ERICSSON

test competence center

Simulation and load testing with TTCN-3 Mobile Node Emulator

ETH/RBD Zsolt Torpis +36 1 437-7731

zsolt.torpis@ericsson.com

By Zsolt Torpis, Tibor Szabó and Sarolta Dibuz

TTCN-3 User Conference

3-5th May 2004

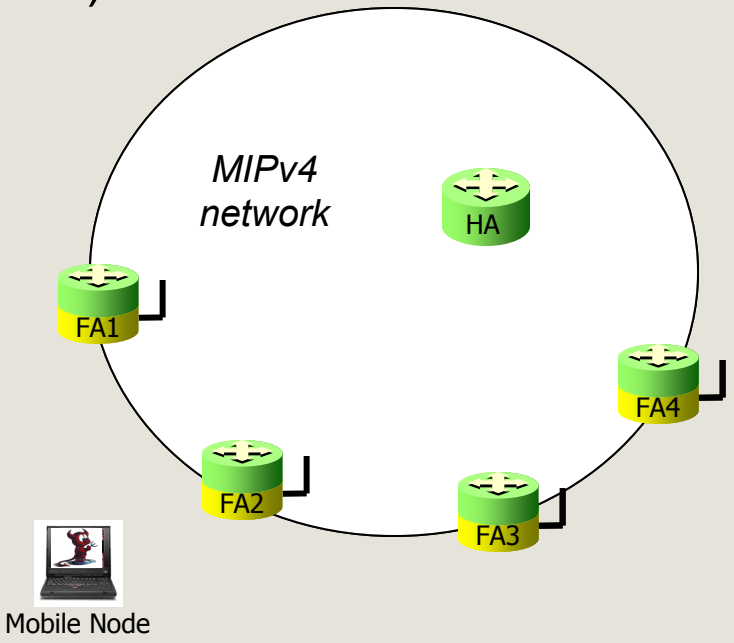
Sophia-Antipolis, France

Content

- Background information
- Implementation
- Example measurement results
- Conclusion

Background - Beyond3G project

- common platform and testbed for different **IP mobility** solutions
  - BCMP (Brain Candidate Mobility Protocol)
  - MIPv4
  - HMIPv6
  - etc.



Background - Test and measurement issues

- conformance and function testing
  - classic TTCN-3 test case execution
- system testing
  - stress (high load)
  - stability (memory leaks, buffer overrun, not handled exceptions, state machine “holes”)
- growing importance of non-functional requirements
  - characteristic measurements (performance evaluation)
    - delays, packet losses, processor load, memory consumption under realistic load and overloaded situations
  - checking correctness at the same time

