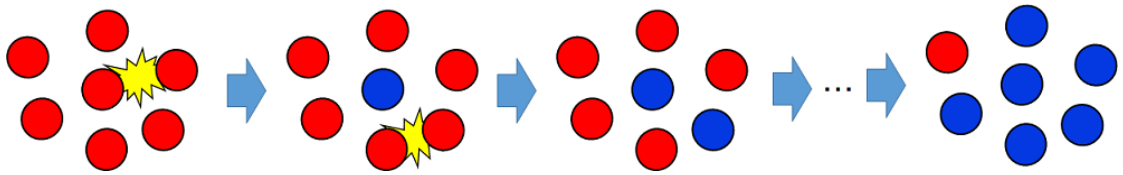


Populations Protocols

Some slides are taken from James Aspnes slides (2007), his keynote talk in PODC 2020, the presentation of “Black Ninjas in the Dark: Formal Analysis of Population Protocols” by Javier Esparza and the presentation of “Composable Computation in Discrete Chemical Reaction Networks” by E.E. Severson.

The Story

- Black Ninjas meet in a thunderous night at a garden in the dark (no moon)

- They must decide by majority to attack or not attack a castle (no attack if tie)

- How can they conduct the vote?

The *Amateur* (*A*) Sensei Thinks Beforehand about a Communication Protocol

- Ninjas wonder randomly, interacting when they bump onto each other
- Ninjas store their current estimation of the final outcome: attack or don't attack
- Ninjas are either active or passive
- Initially: all ninjas active, estimation = own vote

attack
active

don't attack
active

attack
passive

don't attack
passive

The Goal of *A*

Goal of voting protocol:

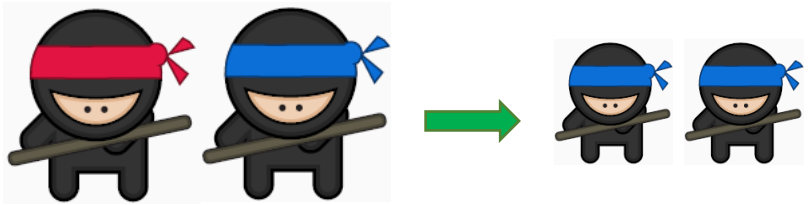
- eventually all ninjas reach the same estimation, and
- this estimation corresponds to the majority.

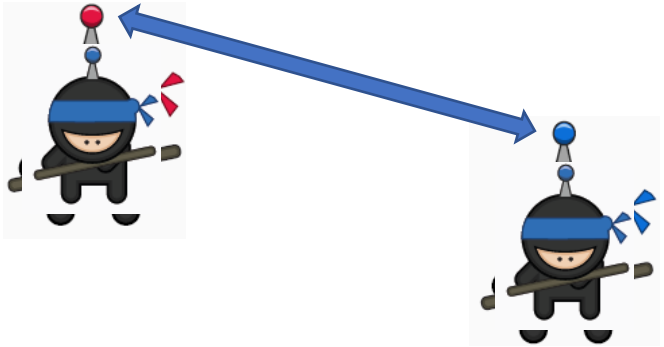
Graphically:

- Initially more red ninjas $\rightarrow$ eventually all ninjas red.
- Initially more blue ninjas or tie $\rightarrow$ eventually all ninjas blue.

The Protocol of *A*

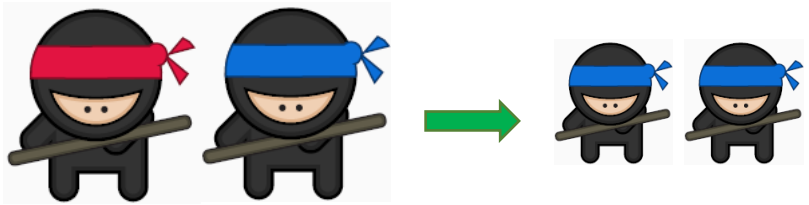
- Active ninjas of opposite colors become passive and blue



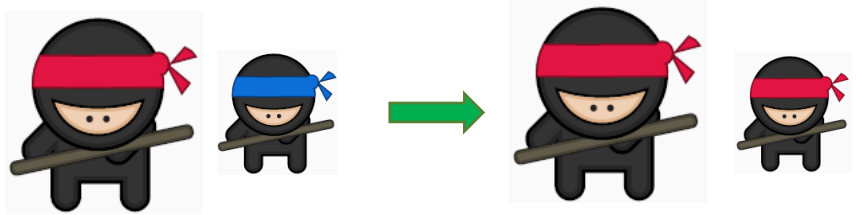


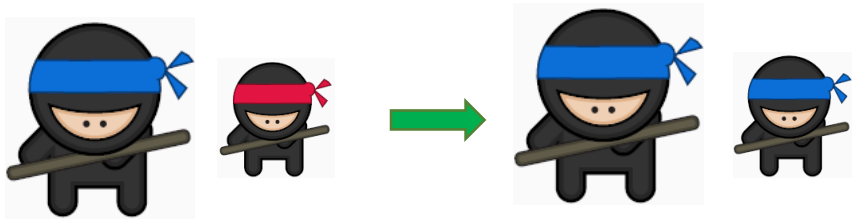
The Protocol of *A*

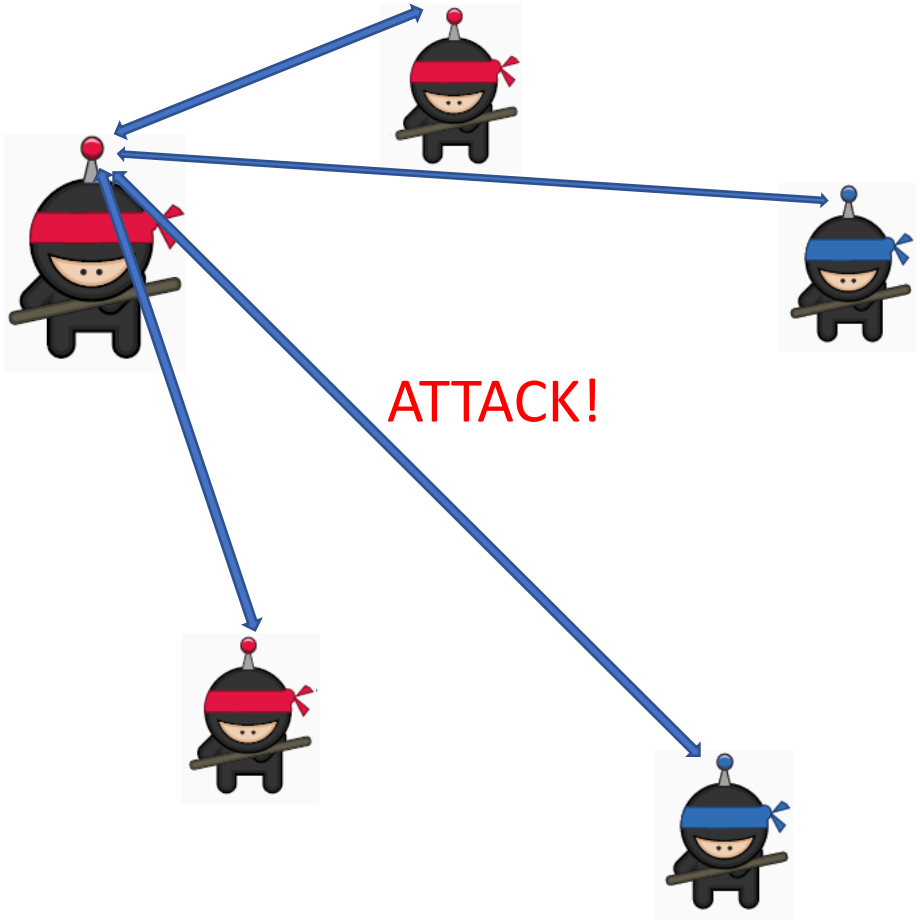
- Active ninjas of opposite colors become passive and blue



