A Self-healing Multipath Routing Protocol

Thomas Meyer, Lidia Yamamoto, Christian Tschudin

Computer Science Department, University of Basel, Switzerland

BIONETICS 2008, Hyogo, Japan

November 26th 2008

UNI BASEL

Motivation

- Today's software depends on reliable hardware.
- But future hardware will be unreliable:
  - Smart Dust: nano computers, large quantity, low price [Bahar 2001]
  - Probabilistic Chips (PCMOS): low energy consumption [Palem 2007]
- Consequences:
  - Execution of wrong instructions
  - Corruption of code in memory

Our Goal:

Reliable execution in unreliable environments

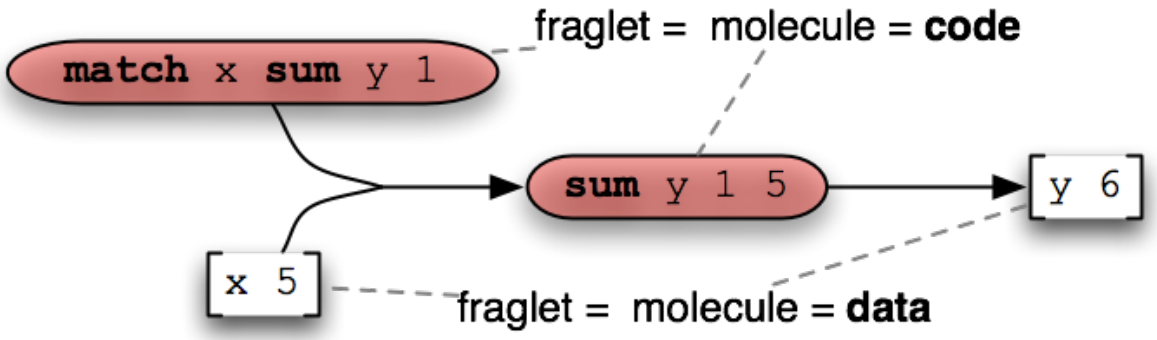
Provide methods for designing resilient software

Contents

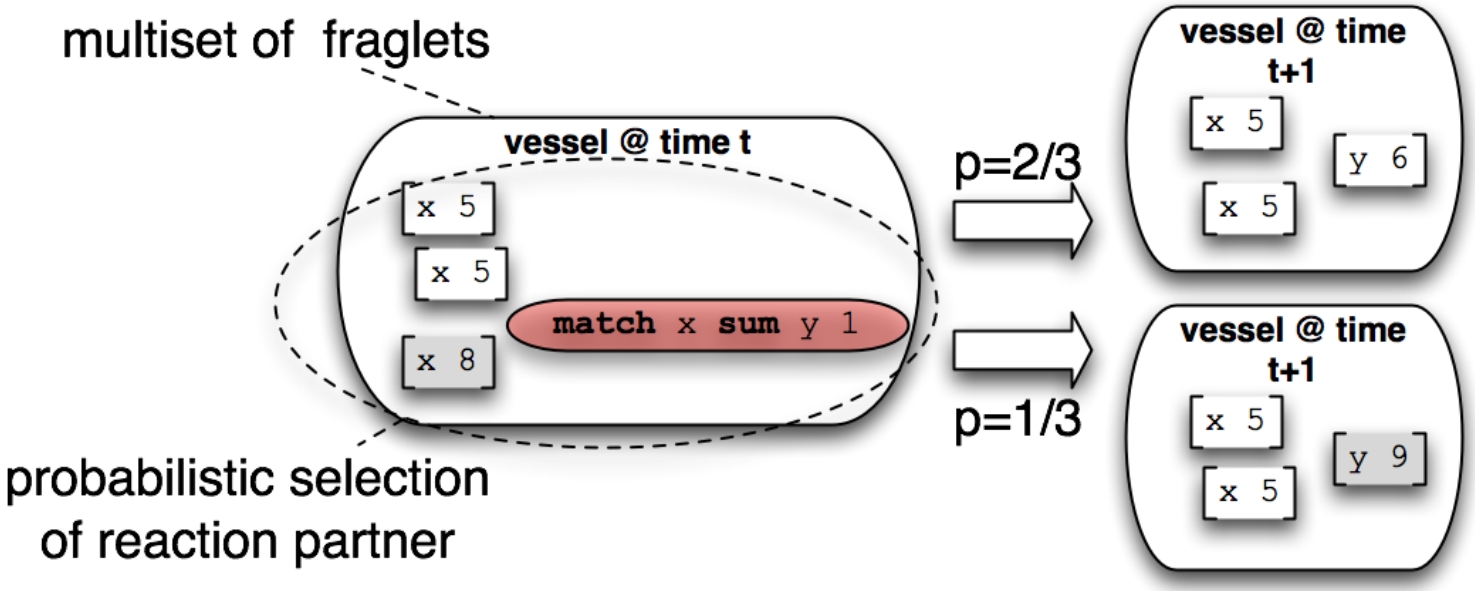
- Motivation
- Method
  - Fraglets: an artificial chemistry
  - Self-healing code using autocatalytic *Quines*
- Multipath routing prototocol design
- Results
- Conclusions

Fraglets: An Artificial Chemistry [Tschudin 2003]

- Fraglets is a constructive artificial chemistry [Dittrich et al 2001] based on string rewriting rules.

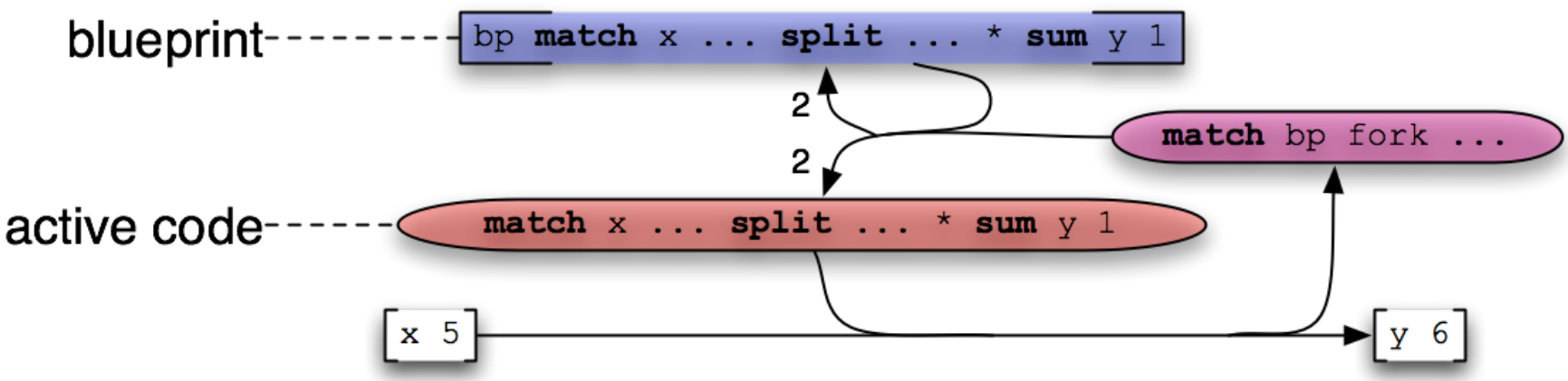


- Fraglets runs in a virtual machine (interpreter).



Self-replicating Code: *Quines*

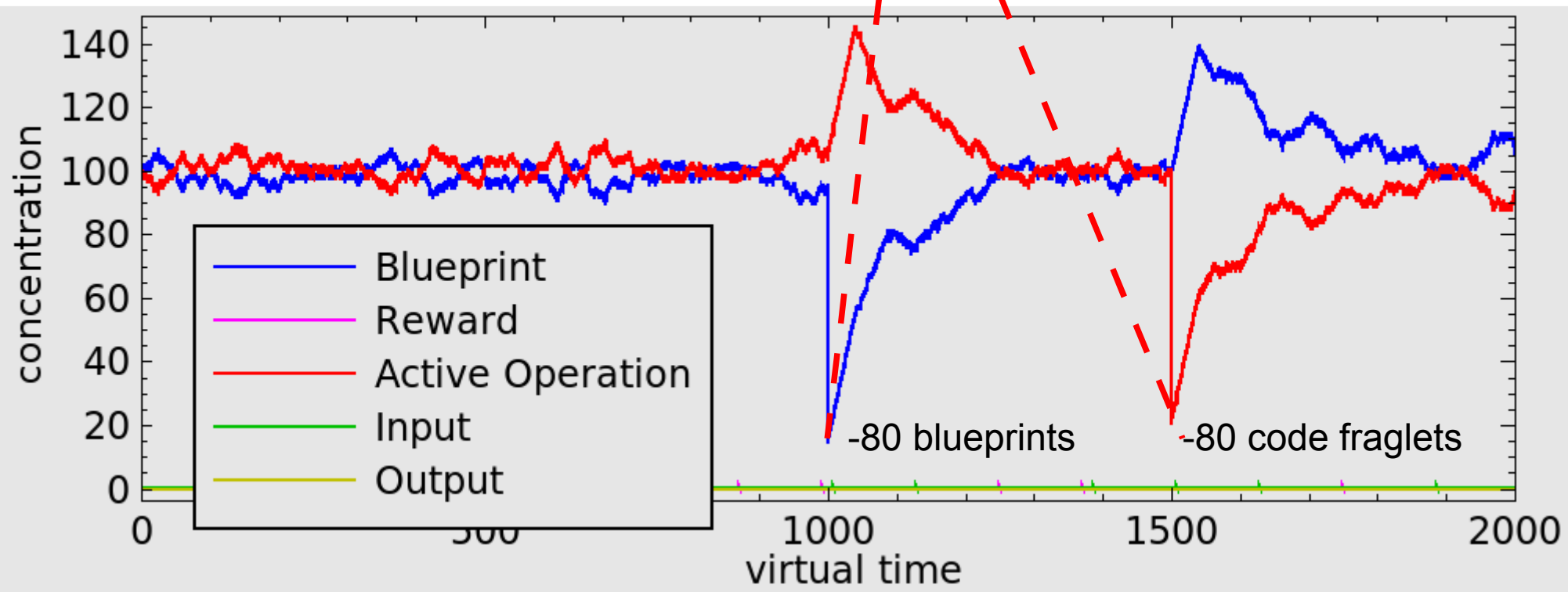
- Reactions consume their educt code and data molecules.
- Code must continously be regenerated. Or better, code has to regenerate itself!
- A self-replicating *Quine* duplicates its active part and blueprint when processing a data molecule. [Yamamoto et al 2007]



- Limited vessel capacity ⇒ selective pressure ⇒ survival of replication (defective, non-replicating code will be displaced)

