# OMNeT++ - Tutorial 1

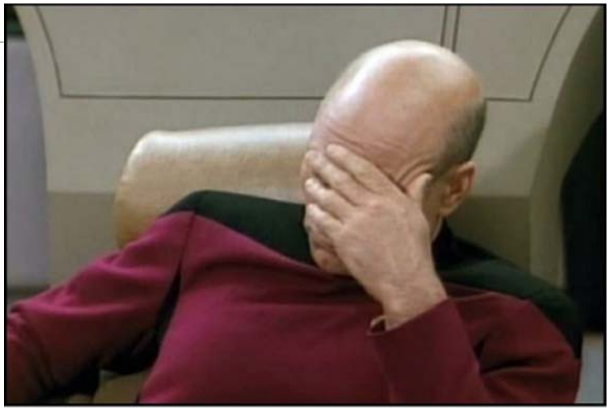
Brian Ricks
Wireless Network Security
Spring 2014

OMNeT++

Carnegie Mellon University

# What is OMNeT++?

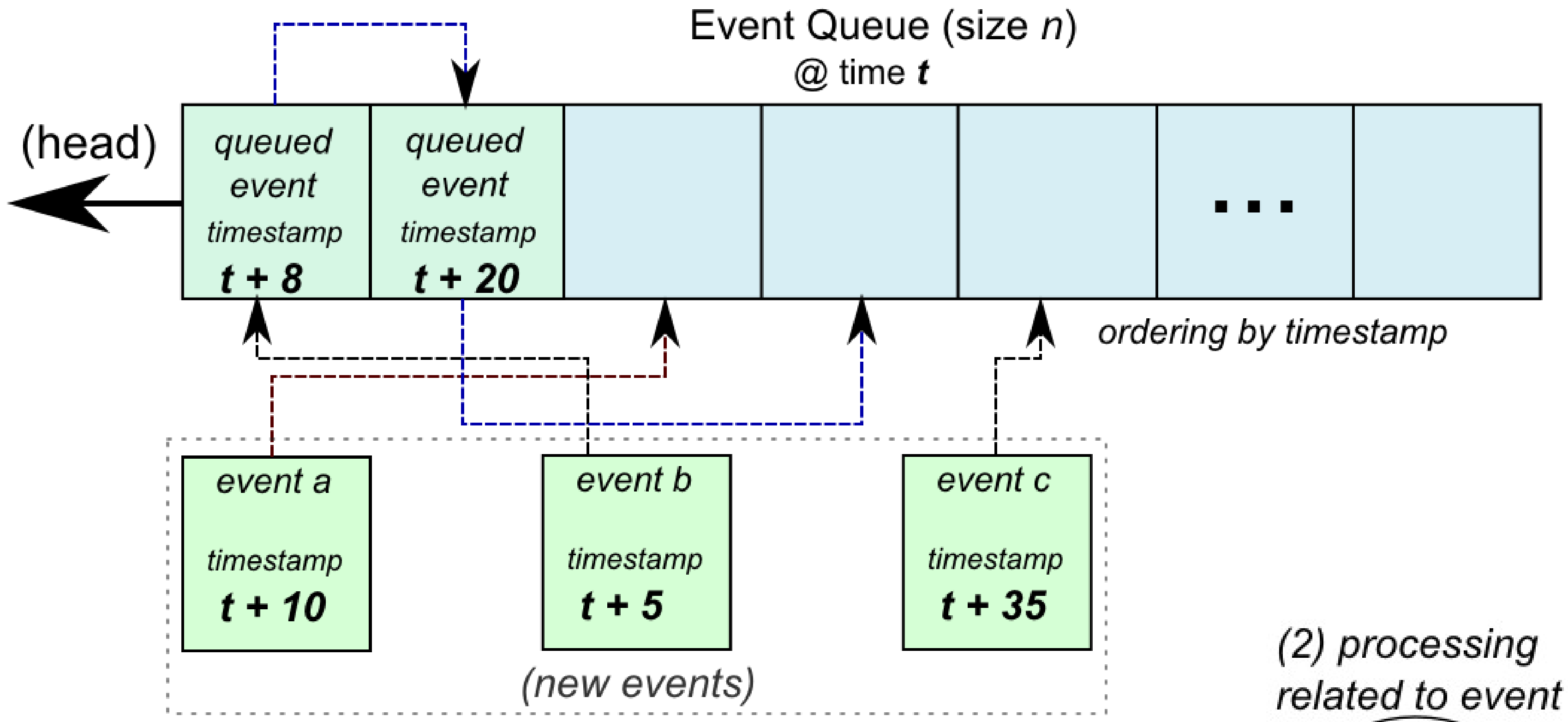
- It's a network simulator!!
  - No, but it provides a base for 'network' simulations
- A generalized framework for building 'network' simulations
  - Communication networks
  - Queuing networks
  - Digital logic networks
  - ' …. ' networks

