INET Framework Development

András Varga

OMNeT++ Workshop
March 21, 2011
Barcelona, Spain

INET Development

A new version of INET is cooking!

“integration” branch on github

1. Change in version numbering
2. What’s been implemented/changed so far
3. Modularization of the codebase
4. Integration of forks and extensions
5. Documenting INET
6. Validation / Regression Testing

Version Numbering

- Traditional: “date” releases, e.g. INET-20100723
- Drawbacks of “date” releases:
  - cannot express code state (unstable/stable)
  - cannot express level of compatibility with previous releases
- New versioning scheme: <major>.<minor>.<patch>
  - even/odd numbering (even=stable, odd=development)
  - current releases are 1.99.x (unstable)
  - working towards 2.0.0 (stable)

INET Development Branch /1

TCP model:

- **TCP_lwIP**: a new TCP module that directly wraps the lwIP stack
  - Note: we also have NSC (Network Simulation Cradle) support in INET
- **transfer mode:**
  - former sendQueueClass / receiveQueueClass parameters
  - now controlled by the app (via a control info field); works across all three TCP models (native OMNeT++, lwIP, NSC)
  - TCP_TRANSFER_BYTECOUNT, TCP_TRANSFER_OBJECT, TCP_TRANSFER_BYTESTREAM (new)
- **application-imposed TCP flow control,** i.e. implementation of "socket read" calls
  - experimental code, not yet released or in the "integration" branch

INET Development Branch /2

New models, model improvements:

- **signal-based statistics recording**: in most modules, cOutVector and recordScalar() was replaced with emitting signals and @statistic properties in NED files
- **BGPv4 model,** contributed by Helene Lageber
- **point-to-point Ethernet links** can now use normal DatarateChannels (EtherMAC's txrate parameter was removed)
- **“global ARP”,** backported from INETMANET
- **multi-radio support,** backported from INETMANET
- **RTP model**: refactored to make use msg files and a controlinfo-based interface toward apps
- **OSPF model**: mass renaming and code formatting
- some improvements implemented in INETMANET taken over into INET (also to decrease INETMANET maintenance cost)

INET Development Branch /3

NED Refactoring:

1. support for more than one type of tcpApp and udpApp in StandardHost
   - StandardHost’s tcpAppType/udpAppType parameter was eliminated
   - tcpApp[] is now declared as:

     ```
     tcpApp[numTcpApps]: <> like TCPApp;
     ```

     and module types can be set using **type-name** in the ini file:

     ```
     **.tcpApp[0].type-name = "TCPBasicClientApp"
     **.tcpApp[1].type-name = "TCPEchoApp"
     **.tcpApp[*].type-name = "TCPSinkApp"
     ```

INET Development Branch /4

NED Refactoring:

2. Consistency: module interfaces were renamed to start with letter “I”:

- BasicMobility → IMobility
- TCPApp → ITCPApp
- UDPApp → IUDPApp
- SCTPApp → ISCTPApp
- MacRelayUnit → IMacRelayUnit
- Radio → IRadio
- Ieee80211Mgmt → IIeee80211Mgmt
- OutputQueue → IOutputQueue
- INetworkInterface → IWiredNic

INET Development Branch /5

NED Refactoring:

