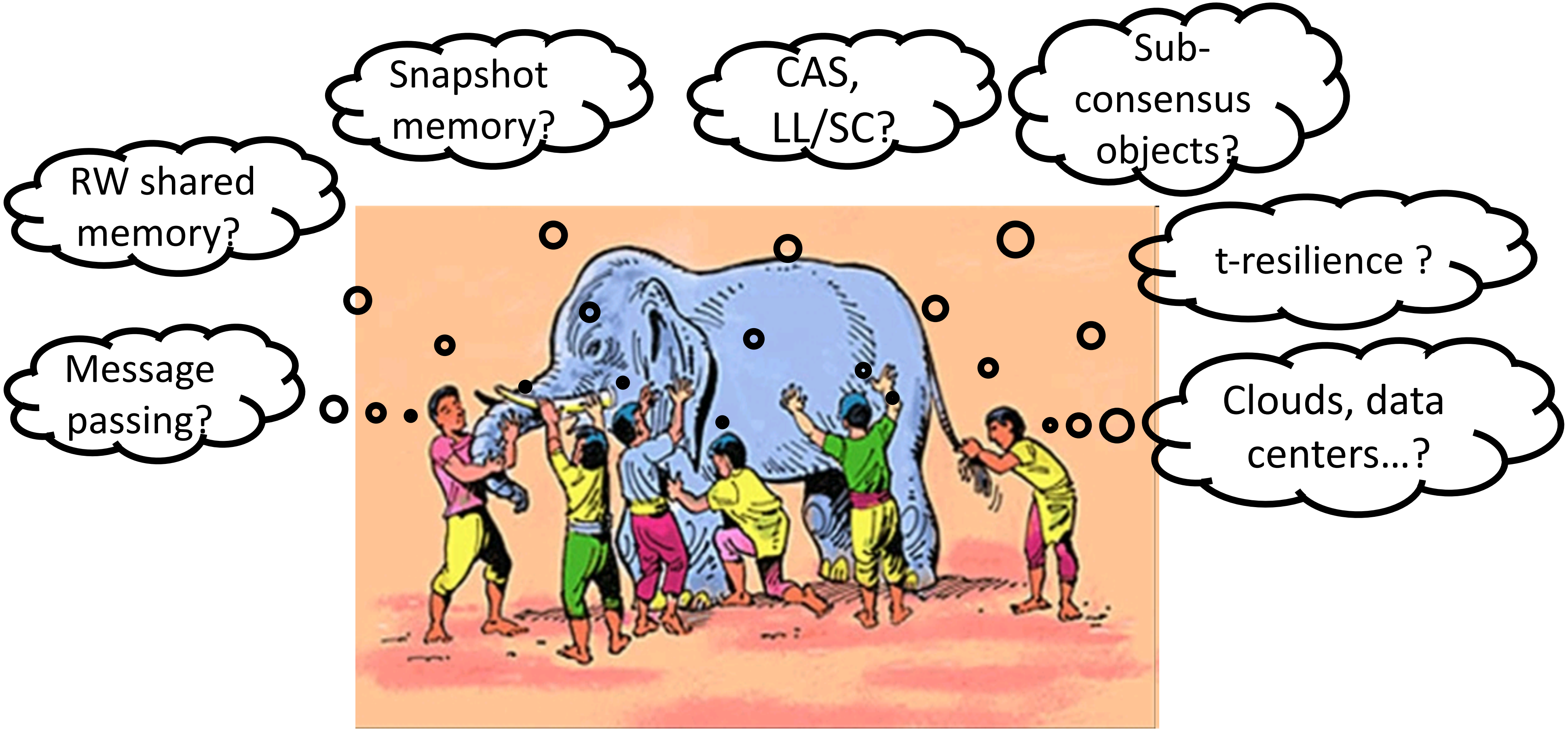
WF=NWF?
On Models which are not Fundamentally Different

Petr Kuznetsov

TU Berlin/DT-Labs

(Joint work with Eli Gafni, UCLA)

