Only those who attempt the absurd
will achieve the impossible.
I think it is in my basement...
let me go upstairs and check.

M.C. Escher (1898–1972)

On Chemical and Self-Healing Networking Protocols

PhD Thesis Presentation

Thomas Meyer
Computer Science Department, University of Basel, Switzerland

December 17th, 2010

This dissertation on chemical and self-healing networking protocols combines two so far separate research fields: chemistry and computer networks.

In the next 30 minutes, I will demonstrate how to translate chemical principles to computer networks.

Chemical networking protocols are easy to handle, their dynamics can be analyzed, and they have properties that are hard to achieve otherwise:

We show, for example, that protocols are able to heal themselves on the code-level.

The Internet — From Functional to Dynamic Challenges

Seemingly perfect packet delivery service:

- **TCP/IP** — addressing, reliable transmission
- **Routing** — autonomic adaption to topology changes

But very complex dynamical system:

- **Complexity** — millions of computers / applications; annual growth rate: 30-40%
- **Protocol Dynamics** — barely understood / side-effect; labor cost = 18 x equipment cost

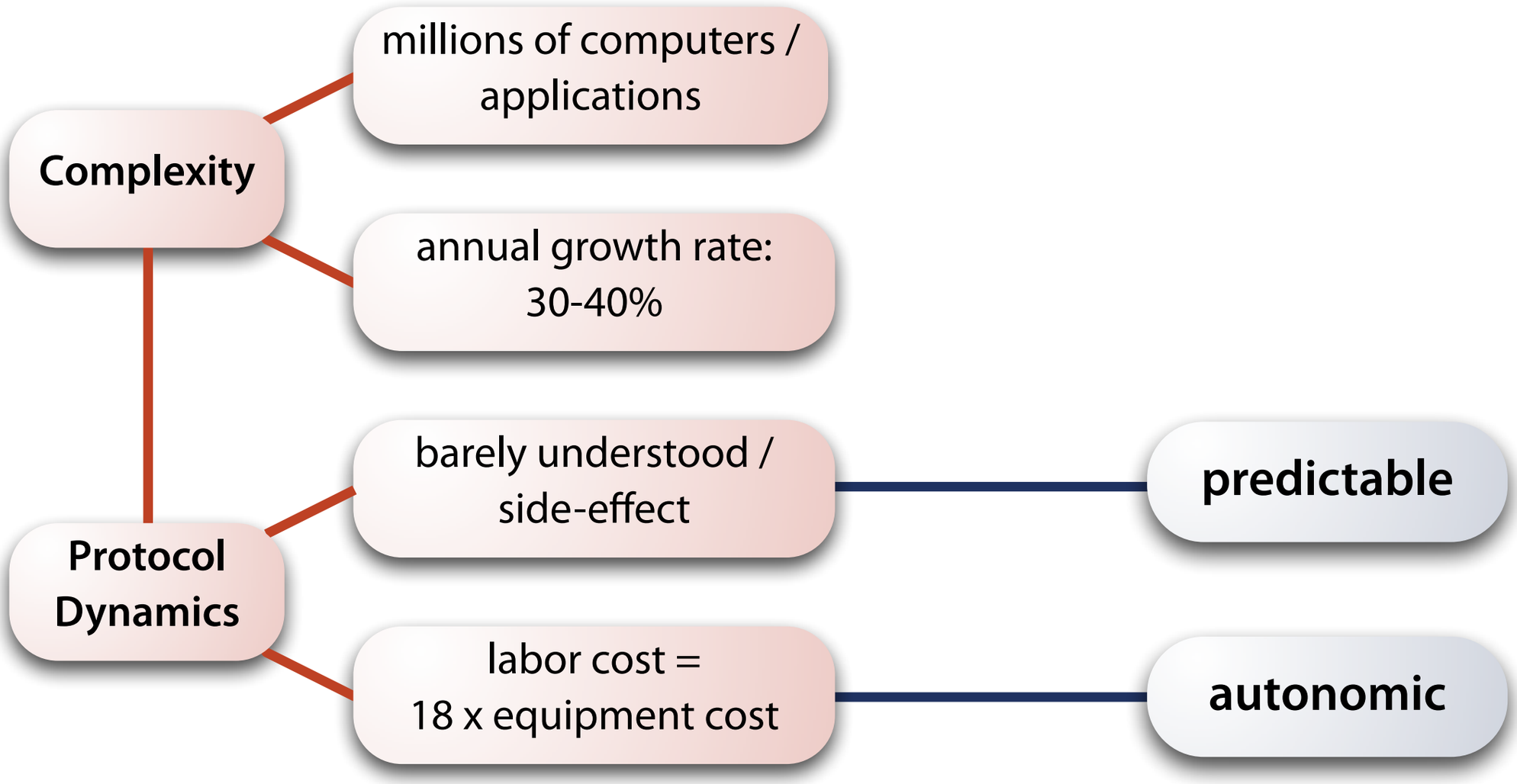
The Internet provides a seemingly perfect service. Packets are delivered reliably and on time. Mainly responsible for the big success of the Internet are the **workhorse protocols TCP and IP**; IP providing a global addressing scheme at the network layer and TCP providing reliable data transmission. Additionally, **routing protocols** make sure that packets find their way through the network. They autonomically adapt to topological changes of the network.

However, the Internet is becoming a very **complex dynamical system**: Millions of computers and applications are connected to each other. The data traffic grows with an annual rate of 30-40 percent, and even up to 100% for traffic generated by mobile devices. **The dynamics of some protocols is barely understood.** Often, the dynamic behavior of protocols is treated as a side-effect and is only looked after if there are problems while competing with other protocols. Furthermore, **the network is far from being autonomic**. The labor cost for maintaining the IT infrastructure is 18 times higher than the equipment cost.

The Internet — From Functional to Dynamic Challenges

But very complex dynamical system:

Desired Properties for the future network:

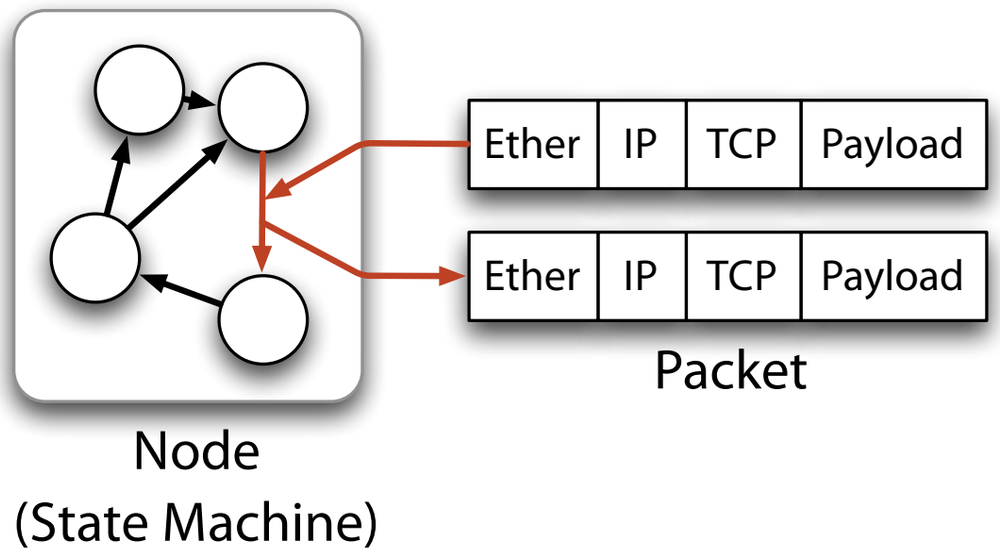


In the future we want the network to be **more predictable**, we want to understand the protocols' dynamics better.

At the same time, IT departments want networks to be **more autonomic**, reacting automatically to perturbations and faults - actually without losing reliability and also predictability.

Computer Networking — Micro/Macro Levels

Microscopic Level (functional)

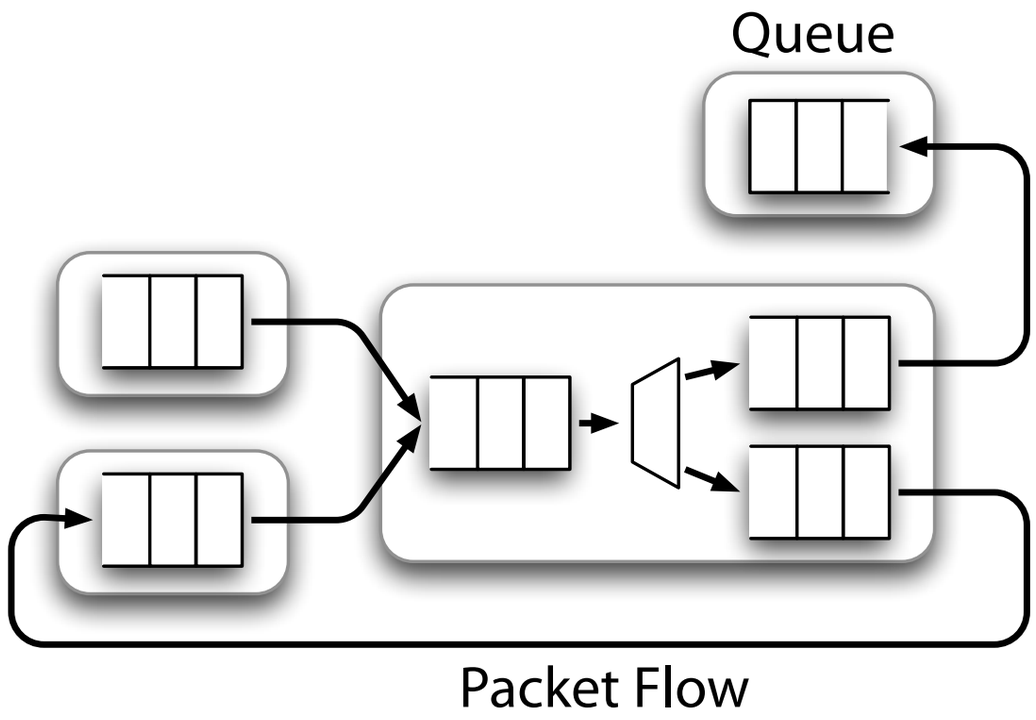


I wish to introduce you now to two levels of granularity at which we can look at protocols and the packets they exchange: a microscopic and a macroscopic level.

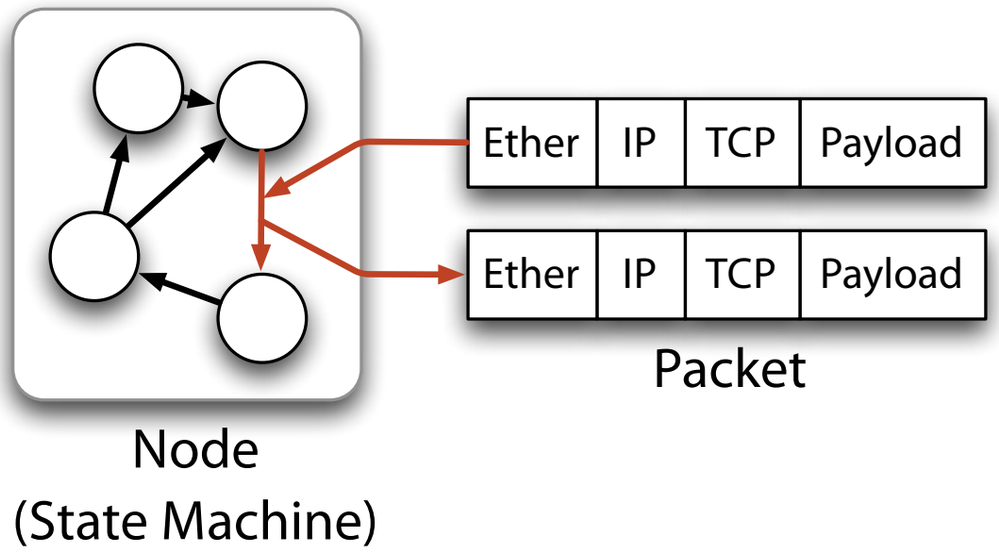
At the **microscopic level** a protocol implementation cares about the individual packets, the format of their header, the actions triggered by a received packet, timers that have to be set, and so on. Protocols are often implemented as state-machines where an asynchronous event such as an arriving packet triggers an action and a state transition.

Computer Networking — Micro/Macro Levels

Macroscopic Level (dynamical)



Microscopic Level (functional)



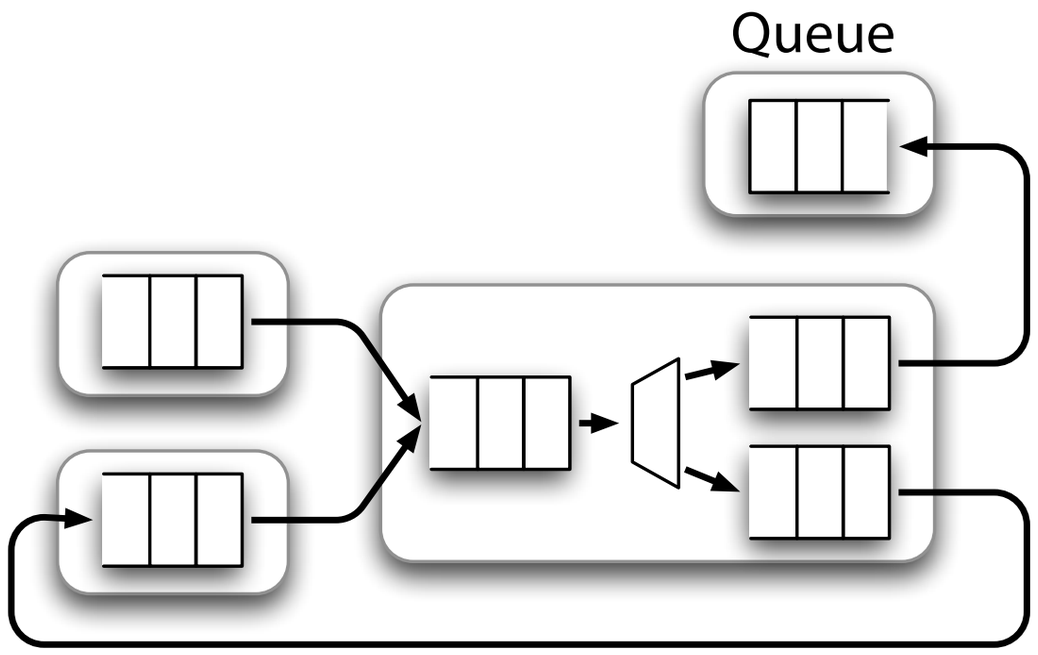
At the **macroscopic level**, we look at packet streams or packet flows. The content of an individual packet is not important anymore.

We focus on the large-scale dynamic behavior of a protocol. For example we can model the computer network as a network of queues, whose dynamic behavior can be analyzed by queueing theory or network calculus, for example.

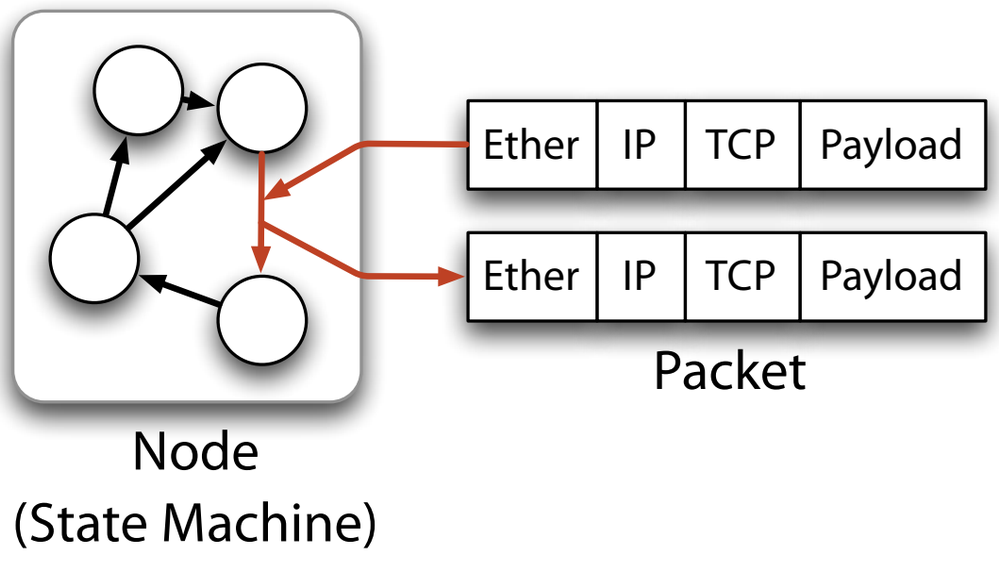
Computer Networking — Micro/Macro Levels

No theory to combine microscopic execution and macroscopic analysis

Macroscopic Level (dynamical)



Microscopic Level (functional)



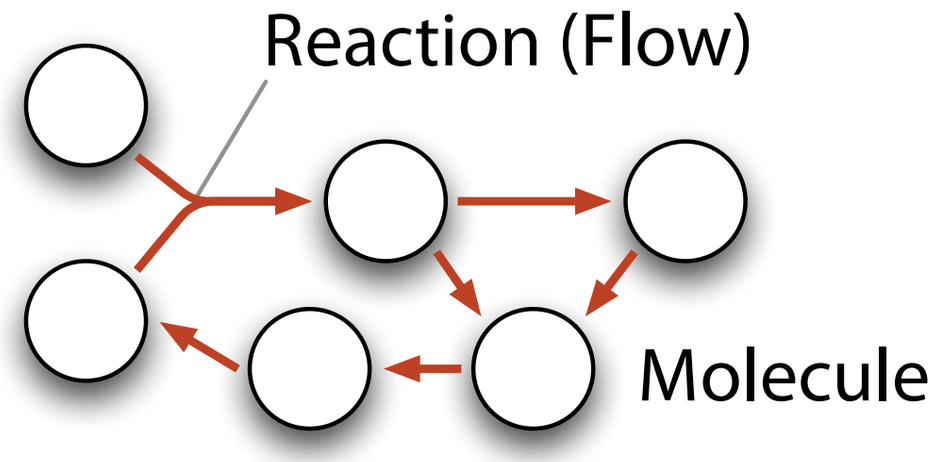
Currently, there is **no unifying theory** that combines the two levels.

It is quite hard to analyze or even proof the dynamic behavior of a protocol state machine.

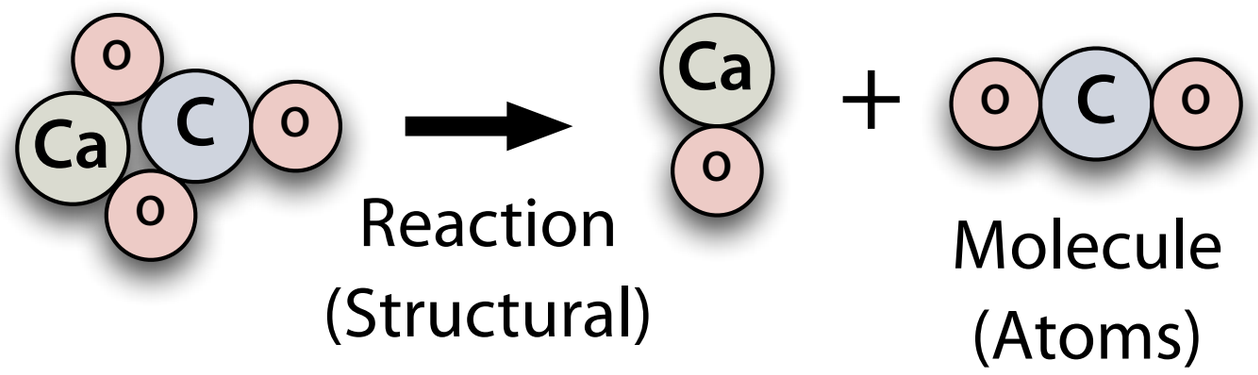
And there are no methods to break down a desired behavior of on the macroscopic flow-level to a discrete implementation.

Chemistry — Same Micro/Micro Levels

Macroscopic Level (dynamical)



Microscopic Level (functional)



Interestingly, in chemistry, we find the same microscopic and macroscopic levels.

At the **microscopic/functional level**, molecules collide and react according to their inner structure, their composition of atoms and their bonds.

On the **macroscopic level**, researchers are not interested in the structure of the molecules anymore. They rather treat them as abstract species and analyze the reaction networks that are spanned between them. This level focuses on the dynamics of the reaction flows.

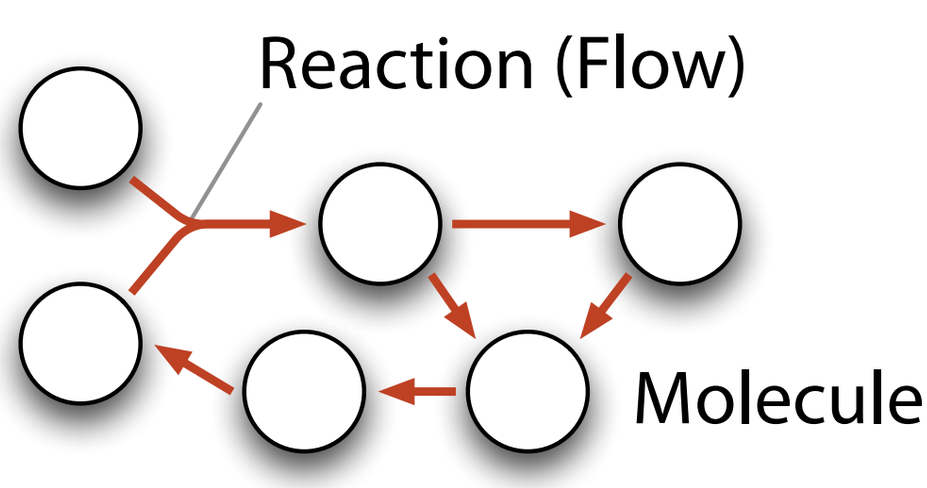
Chemistry — Same Micro/Micro Levels

Nature: Emergence of life-like system level properties

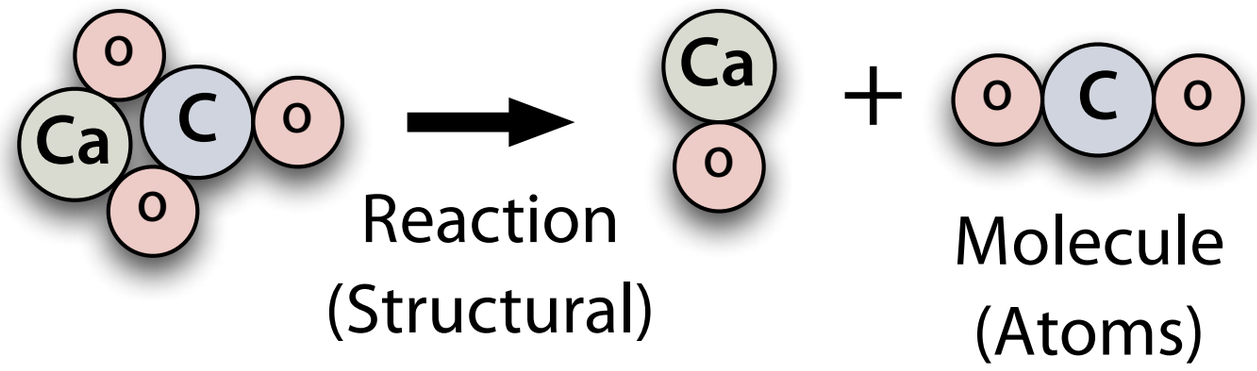
System Level

self-organization, self-healing, self-*

Macroscopic Level (dynamical)



Microscopic Level (functional)



On an even higher, **system level**, we see the emergence of self-organization, self-healing and other self-* properties.

