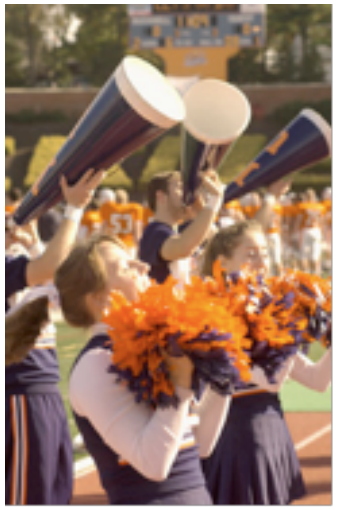
Bucknell UNIVERSITY

The Evolution of a Computer Aided Simulation System

L. Felipe Perrone <*perrone@bucknell.edu*>
Department of Computer Science
Bucknell University




Undergraduate Collaborators

- Christopher Kenna (BSCS '10)
- Bryan C. Ward (BCSE '11)
- Andrew W. Hallagan (BCSE '11)
- Tiago Rodrigues (UFPI, Brazil)
- Christopher Main (BSCS '13)
- Vinícius Felizardo (UNICAMP, Brazil)
- Shelby Kilmer (BSCS '15)

Frameworks for ns-3 Collaborators

- Tom Henderson, Boeing/University of Washington
- Mitch Watrous, University of Washington
- George Riley, Georgia Tech

Related Work

- Akaroa2 (K. Pawlikowski et al.)
- James II (R. Ewald et al.)
- STARS (E. Millman et al.)
- ANSWER (M. Andreozzi and G. Stea)

Scripts for Organizing Simulations (SOS)

Authors: Tim Griffin, Srdjan Petrovic, Anna Poplawski, and BJ Premore

URL: http://ssfnet.org/sos/

"**This set of scripts was put together to ease the process of running a large number of experiments with varying parameters, and manage the resulting data.** The work required to do such things manually can be quite large, especially taking into account that the number of experiments that need to be run in order to obtain representative data is often big. A script which plots the data using gnuplot is also included.

The SOS package was originally put together to run experiments with **SSFNet**. Other researchers heard about it and wanted to use it to run experiments and collect data, so we made it more generic to work with any set of experiments performed on the computer (without making it any less useful for the users of SSFNet)."

Scripts for Organizing 'Spiriments (SOS)

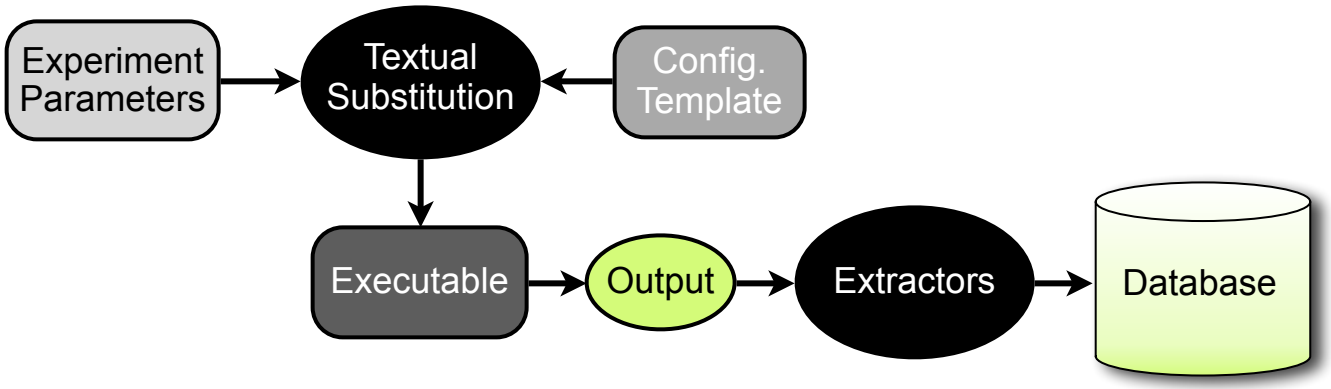
Authors: Tim Griffin, Srdjan Petrovic, Anna Poplawski, BJ Premore

URL: http://ssfnet.org/sos/

"This set of scripts was put together to ease the process of running a large number of experiments with varying parameters, and manage the resulting data. The work required to do such things manually can be quite large, especially taking into account that the number of experiments that need to be run in order to obtain representative data is often big. A script which plots the data using gnuplot is also included.

The SOS package was originally put together to run experiments with SSFNet. **Other researchers heard about it and wanted to use it to run experiments and collect data, so we made it more generic to work with any set of experiments performed on the computer** (without making it any less useful for the users of SSFNet)."

The SOS Workflow



- Database contains complete experimental set up.
- Database schema must be customized to experiment.
- Script carries out execution (might have to customize).
- Experimenter writes extractors (have to customize).
- Scripts to make plots from dB data (have to customize).

Direct Experience with SOS

Modeling and Simulation Best Practices for Wireless Ad Hoc Networks

L. Felipe Perrone, Yougu Yuan, and David M. Nicol
Proceedings of the 2003 Winter Simulation Conference, pp. 685-693.

Empirical analysis showed that:

- Details on the composition of the protocol stack have significant importance.
- Transients in random waypoint mobility and network protocol models create a pronounced impact on run length.
- The choice of interference models matters a lot, specially when one scales up the number of nodes.

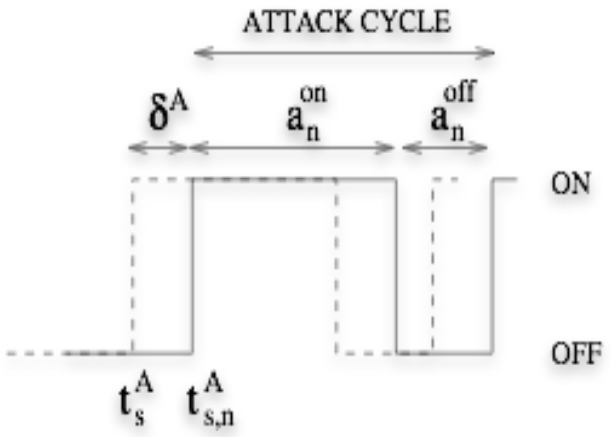
Lessons Learned with SOS

- The database was crucial: having the experimental setup paired with the output of every experiment is priceless.
- Customizing SOS was somewhat complicated:
  - Quite a bit of work to make extractors.
  - Mining the results database was not exactly trivial.
  - Almost every plot required customization of script.
- In the move from one university to another, the experimental database was corrupted and all data was lost.

New Study, Same Needs

A Study of On-Off Attack Models for Wireless Ad Hoc Networks

L. Felipe Perrone and Samuel C. Nelson. First IEEE International Workshop on Operator-Assisted (Wireless Mesh) Community Networks (OpComm 2006). Berlin, Germany.