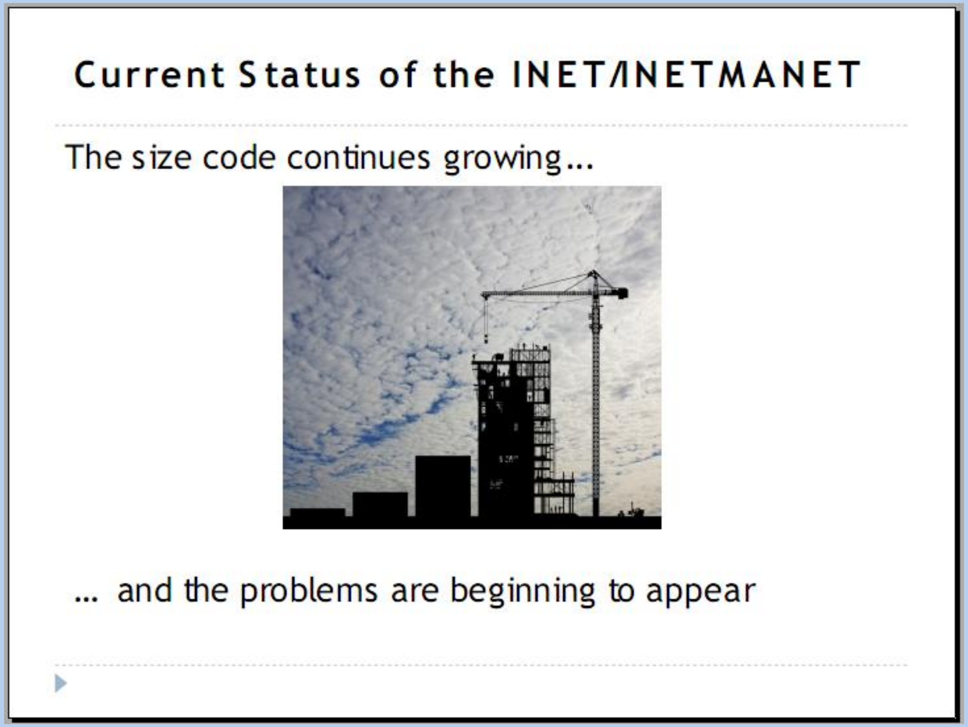
- Problem: we had countless variants on StandardHost,Router, etc:
  - StandardHost, StandardHostWithDLDuplicatesGenerator, StandardHostWithDLThruputMeter, StandardHostWithULDropsGenerator, StandardHostWithULThruputMeter, BustHost, TCPSpoofingHost, MobileHost, MFMobileHost, WirelessHost, WirelessHostSimplified,...
  - Router, ExtRouter, OSPFRouter,...
  - WirelessAP, WirelessAPSimplified, WirelessAPWithEth, WirelessAPWithEthSimplified, ...
- Attempt to unify them:
  - added hook modules (IHook) into NetworkLayer where DropsGenerator, DuplicatesGenerator and other can be substituted
  - elements of StandardHost made replaceable via “like” and module interfaces
    - for example, “tcp” in StandardHost is now declared as:

      ```
      tcp: <tcpType> like ITCP
      ```

      and tcpType can be “TCP”, “TCP_lwIP”, “TCP_NSC”
  - most existing modules can be replaced with the revised StandardHost, Router, AccessPoint modules

Modularization

- Motivation:
  - long build times in INET/INETMANET
    - also: linker command-line limit reached on Windows
  - unclear structure, unwanted cross-dependencies creeping in
- Problems are described in:
  - *“Towards a modularized INET”*. Alfonso Ariza (Univ. de Malaga), Juan-Carlos Maureira (INRIA), International OMNeT++ Workshop. Malaga, Spain, 2010.