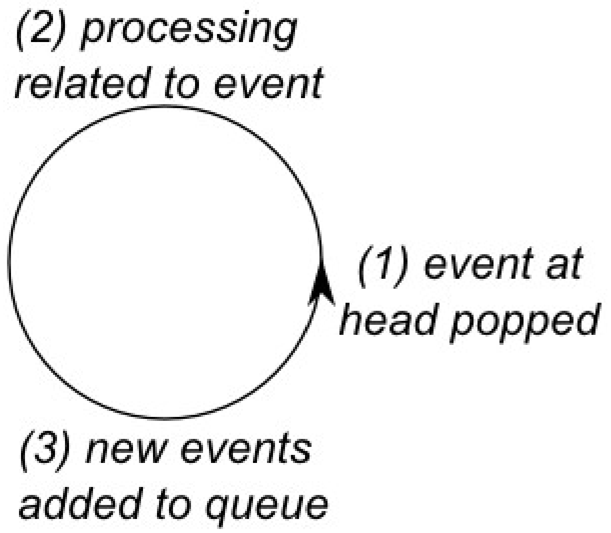


# What is OMNeT++?

- Two components
  - Simulation kernel
    - Event-driven
  - Utility classes
    - Implementations of common functionality for network simulations
      - Math functions
      - Statistics
      - Physical network characteristics helper classes
      - Etc ...

# Simulation Kernel


Event Queue (size *n*)
@ time ***t***
(head)
queued event timestamp ***t* + 8**
queued event timestamp ***t* + 20**
. . .
ordering by timestamp
event a timestamp ***t* + 10**
event b timestamp ***t* + 5**
event c timestamp ***t* + 35**
(new events)


(2) processing related to event
(1) event at head popped
(3) new events added to queue

- Simulation kernel terminates when:
  - No more events in event queue
  - Termination condition reached
  - User terminates

# Installing OMNeT++

- What, you havn't installed it yet??
  - Refer to the Captain Picard pic below
- Windows install
  - Self contained, simply follow directions:
    - Download, unzip somewhere (no spaces in path)
    - Double-click mingwenv.cmd
      - If Windows 8 doesn't like this, click 'more info', then 'run anyway'
    - Type: ./configure
    - Type: make
  - doc/InstallGuide.pdf

# Installing OMNeT++

- What about Linux?
  - InstallGuide.pdf contains instructions
  - Be sure to have all dependencies
    - Not only what OMNeT++ asks for in the guide, but also anything else you may need
      - The configure script will tell if you are missing anything
    - What if it doesn't make?
      - If compilation errors come up, try an older version of GCC
- What about MacOS?
  - InstallGuide.pdf contains instructions

## What is Inet?

- Communication networks simulation package for OMNeT++
  - Provides models for many wired/wireless networking protocols
  - These models build upon each other to create simulation models of communication nodes, and networks.
  - Gives OMNeT++ communication networks support without us having to write our own protocols.

# Installing INET Framework

- Open OMNeT++ IDE (omnetpp)
  - Goto the workbench; the first time you do this a prompt will ask if you want to install INET
    - Keep the boxes checked and proceed
- What if I skipped this step? (opened the workbench but skipped installing INET)
  - Help → Install Simulation Models
    - Currently only INET is listed here, simply follow the prompts
  - If the empty workspace prompt comes up now, uncheck the INET box

## Installing INET Framework

- If you see a warning icon next to the 'inet' folder in the project explorer, ignore it
  - Refers to various warnings within inet src, usually complaining about variables set but not used
- Run some examples!
  - samples/inet/examples

## Diving Deeper – Simulation Models

- A simulation model consists of *modules*, which are grouped/connected together.
  - Modules that are grouped together are themselves modules
  - Provides a module hierarchy
- In OMNeT++, a simulation model is also called a *network*
  - A network (simulation model) is itself a module