➔ ***Needs an application which emulates a big amount of mobile nodes***

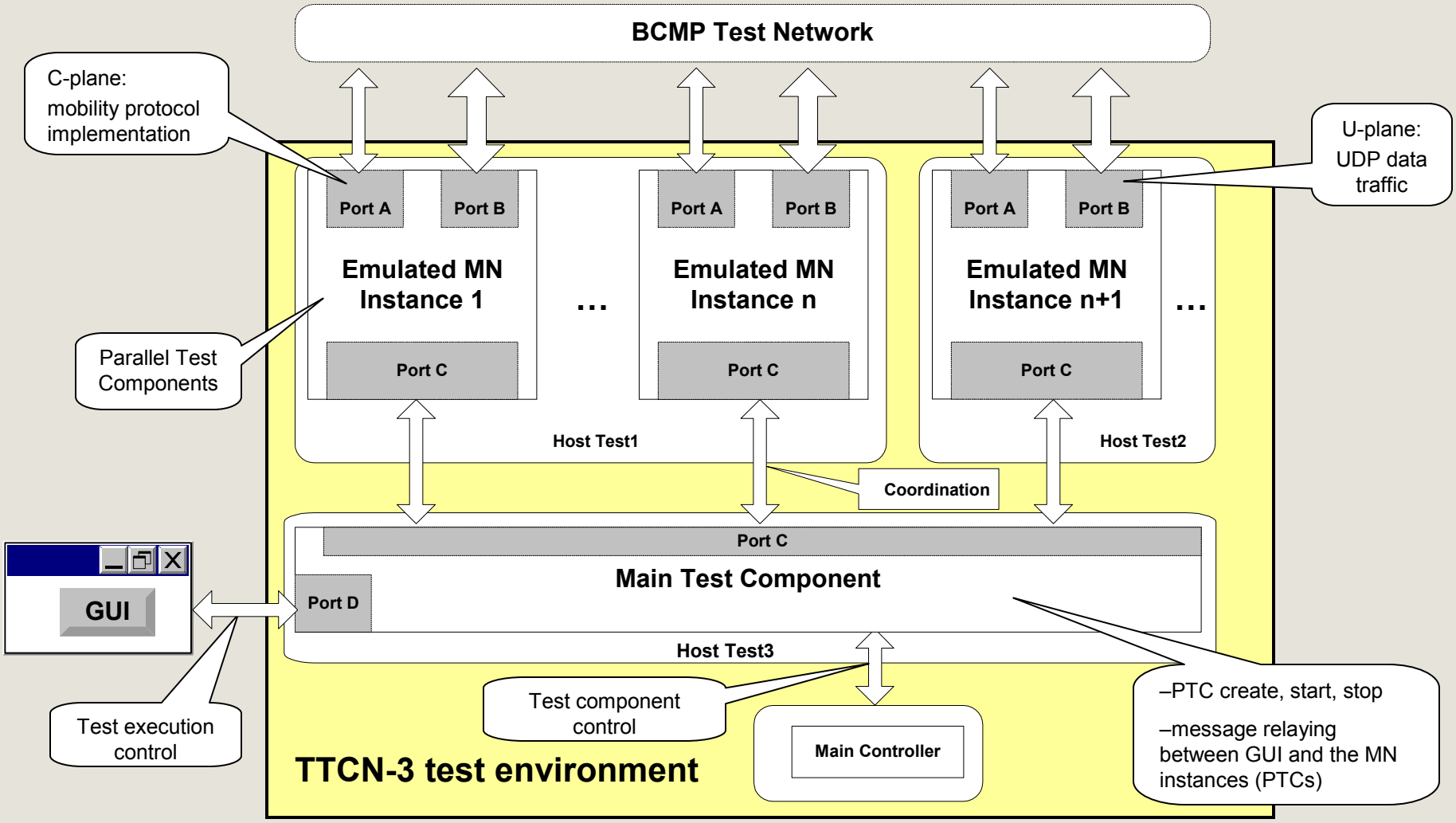
Implementation - Challenges

- independent emulated MN behavior
  - every MN should implement its own state machine with configurable timers
- generation of high load
  - emulating several hundreds of MNs
  - send/receive several thousands packets/s
- prototype testing
  - regularly and sometimes heavily redesigned implementations are to be tested
  - common test scenarios and measurements for different IP mobility protocols
- load test with measurement support