These are the properties we would like to transpose from chemistry to networking.

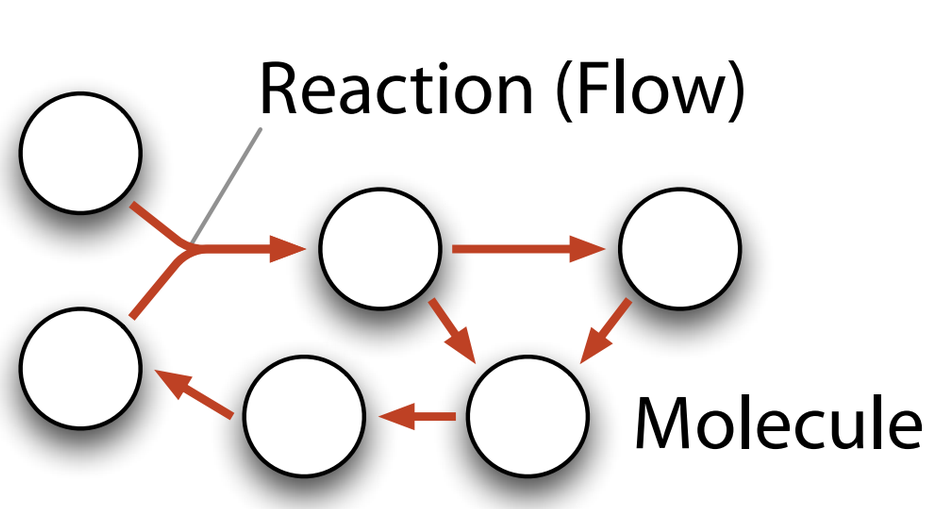
Chemistry — Same Micro/Micro Levels

System level properties can be explained by microscopic interactions

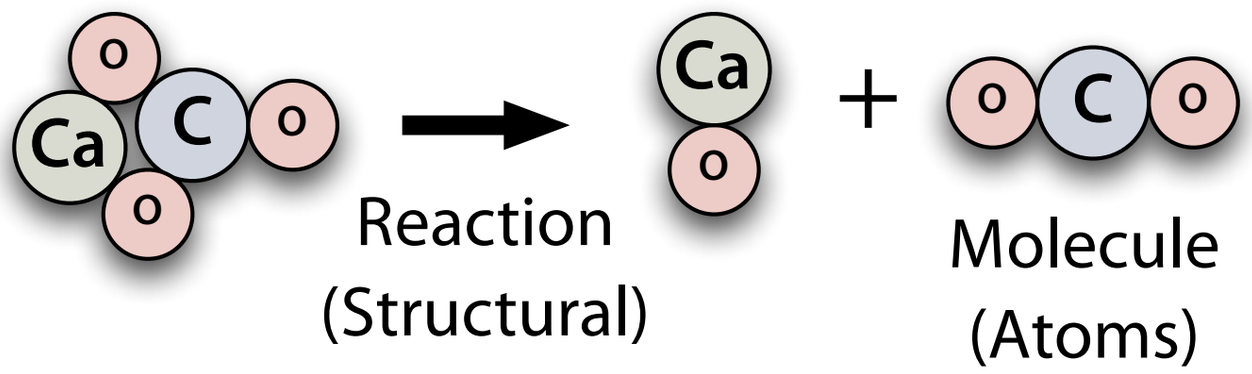
System Level

self-organization, self-healing, self-*

Macroscopic Level (dynamical)



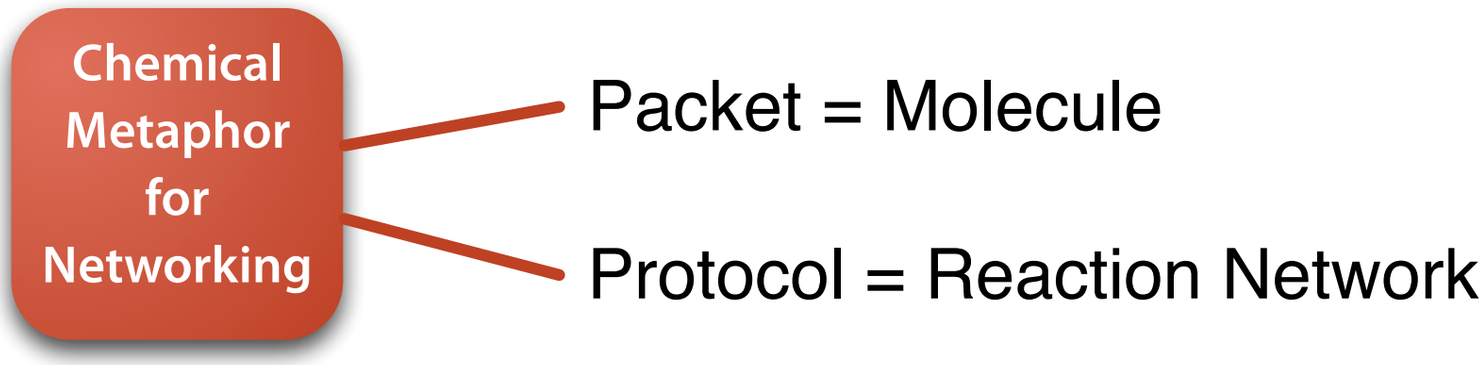
Microscopic Level (functional)



The layers in chemistry are much **more transparent** than in current networking:

We can predict the topology of the reaction network by analyzing the functional aspects of molecular collisions, and with the Chemical Organization Theory developed by Peter Dittrich, we can predict the high-level organization emerging from a reaction network.

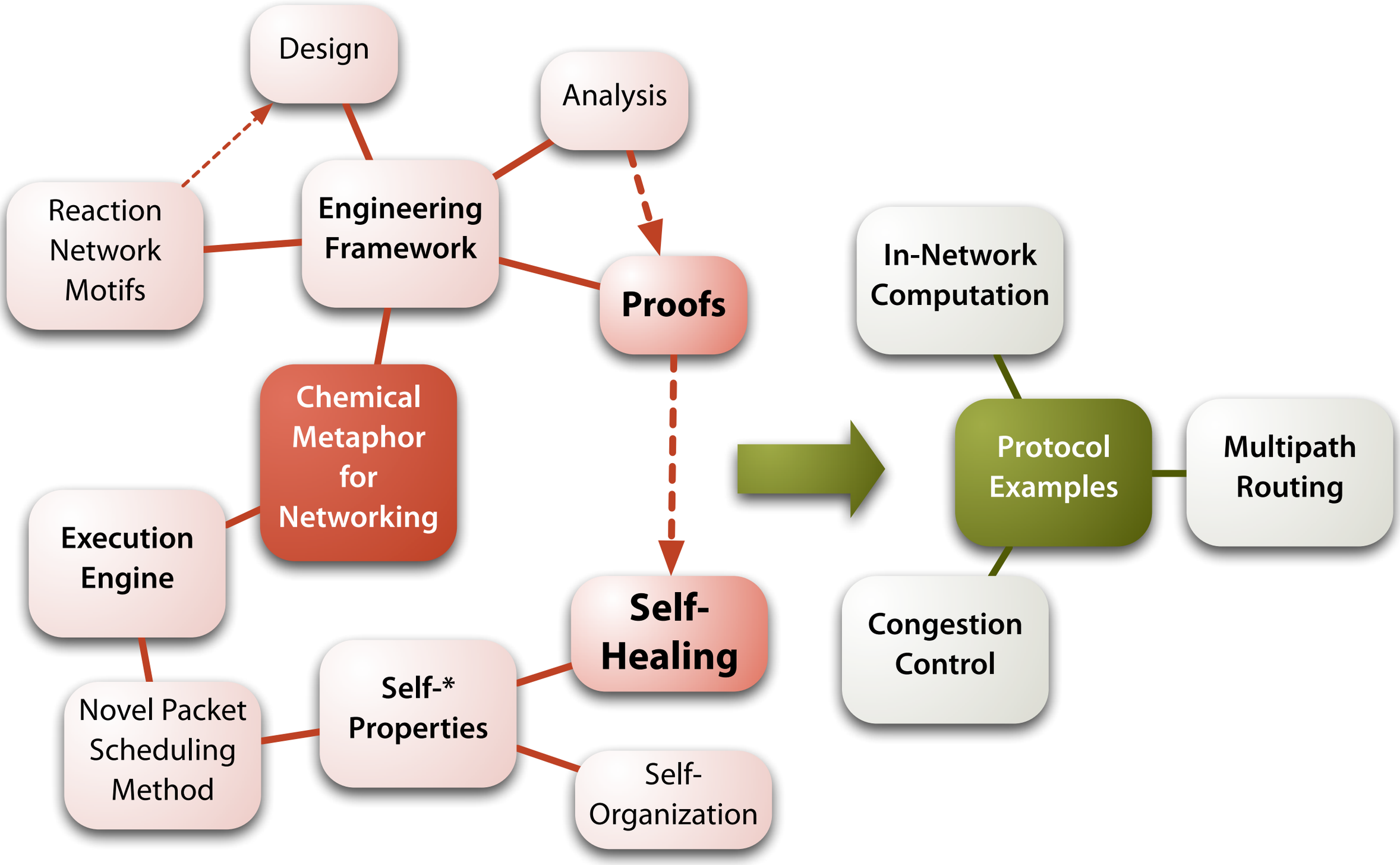
Main Contributions of this Thesis



The core of our approach is to exploit the **chemical metaphor** to solve networking problems. The main idea is to represent **packets as molecules** and to see **networking protocols** as a web of packet collisions akin to **chemical reactions**.

But we need much more than a metaphor to reliably engineer protocols.
Unlike recent bio-inspired approaches that mimic biology to solve a specific problem, we aim for a more general model that enables scientists to quickly come up with solutions to a vast range of networking problems.

Main Contributions of this Thesis



The core of our approach is to exploit the **chemical metaphor** to solve networking problems. The main idea is to represent **packets as molecules** and to see **networking protocols** as a web of packet collisions akin to **chemical reactions**.

But we need much more than a metaphor to reliably engineer protocols.
Unlike recent bio-inspired approaches that mimic biology to solve a specific problem, we aim for a more general model that enables scientists to quickly come up with solutions to a vast range of networking problems.

We developed an **engineering framework** to design and analyze chemical networking protocols.

To assist the design process, we provide a set of **reaction network motifs**, which are small well-understood reaction networks that can be assembled in order to synthesize protocols.

These protocols can then be **analyzed** by methods and tools that we transpose from chemistry to our chemical protocols. It is even possible to come up with **simple convergence proofs**, for example.

On the lower layer, we provide an **chemical virtual machine** to execute chemical protocol software with a **novel packet scheduling method**, borrowed from chemical reaction kinetics. This enables software to exhibit **self-* properties** on the system level such as **self-organization** of software part or **self-healing** of protocol code. We can even use our analysis methods to **quantify the robustness** of the code.

This theoretical work is complemented by a series of **examples and application cases** ranging from in-network computation over a chemical congestion control algorithm to a self-healing multipath routing protocol.

Outline

Motivation

Chemical Networking Protocols

- Introductory Example — The Chemical Metaphor
- Chemical Protocol Engineering Framework
- Application Case: C3A — A Chemical Congestion Control Algorithm

Self-Healing Protocols

- Robustness to Code Deletion
- Application Case: Self-Healing Link-Load Balancing Protocol

Conclusions

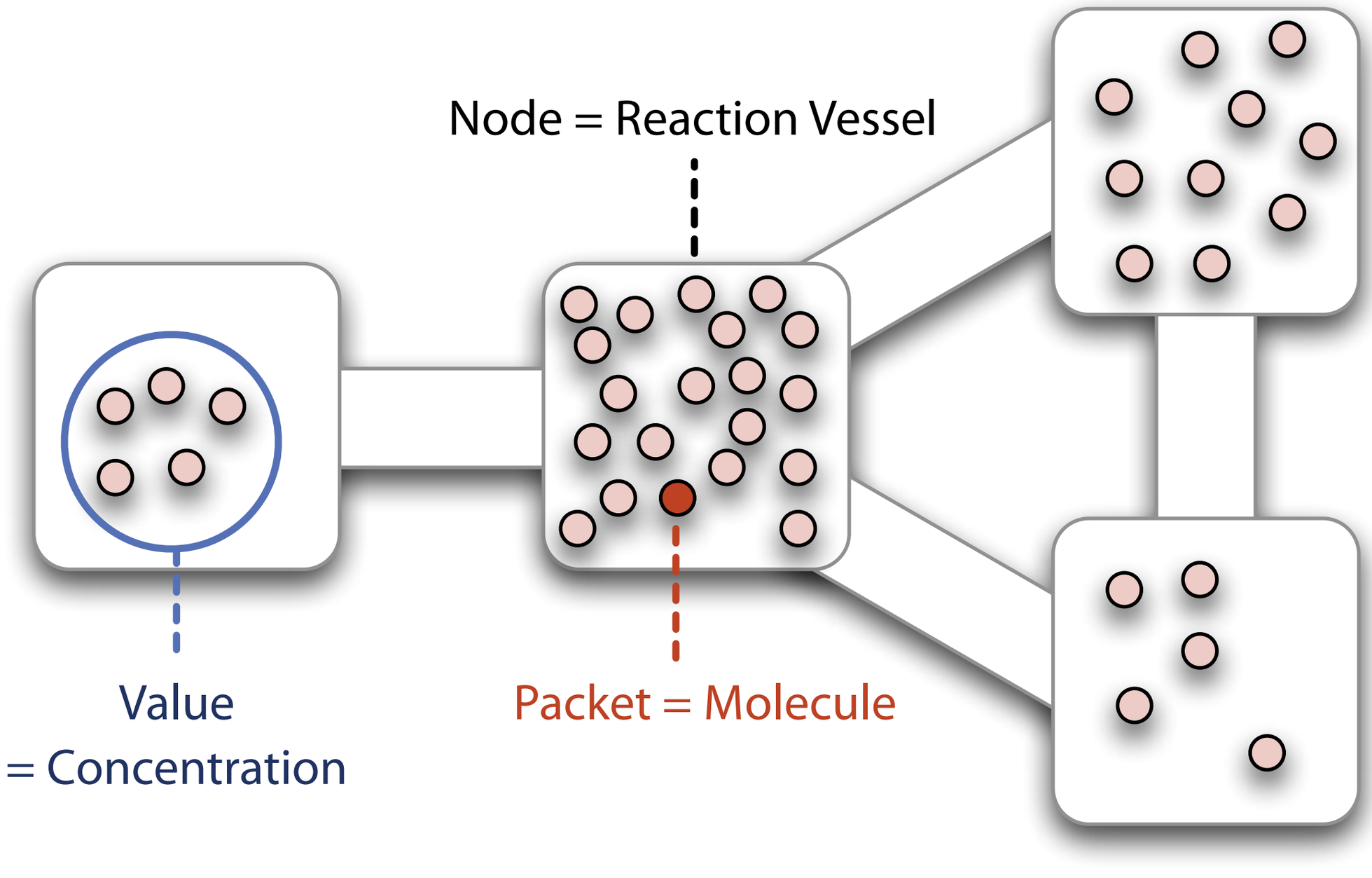
I will structure my talk into two main parts:

The first part on **chemical networking protocols** starts with an example to introduce the chemical metaphor for networking applications.
- I will then outline the proposed chemically inspired protocol engineering framework to design, analyze and execute protocol software.
- After this, I will demonstrate that chemical protocols are able to cooperate with current protocols in the Internet by introducing a chemical congestion control algorithm as an application case.

In the second part on **self-healing protocols** I will focus on a higher level protocol property, on how protocol software can be made robust to random code deletion.

Introductory Example — The Chemical Metaphor

State information is represented by the concentration of molecules



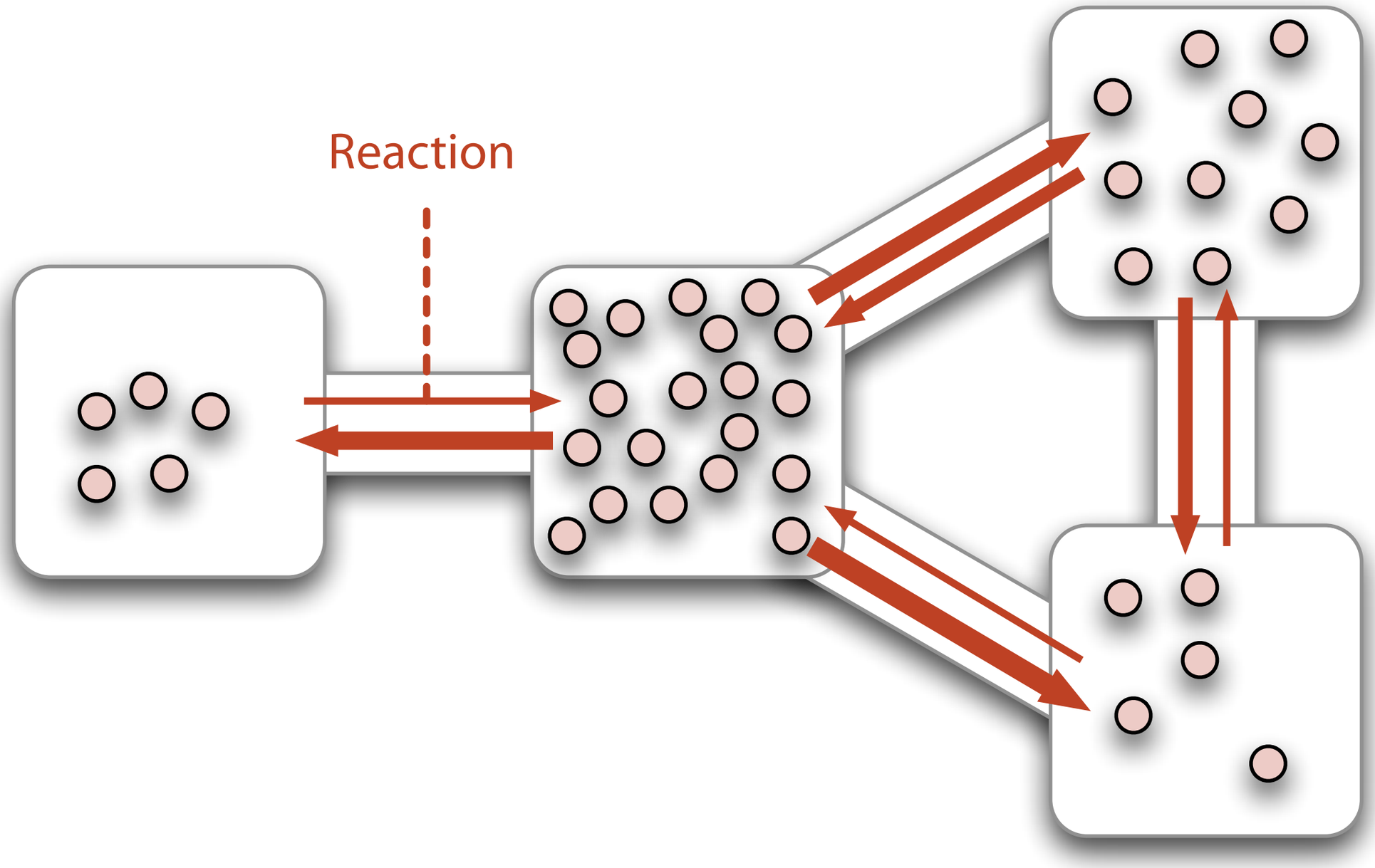
Let us start with a simple example to illustrate the main idea behind a chemical networking protocol.
Our network consists of 4 connected node.
Each node is a virtual reaction vessel hosting a given number of packets, or molecules.

Let's assume each node has a sensor that measures an environmental variable such as the temperature, the humidity or the light intensity. This measured value is now represented by the concentration, that is the quantity of molecules in that vessel. That is, unlike in traditional approaches, where information is encoded symbolically as a binary number, we encode state information as the multiplicity of elements.

We want the network to aggregate this information, for example to compute the average value, e.g. the average temperature in the network.

Introductory Example — The Chemical Metaphor

Reactions may cross vessel boundary; rate proportional to concentration



We may start with an arbitrary distribution...

For this purpose, for each node and link we install a reaction that randomly picks a molecule in one vessel and sends it to a neighbor vessel. So in this example, the reactions represent packet streams across the network.

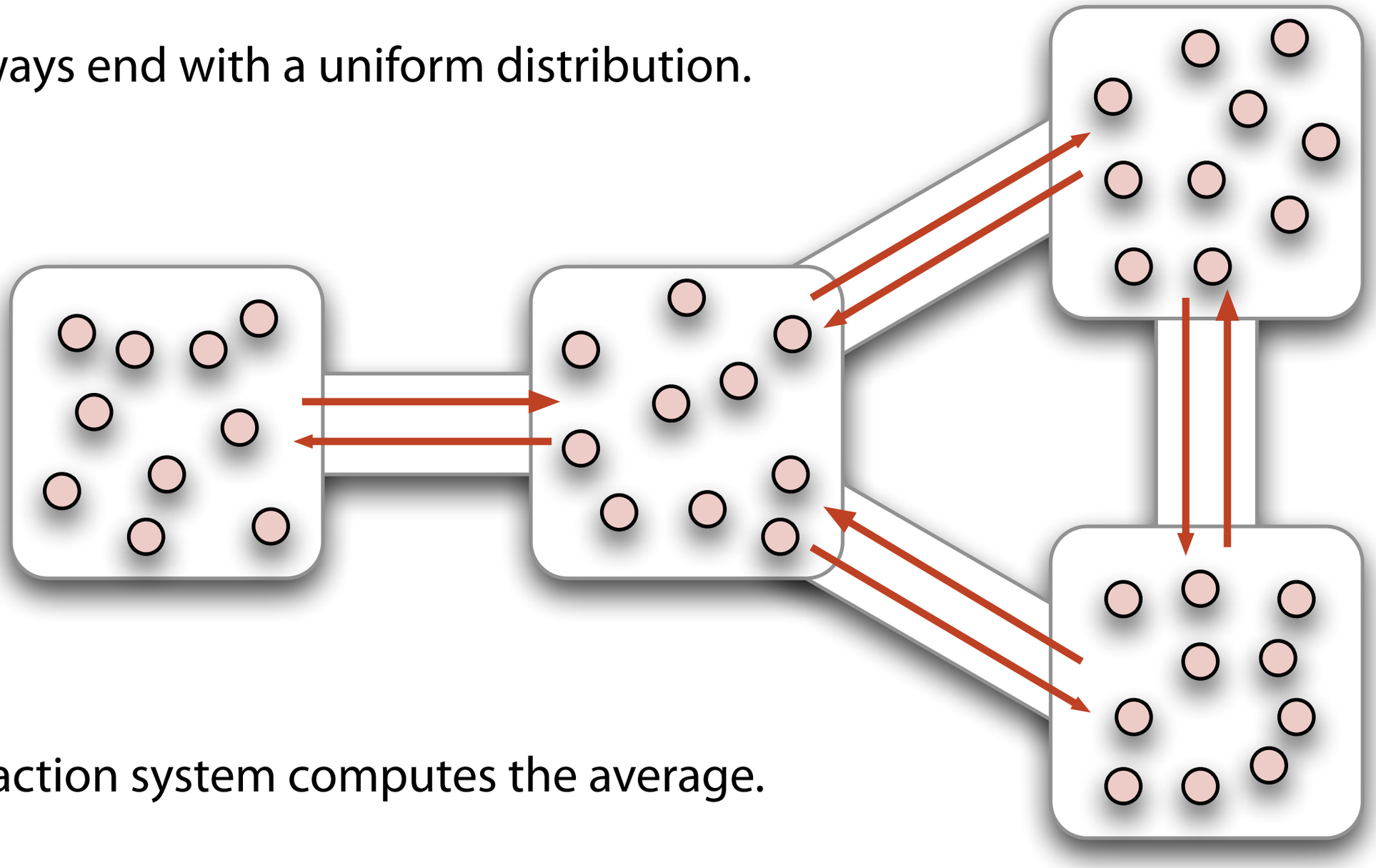
In chemistry, the reaction rates are proportional to the number of reactant molecules. That is, the more molecules there are in a vessel, the faster they are sent to the neighbors.