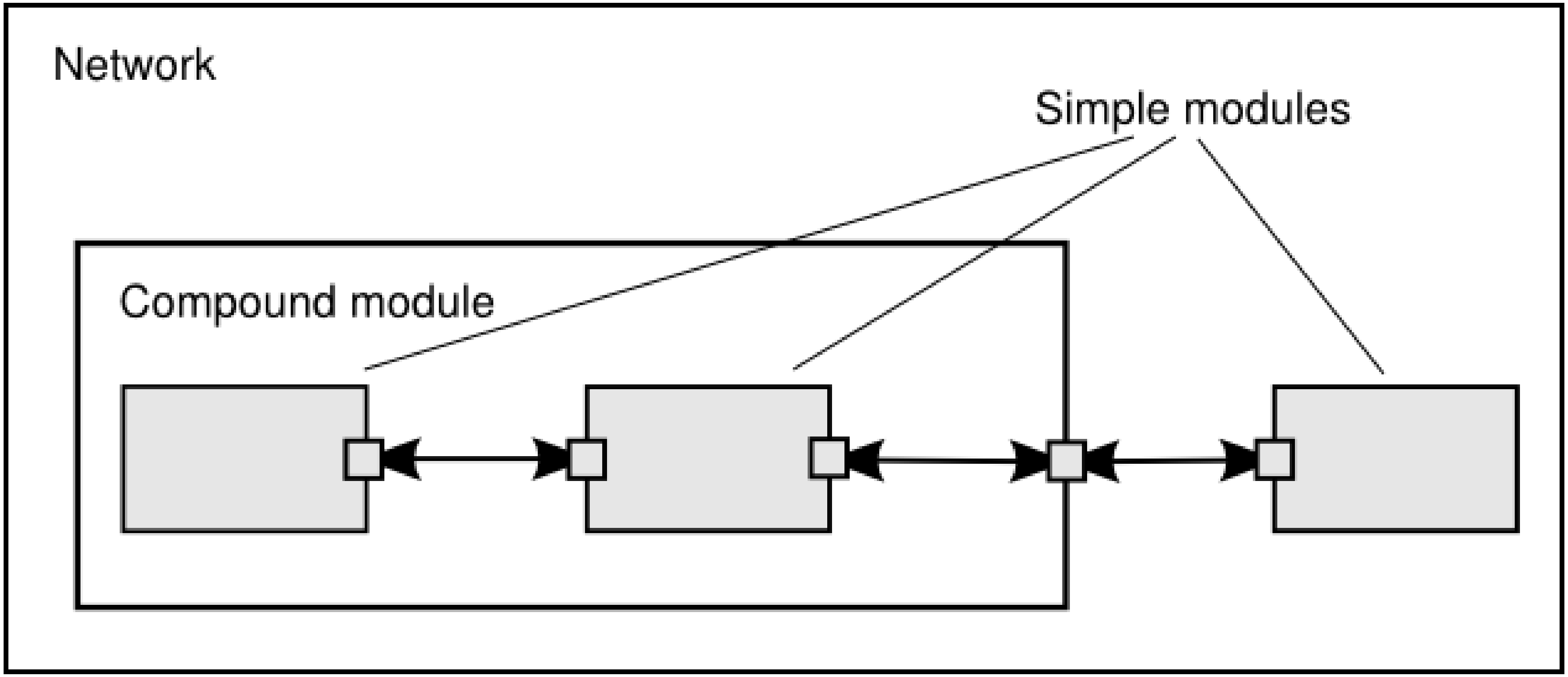
## Simulation Models – Module Types

- What goes into a simulation model?
  - It all starts with *Simple Modules*
    - Base building blocks
    - Declared using the NED language
      - *http://omnetpp.org/doc/omnetpp/manual/usman.html#sec117*
    - Backed by C++ classes which define their behavior
    - Defines parameters to pass to (C++) implementation
  - Simple modules group together to form *Compound Modules*
    - Declared using the NED language
    - Defines parameters to pass to simple modules

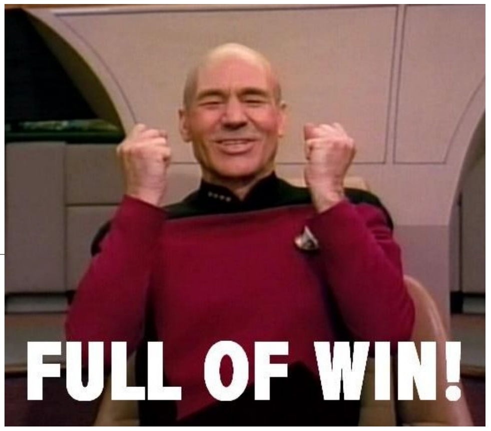
## Simulation Models - Gates

- *Gates* allow for message passing
  - *Messages* pass between gates using *connections*
    - Two gates can be directly linked via connection
      - Think wired communication network
    - Connections can also be used to directly pass a message to an unlinked gate
      - Think wireless communication network
  - Connections can be defined and reused
    - Called *channels*