Self-healing Code: Autocatalytic *Quines*

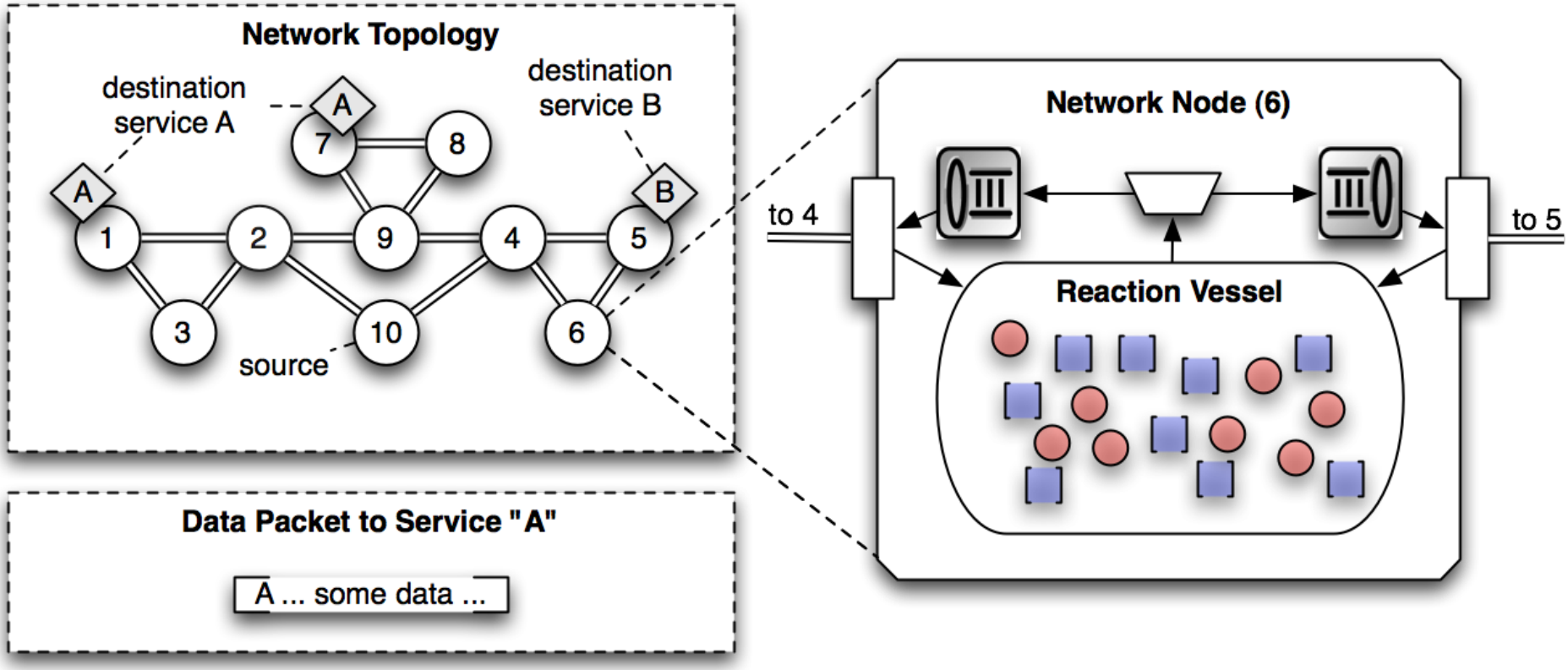
- In steady-state, blueprints and active rules are present with equal concentration.
- The code is resilient to knock-out attacks.
- The code is resilient to most mutation attacks.



- This results in emergent code homeostasis

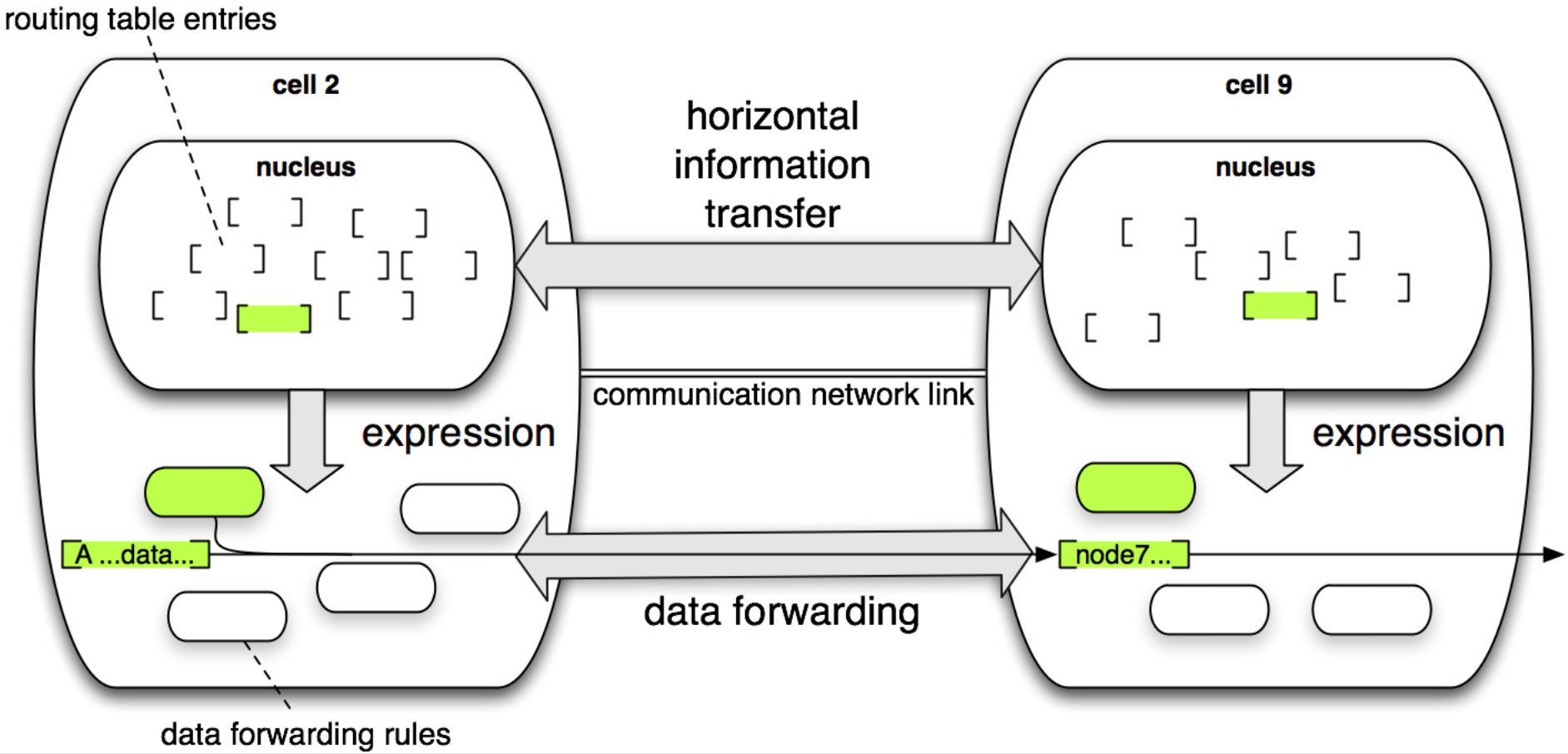
Self-healing Routing Protocol

- How to build more complex self-healing programs based on *Quines*?
- In our case: Route data packets to a labeled service (e.g. “A”) within a network of Fraglet nodes:



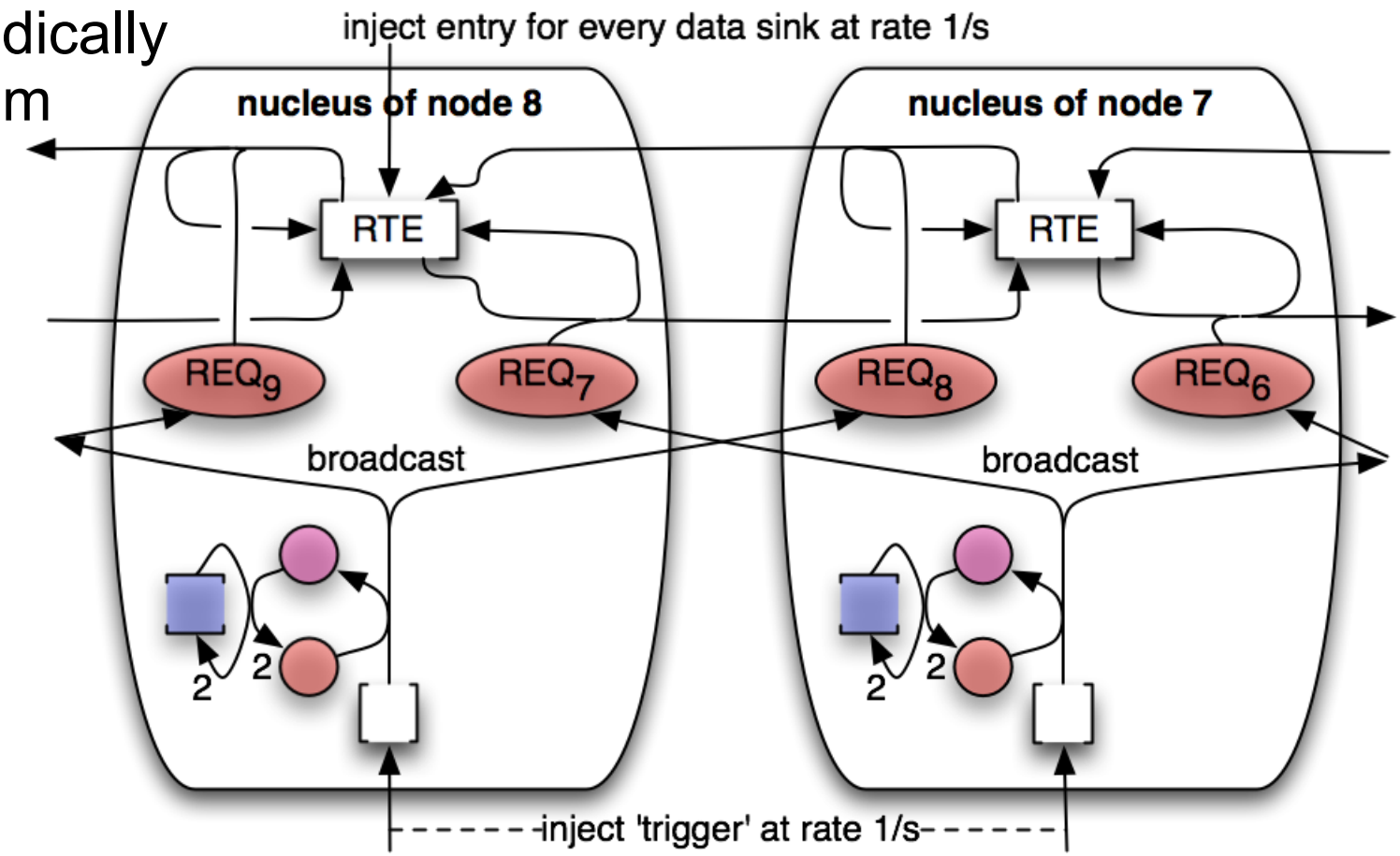
Routing Protocol Overview: The Cell Metaphor

- **Information:** Routing table entries (RTEs) are exchanged between the nuclei of neighbor nodes.
- **Expression:** Passive routing table entries (RTEs) are activated.
- **Operation:** Active rules forward data packets to the next hop.