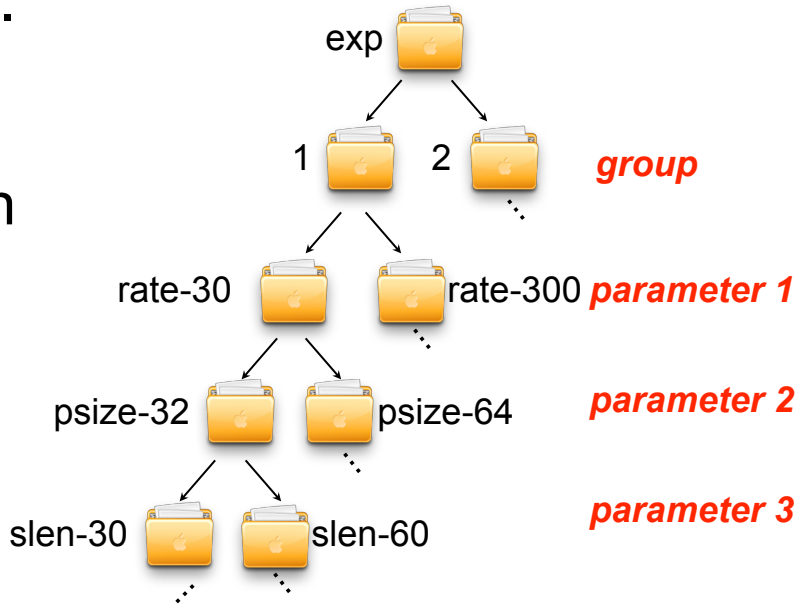


A simple attack model

The Homemade Approach

Created ad hoc Ruby scripts to:

- Generate experimental design points.
- Build configuration file for each experiment from a template and design points.
- Launch experiments on multiple machines.
- Extract data from simulator output and build directory structure.

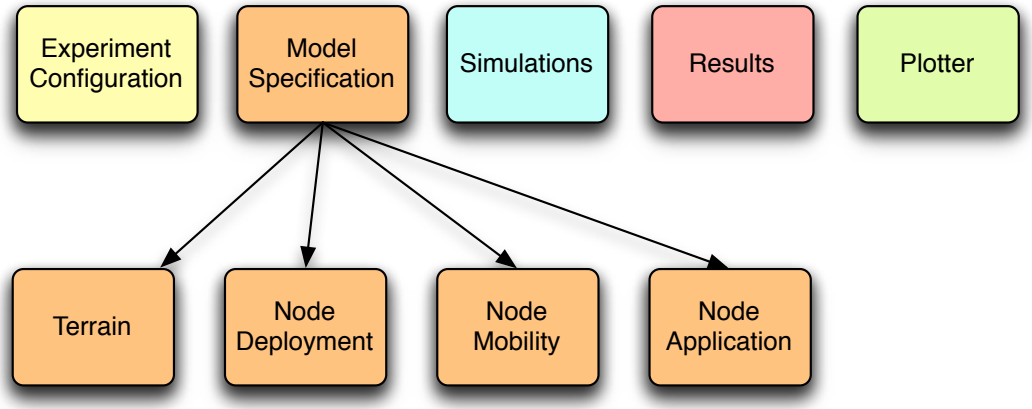


- Traverse directory structure and build custom plots.

A Step in the Right Direction

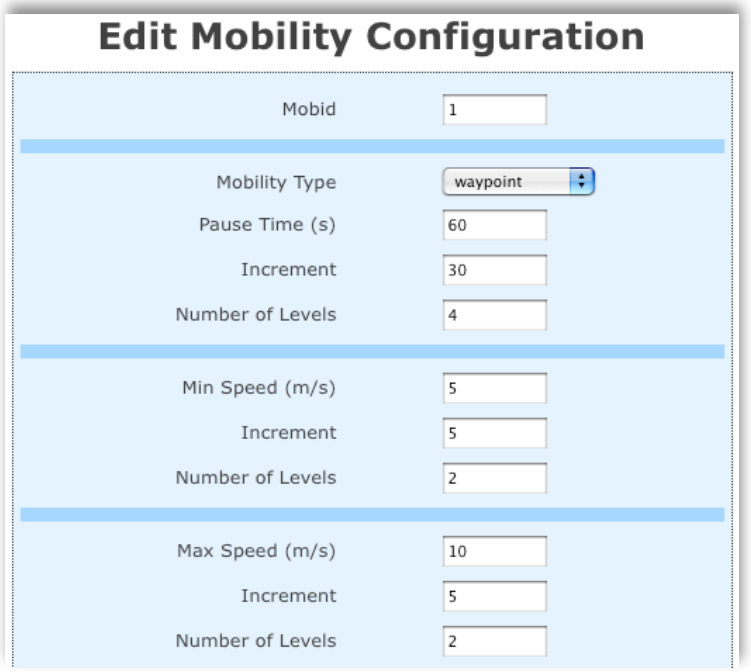
Enhancing the Credibility of Wireless Network Simulations with Experiment Automation

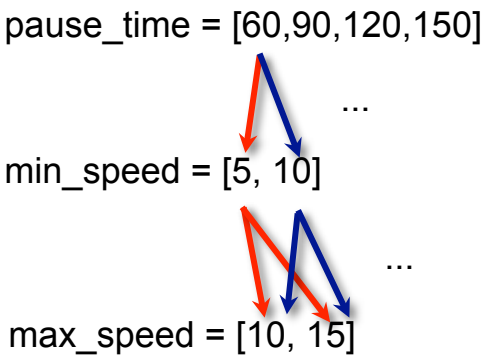
L. Felipe Perrone, Christopher J. Kenna, and Bryan C. Ward
IEEE International Workshop on Selected Topics in Mobile and Wireless Computing 2008.



SWAN Tools

A web based interface for simulating wireless networks with SWAN.





The interface constrained users to "do the right thing."

Our Guiding Light

MANET Simulation Studies: The Incredibles

Stuart Kurkowski, Tracy Camp, and Michael Colagrosso
SIGMOBILE Mob. Comput. Commun. Rev., vol. 9, no. 4, pp. 50–61, 2005.

"For our study we focused on the following four areas of credibility in research.

1. ***Repeatable****: A fellow researcher should be able to repeat the results for his/her own satisfaction, future reviews, or further development.*

2. ***Unbiased****: The results must not be specific to the scenario used in the experiment.*

3. ***Rigorous****: The scenarios and conditions used to test the experiment must truly exercise the aspect of MANETs being studied..*

4. ***Statistically sound****: The execution and analysis of the experiment must be based on mathematical principles."*

A Few Credibility Issues

Experiments published are not always **reproducible**.

- We often don't know the version of the simulator used in the experiments (magic numbers inside code).
- Sometimes we are not told the precise composition of the simulation model.
- We hardly ever know all the attributes for all the model components.
- Flawed output data analysis.

Automation for Simulation Studies

Not all steps of a study can be automated, but *several* definitely can!

- Problem formulation
- Set objectives, plan
- Model conceptualization
- Data collection
- Model translation
- Verified?
- Validated?
- Experimental design
- Production runs
- Output data analysis
- More runs?
- Documentation and reporting
- Implementation

Requirement 1: Self-documenting system

The system stores/generates/returns:

- Simulation source-code
- Model attribute settings
- Experiment parameters
- Raw output data
- Processed output data
- Presentation quality plots

This makes reproducibility, documentation, and reporting fool-proof.

