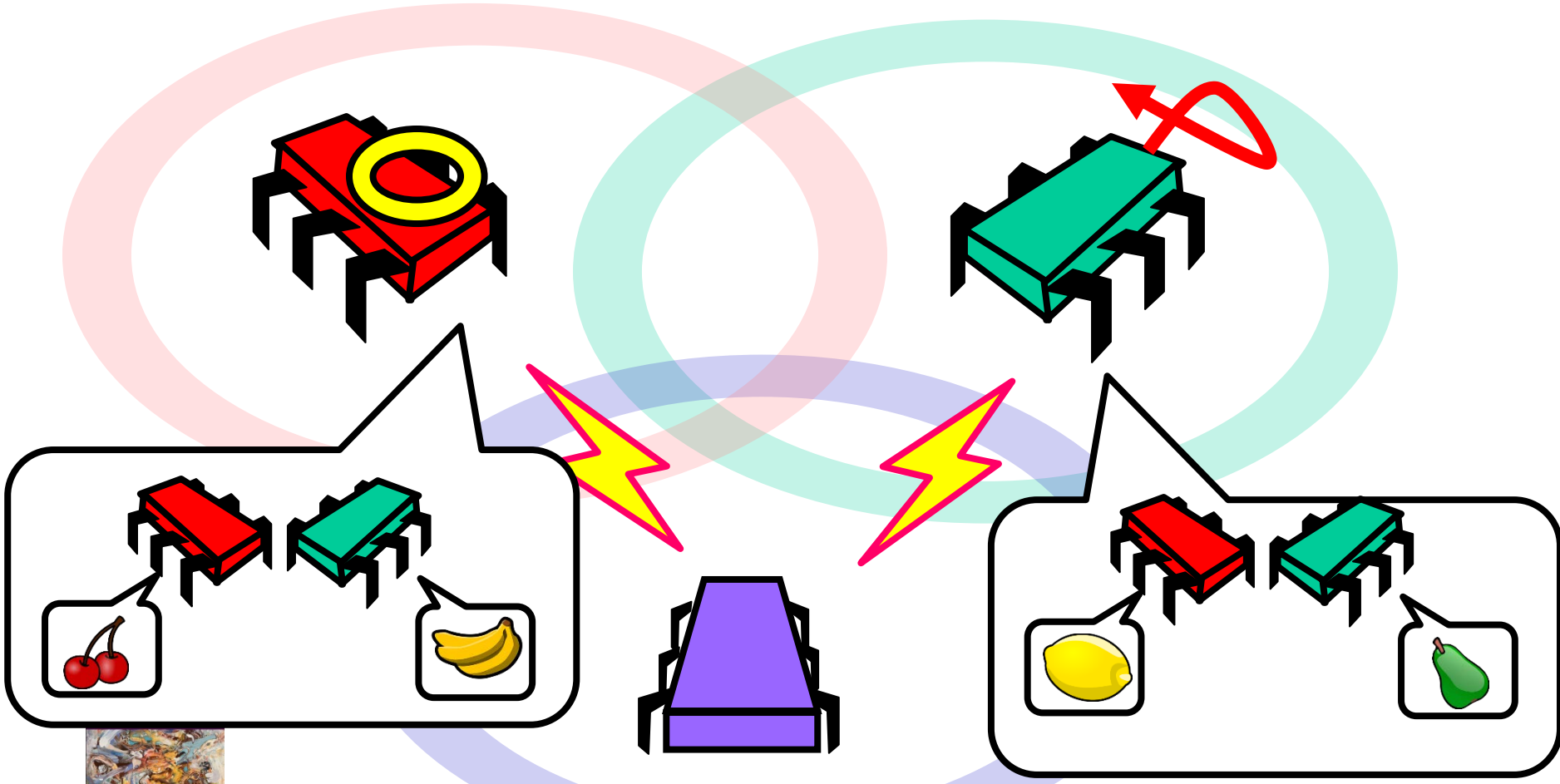
Byzantine Failures



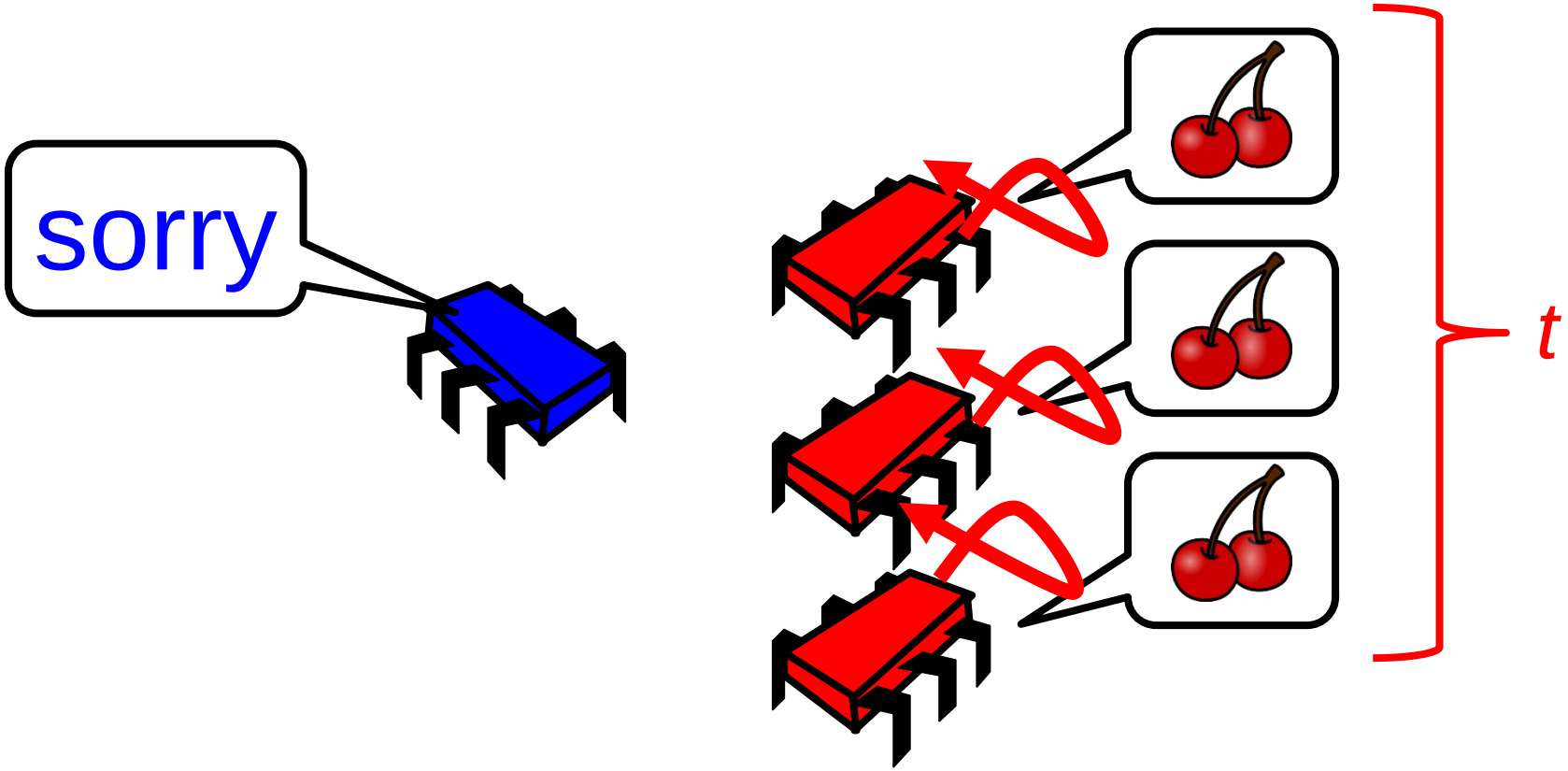
He said, She said …

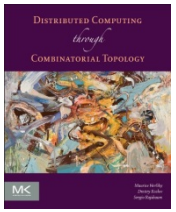


He said, She said …

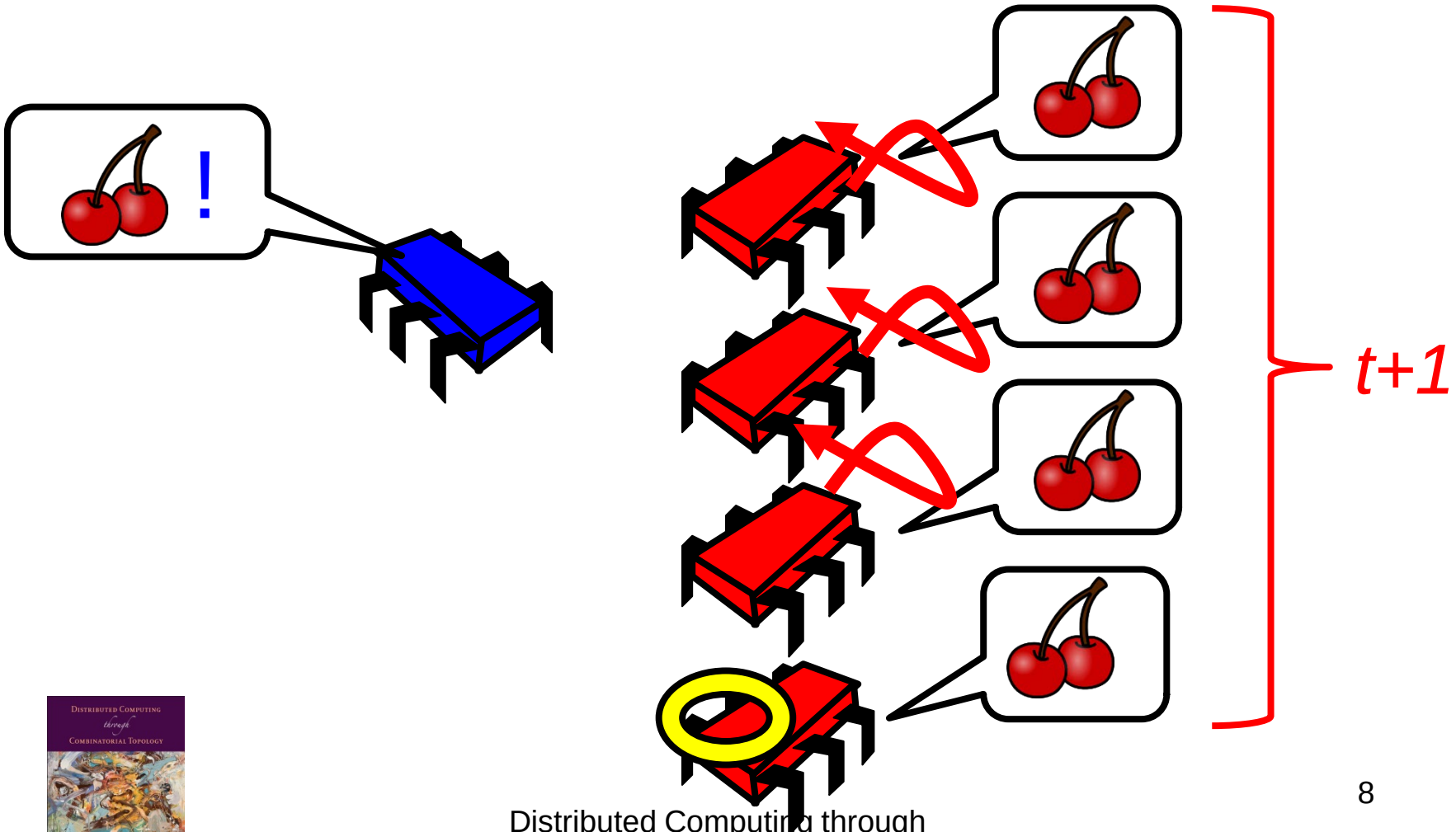


Identifying Input Values





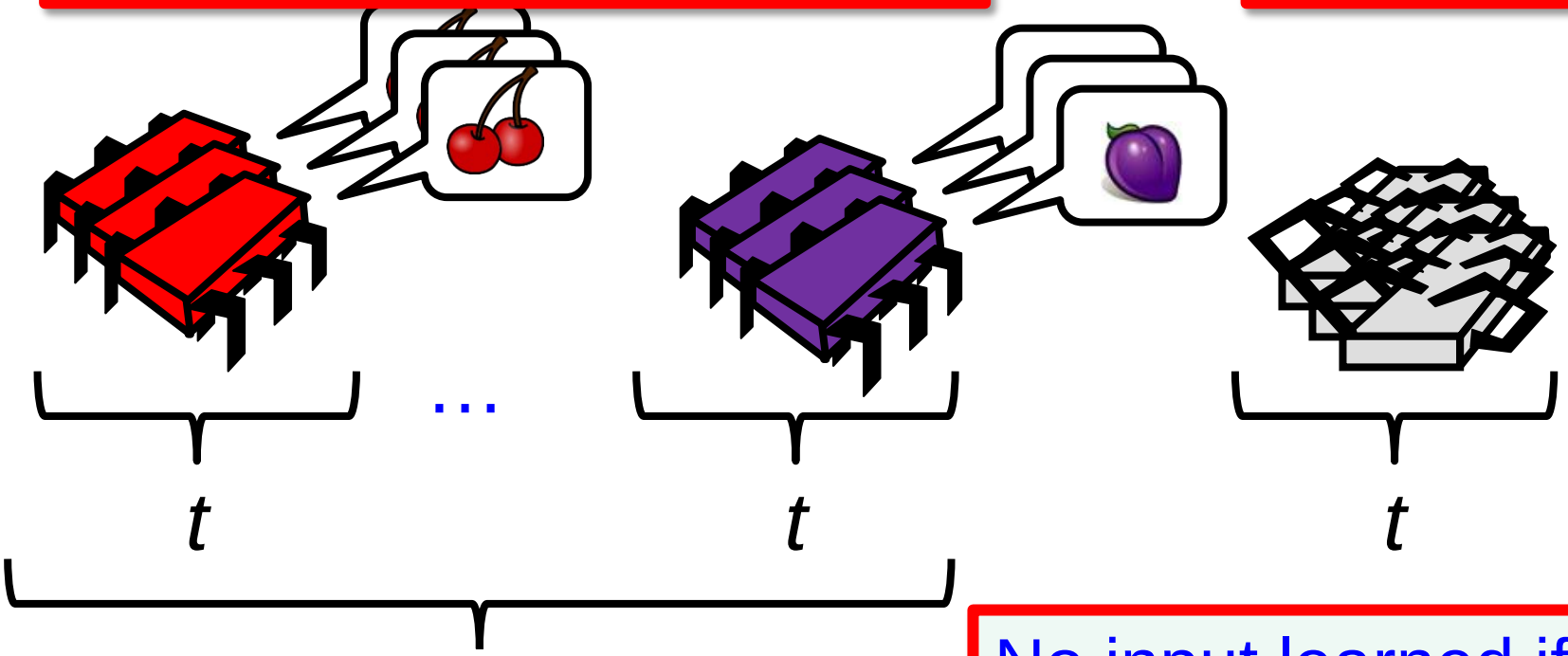
Identifying Input Values



Nothing Learned

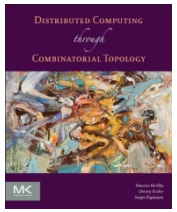
each input reported by only t

t are silent



(dim $\mathcal{I}$ + 1) t

No input learned if
n+1 $\leq$ (dim $\mathcal{I}$+2) t



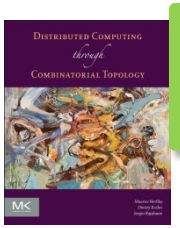
Requirement

requirement that:

$$n+1 > (\dim \mathcal{I} + 2)\, t$$

means t is "pretty small"

Byzantine is harder!

