Verification of GALS Systems by Combining Synchronous Languages and Process Calculi

Hubert Garavel and **Damien Thivolle**

INRIA Rhône-Alpes / VASY

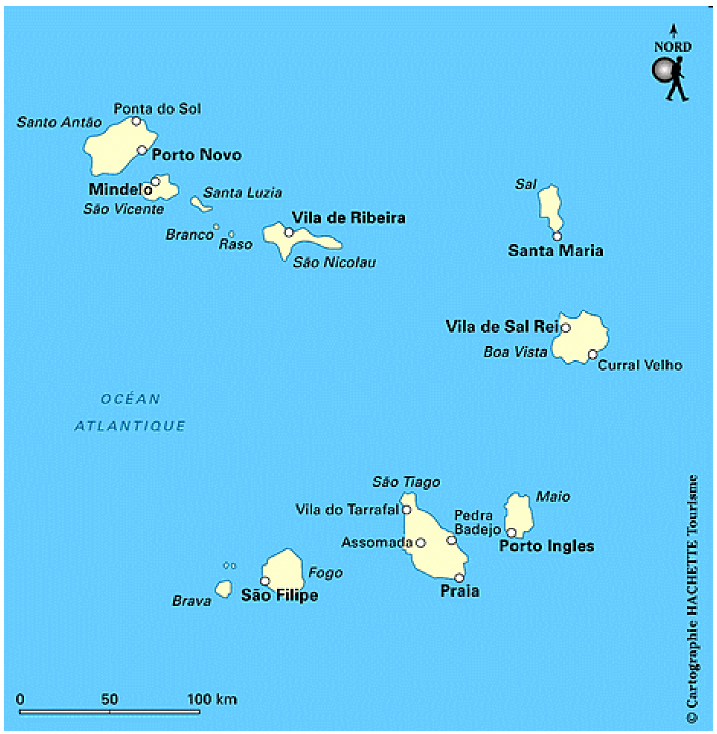
http://www.inrialpes.fr/vasy

Outline

- Motivations
- TFTP case-study and method
- Formal verification
- Simulation

Topic of this talk

- How to perform model checking of GALS?
- GALS *(Globally Asynchronous, Locally Synchronous)*



synchronous islands in
a sea of asynchrony

Model checking GALS

- Why?
  - Avionics companies use synchronous languages:
    - ESTEREL, SCADE, etc.
  - More and more, synchronous components interact with an asynchronous environment:
    - X-BY-WIRE, Modular Avionics, etc.
- Our approach:
  - Encode synchronous components as process algebras functions
  - Write wrappers around the functions for asynchronous communications

The CADP toolbox

- A toolbox for designing asynchronous systems:
  - compilers
  - model checkers
  - equivalence checkers
  - simulation, rapid prototyping, test case generation…
  - performance evaluation
- Developed by the VASY team of INRIA Grenoble
- 43 tools, 18 libraries, 100+ case studies
- 8 supported architectures (32- and 64- bits)
- Important user community (forum…)
- Licensed to several big companies

The TFTP case study

The TFTP case study

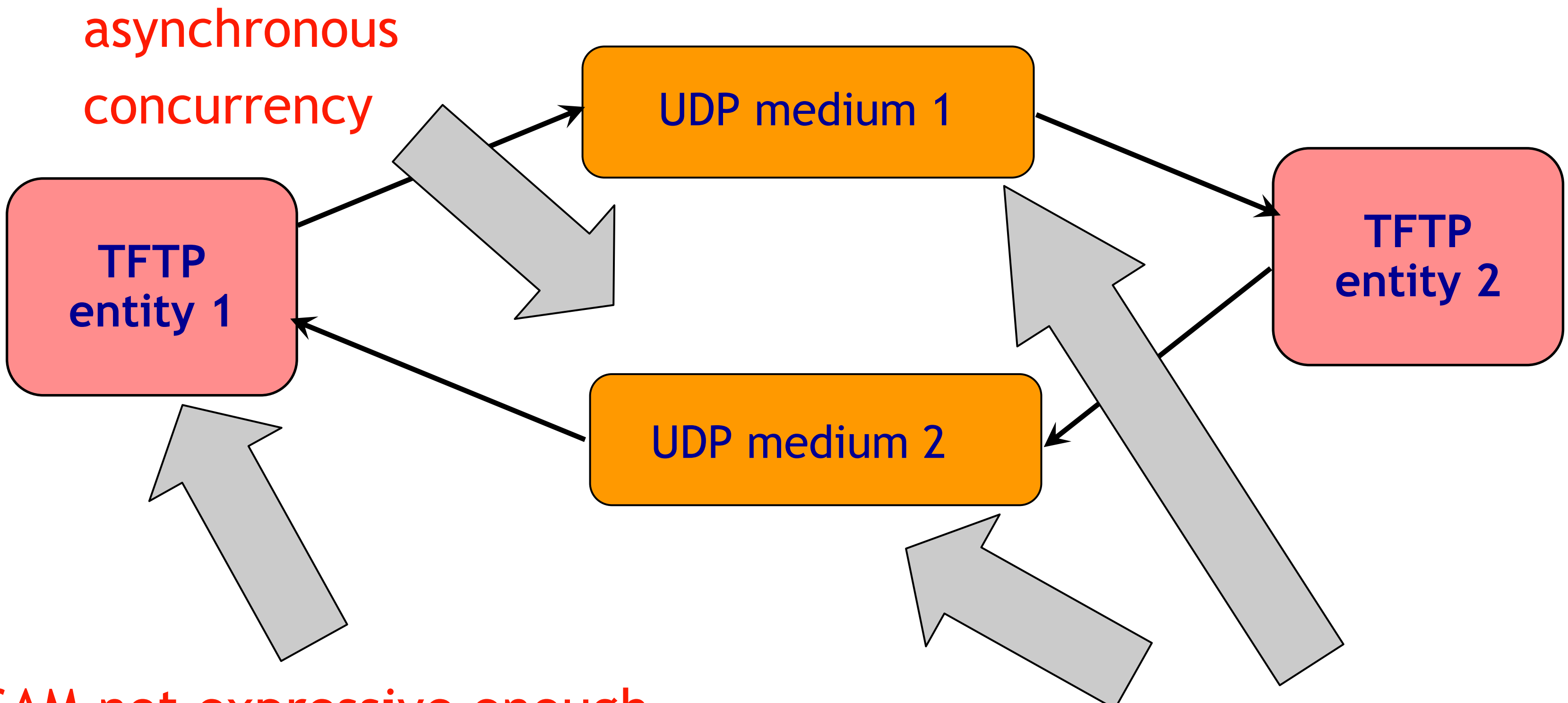
- A real example provided to us by Airbus
- Communications between plane and ground
- A three layer protocol stack
  - ARINC protocol
  - TFTP (Trivial File Transfer Protocol)
  - UDP (datagrams over IP)
- Very "light" specification:
  - SAM automaton (7 states, 39 transitions)
  - two TFTP entities connected head-to-tail via UDP