Requirement 2: Execution Control

- Execution guided by high level experiment description
- Exploit available systems via MRIP
- Collect more samples by running more simulations
- Generates random seeds for each run

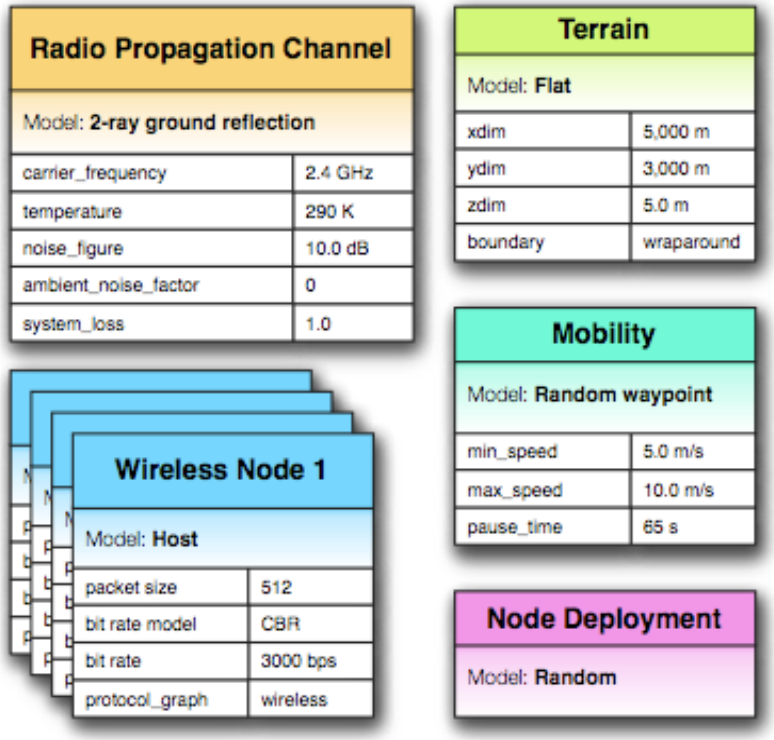
This makes execution easier, safer, and possibly faster.

Requirement 3: Automatic Output Processing

- Results stored in local file system and also communicated to a server
- Samples processed by verified statistical package

This guarantees that output is safe and correctly processed.

Some SWAN Tools Details



Internally, each component is described in a configuration language (DML). Full separation of model configuration and model source code.

How does the user know what components will play nicely together?

There are dependencies and incompatibilities that must be taken into account.

Lessons Learned from SWAN Tools

SWAN Tools was a good first crack at the larger problem.

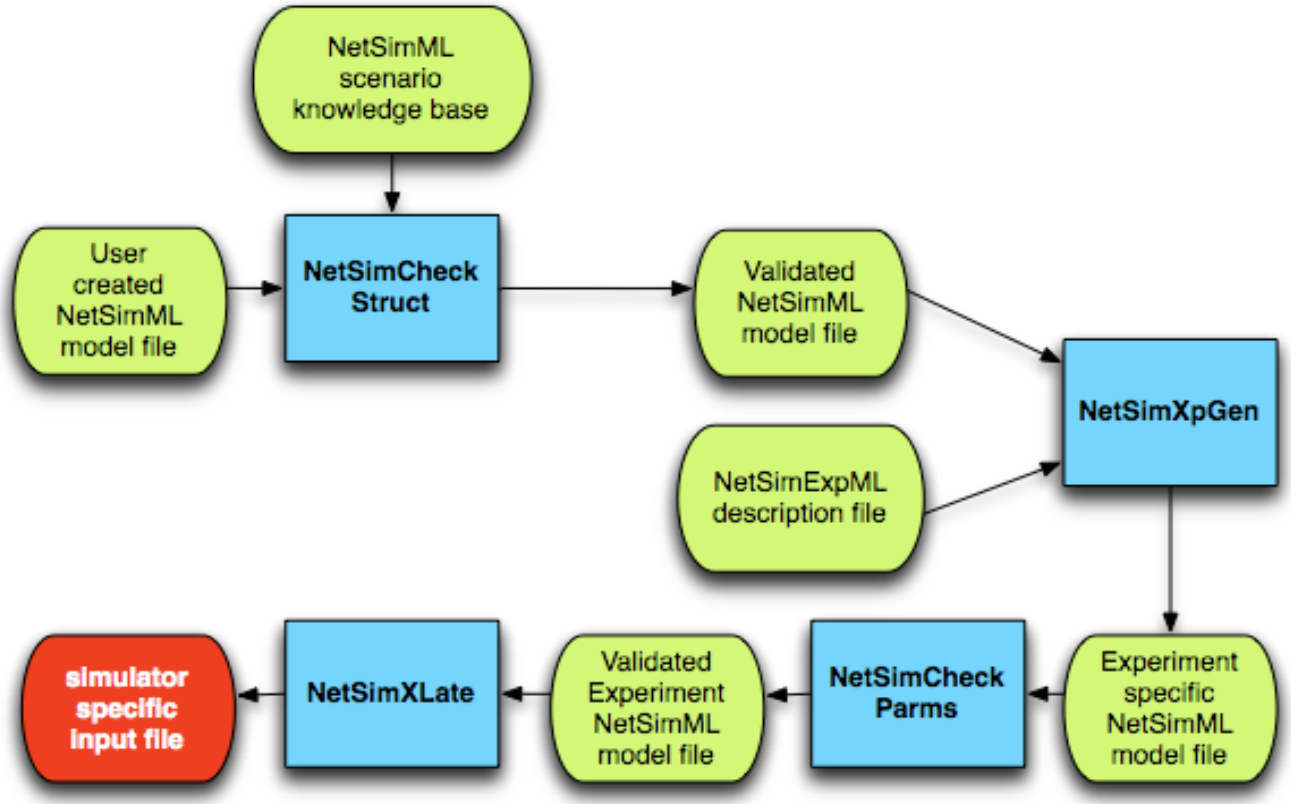
We wished for a more powerful, flexible system that:

- Works with various network simulators (not just with SWAN).
- Allows for more configurability (SWAN Tools restricts the parameters in experiment design space) and controllability.
- Allows for the incorporation of advances in scenario development (model construction).

