INET Framework Evolution

András Varga

OMNeT++ Workshop
March 23, 2012
Desenzano, Italy

Topics from the last Workshop

Current Status of the INET/NETMANE

The size code continues growing...

... and the problems are beginning to appear

INET Development

A new version of INET is cooking!

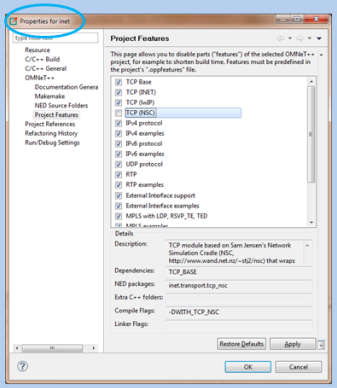
"integration" branch on github

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

INET Roadmap

- Release OMNeT++ 4.2
  - includes the "project features" fe
- INET
  - 1.99.0 (released)
  - 1.99.1, 1.99.2, ... -- developmen
  - 2.0.0 – stable
    - still without (many) extensions integ
    - extension-friendly
  - 2.1.x – unstable
  - 2.2.0 -- integrates many extensio

Project Features in the IDE



- Available in the Project Properties dialog
- Shows feature lets the use er features
- Dependency h
- Modifies CDT and NED excl list

Documentation

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

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

What do we want INET to be?

- What should be INET's role and scope?
  - **SOLID FOUNDATION** for network research
    - well-tested set of standard protocols (IPv4/v6, TCP, UDP, Ethernet, 802.11, ...)
    - infrastructure (radio, mobility, configuration, failure/recovery, statistics, cross-layer communication,...)
    - serve as a base for active projects: INETMANET, OverSim, Veins, ...
    - absorp and integrate useful model code from finished or inactive projects, and maintain them as part of INET (e.g. VoIPTool, HttpTools)

What We Need for INET

1. Protocols need to be reviewed
   - for correctness and completeness
2. Infrastructural features
   - e.g. modularity, failure/recovery, flexible network configuration, battery, obstacles, etc.
3. Documentation
4. Testing / validation
   - to build confidence in the models
5. Animation/visualization capability
6. More, and more organized, community participation

What Happened Since Last Time?

1.99.0 (Mar 3, 2011)

revised NED for hosts, routers, NICs; signal-based statistics recording; multi-radio; TCP improvements (e.g. added lwIP); added BGPv4

1.99.1 (May 27, 2011)

introduced Project Features; revised host, router, AP, NIC compound modules; added VoIPTool, xMIPv6

1.99.2 (Nov 18, 2011)

revised UDP; revised mobility models (sync with MiXiM); revised apps parametrization; added HttpTools

1.99.3 (Feb 22, 2012)

revised IPv4 (fragmentation, multicast, etc.) and Ethernet; UDP improved; test framework, ChangeLogs

1.99.4 (March 20, 2012)

revised IPv4 multicast routing; added IGMPv2, network configurator for IPv4 (replaces .irt files and FlatNetwork-Configurator)

.

.

.

2.0

For 2.0:

- ***revise IPv6; add node failure/recovery, DiffServ infrastructure, battery, IPv6 network configurator; integrate DHCP, STP; merger with MiXiM (!)***
- ***maybe: revise xMIPv6, OSPF, BGP, MPLS/RSVP/LDP; integrate more contributed components***

PROTOCOL REVIEWS

Protocol Reviews

Reviewed/extended:

- **TCP** (review; added SACK; TCP_lwIP, TCP_NSC)
- **UDP** (bugfixes, refactoring; socket options support; multicast revised)
- **IPv4** (bugfixes, refactoring; multicast revised; added IGMP)
- **Ethernet** (extensive refactoring, bugfixes; 40G/100G Ethernet implemented)

Planned reviews:

- **IPv6**
- **IEEE 802.11**
- **OSPFv2**, **BGPv4**
- **MPLS** and related protocols

INFRASTRUCTURE

Infrastructure

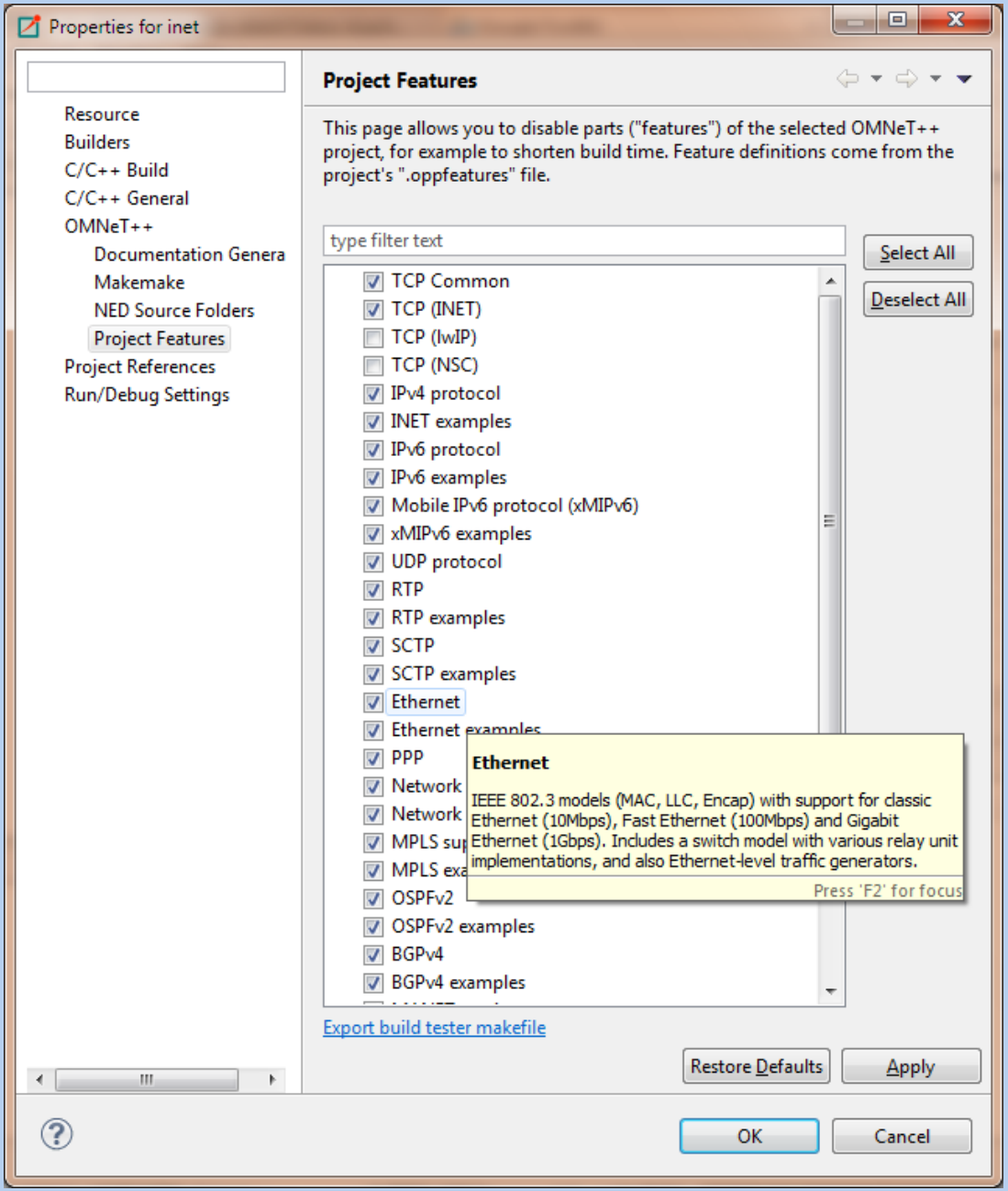
Available

- Modularity*
  - Project Features
- Statistics
  - signal-based statistics recording
- NED refactoring
  - for consistency and extensibility
- Flexible network configuration* (IPv4)
- Multi-radio

Missing (in INET)

- Detailed physical layer modeling (MiXiM)
- Node failure/recovery
- Battery
- Obstacles, etc.