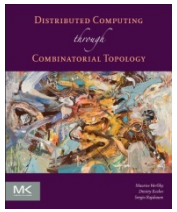


dim $\mathcal{I}$ irrelevant for crash failures

Strict Tasks

$$\Delta(\sigma_0) \cap \Delta(\sigma_1) = \Delta(\sigma_0 \cap \sigma_1)$$

usually require only $\subseteq$



Main Result

$(\mathcal{I},\mathcal{O},\Delta)$ has a wait-free Byzantine protocol …

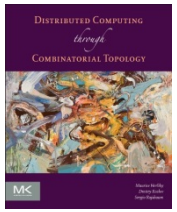
if and only if …

there is a continuous map

f: $|\text{skel}^t\ \mathcal{I}| \rightarrow |\mathcal{O}|$

carried by Δ.

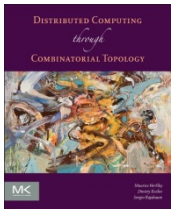
where n+1 > (dim $\mathcal{I}$ + 2) t



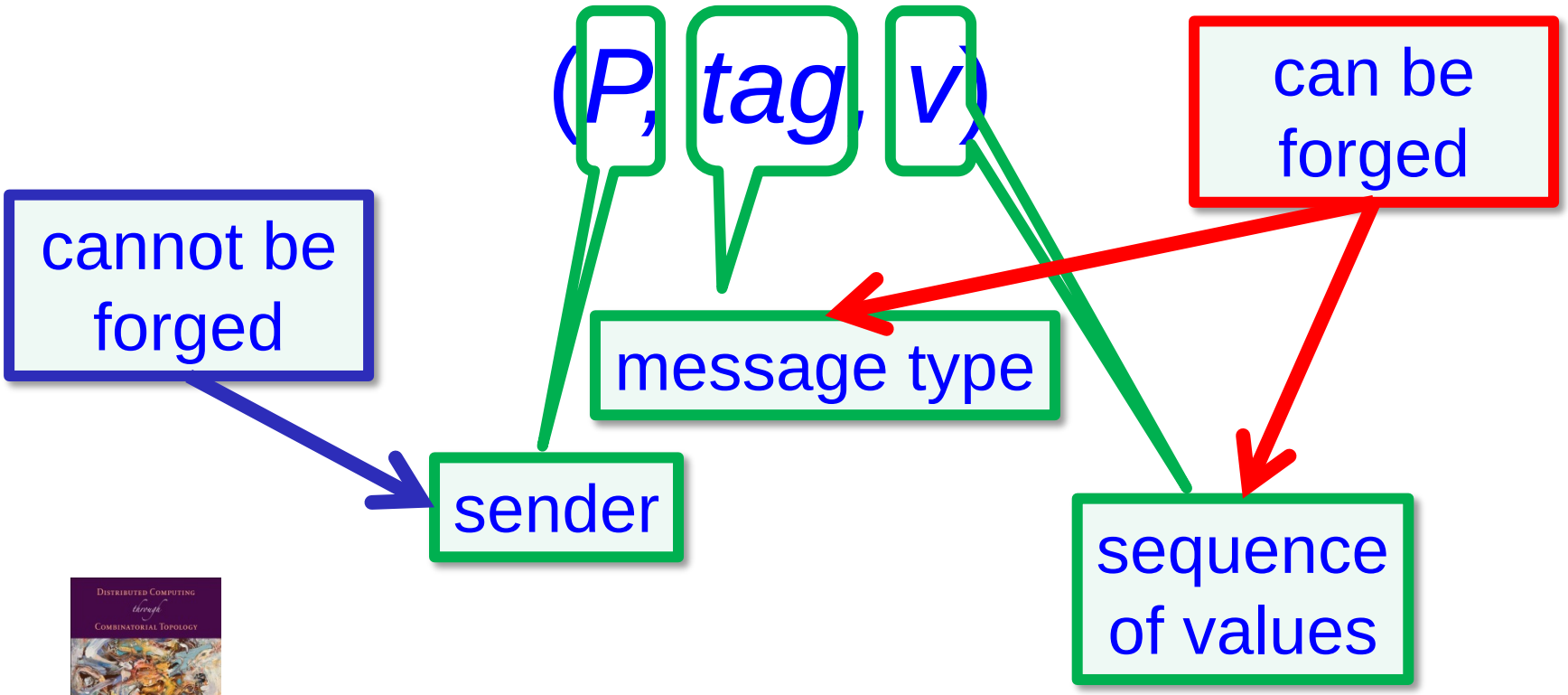
Communication

message-passing

asynchronous layers



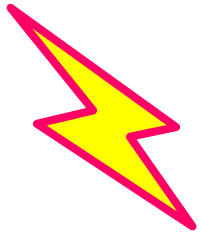
Messages


(P, tag, v)
cannot be forged
can be forged
message type
sender
sequence of values

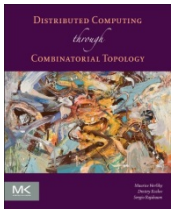
Distributed Computing through Combinatorial Topology

Reliable Broadcast

RBSend(*P, tag, v*)



RBReceive(*P, tag, v*)



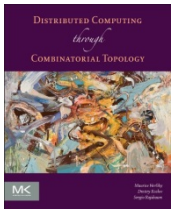
Non-Faulty Integrity

If non-faulty *P* never broadcasts (*P*,*tag*,*v*) …

via RBSend(*P, tag, v*)

Non-faulty *Q* never receives (*P*,*tag*,*v*)

via RBReceive(*P, tag, v*)



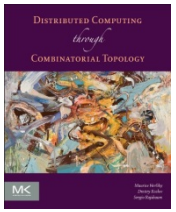
Non-Faulty Liveness

If non-faulty *P* does broadcast (*P,tag,v*) …

via RBSend(*P, tag, v*)

Every non-faulty *Q* receives (*P,tag,v*)

via RBReceive(*P, tag, v*)