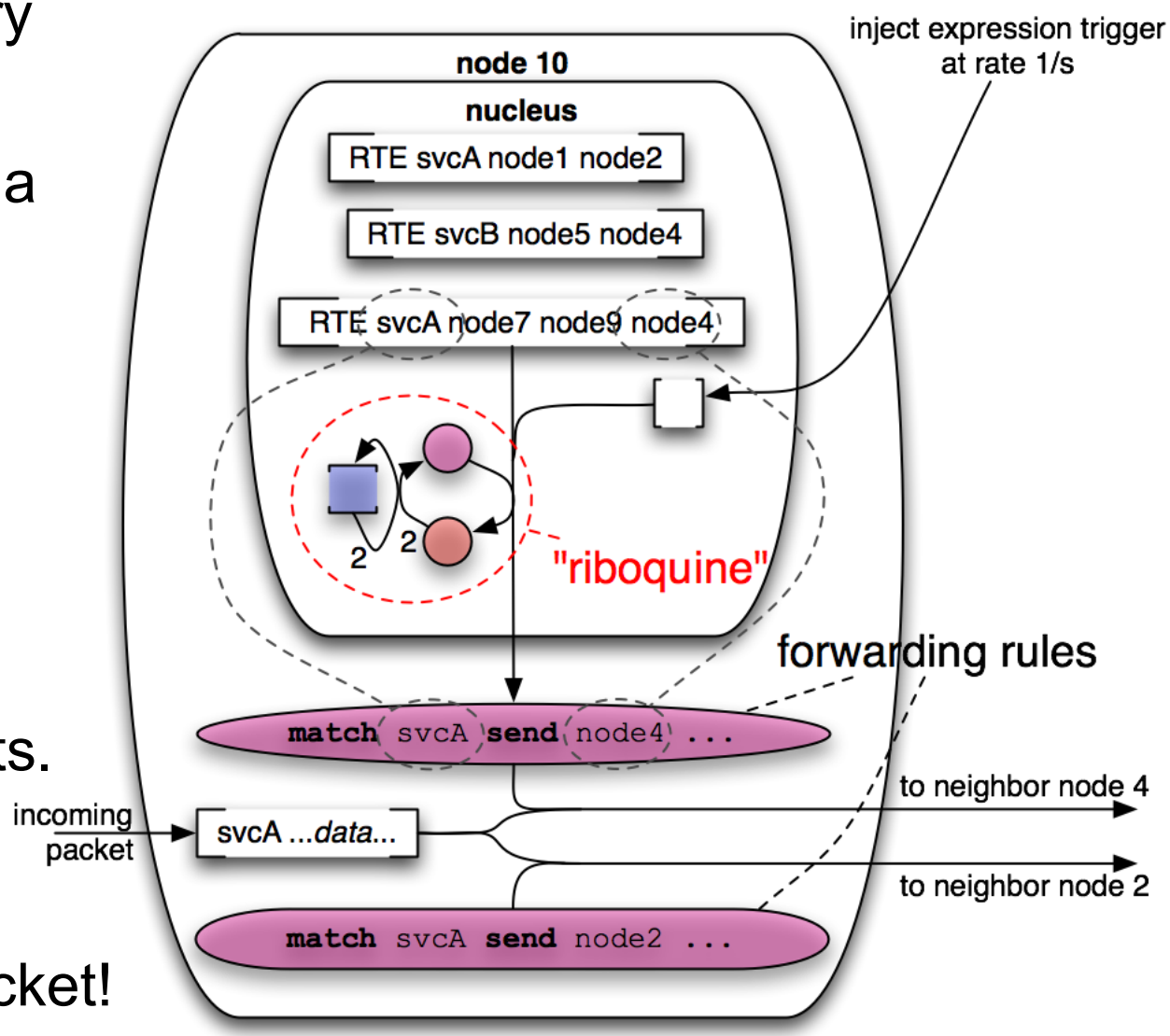
Routing Table Entry (RTE) Dissemination

- Routing Table = Multiset of molecules: **[RTE *svc* *hop1* ... *hopn*]**
- Every service injects a local RTE: **[RTE *svc*]**
- Every node periodically injects trigger molecules.
- A *Quine* periodically obtains random RTEs from neighbors.
- Result: Slow diffusion of reachability information across the network.



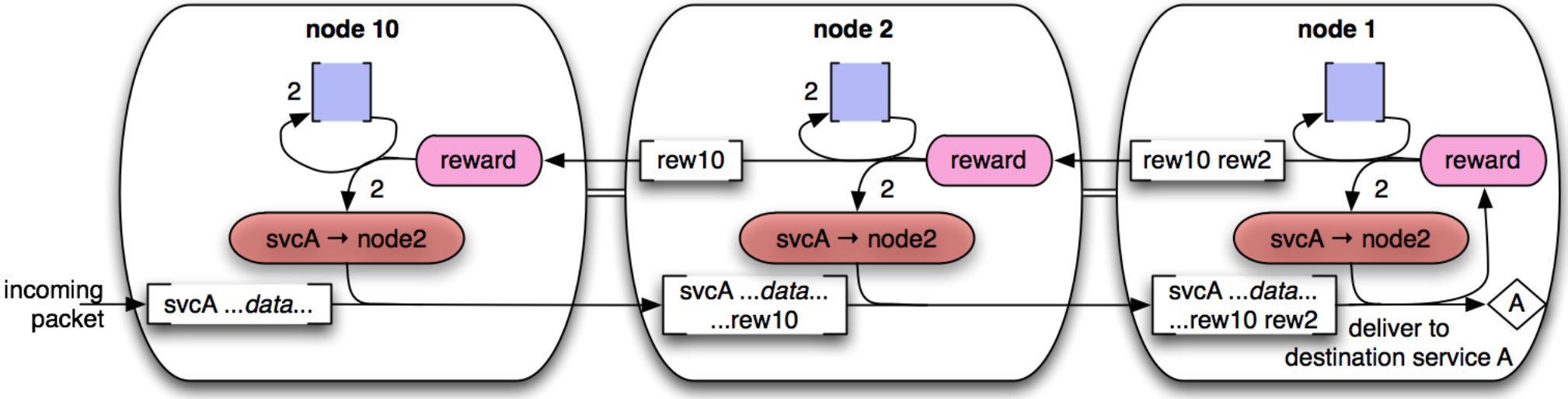
Expression of Routing Table Entries (RTEs)

- The „riboquine“ periodically expresses an arbitrary RTE.
- The concentration of a forwarding rule is proportional to the concentration of its corresponding RTE.
- Forwarding rules for the same destination service compete for incoming data packets.
- Forwarding rules are consumed when forwarding a data packet!



Forwarding Path Reinforcement

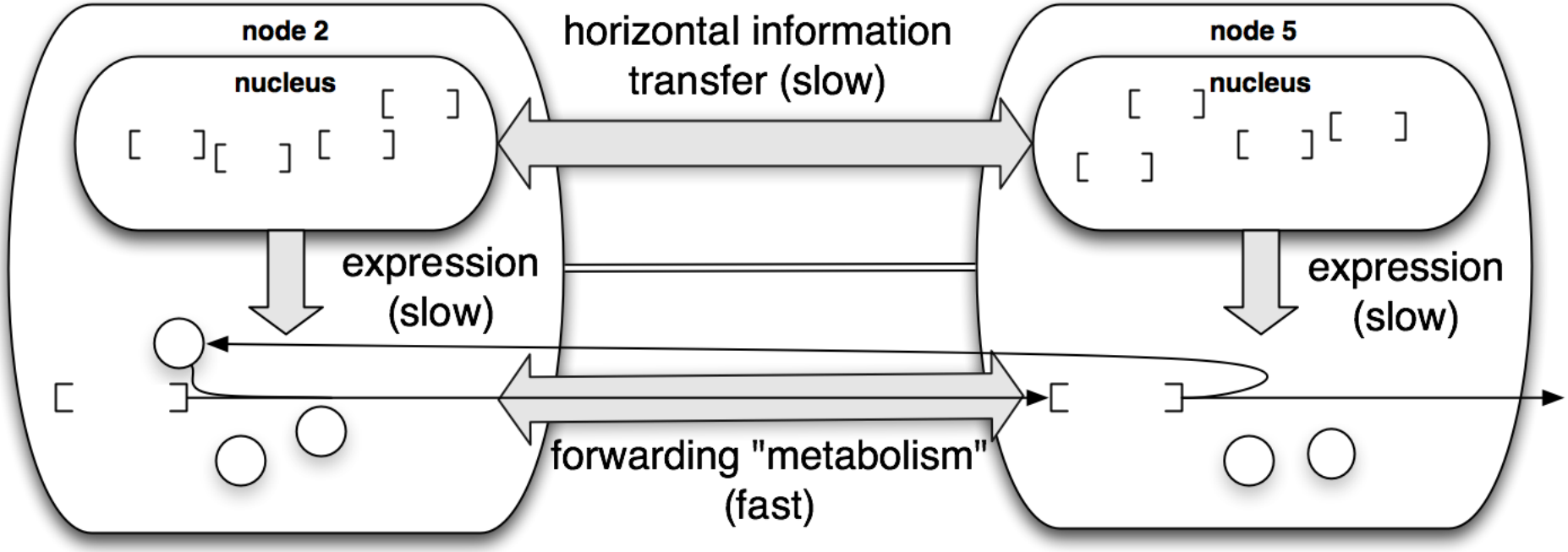
- Forwarding rules must replicate themselves.
- We only let those forwarding rules replicate that successfully delivered a data packet to the destination service.



- The replication latency is equal to the node-to-end round trip time of the data packet.
- The growth rate of a forwarding *Quine* is equal to the rate of successfully delivered data packets.
- Multiple paths may coexist.

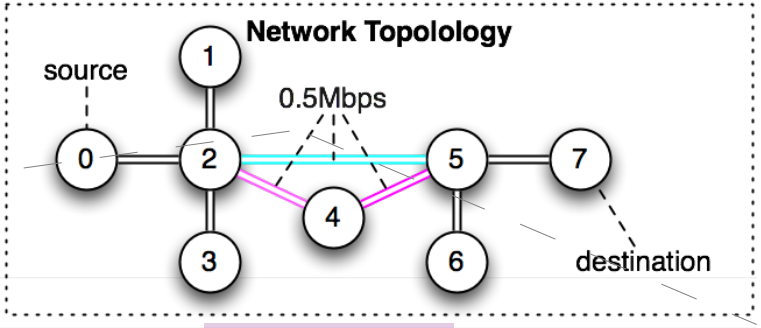
Summary of Operation

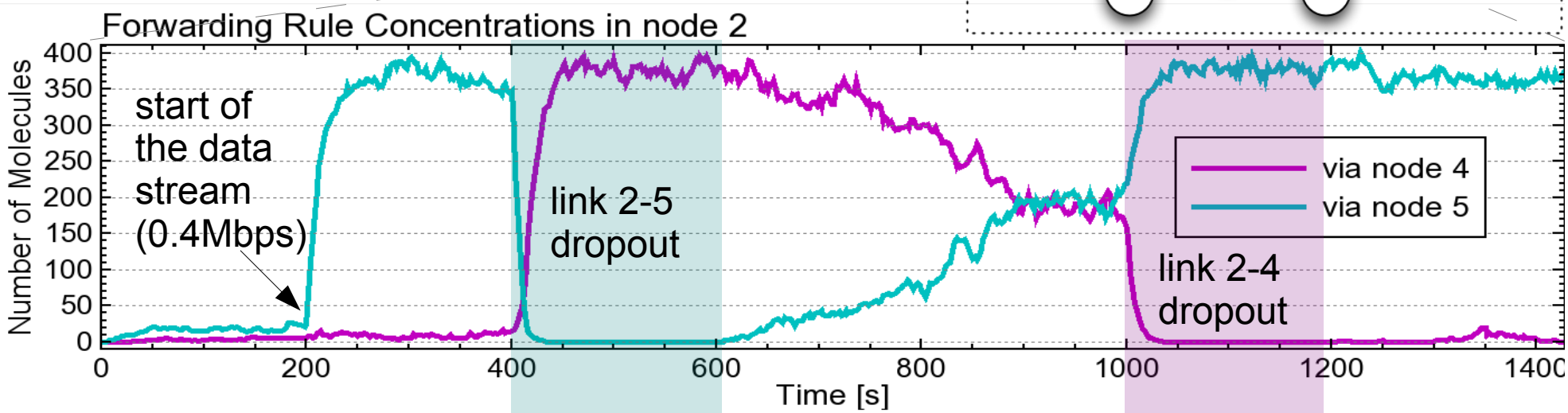
- The exchange of RTEs is needed to announce new services and their reachability (slow process, protected in the nucleus)
- By expressing the RTEs the „riboquine“ injects new or displaced rules (novelty) into the main vessel
- In the harsh environment of the main vessel, forwarding rules compete for data packets; successful rules increase their fitness