Distributed modeling jumble



Similarities and reductions

- Safe bits ≅ atomic read-write registers [Lam85]
- Atomic read-write ≅ atomic snapshots [Afek et al, 93]
- Message-passing ≅ Shared-memory [ABD95]
- Atomic read-write ≅ Immediate snapshots [BG93]
- Atomic read-write ≅ Iterated Immediate Snapshots (NB) [BG93]

- t-resilience ≅ wait-freedom [BG93,Gafni09]

Model equivalence

Models M and M’ are fundamentally equivalent if for every *task* T there exists a task T’(T,M’)

T is solvable in M $\Longleftrightarrow$ T’(T,M) is solvable in M’

(Solvability in M can be reduced to solvability in M’)

Distributed tasks (I,O,Δ)

- I – set of input vectors
- O – set of output vectors
- Task specification $\Delta: I \rightarrow 2^{O}$

k-set agreement

- Processes start with inputs in V ($|V|>k$)
- The set of outputs is a subset of inputs of size at most k
- k=1: consensus

Conjecture

- All (natural) models are fundamentally equivalent to the *wait-free* model (WF)

L-resilience: output if a set in L is live

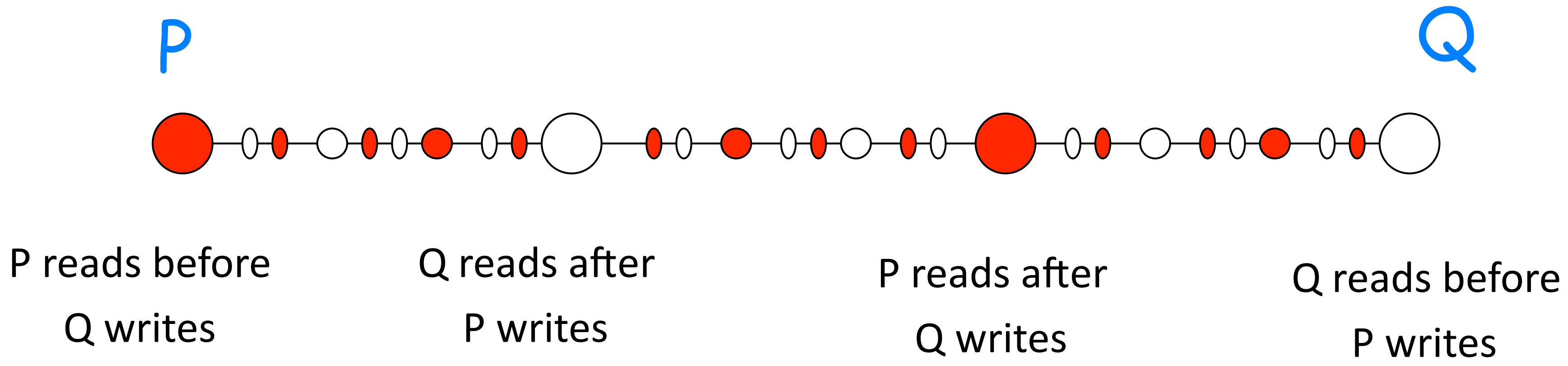
K-concurrency: output if at most k processes concur