Current Status of the INET/INETMANET

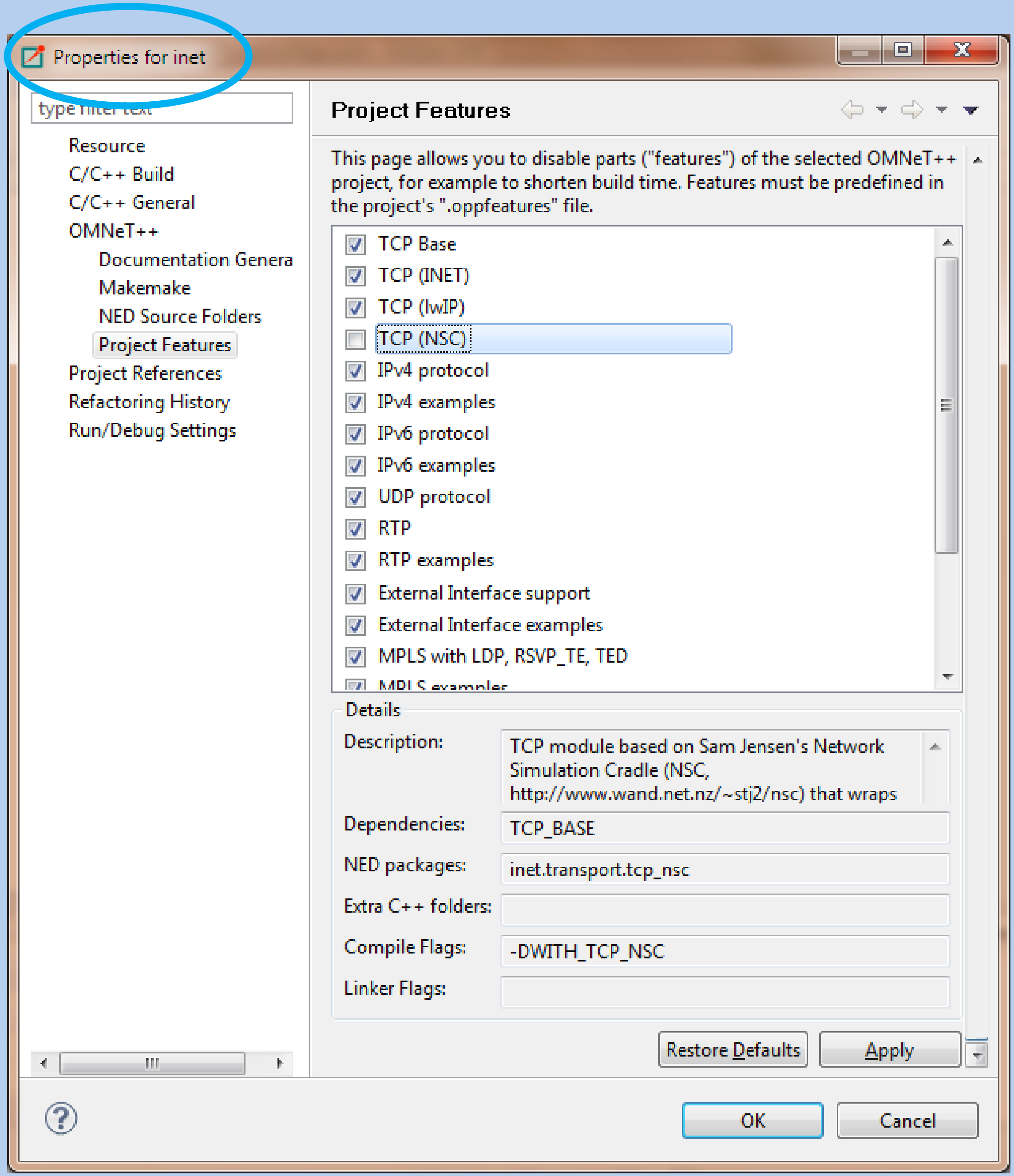
The size code continues growing…

… and the problems are beginning to appear

Modularization: “Project Features”

- OMNeT++ built-in support for project modularization
  - currently experimental, to appear in 4.2 beta 2
- A “project feature”:
  - source code (modules, etc) that can be enabled and disabled for a build
    - granularity: folder
    - includes NED, C++ and msg files
    - may add defines (e.g. -DWITH_IPv6) and libraries (e.g. -lavcodec) to the build
  - may require other features (dependency handling)
  - disabled feature:
    - its NED files are excluded (inaccessible from the rest of the project)
    - folder is (are) excluded from C++ build
    - its header files cannot be (easily) #included from the rest of the project

Project Features in the IDE



Properties for inet

type filter text

- Resource
- C/C++ Build
- C/C++ General
- OMNeT++
  - Documentation Genera
  - Makemake
  - NED Source Folders
  - Project Features
- Project References
- Refactoring History
- Run/Debug Settings

Project Features

This page allows you to disable parts ("features") of the selected OMNeT++ project, for example to shorten build time. Features must be predefined in the project's ".oppfeatures" file.

- ☑ TCP Base
- ☑ TCP (INET)
- ☑ TCP (lwIP)
- □ TCP (NSC)
- ☑ IPv4 protocol
- ☑ IPv4 examples
- ☑ IPv6 protocol
- ☑ IPv6 examples
- ☑ UDP protocol
- ☑ RTP
- ☑ RTP examples
- ☑ External Interface support
- ☑ External Interface examples
- ☑ MPLS with LDP, RSVP_TE, TED
- ☑ MPLS examples

Details

Description:	TCP module based on Sam Jensen's Network Simulation Cradle (NSC, http://www.wand.net.nz/~stj2/nsc) that wraps
Dependencies:	TCP_BASE
NED packages:	inet.transport.tcp_nsc
Extra C++ folders:	
Compile Flags:	-DWITH_TCP_NSC
Linker Flags:	

- Available in the Project Properties dialog
- Shows feature descriptions, lets the use enable/disable features
- Dependency handling
- Modifies CDT configuration and NED excluded package list