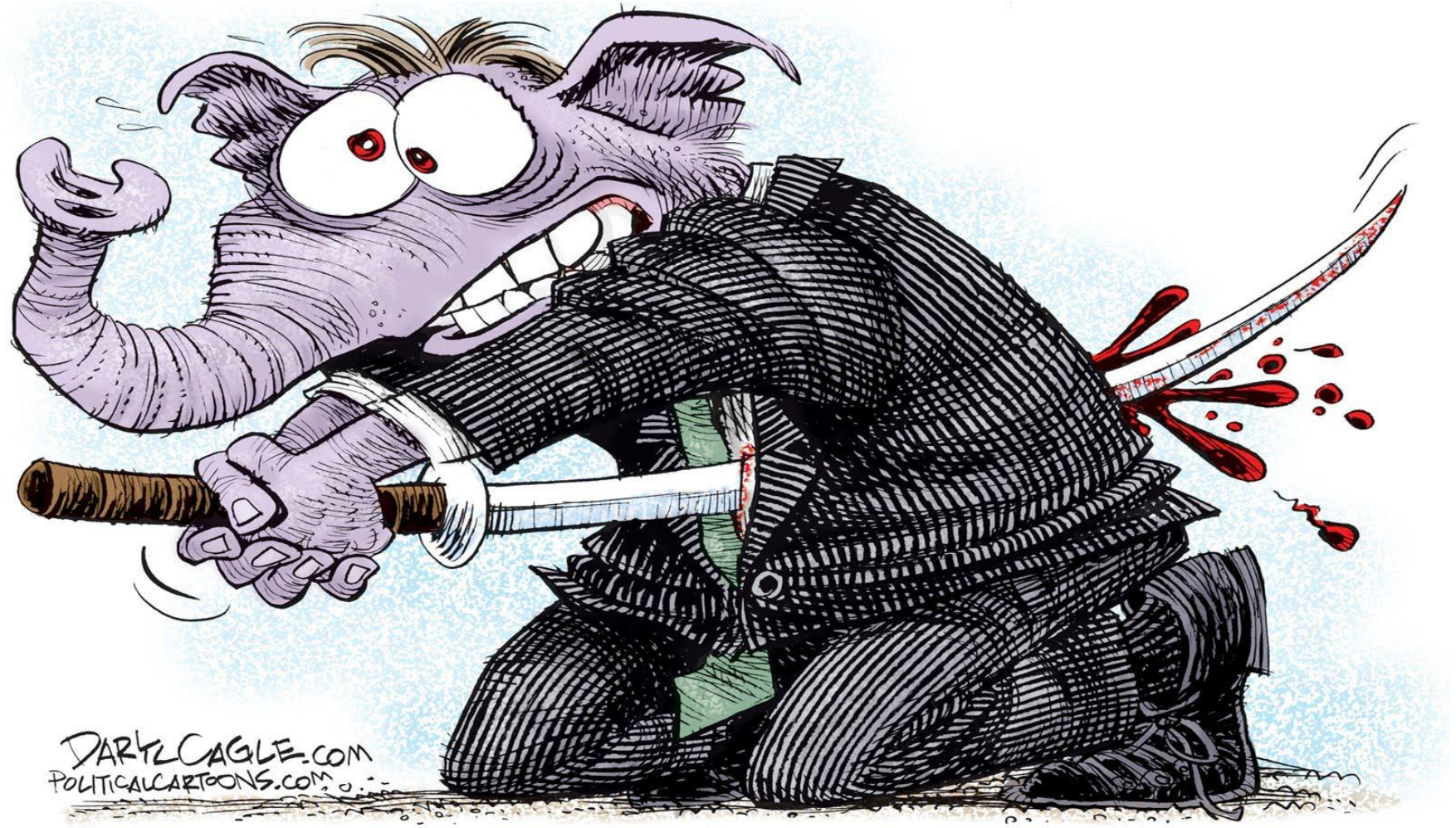
- Active ninjas convert passive ninjas to their color







One Night Disaster Strikes… (Animation)



The *Professional* (*P*) Sensei Takes Over

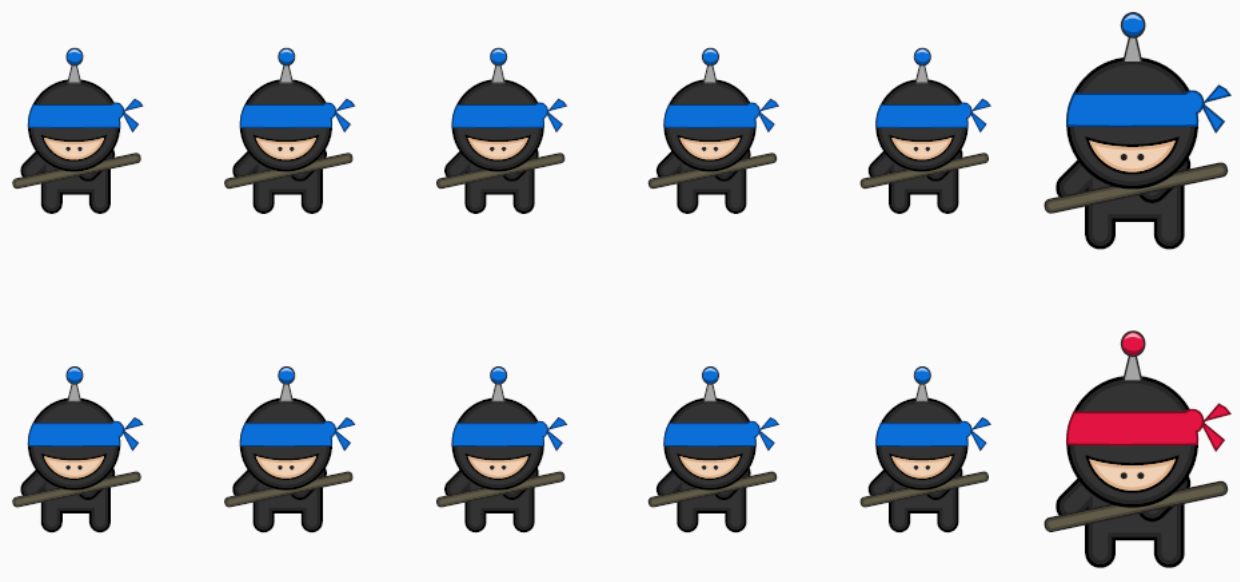
He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two

The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two

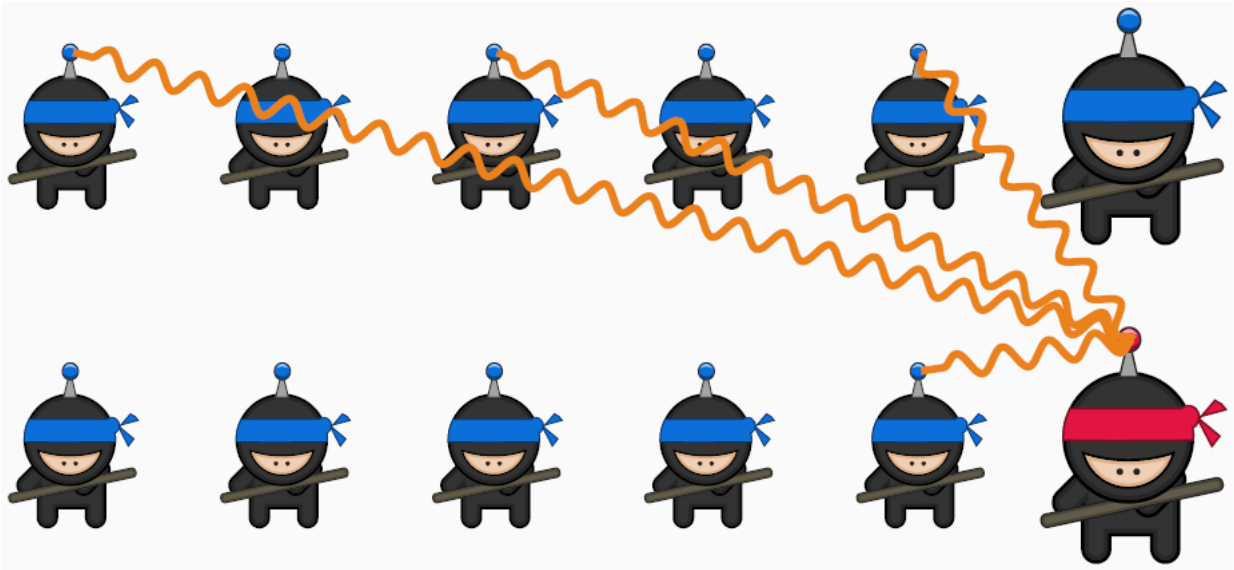
The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two



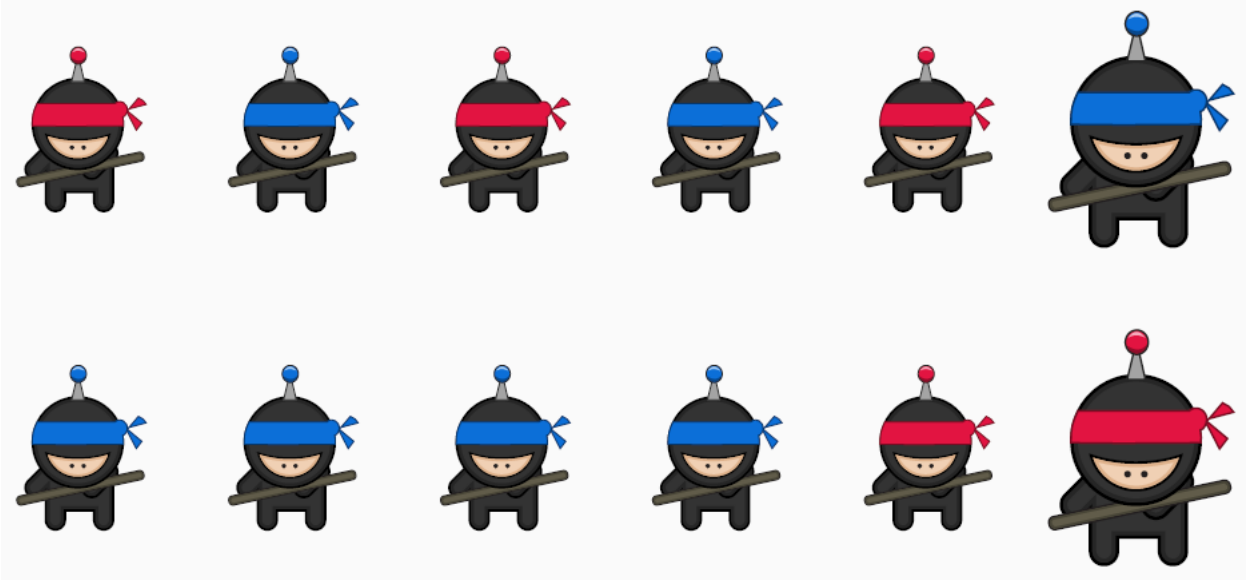
The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two