The wait-free model: 2 processes

while not done

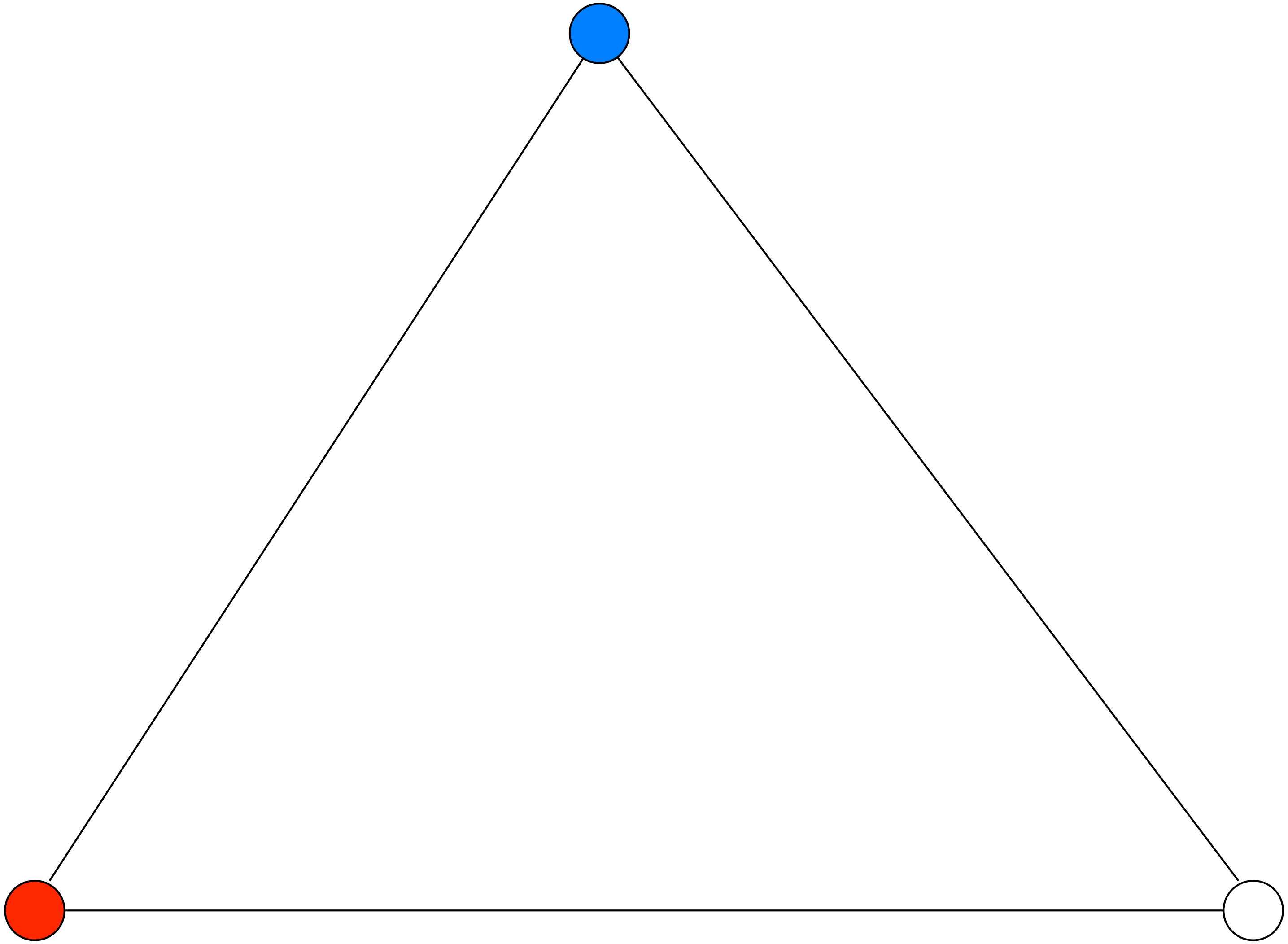
    write(view)

    view := collect-memory()

