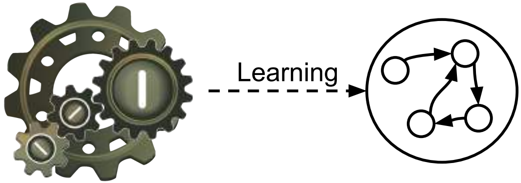
Learning and Model Checking Real-world TCP Implementations

Paul Fiterău-Broştean
Ramon Janssen

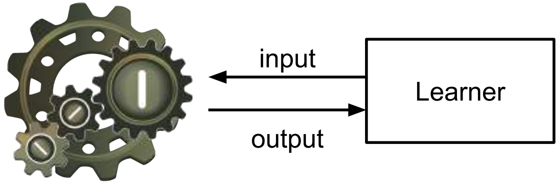
Radboud University

Automata learning



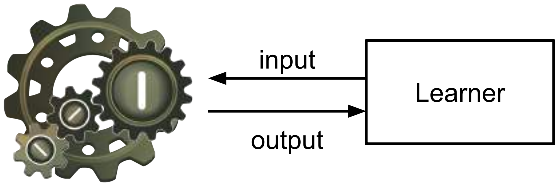
- Black box
- Infer a model automatically

Automata learning



- Black box
- Infer a model automatically
- Learner sends inputs, observes outputs

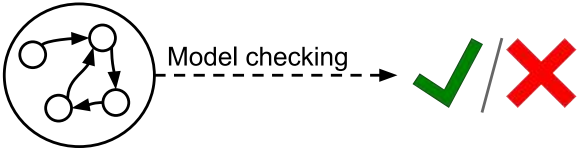
Automata learning



- Black box
- Infer a model automatically
- Learner sends inputs, observes outputs

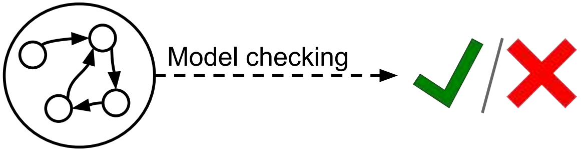
We have a model. . . Now what?

Model checking



- Automated proof
- Flexible: compose models into networks

Model checking



- Automated proof
- Flexible: compose models into networks

Problems:

- Models are often unavailable or incomplete
- Is the model consistent with the system (sut)?

Why combine them?

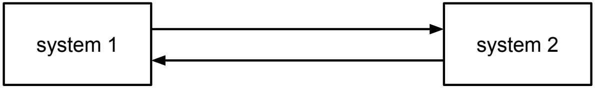
Learning found bugs already:

- E-Readers: security flaw
- TCP-implementation: non-conformance to standard

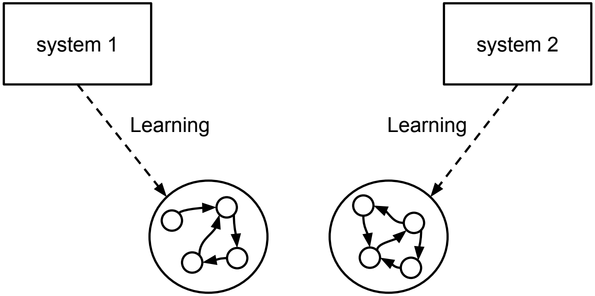
Model checking helps:

- Automatic instead of manual analysis
- Composition into networks

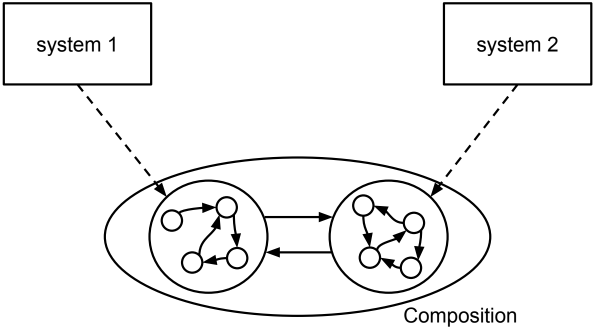
Combining model checking and learning



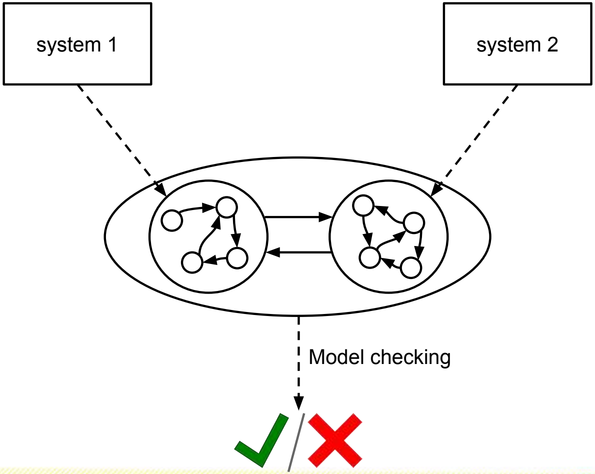
Combining model checking and learning



Combining model checking and learning



Combining model checking and learning



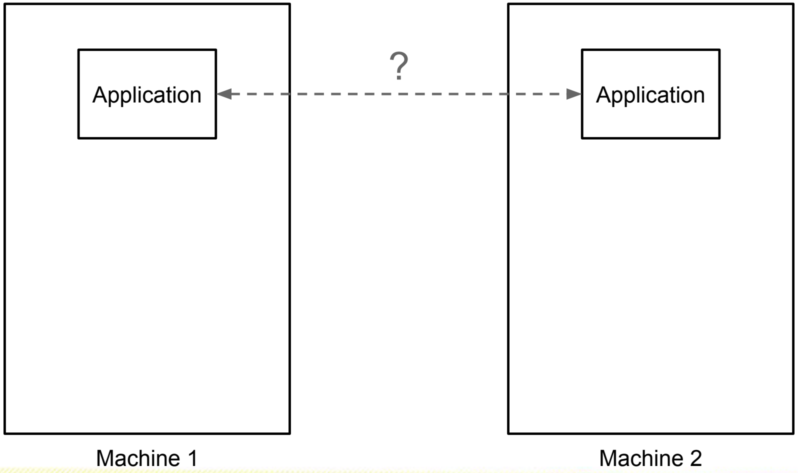
Part 1: Active learning of TCP

Learning TCP

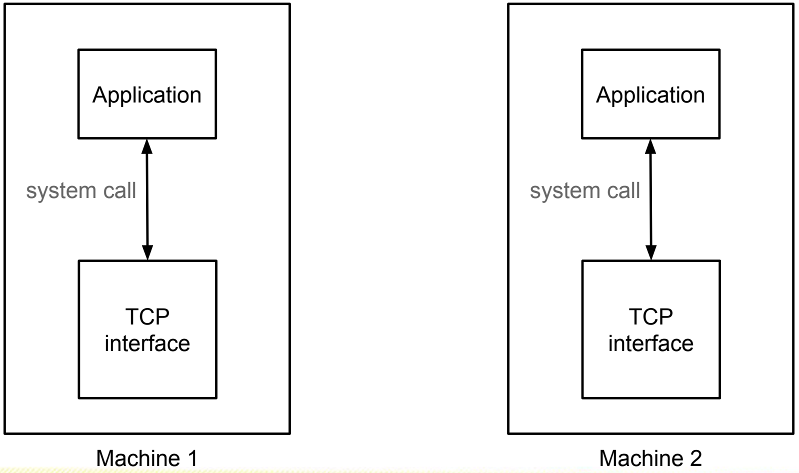
Apply learning to TCP implementations:

- TCP is a network protocol
- A client connects to a server
- TCP is reliable, messages are ordered and acknowledged

