Advantages of Utilizing the OMNEST Simulation Environment in Automotive Research, Testing and Verification

AMAA Berlin
July 23
2014

Andras Ferencz
(Test automation and real-time testing consultant)

From Ideas to Solutions
With Confidence

The following challenges can be addressed using OMNEST

- Rapidly increasing system complexity
- Heterogeneous networks and a variety of coding environments
  in the development
- Verify Control stability, robustness, functionality, determininsm.
- Reduce time to market and costs.

Challenges for the platform/system integrator (OEM)

- OEMs need a process where requirements can be specified and verified on the system level.
- Requirements must be defined at the component level, which can be assigned to individual suppliers.
- Cars need to be integrate with cities, people, other vehicles etc.

Challenges for the component or module developer (supplier)

- Representative stimulus signals should be developed for the component testing
- A good understanding is required about other components affecting the behavior of the component under development
- Messaging stimulus is hard to develop and maintain with parameter and design changes
- Using more and more wireless sensors

What is OMNEST?

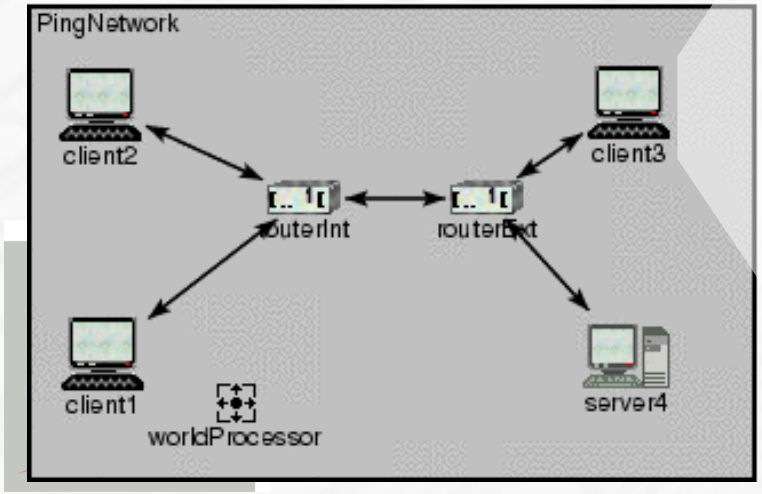
A **generic** simulation framework:

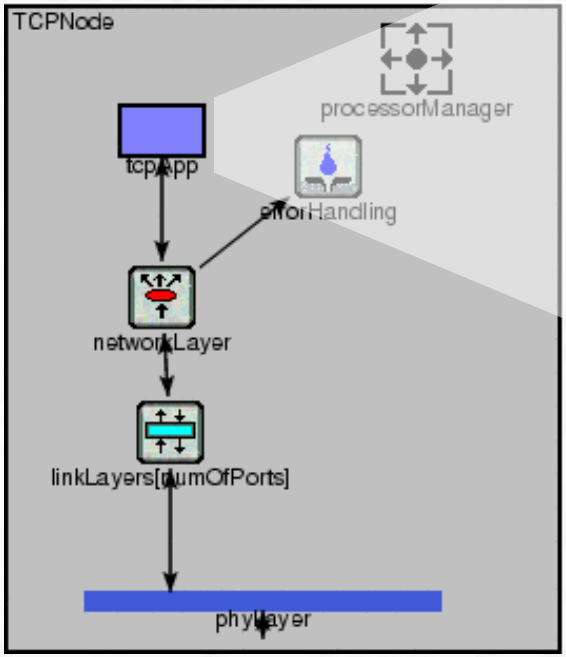
- For the simulation of **complex distributed systems**:
  - distributed hardware and software architectures,
  - communication networks,
  - queuing networks,…
- Technically: a C++-based **simulation kernel** plus a set of **libraries** and **tools** (GUI and command-line)
- An **open** environment
  - in terms of source code, embedding, extensibility, integration, modularity

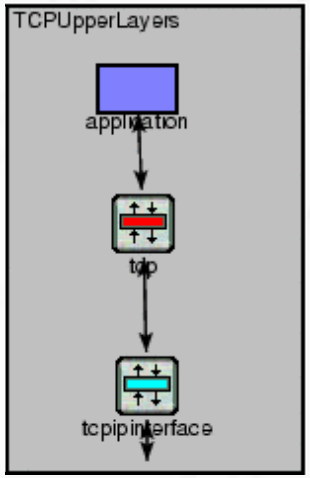
Model Structure

Component-oriented approach:

- The basic building block is a **module**.
- Simple modules can be grouped to form **compound modules**.
- Modules are **connected** with each other.







Defining the Behaviour

Behaviour is encapsulated in **simple modules**.

A simple module:

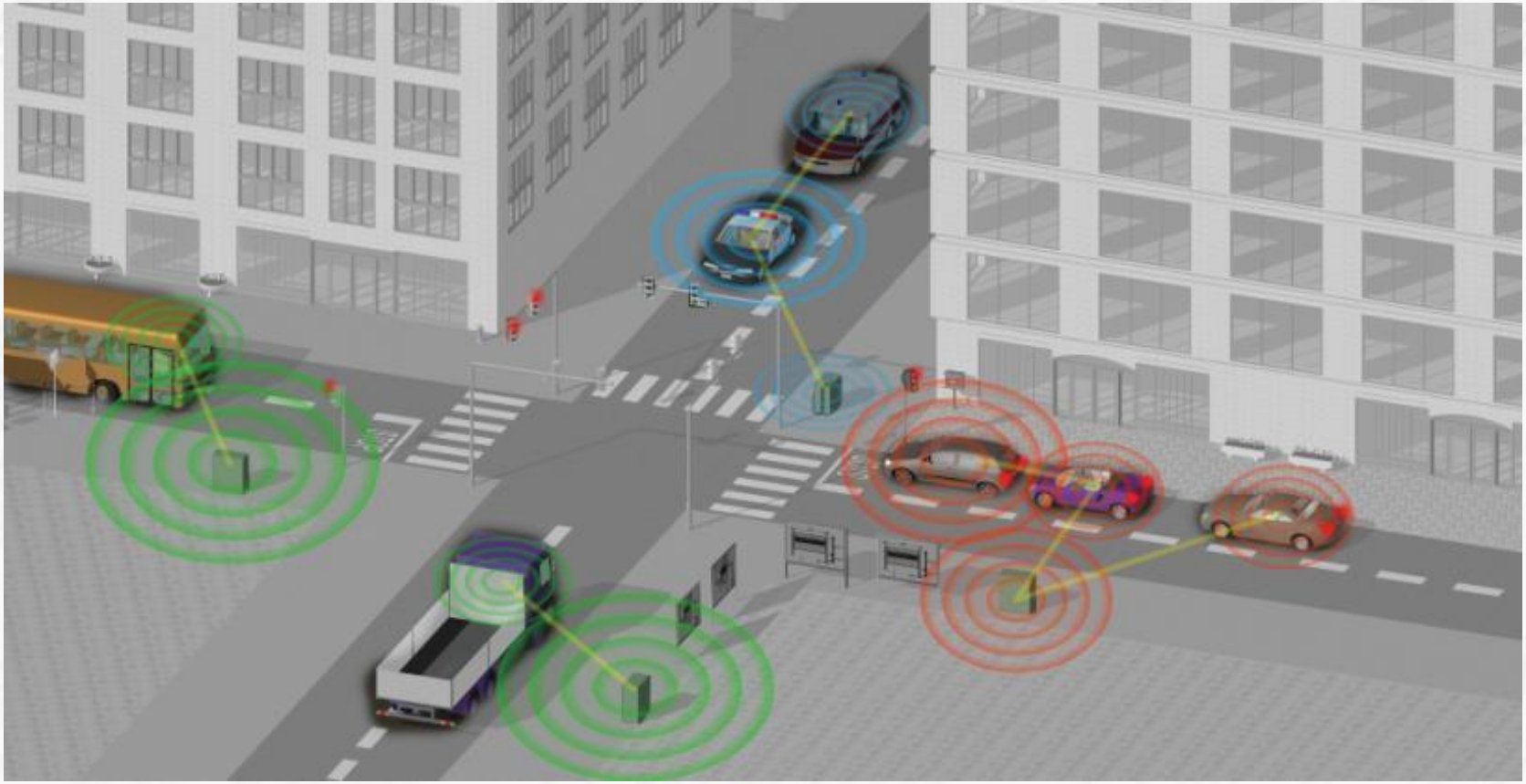
- sends messages,
- reacts to received messages
- collects statistics