What is SAM?

- A graphical language defined/used by Airbus
- Inspired from F. Maraninchi's Argos language
- A synchronous language:
  - boxes connected by arrows (synchronous parallel, no causality loops)
  - each box is a synchronous automaton
  - boolean inputs/outputs
  - determinism (priority between transitions)
- Reference manual written by VASY:

  http://gforge.enseeiht.fr/docman/view.php/33/2745/SAM.pdf

SAM limitations as seen on the TFTP



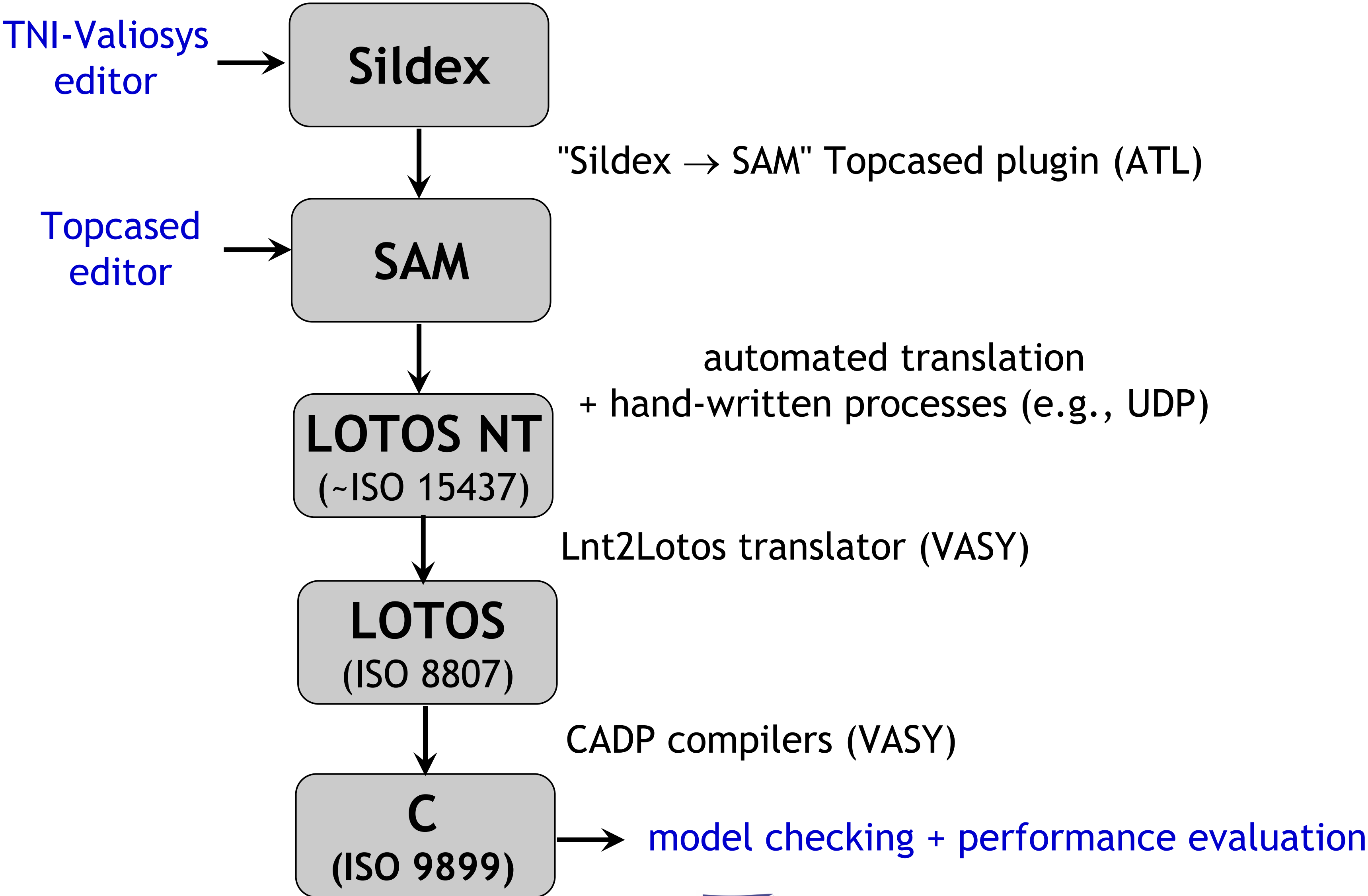
SAM not expressive enough
to describe non-boolean computations
- message contents
- timeout values