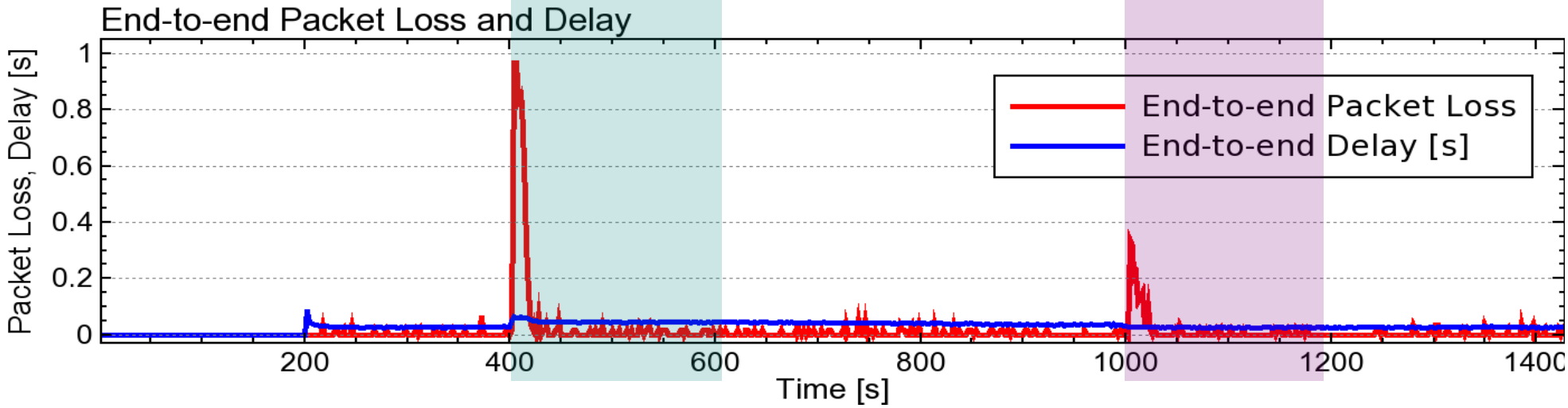


Results: Link Dropout Behavior

- The protocol shows quick reaction to packet loss, but only weakly favors shorter paths

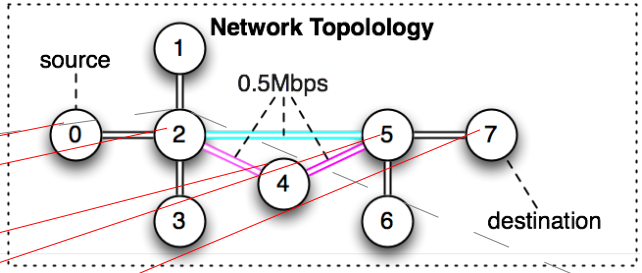


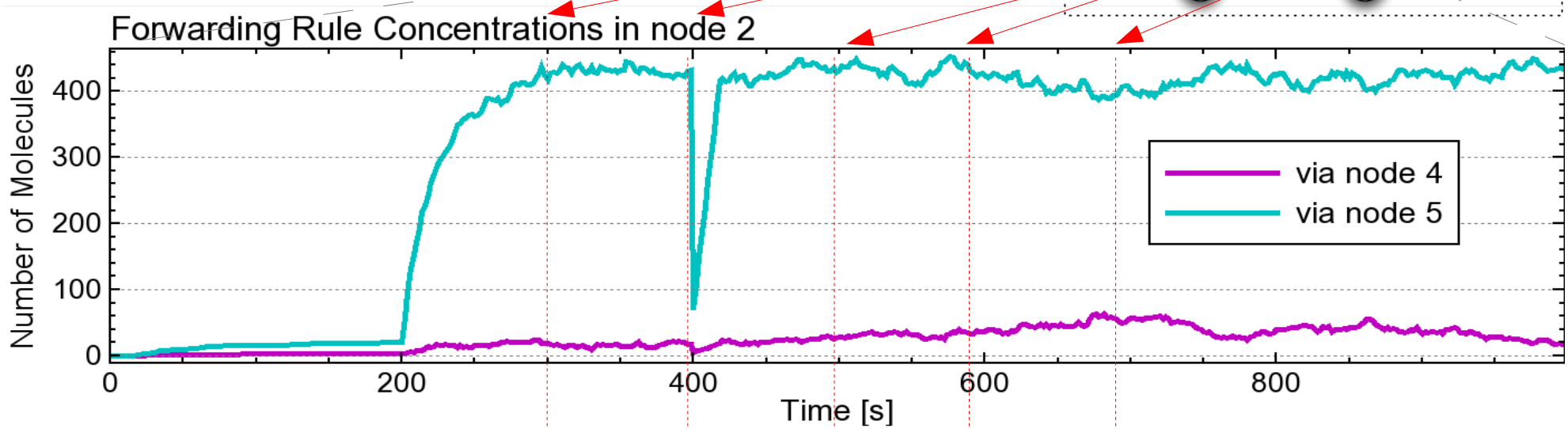


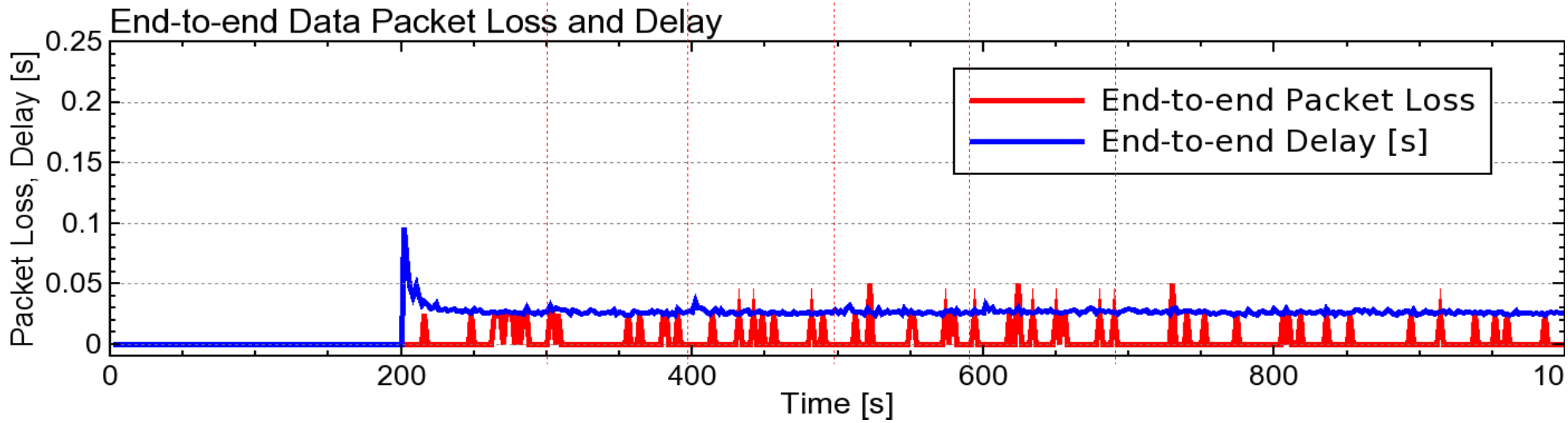


Results: Resilience to Code and Data Destruction

- The software is reslilient to destruction attacks (e.g. Random destruction of 80% of the molecules)







Conclusions

- We are able to build software for unreliable environments.
- Compared to existing routing protocols, our approach makes consequent use of the chemical paradigm on the code level.
  - We gain resilience to instruction loss and to most mutations.
  - The protocol is not forced to symbolically calculate statistics (e.g. link load, throughput, etc.). Molecule concentrations represent protocol states; packet rates are used to to exchange information.
  - The resulting traffic regulation is an emergent property of the “chemical” reaction network that spreads over the communication network.
- However, the proposed protocol only selects for low packet loss paths; paths with long delays are not punished. The RTE dissemination process does not scale well for large networks.
- For the future we need additional methods to design “chemical” protocols that anticipate the emergent effects of local changes to the global reaction network.

```
[fork nop BP11 match TRIG split match BP11 fork fork fork nop BP11 * split smatch fork RT EXP RT *
split DIV * DIF]
[fork nop BP31 match DIF split match BP31 fork fork fork nop BP31 * expel spush THIS snode _ srepl
send all inject nucleus split match RT fork RT RT * match RT expel snode _ send THIS sdup _ snode _
spush 0 syankdupat snode _ seq sif nul inject nucleus RT]
[fork nop BP41 match EXP split match BP41 fork fork fork nop BP41 * slength _ spush 7 seq sif EXPD
EXPF]
[fork nop BP51 match EXPD split match BP51 fork fork fork nop BP51 * shash _ EXPD1]
[fork nop BP52 match EXPD1 split match BP52 fork fork fork nop BP52 * expel spush BP sexch snewname
spush ID sexch srepl spush DEST sexch srepl fork nop ID match DEST releaseat SSS split match ID
fork fork fork nop ID * spush TAG spush SS spush S snewname srepl odeliver DEST TAG SSS]
[fork nop BP61 match EXPF split match BP61 fork fork fork nop BP61 * shoveat shoveat nul _ 2 spush
* EXPF1 1 * sexch release shash _ EXPF1]
[fork nop BP62 match EXPF1 split match BP62 fork fork fork nop BP62 * expel spush BP sexch snewname
spush ID sexch srepl spush NEXT sexch srepl spush DEST sexch srepl snode _ spush THIS sexch srepl
fork nop ID match DEST spush 1 ssum send NEXT DEST send THIS splitat SR splitat SLR SLR match ID
fork fork fork nop ID SR sdup _ snode _ spush 0 syankdupat snode _ seq sif nul]
```

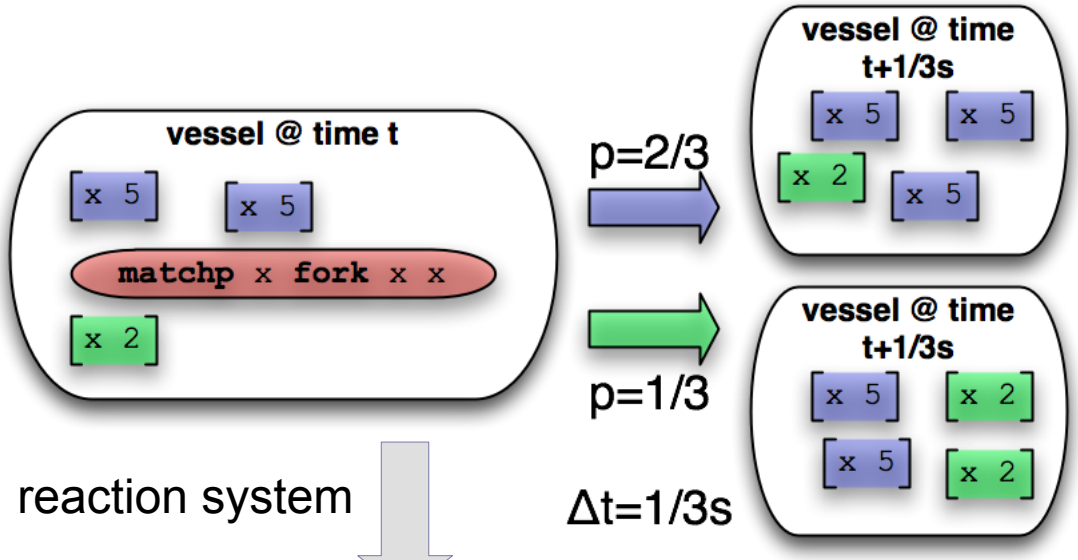
Fraglets program of the self-healing routing protocol

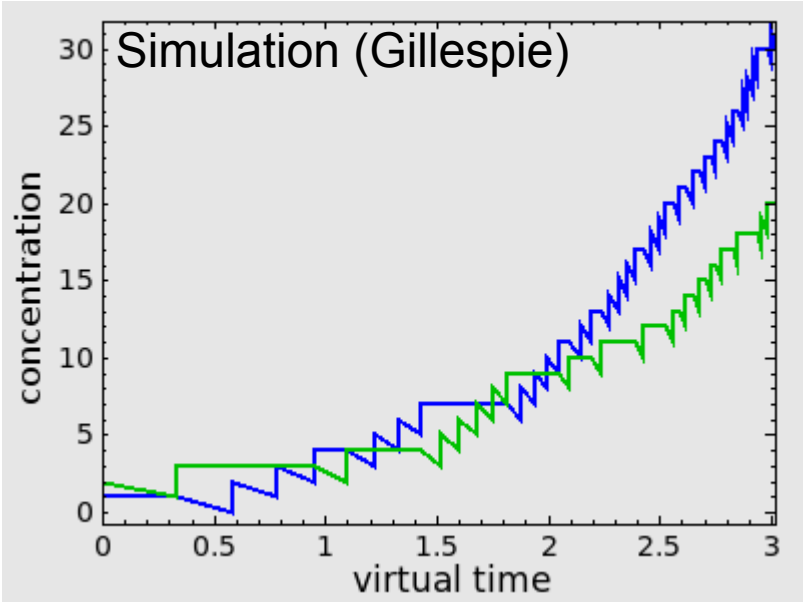
Thank you!