Implementation - Benefits using TTCN-3 for test and measurement application

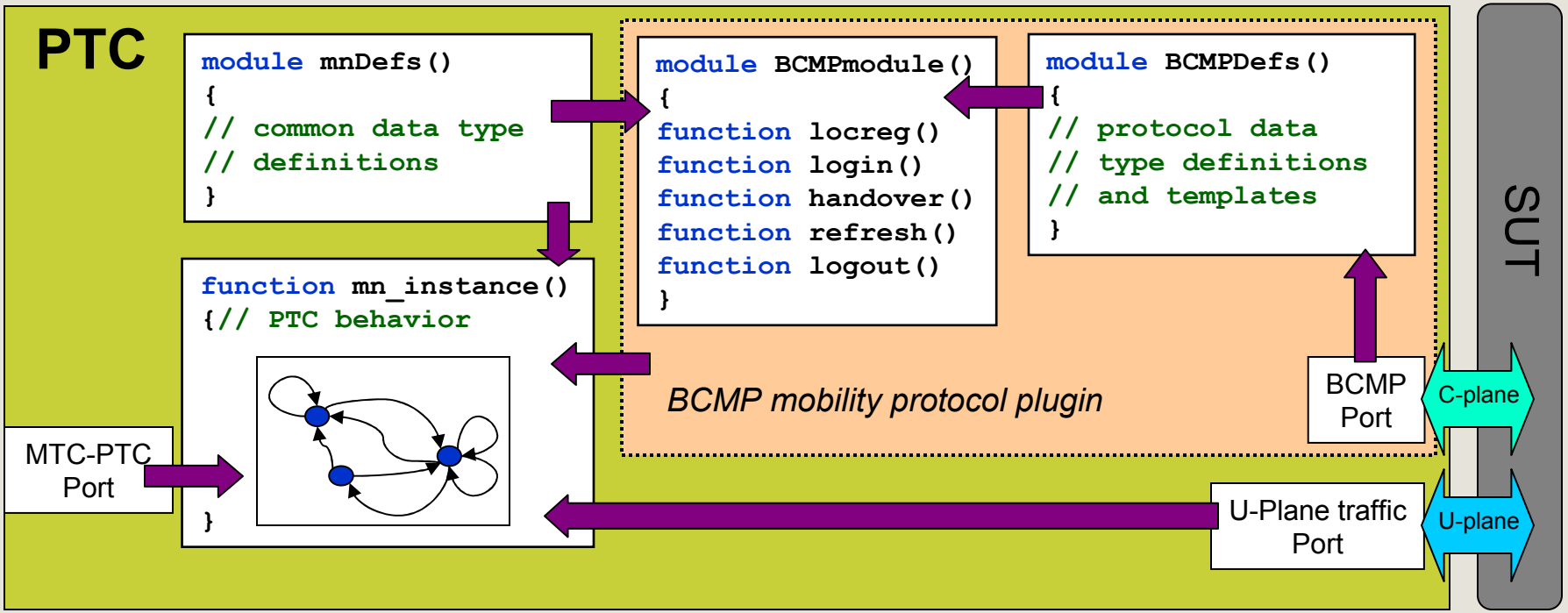
- re-use of existing conformance test environment
  - communication ports, templates, implemented message sequences
- effective test / application development
- well-defined system interfaces and communication layers
  - modularity, extensibility
  - allows repeated use the same test scenario with different protocols
- easy-to-scale
- built-in load balancing