UDP is nondeterministic
- messages can be lost
- message order is not preserved

Chosen methodology

- SAM only models a fragment of the problem
- To model and analyze the complete TFTP, we need an asynchronous language
- Several attempts made (FIACRE, LOTOS…)
- Best solution chosen: LOTOS NT
  - a subset of the ISO standard E-LOTOS
  - funded and used by Bull

Tool chain


TNI-Valiosys editor
Sildex
"Sildex → SAM" Topcased plugin (ATL)
Topcased editor
SAM
automated translation
+ hand-written processes (e.g., UDP)
LOTOS NT
(~ISO 15437)
Lnt2Lotos translator (VASY)
LOTOS
(ISO 8807)
CADP compilers (VASY)
C
(ISO 9899)
model checking + performance evaluation

From SAM to LOTOS NT functions

- Synchronous parallelism in SAM ≠ Asynchronous parallelism in FIACRE, LOTOS…
- Each SAM automaton is translated to a sequential Mealy function
  f (*current_state*, *inputs*) → (*next_state*, *outputs*)
- Synchronous composition of SAM automata is implemented by a composition of the corresponding sequential functions

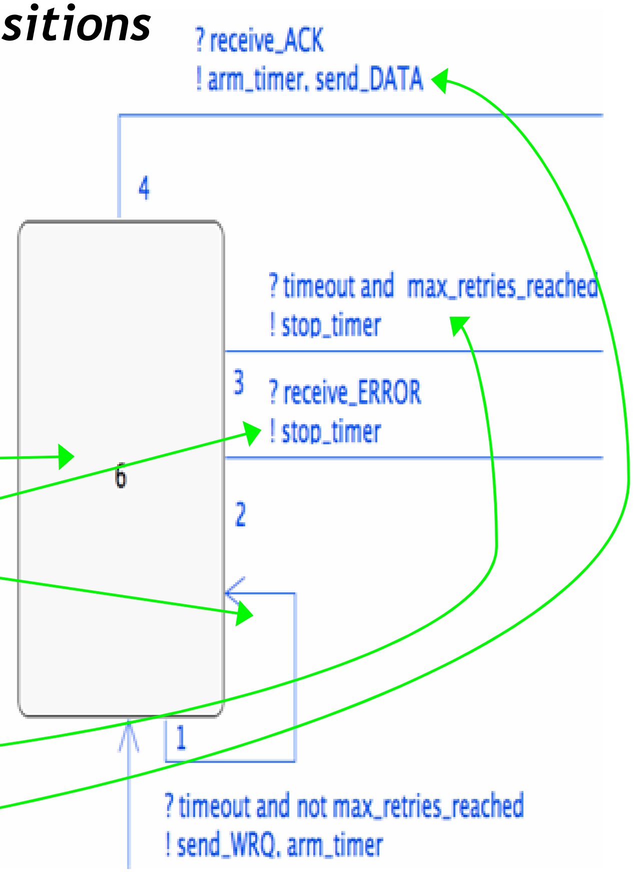
(in the TFTP, only one SAM automaton)

From SAM to LOTOS NT functions

LOTOS NT:
215 lines
of code

SAM:
7 states
39 transitions

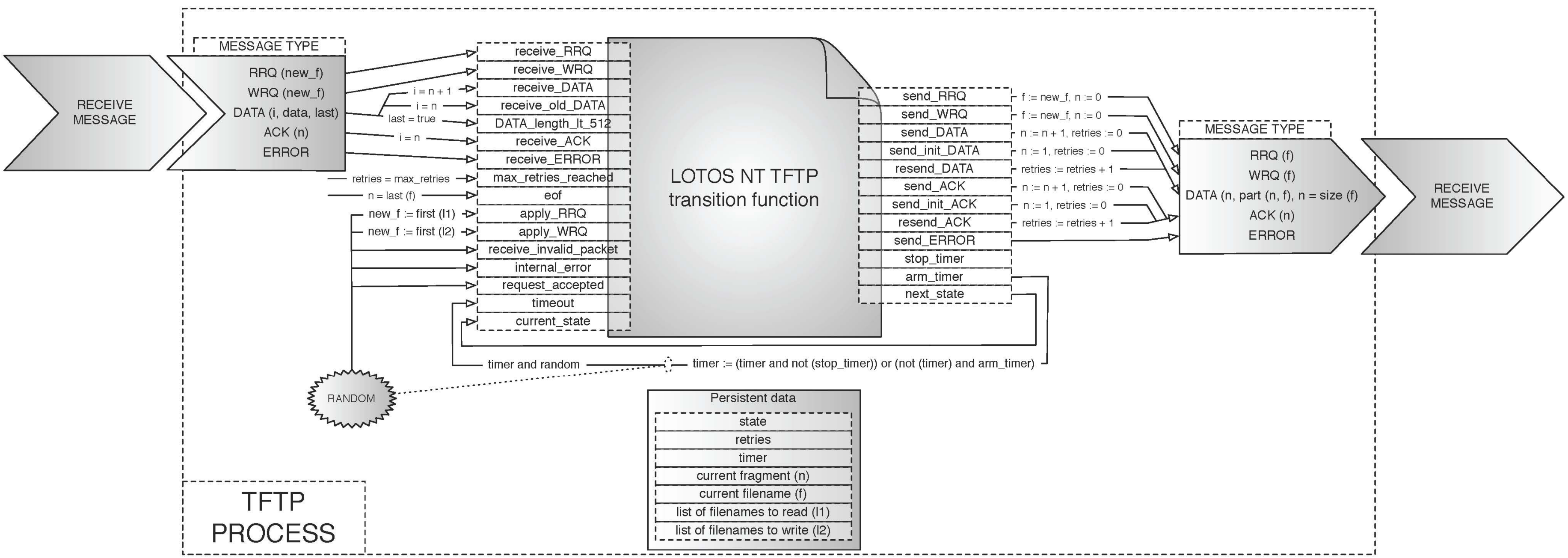
```
function transition (in current:state,
              in receive_ACK:bool,
              in receive_ERROR:bool,
              …
              in timeout:bool,
              in max_retries_reached:bool,
          out next:state,
              out send_DATA:bool,
              out send_WRQ:bool,
              …
              out arm_timer:bool,
              out stop_timer:bool) is

(* Init, out variables := false *)
…
if current == STATE_6 then

 if timeout and not (max_retries_reached) then
      send_WRQ := true;
      arm_timer := true;
      next := current
 elsif receive_ERROR then
      stop_timer := true;
      next := STATE_1
 elsif timeout and max_retries_reached then
      stop_timer := true;
      next := STATE_1
 elsif receive_ACK then
      arm_timer := true;
      send_DATA := true;
      next := STATE_2
```