Modularity: Project Features



Currently 39 features:

- 23 protocols
- 16 examples

Flexible Network Configuration

GOAL:

Replace routing files and *FlatNetworkConfigurator* with something better...

- Problem with *FlatNetworkConfigurator:*
  - all-or-nothing
  - no subnetting
  - per-node addresses instead of per-interface
  - ...
- Problem with routing files
  - (on the next slide)

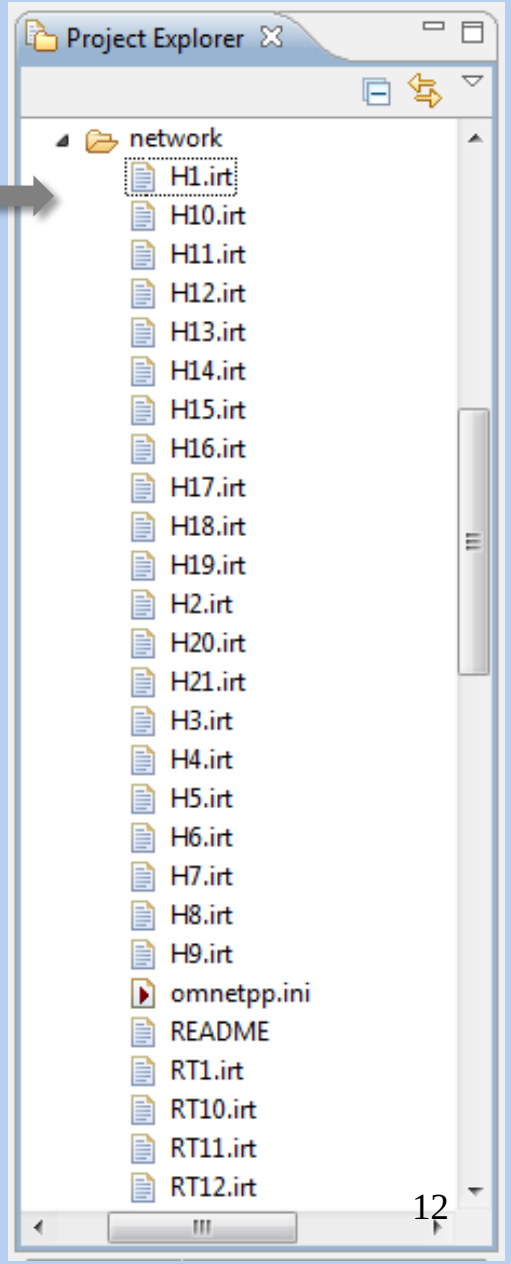
Flexible Network Configuration

- Problem with routing files (.irt/.mrt):
  - too many of them (one per router/host)
  - contains concrete IP addresses and interface names:

bgrouter3.irt | UDPSocket.h | UDPControlInfo.msg | IPv4Cc

```
ifconfig:
name: ppp0  inet_addr: 172.0.2.1    MTU: 1200   Metric: 1
name: ppp1  inet_addr: 172.1.0.0    MTU: 1200   Metric: 1
ifconfigend.

route:
172.0.0.0       172.0.2.0       255.255.0.0     G   0   ppp0
default:        172.1.0.1       0.0.0.0         G   0   ppp1
routeend.
```



→like a puzzle!
Good luck getting an overview without pen and paper

Flexible Network Configuration

The new network configurator:

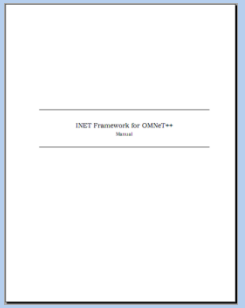
- Replaces other configurators AND routing files
- For manual configuration:
  - all configuration input in one file, not in 1000!
  - symbolic names instead IP addresses wherever possible!
  - more intuitive interface selection (“the interface towards “router7” instead of interface name “ppp2”)
- For automatic configuration:
  - per-interface addresses, subnetting support, all steps of configuration optional, optimized routing tables,...
- All config in a single XML file

DOCUMENTATION

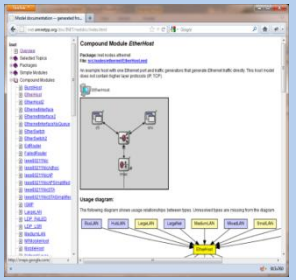
Documentation

IN WORK

- INET Manual:
  - chapters already covered: base architecture, IPv4, Ethernet, PPP, UDP, TCP
  - note: source code of components needs to be reviewed before documentation can be written
  - help welcome



- NED documentation:
  - some of the models are fairly well documented
  - others have no or inappropriate (copy/pasted) comment block



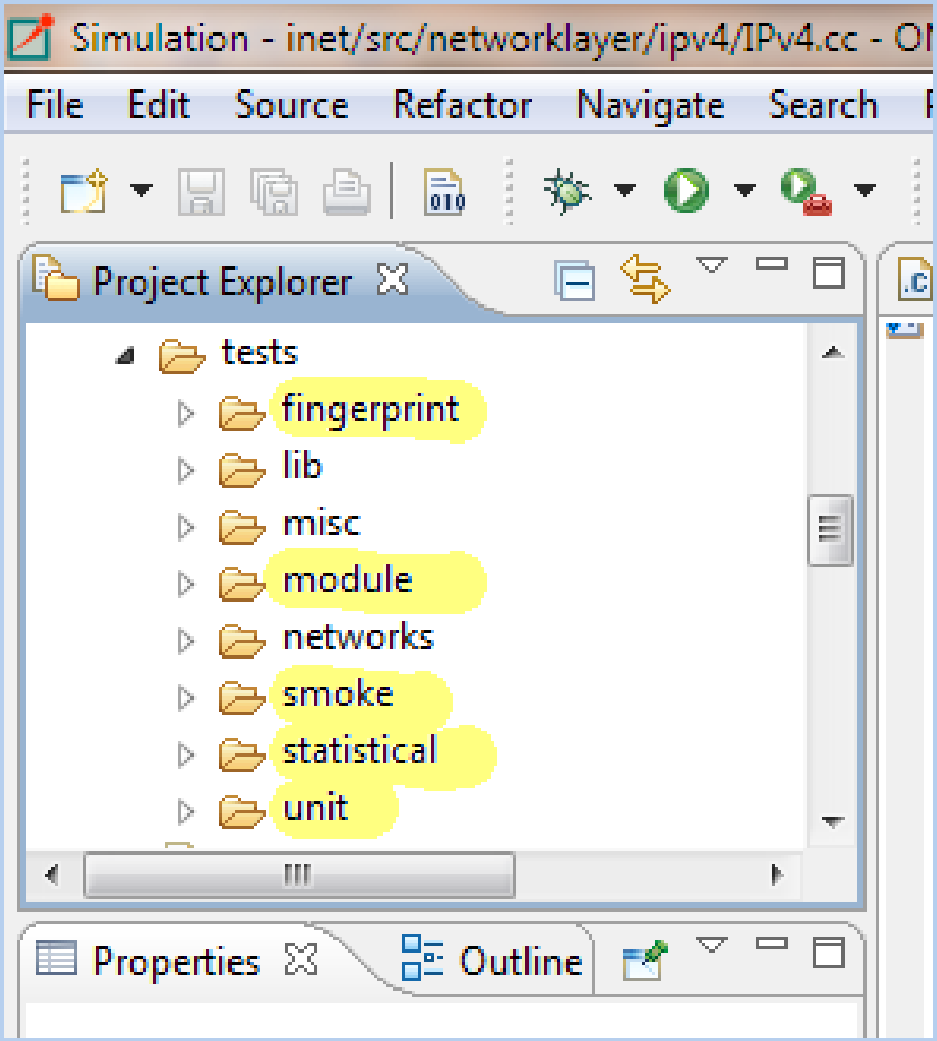
TESTING AND VALIDATION

INET Test Suite

Purpose: to create and maintain confidence in the models

ONE KIND OF TEST DOESN'T CUT IT!

