

OMNeT++ Community Summit 2016, Brno University of Technology (FIT-BUT), Sept 15-16.

Plans for OMNeT++ 5.1

András Varga

Once upon a time...

OMNeT++ 5.0 was released in April.

With new APIs and components, and lots of breaking changes: 2D and 3D graphics support (Canvas, OpenSceneGraph), new graphical runtime (Qtenv), new logging API, and so on.

However...

There were a lot of things we had to cut from 5.0 so as not to delay shipping indefinitely: upgrading to the latest Eclipse, upgrading the Windows toolchain, rethinking the build system, properly finishing Qtenv, polishing the Canvas API, brushing up SQLite code we wrote earlier, etc.

OMNeT++ 5.1

Changes are centered around the following topics:

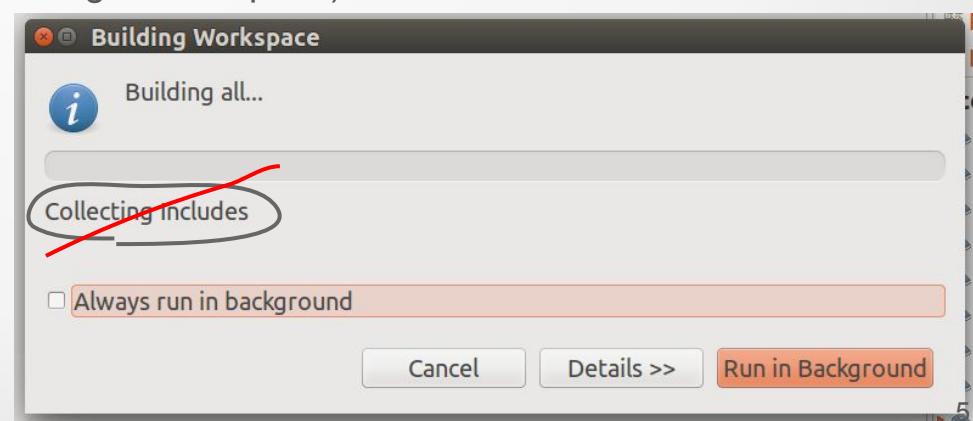
- Upgrading our dependencies (Eclipse, toolchain, etc.)
- Improving Project Features support and C++ build of large models
- Finishing Qtenv
- Canvas API refinements
- Improvements in the simulation kernel
- Better animation support (in progress, maybe 5.2 only)
- Better support for simulation campaigns (in progress, maybe 5.2 only)

Updating our dependencies

- New Eclipse version
 - Eclipse 4.6 Neon (Java 1.8 required)
 - CDT 9.0
- Support for 64-bit Windows using MinGW-w64
 - Dropped support for 32-bit (reason: 32-bit OSes are on the way out, and shipping both 32 and 64-bit libraries would blow up download size)
- Qt5 required
 - Qt5.0 was released in 2012, 4 years ago
- OSG 3.2–3.5, osgEarth 2.5–2.7

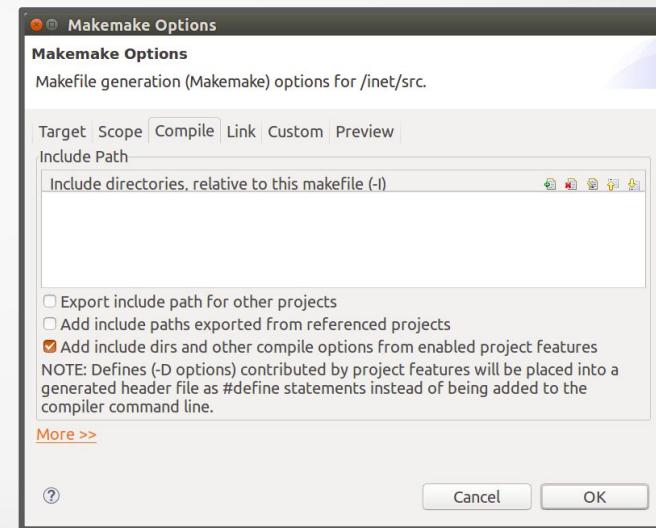
Build: Using compiler-generated dependencies

- What are compiler-generated dependencies?
 - *.d files under out/ (one for each .o file); makefile includes out/../*.d
 - Produced as a “side effect” of compilation, and reused for subsequent builds; make clean deletes them
 - Used both for OMNeT++ and models
 - Requires compiler support (e.g. gcc/clang -MMD option)
- Advantages:
 - “make depend” no longer needed
 - IDE: No more waiting for the “Collecting includes...” dialog