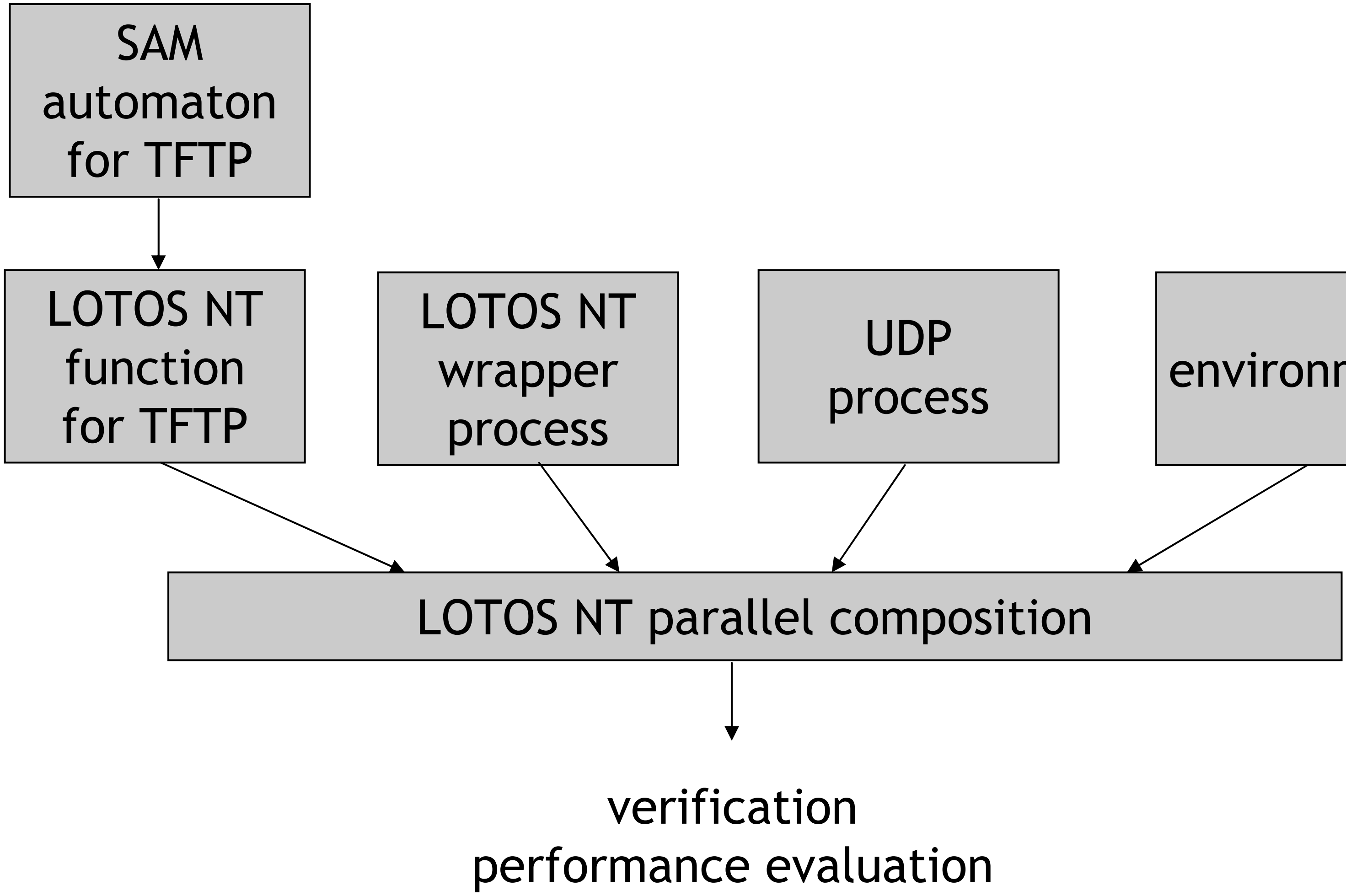
Wrapping functions into LOTOS NT processes

Mealy functions encapsulated into processes:

- converting boolean variables into I/O events
  - Transforms a Mealy function into an LTS
- *adding non-boolean code not described in SAM*