Questions?

Gillespie Algorithm

- Goal: Simulate the behavior of chemical reactions (ODE)
- Algorithm: **Exact Stochastic Simulation of Chemical Reactions** [Gillespie 1977]



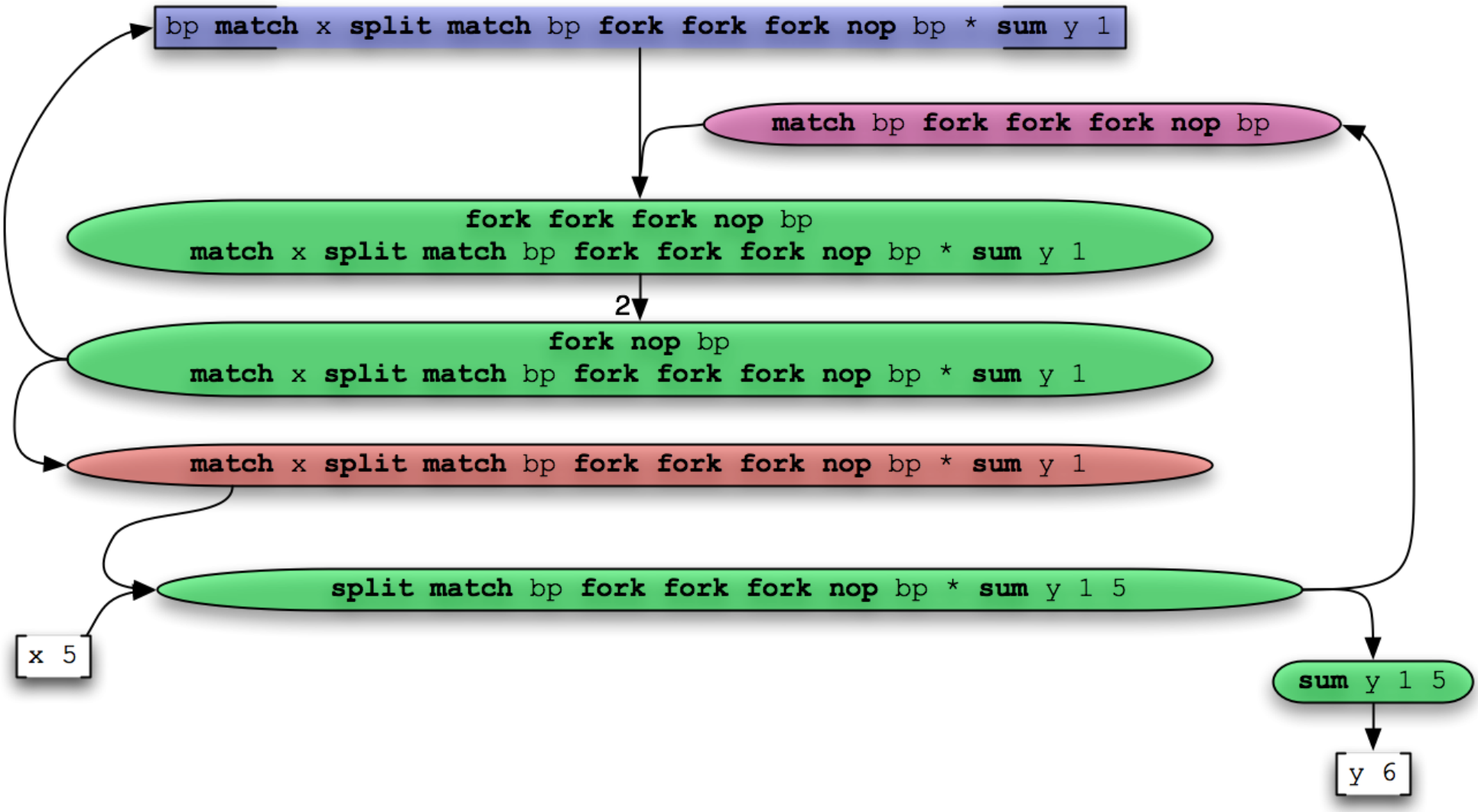


reaction system

$$\left| \begin{array}{l} X_5 + M_x \rightarrow 2X_0 + M_x \\ X_2 + M_x \rightarrow 2X_2 + M_x \end{array} \right|$$

$$\frac{d\vec{x}_5}{dt} = x_5 m_x \approx \frac{P(R_1)}{\Delta t} = \frac{\frac{x_5 m_x}{w_0}}{\frac{1}{w_0}}$$

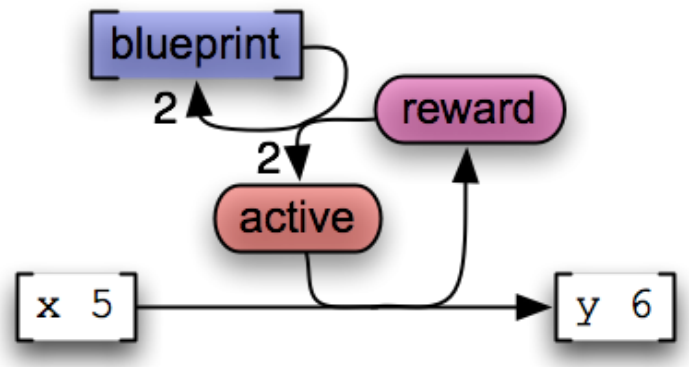
Autocatalytic Quine – Reaction Network

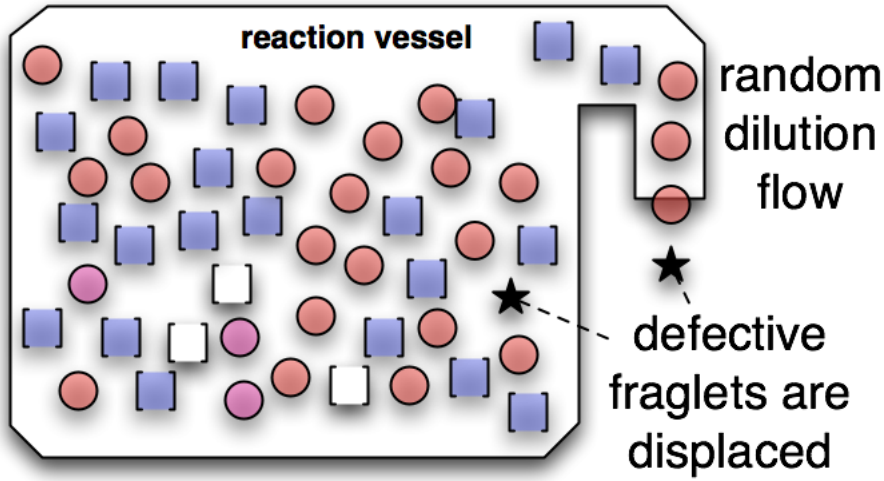


Selective Pressure

a) (exponential) growth:
duplication during replication

b) non-selective death:
randomly delete molecules to maintain a const. population



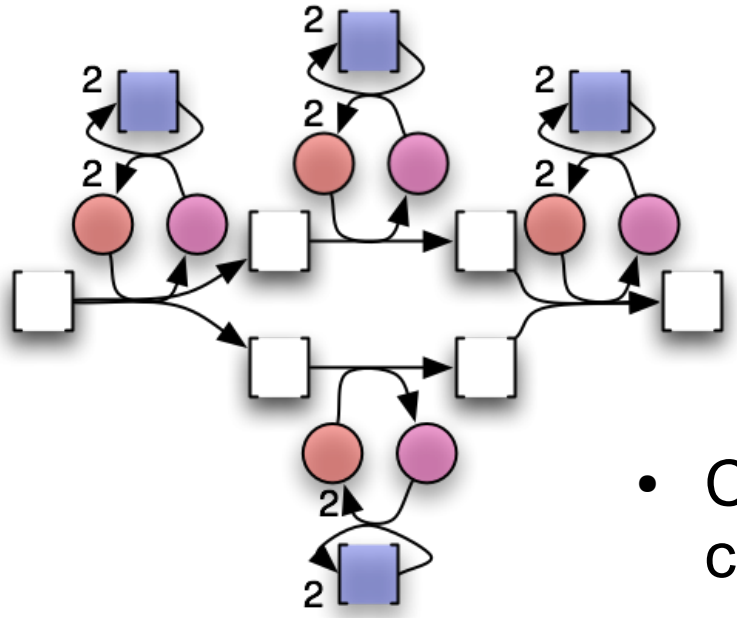


- Only autocatalytic sets survive.
- Defective, non-replicating code will be displaced.

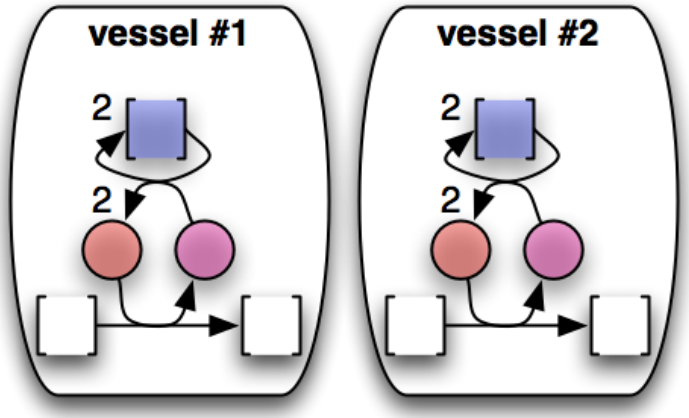
Cooperative *Quines*

- Independent *Quines* compete against each other
  ⇒ only one will survive.
- We build self-healing programs on cooperative *Quines* by using composition and compartmentation.