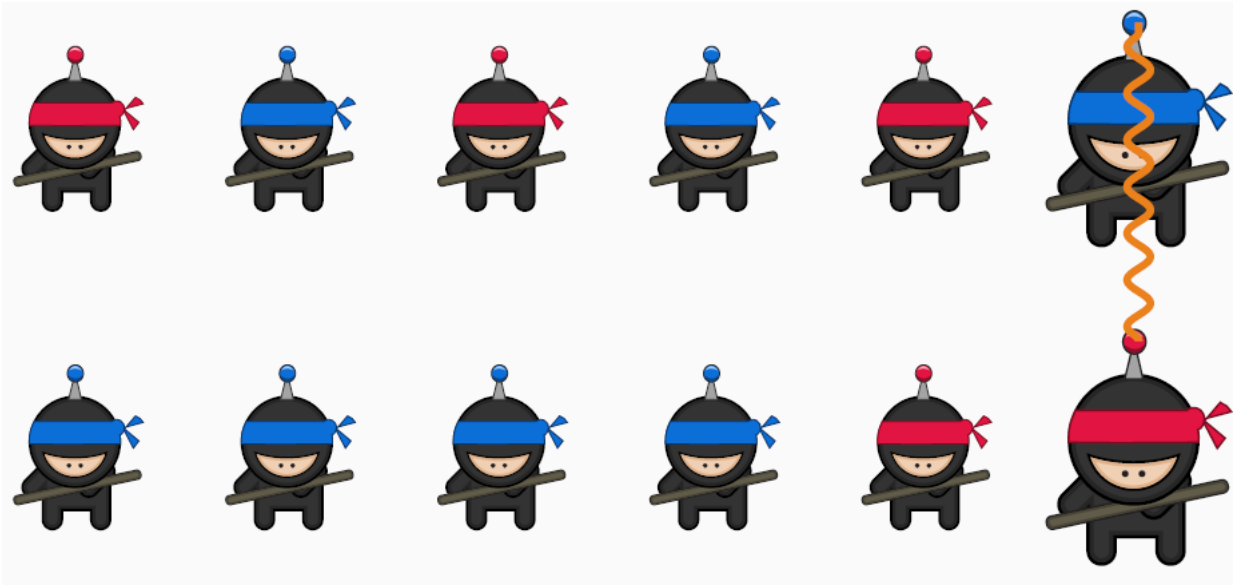
The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two



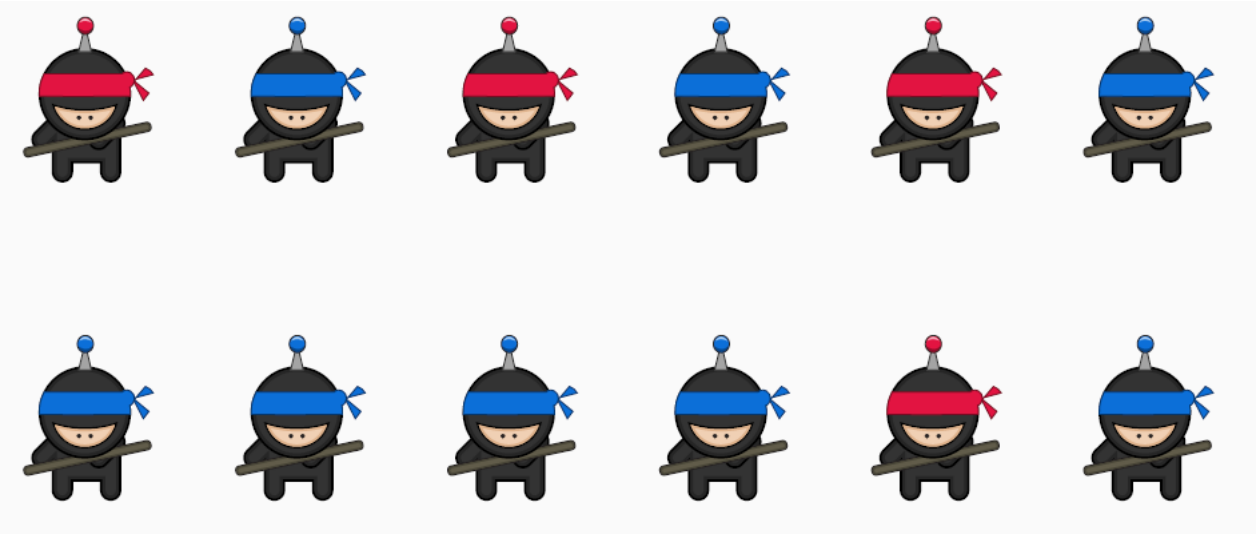
The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two

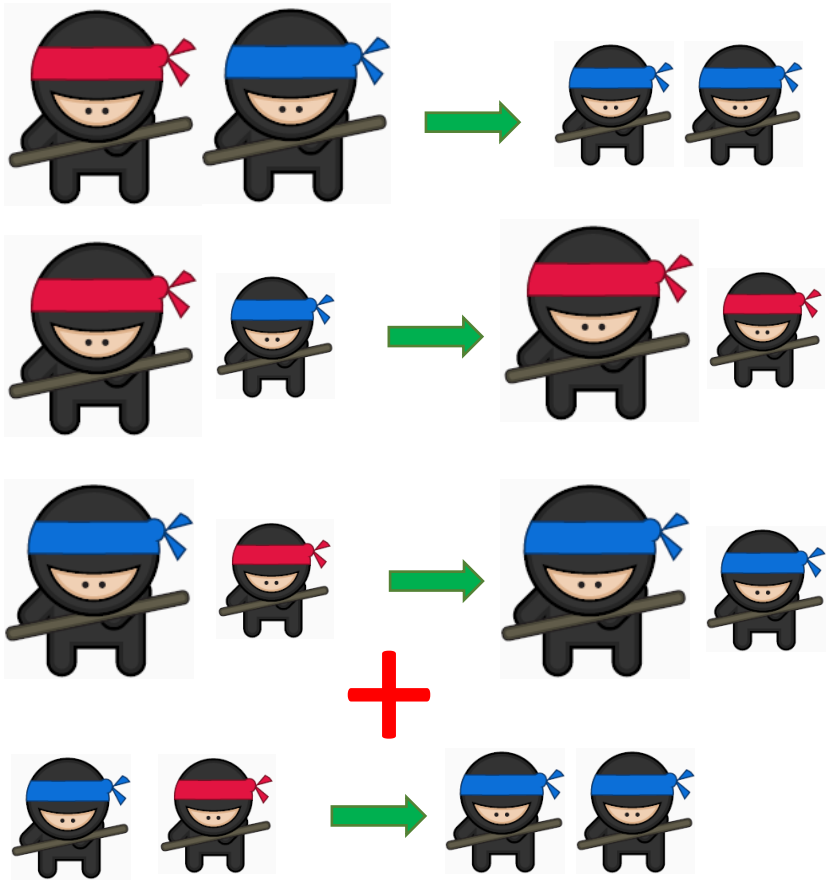


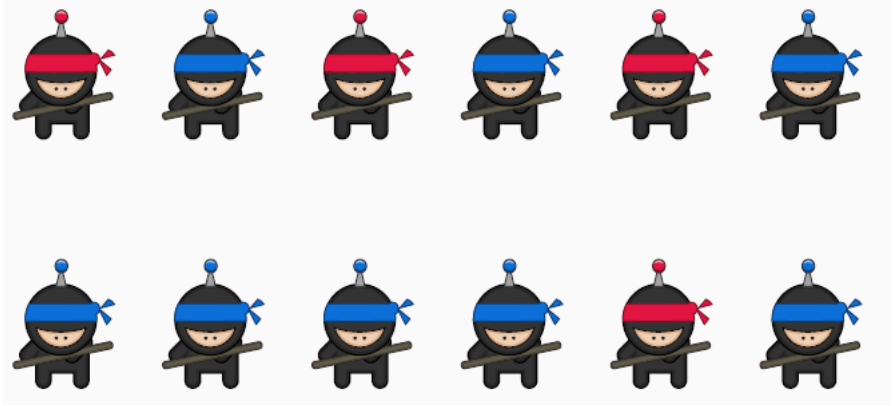
The *Professional* (*P*) Sensei Takes Over

He finds out that in case of a tie, things can go bad because the first rule has no priority over the other two NO CONSENSUS