Simple modules are programmed in C++.

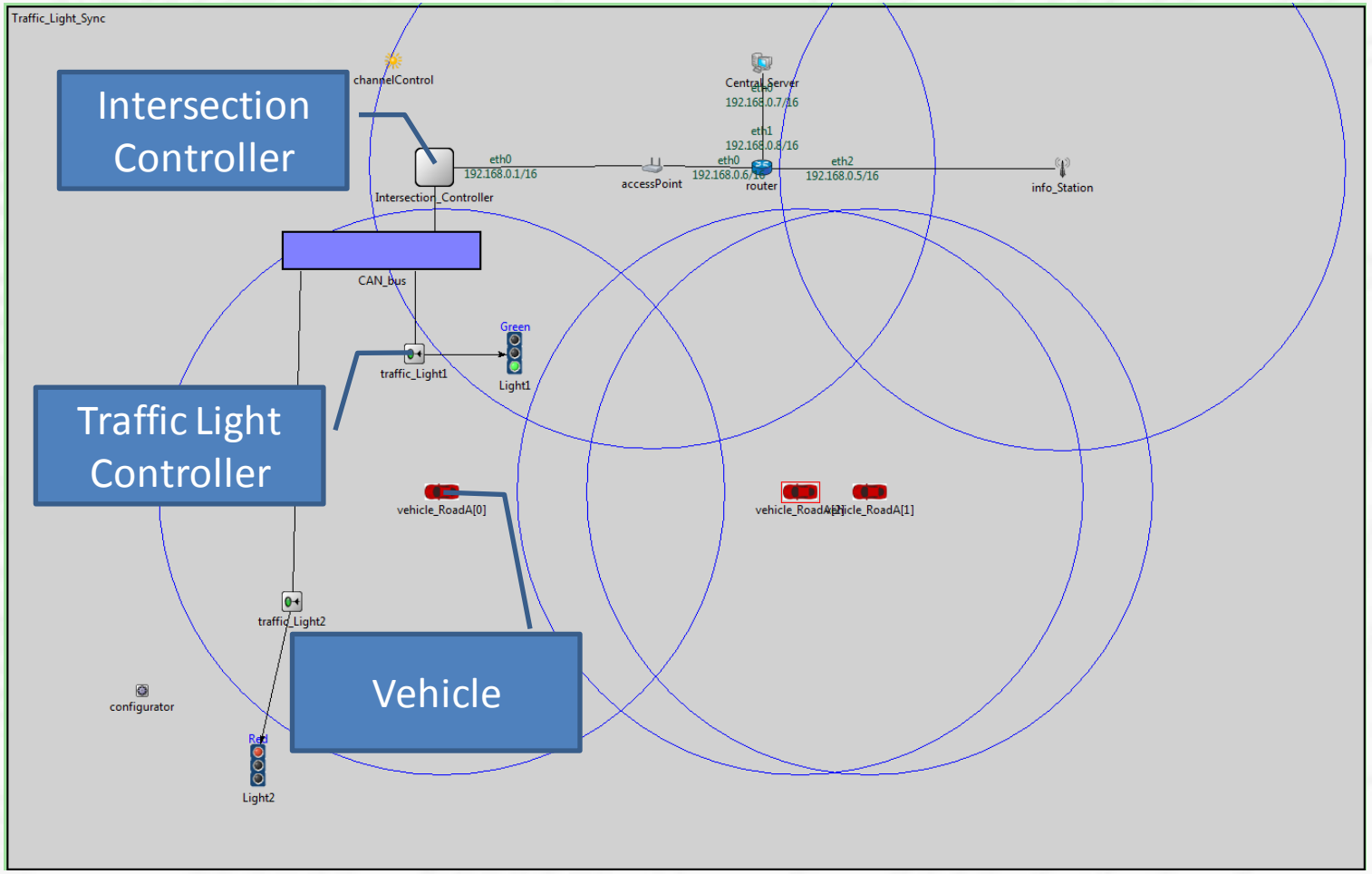
Simulation Models Available for Several Domains

- **Communication network protocols**: TCP, IPv4/IPv6, Ethernet, VoIP, WiFi, ad-hoc wireless networks...
- **Automotive protocols**: CAN, LIN, DC-BUS, FlexRay, IEEE 802.1 AVB
- Wired and wireless sensor networks
- Support for Hardware-in-the-Loop simulations