Design Ideas

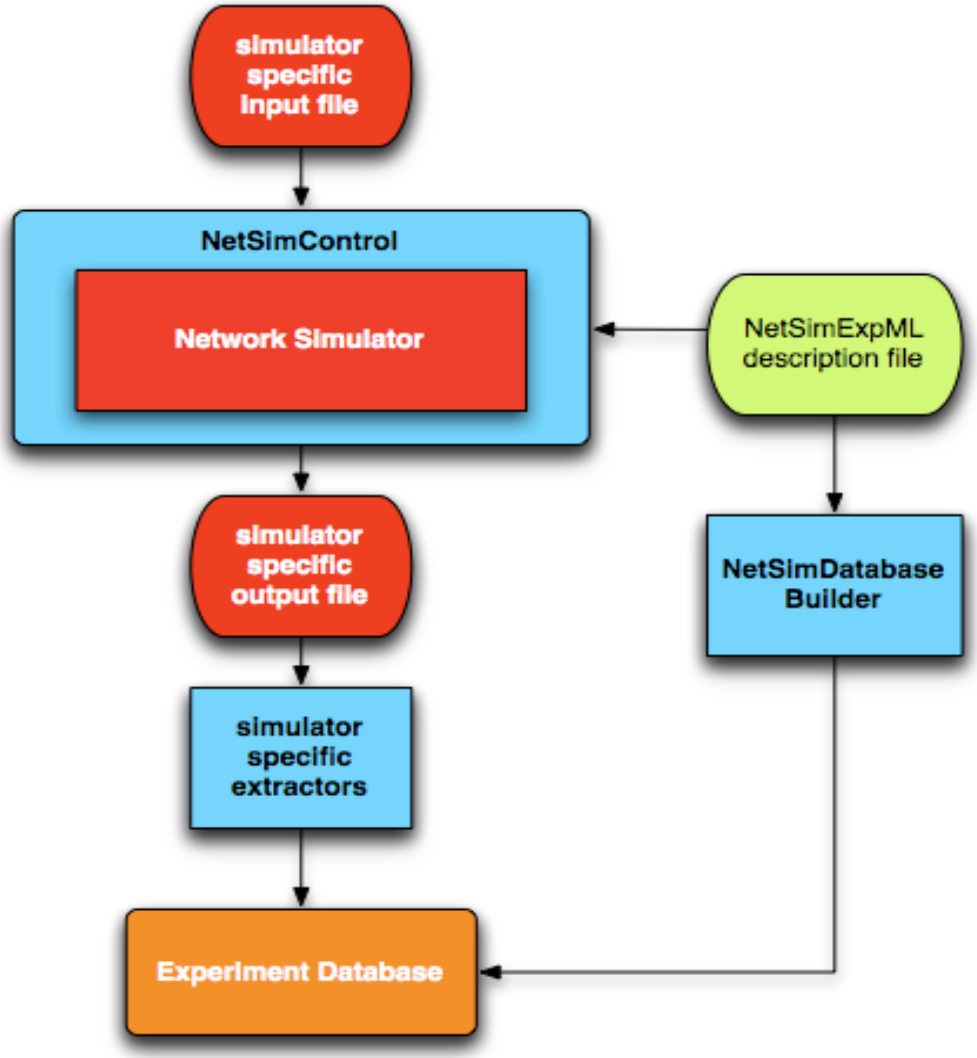
The Configuration Pipeline

- Check if components of a model fit nicely together.
- Define a simulator independent modeling language that can be translated to simulator specific configuration files.



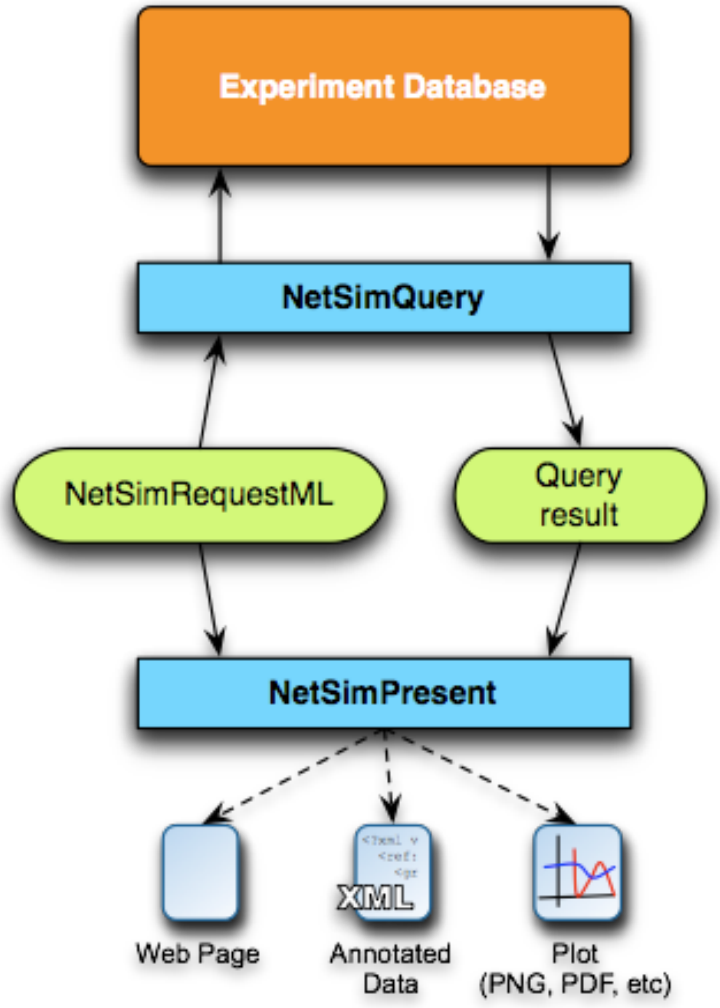
The Control Pipeline

- Determine how long a simulation needs to be run.
- Determine if/when steady state has been reached for data deletion.
- Dispatch simulation runs to various machines.
- “Speak the language” of different underlying simulators.



The Output Pipeline

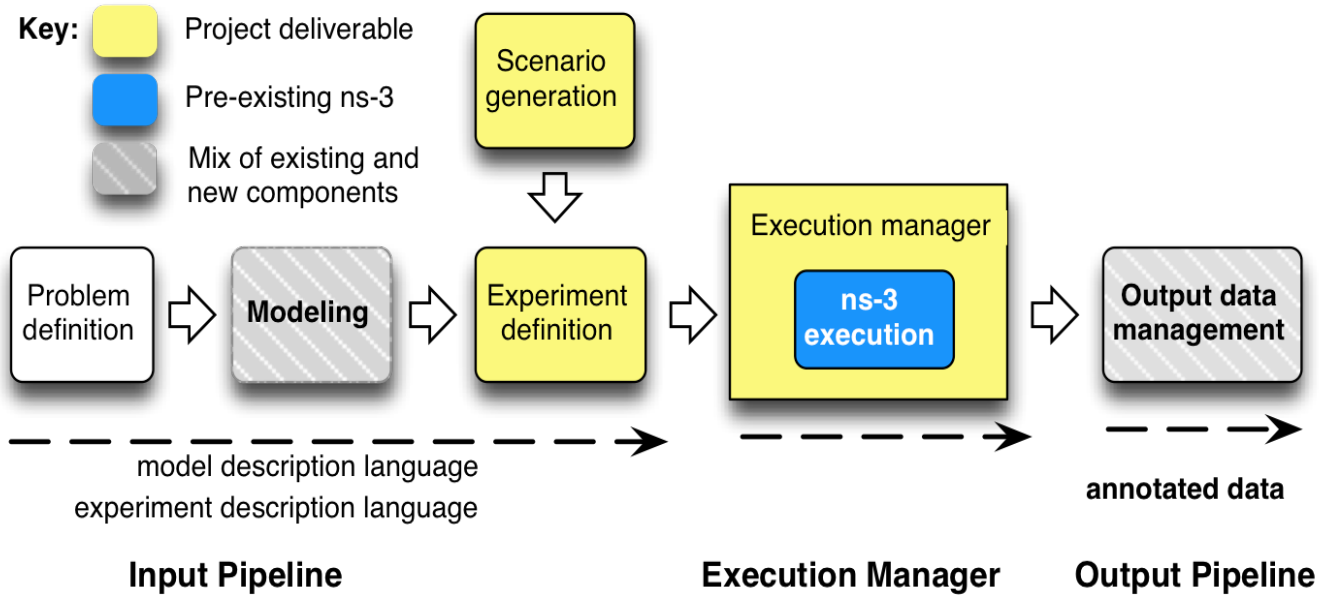
- Process data previously stored in experiment dB.
- Use correct statistical methods.
- Detect steady-state and termination.
- Generate custom plots in various formats.
- Allow one to retrieve the experimental setup and associated results.



