Brandenburg
University of Technology
Cottbus - Senftenberg

RNKS

2nd OMNeT++ Community Summit 2015

µIP Support for the Network Simulation Cradle

Michael Kirsche and Roman Kremmer
Computer Networks Communication Systems Group
Brandenburg University of Technology, Germany

Excerpt of Available TCP/IP Stacks

micro IP (µIP)	vs.	lightweight IP (lwIP)	vs.	FreeBSD IP-Stack
o 8-bit and 16-bit µC		o Embedded hardware		o 32-/64-bit systems
o ~4KB RAM / ~10KB ROM		o ~20KB RAM / ~40KB ROM		o KB → MBs RAM / ROM
o Compliant with TCP, UDP and IP RFCs		+ Full-scale stack with DNS, PPP, ARP, DHCP, …		+ Full-scale stack with DNS, PPP, ARP, DHCP, …
+ Standalone version as well as Contiki integration		+ Standalone as well as OS support (multiple systems)		o Embedded in FreeBSD
+ IPv6-ready (µIPv6)		– Experimental IPv6 support		+ Full IPv6 support
– Uses only 1 packet buffer, → throughput problems → AppLayer retransmission		+ High performance in almost all use cases		+ Highest performance in all use cases
– Standalone version does not support socket API		+ Socket as well as raw / native API for performance		– Requires Linux OS, no standalone support

Excerpt of Available TCP/IP Stacks

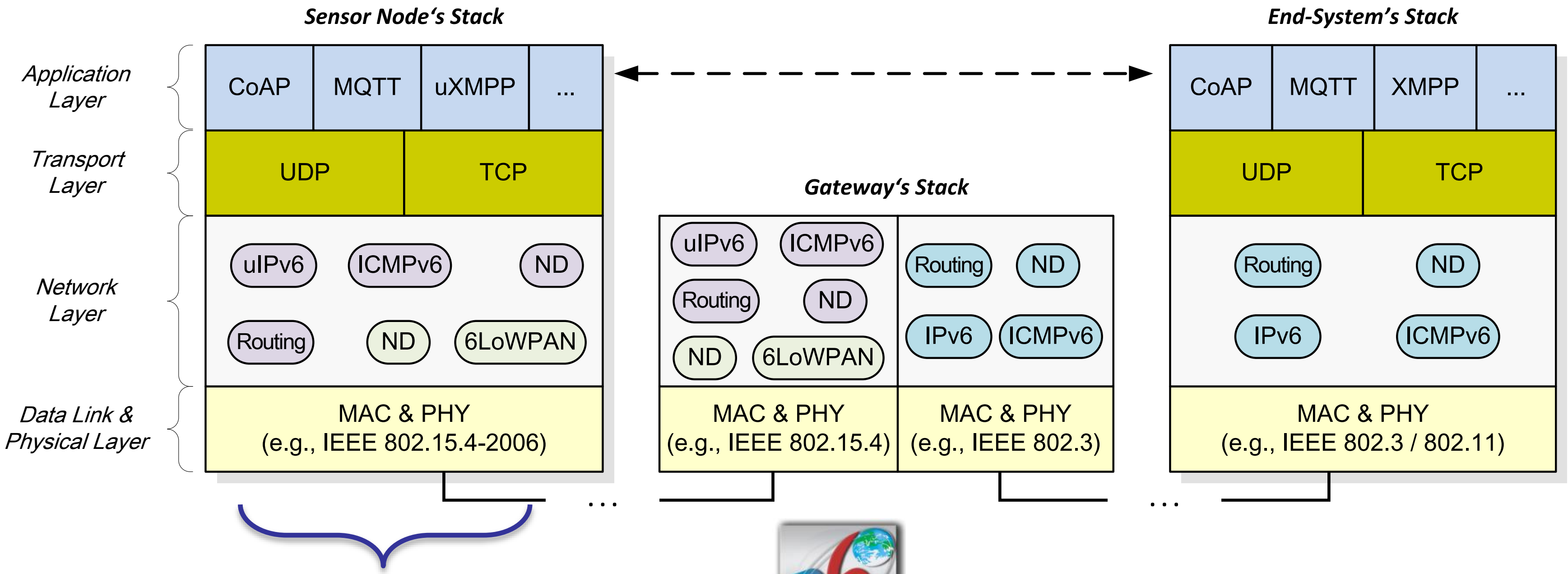
micro IP (µIP)	vs.	lightweight IP (lwIP)	vs.	FreeBSD IP-Stack
o 8-bit and 16-bit µC		o Embedded hardware		o 32-/64-bit systems
o ~4KB RAM / ~10KB ROM		o ~20KB RAM / ~40KB ROM		o KB → MBs RAM / ROM
o Compliant with TCP, UDP and IP RFCs		+ Full-scale stack with DNS, PPP, ARP, DHCP,		+ Full-scale stack with DNS, PPP, ARP, DHCP, …
+ Standalone version as well as Contiki integration		+ Standalone as well as OS support (multiple systems)		o Embedded in FreeBSD
+ IPv6-ready (uIPv6)		– Experimental IPv6 support		+ Full IPv6 support
– Uses only 1 packet buffer, → throughput problems → AppLayer retransmission		+ High performance in almost all use cases		+ Highest performance in all use cases
– Standalone version does not support socket API		+ Socket as well as raw / native API for performance		– Requires Linux OS, no standalone support