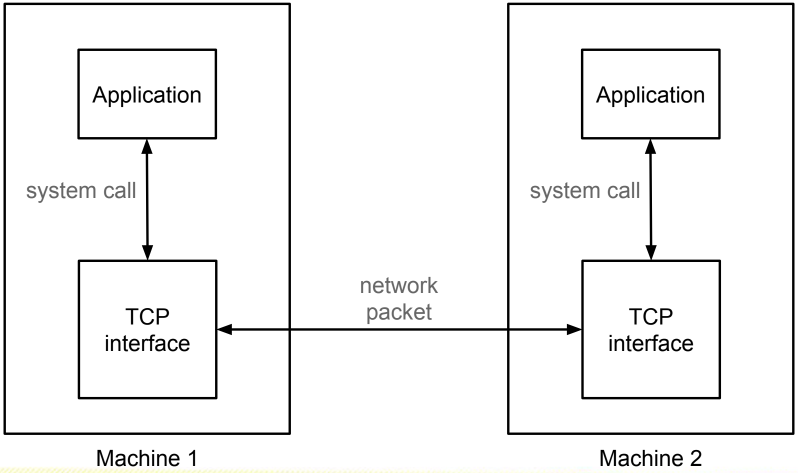
TCP as a black box



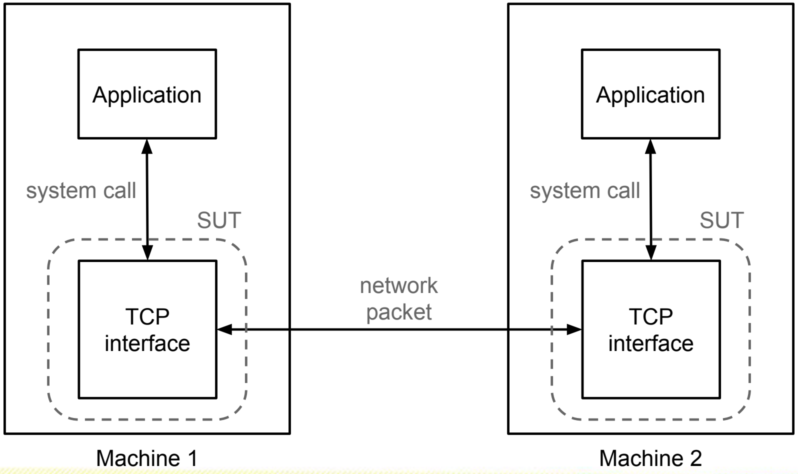
TCP as a black box



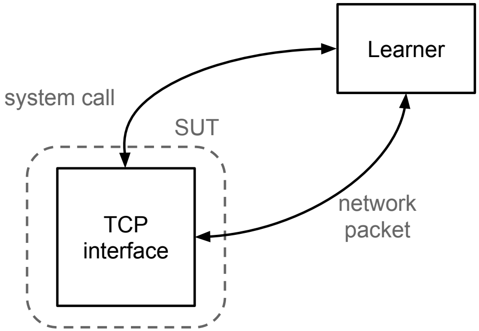
TCP as a black box



TCP as a black box



TCP as a black box



Learning TCP

Inputs:

- network packets
  - flags
  - sequence number
  - acknowledgement number

Learning TCP

Inputs:

- network packets
  - flags
  - sequence number
  - acknowledgement number
- system calls:
  - listen, accept, close (server only)
  - connect, close (client only)

Learning TCP

Inputs:

- network packets
  - flags
  - sequence number
  - acknowledgement number
- system calls:
  - listen, accept, close (server only)
  - connect, close (client only)

Outputs:

- network packets

Learning TCP

Inputs:

- network packets
  - flags
  - sequence number
  - acknowledgement number
- system calls:
  - listen, accept, close (server only)
  - connect, close (client only)

Outputs:

- network packets

(This is a restricted scope)

Problem: parameters

TCP packets have parameters

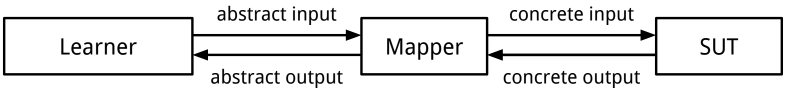
- Each packet behaves differently, depending on these values
- Number of unique inputs explodes