Frameworks for ns-3

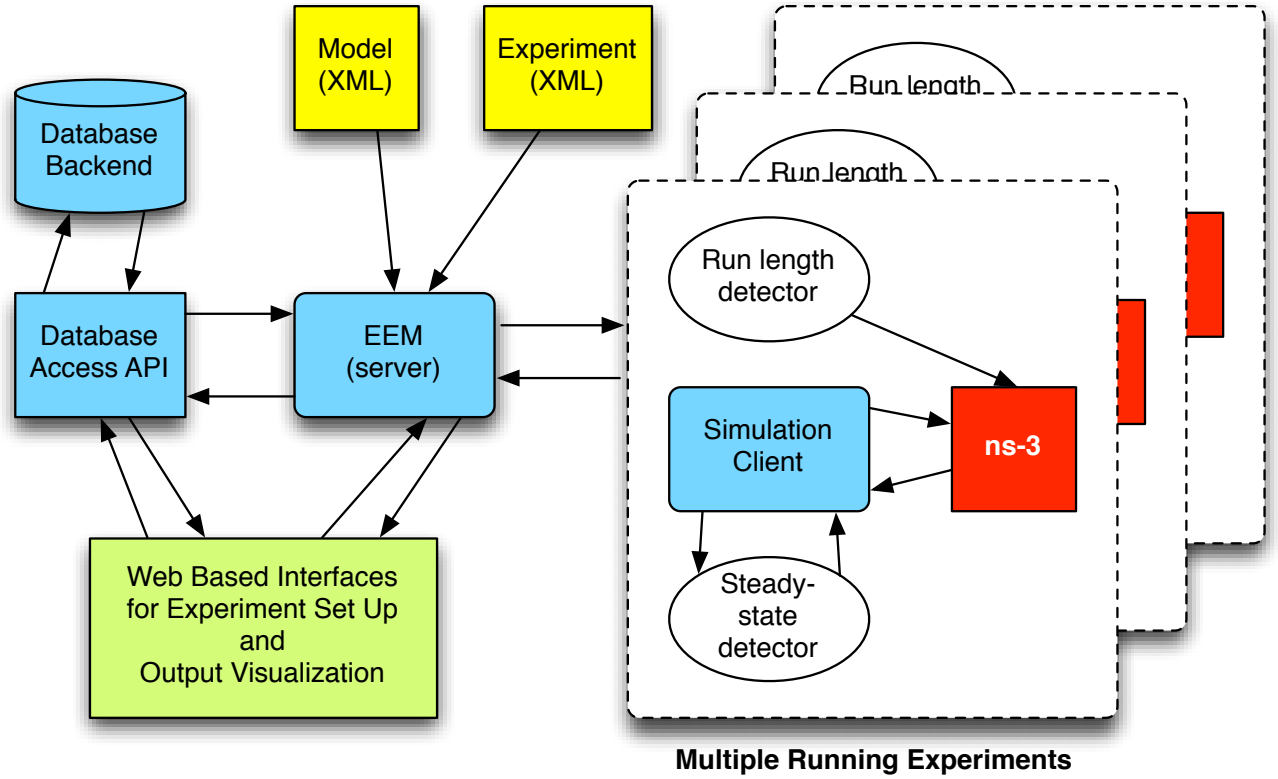
NSF CISE Community Research Infrastructure

- University of Washington (Tom Henderson), Georgia Tech (George Riley), Bucknell Univ. (Felipe Perrone)
- Project timeline: 2010-14



SAFE: Simulation Automation Frameworks for Experiments

L. Felipe Perrone, Christopher S. Main, and Bryan C. Ward
Proceedings of the 2012 Winter Simulation Conference

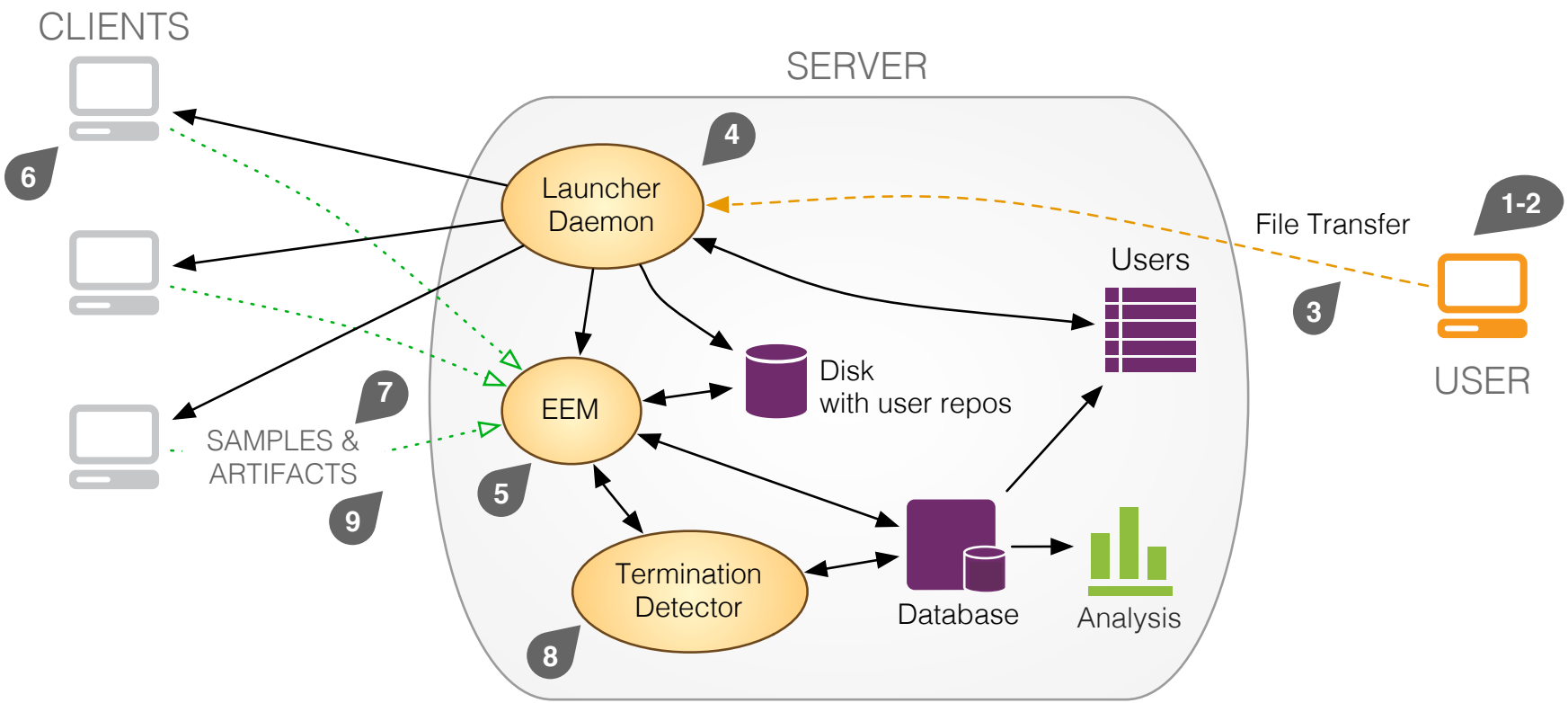


User Stories

- **Power user:** develops models, write ns-3 scripts, uses SAFE to launch experiments, process and safekeep results, generate presentation quality graphs. Mostly via command-line tools.

- **Novice user:** uses SAFE to configure experiments with pre-canned ns-3 scripts, process and safekeep results, generate presentation quality graphs. Mostly via web-browser interface.

Workflow (1-2)



1) User writes ns-3 script for the experiment; stores within local ns-3 installation.