Composition

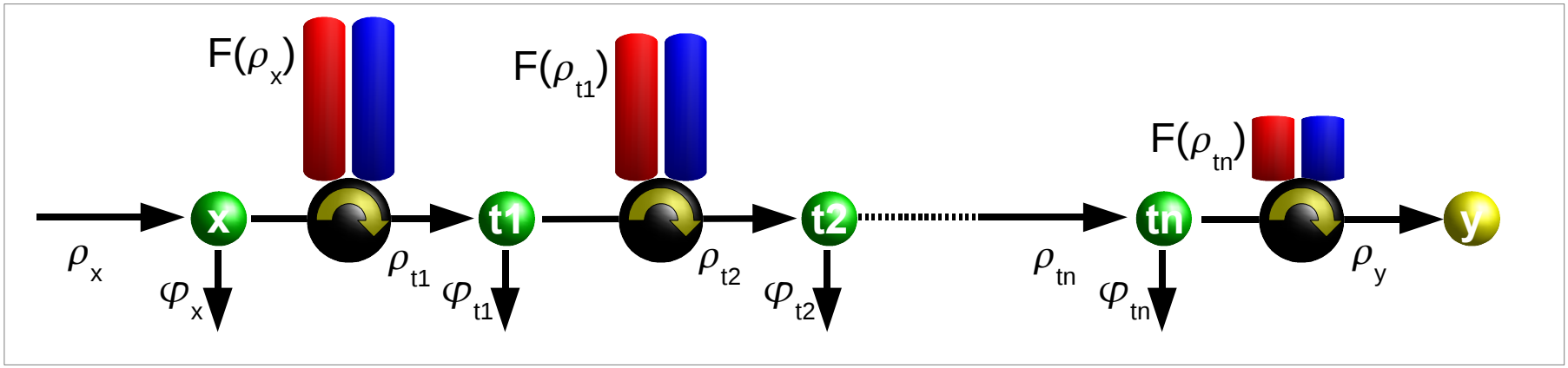


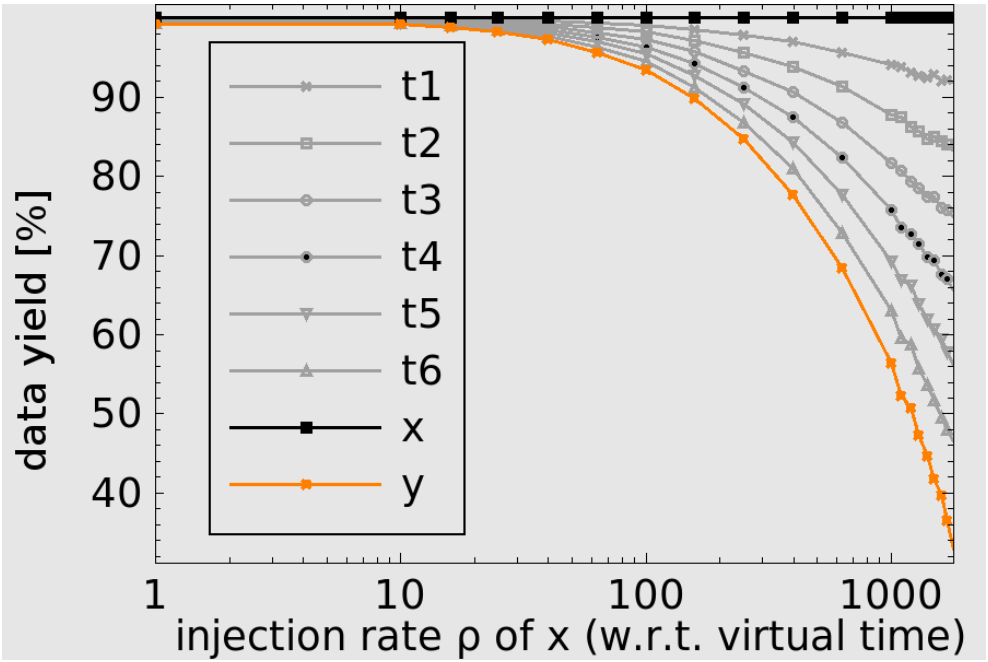
Compartmentation

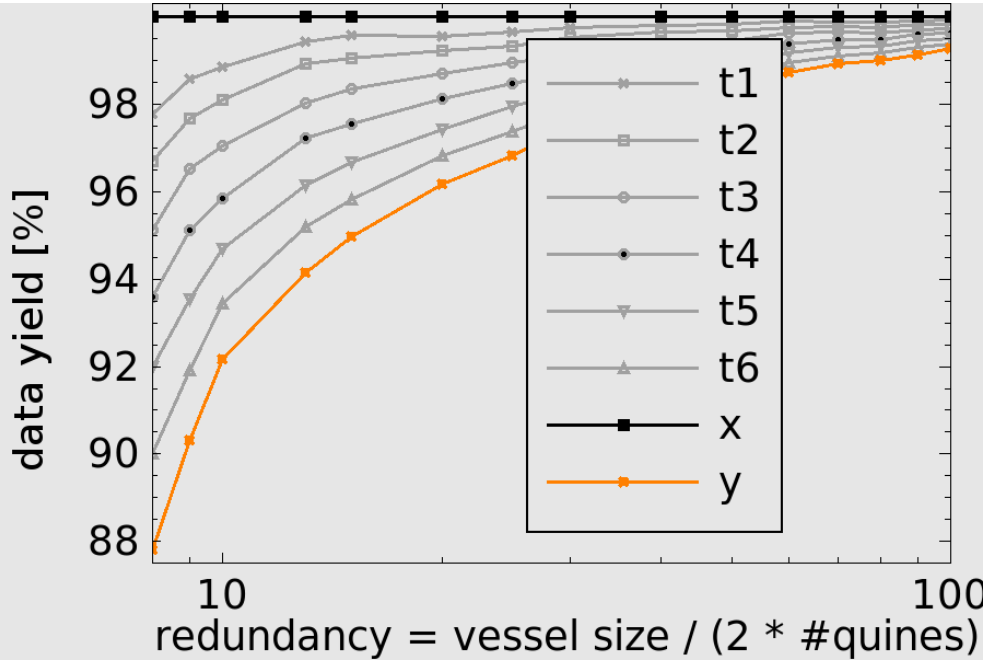


- Composition links *Quines by* a common data stream.
- Compartmentation simulates spatial separation.

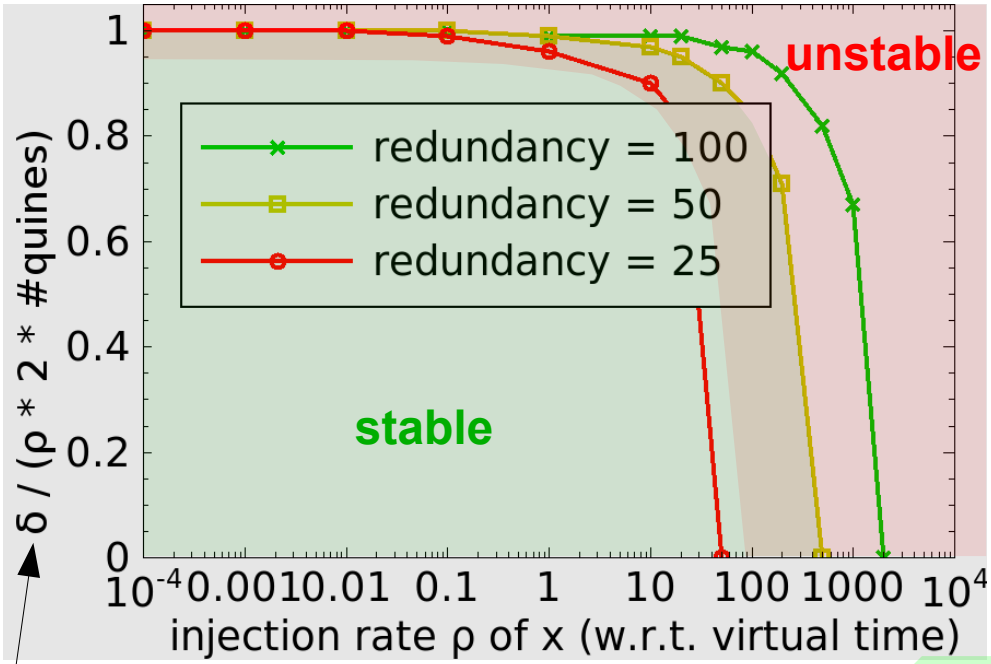
Data Yield in Serial Autocatalytic *Quines*