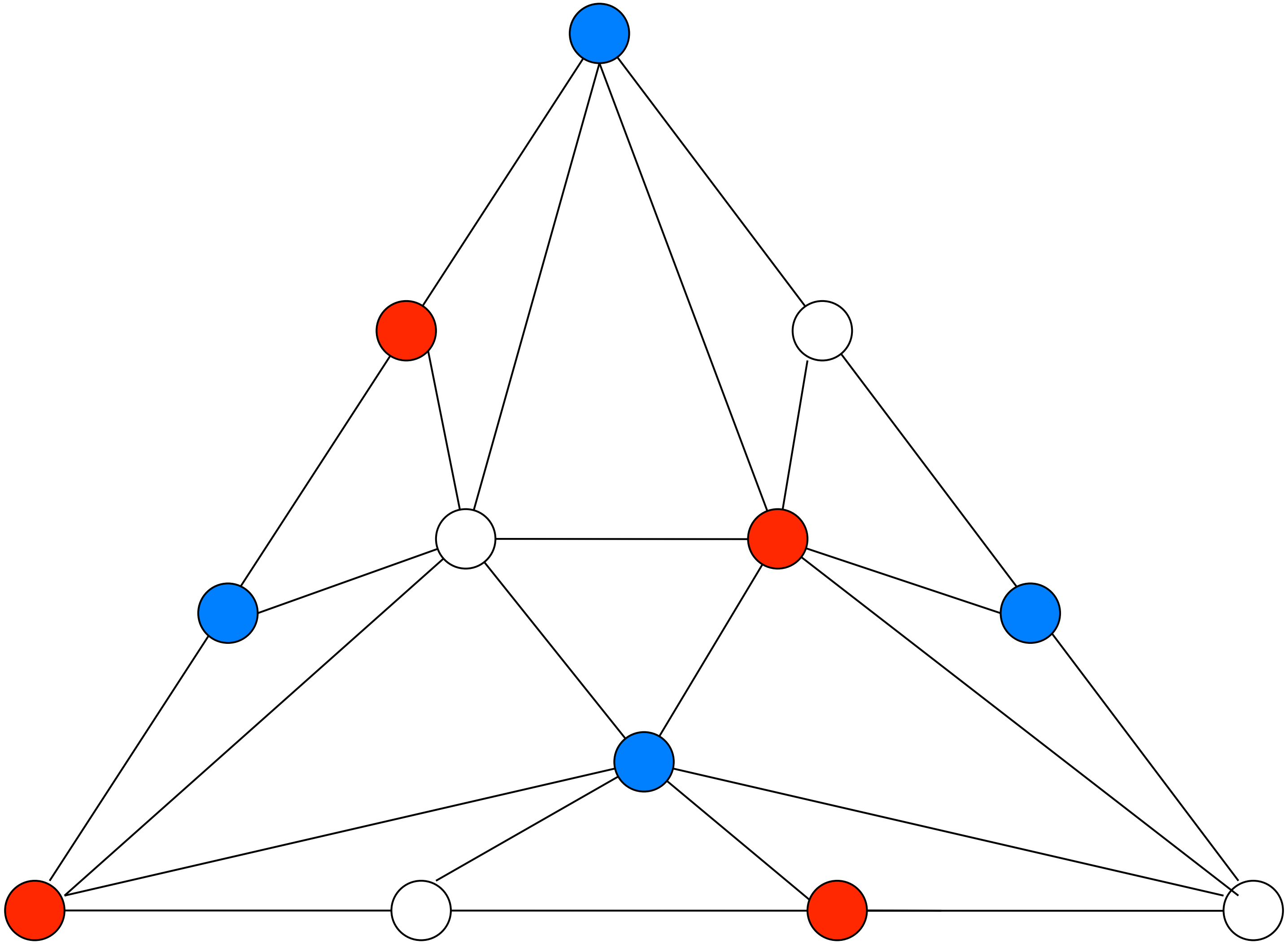


Wait-free consensus is impossible!