Scenarios

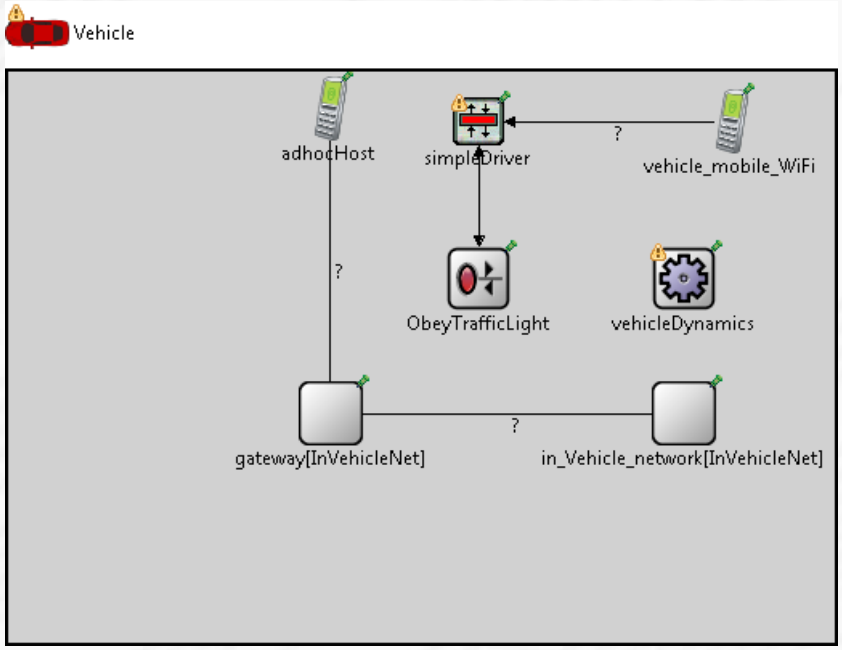


Example application using both in- and inter-vehicle communication

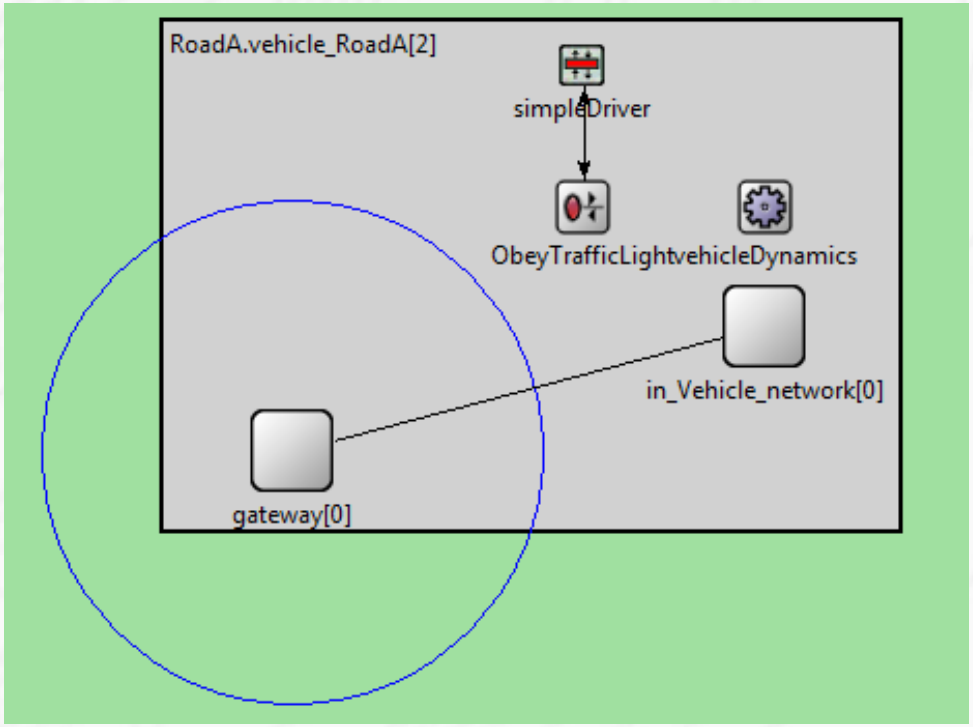


Vehicle Model