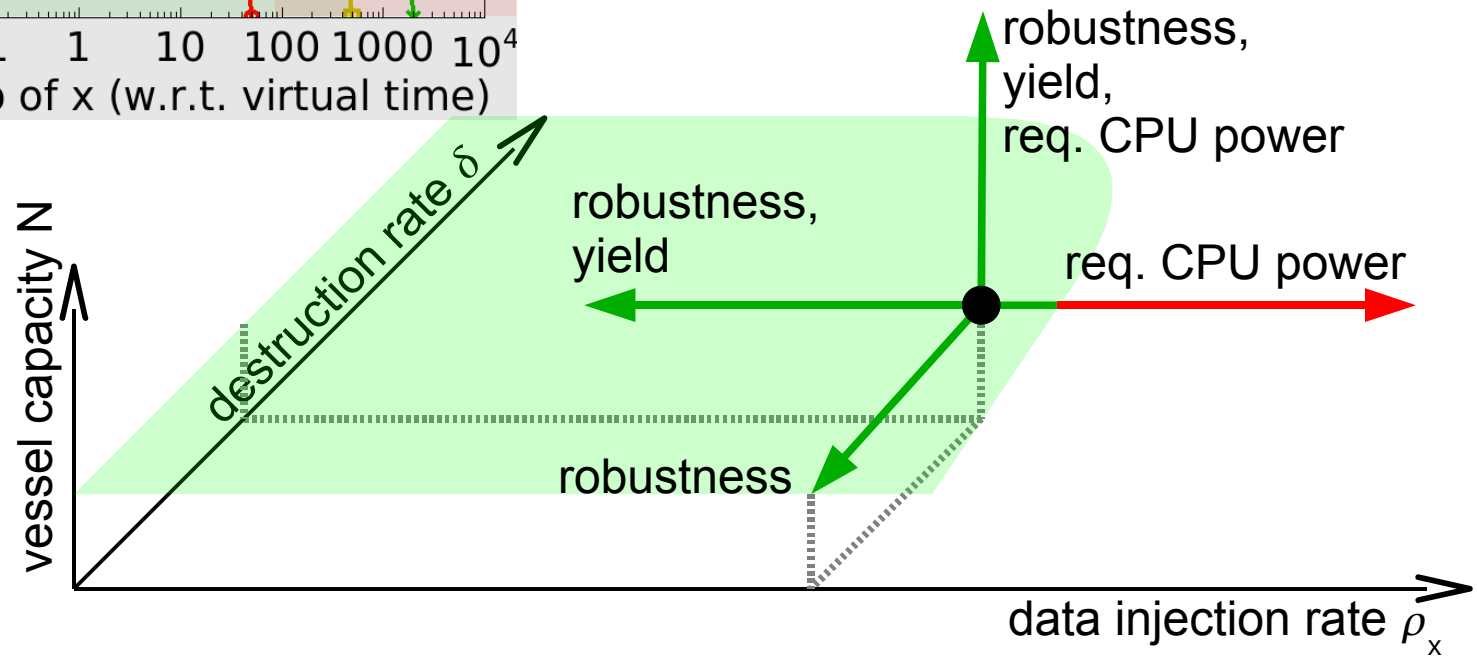




Robustness to Destruction Attacks

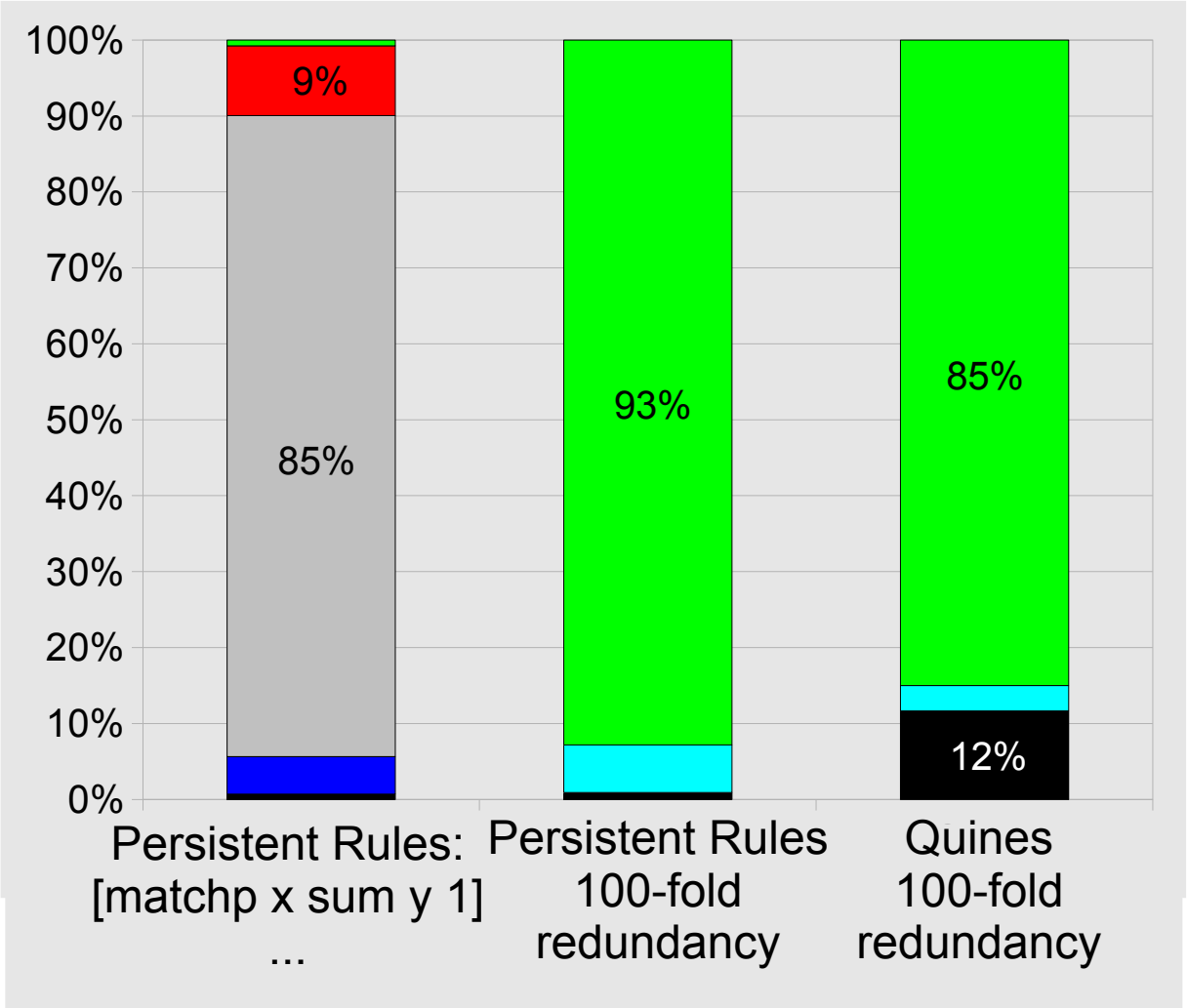


N = 100 / (2 * #quines)



Robustness to Mutation Attacks

- Robustness to a single mutation event (edit distance = 1)



Infinite closure
[matchp x x] + [x] → [matchp x x] + [x x]

Inert
no reaction for injected input data fraglet

No results

Invalid results (>95%)

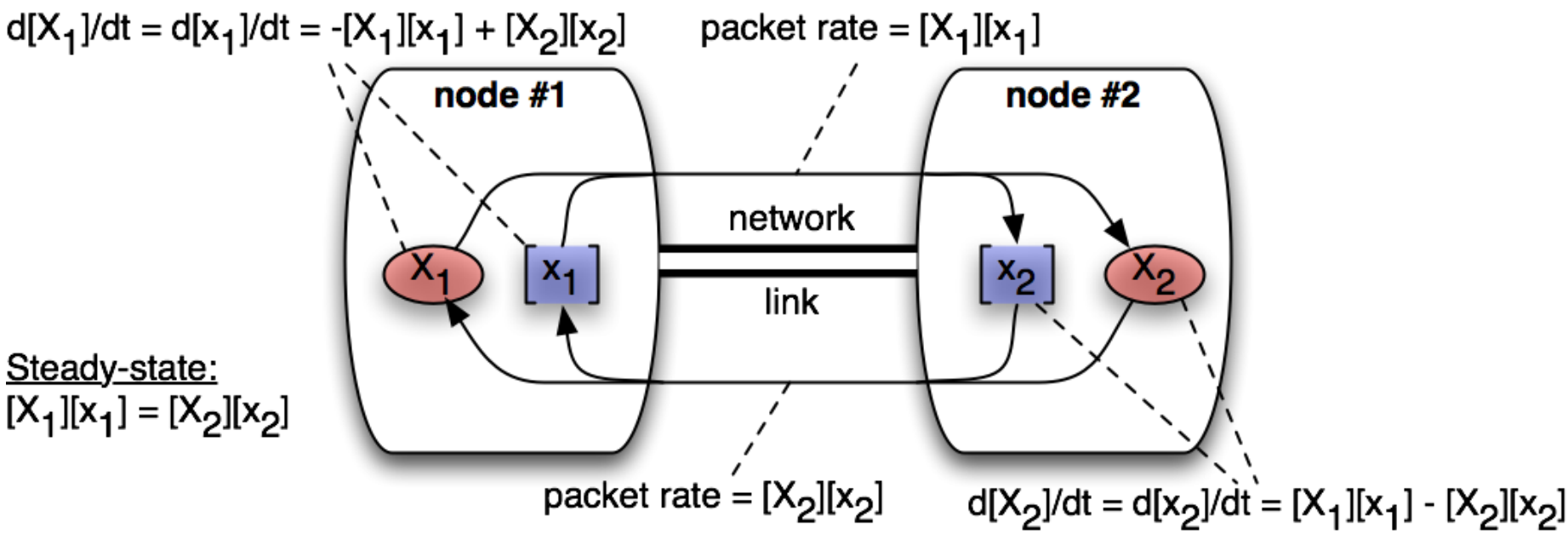
Invalid results (>50%)

Valid results (>50%)

Valid results (>95%)

“Chemical“ Communication Protocols

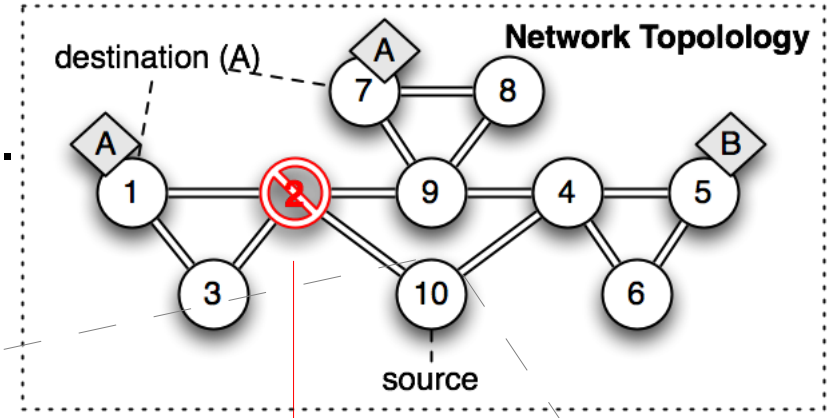
- The Fraglet instruction set allows for sending molecules (unicast `[send node2 ...data...]` and broadcast `[send all ...data...]`).

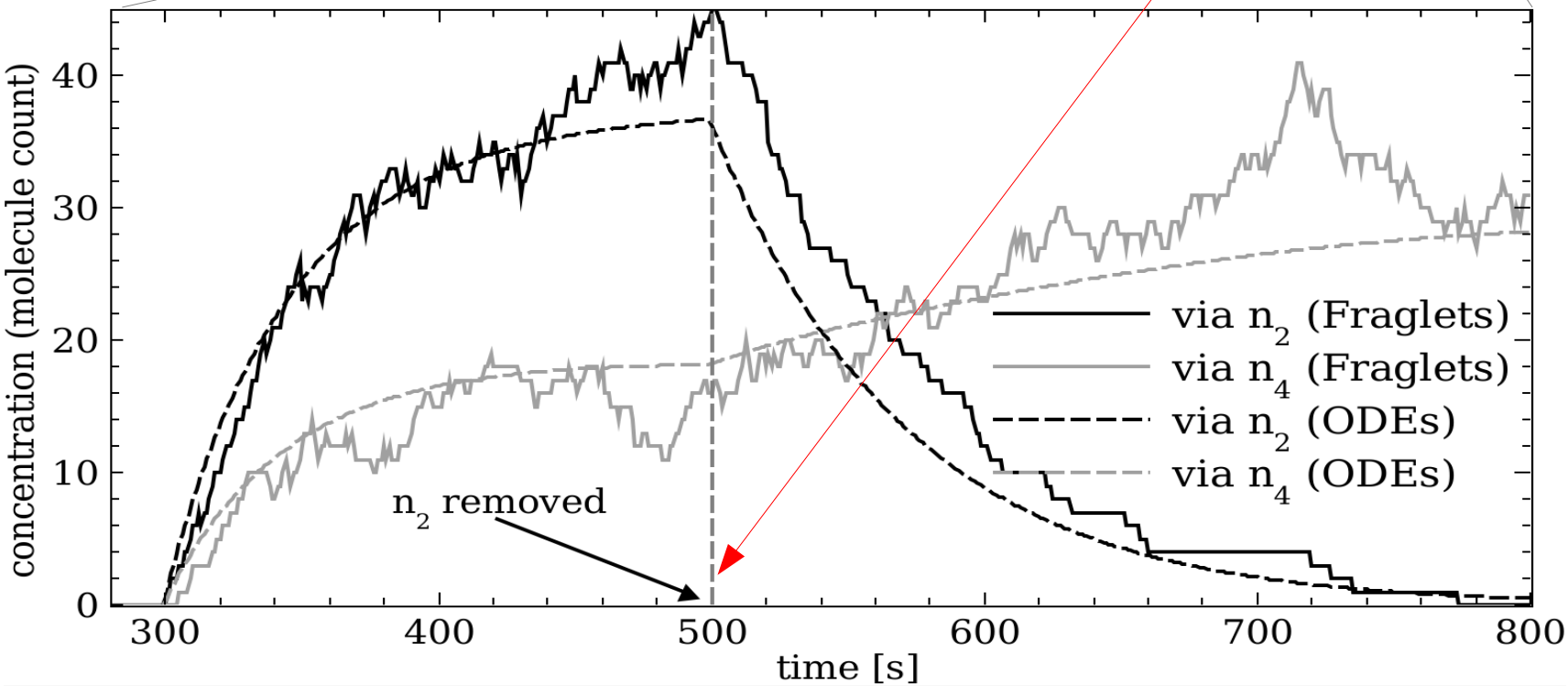


- Protocol **states** are encoded as molecule **concentrations**.
- **Information** is exchanged by means of packet **rates**. The packet rate is proportional to the reaction rate according to the Gillespie algorithm. [Gillespie 1977]