This makes the input alphabet too large to learn a model of TCP

Problem: parameters

Solution: abstraction

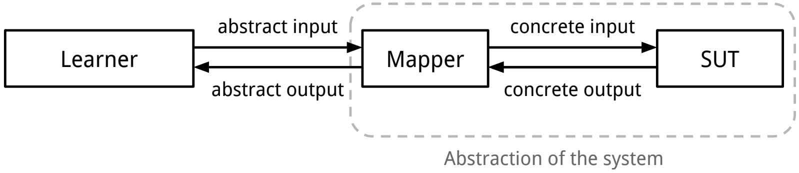
- Map large set of concrete parameters to small set of abstract parameters



Problem: parameters

Solution: abstraction

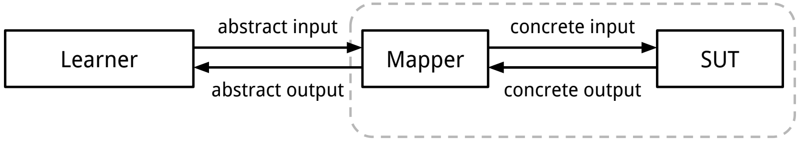
- Map large set of concrete parameters to small set of abstract parameters



Problem: parameters

Solution: abstraction

- Map large set of concrete parameters to small set of abstract parameters



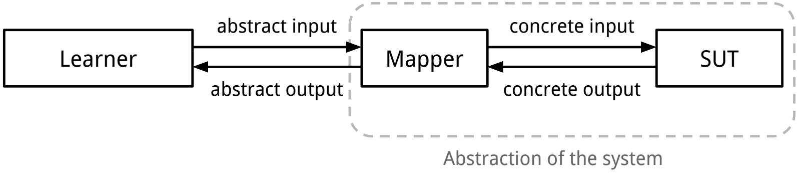
Abstraction of the system

Map input numbers to $\{valid, invalid\}$

Problem: parameters

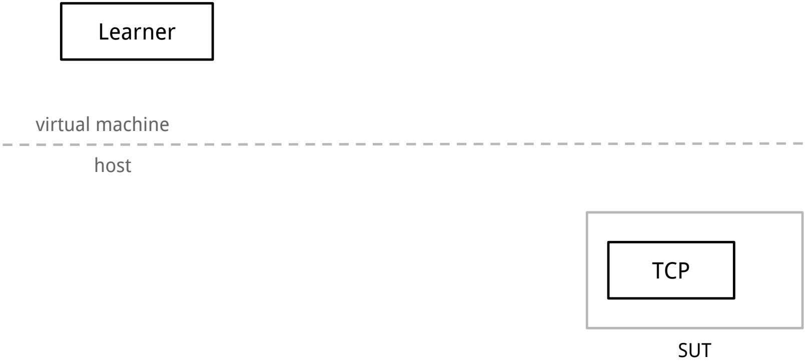
Solution: abstraction

- Map large set of concrete parameters to small set of abstract parameters

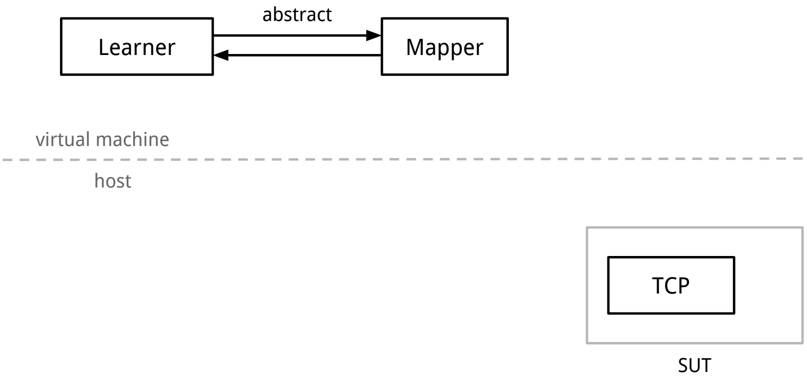


Map input numbers to $\{valid, invalid\}$
Map output numbers to $\{zero, fresh, current, next\}$