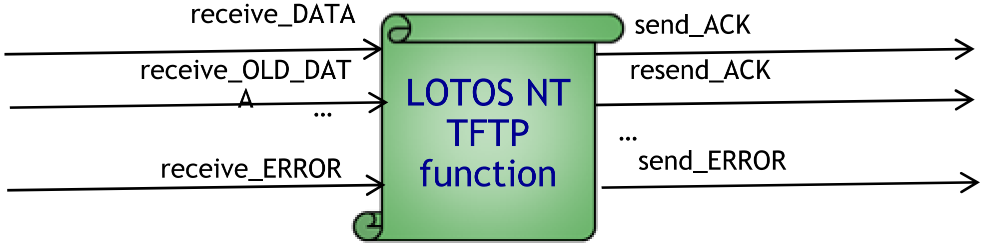


Global view

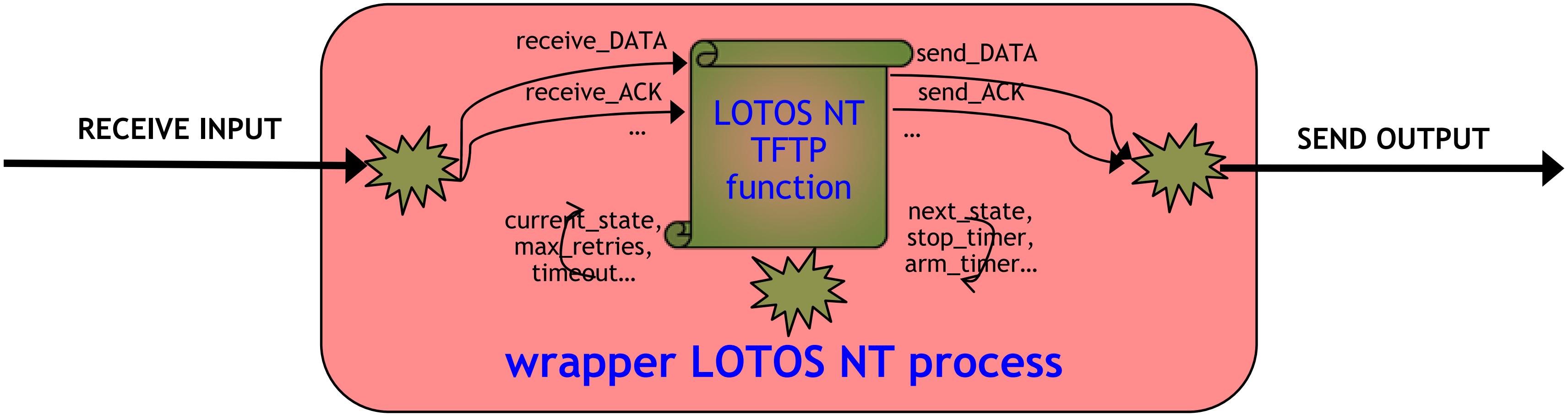

SAM automaton for TFTP
LOTOS NT function for TFTP
LOTOS NT wrapper process
UDP process
environment
LOTOS NT parallel composition
verification
performance evaluation

Verification
of the "basic" TFTP

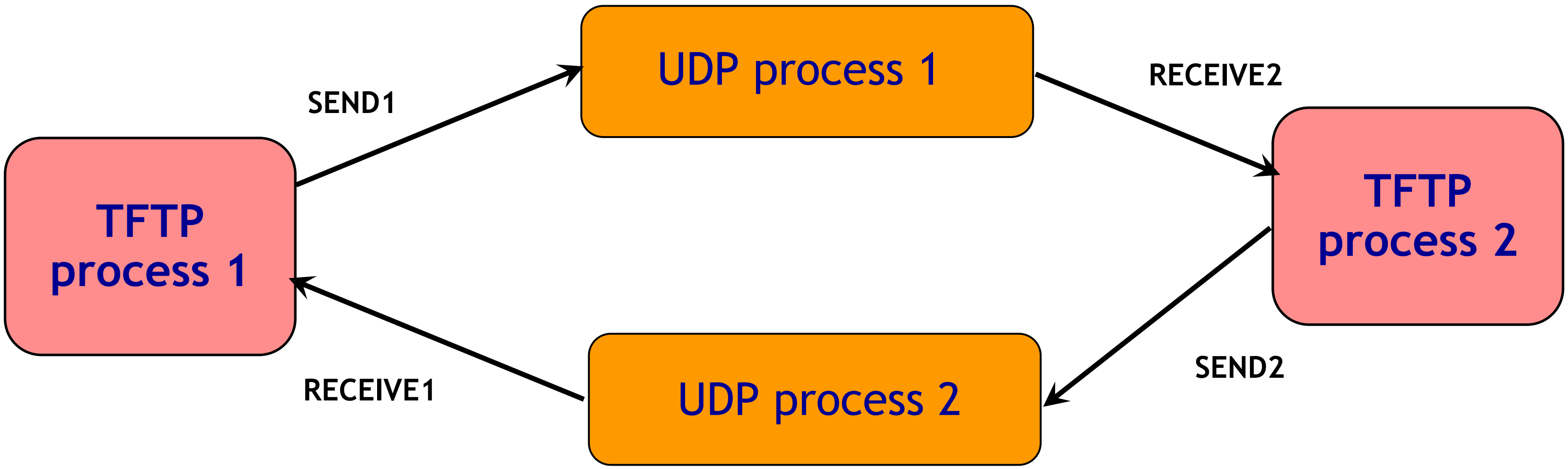
Encapsulation of the SAM automaton



- This function is encapsulated into a wrapper LOTOS NT process
- This wrapper is very simple (193 lines): messages do not carry data