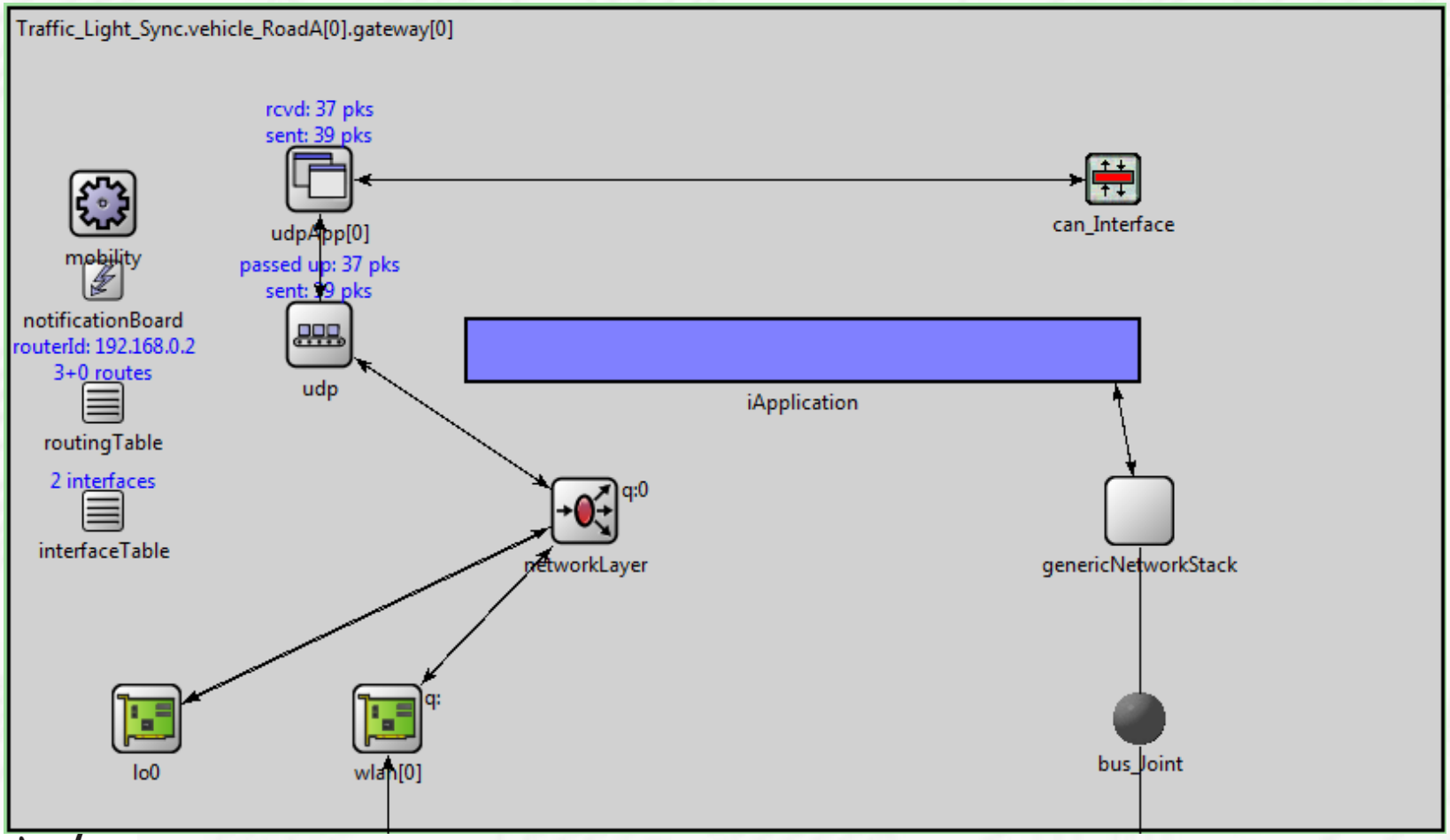
General Vehicle Model



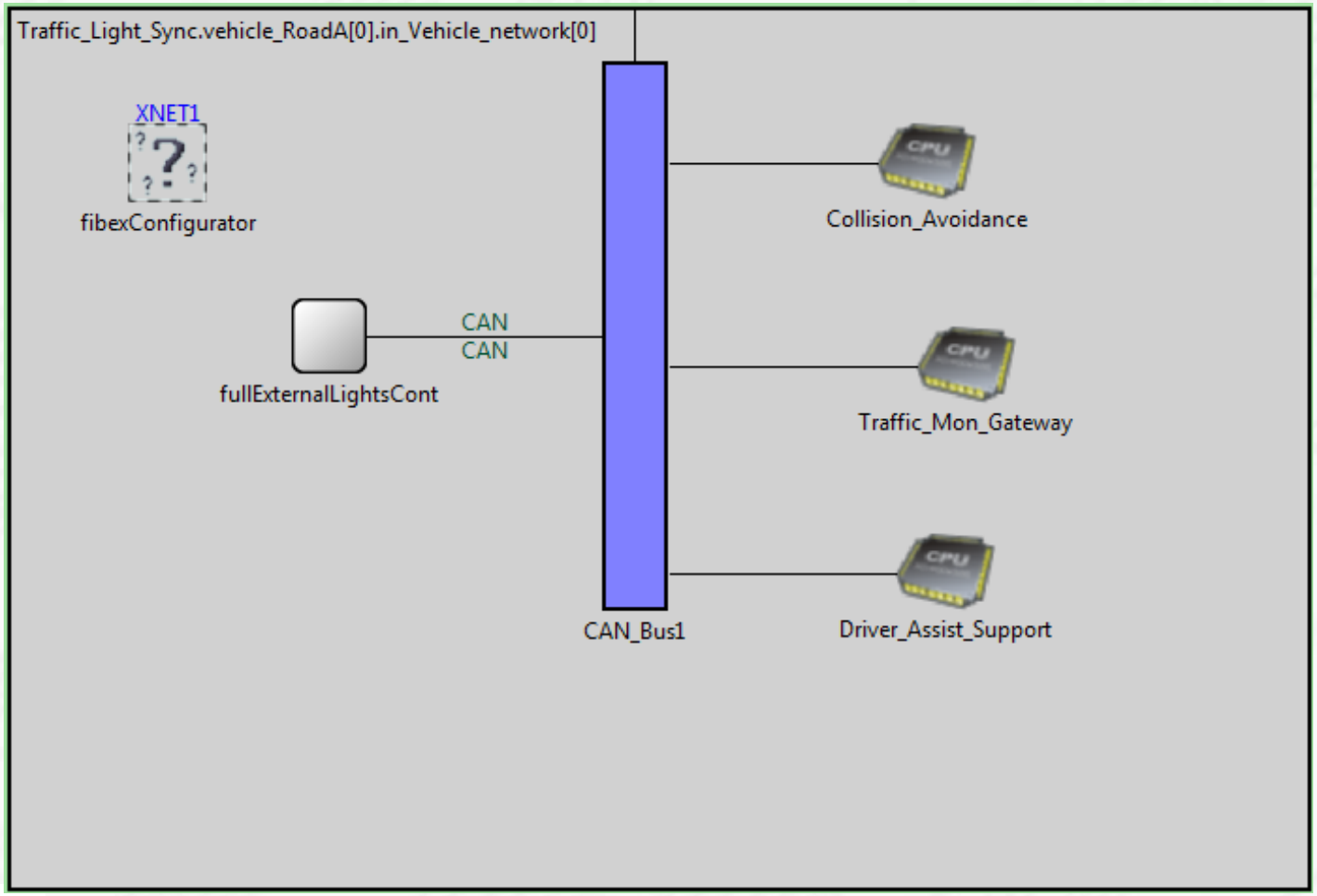
Specific configuration of the Vehicle Model in one of the cars during the simulation