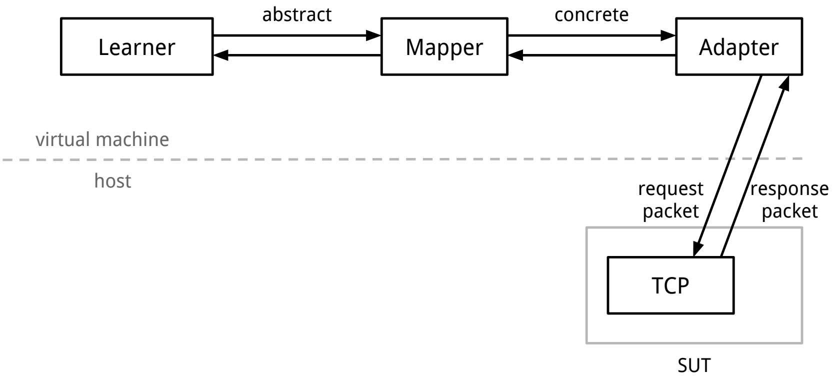
Learning setup



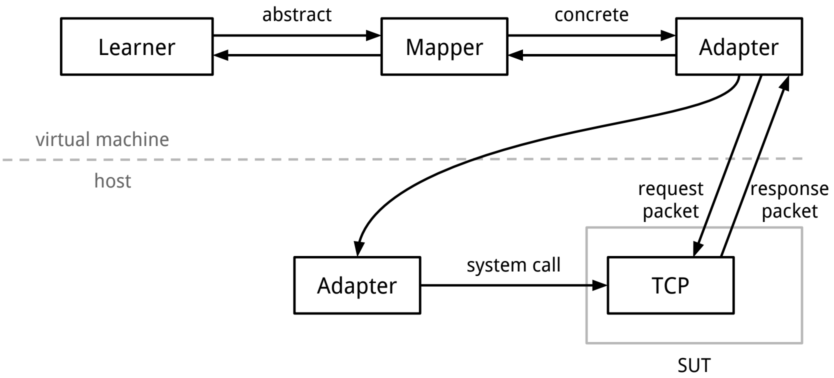
Learning setup



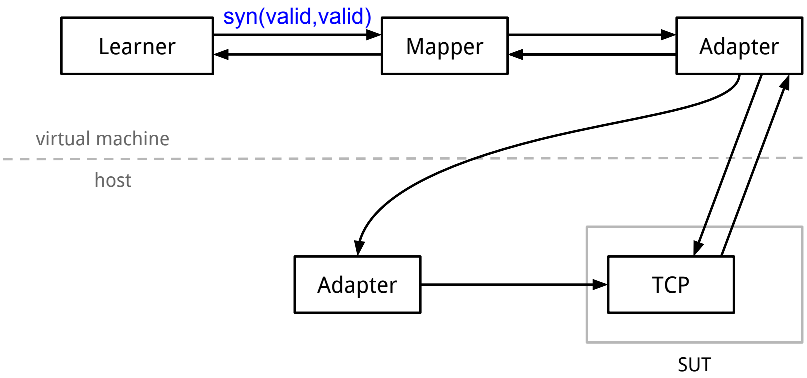
Learning setup



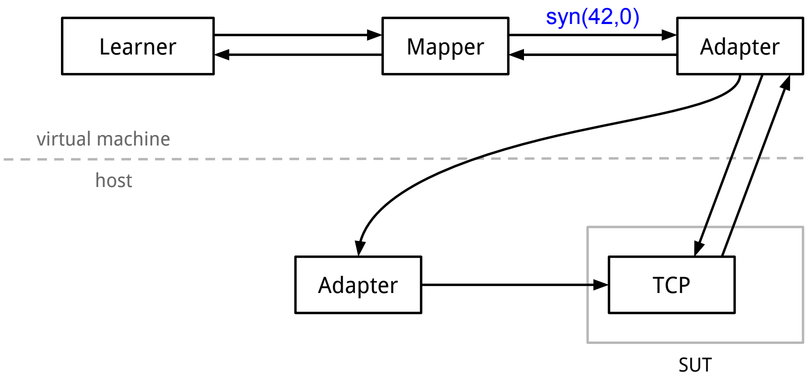
Learning setup



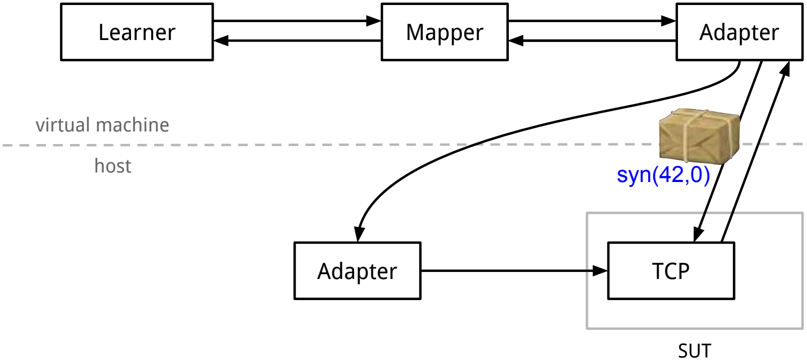
Learning example



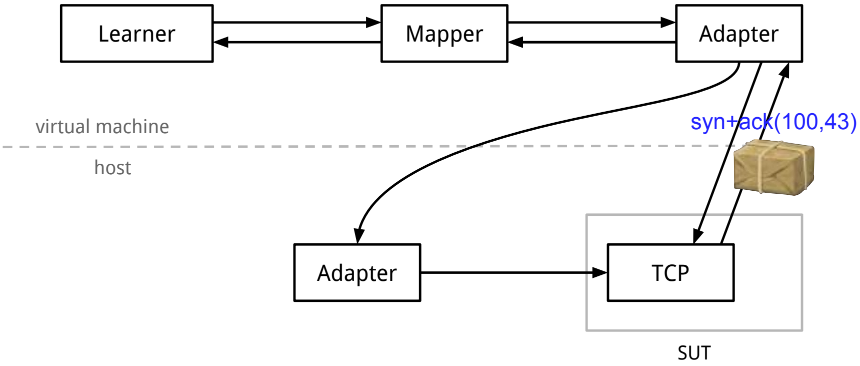