This means that with whatever molecule distribution we start in the beginning...

Introductory Example — The Chemical Metaphor

The molecules are dispersed over the network

We always end with a uniform distribution.



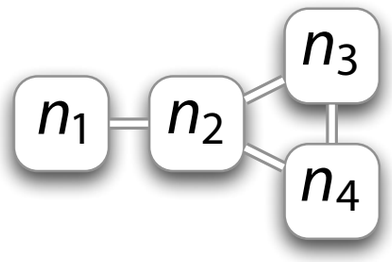
The reaction system computes the average.

...we always obtain a uniform distribution where each node contains the average number of molecules – the network actually computed the average temperature, for example.

An Artificial Chemistry for Networking

Formal Definition of the *Disperser* Protocol

Network Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

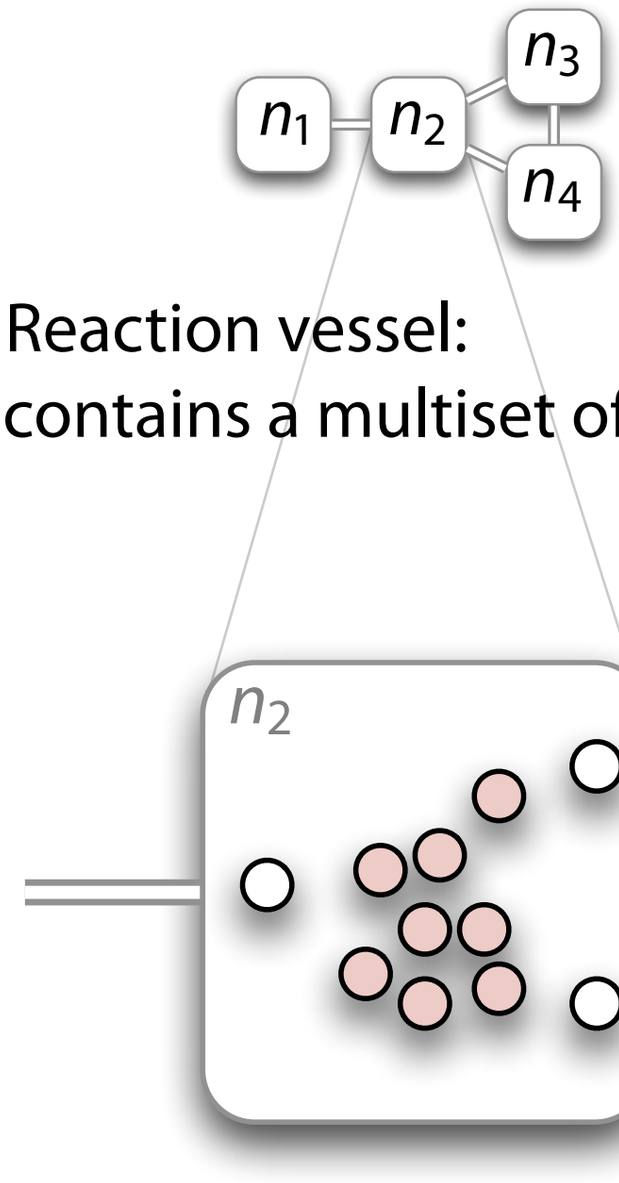


Let me now formally define this simple Disperser protocol as a distributed artificial chemistry: First, we define the network graph in a standard way, as set of vertices and edges connecting them.

An Artificial Chemistry for Networking

Formal Definition of the *Disperser* Protocol

Network Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$



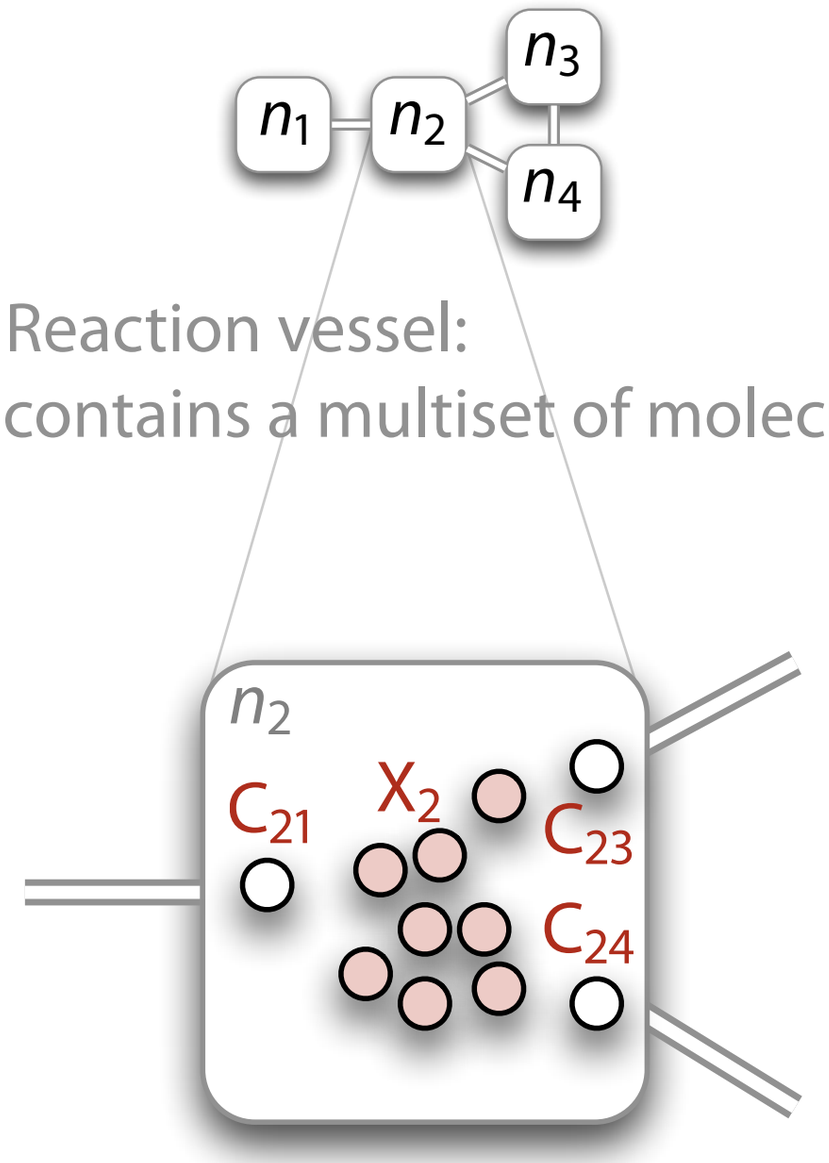
Node: Reaction vessel:
contains a multiset of molecules

Each node is a reaction vessel containing a multiset of molecules.

An Artificial Chemistry for Networking

Formal Definition of the *Disperser* Protocol

Network Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$



Node: Reaction vessel:
contains a multiset of molecules

Set of Molecular Species:

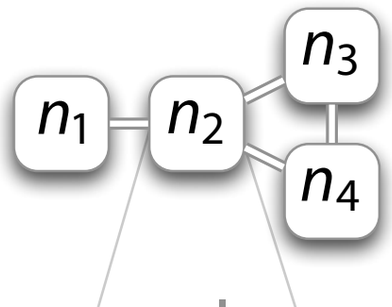
$$\mathcal{S} = \left\{ X_i, C_{ij} \right\} \qquad \forall i \in \mathcal{V},\ (i,j) \in \mathcal{E}$$

S is the set of molecular species or molecule types our protocol works with. For the Disperser protocol, each node contains an X-species representing the data-value and a control-species for each link.

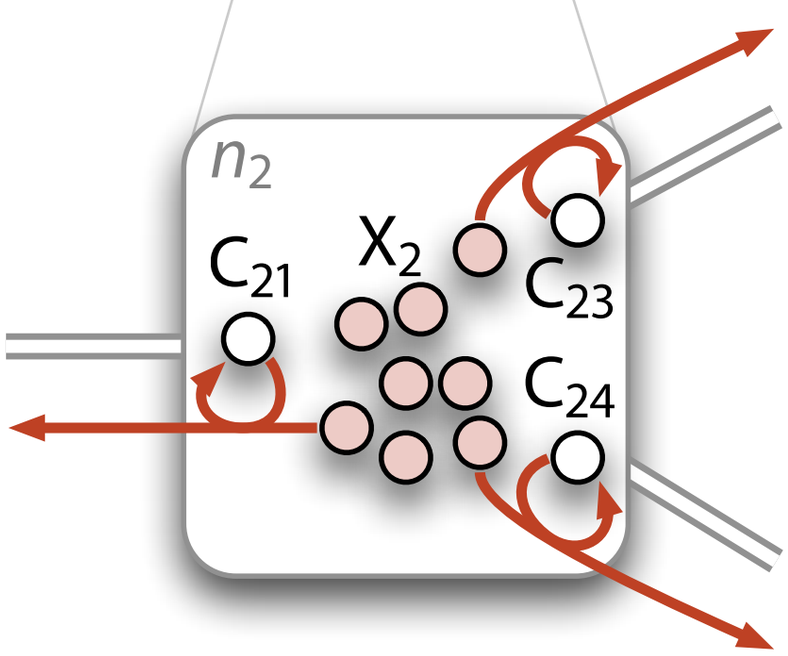
An Artificial Chemistry for Networking

Formal Definition of the *Disperser* Protocol

Network Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$



Node: Reaction vessel:
contains a multiset of molecules



Set of Molecular Species:

$$\mathcal{S} = \{X_i, C_{ij}\} \qquad \forall i \in \mathcal{V},\ (i,j) \in \mathcal{E}$$

Set of Reaction Rules:

$$\mathcal{R} = \{r_{ij} \mid (i,j) \in \mathcal{E}\}$$

$$r_{ij}:\ C_{ij} + X_i \longrightarrow C_{ij} + X_j$$

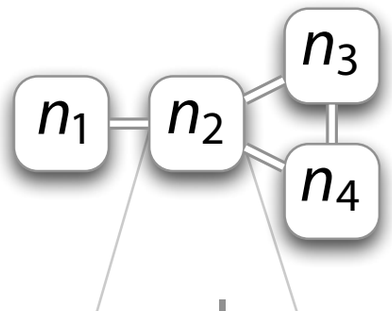
For each link we define a reaction that consumes a random X-molecule and a control-molecule. The control-molecule is regenerated and the X-molecules is sent to the corresponding neighbor node.

In general, a reaction rule can only consume molecules in the same node. The products, on the other hand may be put to the local vessel or to a neighbor vessel. This is how communication is organized, as a reaction with remote products.

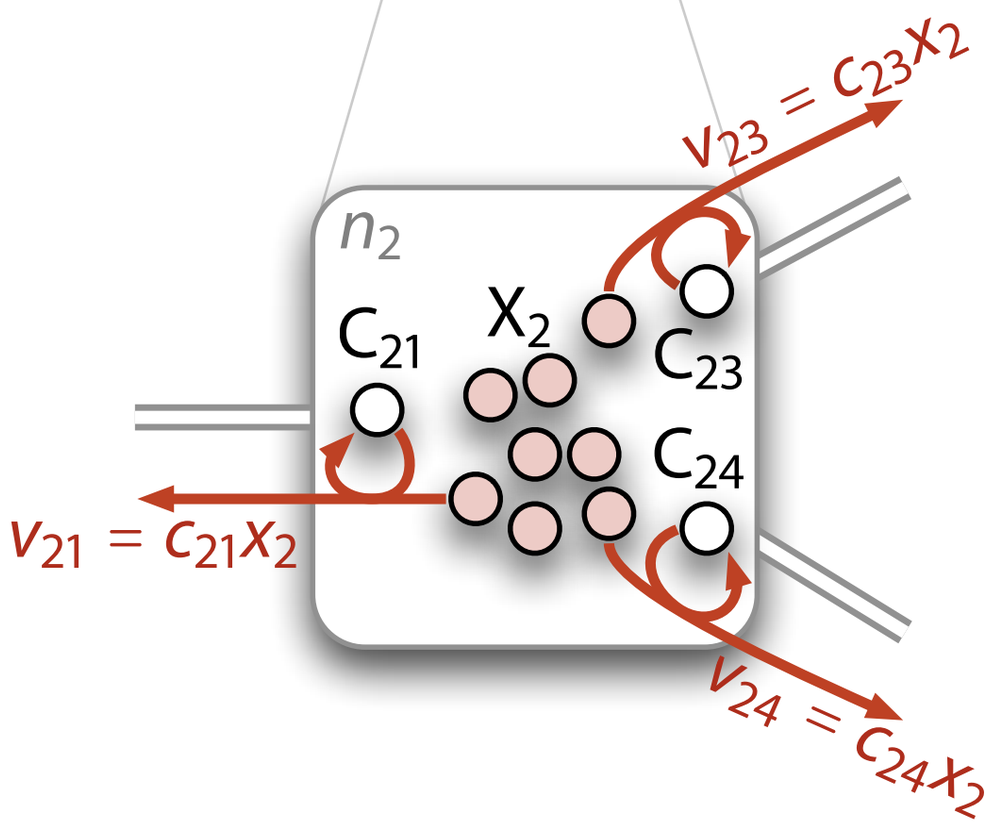
An Artificial Chemistry for Networking

Formal Definition of the *Disperser* Protocol

Network Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$



Node: Reaction vessel:
contains a multiset of molecules



Set of Molecular Species:

$$\mathcal{S} = \{X_i, C_{ij}\} \qquad \forall i \in \mathcal{V},\ (i,j) \in \mathcal{E}$$

Set of Reaction Rules:

$$\mathcal{R} = \{r_{ij} \mid (i,j) \in \mathcal{E}\}$$

$$r_{ij}: C_{ij} + X_i \longrightarrow C_{ij} + X_j$$

Reactor Algorithm: $\mathcal{A}$

Law of mass action scheduler
reaction rate = product of the multiplicity of the reactant species

$$v_{ij} = c_{ij}x_i$$

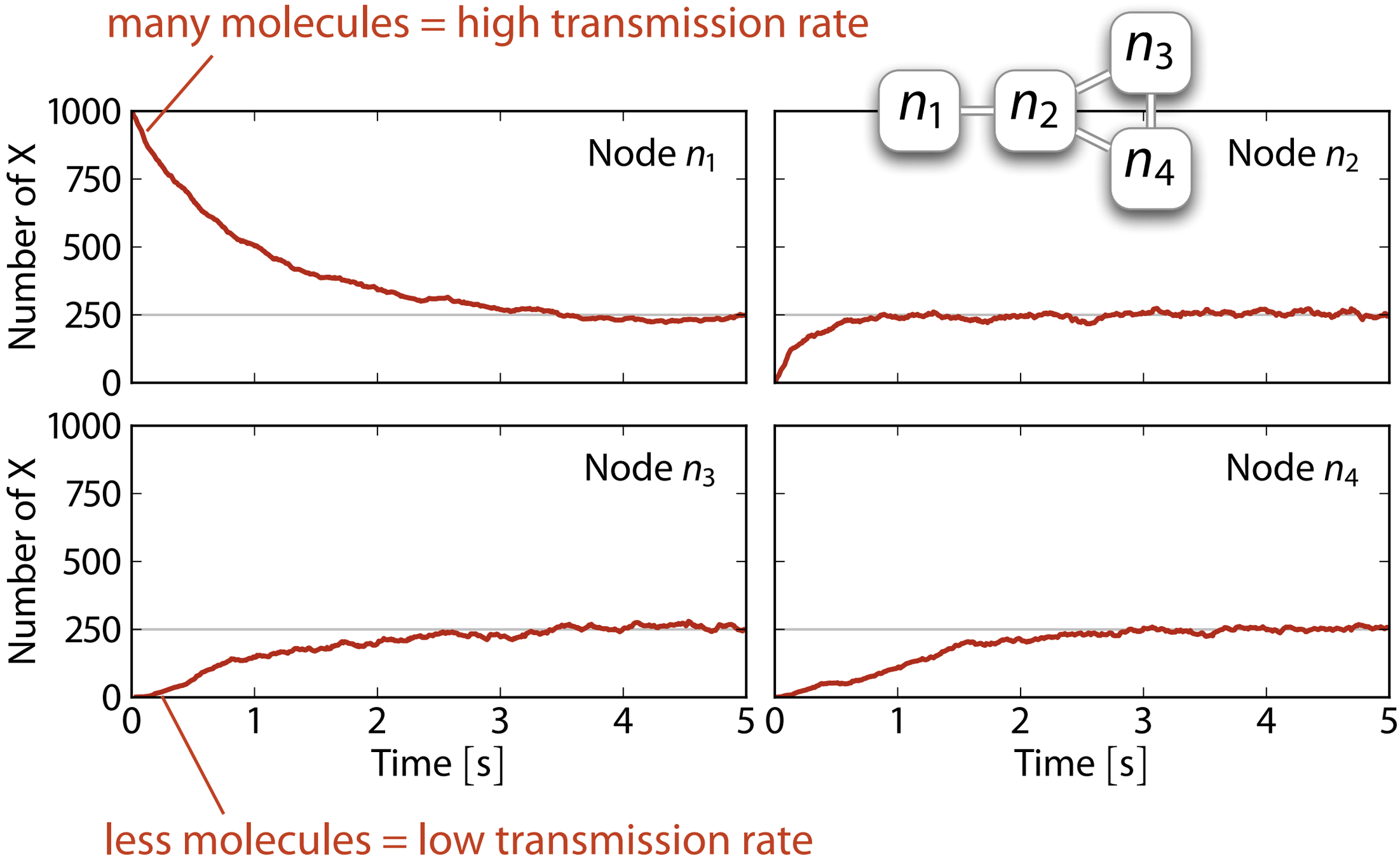
We also need to define when the reaction occur. Unlike most traditional scheduling algorithms, jobs/ reactions are not scheduled for the earliest possible time.

In chemistry, the reaction rate is proportional to the quantity of the reactant species. This is called the law of mass action. The more molecules there are in a vessel of fixed volume, the more frequently the molecules collide and react.

In our case, this means that the reaction rate is equal to the number of X-molecules in the vessel times the number of corresponding control-species.

With this definition of (S,R,A) we follow a recommendation of Dittrich to describe an artificial chemical system. We extended this by the network graph in order to obtain model for networking.

Law of Mass Action Scheduler → Equilibrium

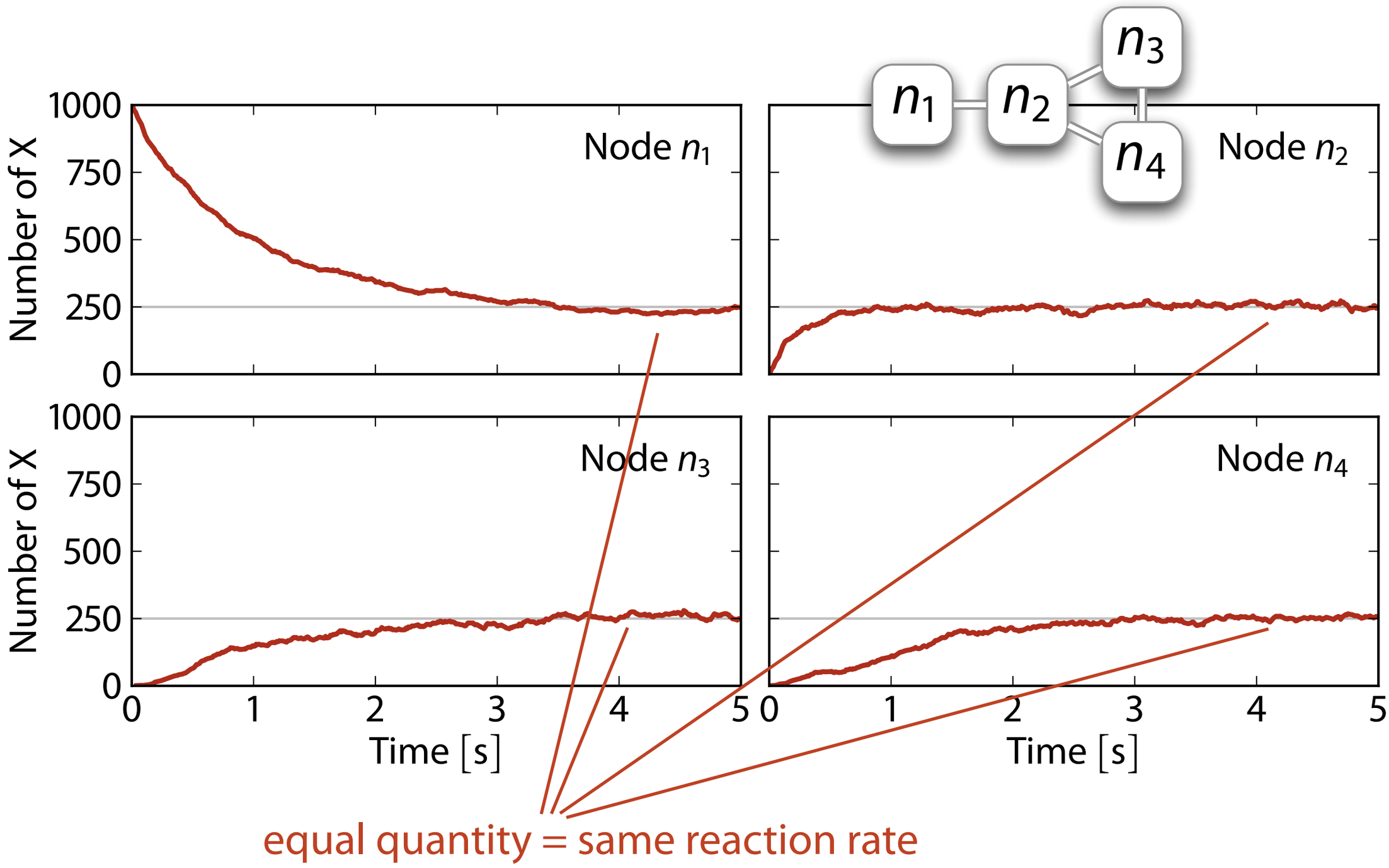


If each node now schedules its reactions according to this law of mass action, the desired outcome emerges.