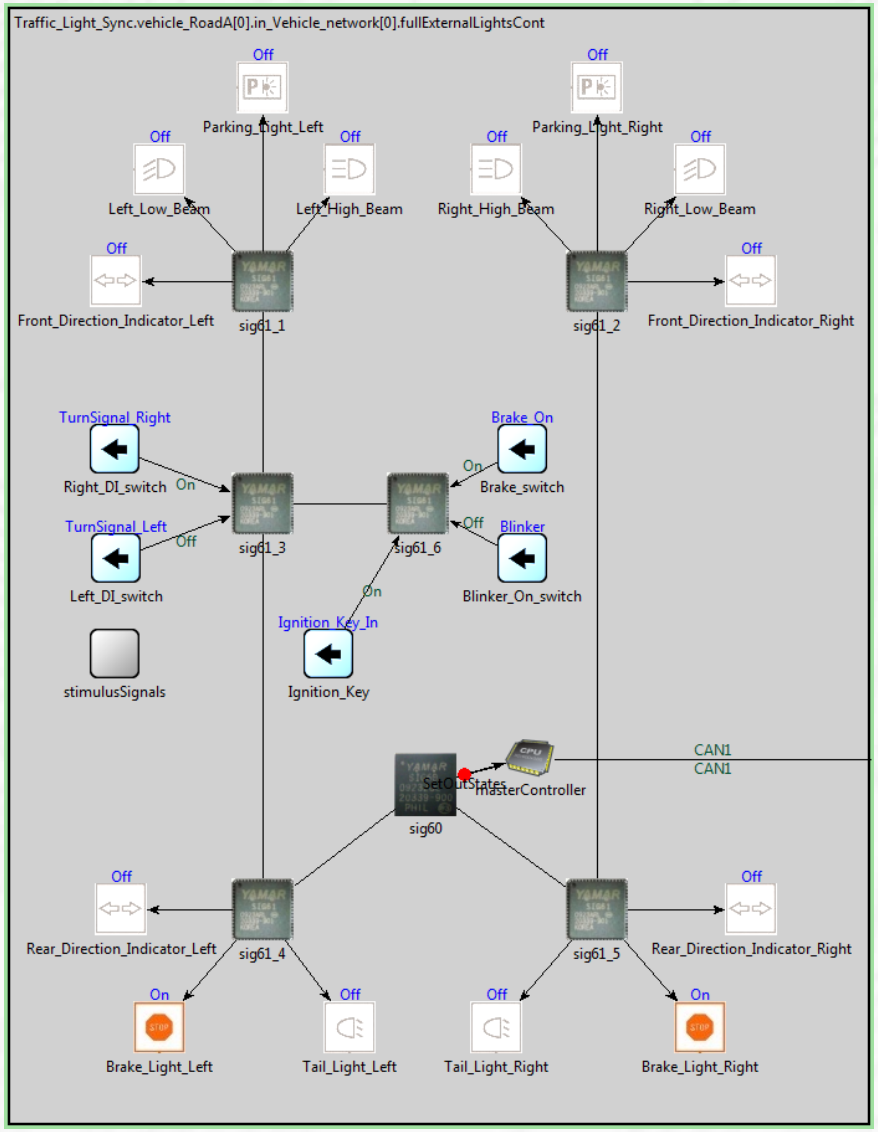
Gateway Model



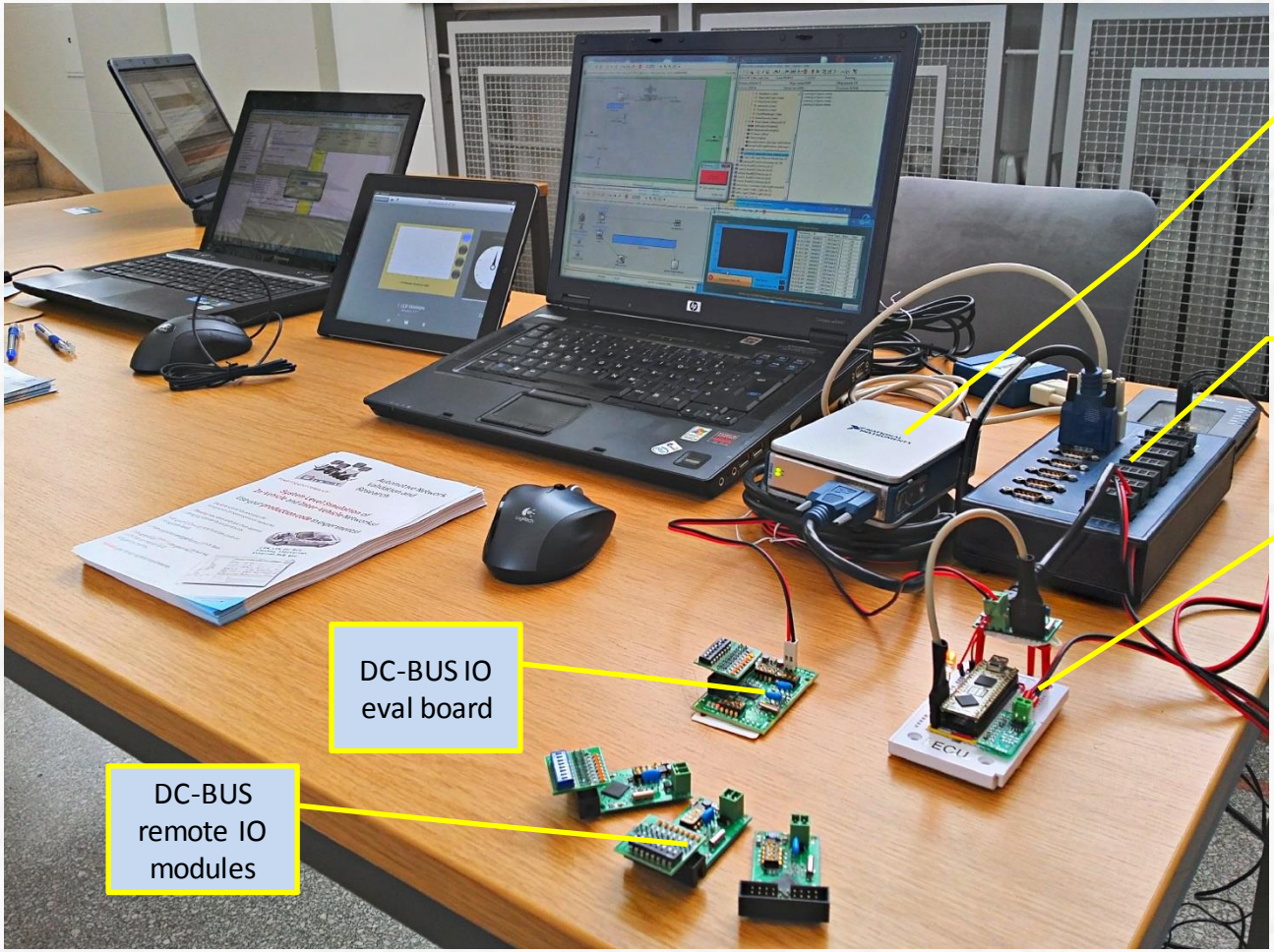
In-Vehicle Network Model



Full External Lights Control model

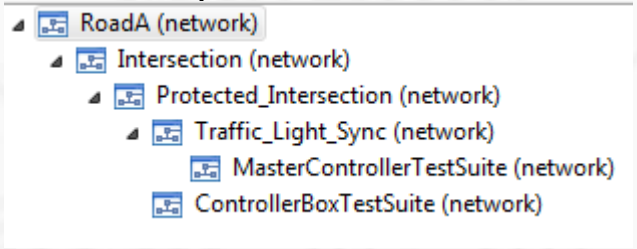


Demonstration Setup

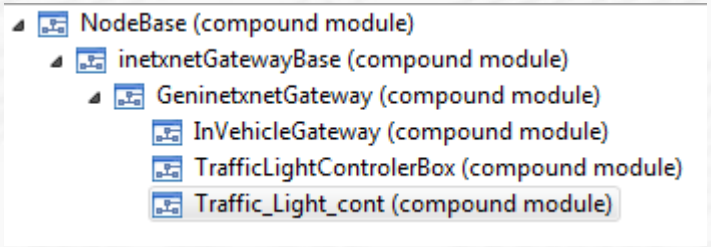


Use inheritance

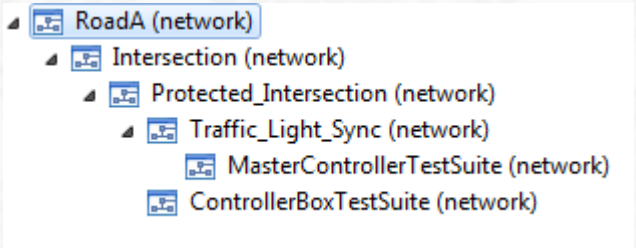
- Network/system inheritance

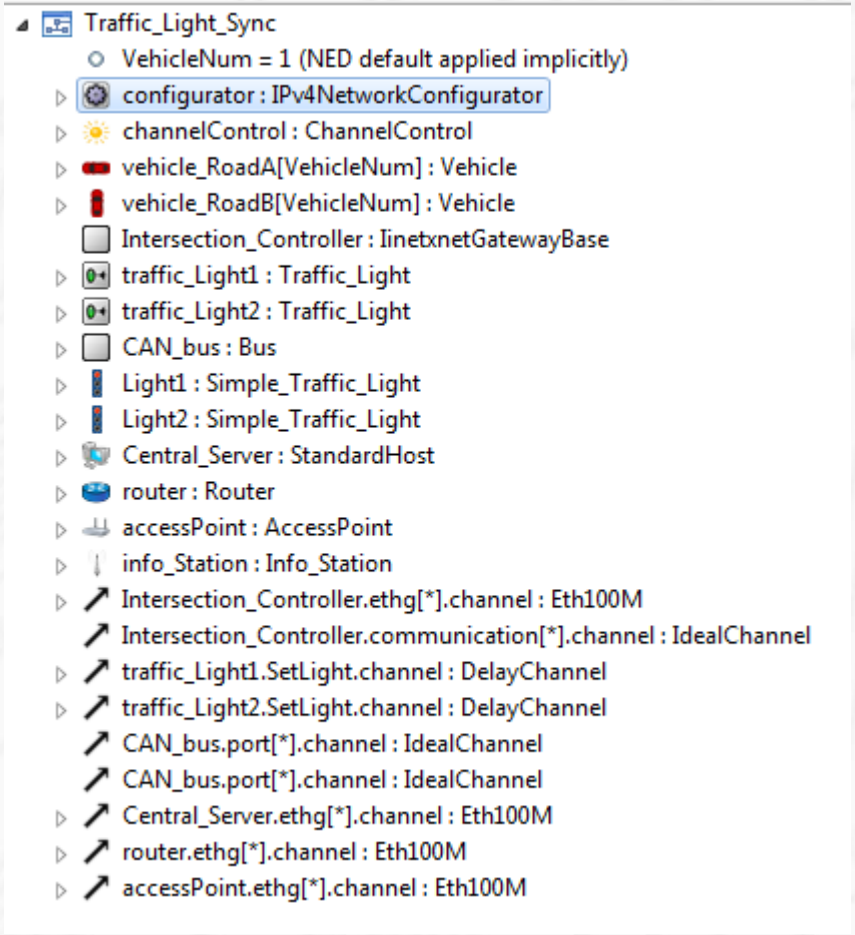


- Module inheritance



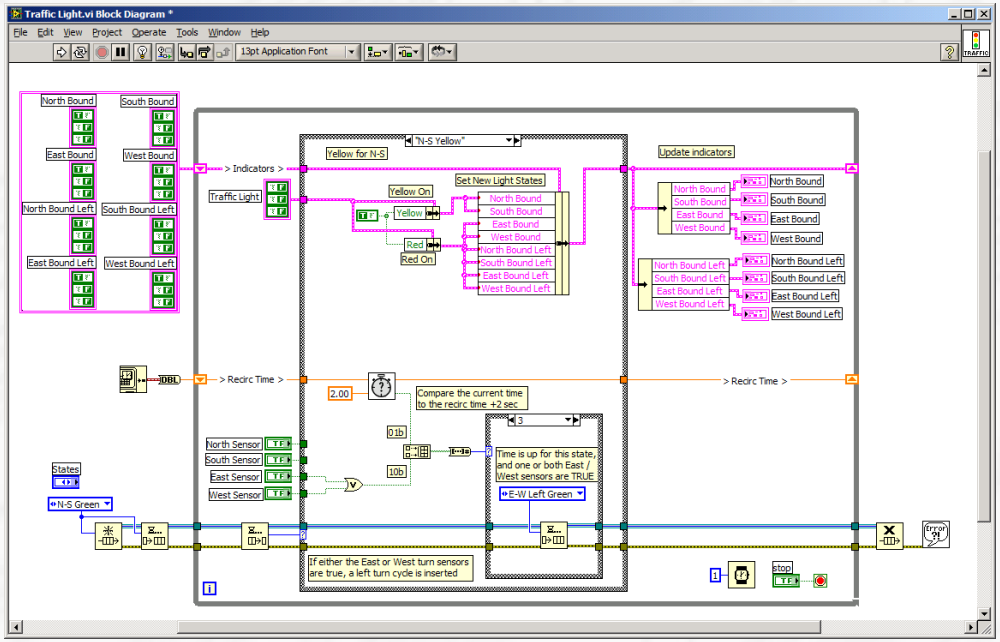
- Initial parameter inheritance (Configuration)





Reuse existing code and custom libraries

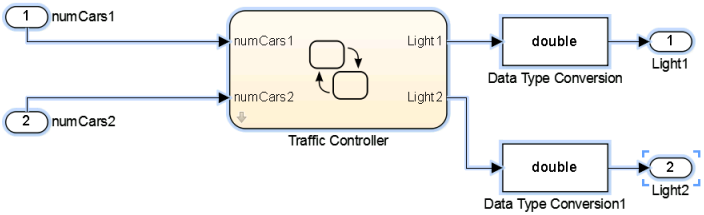
Application layer control code can be integrated for testing functionality




```
void UDPPositionApp::handleMessage(cMessage *msg)