More changes in C++ build support

- Support for *deep includes* has been dropped
 - Deep includes: automatically adding each subfolder to the include path, so #include don't need to specify folders
 - Experience has shown it was not really useful:
 - Not needed for small projects
 - Too error-prone in large projects
- IDE: Refinements in the Makemake Options dialog and in makefile generation



Improving Project Features support

- What is Project Features again?
 - A way to break up a large project into smaller pieces that can be turned on/off separately (added to / removed from the build)
 - Accessible from both IDE and command line (opp_featuretool)
- Change: Symbols for enabled features (WITH_IPv4) are now placed into a generated header file, not passed to the compiler via -D options
 - Name of header file is part of .oppfeatures (feature definition file)
 - Advantages:
 - Indexer knows about them (and #ifdef blocks are shown with proper enablement state)
 - Easier access from derived projects

- TCP Common
- TCP (INET)
- TCP (lwIP)
- TCP (NSC)
- IPv4 protocol
- INET examples

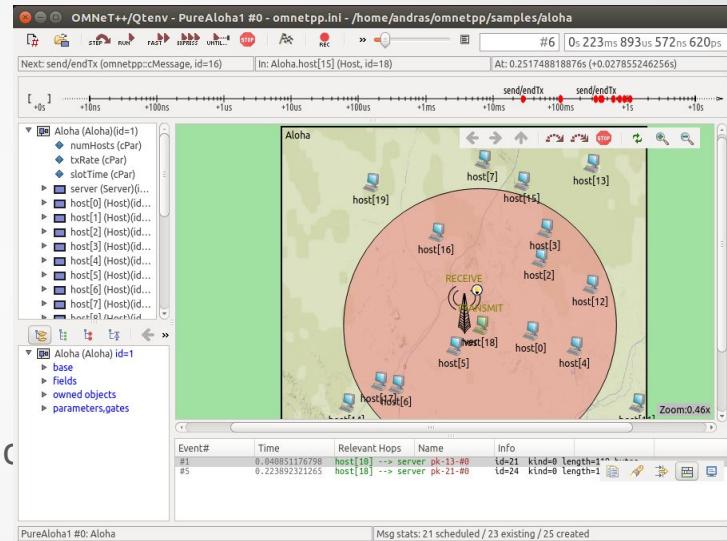
```
INETDefs.h features.h
// Generated file, do not
#ifndef WITH_AODV
#define WITH_AODV
#endif

#ifndef WITH_APSKRADIO
#define WITH_APSKRADIO
#endif

#ifndef WITH_BGPv4
#define WITH_BGPv4
#endif
```

Qtenv

- Qtenv has reached maturity
 - Tons of bug fixes and improvements
 - It is now the default GUI for simulations
- Tkenv
 - Can still be activated using *-u Tkenv*
 - Maintained, but not actively developed any more
(new features will be Qtenv-only)
 - Will be kept around until there is consensus that it can be removed
- UI improvements
 - Improved simulation time display (digit grouping and units)
 - Context menu adjustments
 - Other usability improvements



#88,090

2'710s 437ms 822us 631ns 201ps

Canvas improvements

- cFigure additions
 - Tooltip
 - Associated cObject
 - zIndex
 - Text halo
- Support for text extent and image size
 - Getting the bounding box for text and image (icon) items
 - Involves calling into Qtenv/Tkenv!
- Self-refreshing figures
 - cFigure::refreshDisplay(), only called if containing canvas is open in the GUI
 - Useful for certain figures, e.g. compound figures implementing plots, gauges, etc.
- Implemented in Qtenv
 - Tkenv only has partial or no support for new features
- Tests
 - better structured, more coverage

Core / Simulation Kernel

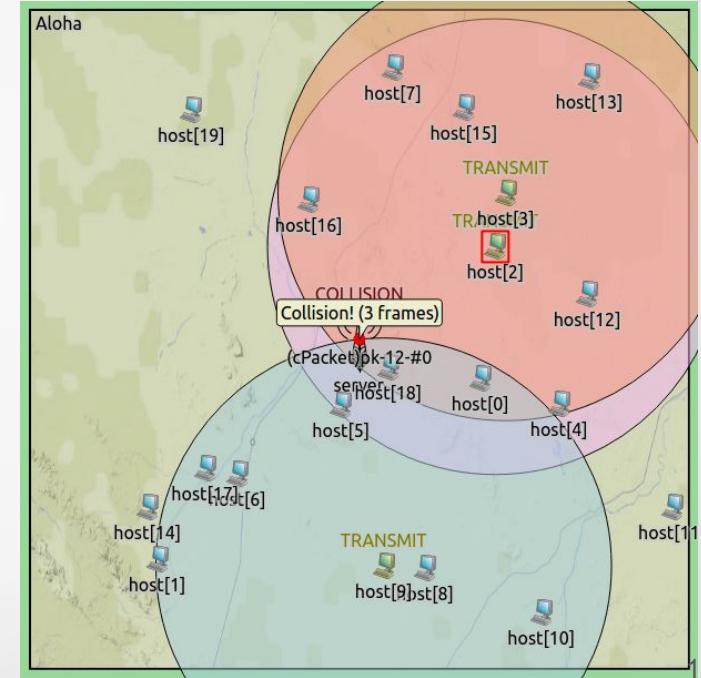
- Little API changes
 - info() renamed to str(), but old method still exists and delegates to the new one
 - detailedInfo() deprecated due to little raison d'être
- Enhancements:
 - @statistic: source can be a signal of a (direct or indirect) submodule
 - Consequence: @statistic parser moved from envir into the sim. Kernel
 - More items made inspectable in Qtenv/Tkenv:
 - Listeners lists per signal
 - Simulation results being collected (i.e. result recorders added by @statistic)
 - XML-valued module parameters, XML values (cXMLElement trees)