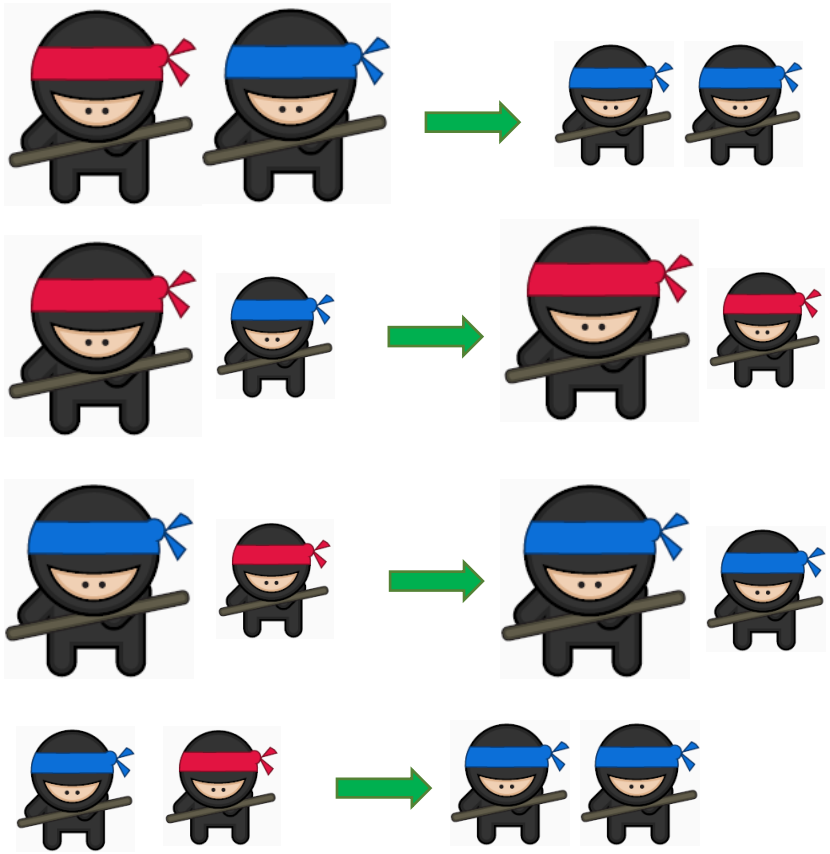
The New Protocol of Sensei *P*



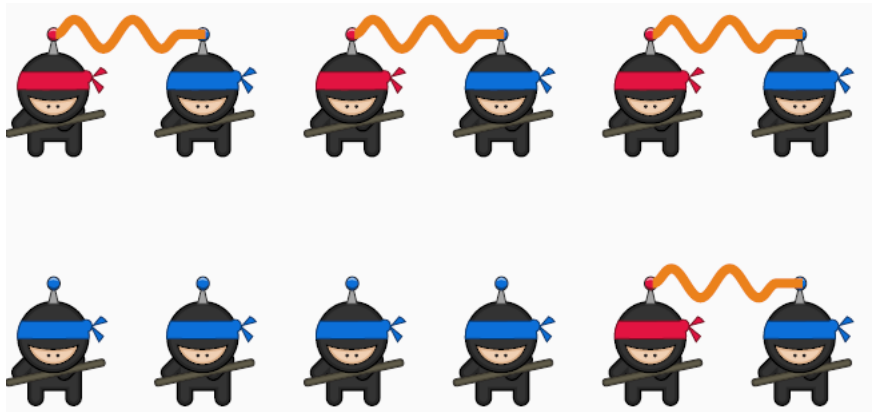


Passive blue ninjas convert
passive red ninjas to their color

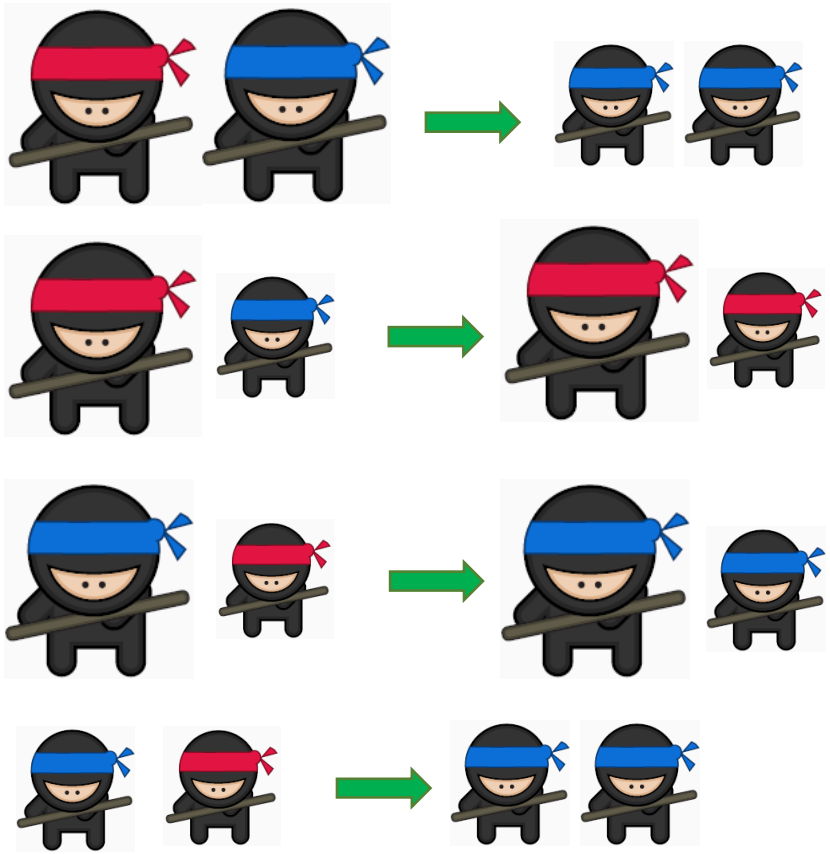
The New Protocol of Sensei *P*



Passive blue ninjas convert
passive red ninjas to their color

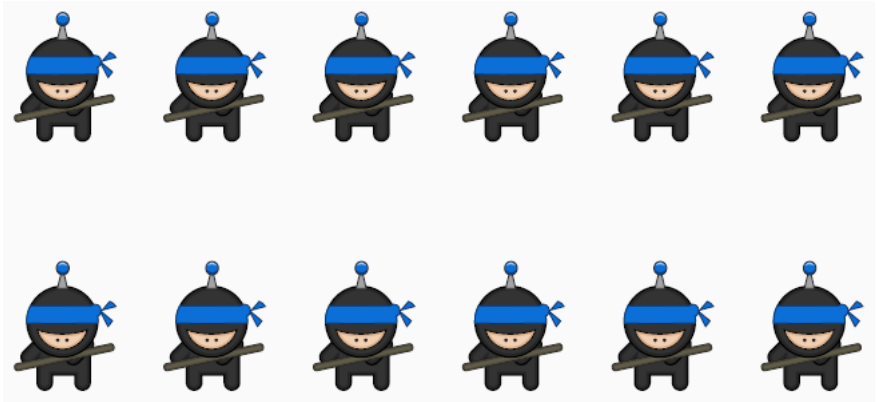


The New Protocol of Sensei *P* (Animation)



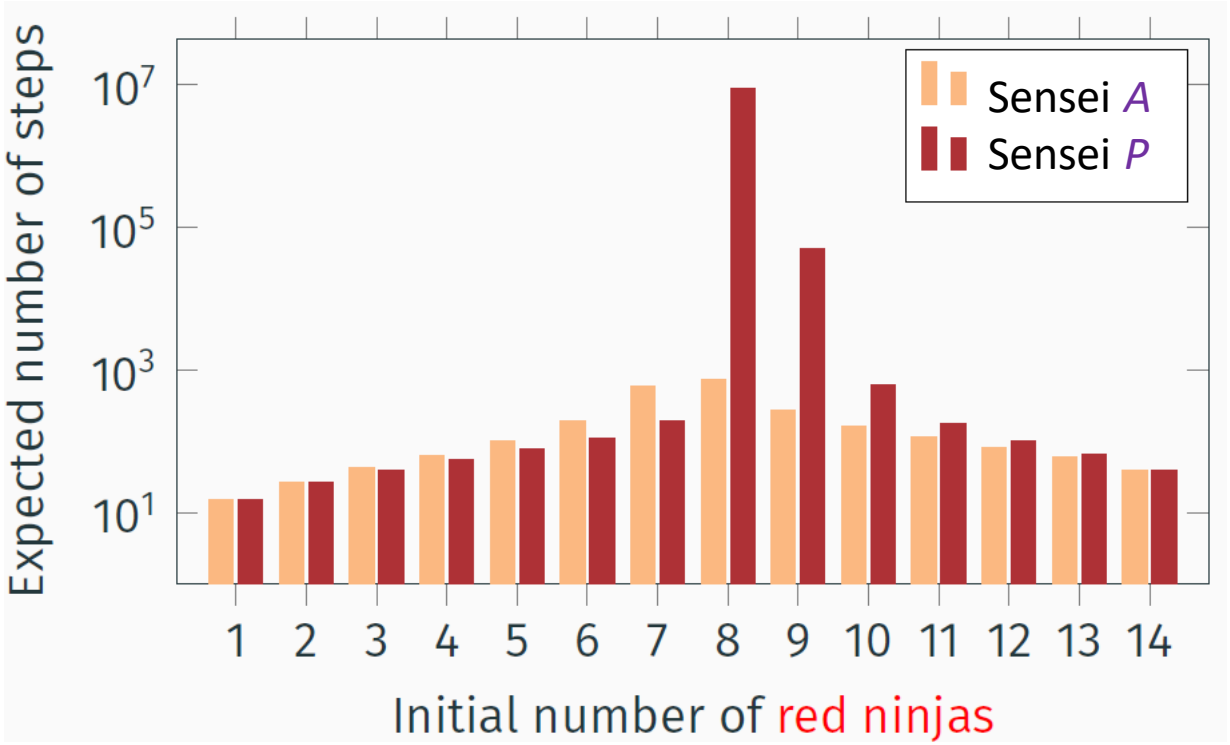
Passive blue ninjas convert
passive red ninjas to their color

Don’t Attack



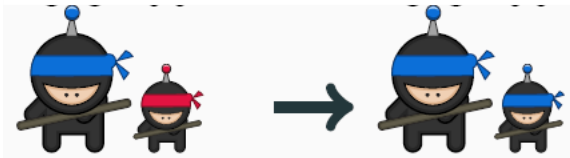
Another Night Disaster Strikes… (Animation)