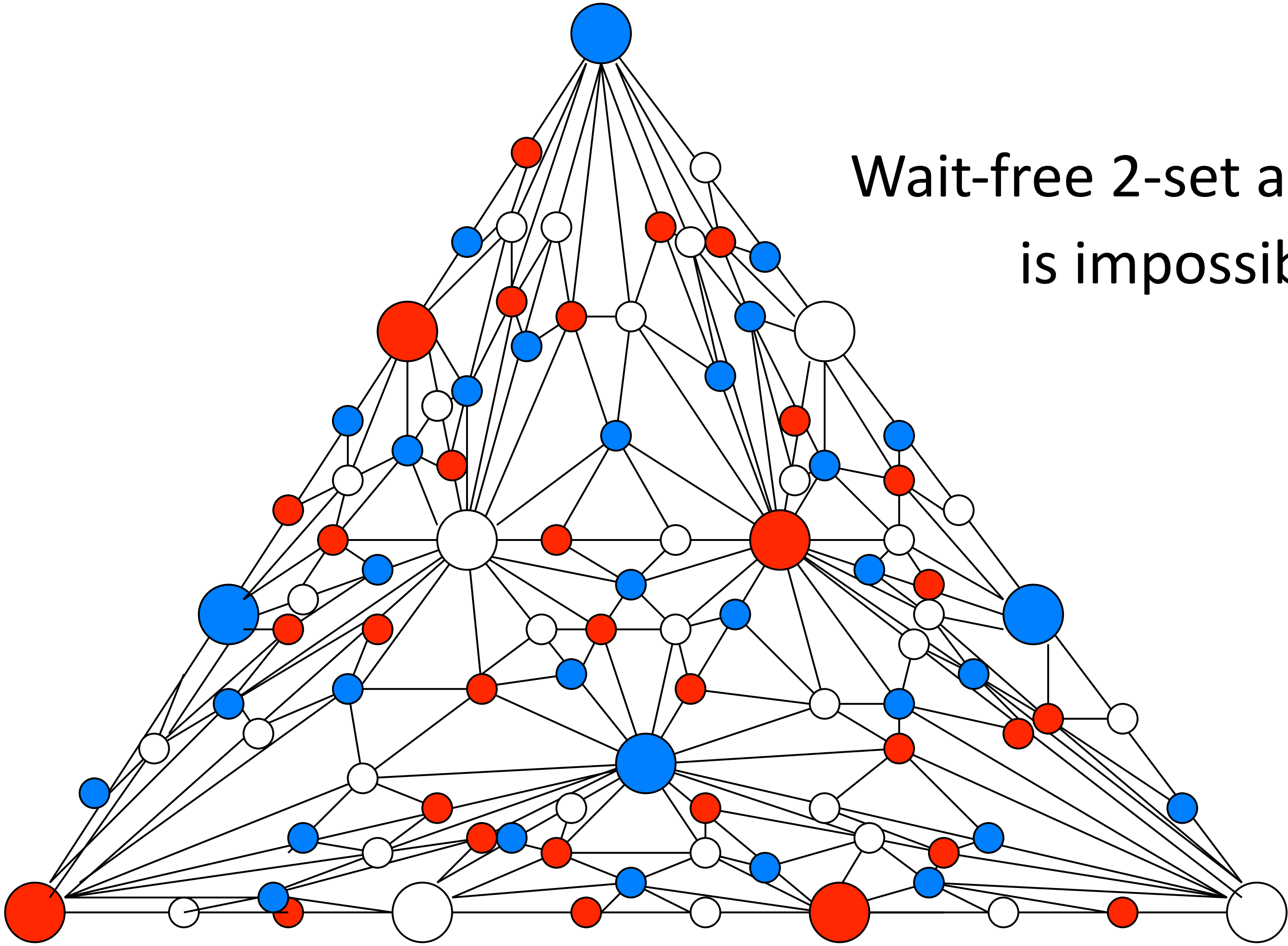
The wait-free model: 3 processes



The wait-free model: 3 processes



The wait-free model: 3 processes



Wait-free 2-set agreement is impossible!

Why wait-freedom?

- Simple structure: contains all possible interleavings
  - ✓WF computing: a process makes progress, regardless of others

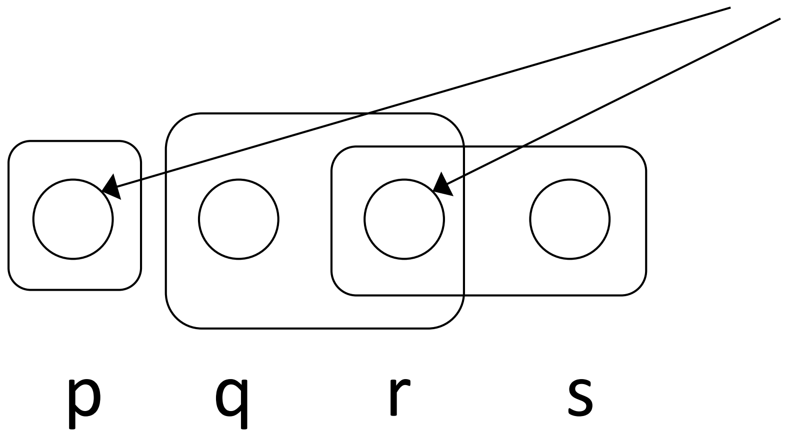
- WF solvability has a precise topological characterization [Herlihy-Shavit,99]
  - ✓A continuous map from a subdivision to the outputs
  - ✓Undecidable for >2 processes [HR97,GK99]

L-resilience

L is a set of process subsets

L={p,qr,rs}

Hitting set of L



The power of L is characterized by its hitting set size hs(L)!