We start with 1000 molecules in node 1. In the beginning, node 1 has a lot of molecules and therefore, it sends molecules with a high rate.

Node 3 on the other hand only receives a small amount of molecules and therefore sends them with a low rate.

Law of Mass Action Scheduler → Equilibrium



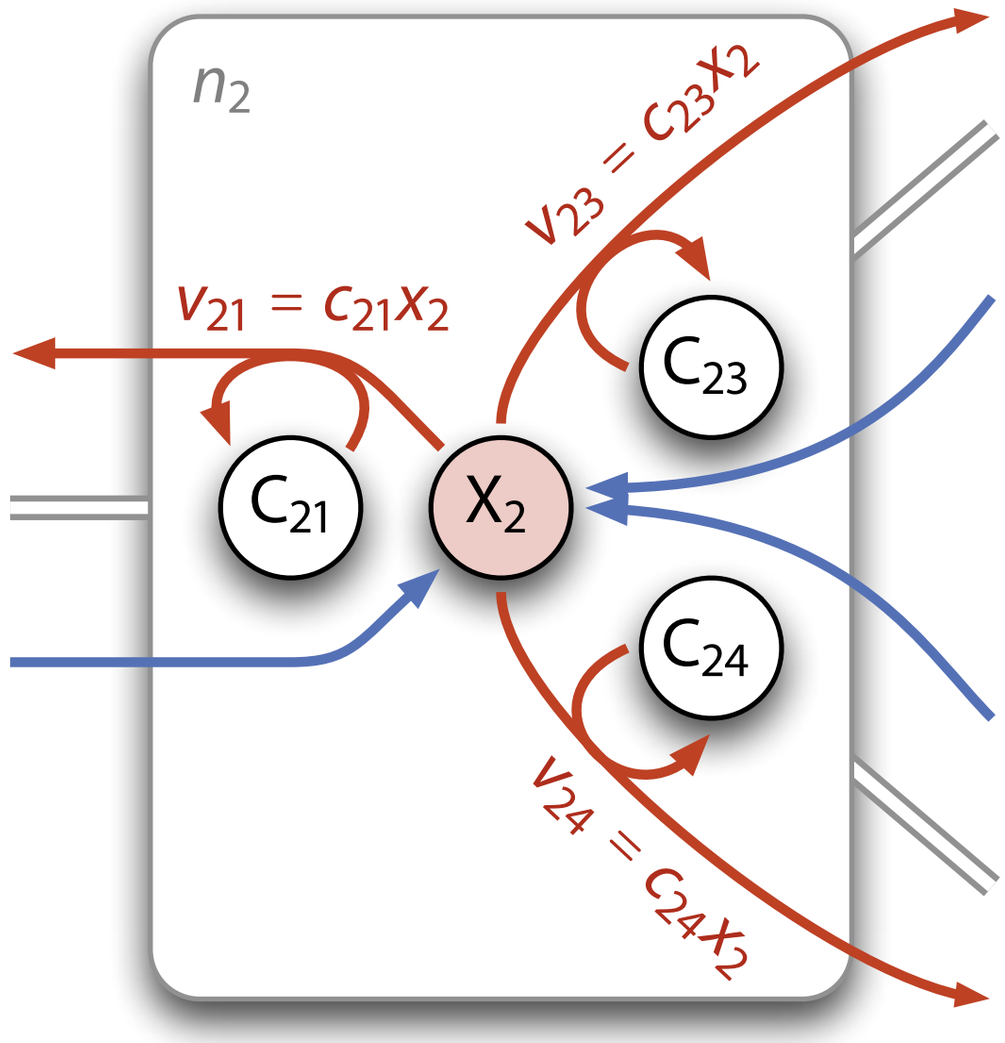
At equilibrium, each node contains the same amount of molecules, and therefore sends them with the same rate to all neighbors. The overall distribution does not change apart from some stochastic fluctuation around the computed result, the average number of molecules.

We are now able to apply analysis methods from chemistry to analyze the protocol's behavior, for example to determine the amount of intrinsic noise we have to expect.

Formal Convergence Proof of the *Disperser* Protocol

1. Write down the differential equations:

$$\dot{x}_2 = \underbrace{x_1 + x_3 + x_4}_{\text{inflow}} \underbrace{-3x_2}_{\text{outflow}}$$



2. Find the fixed point: $\dot{x}_2 \equiv 0$

Concentrations do not change anymore at equilibrium:

$$\hat{x}_2 = \frac{\hat{x}_1 + \hat{x}_3 + \hat{x}_4}{3}$$ local average

$$\hat{x}_i = \frac{\sum_{j \in \mathcal{V}} \hat{x}_j}{|\mathcal{V}|}$$ global average

3. Show that the fixed point is asymptotically stable

Linearize system around fixed point (Jacobian), show that the real part of all eigenvalues are negative.

In this presentation, I will show that we can even prove that the global reaction network strives to the right result.

1. We first write down the differential equation for the change of species X in each node. For node 2, there is an inflow from three neighbors, each rate is equal to the neighbor's concentration of X. And then there are three outflows with a rate equal to the local concentration. We here assume that all control molecules are present with one instance.

2. Next we find the fixed point by setting the time derivative of our state variables to zero, meaning that the concentrations do not change anymore at equilibrium. The above equation yields to this equation, which states that the concentration in node 2 is equal to the average concentration in the second node's neighbors.

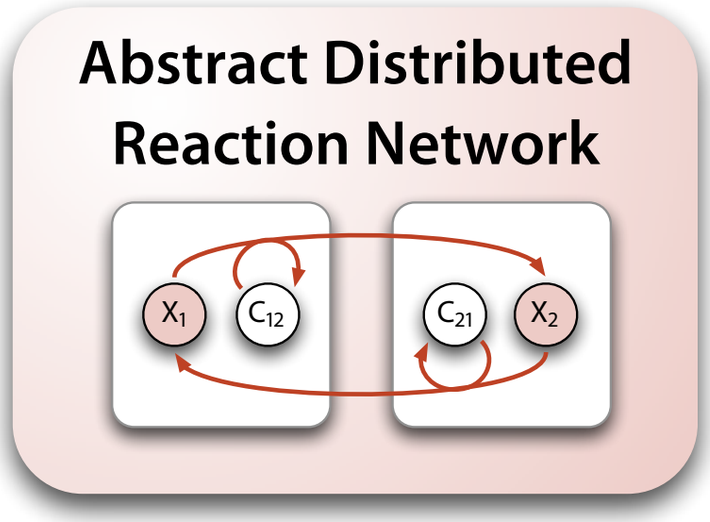
By recognizing that the total number of molecules is conserved in the network, we come to the conclusion that the concentration in each node must be the global average, that is the total number of molecules divided by the number of network nodes.

3. The last step is to show that the found fixed point is asymptotically stable. We do this with a classical perturbation analysis; by linearizing the system around the fixed point and show that the real part of the eigenvalues are negative.

The Chemical Engineering Framework

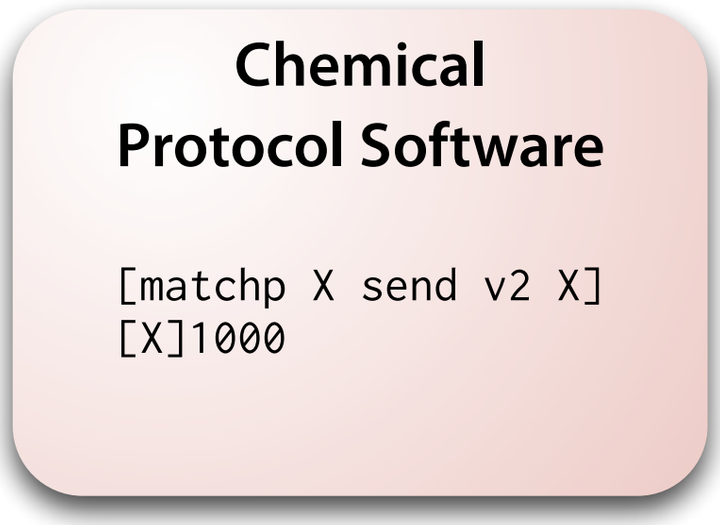
Two levels of granularity

Macro Level



Abstract Model:
Design and Formal Analysis

Micro Level



Execution Model:
Virtual Machine

This high abstraction level where we treat protocols as abstract molecular species is perfect for designing and analyzing chemical protocols.

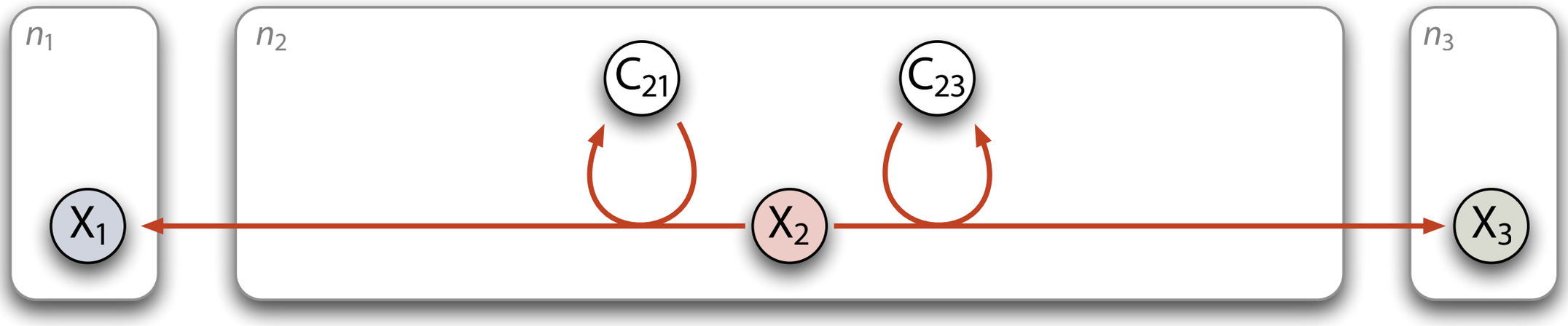
But at the end of the day, we would like to have a language in which we can program chemical reaction networks.

This is why at the microscopic level, we provide a virtual chemical machine that executes chemical reactions described in Fraglets, a chemical programming language.

Chemical Protocol Software

Fraglets — A language to express distributed reaction networks

Abstract Model (for design / analysis)

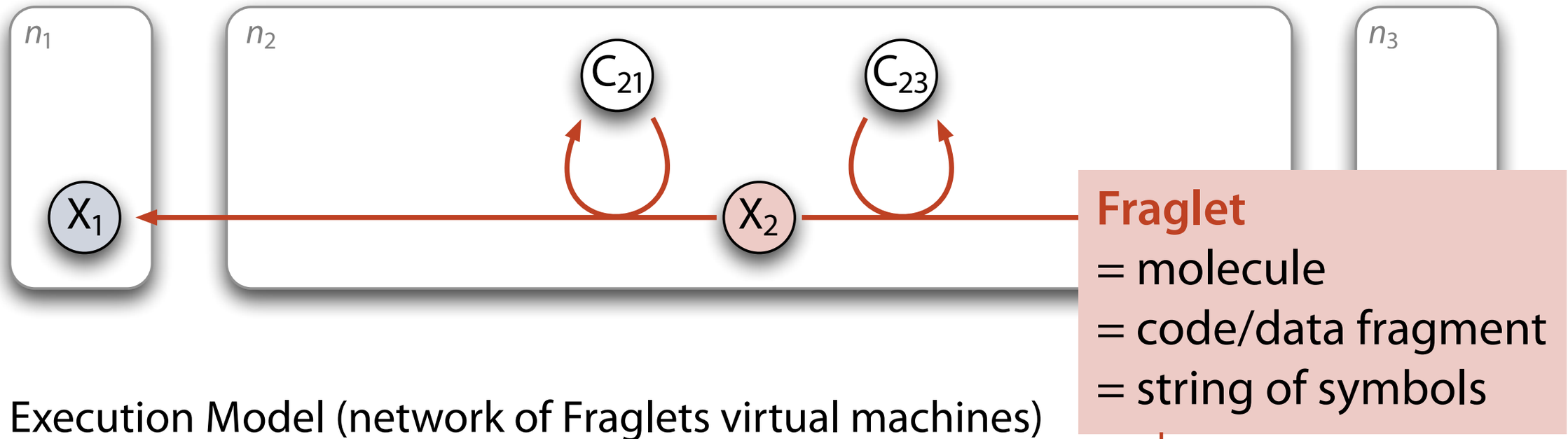


Let us assume that we want to write the “code” for the disperser protocol in node 2, for which the abstract model is depicted here.

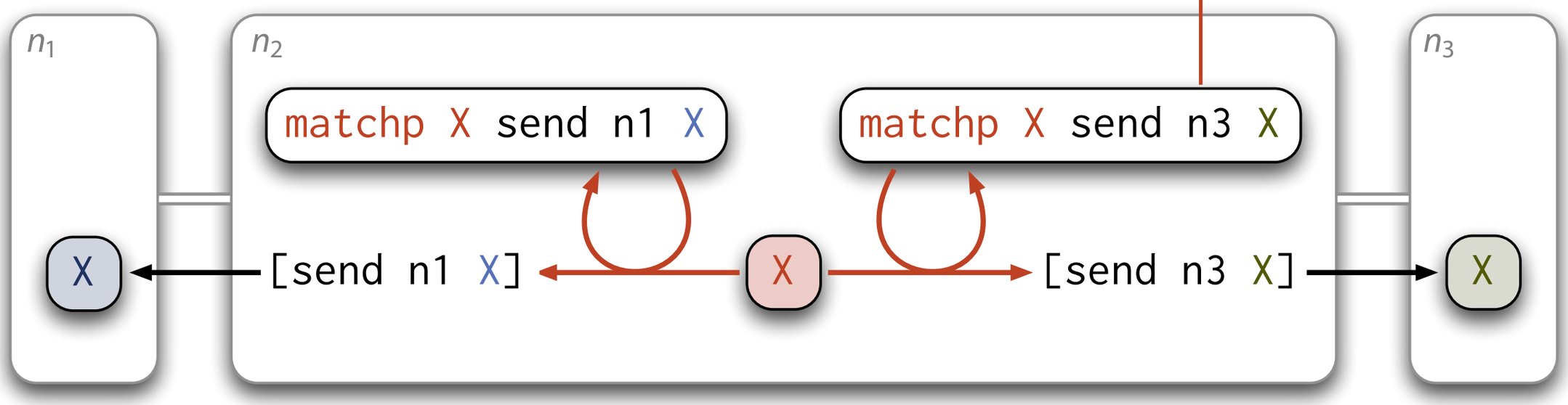
Chemical Protocol Software

Fraglets — A language to express distributed reaction networks

Abstract Model (for design / analysis)



Execution Model (network of Fraglets virtual machines)



In the chemical programming language Fraglets, we would have to install the following two fraglets:

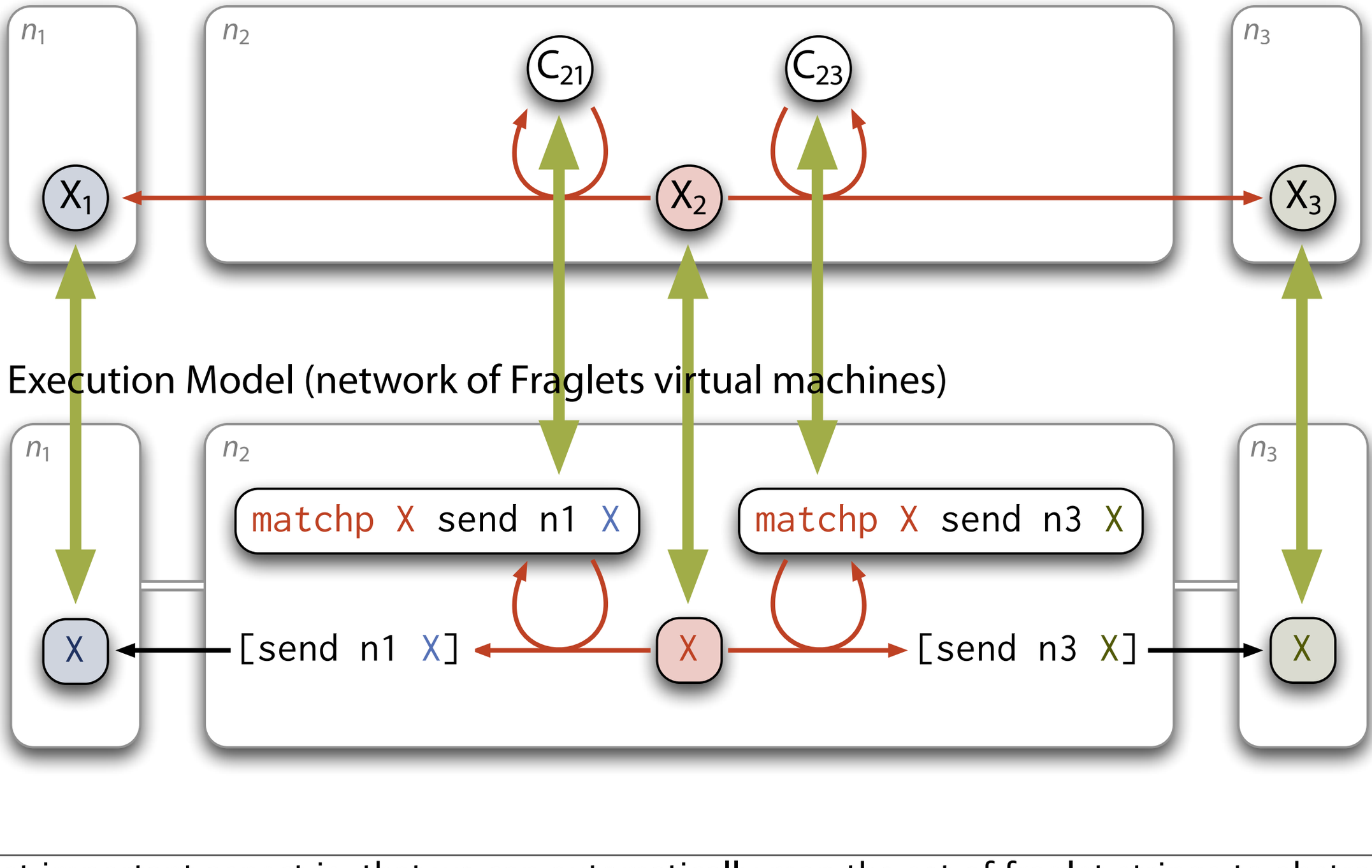
Fraglets is a rule-based programming language developed by Christian Tschudin for efficient packet-processing in a computer network.

A fraglet is a fragment of code and/or data and is represented by a string of symbols. Computation is carried out by string-rewriting operations.
The header tag (match in our case) defines how the string is being rewritten.

For our Disperser example, we have to install such a matchp fraglet for each link, and initialize the node with a bunch of X-molecules, representing our initial data value.

The two matchp fraglets now compete for the X-fraglets. Once a reaction happens, the matchp-fraglet is regenerated and the concatenation of the two is this send-fraglet. The send-instruction actually sends the tail to the neighbor node indicated by this second symbol here.

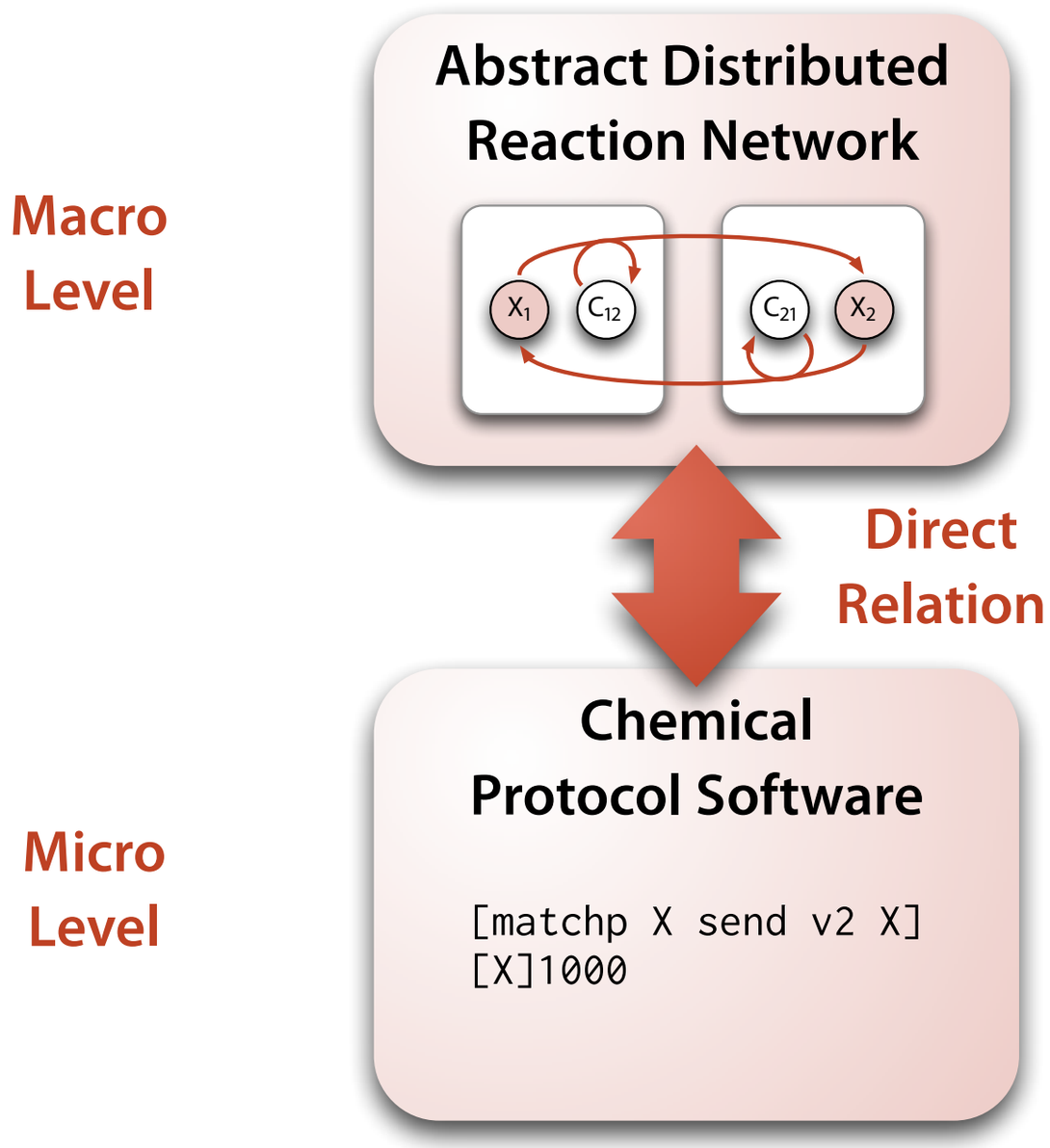
Dynamic Equivalence: Execution ↔ Abstract Model



The most important aspect is, that we can automatically map the set of fraglet strings to abstract species in the macroscopic world, in our abstract model. The two models are dynamically equivalent. This allows our analysis being taken as a prediction of the behavior of the protocol implementation in Fraglets.