{
  updateMyPosition();

  if (msg->isSelfMessage())
  {
    processTimer(msg);
  }
  else if (msg->getKind() == UDP_I_DATA)
  {
    // process incoming packet
    processPacket(PK(msg));
  }
  else if (msg->getKind() == UDP_I_ERROR)
  {
    EV << "Ignoring UDP error report\n";
    delete msg;
  }
  else
  {
    error("Unrecognized message (%s)%s", msg->getClassName(), msg->getName());
  }

  if (ev.isGUI())
  {
    char buf[40];
    sprintf(buf, "rcvd: %d pks\nsent: %d pks", numReceived, numSent);
    getDisplayString().setTagArg("t", 0, buf);
  }
}
```



Reuse existing models

Models written in:

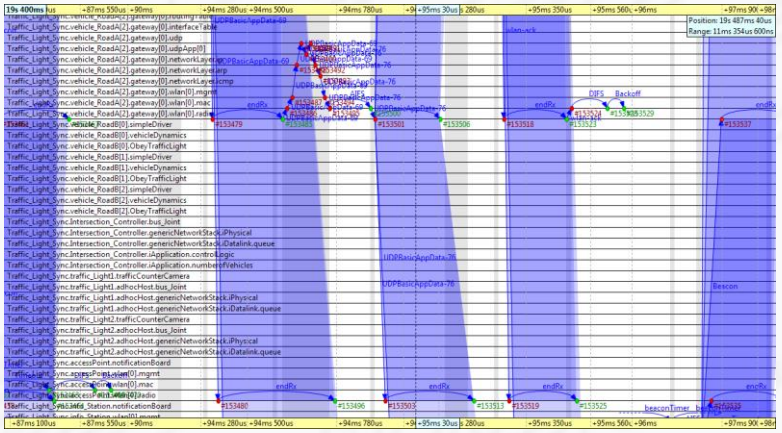
- C/C++
- Simulink
- SystemC
- NI Model Interface Toolkit compatible models