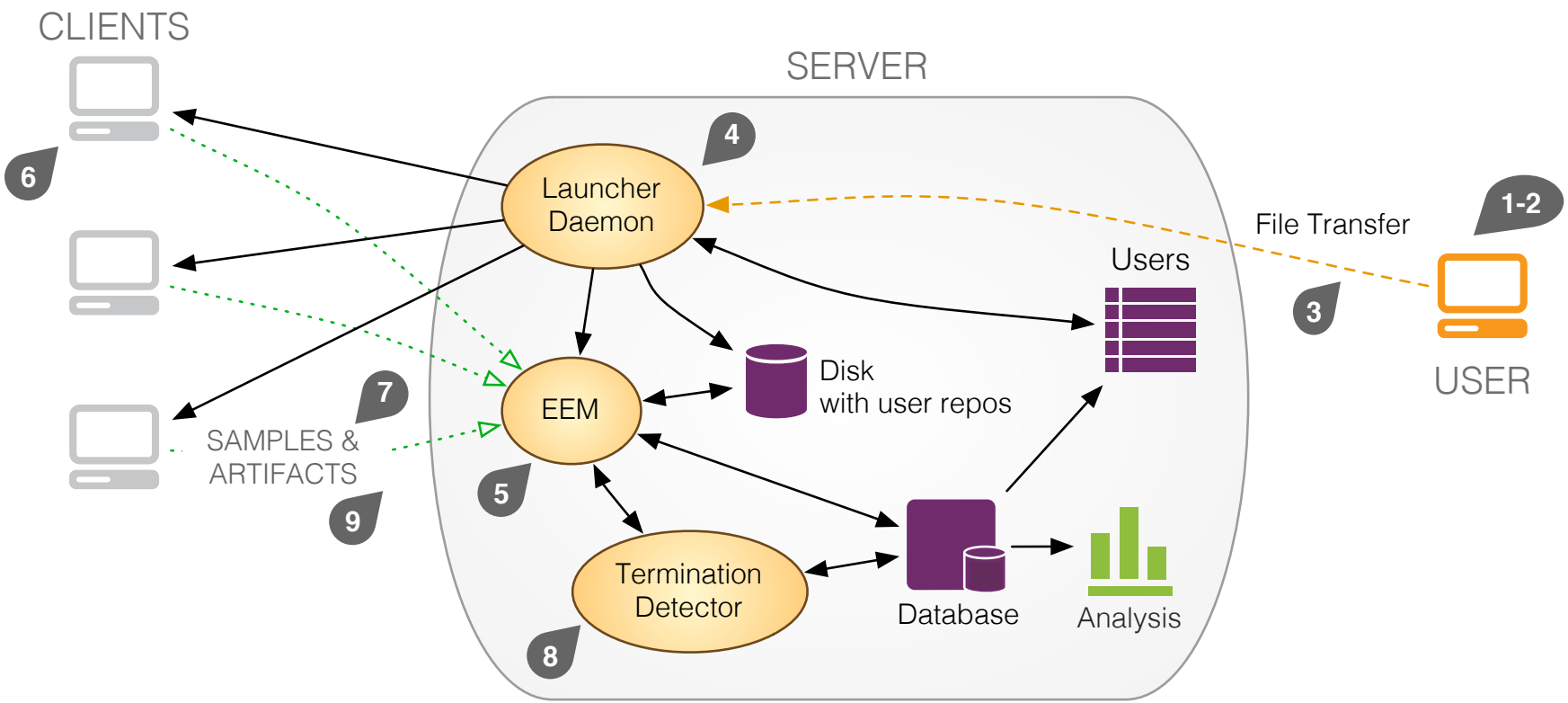
2) User (or system) generates experiment configuration file in NEDL.

Language Support

- ns-3 Experiment Description Language (NEDL): XML-based specification of the design of experiment space.

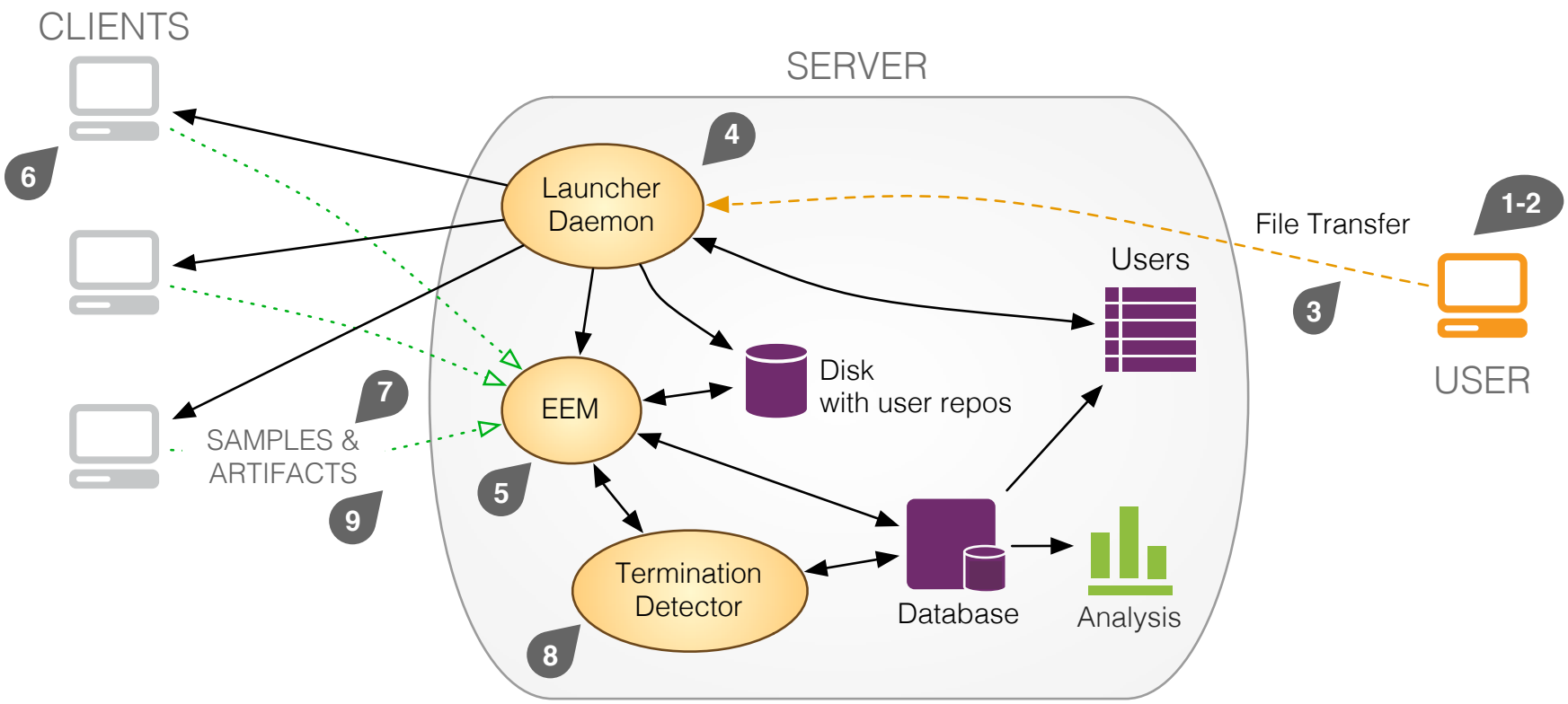
- ns-3 Script Templating Language (NSTL): XML-based description of a simulation model that can be modified automatically via substitution of code blocks.