The whole TFTP protocol



- UDP entities are modelled in LOTOS NT too:
  - bounded FIFOs (messages lost)
  - bounded BAGs (messages lost or re-ordered)
- The 4 processes execute asynchronously
- LOTOS NT parallel composition is used for this

State space generation

- Successive steps: LOTOS NT → LOTOS → LTS
- Direct state space generation not efficient
- Compositional generation used instead:
  - each sequential process is minimized
  - minimized processes are recombined

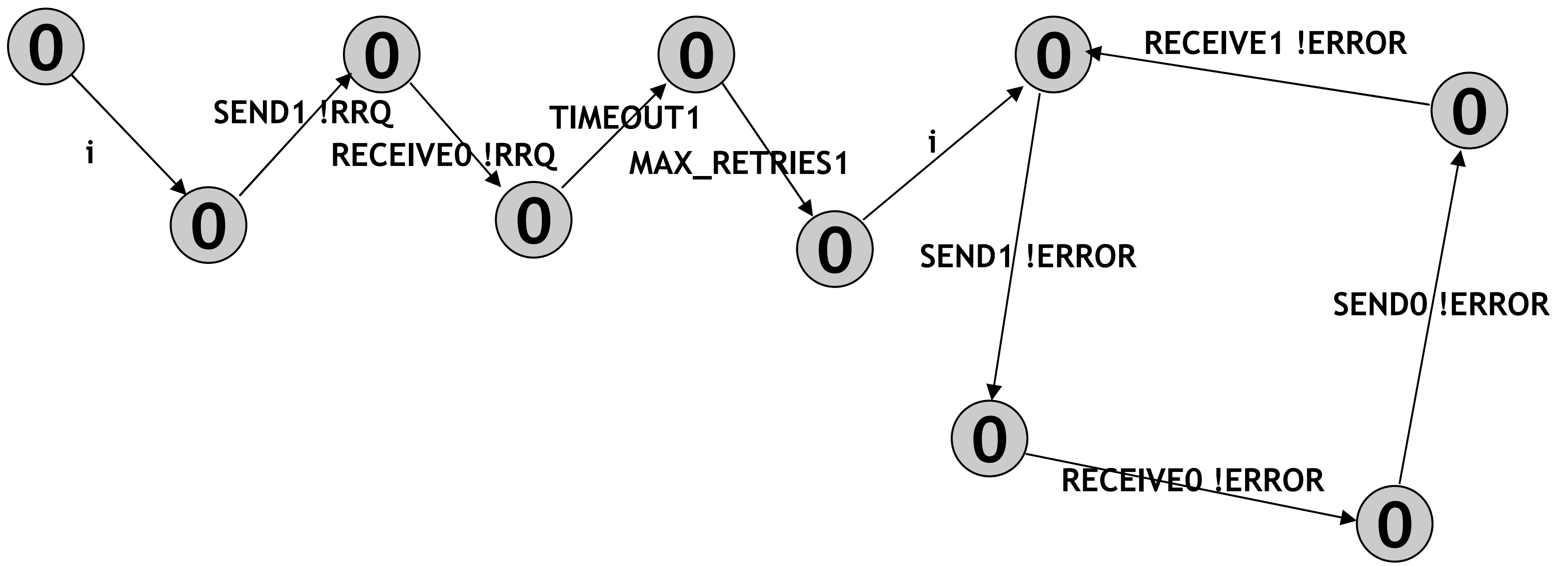
Example with two UDP FIFOs of size 2:
  - 846,888 states
  - 3.7 million transitions
  - compositional generation: 15 s

Model checking verification

- 12 properties expressed with Evaluator
- 8 problems detected:
  - timer does not stop after a transfer is finished
  - after loss of final ACK, resent DATA is ignored
  - new transfer impossible right after final ACK
  - invalid packets and invalid acknowledgements are simply ignored whereas they should abort the transfer
  - ...

Generation of diagnostics

- Property "absence of error loop" not satisfied
- Diagnostic generated by breadth-first search (in 0.53 s):



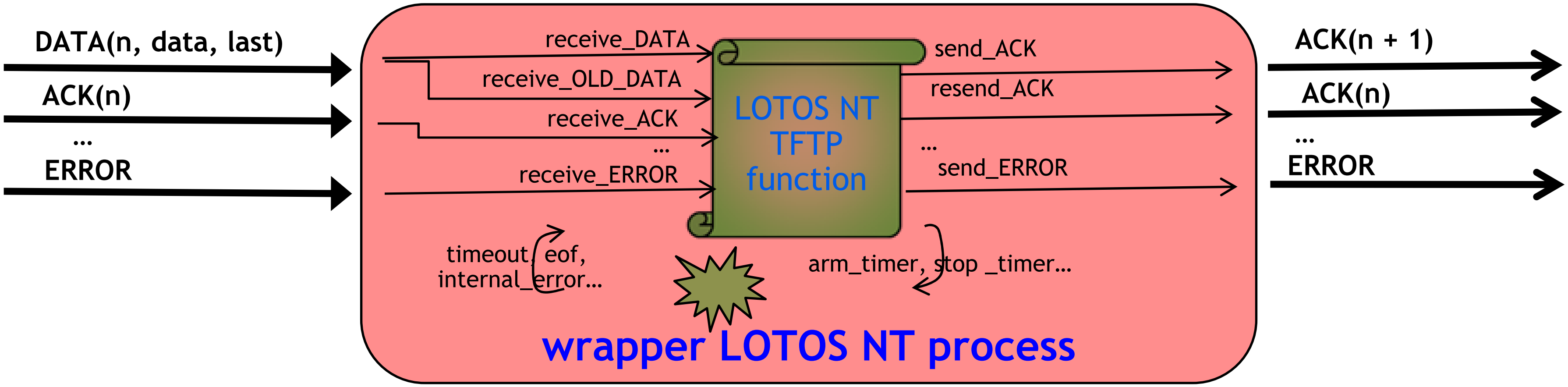
Verification
of the "accurate" TFTP

Limitations of the "basic" verification

- The SAM model of the TFTP does not express certain important details:
  - only Boolean variables
  - not represented: counters, number of retries, packet contents, fragment numbers, list of files to be sent or received, etc.
- There are properties that cannot be expressed
  - Example:
    ACK (x) cannot be received before ACK (x–1)