Modularization: .oppfeatures

Features are described in the *.oppfeatures* file in the project root

```
<features>
    ...
    <feature
        id="IPv6"
        name="IPv6 protocol"
        description = "Basic IPv6 support"
        requires = ""
        labels = "protocols"
        nedPackages = "inet.networklayer.ipv6
                       inet.networklayer.autorouting.ipv6
                       inet.networklayer.icmpv6
                       inet.nodes.ipv6"
        extraSourceFolders = ""
        compileFlags = "-DWITH_IPv6"
        linkerFlags = ""
        />
    ...
</features>
```

Project Features

- Additional advantages
  - allows integration of “problematic” models
    - doesn’t compile on all platforms
    - requires exotic libraries
    - rarely used

      → *can be turned off when not used*
  - can be the first step when you want to factor out parts of a project into a separate project

Integration of Forks and Extensions

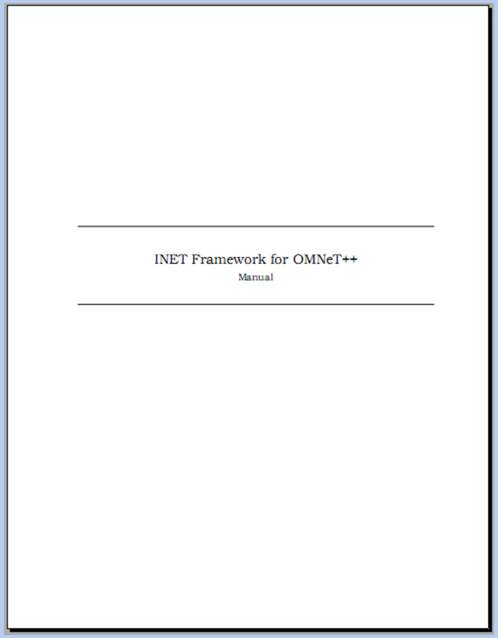
Candidates for integration into INET as “features”:

- xMIPv6
- VoIPTool ?
- HttpTools ?
- routing protocols in INETMANET ?
- MiXiM ?
- INET-HNRL ?
- ...
- ...

Help in reviewing and/or integrating these extensions will be welcome!

Documentation

- INET Manual
  - concepts, architectural overview
  - high-level protocol descriptions
  - useful for newcomers
  - mostly TBD
- Neddoc
  - useful as reference



INET Framework for OMNeT++

Manual

INET Manual

- Repo: github.com/inet-framework/inet-doc (LaTeX)
- Working Table of Contents:

  1. Introduction
  2. Using the INET Framework (incomplete)
  3. Node Architecture (incomplete)
  4. Point-to-Point Links (TBD)
  5. The Ethernet Model (incomplete)
  6. The Radio/Wireless Infrastructure (TBD)
  7. The 802.11 Model (TBD)
  8. Node Mobility (TBD)
  9. IPv4 (incomplete)
  10. IPv6 and Mobile IPv6 (TBD)
  11. The UDP Model (TBD)
  12. The TCP Model (TBD)
  13. The SCTP Model (TBD)
  14. Internet Routing (TBD)
  15. The MPLS models (TBD)
  16. Applications (TBD)
  17. History

  TODO:

  - ad-hoc routing protocols (INETMANET)
  - ...

- Volunteers welcome!

Validation / Regression Testing

- Needed for credibility
- Fingerprints are too fragile
- Instead: simulation + result evaluation script
  - Example criteria:
    - “TCP overall throughput should be between 50kbs and 52kbps”
    - “Hosts get a fair share of the throughput, e.g. each within 10% of the average”
  - Evaluation script: use GNU R (r-project.org) with the “omnetpp” R package (github.com/omnetpp/omnetpp-resultfiles)
    - R: *“a free software environment for statistical computing and graphics”:* linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc.
    - the “omnetpp” package provides loadDataset() and other functions
    - loadDataset(): loads vector and scalar files in whole or filtered
    - loaded data can be processed and evaluated using R’s capabilities

INET Roadmap

- Release OMNeT++ 4.2
  - includes the “project features” feature
- INET
  - 1.99.0 (released)
  - 1.99.1, 1.99.2, ... -- development / testing versions
  - 2.0.0 – stable
    - still without (many) extensions integrated
    - extension-friendly
  - 2.1.x – unstable
  - 2.2.0 -- integrates many extensions (to be decided)