Routing Table Entry (RTE) Dissemination

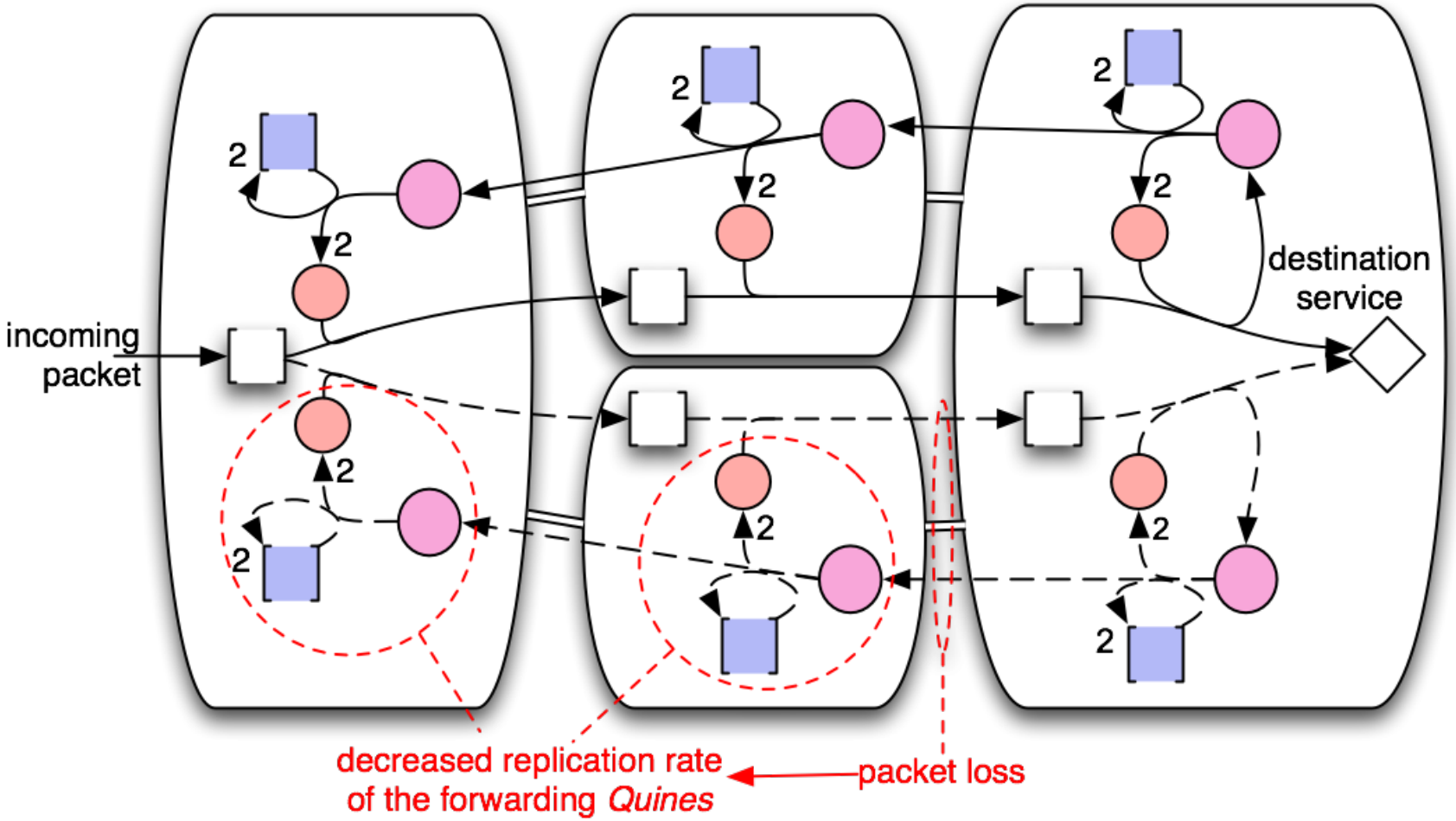
- The RTE concentration slowly adjusts to network topology changes.
- The stochastic process (random educt selection + Gillespie) is accurately approximated by a deterministic model described by ODEs.





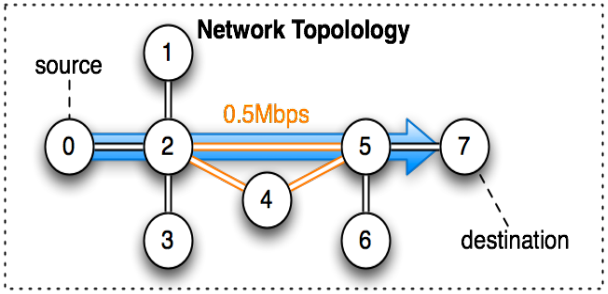
Forwarding Path Reinforcement

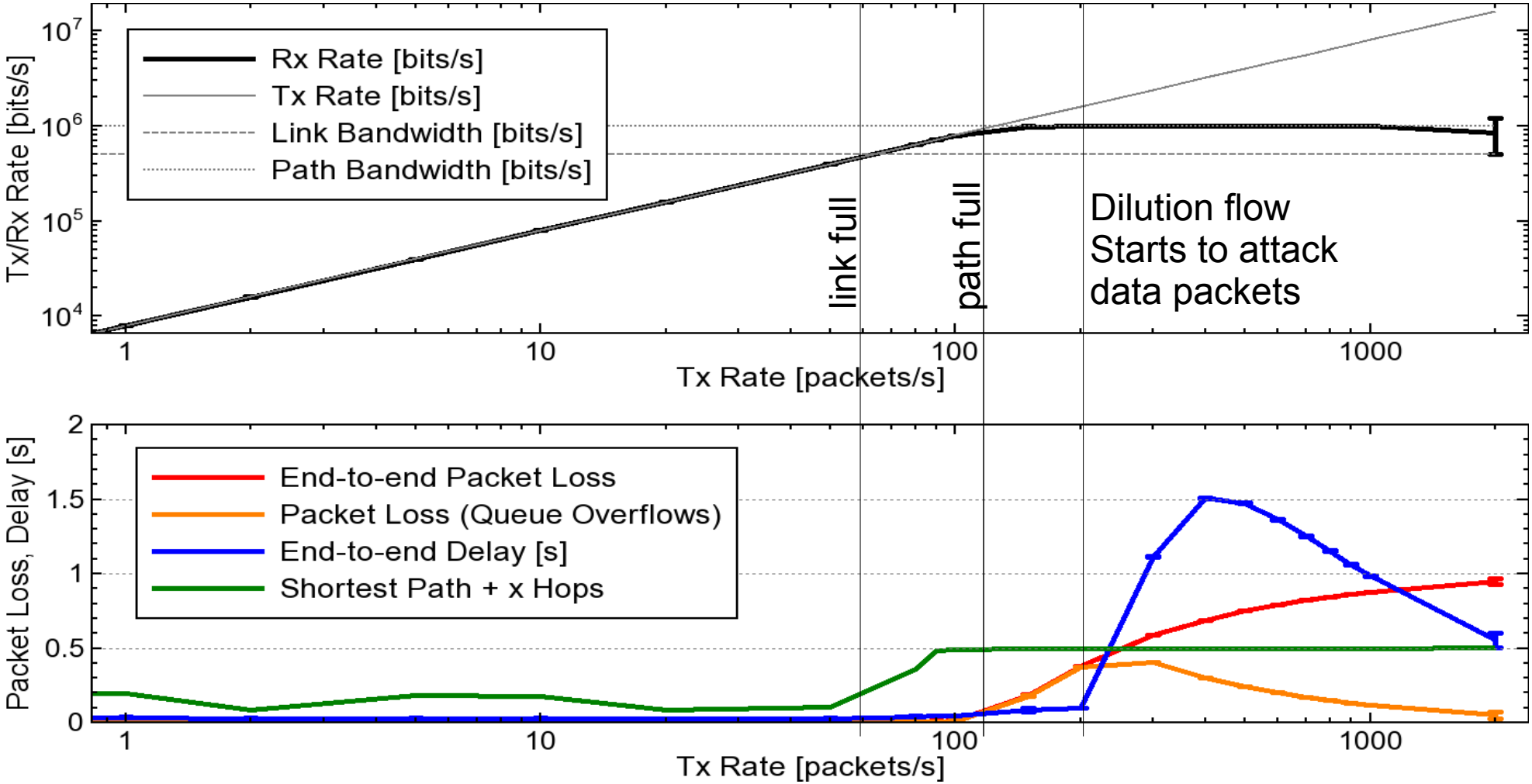
- Competition among forwarding rules yields a higher concentration of successful rules ⇒ higher reaction probability
- Coexistence of different forwarding paths is possible: multipath routing



Multipath

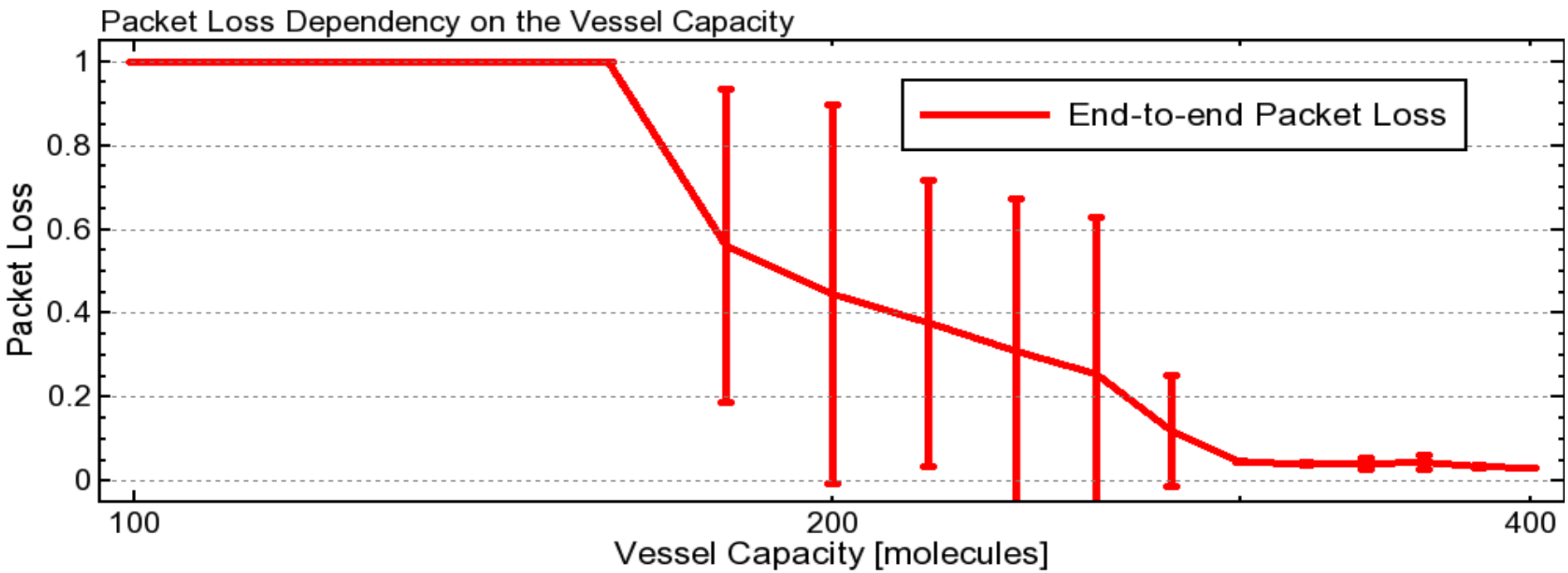
- The protocol exploits multiple paths
- For high packet rates packets are dropped by the dilution flow!





Packet Loss Caused by the Dilution Flow

- In vessels with low capacity the incoming data packet stream may displace forwarding rules faster than they can be regenerated



- The vessel capacity (external parameter) correlates with the node's CPU speed (more molecules ⇒ higher reaction rates)

- The resulting packet loss allows neighbor nodes to deviate traffic around heavily loaded nodes