The *Master* (*M*) Sensei Takes Over…

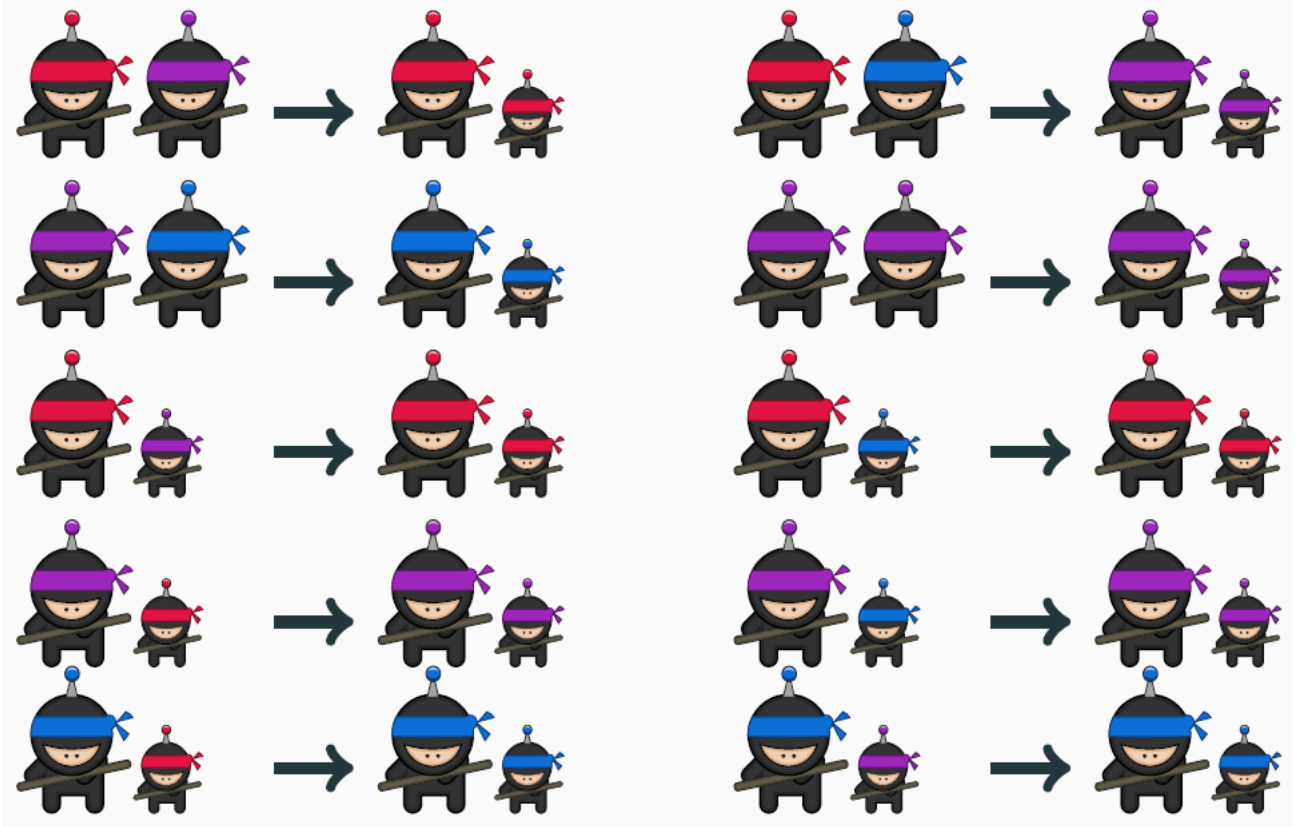


Expected number of steps to stable consensus for a population of 15 ninjas

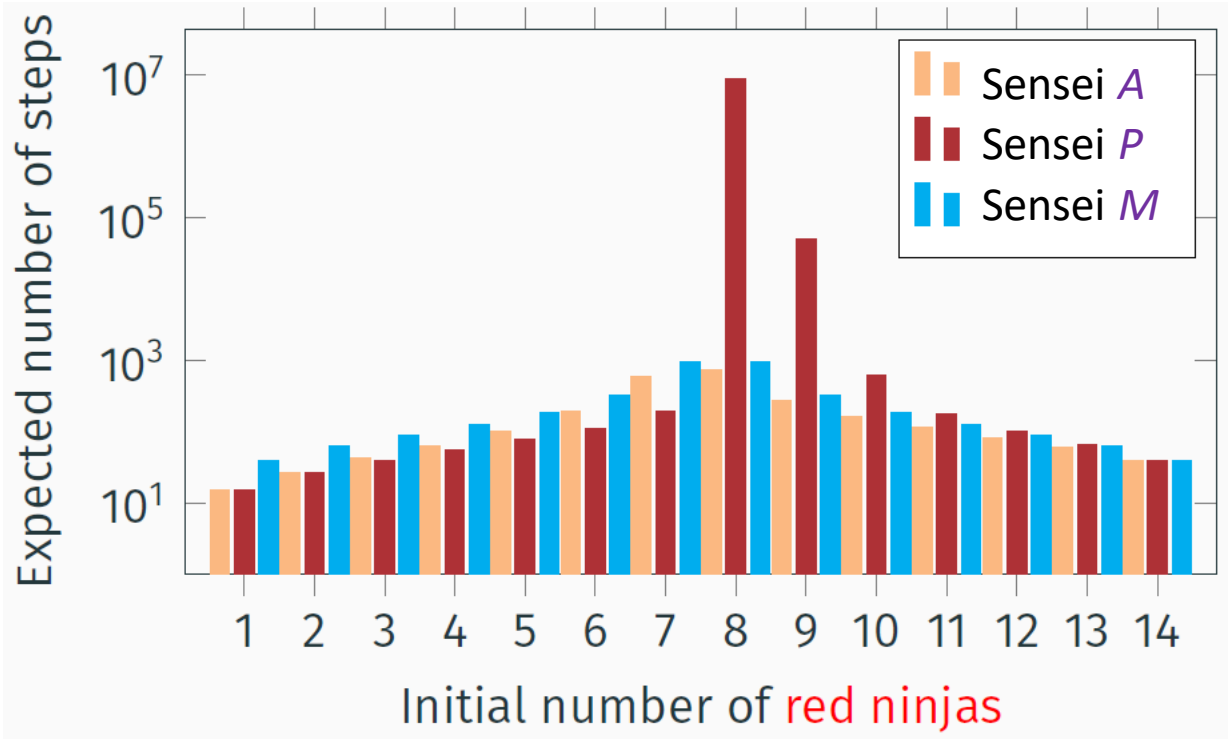
The New Protocol of Sensei *M* (Animation)



The New Protocol of Sensei *M* (Animation)



The New Protocol



Expected number of steps to stable consensus for a population of 15 ninjas

M Wonders while Wandering …

Formalization Questions:

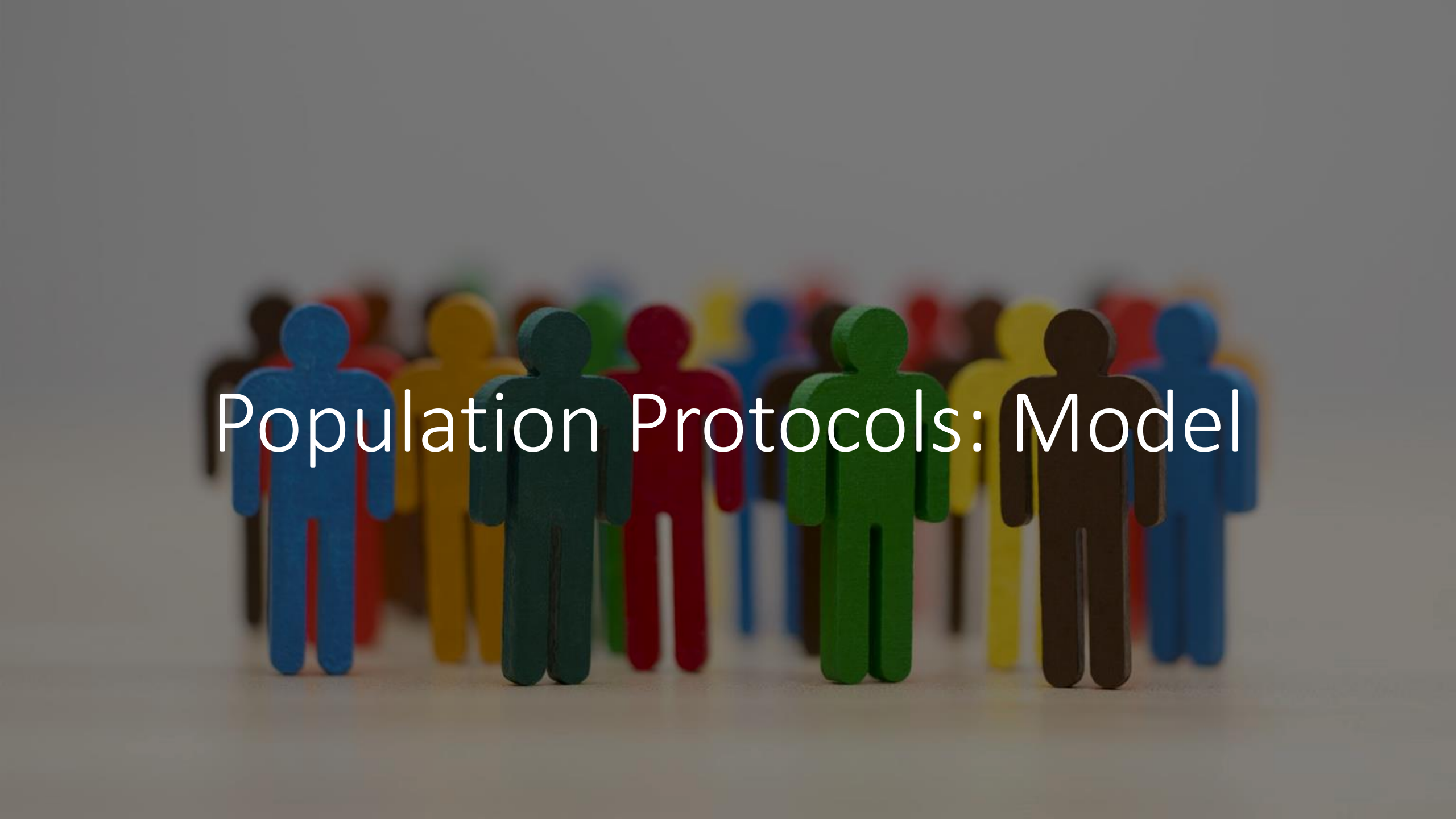
- What is a protocol?
- When is a protocol "correct"?
- When is a protocol "efficient"?