"Accurate" TFTP modeling

- The SAM automaton (encoded as a LOTOS NT function) is kept unchanged (215 lines)
- But it is encapsulated in a more elaborate "wrapper" written manually (418 lines) based upon knowledge of the TFTP standard



State space generation

- CADP tools used to generates the space state for various configurations
- Example:
  - TFTP entity 1 has one file to write
  - TFTP entity 2 has one file to read
  - two UDP FIFOs of size 2
  - 44 million states, 221 million transitions
  - compositional generation = 24 mn 22'
- Verified up to size 3 for the UDP FIFOs and BAGs

Results

- 12 formulas of “basic” specification

+ 17 new formulas

- 8 new problems detected:
  - any old DATA received is not acknowledged
  - when initiating a write, receipt of an invalid packet is ignored
  - when initiating a read, receipt of an invalid packet is ignored
  - if both processes send RRQ or WRQ at the same time, their requests will not be answered
  - INTERNAL_ERROR is ignored in several cases
  - etc .

Performance evaluation

Performance issues

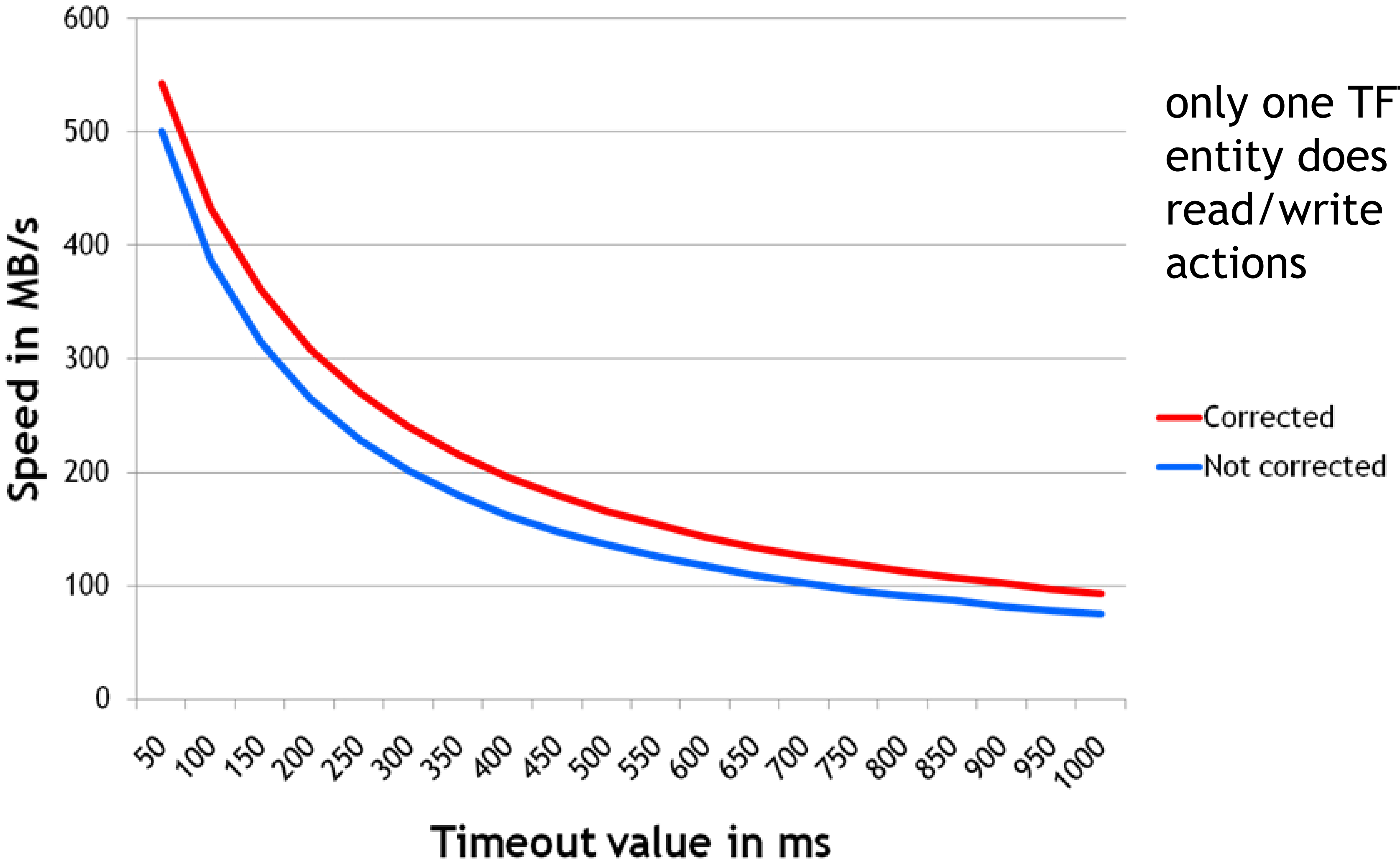
- Do problems in TFTP specification affect runtime performance?
- If so, how much?