Support wanted

Supported by OMNeT++

Supported by OMNeT++

Why microIP ?



Part of a cyber physical system

Why simulate µIP in OMNeT++ ?

- microIP is usually tested via:
  1. Live experimentation on real systems
     - Deployments hard to control, low repeatability, costs, …
  2. Testbeds
     - Low scalability, set-up inflexible, limited control of external factors, …
  3. Cooja (Contiki OS simulator)
     - Cycle accurate emulation and possible interconnection to real systems
     - Limited simulation and comparison of/with systems/models outside the Contik world

Tackle these issues (+ more) through generic OMNeT++ simulation

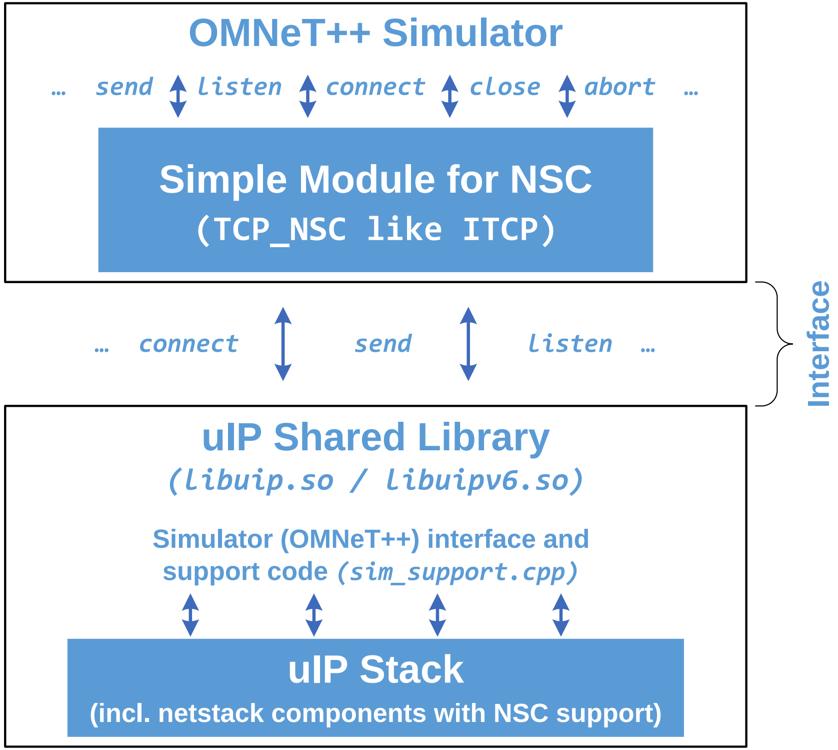
What is the Network Simulation Cradle ?