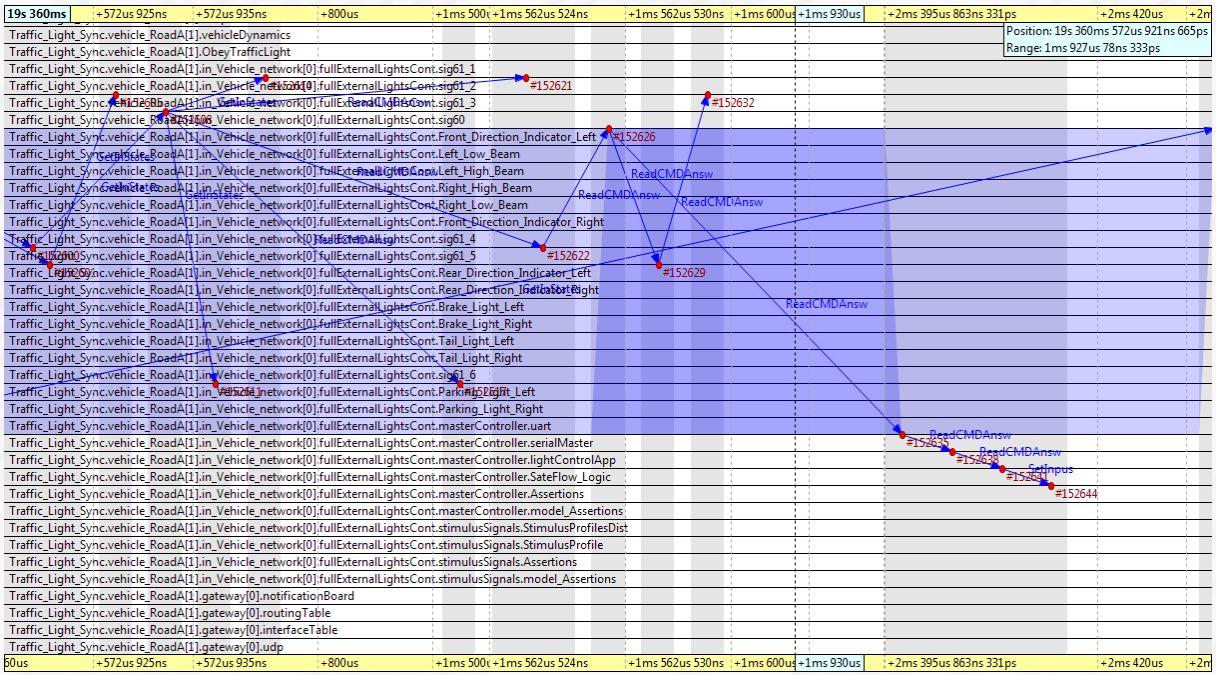
Continuously growing scientific and student community is developing models for OMNEST/OMNeT++

- Over 10,000 academic installations
- About 300 publications each year, growing steadily in number (Google Scholar data)
- Open source models

Network Analysis (network architect)

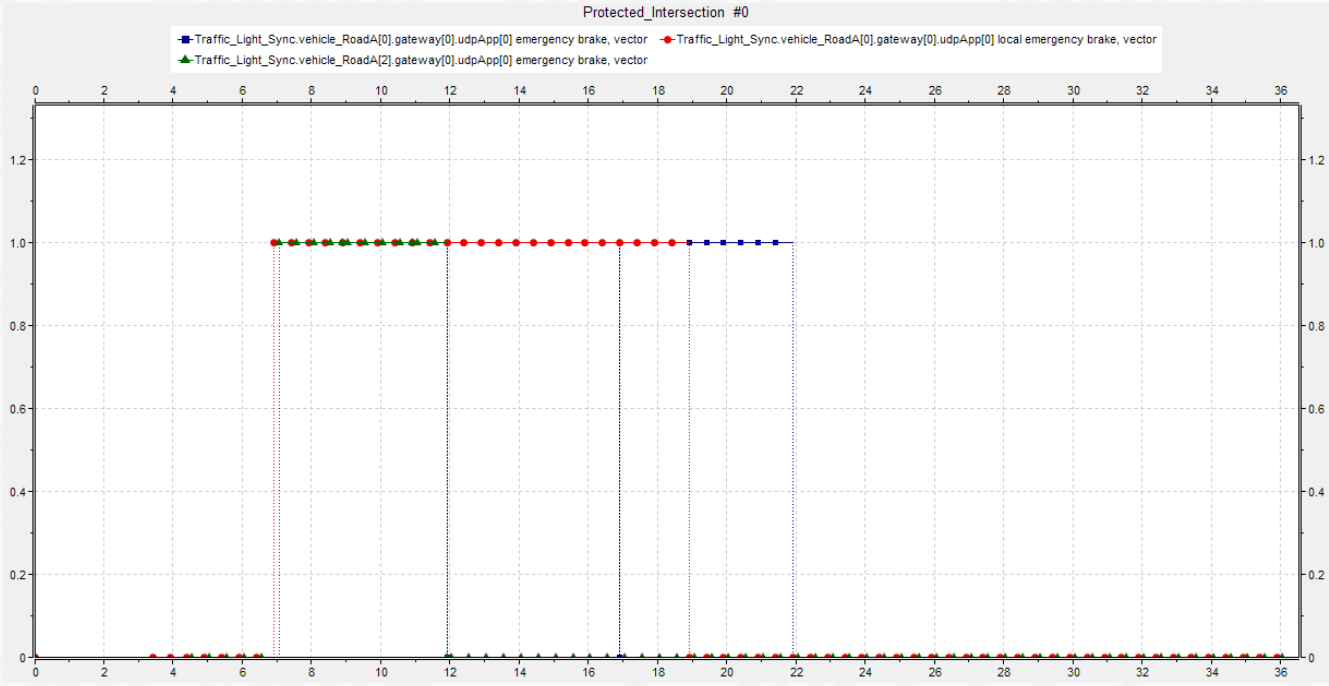
- Jitter, latency analysis
- Signal propagation time check
- Bandwidth analysis
- Event diagrams
- …





Full virtual testing takes advantage of the high performance simulation kernel

- 500k-1000k events/sec
- Complex systems run at 5-10x real time.
- Scalable by using parallel simulation
- Check signal behavior and timing

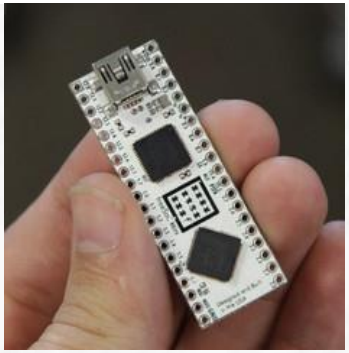


Reuse models in component testing and system integration for **Rest-Bus Simulation**