Workflow (3)



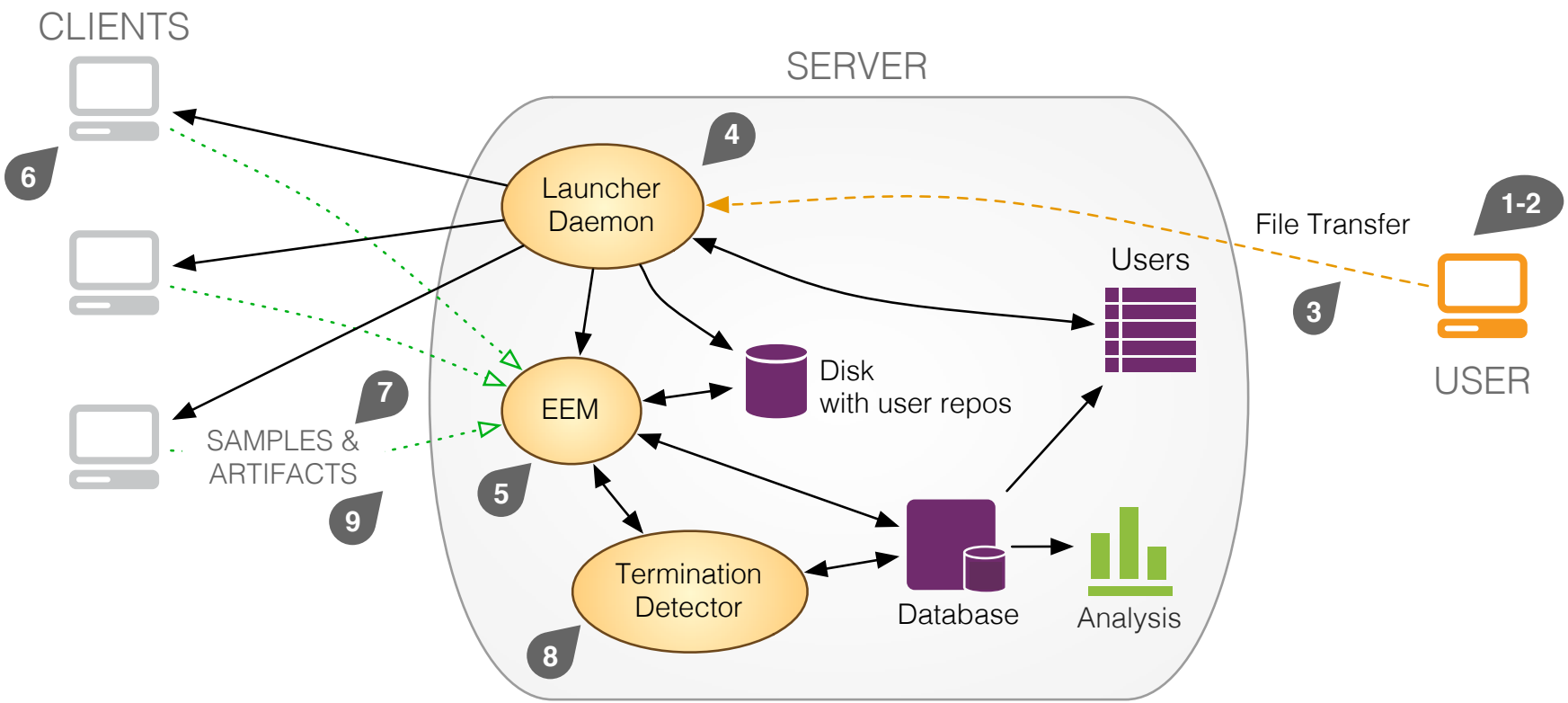
3) ns-3 installation archived; credentials verified with server; bundle transferred to user compartment in server under unique experiment id.

Workflow (4)



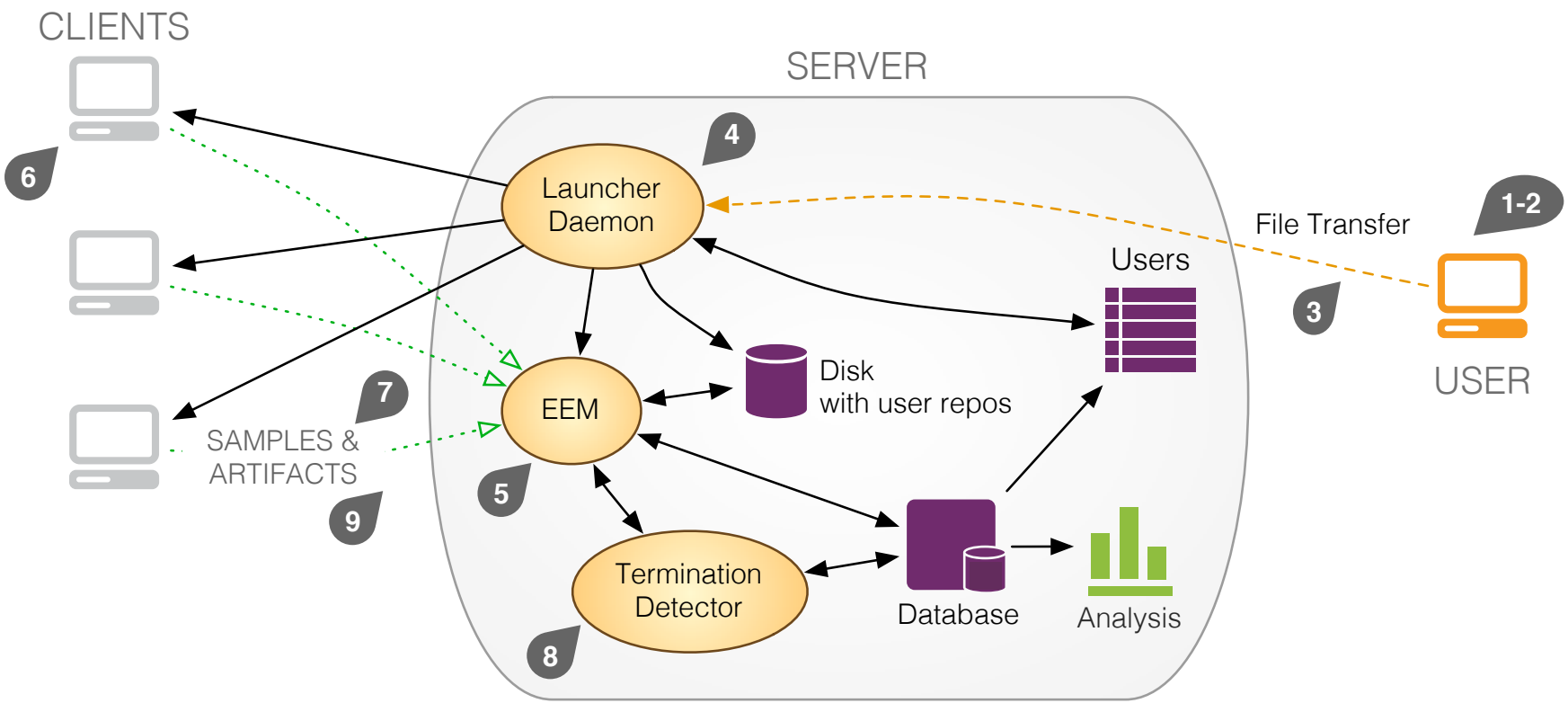
4) Server deploys bundle across worker machines (clients); builds ns-3 locally in each.