# Simulation Models



- http://www.omnetpp.org/doc/omnetpp/manual/usman.html#sec101
- Not having to draw my own illustration for this slide



## Simulation Models – Structure

- Compound modules describe the structure of a simulation model.
  - A network is itself a compound module
  - Module at top of hierarchy = *system module*
  - Modules below = *submodules*
    - Submodules = compound or simple modules
  - Modules at bottom of hierarchy = simple modules
- Hierarchy depth is unlimited
  - Reflect logical structure of system being modeled

# Simulation Models - Messages

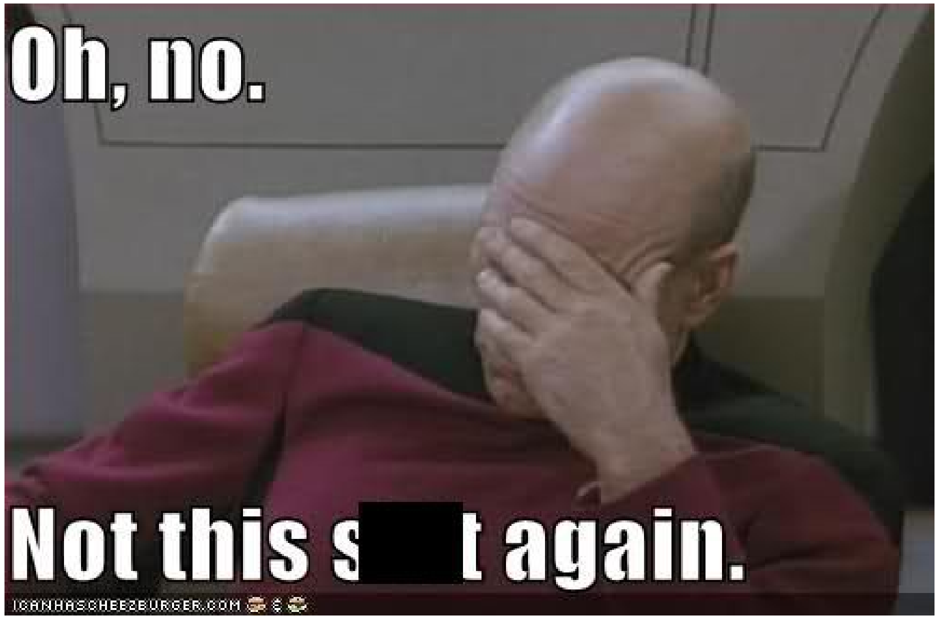
- Defined using message definitions
  - Translated to C++ classes

## An Example!

- Time to see how this done!
- OMNeT++ Tic-Toc example
  - This does not require Inet

## Statistics Collection

- Now for the non-fun part
  - But a very important part if you want a grade for your assignments.



## Statistics Collection

- Output vectors
  - Time-series data
  - Stuff that gets recorded during a simulation
- Output scalars
  - Stuff that gets recorded at the end of a simulation
  - Mean of something, std dev of something, etc...

# Statistics Collection

- Declaring Statistics
  - http://omnetpp.org/doc/omnetpp/manual/usman.html#sec195
  - In NED:
    - @statistic[stat_name](properties)
      - stat_name = variable emitted from the C++ class
      - properties = what to record, and in which form (scalar, vector)
    - Example:
      - @statistic[received_pkt](record=sum,vector?)
        - received_pkt is a variable emitted each time a packet is received
        - We are recording to a scalar the total packets received
        - We are recording to a vector each time a packet is received
          - Note the '?' - this means its optional

# Statistics Collection

- Emitting variables (signals)
  - http://omnetpp.org/doc/omnetpp/manual/usman.html#sec193
  - Register the signal by name
    - registerSignal(“stat_name”)
      - stat_name must match that given in the NED declaration
      - Function returns an id for the signal
  - Emit the signal when appropriate
    - emit(signal_id, value)
      - signal_id = id of signal (mapped to stat_name)
        - *simsignal_t signal_id = registerSignal("stat_name")*

# THE END (for now)

WHO THE F[redacted]K
HAS BEEN CUTTING EVERYONE'S GRASS DURING THE ZOMBIE APOCALYPSE?