1. Smoke tests
2. Fingerprint tests
3. Unit tests
4. Module tests
5. Statistical tests



Smoke Tests

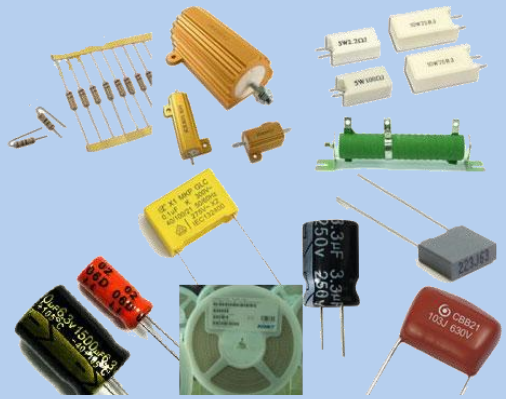
- Run the simulation for a while, and see if it crashes or stops with a runtime error
  - simplest kind of test, provides low confidence in the models
  - crude but easy to implement
  - INET smoke tests:
    - `smoketest` script + csv file (columns: working-dir, command-to-run)
    - script runs all example simulations with cpu-time-limit=3s

Fingerprint Tests

- "What is fingerprint again?"
  - hash of certain properties of the simulation, currently (time, module ID) for each event
  - designed to change if simulation trajectory changes
  - suitable for regression testing
- Fingerprint tests:
  - **fingerprints** script; runs example simulations plus some test simulations; input in CSV

```
# workingdir,                 args,                                  simtimelimit,  fingerprint
/examples/adhoc/ieee80211/,   -f fingerprints.ini -c Ping1 -r 0,      1000s,         621c-2640
/examples/adhoc/ieee80211/,   -f omnetpp.ini -c Ping1 -r 0,          100s,          0cad-2371
/examples/adhoc/mf80211/,     -f fingerprints.ini -c Ping1 -r 0,      1000s,         a867-4a02
/examples/bgpv4/BGP3Routers/, -f omnetpp.ini -c config1 -r 0,        1000s,         3fac-2c12
/examples/bgpv4/BGPandOSPF/,  -f omnetpp.ini -c config1 -r 0,        1000s,         5161-8ab8
/examples/ethernet/arptest/,  -f omnetpp.ini -c ARPTest -r 0,        500s,          e1f3-3ca1
/examples/ethernet/lans/,     -f bus.ini -c BusLAN -r 0,             100s,          9999-0785
```

Unit Tests



- For testing individual classes
  - MACAddress, IPv4FragmentationBuffer, TCPMsgBaseReceiveQueue, ByteArrayPacket, HeaderSerializer, Coords, ErrorRateModel, etc.
  - use OMNeT++ unit test framework (`opp_test`) and `.test` files

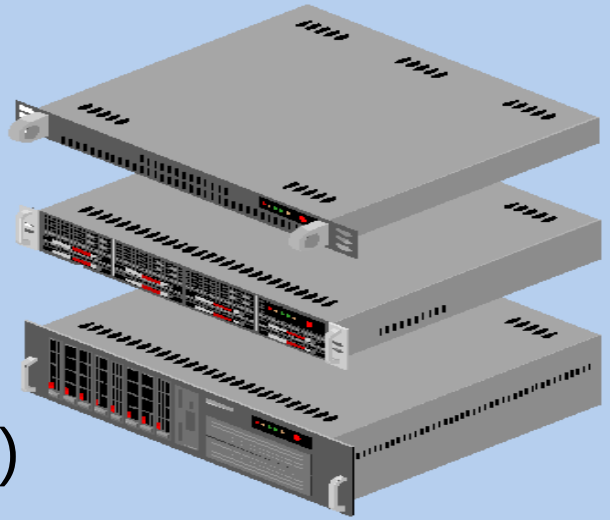
```
%description:
Tests TCPMsgBasedSendQueue, TCPMsgBasedRcvQueue classes

%activity:
...
enqueue(sq, "msg1", 100); // 1000..1100
enqueue(sq, "msg2", 400); // 1100..1500
...

%contains: stdout
[1000..1000), 0 packets
rcv_nxt=1000 0 msgs
SQ:enqueue("msg1", 100): --> [1000..1100), 1 packets
SQ:enqueue("msg2", 400): --> [1000..1500), 2 packets
SQ:enqueue("msg3", 600): --> [1000..2100), 3 packets
...
```

MORE TESTS NEEDED

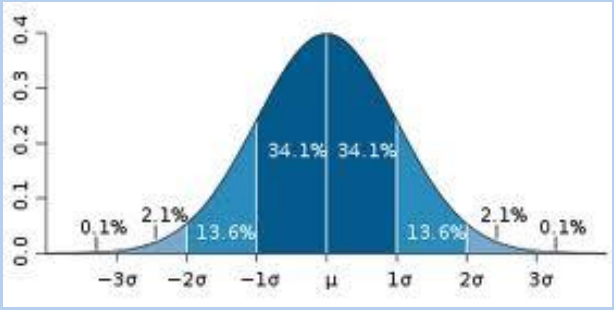
Module Tests



- Functional test of individual modules, typically protocol implementations
  - send the module some input, then check how it reacts (messages and/or log output)
  - OMNeT++ unit testing framework (`opp_test`) can be used
  - we are also considering Python for scripting

MORE TESTS NEEDED

Statistical Tests



- Statistical regression tests
  - check that model produces statistically the same results as before
    - e.g. perform 100 runs “before” and “after” a change, and use Student t-test [for mean] and F-test [for variance] to check that both set of results are from the same distribution

      implementation: `inet/tests/misc/statistical/test.R`
- Validation tests
  - e.g. performance tests: throughput corresponds to expectation (theoretical values, physical measurements, or other simulator’s results)
    - we have such tests for Ethernet (implemented using R)
    - TODO: reuse results of 802.11 model validation workshop paper

TBD

Automated Build Testing

Benefits:

- build errors and broken test cases are usually detected earlier
  - (even though our INET tests run only once a day)
- tests for you on other platforms
  - i.e. develop on Windows, test on Linux or vice versa
- it is for the lazy
  - after a change, it is less effort to push “Start Build” button on a web page than run the test suite manually on your own computer!

Automated Build Testing

We use:

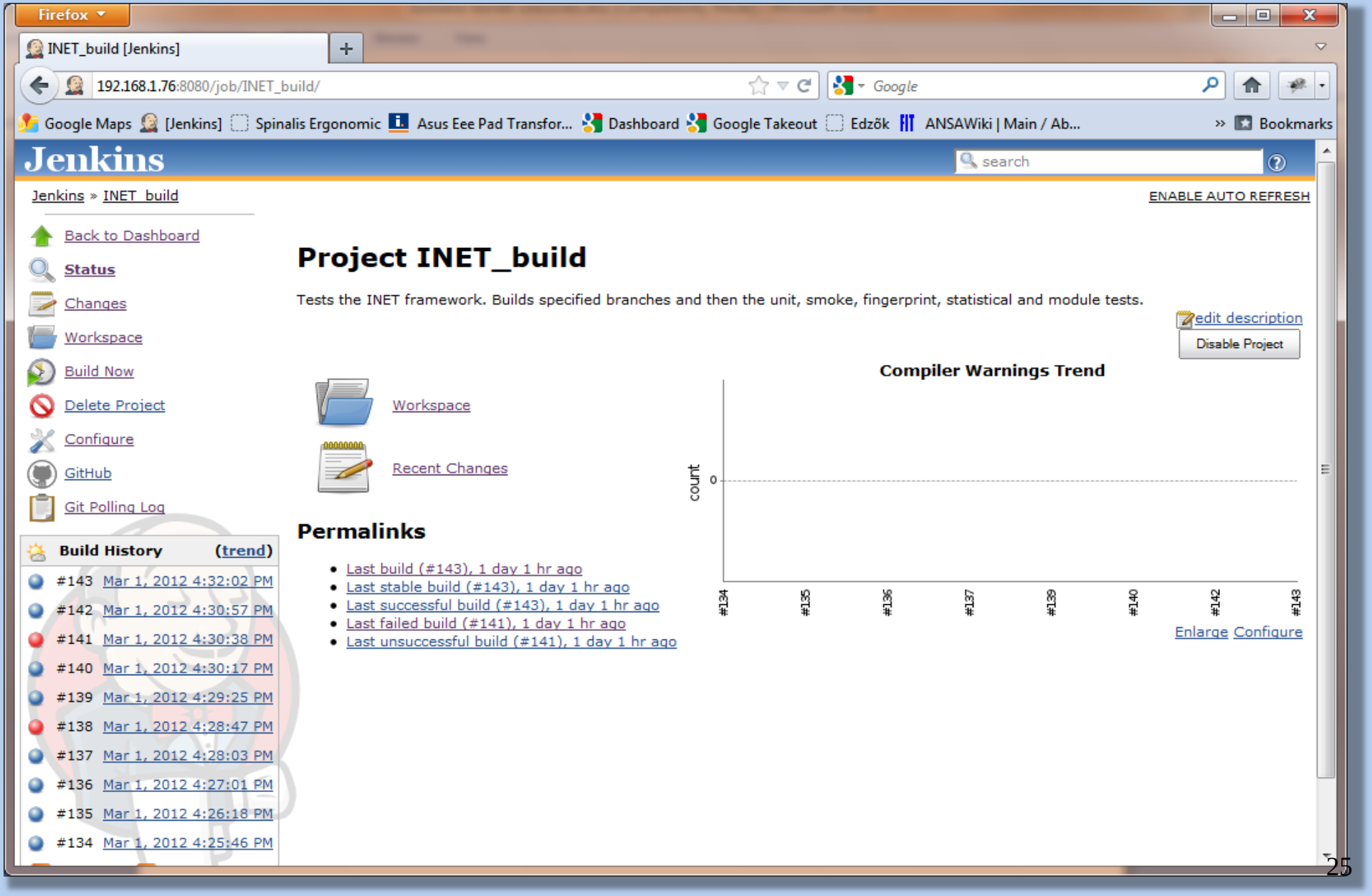
Jenkins

http://jenkins-ci.org

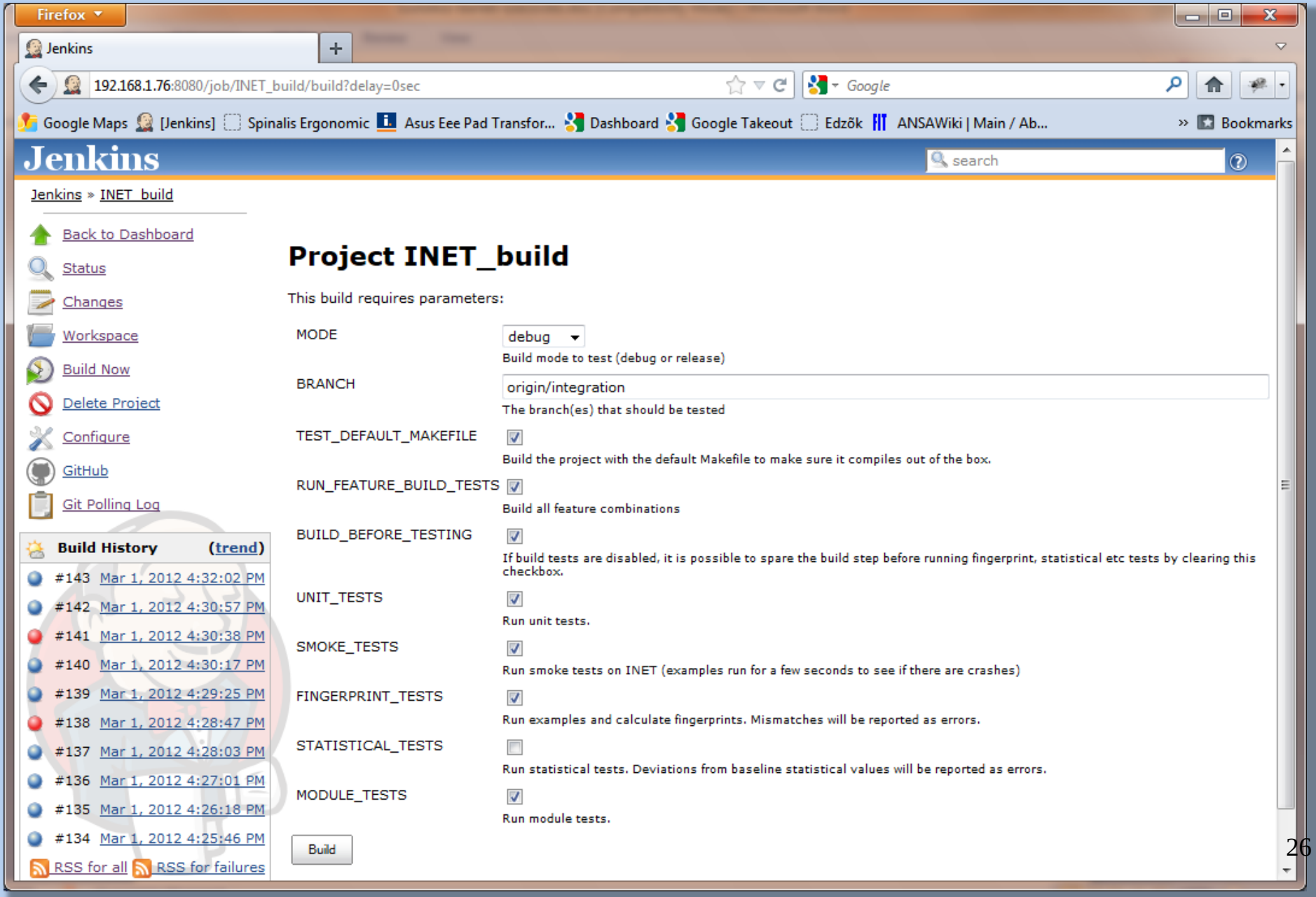
"An extendable open source continuous integration server"

- Packaged for multiple Linux distros, Windows, OS X, etc. (we use it on Ubuntu)
- Web-based administration; builds can be triggered by scheduling (cron), by commit, or manually; lots of plug-in extensions for various purposes (400+)
- How we use it for INET: checks out latest INET (given branch) from github repo, builds it with different feature combinations, runs test suite, reports results; runs once every night
- IF YOU WANT TO SET UP YOUR OWN JENKINS: our Jenkins config file is available from the INET repo

Automated Build Testing



Automated Build Testing



COMMUNITY INVOLVEMENT

Community Involvement

We need to agree on:

- how do new protocols make it into INET?
  - code review
  - formal requirements* (documentation, commenting, code style, existence of examples and tests)
- how do patches make it into INET?
  - formal requirements (clear statement of what it solves, etc.)
  - code review
  - tests* (to demonstrate that it solves the problem, and doesn't break anything else)

* if author does not provide them, someone else has to do it

Community Involvement

- The OMNeT++ team can do much, but...
  - some tasks require domain knowledge, i.e. help from the community
    - **code review** (for conformance)
    - **validation**
    - **setting priorities**
- Forum, tools
  - inetframework-devel@googlegroups.com
  - Gerrit code review tool (if proves useful)
  - Is there interest / willingness to participate?

ANIMATION

Animation

What do we want to animate/visualize?