Verification Questions:

- How do I check that my protocol is correct?
- How do I check that my protocol is efficient?

Expressivity Questions:

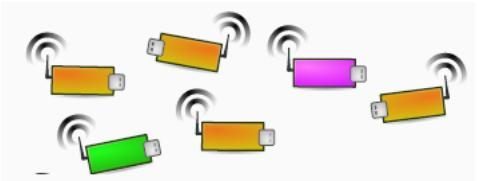
- Are there protocols for other problems?
- How large is the smallest protocol for a problem?



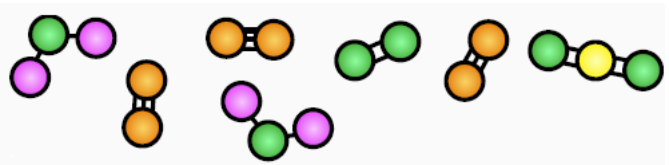
Population Protocols: Model

Motivation

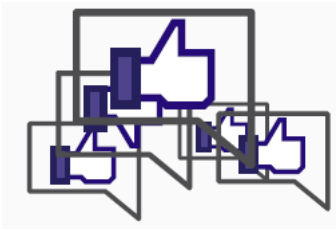
Formal model of distributed computation by collections of identical (anonymous), finite-state (weak) and mobile agents (asynchronous – unpredictable interactions).



Ad-hoc networks of mobile sensors



“Soups” of molecules (Chemical Reaction Networks)



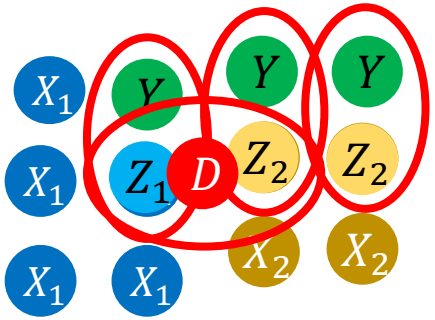
People in Social Networks

Discrete Chemical Reaction Network (CRN) Model

- Finite set of **species** and finite set of **reactions**
- **Configuration**: integer counts of species, changes by successive asynchronous reactions
- Similar Models: Population Protocols, Petri Nets, Vector Addition Systems

$$X_1 \to Z_1 + Y$$
$$X_2 \to Z_2 + Y$$
$$Z_1 + Z_2 \to D$$
$$D + Y \to \emptyset$$

CRN	Population Protocols
Molecule (anonymous)	Agent (anonymous)
Species	State
Reaction (asynchronous)	Transition Function (2 input, 2 output)

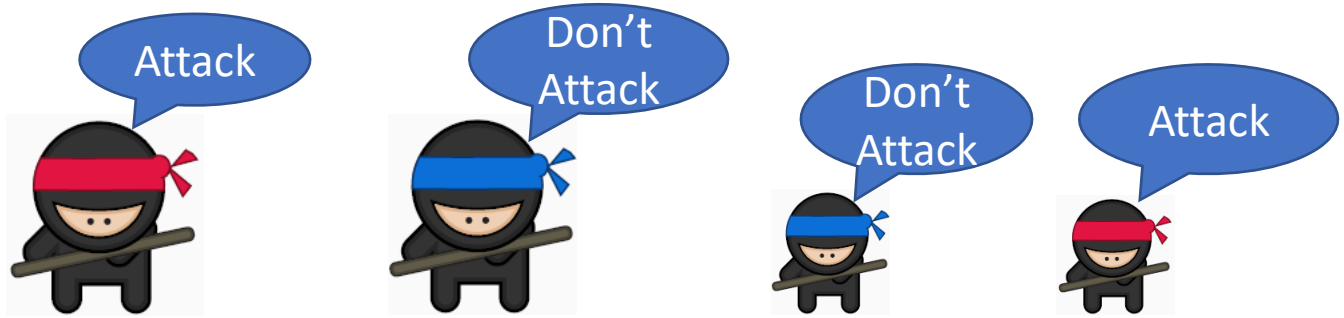


Formal Model

- States: Finite set Q
- Output: $O: Q \to \{0,1\}$
- Initial States: $I \subseteq Q$
- Transitions: $T \subseteq Q^2 \times Q^2$

Formal Model

- States: Finite set Q
- Output: $O: Q \to \{0,1\}$
- Initial States: $I \subseteq Q$
- Transitions: $T \subseteq Q^2 \times Q^2$

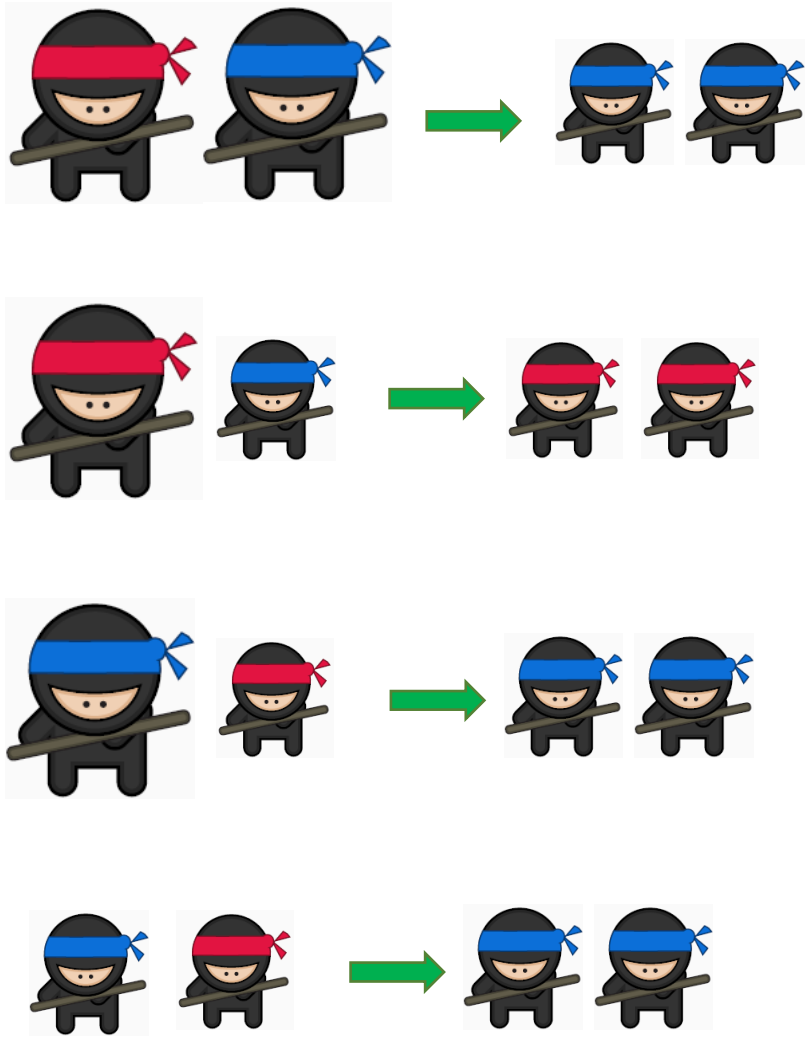


Formal Model

- States: Finite set Q
- Output: $O: Q \to \{0,1\}$
- Initial States: $I \subseteq Q$
- Transitions: $T \subseteq Q^2 \times Q^2$

Formal Model

- States: Finite set Q
- Output: $O: Q \to \{0,1\}$
- Initial States: $I \subseteq Q$
- Transitions: $T \subseteq Q^2 \times Q^2$