The Chemical Engineering Framework

Two **related** levels of granularity

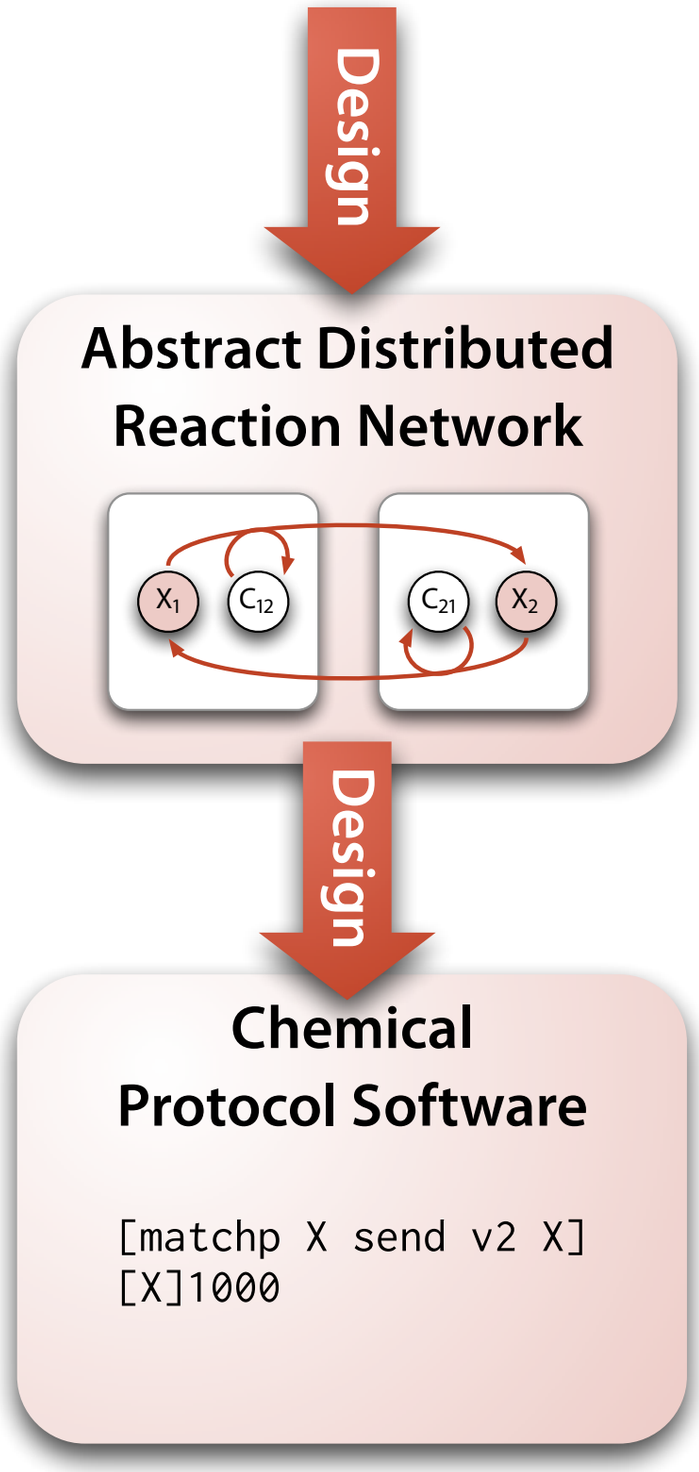


That is, our two models, the microscopic execution model and the macroscopic abstract model are directly related, which is a huge benefit during the design process.

But how are chemical protocols designed? How do we get from a problem statement or idea to a chemical algorithm and its implementation?

The Chemical Engineering Framework

Top-down design: Dynamics first, functional aspects later



Macro Level

Abstract Model:
Design and Formal Analysis

Micro Level

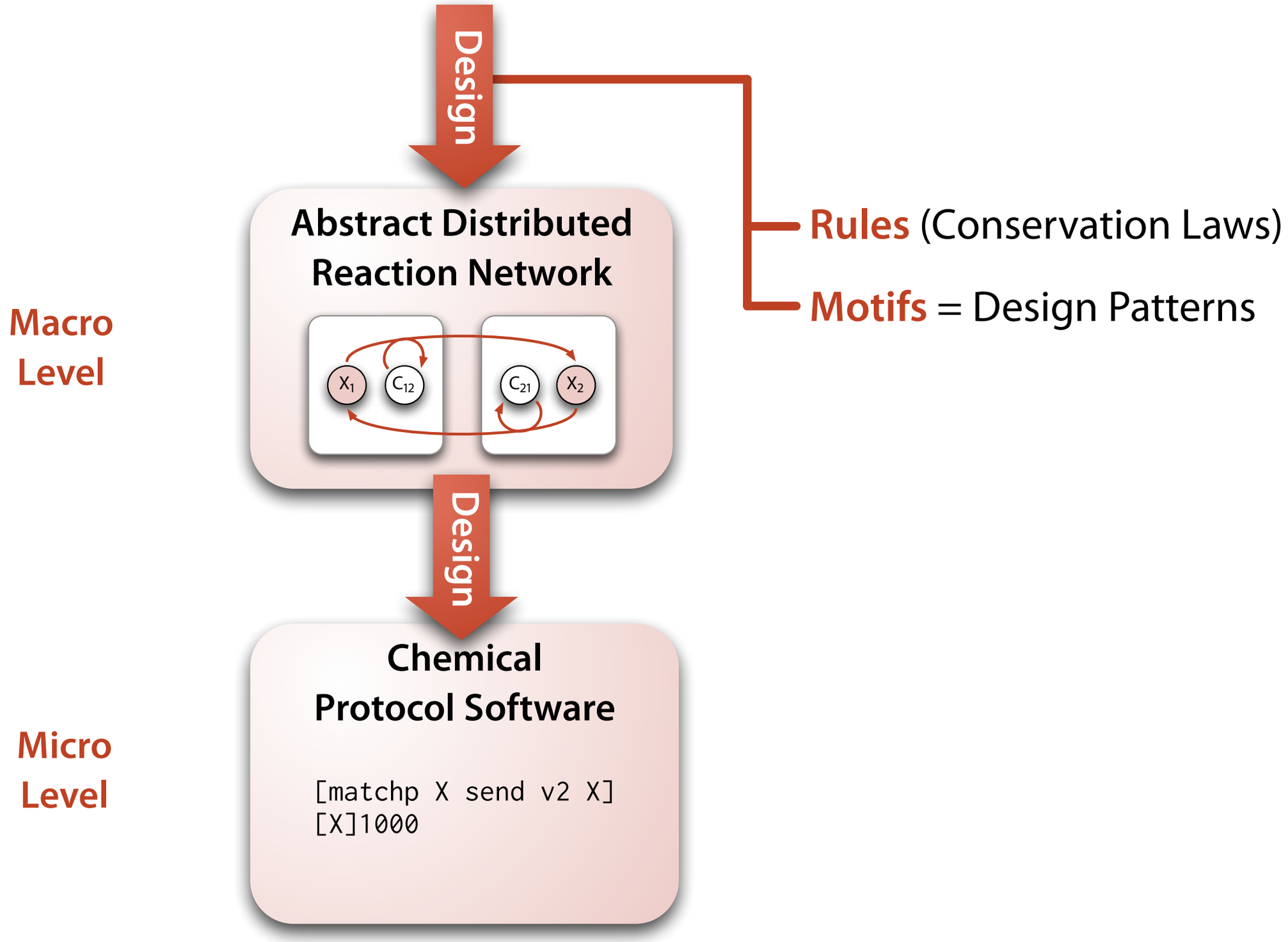
Execution Model:
Virtual Machine

We recommend to design a protocol on the macroscopic level first, first caring about the dynamic behavior before we look at the functional aspects of the fraglets.

This is because the reaction network graph is an intuitive tool to reason about the functionality of a protocol. Later, it is easy to come up with a Fraglets program that implements the designed reaction network.

The Chemical Engineering Framework

Rules and motifs assist the designer in synthesizing protocols



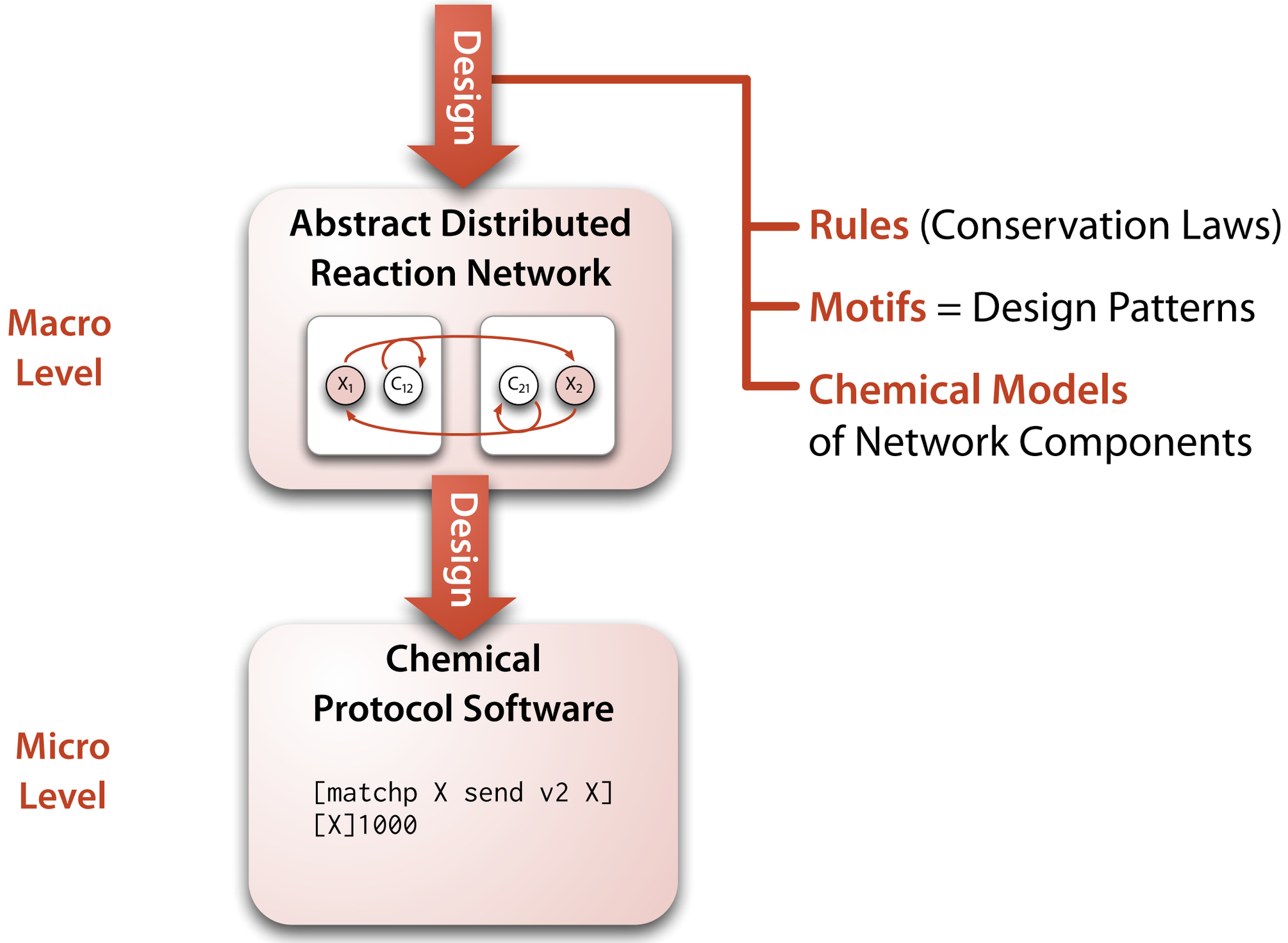
Design is never a process that can be automated. If we remember how object oriented software or electrical circuits are designed, we see that a mix of experience, and a set of rules and common design patterns support the engineer in finding a correct solution for a given problem.

Chemical protocol engineering is perhaps closer to electrical circuit design than to traditional software design.

We provide a set of rules and motifs, that, akin to design patterns can be assembled to synthesize protocol. For example, we came up with motifs to perform arithmetic computation with concentrations, we have a rate limiter motif and a motif doing anycast transmission.

The Chemical Engineering Framework

Model of both, the protocol and the underlying network



We also provide chemical reaction models for networking components like queues or bandwidth limited network links, for example.

This allows both, the protocol and the underlying computer network to be modeled and analyzed within the same framework.

Outline

Motivation

Chemical Networking Protocols

- Introductory Example / The Chemical Metaphor
- Chemical Protocol Engineering Framework
- **Application Case: C3A — A Chemical Congestion Control Algorithm**

Self-Healing Protocols

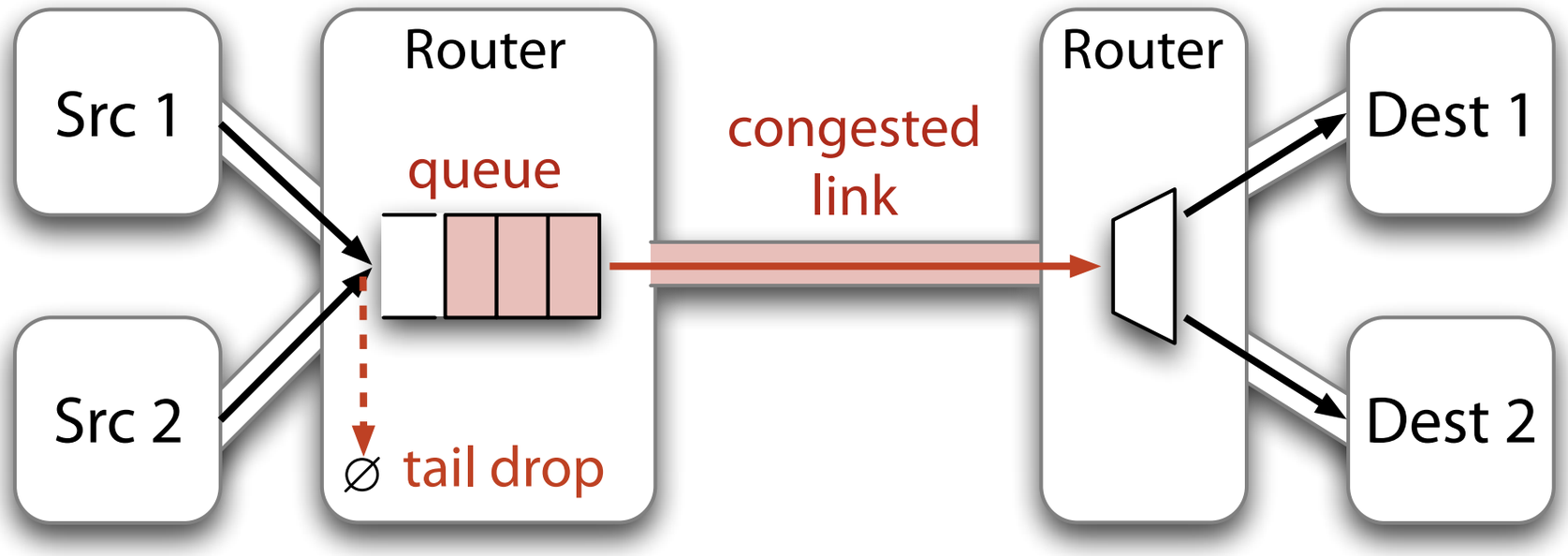
- Robustness to Code Deletion
- Application Case: Self-Healing Link-Load Balancing Protocol

Conclusions

As an application case, I will now show you a chemical congestion control algorithm.

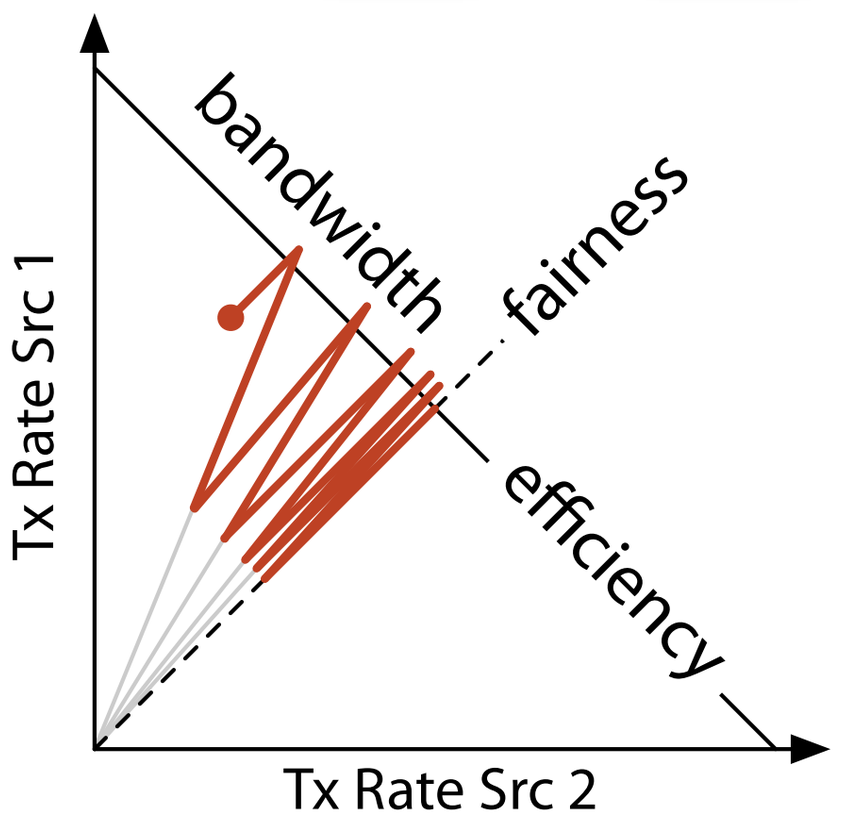
For a coexistence between chemical and traditional protocols, it is essential that a molecule stream through the internet is able to adapt its own transmission rate.

Congestion Control in the Internet



TCP Reno:

additive increase,
multiplicative decrease



Congestion appears in a network if source nodes send more data than the actual capacity of the path. Congestion should provide all participants with a fair share of the limited bandwidth.

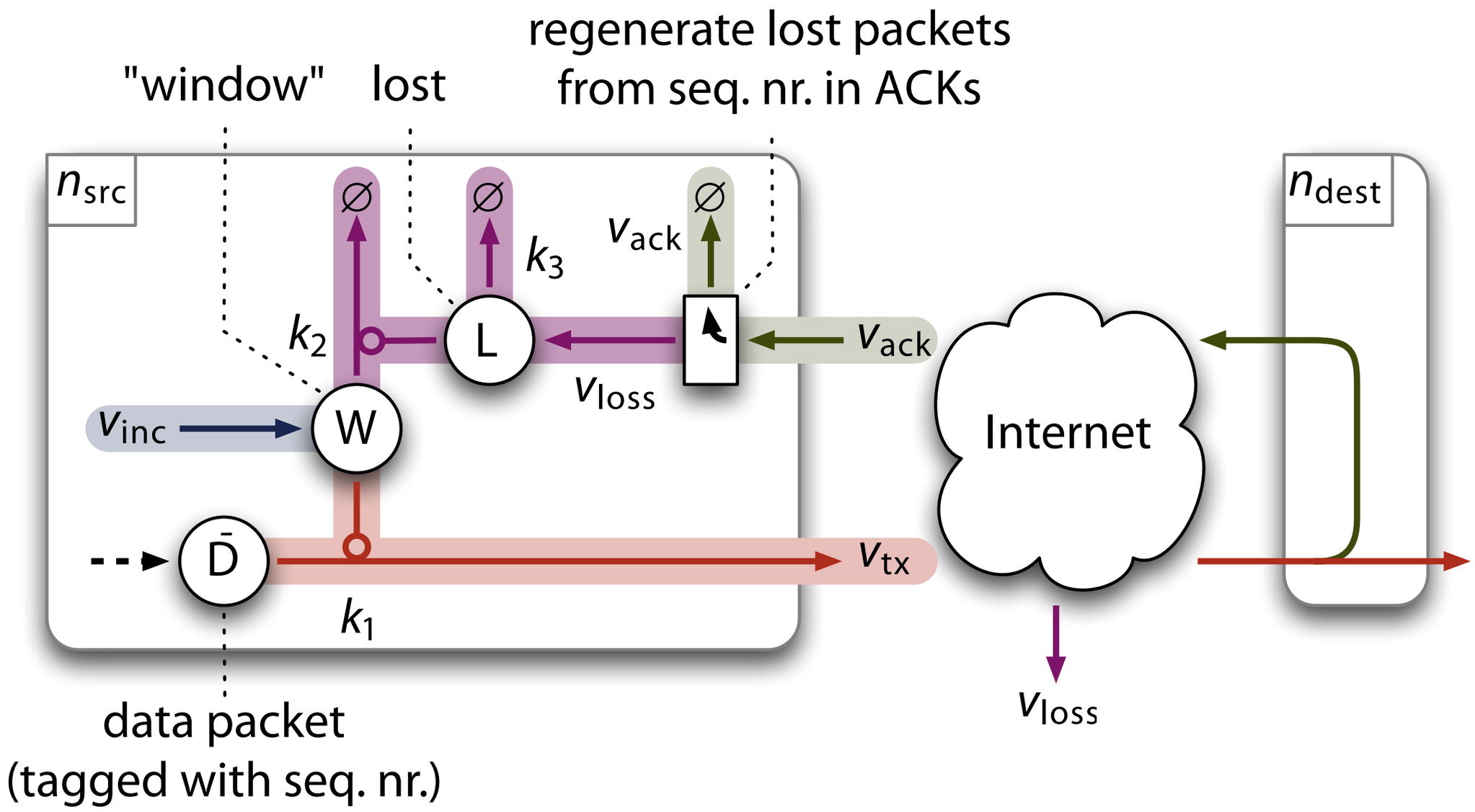
TCP uses an end-to-end congestion control algorithm where the sender throttles its transmission rate based on lost acknowledgments.

The additive increase/multiplicative decrease algorithm increases its transmission rate linearly, and, when a packet loss is detected, it divides the transmission rate by two and start to increase it again.

As shown in this figure this leads to an equilibrium which is fair and efficient, meaning that the whole bandwidth is evenly distributed between all participants.