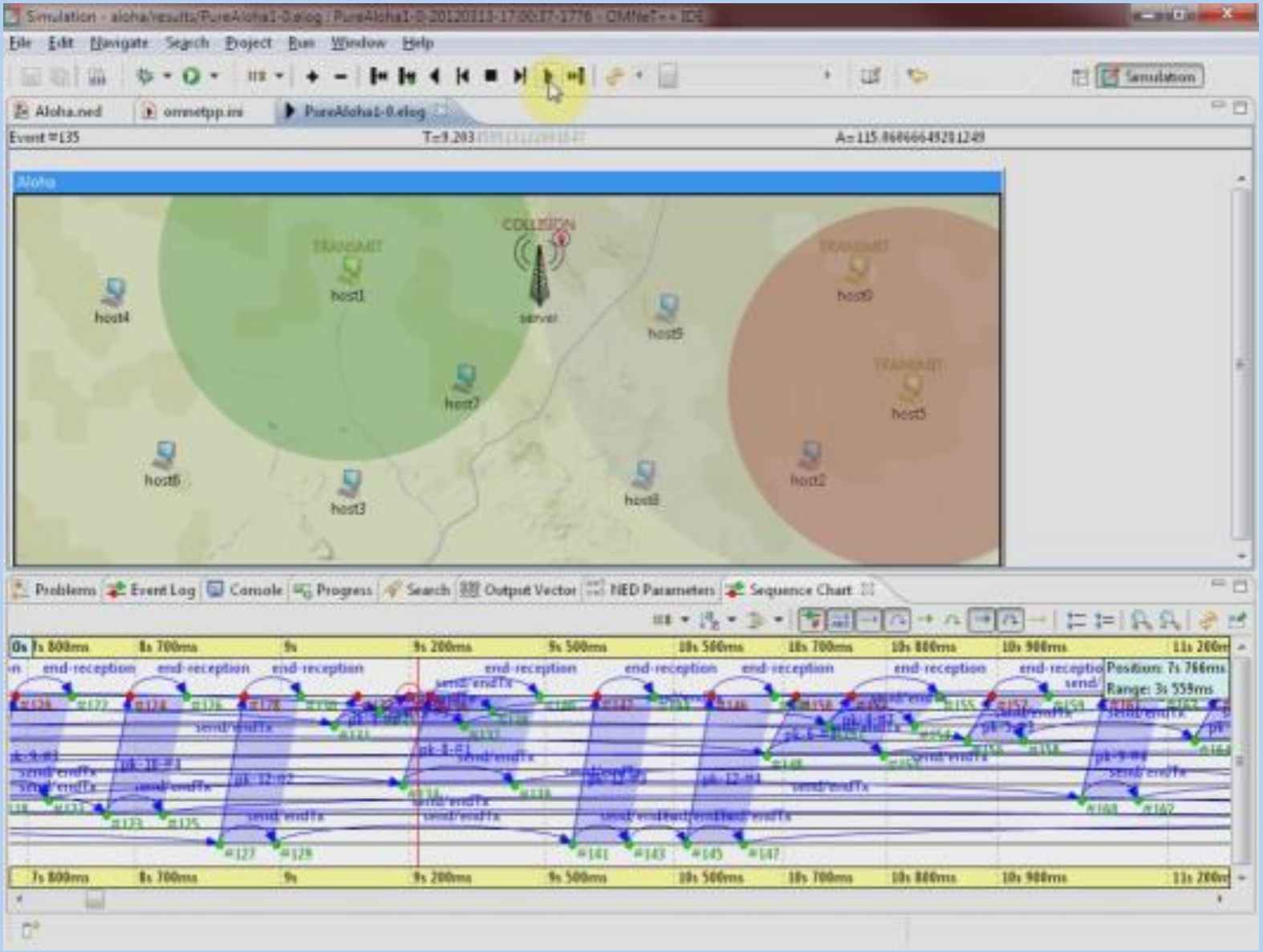
- frame transmissions, wired/wireless
  - wireless: dest node? successful?
- node movement, e.g. trajectories
- higher-level information: reachability, routing, overlay network topology, ...
- vital statistics (as annotation, graph, chart, gauge or meter)
- ...
- [you name it]

Animation

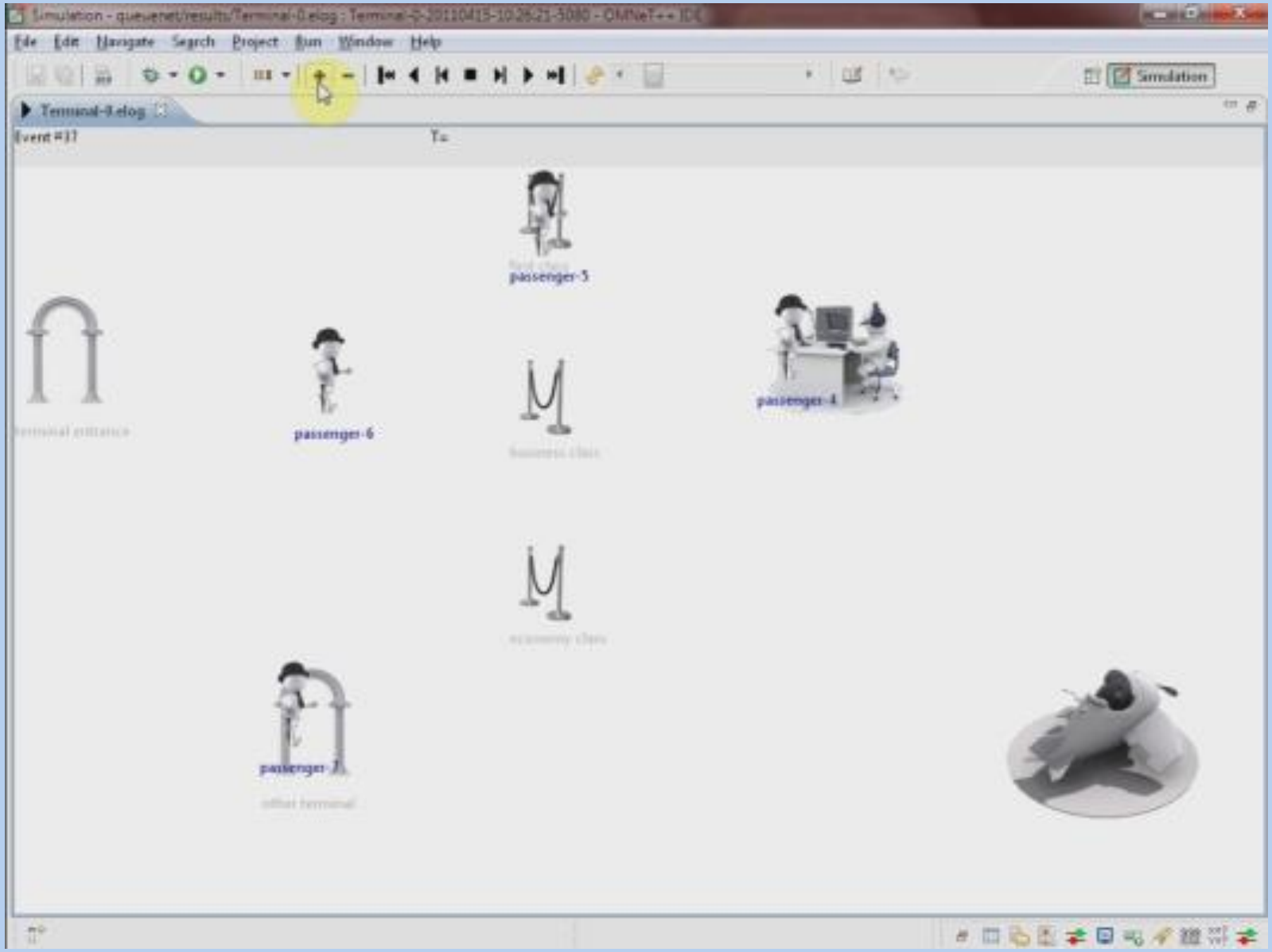
IN WORK

- Animation Framework
  - extends the IDE
  - input: eventlog files + model specific files
  - like an interactive video player
    - time linear/nonlinear
    - content can be filtered
    - can be interactive!
  - extensible with model- (INET-) specific animations
    - support for new animation effects, visualizations, layers, interactivity, etc.
    - Java API
    - can be deployed with the model