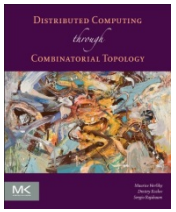
Global Uniqueness

If non-faulty *Q*,*R* reliably receive…

(*P*,*tag*,*v*), (*P*,*tag'*,*v'*) respectively …

then *tag* = *tag'* and *v* = *v'*

even if *P* is faulty



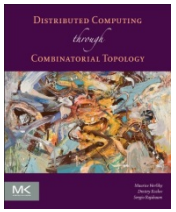
Global Liveness

for non-faulty *Q*,*R* …

if *Q* reliably receives (*P*,*tag*,*v*)…

then R reliably receives (*P*,*tag*,*v*)…

even if *P* is faulty

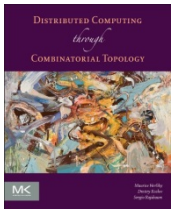


Summary

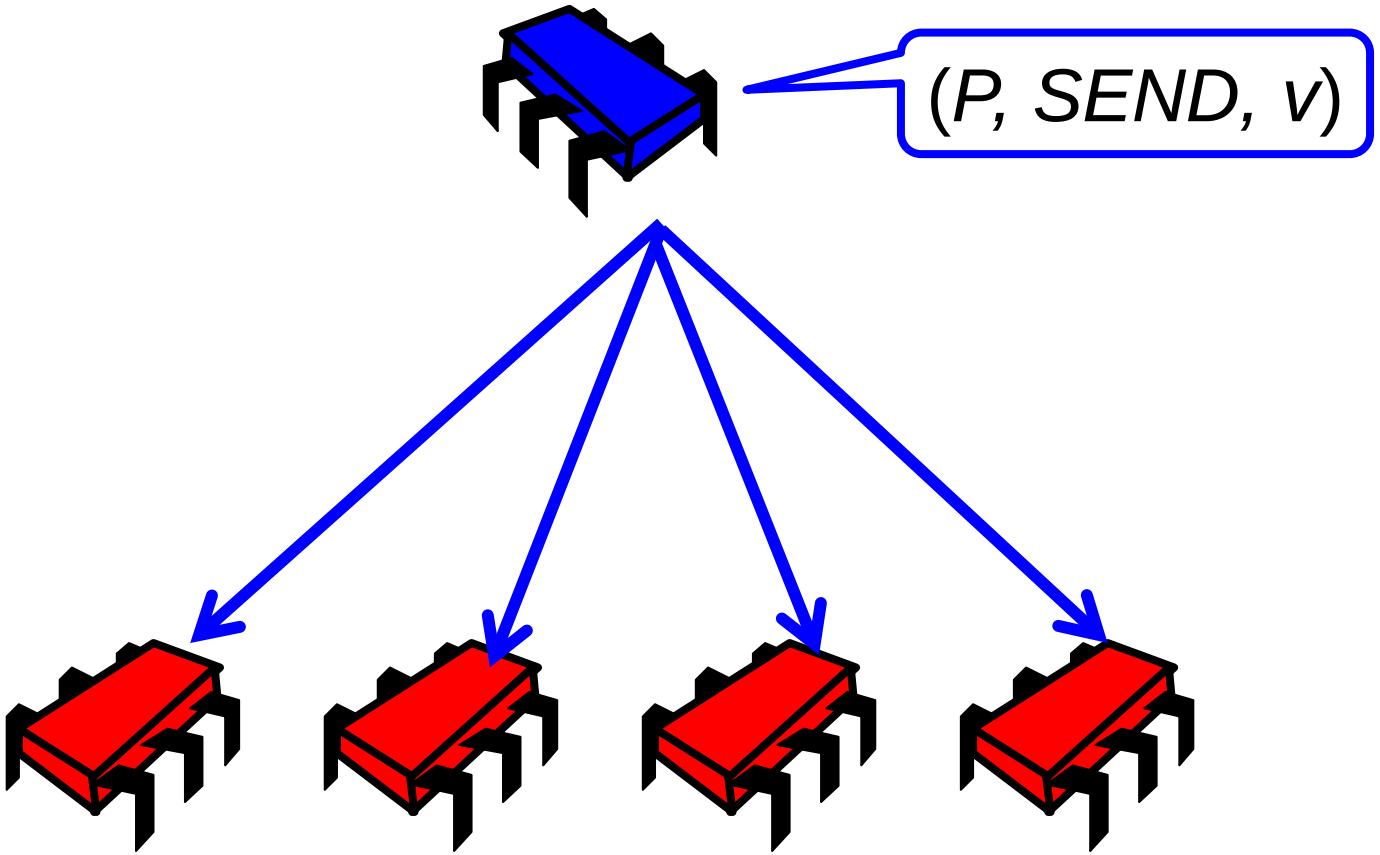
The only way ….

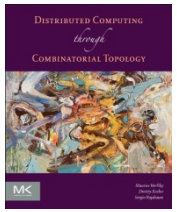
a Byzantine process can misbehave ….