Implementation - MN Emulator components

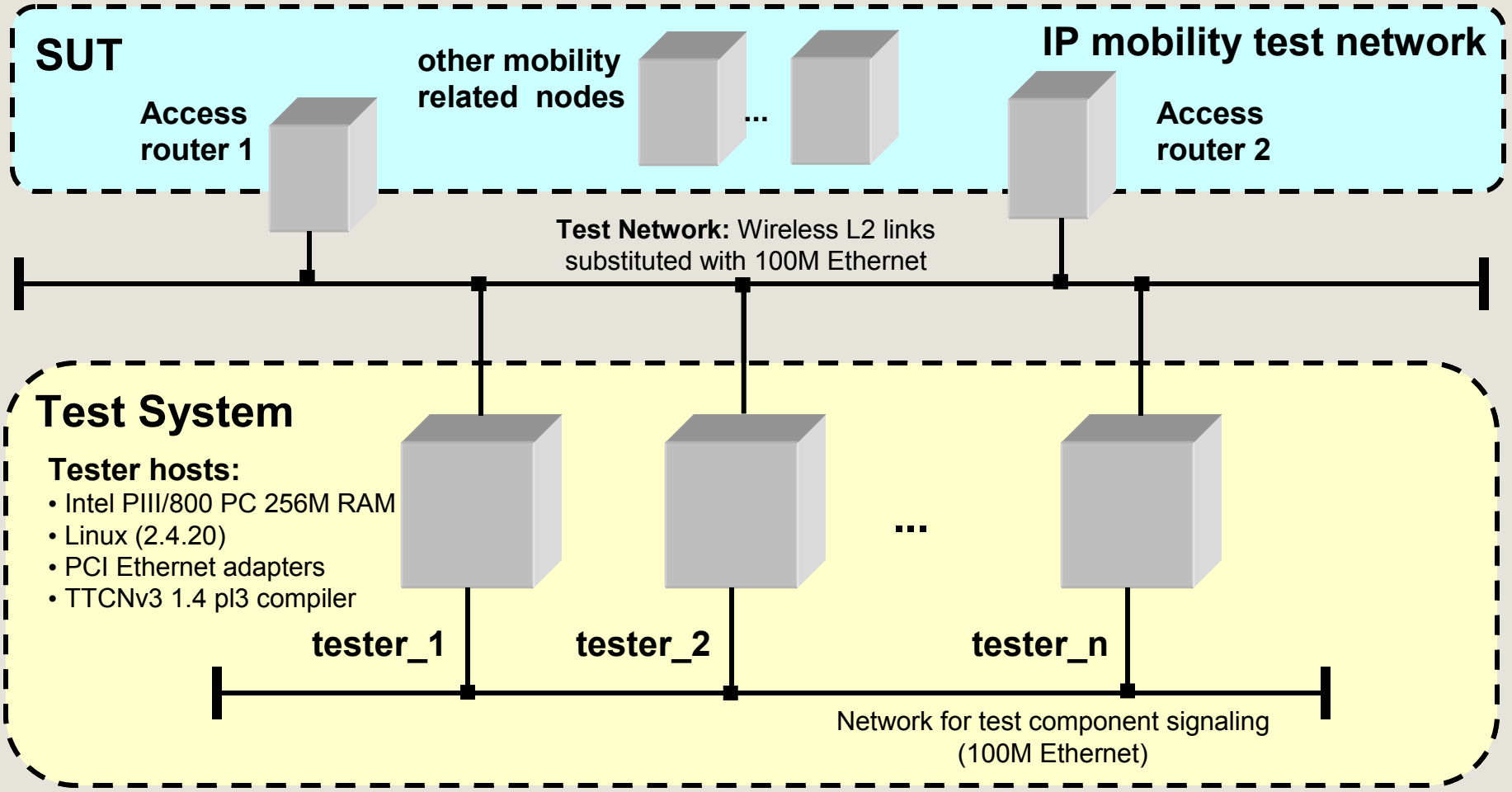


Implementation - Module architecture

- similar functions in all protocol:
  - common MN state machine
  - protocol module interface (“API”): multiprotocol support



Implementation - Test configuration



Implementation - Special design considerations

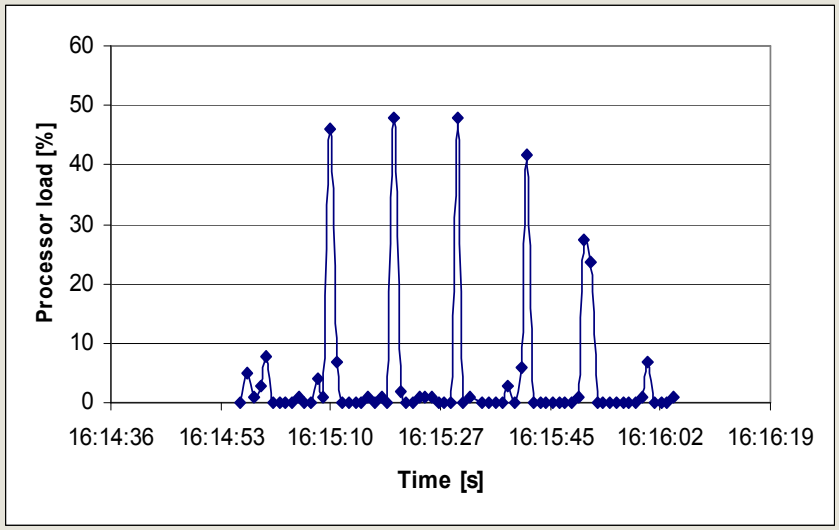
- Focusing on the performance of the test system
  - scalability
  - good performance of the test software
    - communication port software performance can be critical
      - simplified and optimized version: conformance of the implementation is assumed
    - keep data definitions as simple as possible
    - special timer handling philosophy
      - artificial delays and deviation (do not get synchronized)
    - carefully designed state machine
    - main event loop (`alt` statement)
      - minimized number of receive operations
    - log level

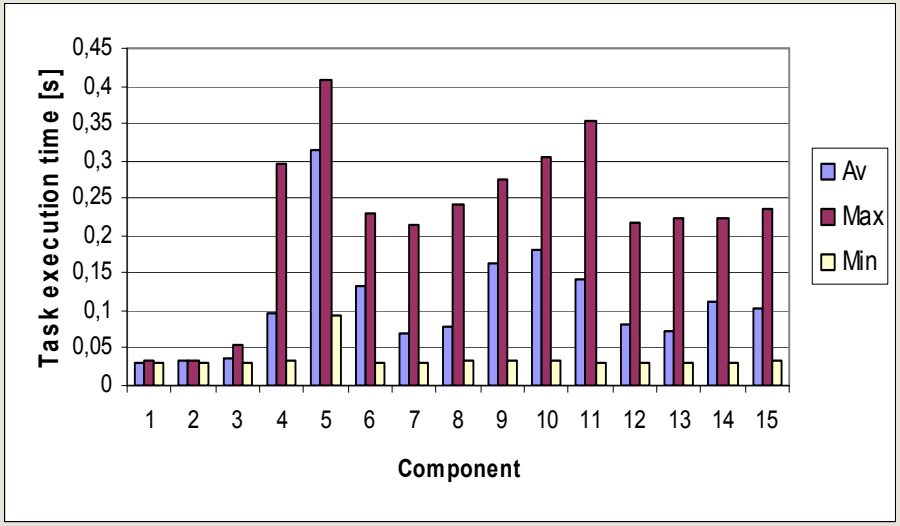
Implementation - Measuring the Test System performance