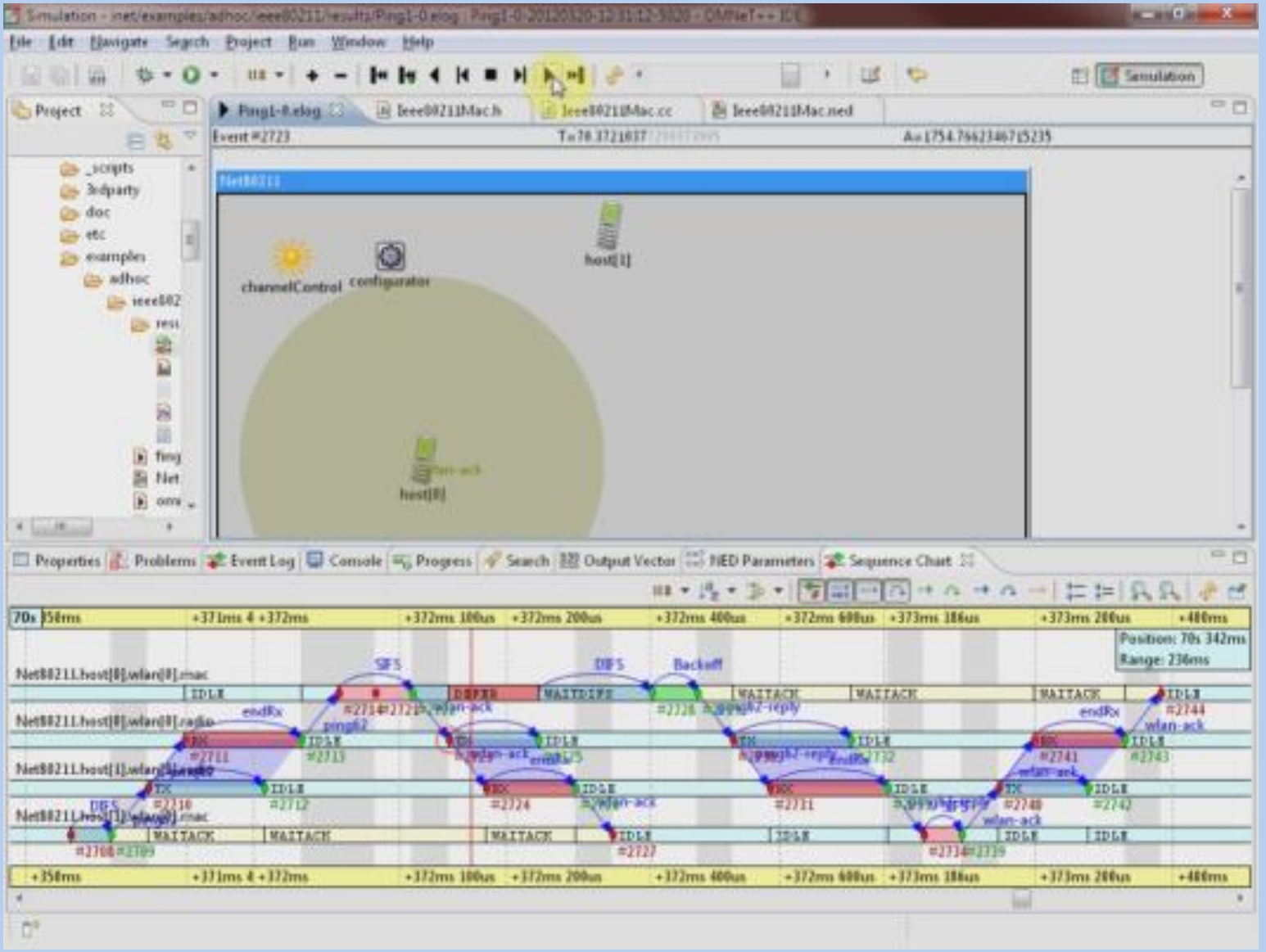
Animation Demo: Aloha



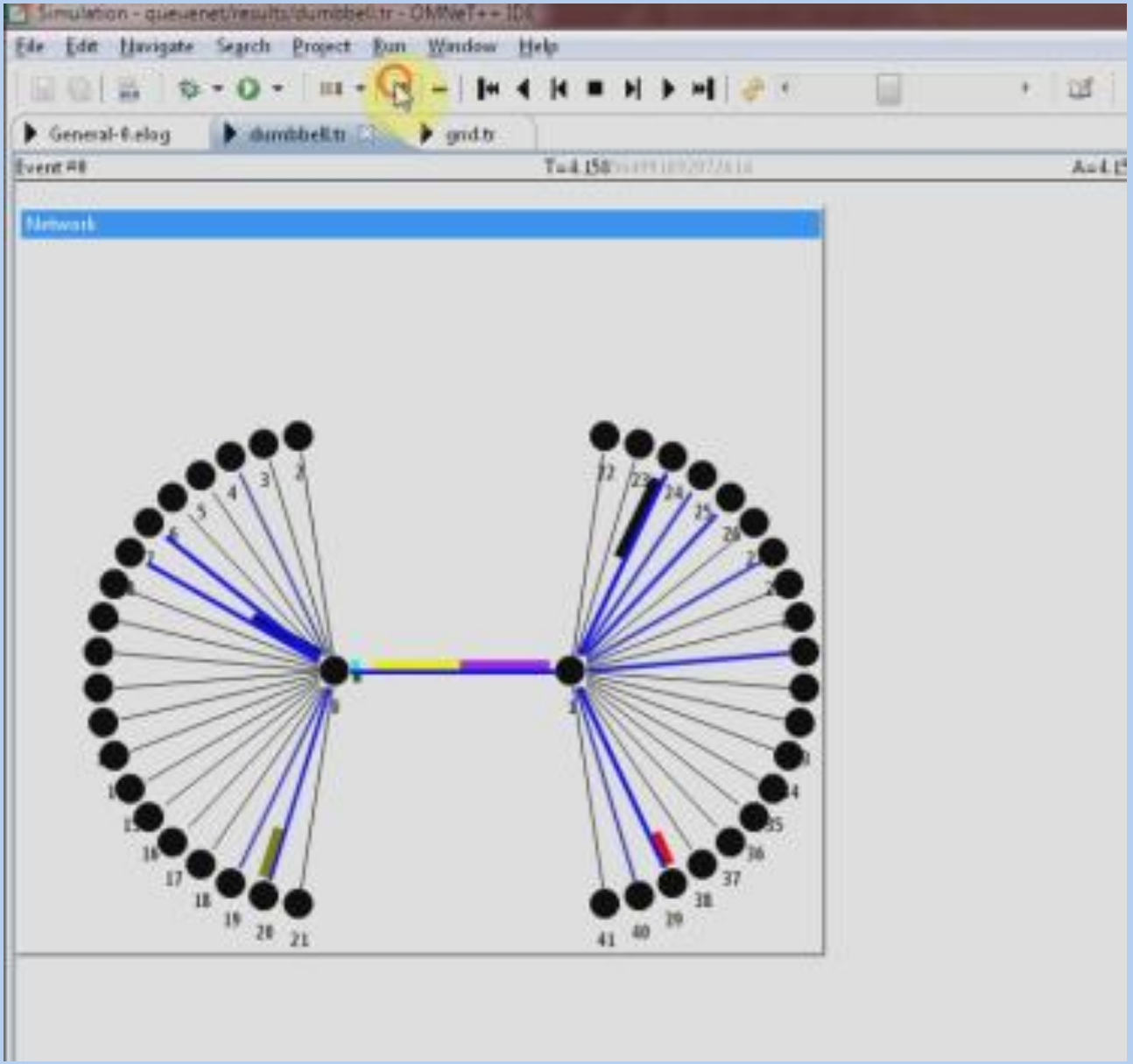
Animation Demo: Flight Terminal



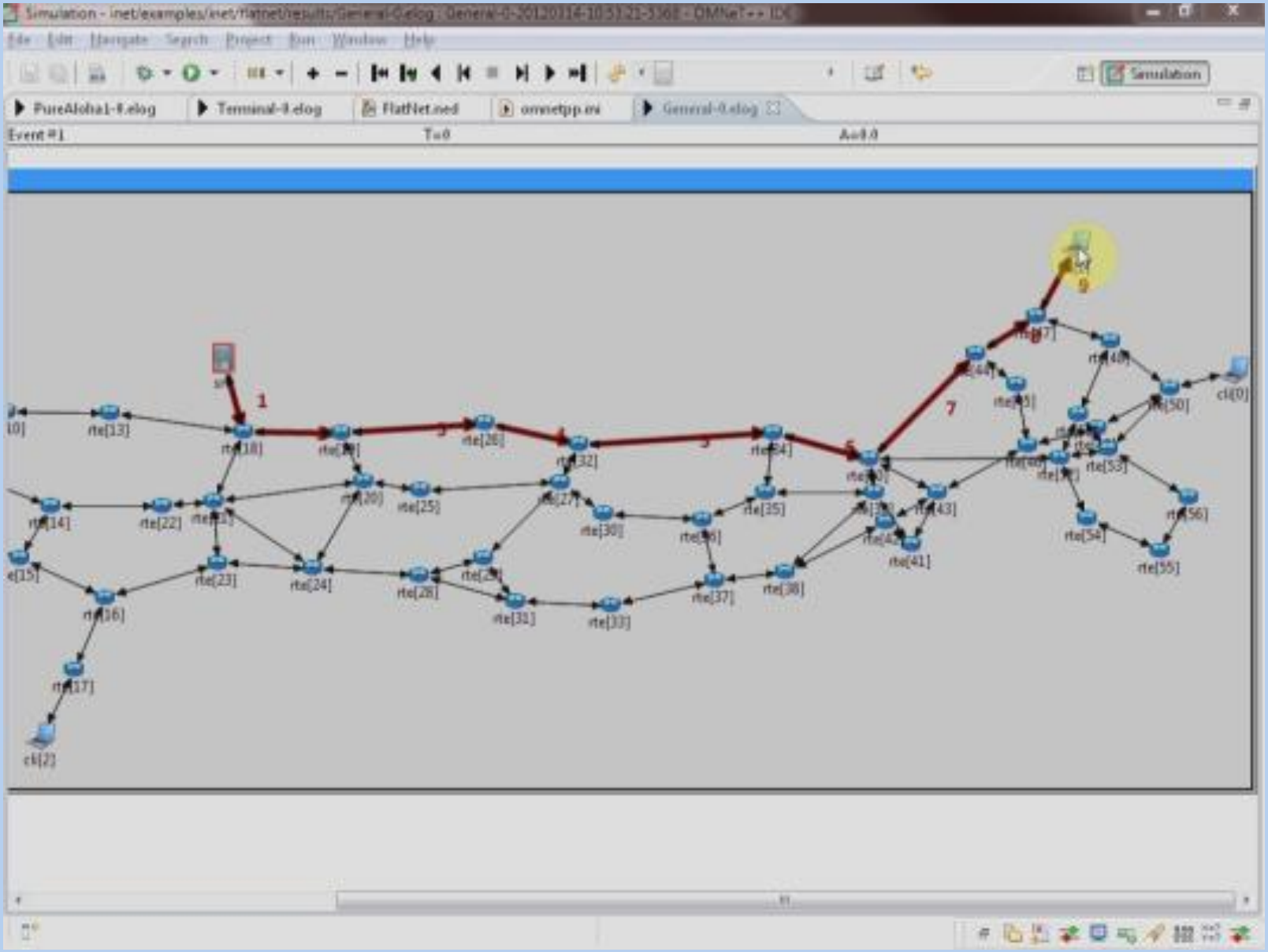