is by sending the *same* fake value to everyone

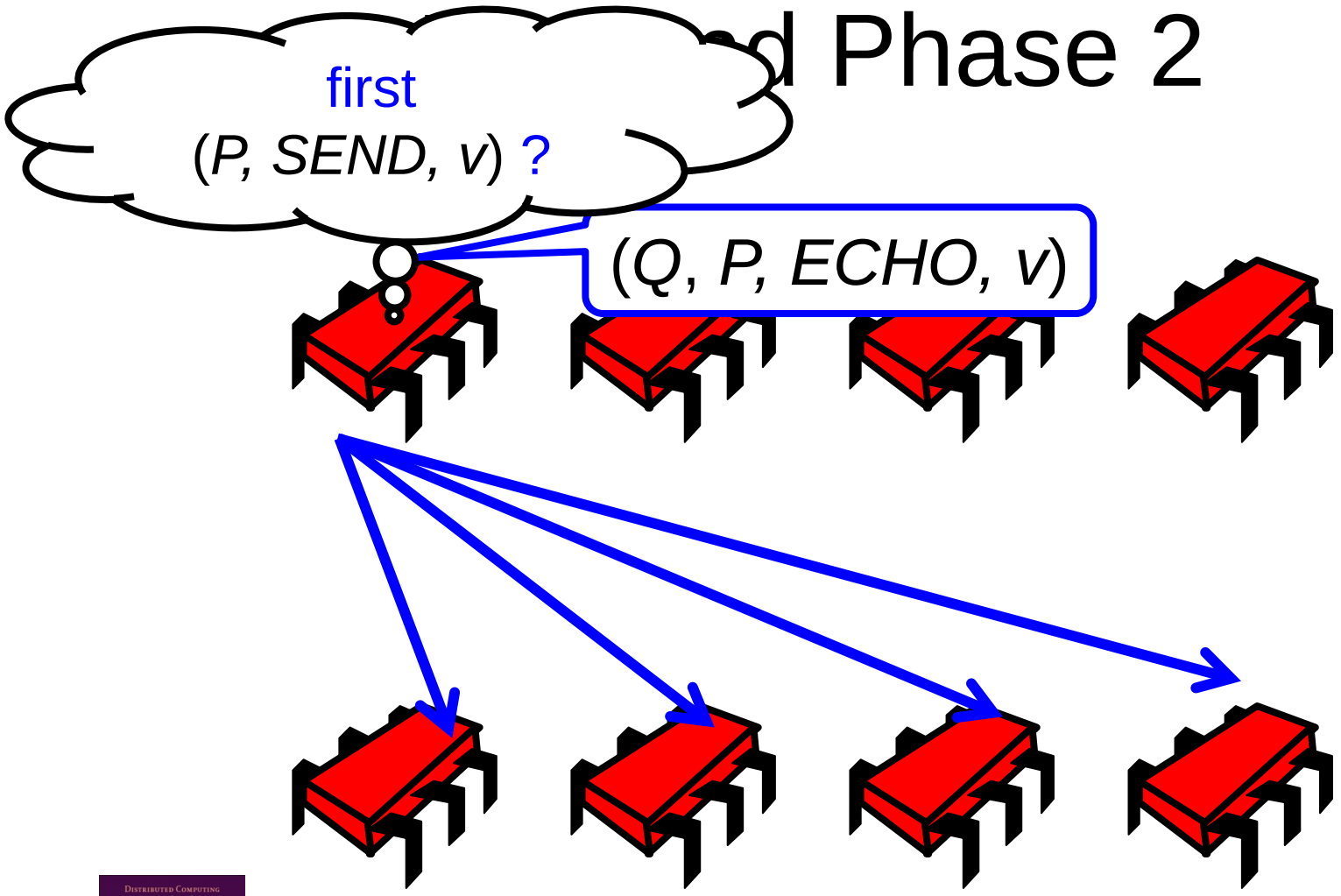


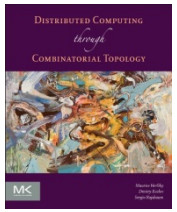
RBSend Phase 1



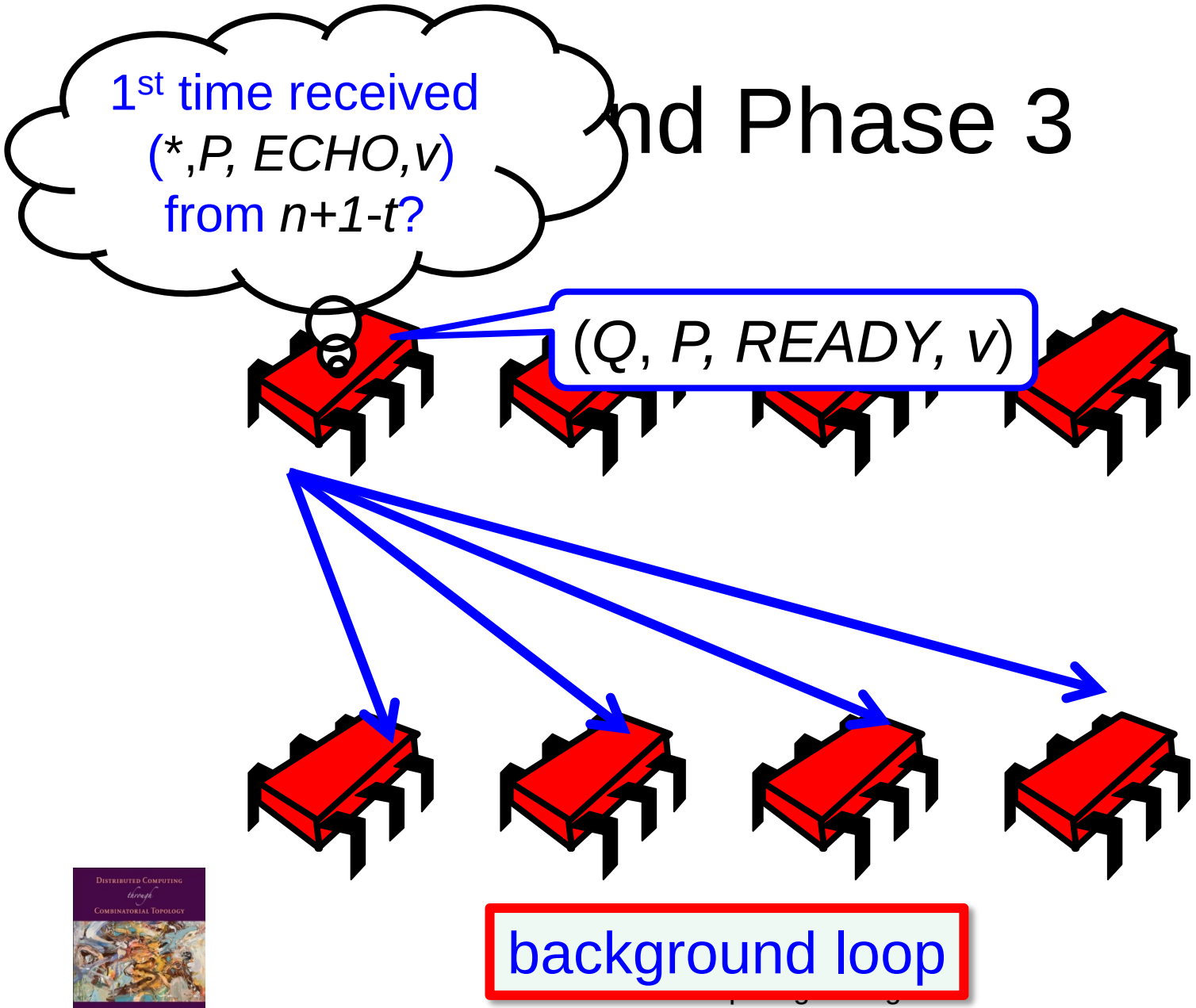


d Phase 2


first
(P, SEND, v) ?
(Q, P, ECHO, v)
background loop

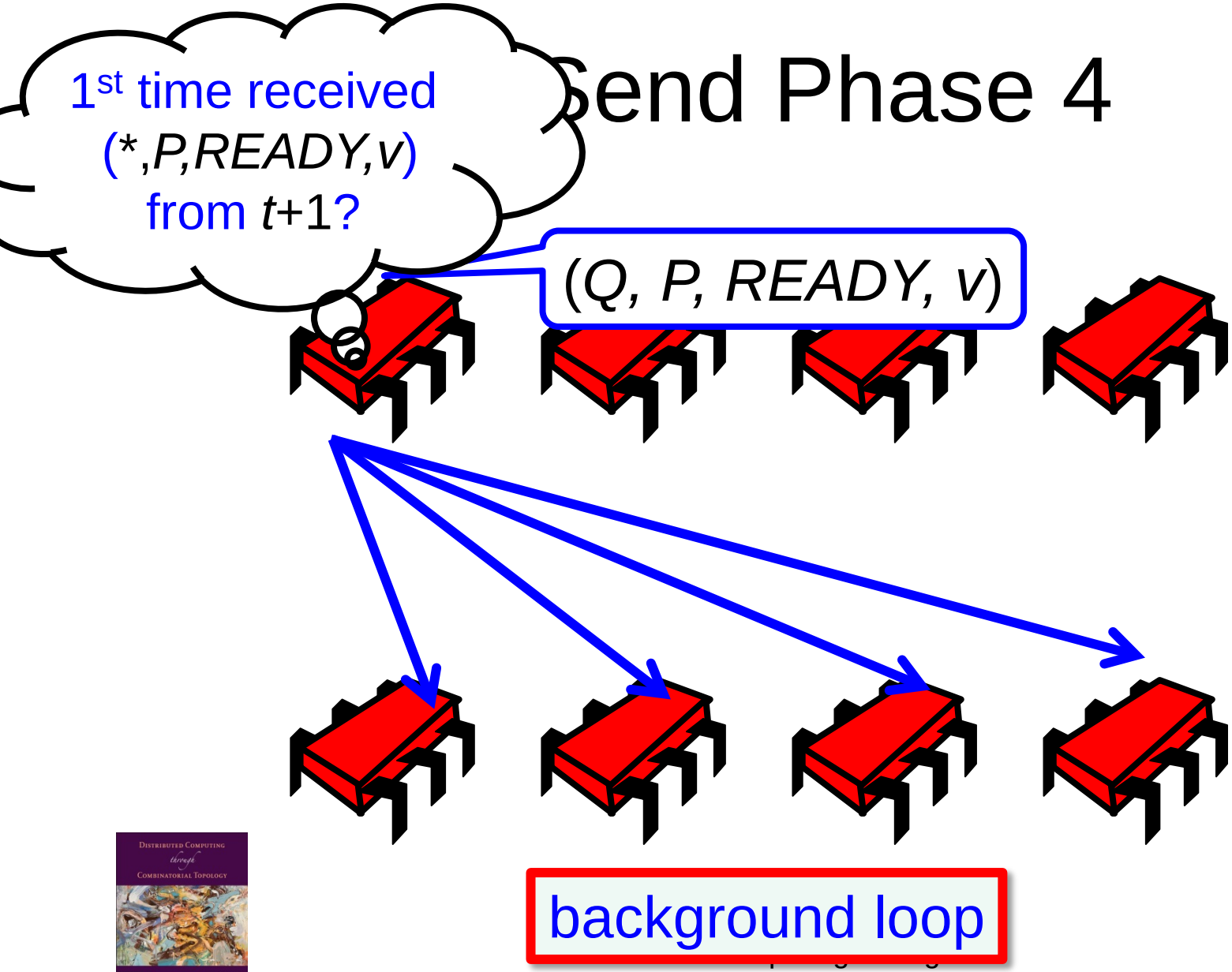

Distributed Computing
through
Combinatorial Topology

nd Phase 3


1st time received
(*,P, ECHO,v)
from n+1-t?
(Q, P, READY, v)
background loop

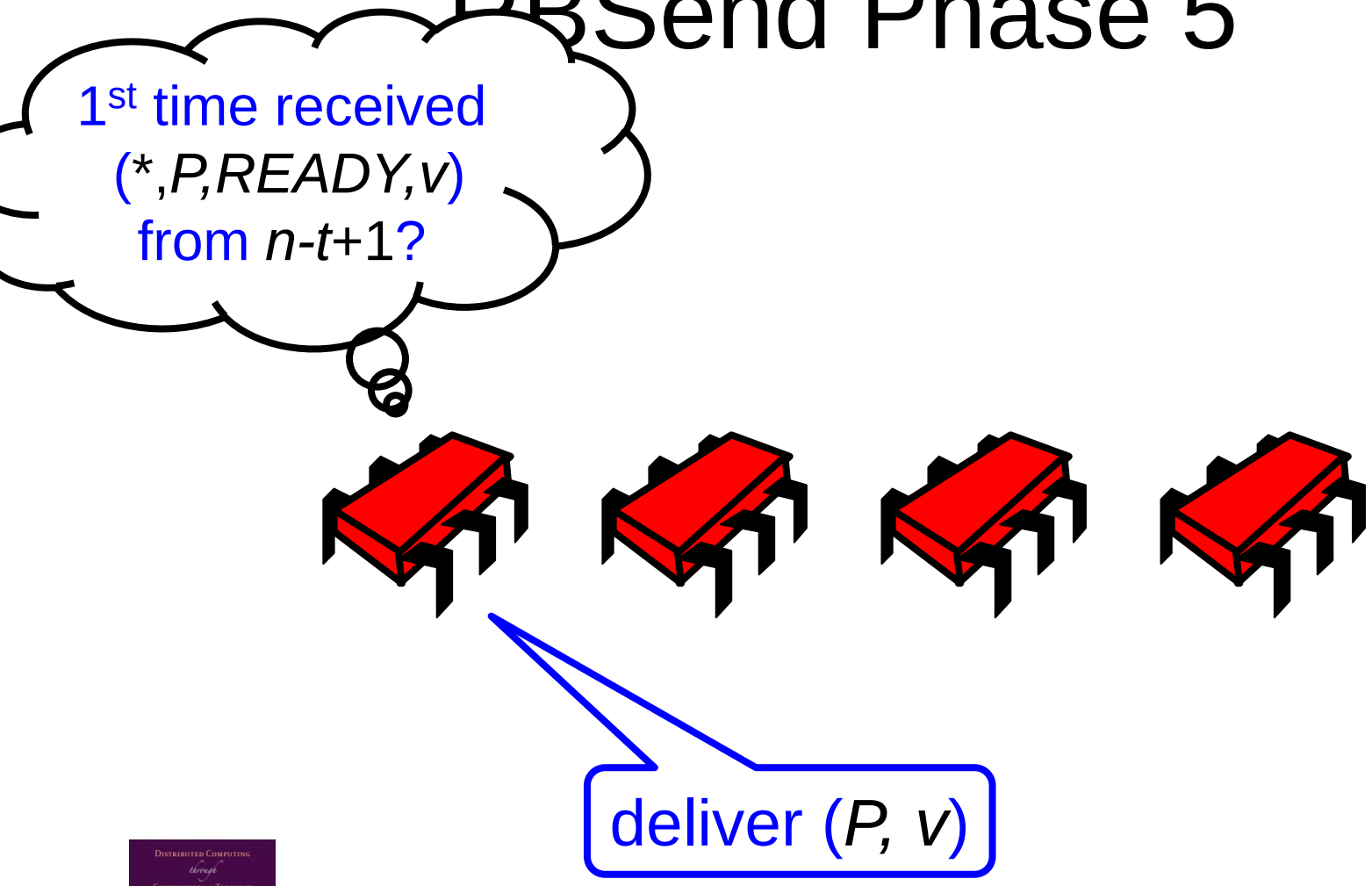
Distributed Computing
through
Combinatorial Topology

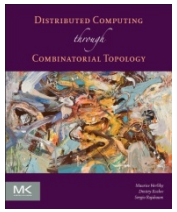
Send Phase 4


1st time received
(*,P,READY,v)
from t+1?
(Q, P, READY, v)
background loop

Distributed Computing
through
Combinatorial Topology

RBSend Phase 5





Lemma: Non-Faulty Integrity

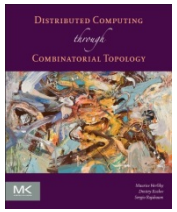
Suppose non-faulty *P* reliably receives (*Q*,*v*) from non-faulty *Q*

P must have received *n*+1-*t* (*, *READY*, *Q*, *v*) messages

At least *n*+1-2*t* non-faulty processes sent (*,*ECHO*,*Q*,*v*)

Let *R* be the first

R must have received (*Q*,*SEND*,*v*) from *Q*



Lemma: Non-Faulty Liveness

Non-faulty *P* broadcasts (*P*,*SEND*,*v*)

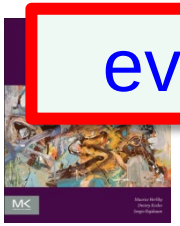
eventually received by *n*+1-*t* non-faulty processes

each sends (*,*ECHO*,*P*,*v*)