- Do not overload the test equipment : incorrect behavior or bad measurement results
  - Insufficient CPU: scale up (add more host) when more CPU power needed
  - memory consumption:
    - estimate the memory requirements of a single PTC
    - avoiding memory swapping: add a necessary amount of physical memory
- How to estimate the performance limit of the test system?
  - running the performance test suites
  - measuring the test system load
  - determining timing accuracy

Implementation – Avoiding overload with special timer handling #1

- **Fully deterministic timing, no random deviation**

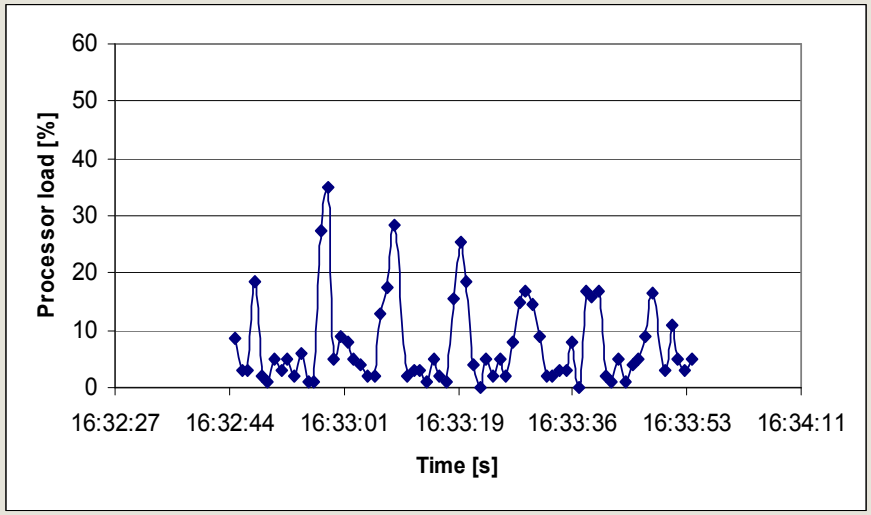


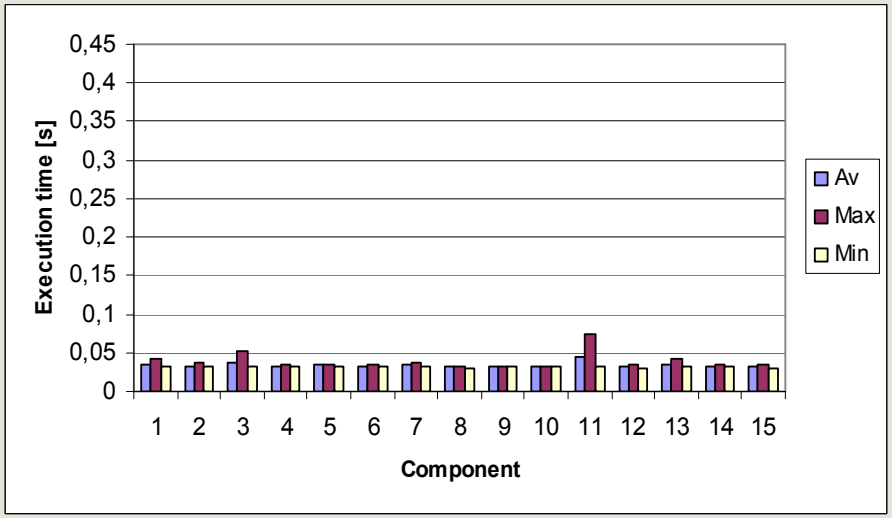


- Many events to be executed at the same point of time
- High peaks in the test system processor load
- Highly varying task execution times, although the CPU is not overloaded