- Developed by Sam Jansen
- Basic idea: Integrate kernel-space implementations of real world network stacks into ns-2 / OMNeT++ *(instead of failure prone / abstract modeling)*
- Basic approach:
  - Parse the C-code,
  - Substitute global variables through arrays of per-node-instance variables,
  - Recompile as a shared library,
  - Map interfaces to ns-2 / OMNeT++ through glue code.
- Works without manual code changes in contrast to “plain” porting of stacks
  - E.g.: Bless and Doll *“Integration of the FreeBSD TCP/IP-Stack into the Discrete Event Simulator OMNet++”*

How to integrate µIP into the NSC ?

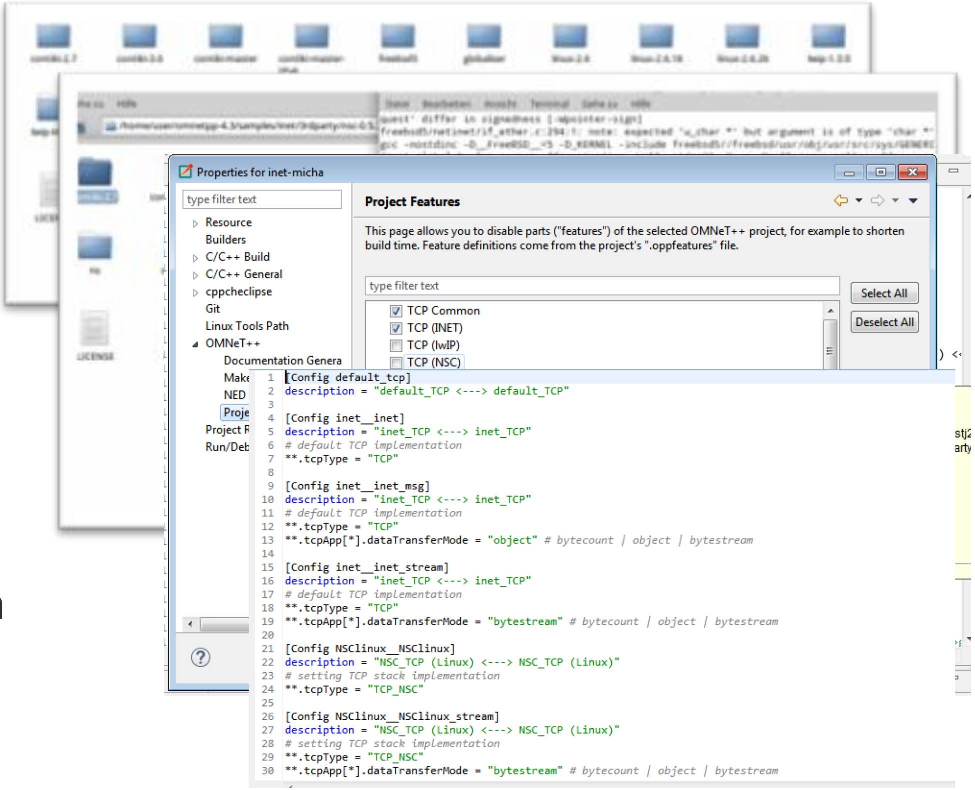
- Process differs a bit for different microIP versions (w/o API)

1. Integrate µIP source code into NSC build process
2. Implement stubs for references to unused system functions
3. Adjust globalizer parser for µIP
4. Create new netstack drivers for Contiki (to redirect calls to NSC)
5. Create config files and integrate into an OMNeT++ simulation



How to use µIP in OMNeT++ ?

- Prerequisite:
  - 32-bit Linux → NSC requirement
  - Source code from Github
- Steps:
  - Copy our code into extracted NSC
  - Compile shared µIP library (`libuip.so / libuipv6.so`)
  - Adjust `LD_LIBRARY_PATH`
  - Enable NSC / recompile simulation
  - Setup `omnetpp.ini`



In Conclusion

- µIP support in OMNeT++:
  - Provided via the NSC
  - Currently support for IPv4
  - Packet exchange between different stacks possible

➢ *Another stop along the road of IoT simulations with OMNeT++*

- Further actions:
  - Full IPv6 integration (NSC officially has IPv6 support, function calls yet always go to v4)
  - Combine with IEEE 802.15.4, 6LoWPAN and applayer protocol

➢ *Possible integration of other stacks in the future (e.g., RiotOS)*

Thank you for your attention! **Questions?**