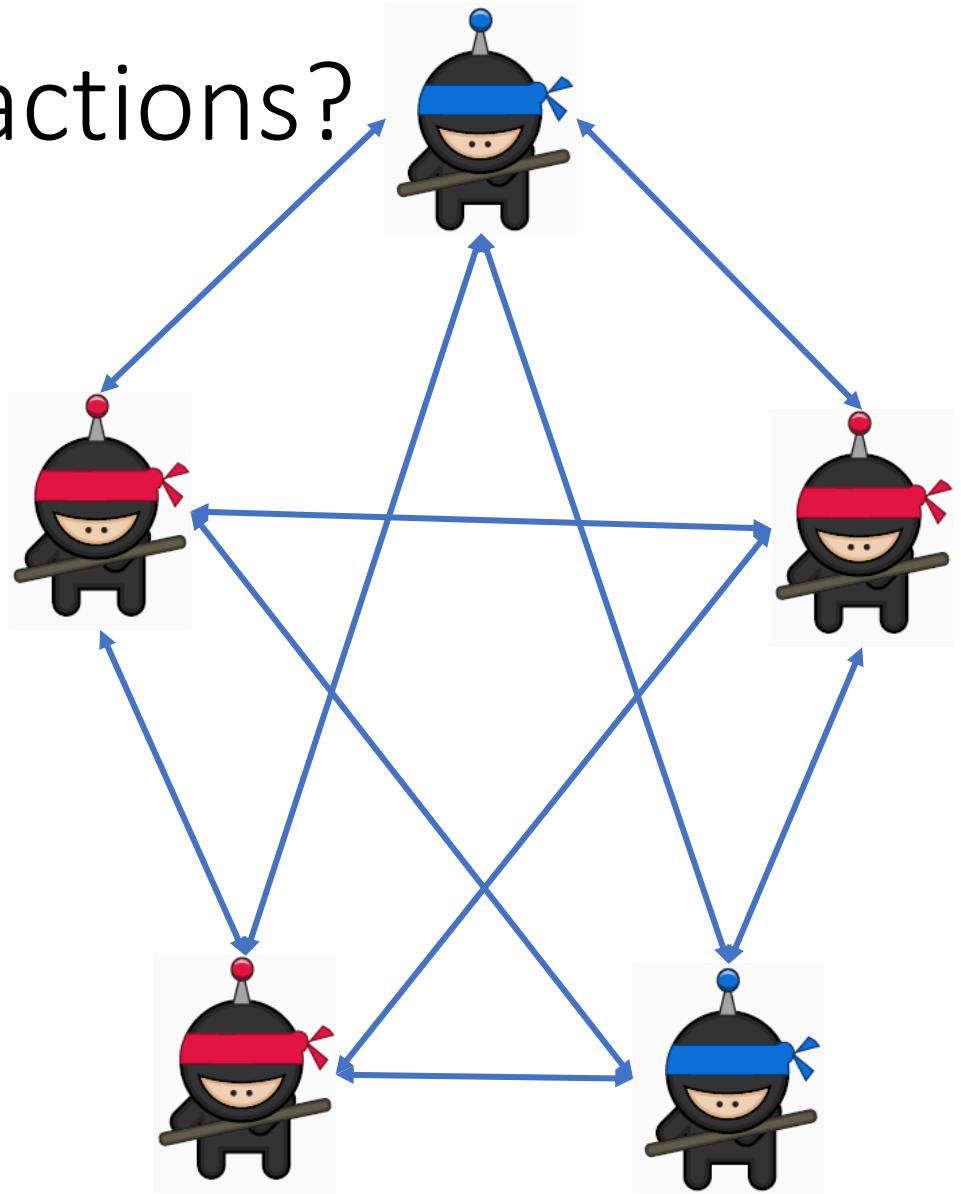


How do “we” Choose Interactions?

- Interaction Graph (usually we assume the complete (un)directed graph)
- Interaction: initiator – responder
  - 2-way communication
  - 1-way communication
    - Transmission model
    - Observation model

A scheduler (daemon ☺) chooses the next interaction:

- Adversarial scheduler (worst-case) but strongly fair
- Other: Uniformly random scheduler



Formal Model

- States: Finite set Q
- Output: $O: Q \to \{0,1\}$
- Initial States: $I \subseteq Q$
- Transitions: $T \subseteq Q^2 \times Q^2$
- Configurations: $Q \to \mathbb{N}$ *(for complete interaction graphs)*
- Initial Configurations: $I \to \mathbb{N}$ *(for complete interaction graphs)*

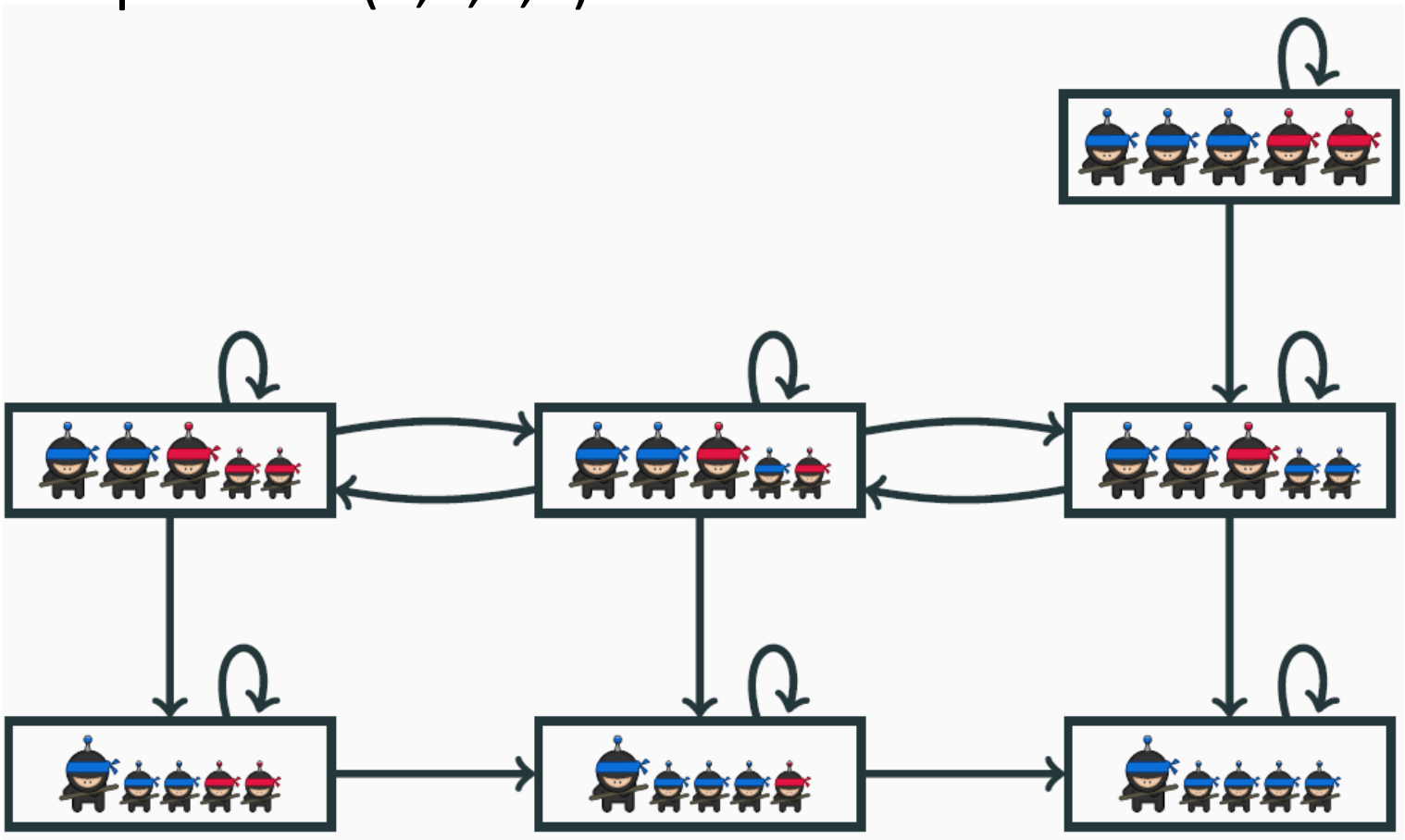
(4 3 6 1)

(6 8 0 0)

Configuration Graph

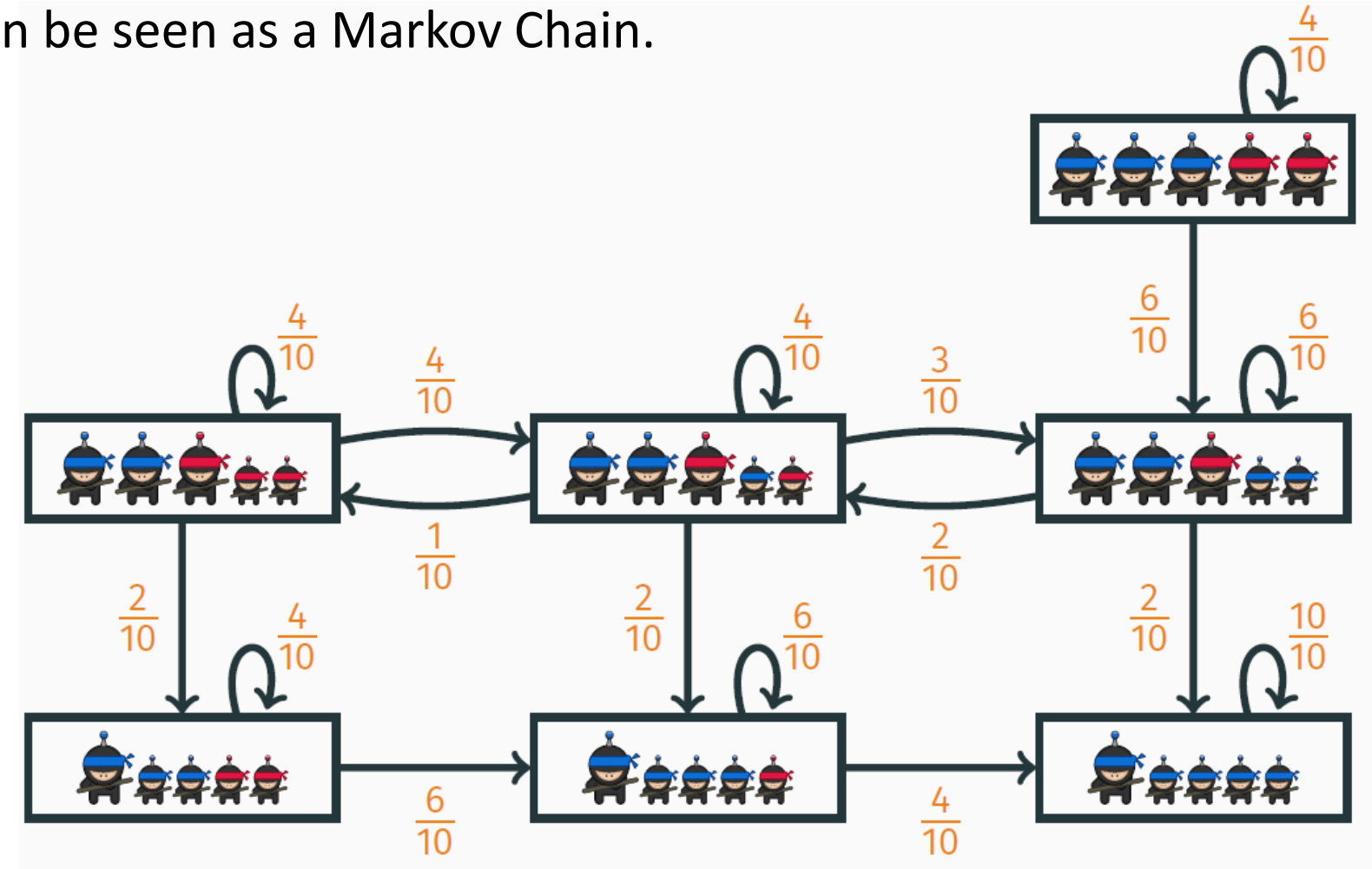
An execution is an infinite path from initial configuration in the configuration graph.