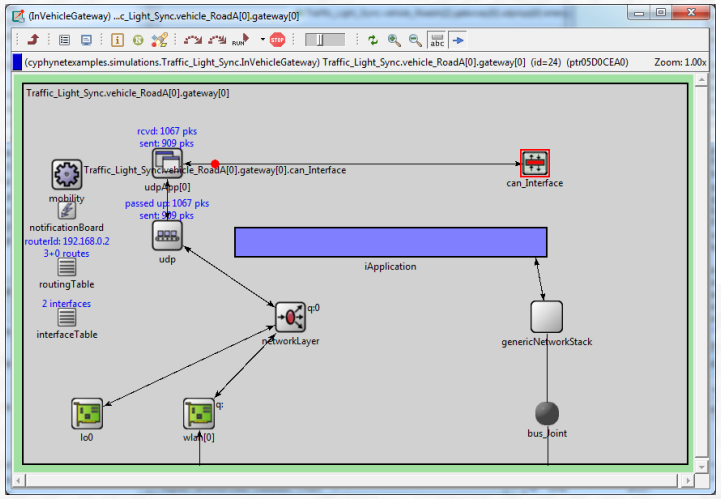
- Use **real hardware** together with **prototype hardware** or connect an **existing production component** to the rest of the system

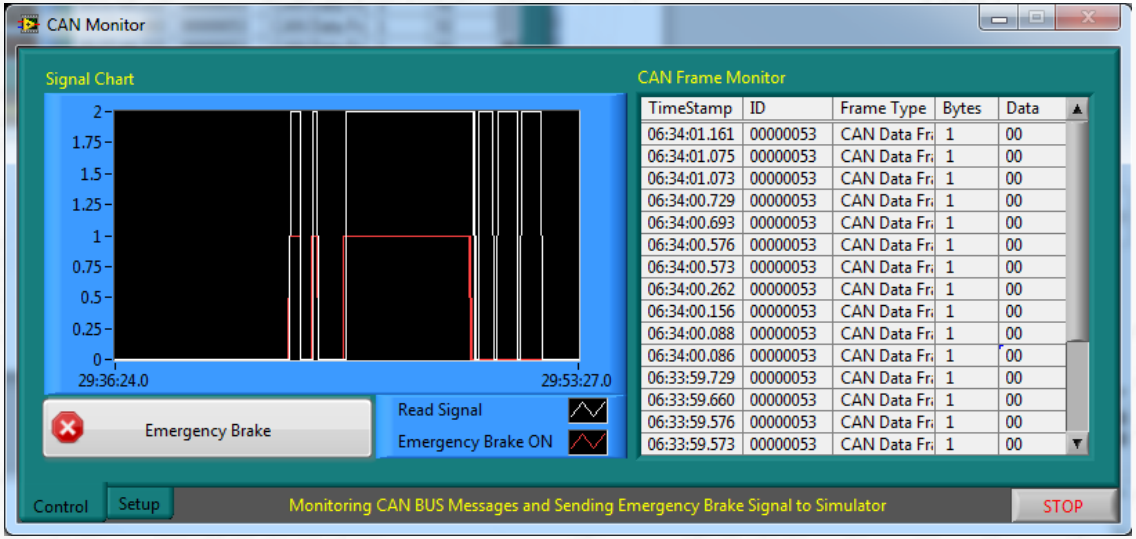
- External Ethernet adapter
- External WiFi
- NI-XNET CAN, LIN, FlexRay

Use COTS hardware to quickly prototype the new component



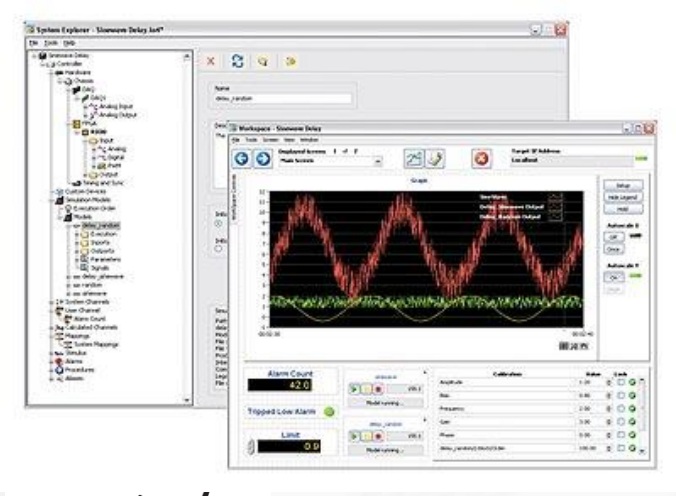
Capture and visualize bus traffic with your favorite tools

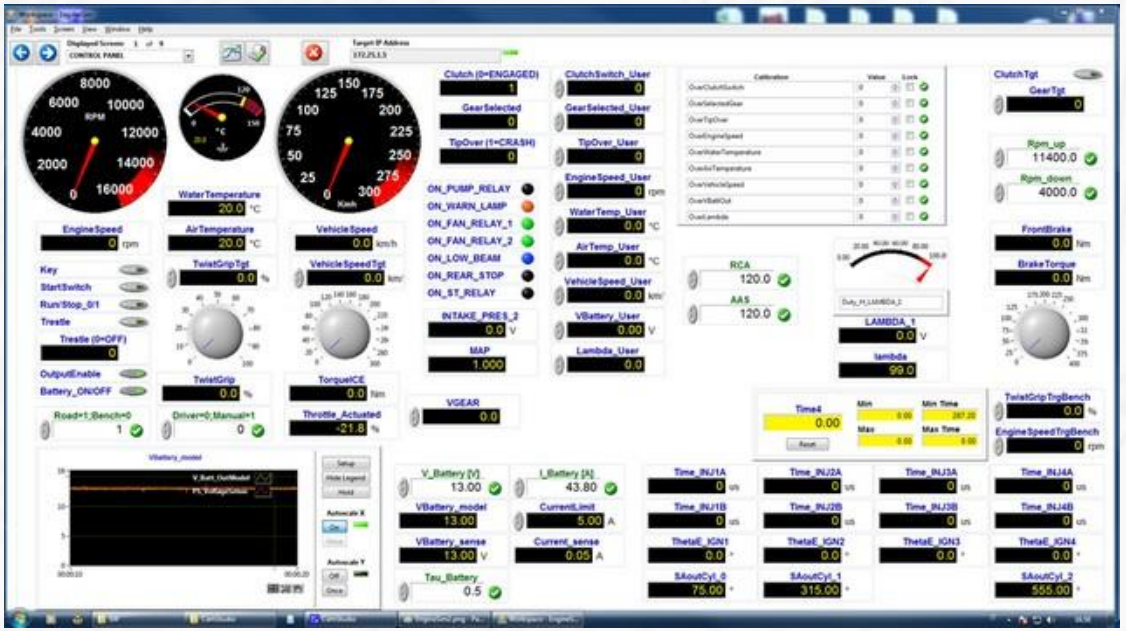




Reuse models and simulation components in hardware validation testing

- Send the components to a Real-Time hardware testing environment such as NI-VeriStand
- Network configuration and connections can be exported to an NI-VeriStand System Definition file





Thanks you for your attention!

Come to the table, ask us questions, and visit www.omnest.com !

From Ideas to Solutions
With Confidence