Learning example



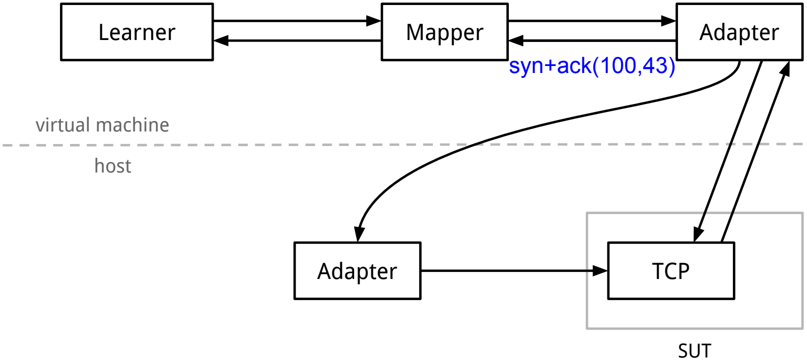
Learning example



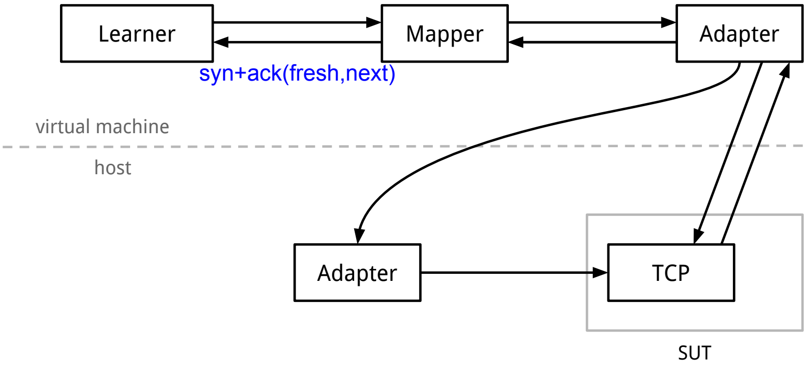
Learning example



Learning example

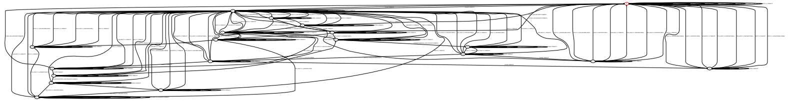


Learning example

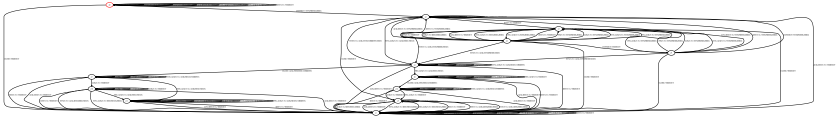


Resulting models

Server:

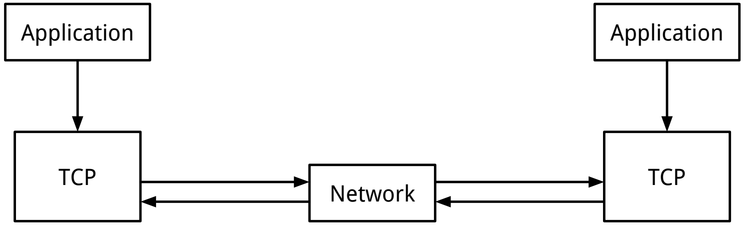


Client:

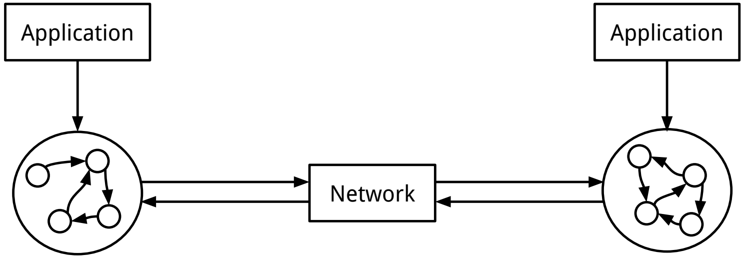


Part 2: Model checking of TCP

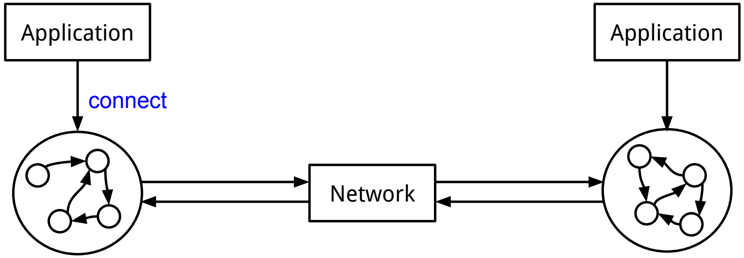
Model checking: setup



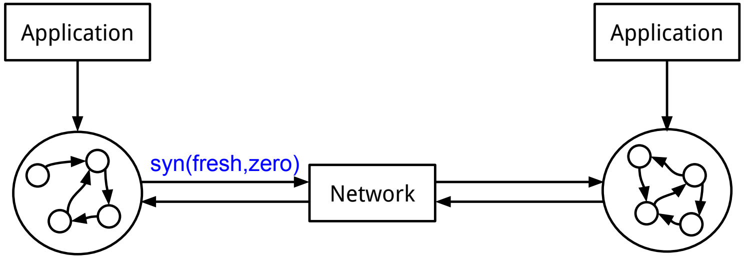
Model checking: setup



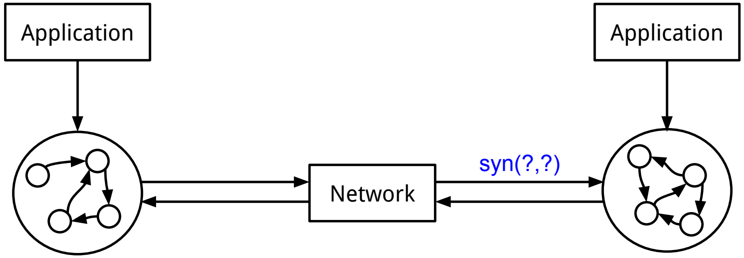
Model checking: setup



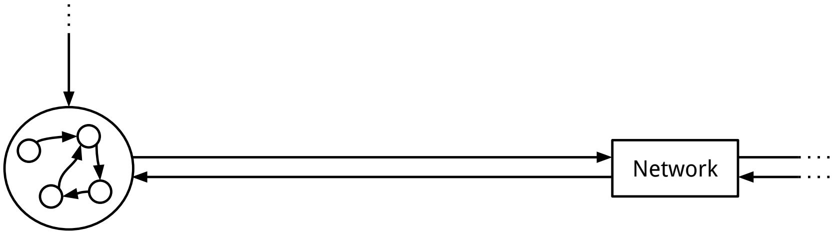
Model checking: setup



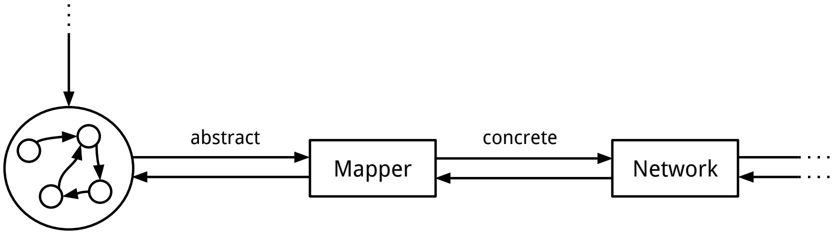
Model checking: setup



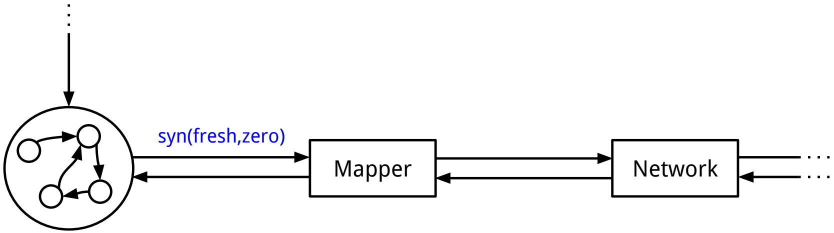
Model checking: setup



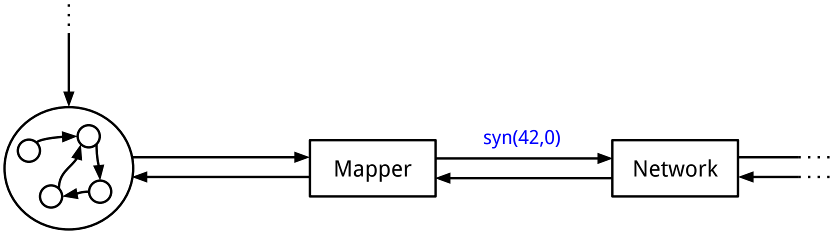
Model checking: setup



Model checking: setup



Model checking: setup



Model checking: results

Specification: only valid inputs should be sent.

Some invalid inputs were found:

1. Model incorrect
2. Implementation incorrect

Model checking: results

1. Model incorrect

- Cases which result in invalid inputs are re-tested
- The sut and model may behave differently
- This case is included in the learning for an improved model

Model checking: results

2. Implementation incorrect

- No actual bugs were found
- The tested part of TCP is shown to interact correctly

Concluding model checking learned models

- Combining the techniques works well
- We can utilize the advantages of both:
  - Testing of real system with learning
  - Composition in model checking
  - Quick and easy analysis with model checker
- Proof: only valid inputs are sent, with learned parts of TCP

To do:

- Mapping to abstract values automatically
- Define and check new properties
- Extend model: data transfer, timing...

Questions?