Implementation – Avoiding overload with special timer handling #2

- **10% random timer deviation**



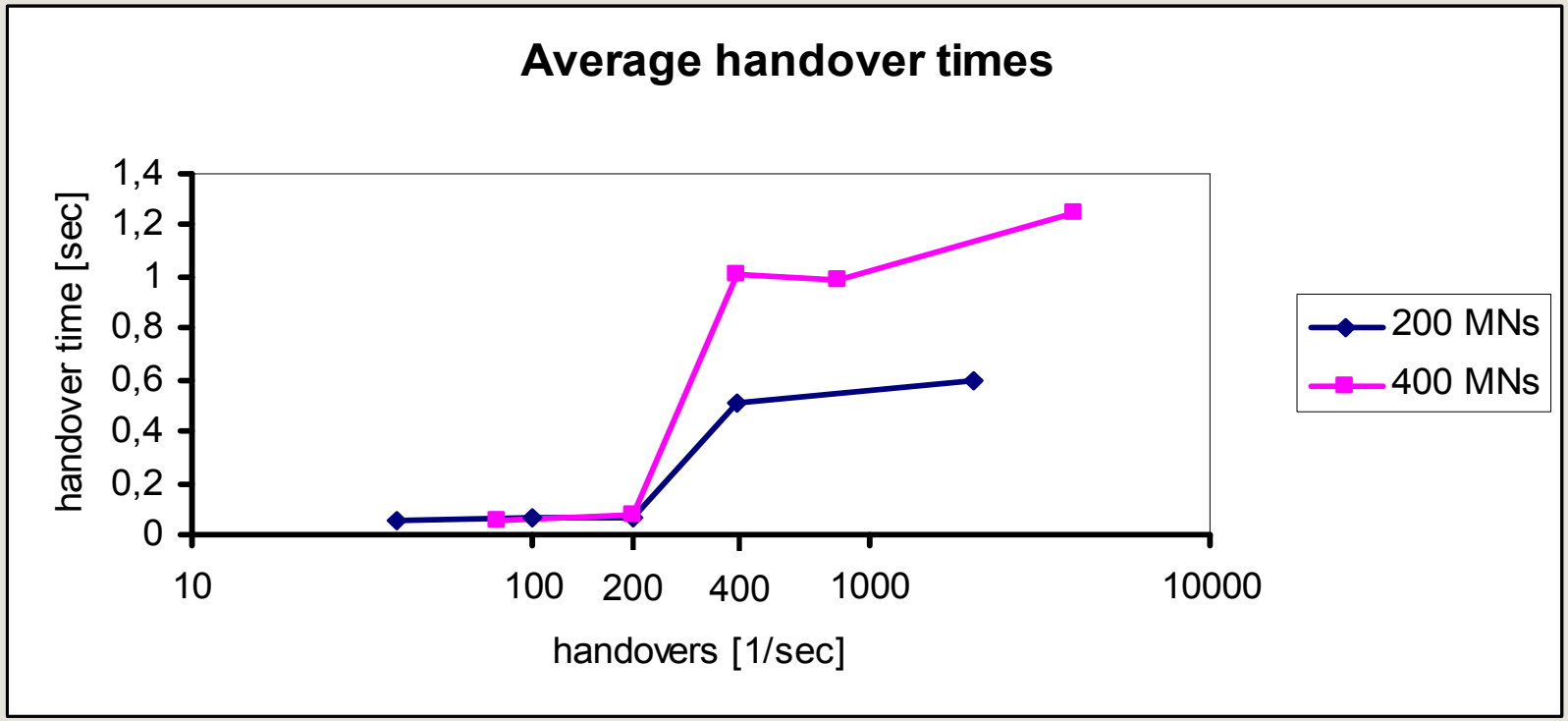


- Smoother test system processor load
- Faster execution, nearly equal execution times