eventually receives *n*+1-*t* (*,*ECHO*,*P*,*v*)

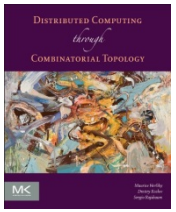
each sends (*,*READY*,*P*,*v*)

eventually receives *n*+1-*t* (*,*READY*,*P*,*v*)



Lemma: Global Uniqueness

The uniqueness tests ensure that any process that broadcasts (*,*ECHO*,*P*,*v*) or (*,*READY*,*P*,*v*) will not broadcast (*,*ECHO*,*P*,*v*') or (*,*READY*,*P*,*v*) where v ≠ v'.



Lemma: Global Liveness

Non-faulty Q reliably receives (P, v) …

R another non-faulty process

Q received $\geq n+1-t$ $(*,READY,P,v)$

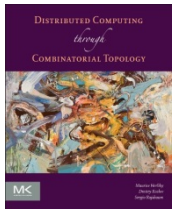
$\geq n+1-2t$ from non-faulty

if $\geq t+1$ non-faulty send $(*,READY,P,v)$

every non-faulty will receive & rebroadcast

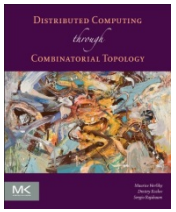
every non-faulty eventually receives $n+1-t$

and delivers



Reading a Value

```
getQuorum(Tag: tag): Set of Message
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

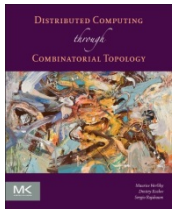


Reading a Value

```
getQuorum(Tag: tag):
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

Set of Message

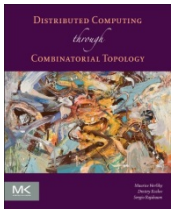
returns set of messages (values)



Reading a Value

```
getQuorum(Tag: tag): Set of Message
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

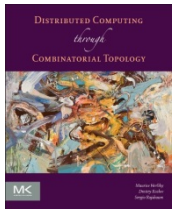
wait to hear from all but *t*



Reading a Value