C3A — A Chemical Congestion Control Algorithm

Feedback control of the transmission rate



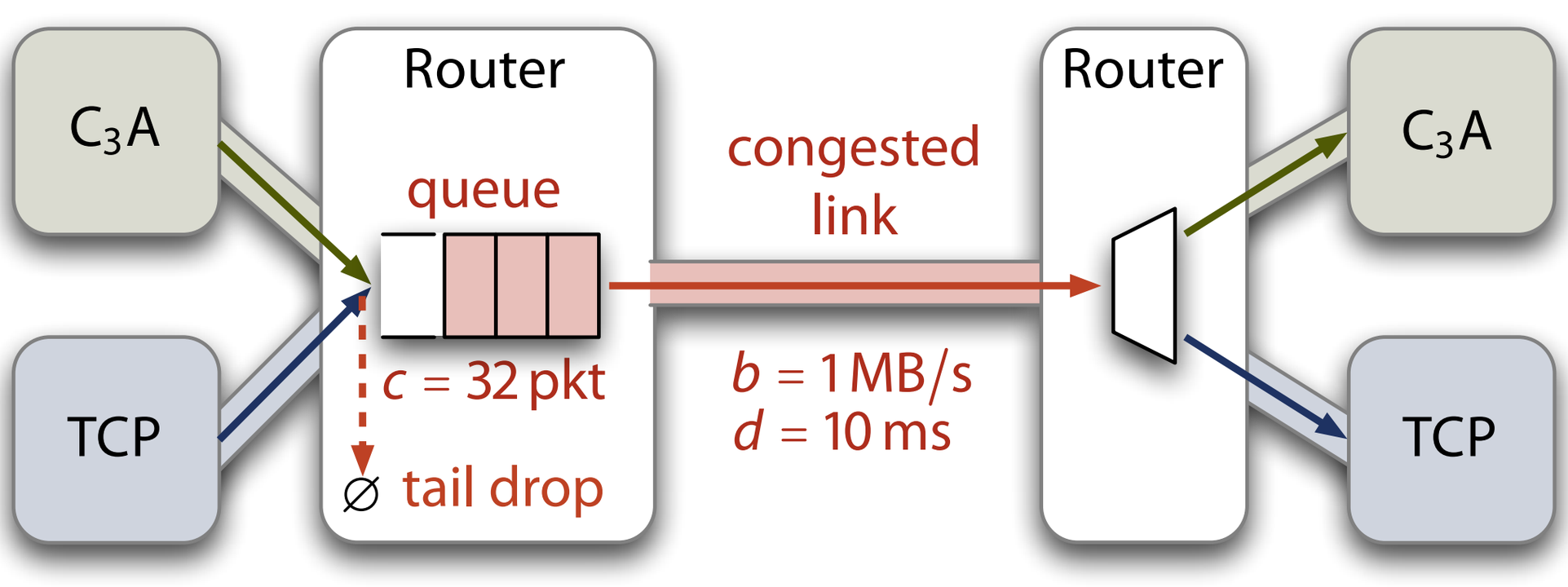
This is the chemical reaction network that has exhibits the same additive increase /multiplicative decrease mechanism.

We don't dig into the detail of the operating principles of the protocol, but I want to highlight two aspects:

1. compared to classical computer programs, the reaction network visualizes the function quite well. We can see the reaction responsible for the linear increase of the transmission rate and we see the feedback control loop.
2. the protocol does not have to compute the transmission rate explicitly and set timers to send packets accordingly. The underlying scheduling algorithm takes care of this. We just have to weave the reaction network properly.

C3A — Competing for Bandwidth against TCP

Network topology for OMNeT++ simulations



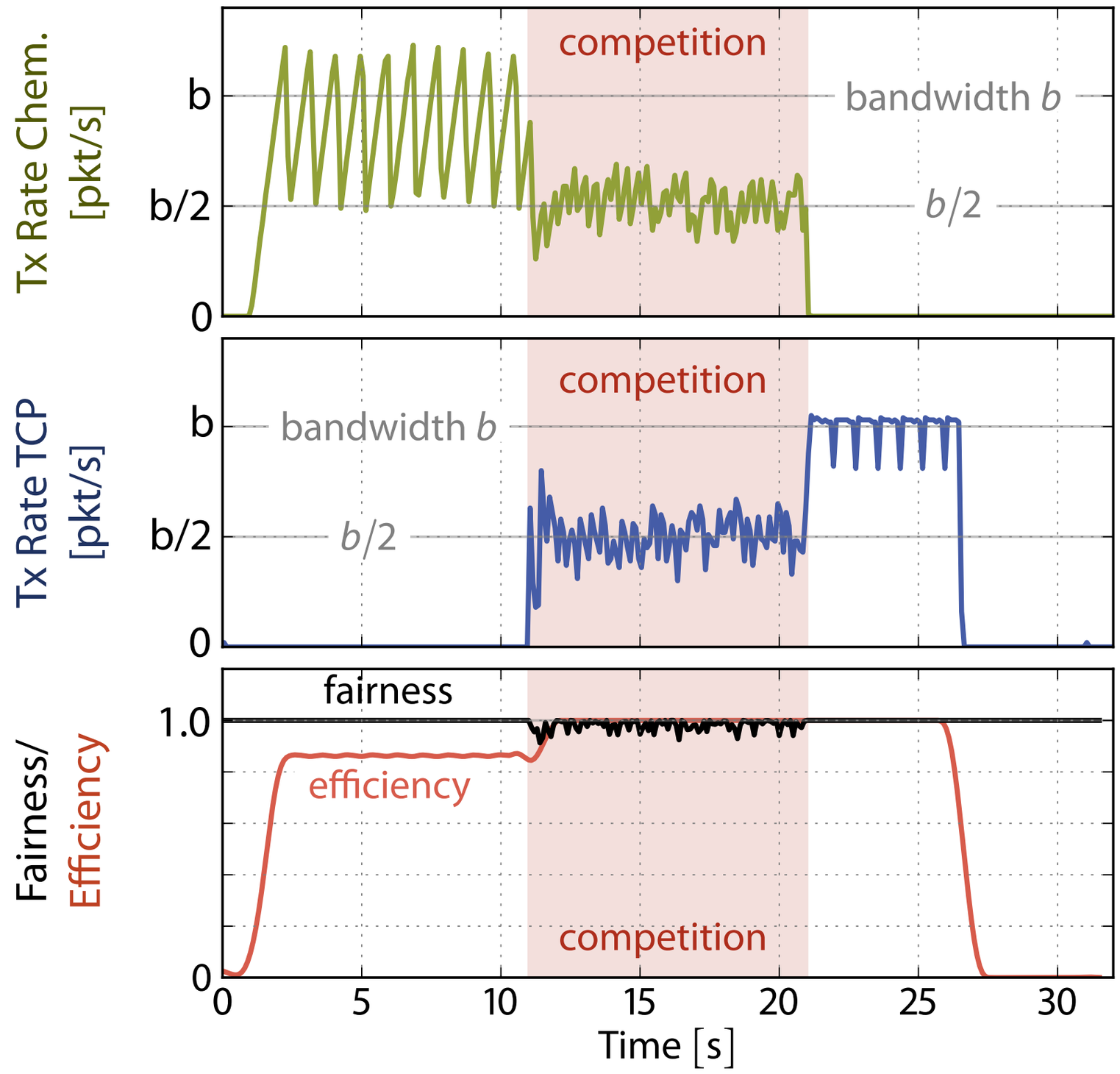
We performed simulation runs in OMNET, a network simulator. Two packet streams, a chemically controlled competes with a TCP stream for limited link bandwidth.

C3A — Competing for Bandwidth against TCP

OMNeT++ simulation results

C3A

TCP

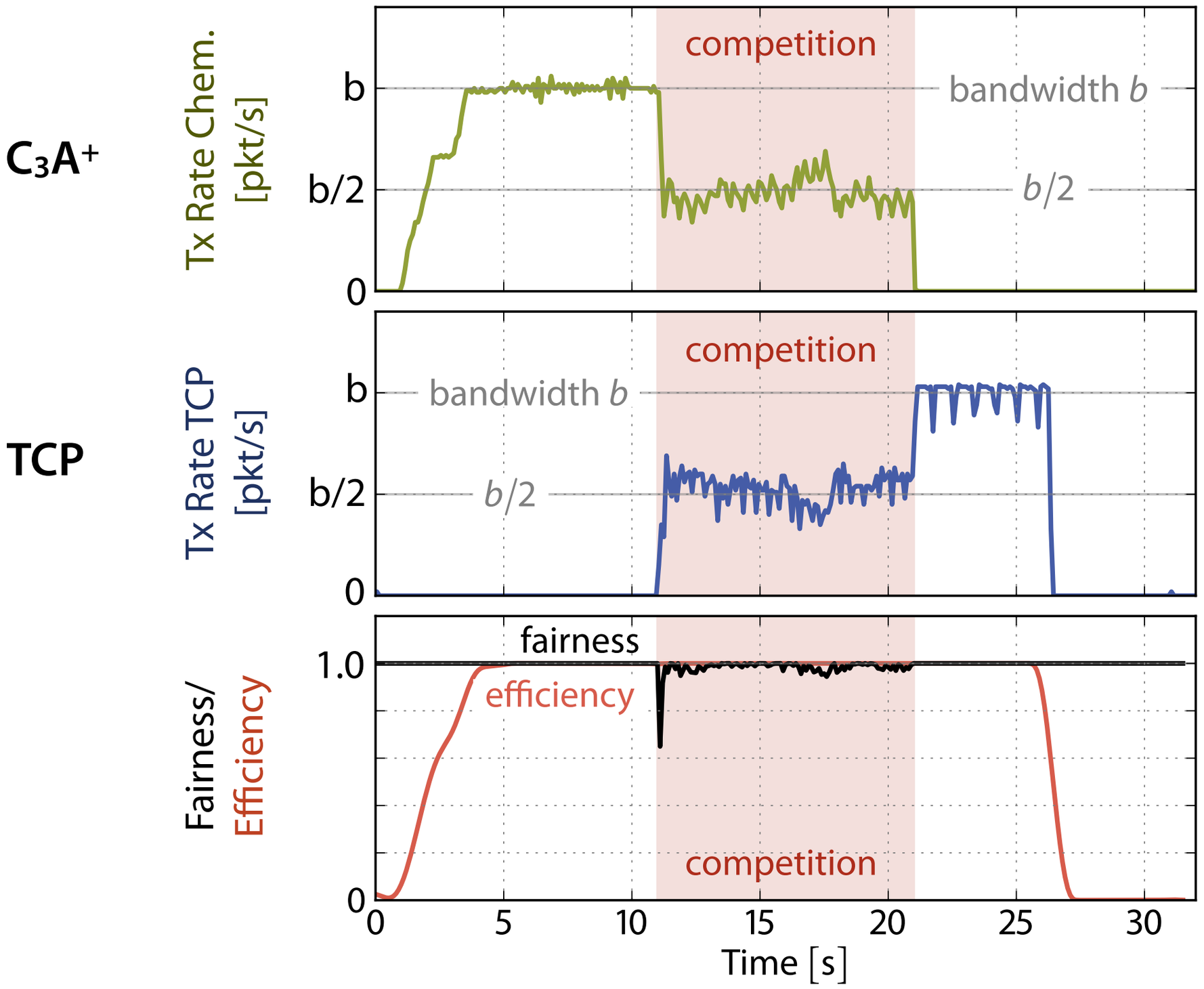


In the top figure we can observe the typical saw-tooth pattern of linear increase and multiplicative decrease in case of a packet loss.

Then, at time 11s, TCP starts to compete for the bandwidth. As we can see, both streams obtain the same bandwidth share. This is also indicated by the fairness index shown in the bottom figure. However, the efficiency of C3A is not 100% – the algorithm cannot exploit the full bandwidth alone.

C3A+ — Competing for Bandwidth against TCP

OMNeT++ simulation results of the improved algorithm



We came up with an improved version that is able to fully exploit the bandwidth and which is smoother than the original one.

Outline

Motivation

Chemical Networking Protocols

- Introductory Example / The Chemical Metaphor
- Chemical Protocol Engineering Framework
- Application Case: C3A — A Chemical Congestion Control Algorithm

Self-Healing Protocols

- Robustness to Code Deletion
- Application Case: Self-Healing Link-Load Balancing Protocol

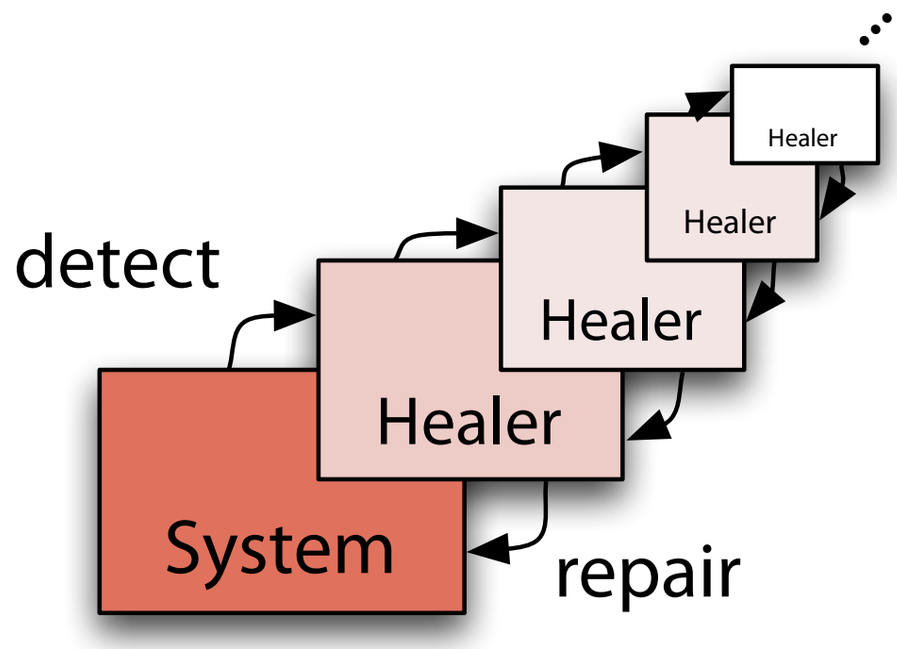
Conclusions

In the second part of my talk I will address another aspect of the chemical model we studied. In the chemical setting, it is quite natural to come up with self-healing software, whereas this is barely possible with traditional execution models.

Like transmission control protocols are able to detect the loss of packets, we want software to be able to detect the accidental loss of parts of the protocol's code.

Traditional Self-Healing Architecture

Feedback control by an external "Healer" module



Problem: Infinite regression

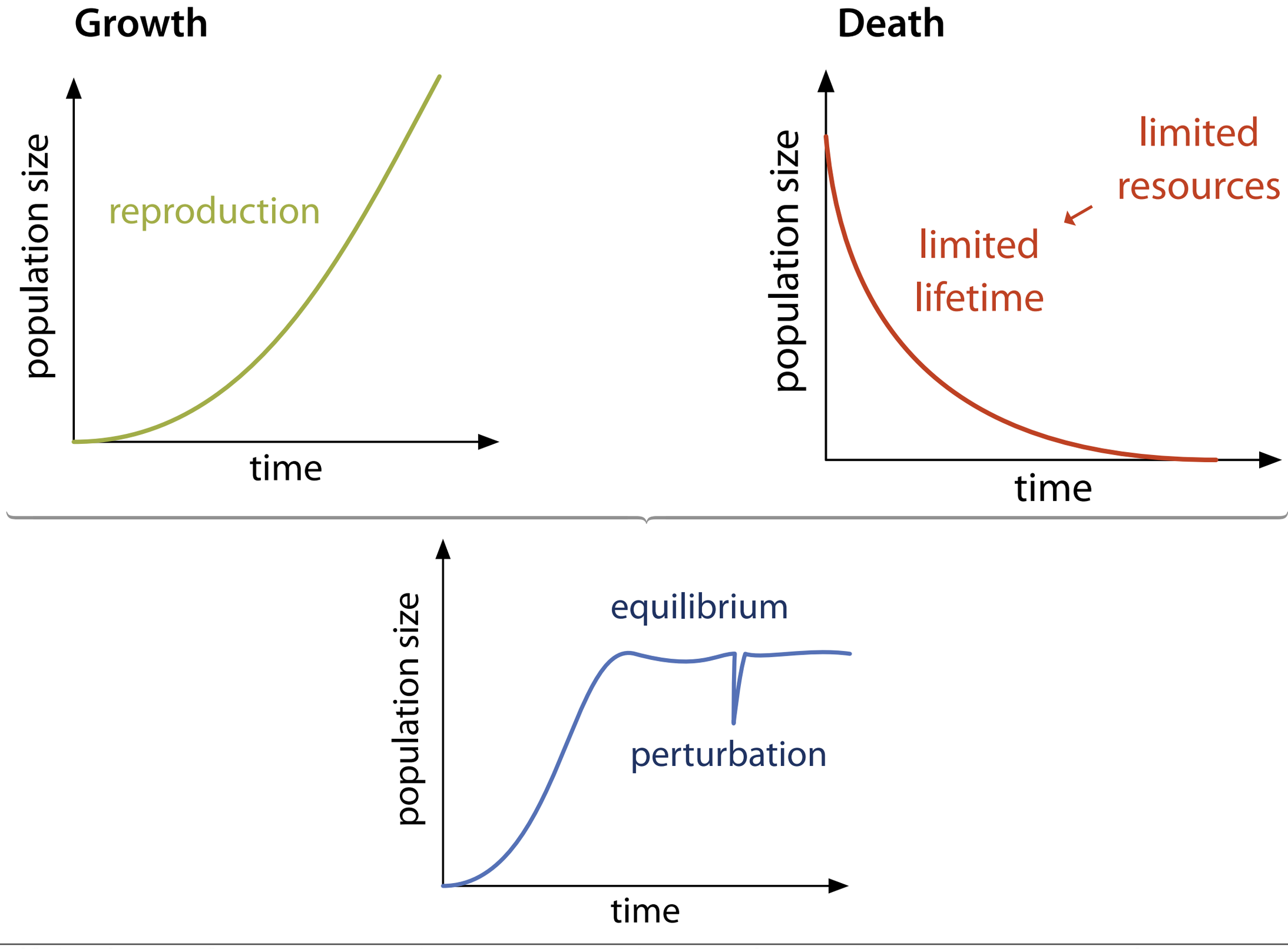
Who is healing the healer?

The classical system architecture for self-healing systems is to separate the system from the healer component:

The system is continuously observed by the healer, which tries to detect possible faults and plans and carries out the necessary repair actions.

But the header is also a software component, which may fail too. In this case, we require a healer of the healer, and so on, leading to infinite regression.

Population Dynamics: Balance of Growth and Death



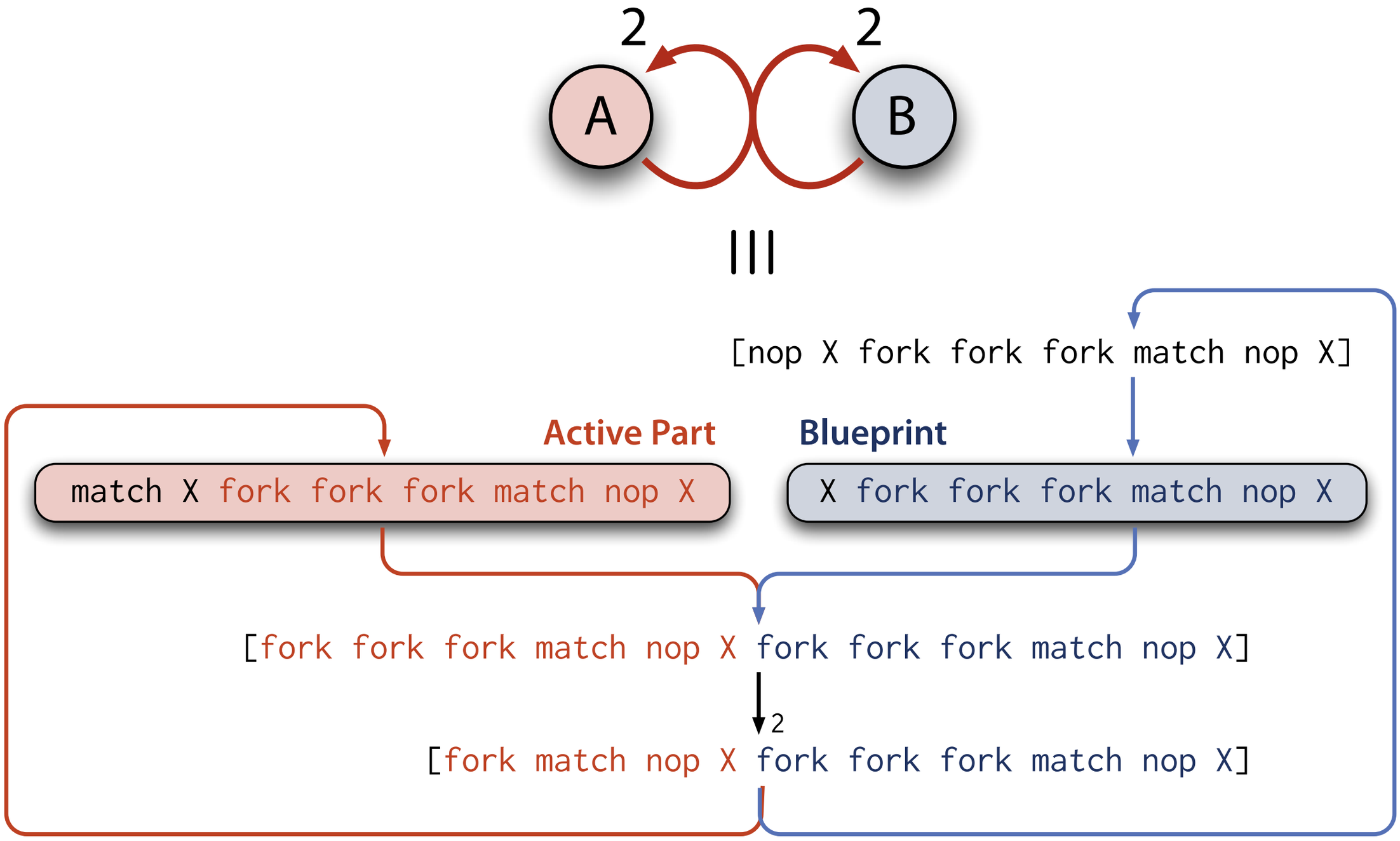
Our self-healing software approach was inspired by population dynamics. In a population, two opponent forces – growth and death – are balanced:

Individuals of a population reproduce, leading to population growth. On the other hand, individuals die, because of limited resource or because of aging.
Together, these two forces lead to an equilibrium that stabilizes the population size and is resilient to perturbations: If we remove some individual, the population size returns to equilibrium.

But how can we transpose growth and death to software? Growth actually means that software parts continuously replicate themselves in memory.