Animation Demo: 802.11



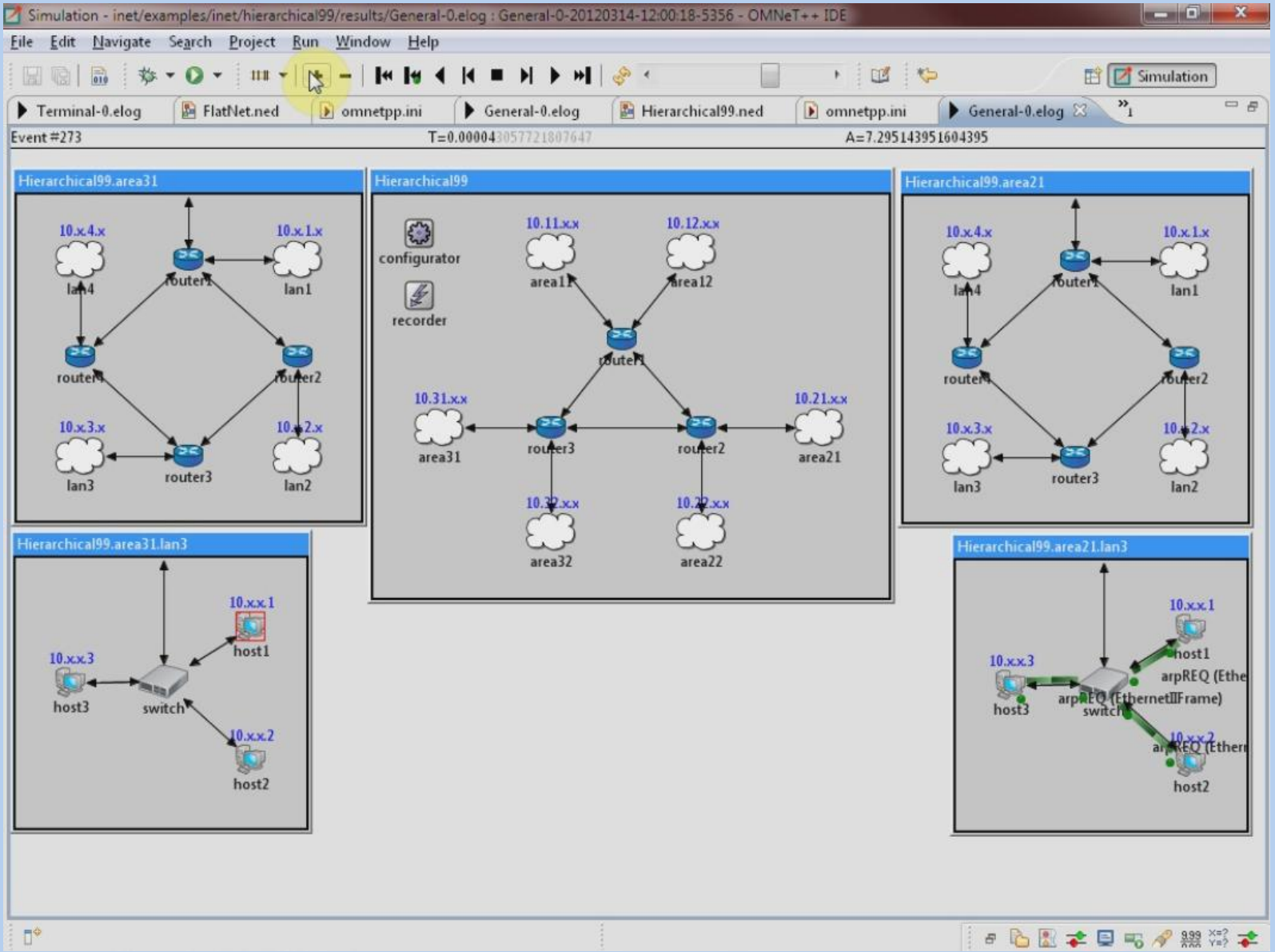
Animation Demo: Dumbbell



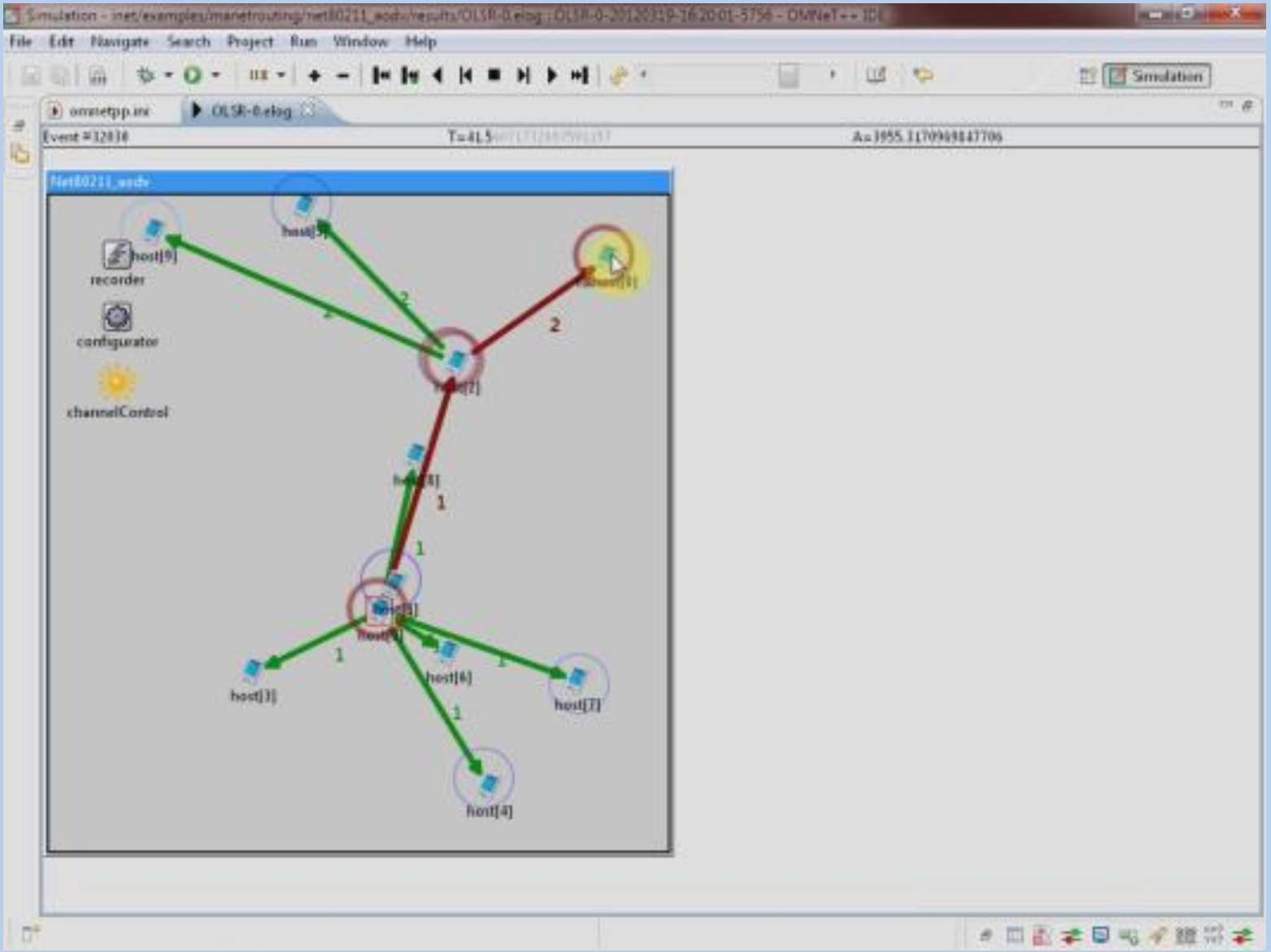
Animation Demo: Routing



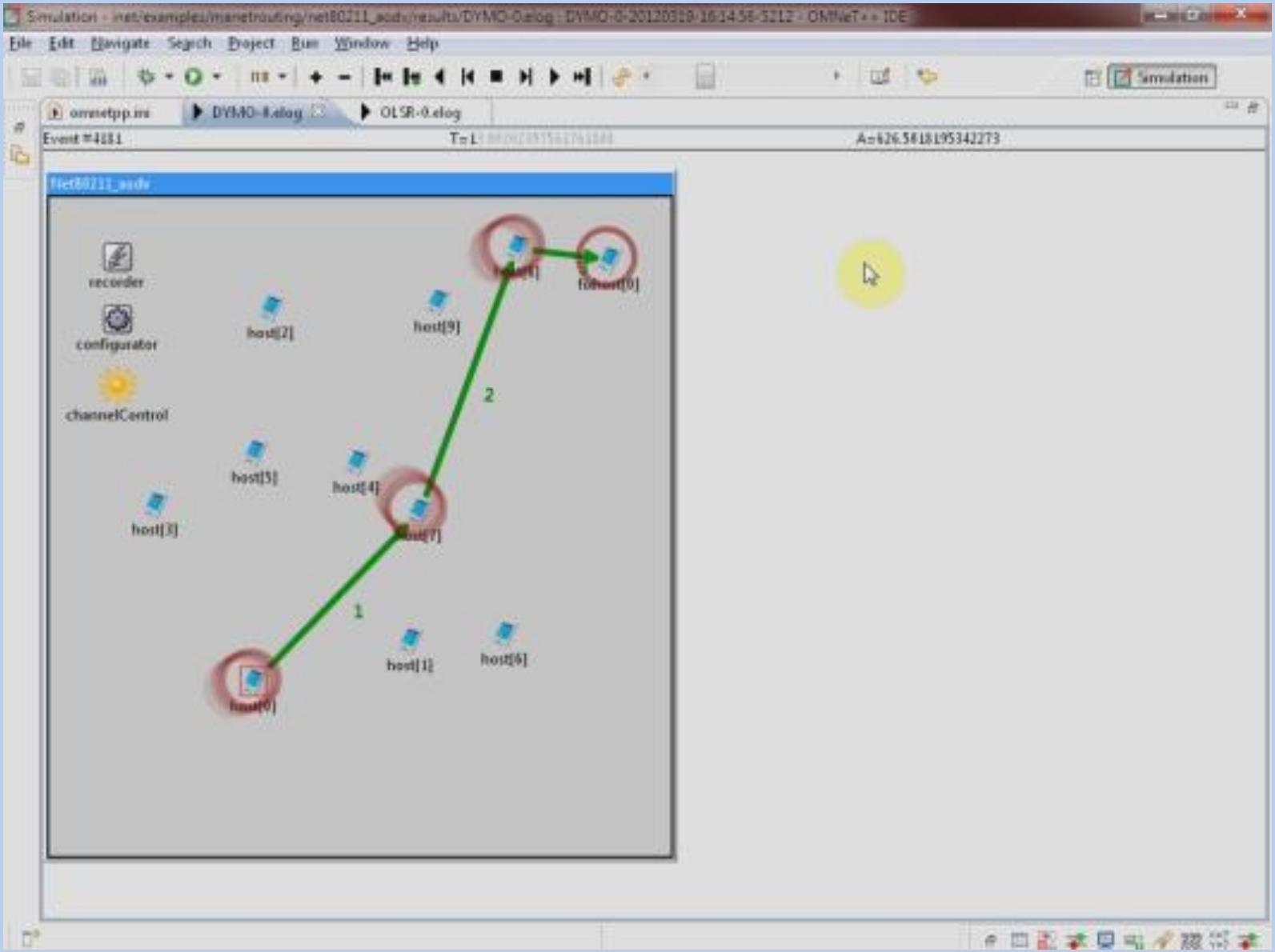
Animation Demo: Routing



Animation Demo: OLSR Routing



Animation Demo: DYMO Routing



Discussion

We only have a few minutes now, but

we can continue in the Closing Session, 17.30-18.00