End of implemented features.
Plans start here.

Support for smooth custom animations

- Goal: add infrastructure for creating arbitrary animations
- Introduced in a separate presentation
- “Proof-of-concept” implementation exists (part of Tech Preview)



Better support for simulation campaigns

- Exploring large parameter space with simulation:
 - Many iteration variables, replications → generates a large number of runs
 - Being able to (re)run a subset of runs is important for incremental execution of parameter studies
- Improvements in the following areas:
 - Managing simulation runs
 - Result analysis
- *Inspired by feedback from Antonio Virdis*

Run filter

- The run filter (Cmdenv's -r option) allows selecting a subset of runs for execution
 - Previously, -r only accepted run numbers and run number ranges, like 1,5,8..12
- The -r option has been extended to accept a match expression
 - A plain wildcard expression is matched against the \${iterationvars} string
 - Match expression can also refer to iteration variables, or a boolean expression formed from them (AND, OR, parentheses)
 - Examples: -r '*mean=4.3*'; -r 'mean(4.3) AND numHosts({10..20})'
 - There are plans to extend/revisit the match expression syntax (also accept var~patt for var(patt), etc.)

Run filter, cont'd

- A query (-q) option has also been added
 - -c <configname> -r <runfilter> -q numruns
 - -c <configname> -r <runfilter> -q runs
 - -c <configname> -r <runfilter> -q rundetails

```
$ ./aloha -c PureAlohaExperiment -r "numHosts(10)" -q runs
OMNeT++ Discrete Event Simulation (C) 1992-2016 Andras Varga, OpenSim Ltd.
Version: 5.1tp, build: 160908-ea40fdc, edition: Academic Public License -- NO
See the license for distribution terms and warranty disclaimer
Setting up Qtenv...

Config: PureAlohaExperiment
Number of runs: 42
Number of runs selected: 14

Run 0: $repetition=0, $numHosts=10, $mean=1
Run 1: $repetition=0, $numHosts=10, $mean=2
Run 2: $repetition=0, $numHosts=10, $mean=3
Run 3: $repetition=0, $numHosts=10, $mean=4
Run 4: $repetition=0, $numHosts=10, $mean=5
```

Result file naming

- The traditional naming scheme contains the run number, which is not very convenient
 - <configName>-<runNumber>.{sca|vec}
 - Problems: difficult to identify runs; nightmare when incrementally adding new runs
- Solution: iteration variables as part of default file names
 - Example: SlottedAloha-numHosts=10,mean=0.9-#3.sca
 - Illegal and inconvenient characters encoded in an urlencode-like manner

Nesting order of iterations

- Repetition used to be the innermost loop
 - Good? Bad?
 - It performs all replications for a data point before going on to the next
- Changed to be the outermost loop
 - Allows one to get early results for all data points, then refines the picture by executing more runs
 - Analogy: loading a JPEG image: line-by-line vs progressive
- Potential further improvement: specify nesting order explicitly
 - Concept: `itervars-nesting-order = repetition, *, numHosts`

Revisiting result analysis

- Improving CSV (and other) export
 - Added run attributes (iteration variables, etc) as columns
 - In scavetool as well as the IDE
- SQLite as result file format?
 - To be explored further
 - Co-exist with traditional file format (line-oriented text file)
- Long-term plans to improve the Analysis Tool
 - Usability needs to improve
 - Should assist advanced users transition to programmatic result analysis and plotting, e.g. using Python or R

Workshop Release



OMNeT++ 5.1 Technology Preview

Contains:

- Snapshot of 5.1 development
- Experimental support for smooth custom animation
- Experimental support for SQLite result files

Thank you