```
getQuorum(Tag: tag): Set of Message
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

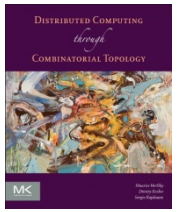
values reported by t+1



Reading a Value

```
getQuorum(Tag: tag): Set of Message
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

wait until you learn a trusted value

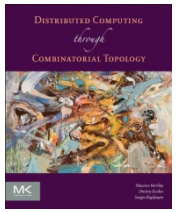


Reading a Value

```
getQuorum(Tag: tag): Set of Message
  M := ∅
  while |M| < n+1-t or trusted(M) = ∅ do
    upon RBReceive(Q,tag,v) do
      M := M ∪ {(Q,tag,v)}
  return M
```

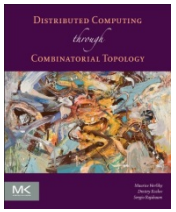
safe to wait for both (why?)

returns $\geq$ 1 trusted values



Set Agreement

```
SetAgree(v): value
   RBSend(P,INPUT,v)
   M: Set of Message := getQuorum(INPUT)
   return min Trusted(M)
```



Set Agreement

```
SetAgree(v): value
   RBSend(P,INPUT,v)
   M: Set of Message := getQuorum(INPUT)
   return min Trusted(M)
```

at most *t* lower values missing

Barycenrtric Agreement

$\mathbf{R_i[j]}$ = messages received by P_i from P_j

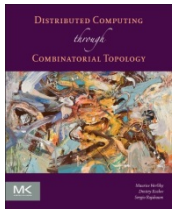
$\mathbf{R_i := (\emptyset, \ldots, \emptyset)}$

$\mathbf{M_i := \emptyset}$

$\mathbf{B_i := \emptyset}$

$\mathbf{M_i}$ = messages received by P_i

$\mathbf{B_i}$ = P_i 's *buddies*: processes that reported same set of vertices



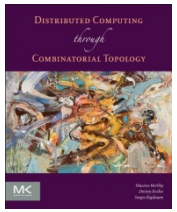
Barycentric Agreement (1)

reliably broadcast my input vertex

RBSend(P_i,INPUT,v)

M_i := **getQuorum(INPUT)**

reliably receive others' input vertices



Barycentric Agreement (2)

record newly-learned input vertices

```
background // run forever
  upon RBReceive(Pj,INPUT,u) do
    Mi := Mi ∪ {u}
    RBSend(Pi,REPORT,Mi)
```

forward set of vertices with REPORT tag

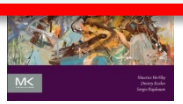
Baryc

until I have *n+1-t* buddies…