Configuration Space for (3,2,0,0):



Uniformly Random Scheduler

The configuration graph can be seen as a Markov Chain.



Stable Computable Predicates

A predicate $P: I \to \{0,1\}$ is stably computable under a strongly fair adversarial scheduler, if there exists a population protocol such that for every $c \in I$, all executions starting at c reach eventually a stationary configuration where each agent correctly outputs whether P is true or false.

For a uniformly random scheduler, the execution reaches a stationary configuration with probability 1. Note that in this case the scheduler is strongly fair as well.

Time Complexity

- For an adversarial scheduler the fairness condition is not enough to reason about time complexity.

- For uniformly random schedulers we can count the number of interactions to convergence
  - Parallel time: since it is probably folly to assume that in each step only one interaction takes place (no parallelism!!!) we use the parallel time which is defined as the number of interactions divided by the number of agents.
    - A Poisson process governs the occurrence of interactions in parallel: expected O(1) interactions per time unit.

Compute the “or” Function

1. Write the program (2-way communication) (Q, O, I, δ)
2. Prove Correctness
3. Prove Complexity

Some Protocols

Flock of Birds (adversarial fair scheduler)

We want to find out whether at least 5 birds in a flock are sick. Each bird is equipped with a sensor that detects elevated temperature:

- States: $Q = \{q_0, q_1, \dots, q_5\}$
- Initial States: $I(0) = q_0$ and $I(1) = q_1$
- Output: $O(q_i) = 0, 0 \leq i \leq 4$ and $O(q_5) = 1$
- Transitions: $(q_i, q_j) \rightarrow (q_{i+j}, q_0)$, if $i + j < 5$
  $(q_i, q_j) \rightarrow (q_5, q_5)$, otherwise

Epidemics (Uniform Scheduler)

If one is sick (state 1) then everyone will get infected:

One-way communication $(initiator, responder)$.

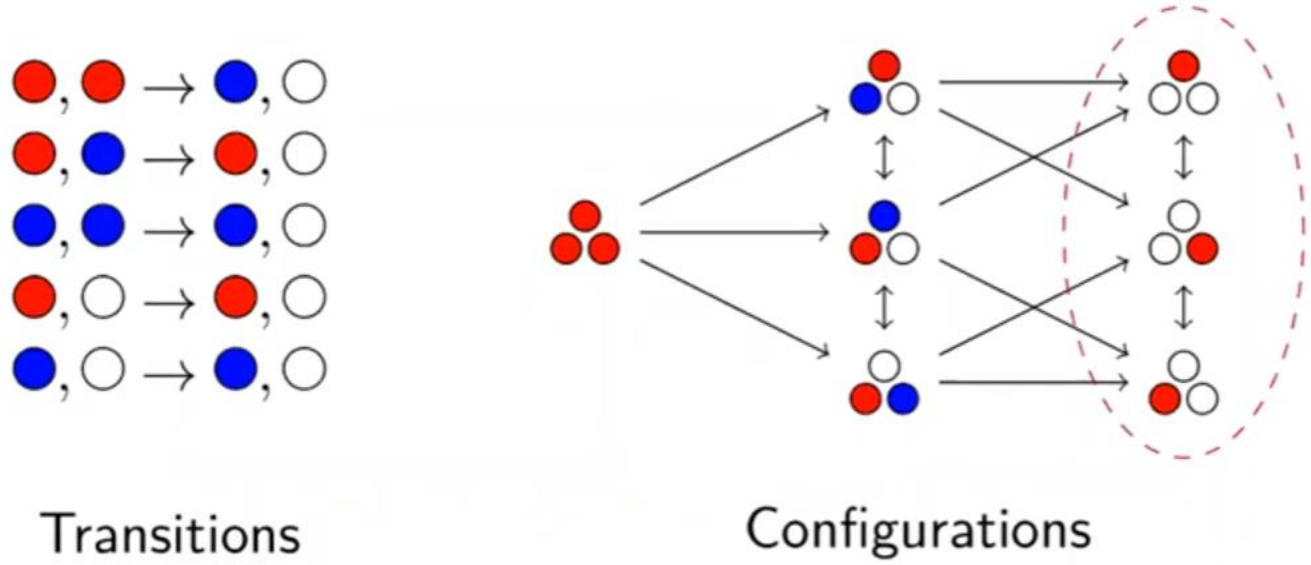
- States: $Q = \{0\ (susceptible), 1(infected)\}$
- Initial States: $I(0) = q_0$ and $I(1) = q_1$
- Output: $O(q_0) = 0$ and $O(q_1) = 1$
- Transitions: $(q_i, q_j) \rightarrow (q_i, \max\{q_i, q_j\})$

Leader Election

Among n agents we wish one to become the leader.

- States: $Q = \{L, F\}$ (Leaders and Followers)
- Initial States: $I = \{L\}$, for all nodes – initially all nodes are leaders
- Output: $O(L) = 1$ for some node, and $O(F) = 0$, for the rest
- Transitions: $(L, L) \rightarrow (L, F)$

Using Leader Election: Parity



- red: odd, blue: even, white non-leader
- Initially all are red
- Coalesce values along with leadership
- Last remaining leader shows parity
- Stable computation: converges to correct answer then stays there

Using Leader Election: Remainder $\mathbf{mod}\ m$

Count the number of agents in some special state A, modulo a constant m.

- States: $Q = \{\langle l, x\rangle\}, l \in \{L, F\}$ and $x \in \{0,1, \dots, m-1\}$
- Initial States: $I(A) = \langle L, 1\rangle$ and everything else to$\langle L, 0\rangle$
- Output: The leader contains the number of As modulo m.
- Transitions: $(\langle L, x\rangle, \langle L, y\rangle) \to (\langle L, (x+y)\ mod\ m\rangle, \langle F, 0\rangle)$

Proof of Correctness

Invariant: the sum over all agents of the second component (mod m) is unchanged by the transition.

Thus, the unique leader will contain the desired output.

3-State Approximate Majority Protocol

Uniform scheduler. Initial configuration of x, y and b $(blanks)$ reach consensus provided that the majority exceeds minority by a sufficient margin. One-way communication.

- States: $Q = \{x, y, b\}$
- Initial States: $I(b) = b, I(x) = x, I(y) = y$
- Output: $O(x) = O(y) = 1$ and $O(b) = 0$
- Transitions: $(x, y) \to (x, b)$

  $(x, b) \to (x, x)$

  $(y, x) \to (y, b)$

  $(y, b) \to (y, y)$

There are similar biological switches at the level of a cell (e.g., delta-notch mechanism)

Theorem: Let τ_* be the time at which all are x or all are y for the first time. Then for any fixed $c > 0$ and sufficiently large n:

$$Pr[\tau_* \geq 6754nlogn + 6759cnlogn] \leq 5n^{-c}$$

Theorem: With high probability, the 3-state approximate majority protocol converges to the initial majority value if the difference between the initial majority and initial minority populations is $\omega(\sqrt{nlogn})$.

Expressiveness

Basic population protocols with adversarial and strongly fair scheduler with a complete interaction graph compute precisely the predicates definable in Presburger Arithmetic:

- 1st order theory of natural numbers with addition, equality, 0 and 1
  - e.g., "x is even": $\exists y(x = y + y)$
- Decidable (although in double exponential time)
- Quantifier elimination (using predicates $<$ and $\equiv_k$)

What can(‘t) we do?

We can:

- mod_k for fixed k (coalescence)
- $<$ and $=$ (cancellation)
- Addition by renaming: $A \to B$ and $A \to C$ implements $C = A + B$
- Run protocols in parallel for $f \vee g$, $f \wedge g$, etc.
- Relabel output for $\neg f$

We can’t:

- Everything else ☺
  - Anything that requires nested iterations like multiplication by a non-constant and division

Main Characteristic of Population Protocols and Variants

Population protocols have no communication structure:

- No network addresses
- No persistent connections
- No addressable memory

Nowadays, it seems that agent ids and relatively large states are OK.

Variants of Population Protocols that Make it Stronger

- Allow for ids (community protocols)
- $O(1)$ bits on edges (mediated population protocols)
- Base station (sensors communicate with a powerful computational unit)
- Oracles (existence of a leader, stationary configuration, leader has interacted with all agents)
- Randomized scheduling

References

1. Black Ninjas in the Dark: Formal Analysis of Population Protocols
2. An Introduction to Population Protocols
3. New Models for Population Protocols