Growth in Fraglets — The Replicating Quine

Functional Aspect: Self-Replicating Code

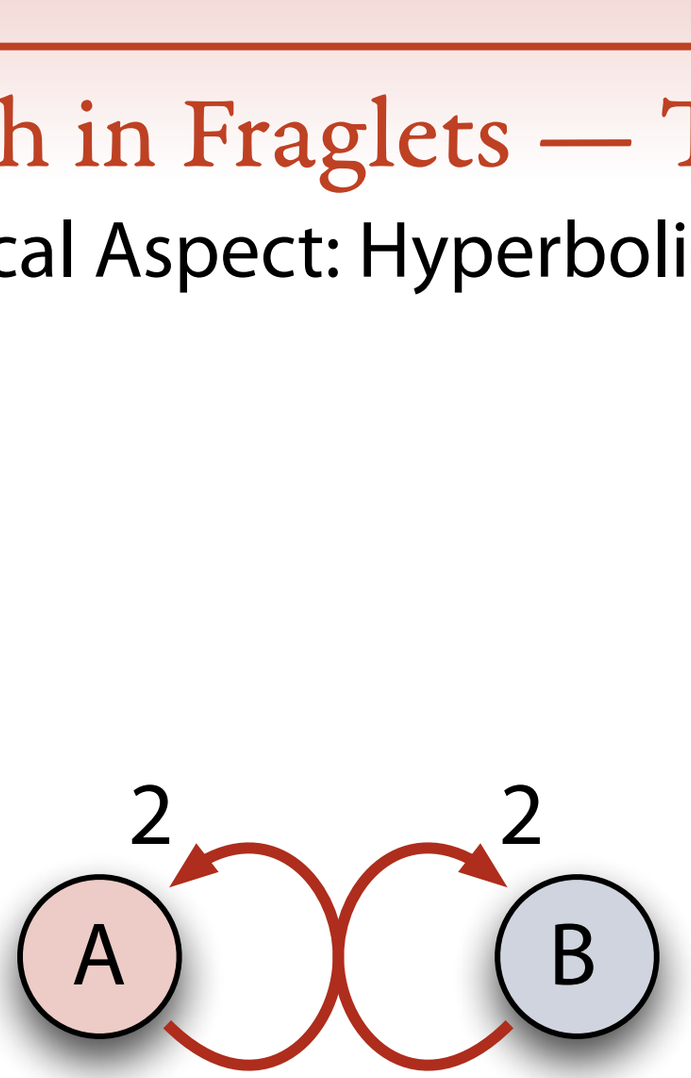


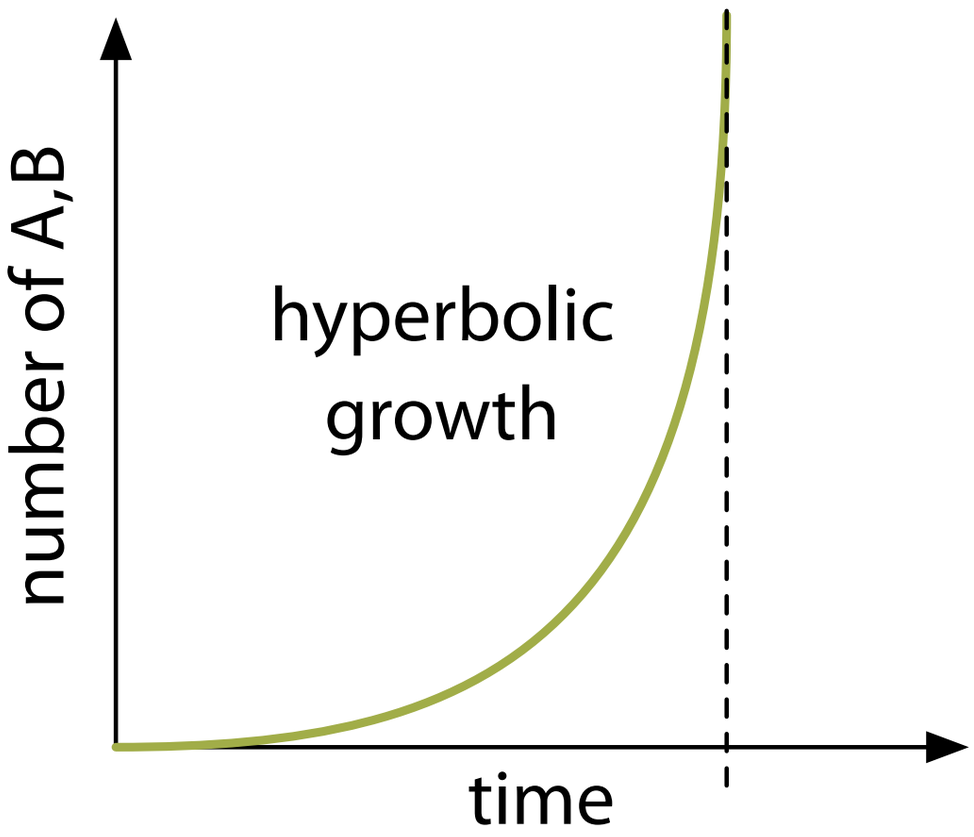
We implemented growth by a replicating Quine. A Quine is a well-known concept in computer science meaning that a computer program actually prints its own source code as output. In the chemical context of Fraglets, we place these two molecules into a reaction vessel. They contain the same information, once as passive information in the blueprint and once as active version of it. The two fraglets react and – via some rewriting steps – generate two copies of themselves.

The reaction network at the top shows the corresponding abstract reaction network in the macroscopic model.

Growth in Fraglets — The Replicating Quine

Dynamical Aspect: Hyperbolic Growth





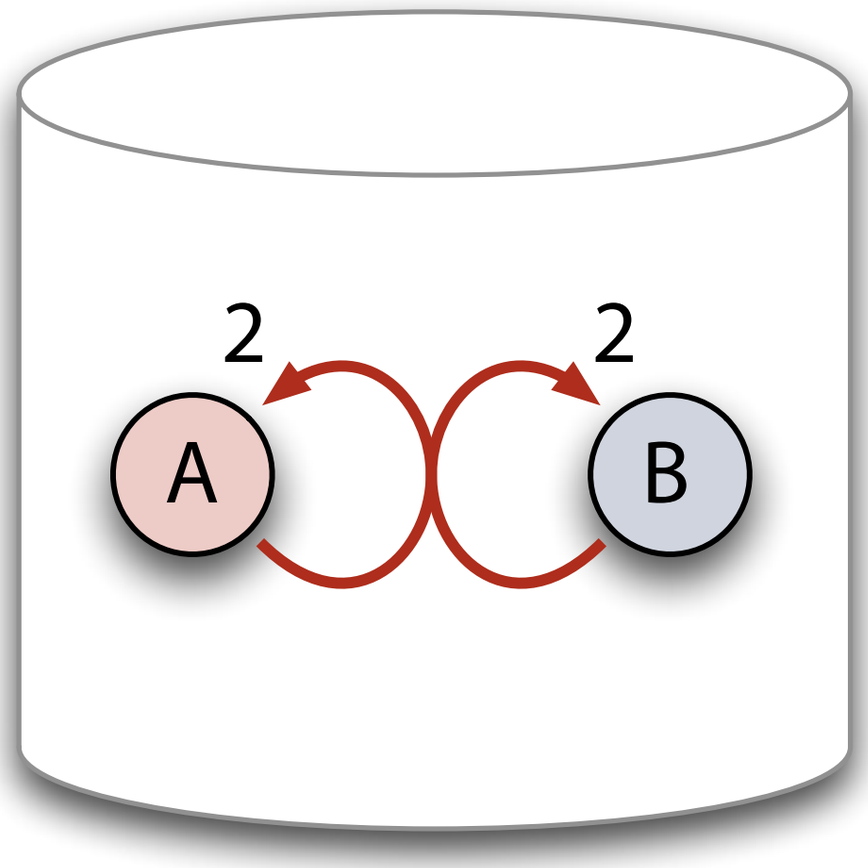
Hyperbolic growth not realistic: **Resource (memory) is limited**

Its dynamics is also governed by the law of mass action leading to a hyperbolic growth. Hyperbolic growth actually means that in finite time, the software population reaches an infinite population size.

This is clearly not realistic as memory is always limited.

Death in Fraglets — Random Dilution Flux

Memory is limited

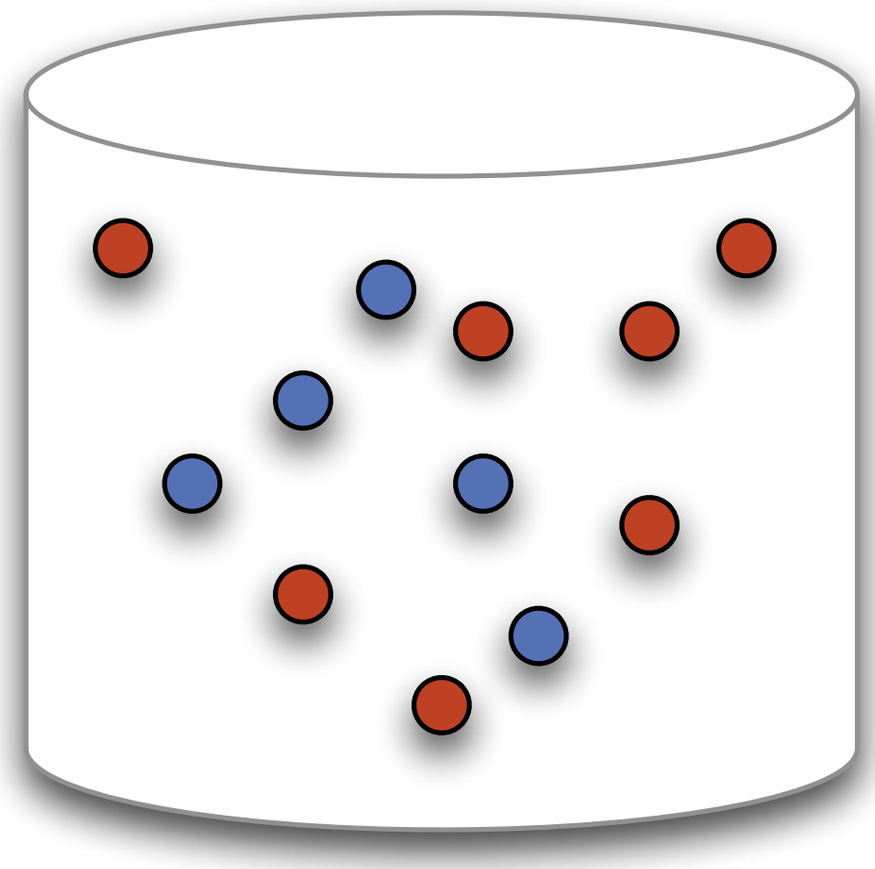


Reaction vessel
of
finite volume

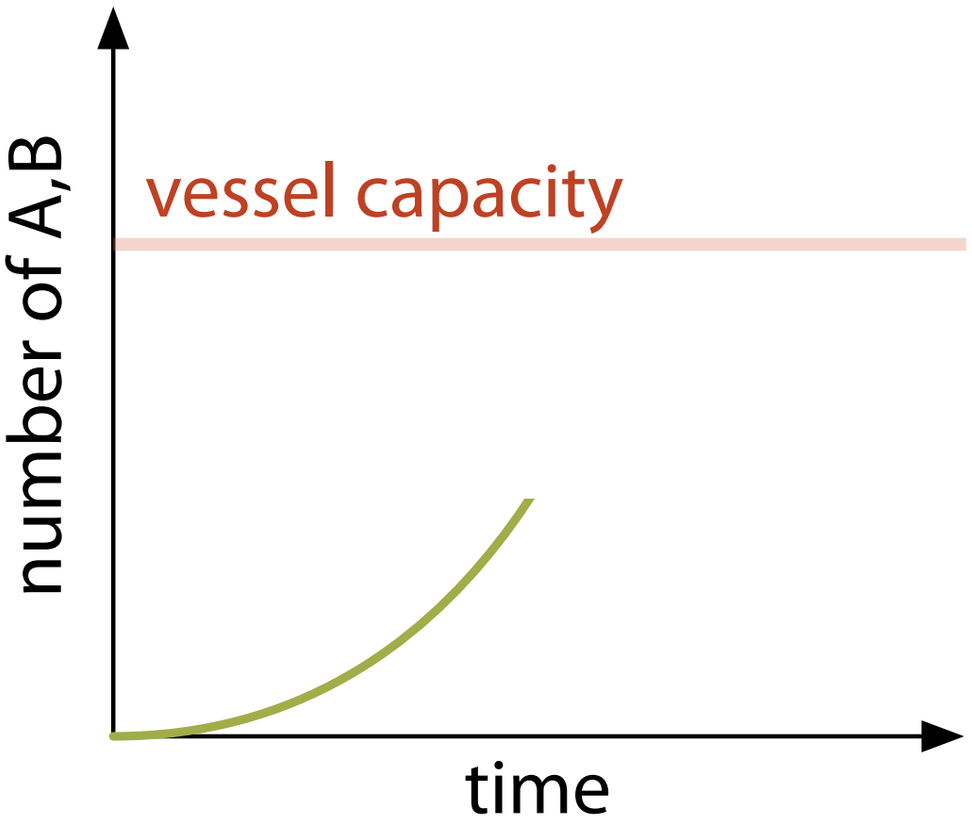
We therefore envision the Quine to reside in a reaction vessel of finite volume...

Death in Fraglets — Random Dilution Flux

Memory is limited; population grows



Reaction vessel
of
finite volume

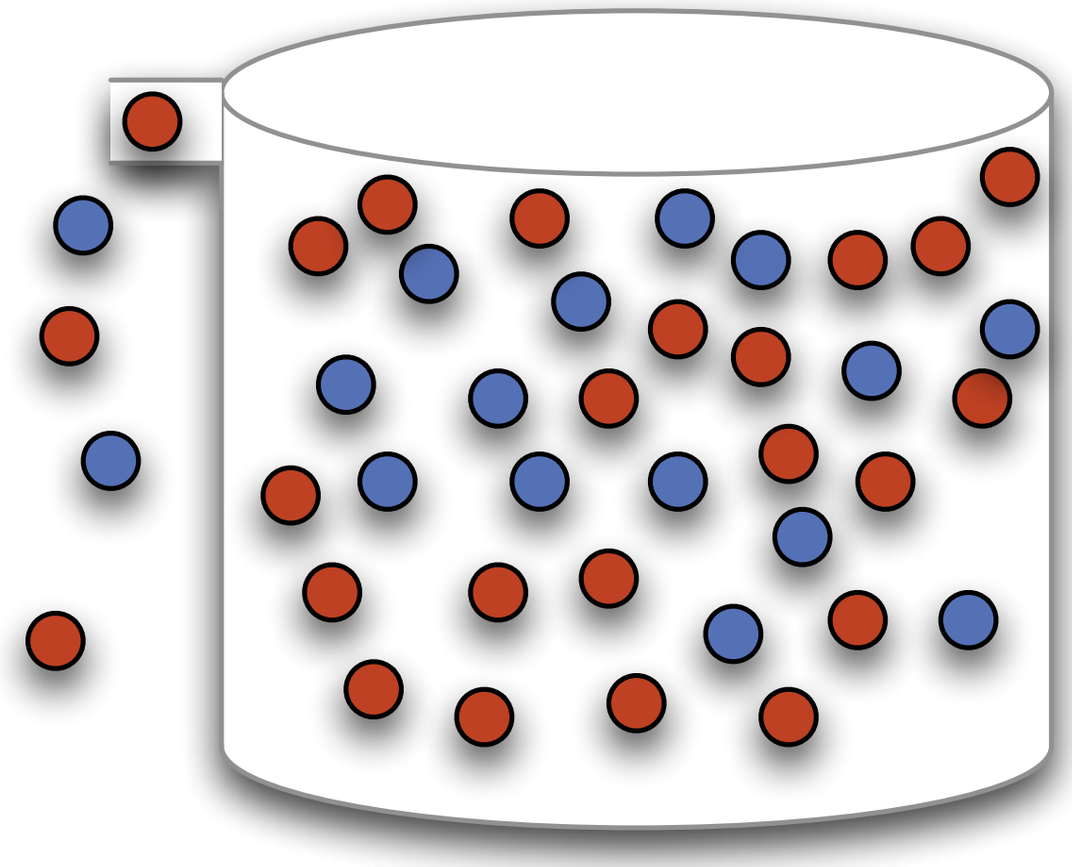


...where an initial population first grows...

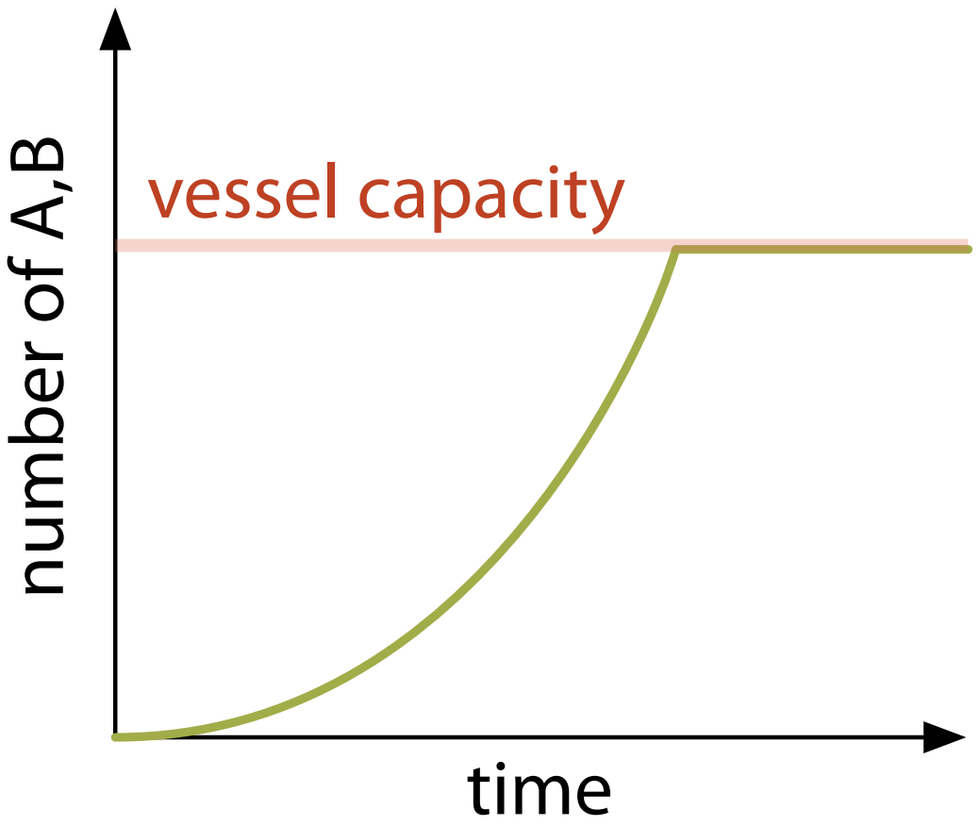
Death in Fraglets — Random Dilution Flux

Memory is limited; population grows until vessel capacity is reached

Dilution Flux



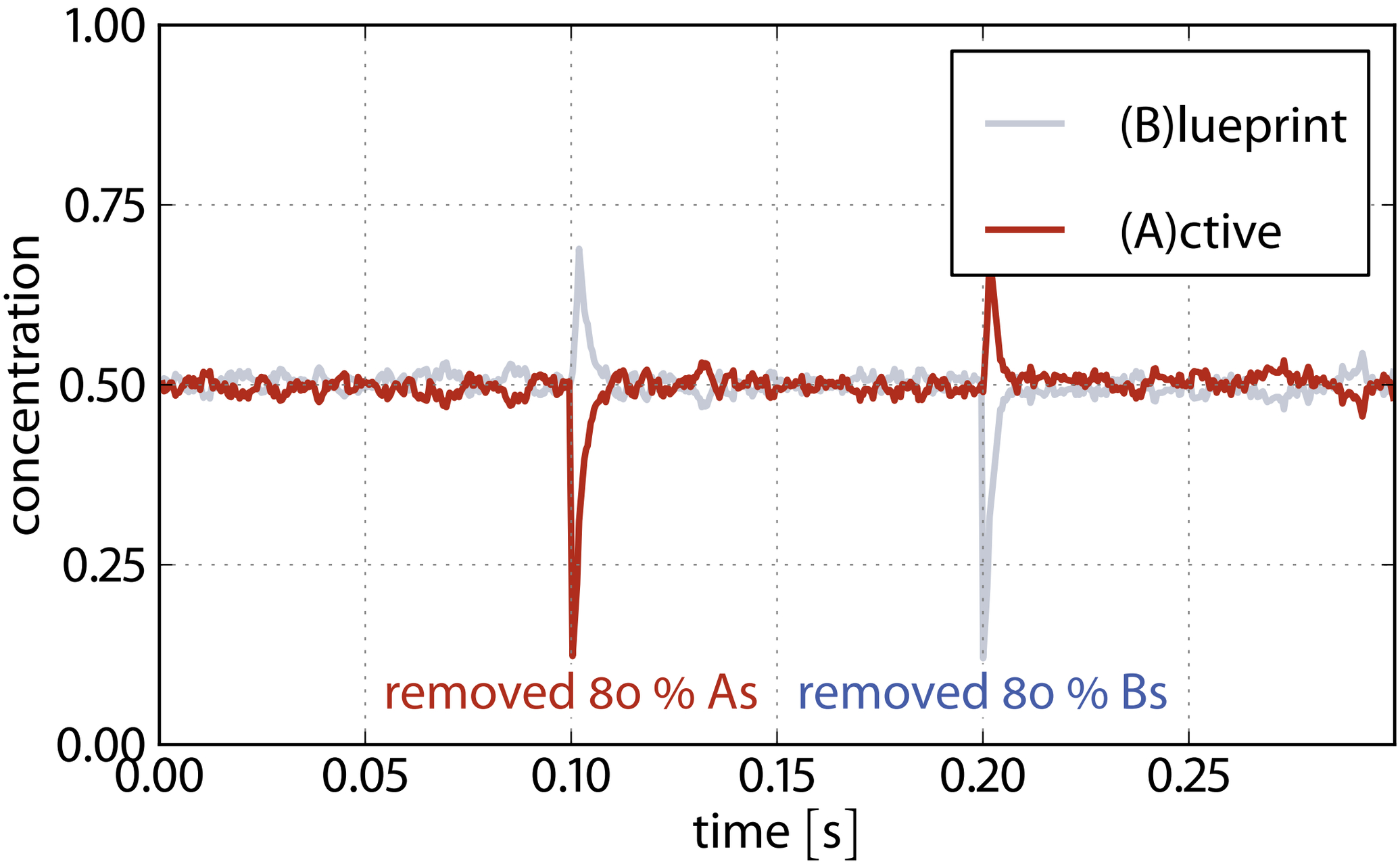
Reaction vessel
of
finite volume



until the vessel capacity is reached. In this case, newly generated molecules just overwrite randomly picked other existing molecules.