```
while |B_i| < n+1-t do
  upon RBReceive(Pj,REPORT,M_j) do
    R_i[j] := M_j
    B_i := {P: R_i[l] = M_i, 0 ≤ ℓ ≤ n}
return trusted(M_i)
```

accumulate new reports …

return set of trusted input vertices.

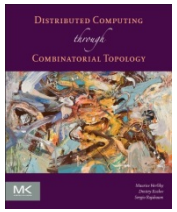


Lemma

Sequence of $\mathtt{M_i}$ vertex sets reliably broadcast by P_i are monotonically increasing as sent & as received

When sent, by construction …

when received, because FIFO message channels



Lemma: Protocol Terminates

Suppose BWOC P_i runs forever …

Eventually M_i assumes final value M …

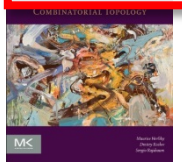
Non-faulty P_j, with M_j, receives M

P_j must have sent M_j to P_i

$M_j \subset M$ OR $M_j = M$

P_i will sent M to P_j

P_j has sent M to P_i



All M_i, M_j Totally Ordered

If P_i broadcasts $M^{(0)}, \ldots, M^{(k)}$, then $M^{(i)} \subset M^{(i+1)}$

To decide …

P_i received M_i from X, $|X| \geq n+1-t$

P_j received M_j from Y, $|Y| \geq n+1-t$

some $P_k \in X \cap Y$ sent both M_i, M_j

so M_i, M_j are ordered.

Lemma

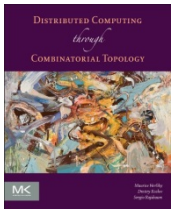
Non-faulty P_i and P_j have

$|M_i \cap M_j| \geq n+1-t$

$\text{trusted}(M_i \cap M_j) \neq \emptyset$

$M_i \subseteq M_j$ or $M_j \subseteq M_i$

proof in book



Byzantine Computability

if $n+1 > (\dim \mathcal{I} + 2)\, t$

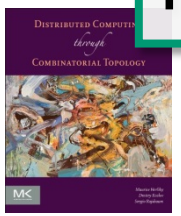
$(\mathcal{I},\mathcal{O},\Delta)$ has a wait-free Byzantine protocol …

if and only if …

there is a continuous map

f: $|\text{skel}^t \mathcal{I}| \rightarrow |\mathcal{O}|$

carried by Δ.



Map Implies Protocol

f: $|\text{skel}^t \mathcal{I}| \to |\mathcal{O}|$

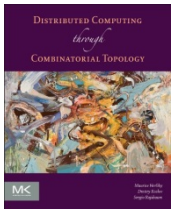
ϕ: $\text{Bary}^N \text{skel}^t \mathcal{I} \to \mathcal{O}$

carried by Δ.

Solve using …

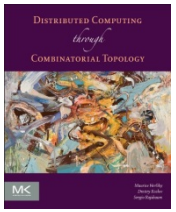
barycentric agreement

t-set agreement



Protocol implies Map

reduction to crash-failure model

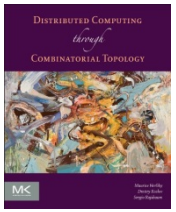


Lower Bound

A strict colorless task $(\mathcal{I},\mathcal{O},\Delta)$
is *trivial* if there is a simplicial map

$\phi: \mathcal{I} \to \mathcal{O}$ carried by Δ.

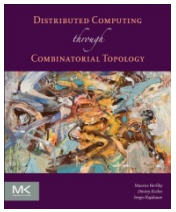
(A trivial task can be solved without communication)



Theorem

If a strict colorless task $(\mathcal{I},\mathcal{O},\Delta)$
has a protocol, where $n+1 \leq (\dim \mathcal{I} + 2)\, t$
then that task is trivial

(A trivial task can be solved without communication)



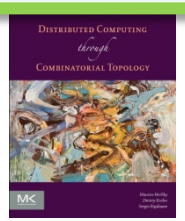
Proof

Let $\sigma = \{v_0, \ldots, v_d\}$ be a d-simplex of $\mathcal{I}$

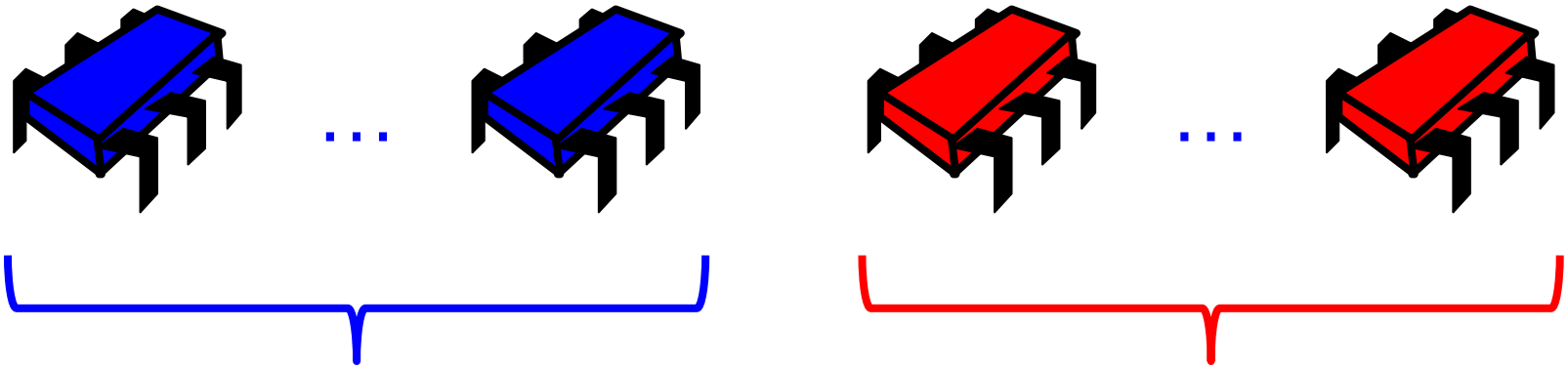
consider an execution where …

P_i has input $v_{i \bmod d+1}$

no failures



Proof



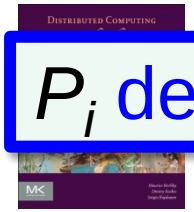
$S = \{P_0, \ldots, P_{n-t}\}$

$T = \{P_{n+1-t}, \ldots, P_n\}$

$S_i = \{P |\ P \text{ in } S \text{ has input } v_i\}$

no communication

P_i decides u_i



Proof

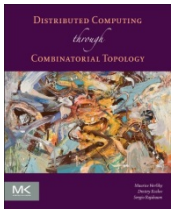
If $u_i \in \Delta(\sigma_i)$ and …

$u_i \in \Delta(\sigma_i')$ then …

$u_i \in \Delta(\sigma_i \cap \sigma_i')$ so …

there is a unique minimal σ_i such that

$u_i \in \Delta(\sigma_i)$

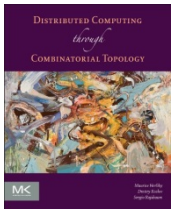


Proof

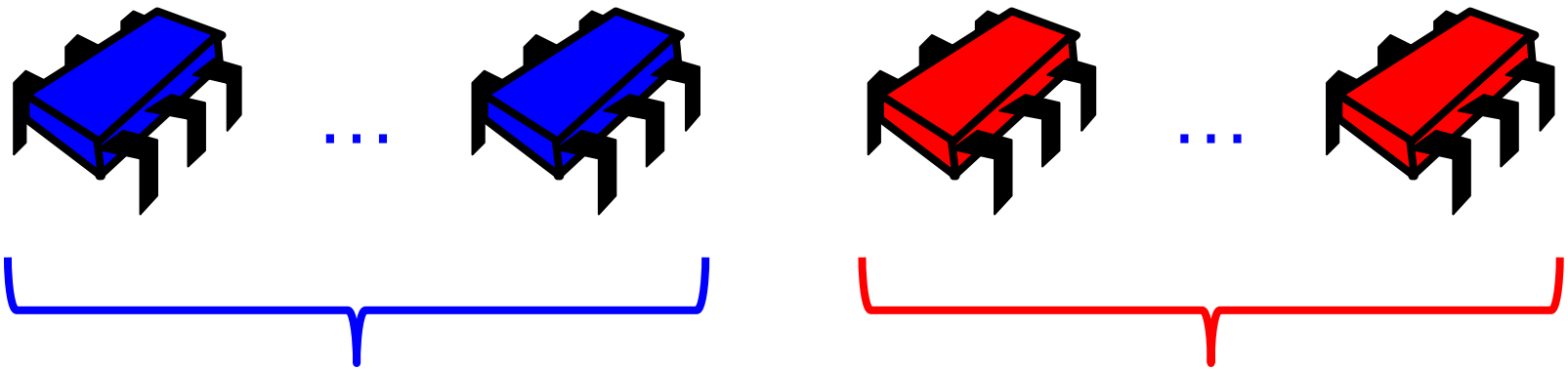
If all $\sigma_i = \{v_i\}$ then the task is trivial.

so some minimal $\sigma_i = \{v_i, v_j\}$

Every simplex τ such that $u_i \in \Delta(\tau)$ contains v_j



Proof



S_j change input to v_i

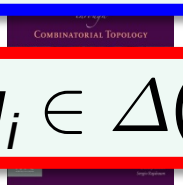
T have input v_i

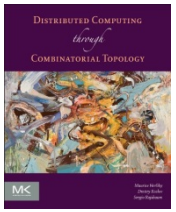
S_j are Byzantine, still claim v_j

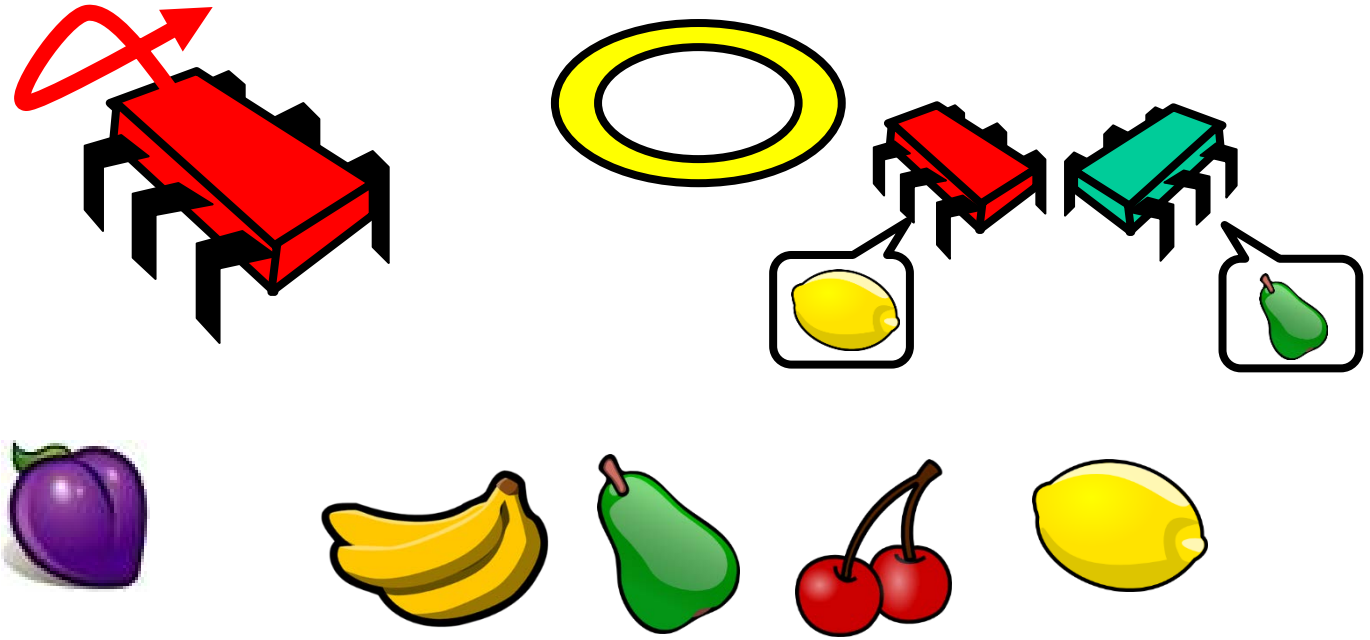
no communication

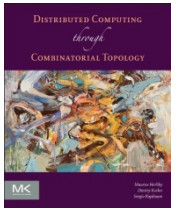
P_i still decides u_i

$u_i \in \Delta(\sigma_i \setminus \{v_j\})$ contradiction








Distributed Computing
through
Combinatorial Topology