L-resilience: defining T'(T,L)

- A process in T'(T,L) is a tuple (i,S)
  - ✓ i = 1,…,hs(L)
  - ✓ S in L
- (I,S) outputs a value for each process in S: an output of T or "?"
  - ✓ All outputs are consistent with T
- If (i,S) decides, then
  - ✓ there is (j,S') such that S is subset of S'
  - ✓ or hs(L')≤i-1, L' – the set of "undecided" sets in L

Relating T and T'(T,L): simulating many by few

- hs(L) processes in T'(T,L) simulate an L-resilient execution:
  - ✓(1,S),…,(hs(L),S)

- If (eventually) the number of simulators is j and the number of simulated processes is m, then at least m-j+1 simulated processes make progress [Gaf09]

Simulating L-resilience

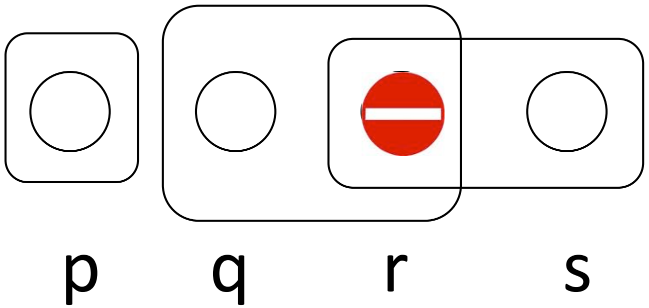
L={p,qr,rs}

✓ hs(L)=2

✓at most two simulators, (1,S) and (2,S)

✓one faulty simulator cannot block all sets
in L: at least one set in L is live

{q,r} and {r,s} cannot be live



but {p} can!

K-concurrency

- Output if at most k processes run concurrently
  - ✓ Equivalent to WF with k-set agreement objects
  - ✓ k=1: consensus, every task is solvable

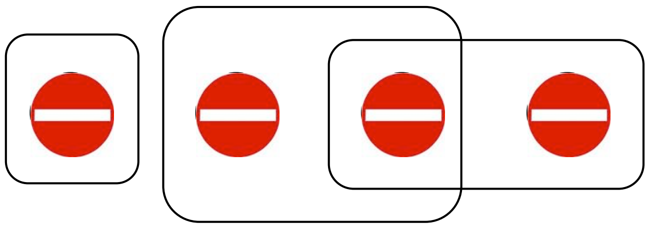
- Relating WF and k-concurrency:
  - ✓ Simulate few by many
  - ✓ k-state machines [Guerraoui, Gafni '10]

Filling the gap

- L-resilience ≅ WF
- K-concurrency ≅ WF

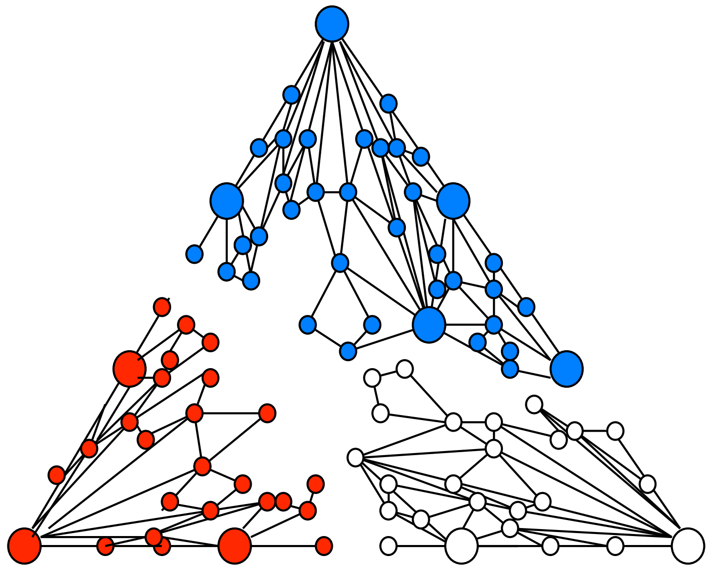
What about generic *adversaries* [Delporte et al., 2009]?