Growth & Death → Code Equilibrium

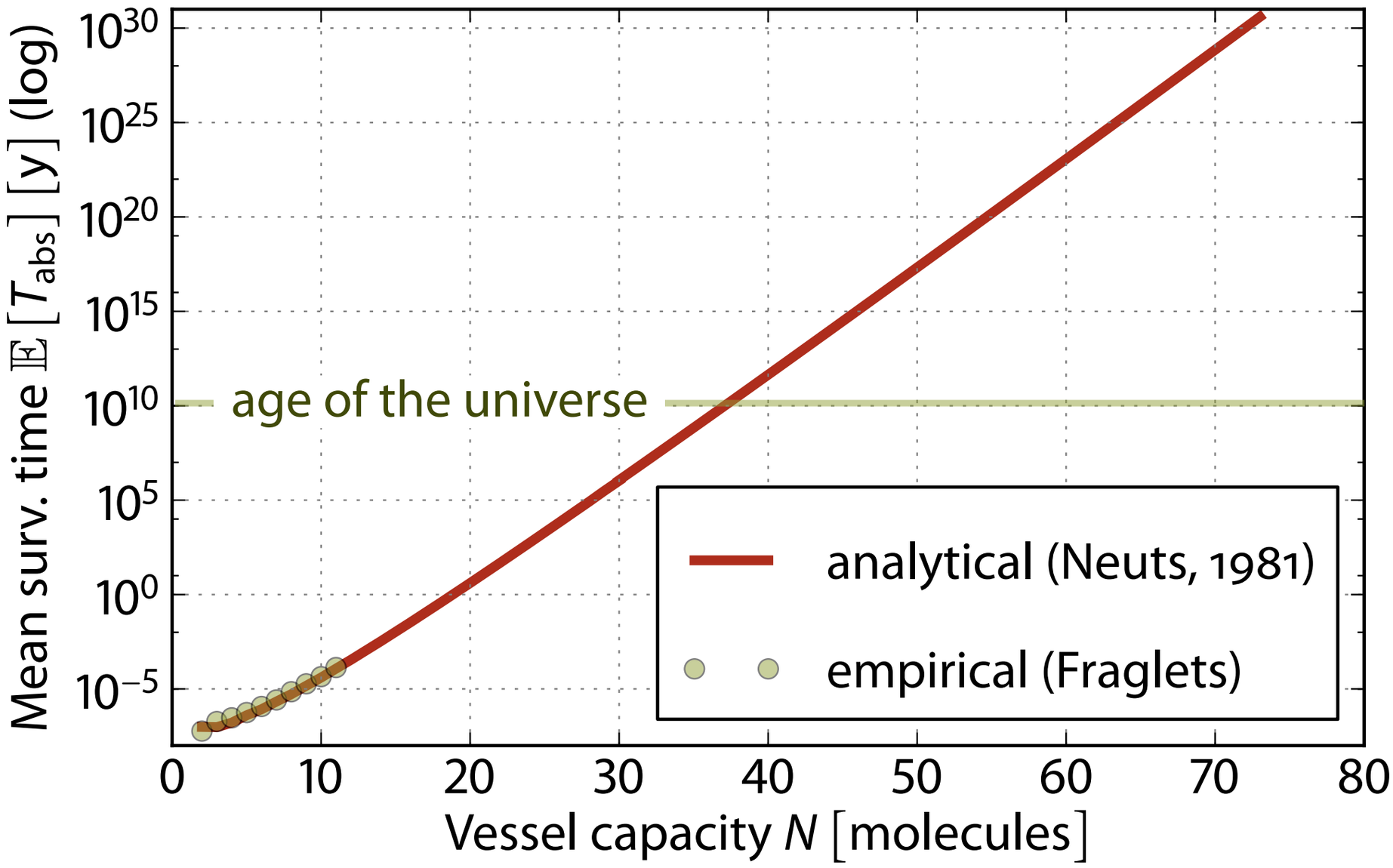
Chemical program is self-healing to the destruction of molecules



This leads to a code equilibrium in which the concentrations of both the active part and the blueprint are perfectly balanced. The Quine is also robust to deletion to large code parts. For example, here, we removed 80% of all active molecules. The remaining Quines still replicate and fill the hole in the vessel. In the first moment afterwards, there are more blueprints. However, in saturation, they are also overwritten more often such that the perfect balance is reached again.

Quantification of the Replicating Quine's Robustness

Based on the phase-type distribution of the underlying Markov process

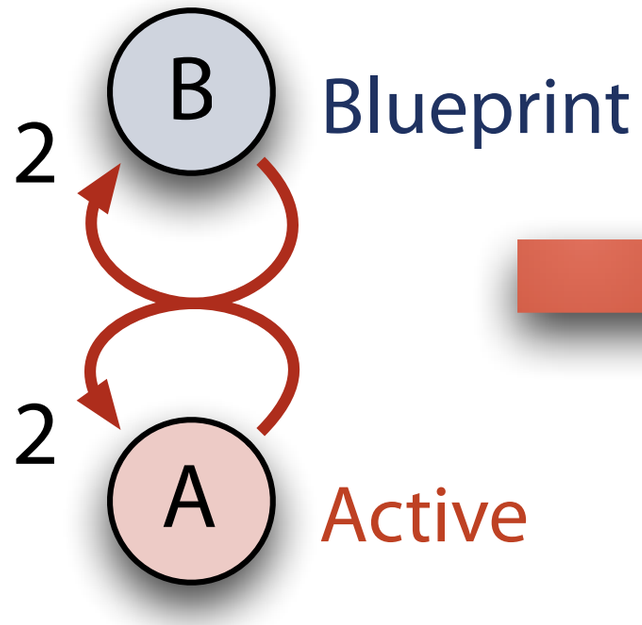


We were even able to quantify the robustness of the system by studying the underlying stochastic Markov process.

For moderate molecule deletion rates the mean survival time for small vessels of 40 molecules already exceeds the current age of the universe.

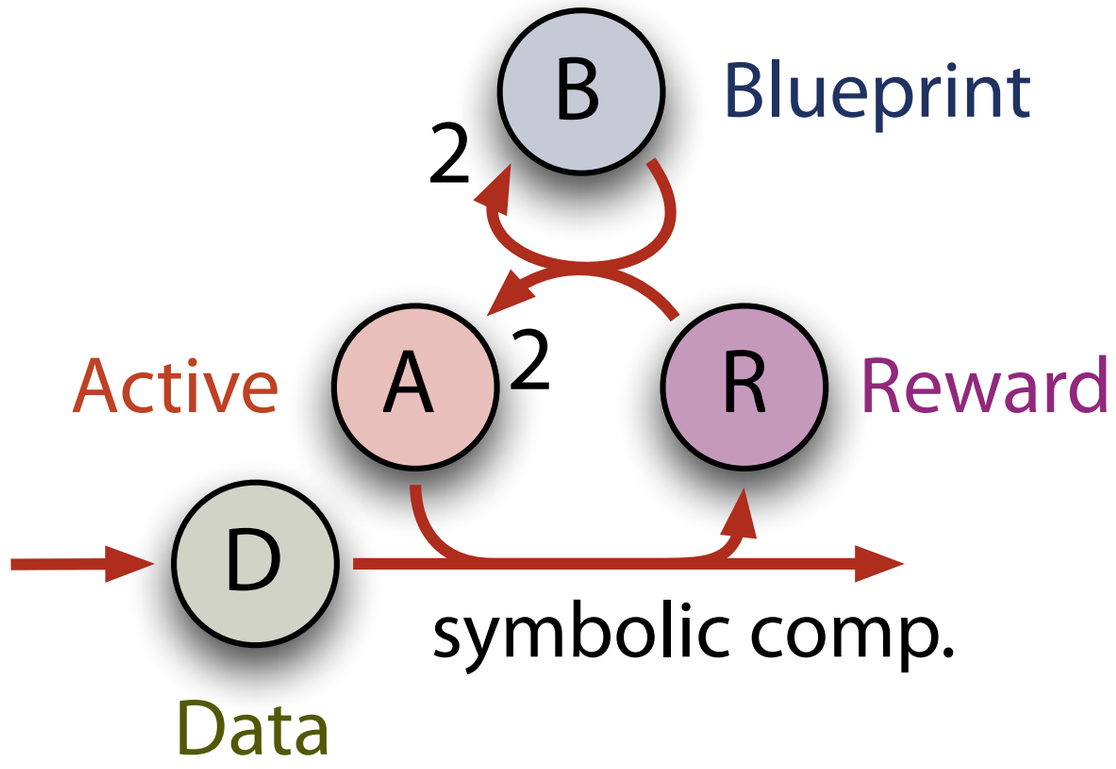
Decorating the Quine with Useful Computation

Replicating Quine



replicates autonomously

Data-Processing Quine



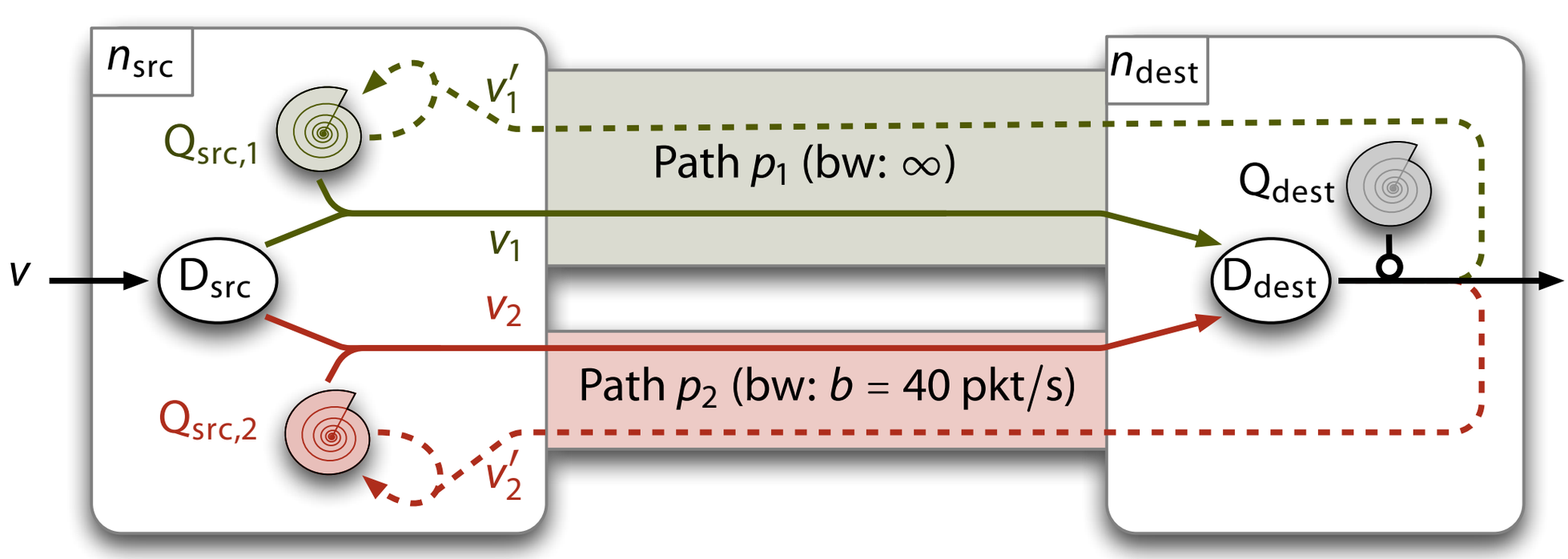
replicates when processing data

The replicating Quine only replicates without doing useful computation. By a small change, we let the active molecule first consume an input data molecule and perform some symbolic computation on it. At the same time, a reward molecule is generate, that rewards the Quine such that it generates a replica.

The data-processing Quine on the right can be used as a template to build protocol that, as a whole, is self healing. I want to demonstrate how this can be achieved with an application case of a link load-balancing protocol.

A Self-Healing Load-Balancing Protocol

Forwarding Quines compete for data packets, replicate with ACK rate



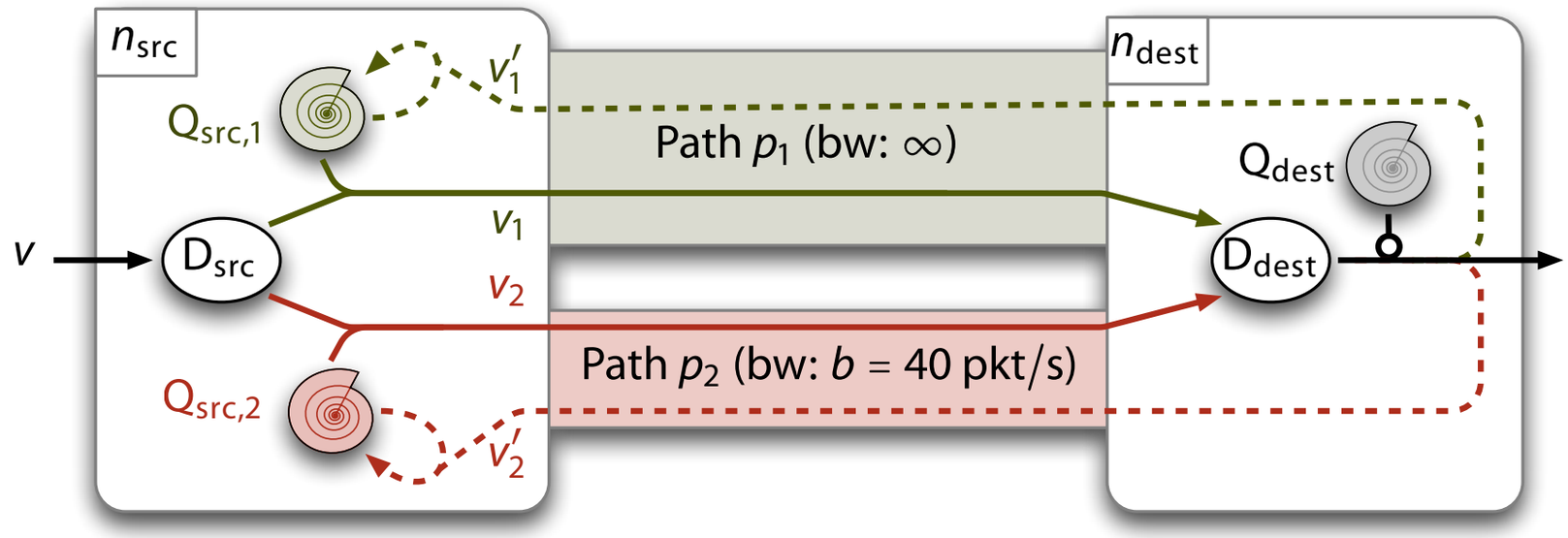
The goal is to send a data stream from source to destination over two paths, the first having an infinite bandwidth whereas the bandwidth of the second path is limited to 40 packets per second.

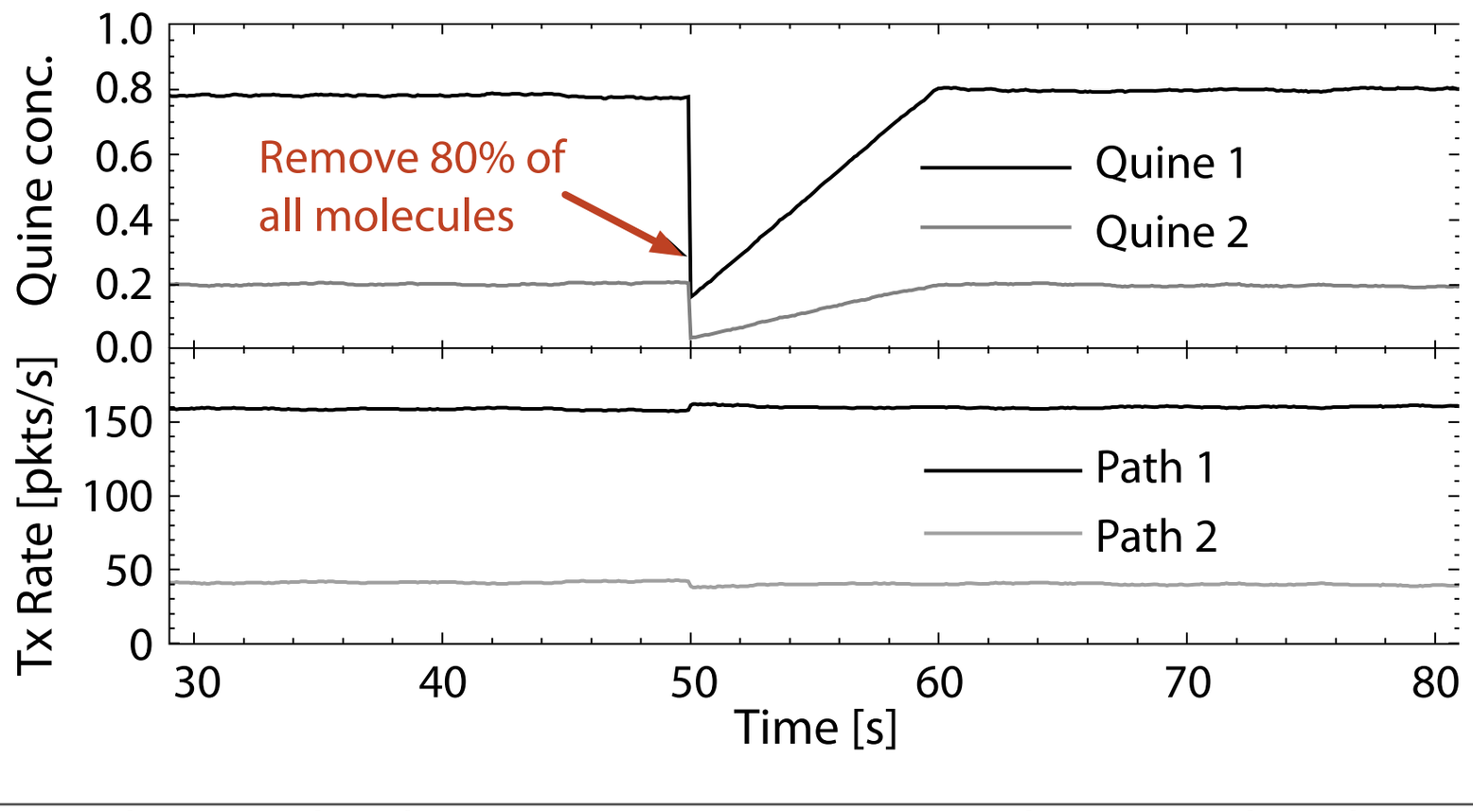
For each path, we install a Quine that competes for the packet and sends it over the corresponding path. The trick is to not let the Quine replicate immediately. Instead the acknowledgment packet generates the reward. Like this, the Quines replicate with the rate at which they receive the ACKs.

The second Quine can only grow at 40 copies per second, the first Quine faster. This leads to a lower concentration for Quine 2. Because of the law of mass action, the data packets are therefore more frequently forwarded by Quine 1 over the first path.

A Self-Healing Load-Balancing Protocol

Packets are balanced proportionally to the Quines' concentration





This can be shown in this graph. The transmission rate over the second path settles at 40 packets per second while the first path obtains the rest. The concentrations of the two Quines are distributed proportionally.

At time 50s, we removed 80% of all molecules from all vessels: not just data, but also code molecules. As we see, the system completely recovers within 10 seconds without having an affect on the forwarding performance.

Outline

Motivation

Chemical Networking Protocols

- Introductory Example / The Chemical Metaphor
- Chemical Protocol Engineering Framework
- Application Case: C3A — A Chemical Congestion Control Algorithm

Self-Healing Protocols

- Robustness to Code Deletion
- Application Case: Self-Healing Link-Load Balancing Protocol

Conclusions

I'll now come to the conclusions of my talk.

Conclusions

The **chemical engineering framework** helps to bridge the micro/macro gap in networking.

- **Analysis:** Simplified formal analysis of protocol dynamics
- **Design:** Allows to realize new protocol ideas easily, quickly, and incrementally
- **Execution:** Law of mass action scheduler promotes good equilibrium solutions

Systemic Protocol Engineering Approach

Integrated model for analysis, design, and execution of protocols

The presented work on chemical networking protocols describes an engineering framework inspired by chemistry that helps to bridge the micro/macro gap in networking.

Due to the tight link to chemical kinetics we are able to apply analysis methods from chemistry directly to chemical protocols. This allows us to prove certain dynamic properties of protocols.

The chemical reaction network graph is a tools to easily and quickly reason about new protocol designs.

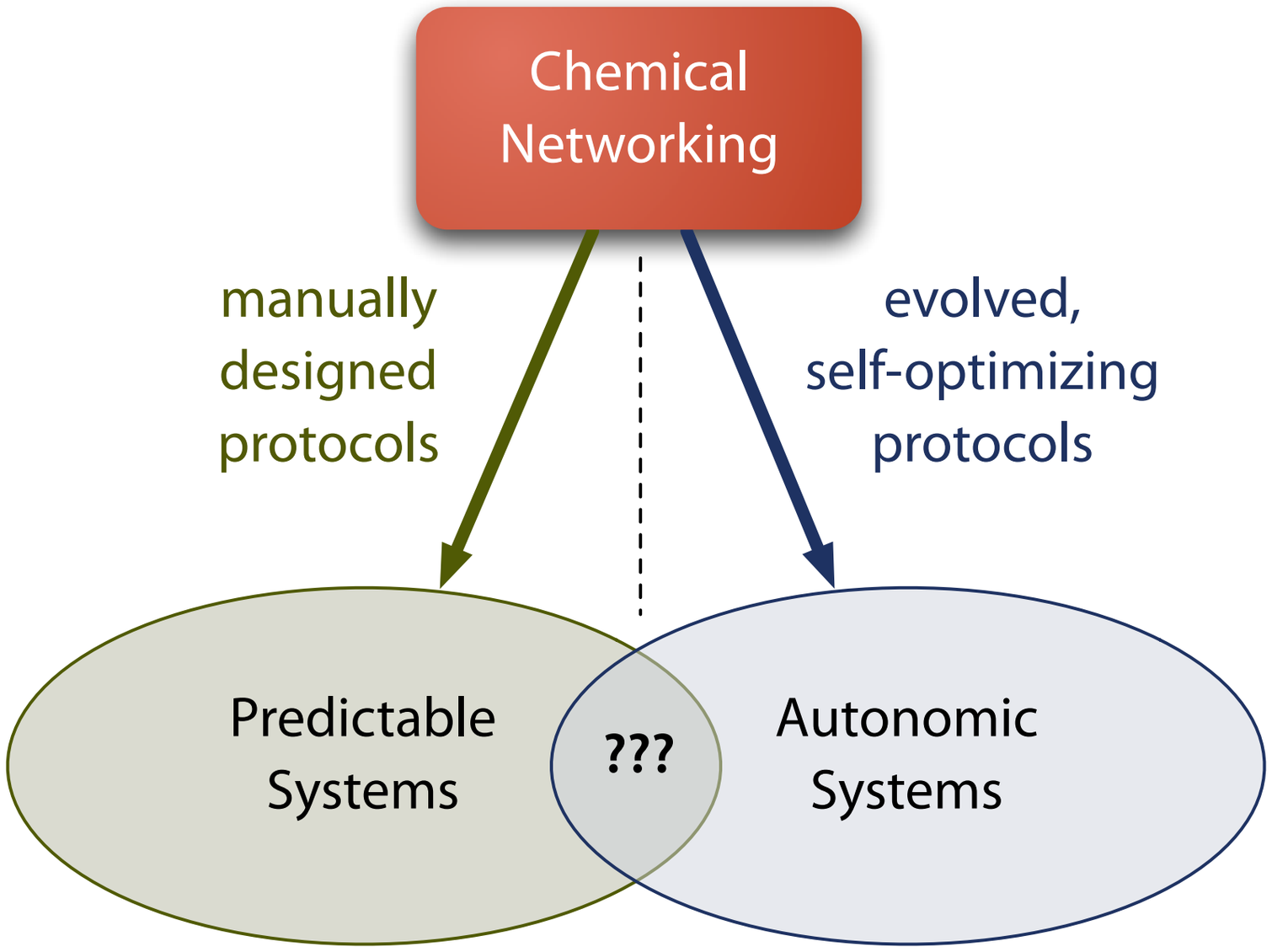
Another important aspect is that protocols can be synthesized incrementally. By continuously adding reactions to an already running solution at run-time, protocols can undergo a developmental process.

Chemical protocols are designed such that they present the solution at equilibrium. This makes the protocols stable to perturbation. The law of mass action behavior of our scheduler helps by automatically adjusting the reaction rate to concentration values.

The most important aspect of our system is that analysis, design and execution of protocols are treated within the same model. This leads to a systemic protocol engineering approach in which the gap between code and behavior is closed.

Future Networking

Predictable and/or autonomic distributed dynamical systems



I've started the talk by demanding that future networking must be predictable and autonomic.
The chemical networking approach has the potential to fulfill both needs:

By manually designing protocols we keep full control over the behavior and are able to predict certain aspects.

On the other hand, our self-healing experiments are a good starting point to investigate in the direction of evolving and self-optimizing protocols.

But only the future will show whether there is an intersection between the two - if it is possible to have fully autonomic systems that are still predictable.

Thank You! Questions?