Implementation - Mobile Node Emulator load generation features

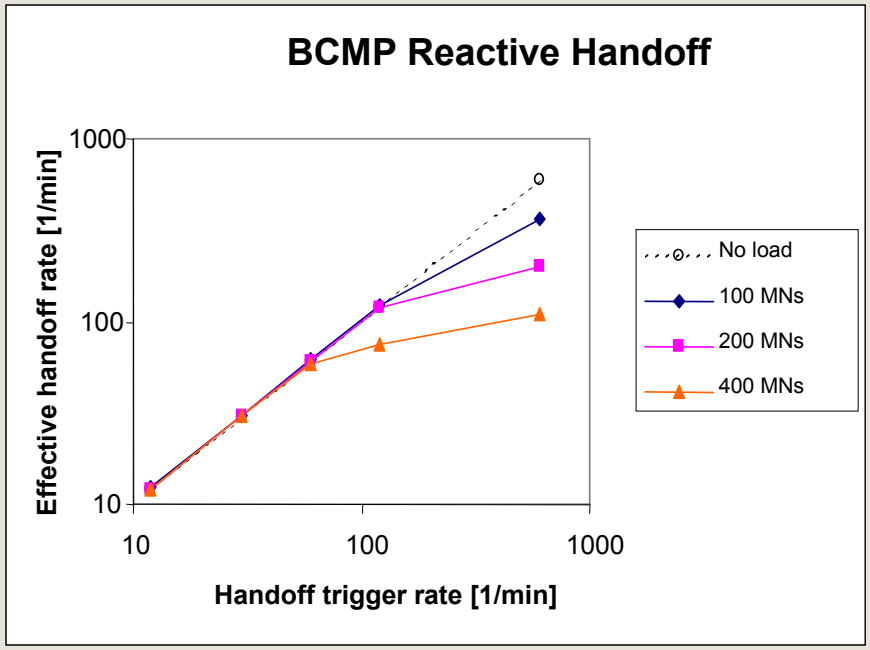
- arbitrary number of emulated mobile nodes up to the actual OS and hw-platform (CPU, memory) limitation
  - >1000 MNs, running on several Linux boxes, ca. 3-500 on each
- high execution performance
  - generate C-plane / U-plane traffic, up to ~4000pps / host
  - message sending takes 0.1…0.2 ms (highly depends on machine performance and test port code efficiency)

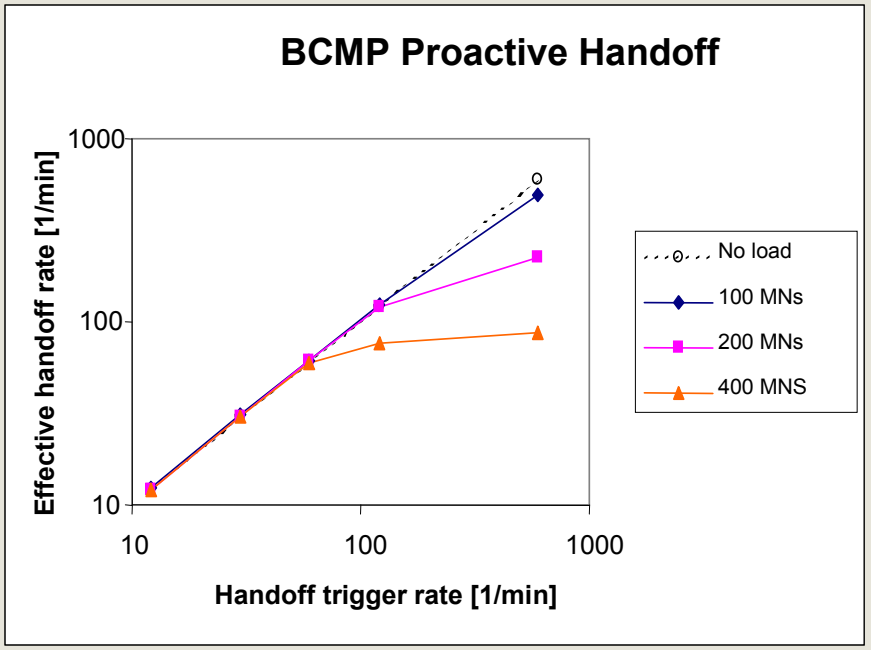
Example measurement results - BCMP access router average handover time



Example measurement results - BCMP access router effective handoff rate

Reactive handoff: IP layer notified after radio handover
Proactive handoff: IP layer notified before the radio handover (needed for real-time applications)





Dotted line: hypothetical reference (all triggered handoffs are executed)

Conclusion - TTCN-3 is ready for load and performance testing

- distributed, easy-to-scale architecture
- heterogeneous environment, low-cost hw elements
- well suits for multiprotocol and higher layer traffic generation scenarios
- extra benefits for prototype testing
  - simulation
  - load generation
  - measurement
- performance considerations are needed for the test system

Thank You for Your attention!