- Example: problem 08 (infinite error loop)
  - in many cases, one may exit this error by reinitializing the TFTP entities after a timeout
  - but timeout reinitialization causes performance degradation
  - can we quantify this degradation?

A simulation-based approach

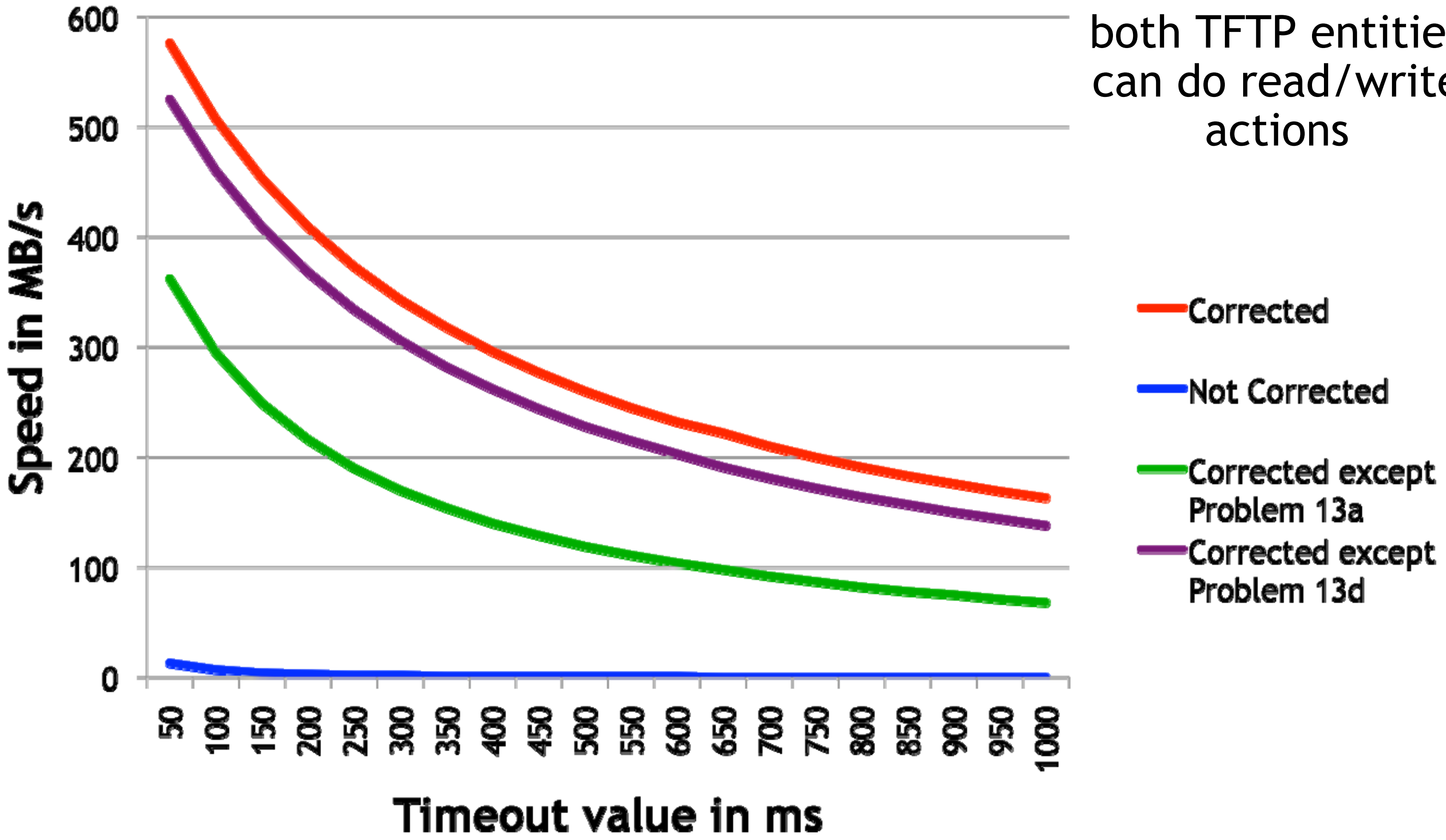
- Instrument the Executor tool of CADP
- Generate random execution traces
- Measure TFTP transfer speed on these traces
- Different scenarios:
  - scenario 1: one TFTP entity does read/write
  - scenario 2: both TFTP entities do read/write
- Chosen TFTP parameters:
  - 10,000 files written or read in each scenario
  - packet size: 32 kB
  - medium speed: 1 MB/s
  - medium latency: 8 ms
  - medium losses: 1%

Performance impact of problems (scenario 1)



only one TFTP entity does read/write actions

Performance impact of problems (scenario 2)



both TFTP entities can do read/write actions

Conclusion

Summary of TFTP results

- Verification of the "basic" TFTP:
  - 12 properties checked
  - 8 errors detected
- Verification of the "accurate" TFTP:
  - 29 properties checked
  - 19 errors detected
- Performance evaluation:
  - confirms quantitative issues
  - done by simulation, but other approaches exist in CADP (tools for Markov chains)

Conclusion

- Model checking of GALS
  - reuse synchronous processes (written in Sildex/SAM) composed asynchronously (in LOTOS NT/LOTOS)
  - verification and performance evaluation with CADP
  - a sound solution for validating GALS
  - mostly automated
- Positive feedback from Airbus
  - apreciated flow combining Topcased, ATL, and CADP
    - based on formal transformations
    - 5 languages: Sildex $\rightarrow$ SAM $\rightarrow$ LOTOS NT $\rightarrow$ LOTOS $\rightarrow$ C (+ MCL)
  - ongoing collaboration on a new avionics application