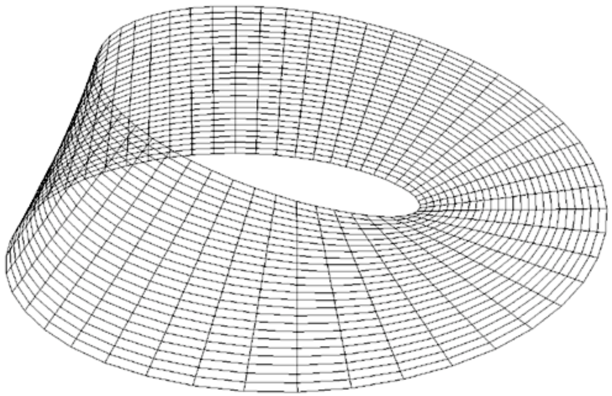
A= {p,qr,rs}



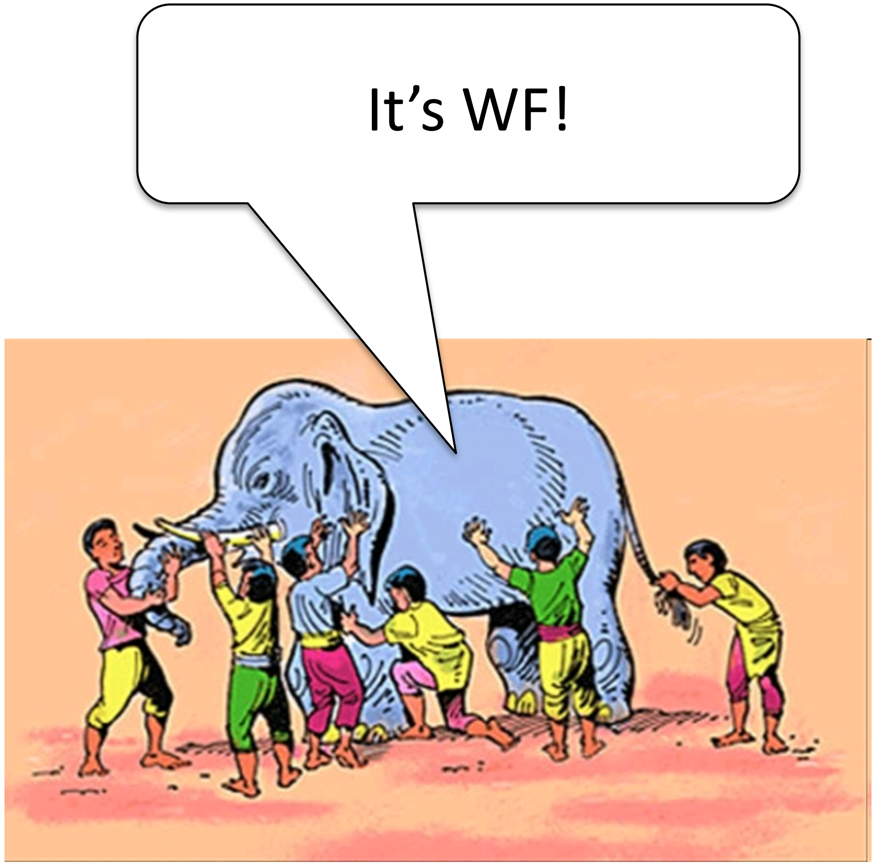
On natural models

- Natural: restricted wait-free
  - ✓Adversaries
  - ✓Deterministic objects





- “Unnatural”
  - ✓“Sub-agreement” objects


It’s WF!

THANK YOU!