Workflow (5)



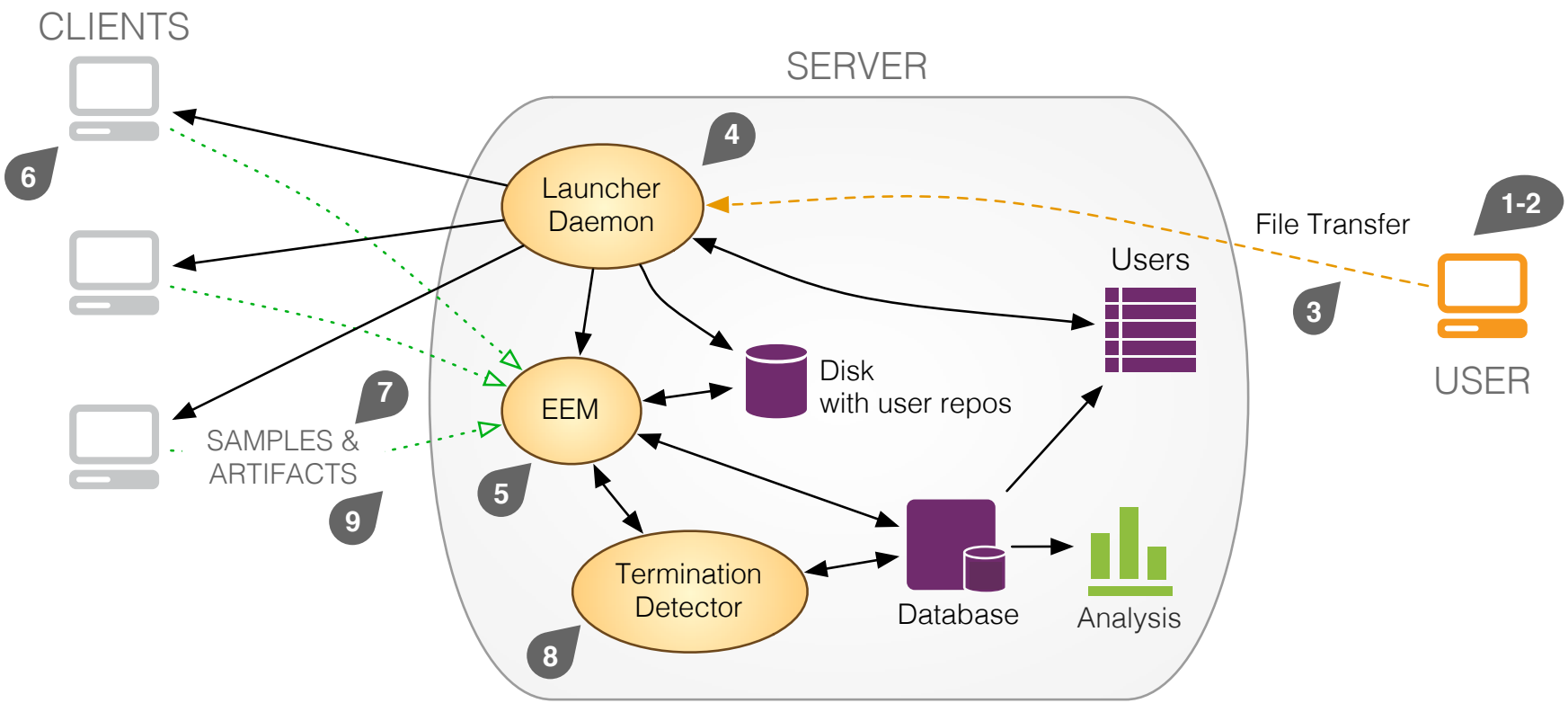
5) Launcher daemon starts: EEM with NEDL file; termination detector process. EEM computes design points and waits for requests.

Workflow (6)



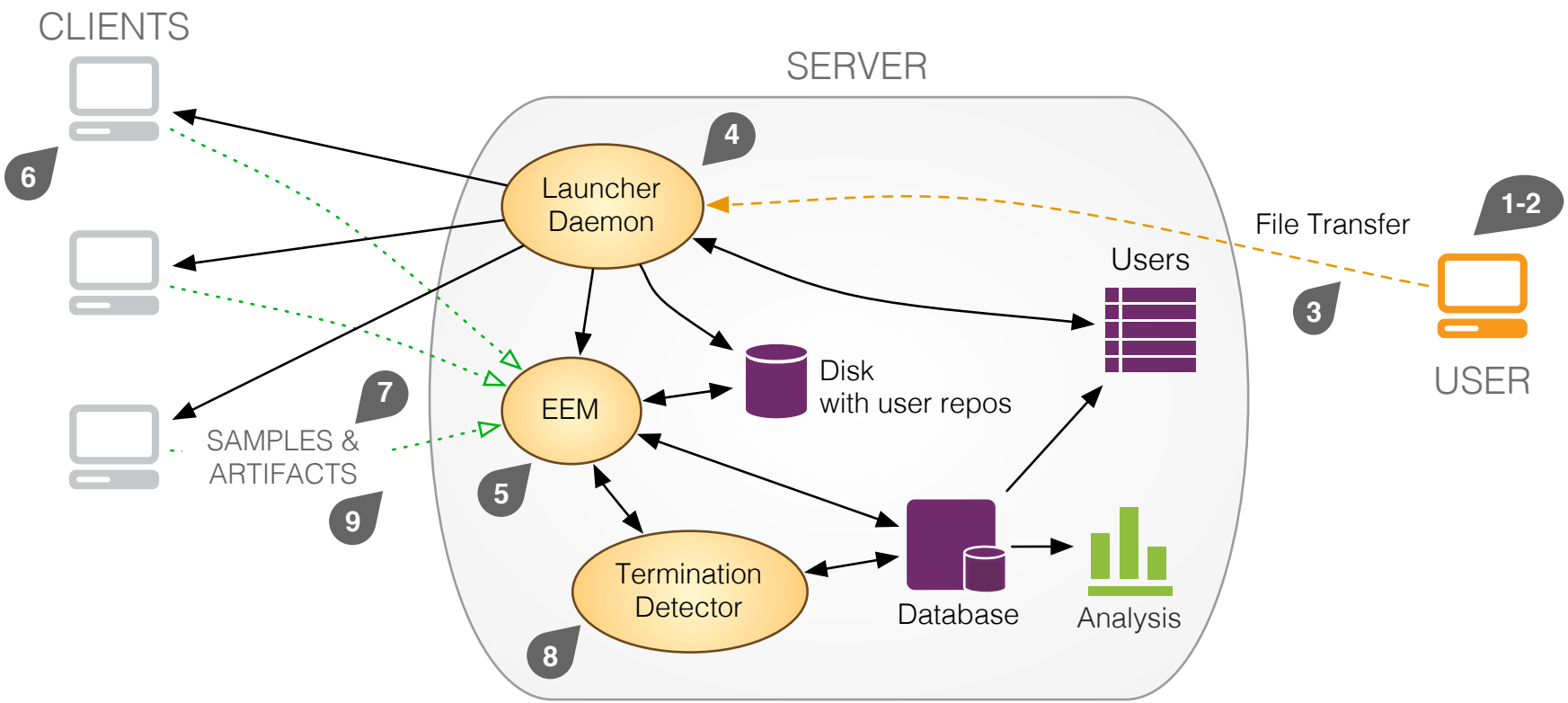
6) Clients detect number of cores and spawn one SC for each. SCs request a design point and spawn ns-3 execution: samples generated sent to EEM.

Workflow (7)



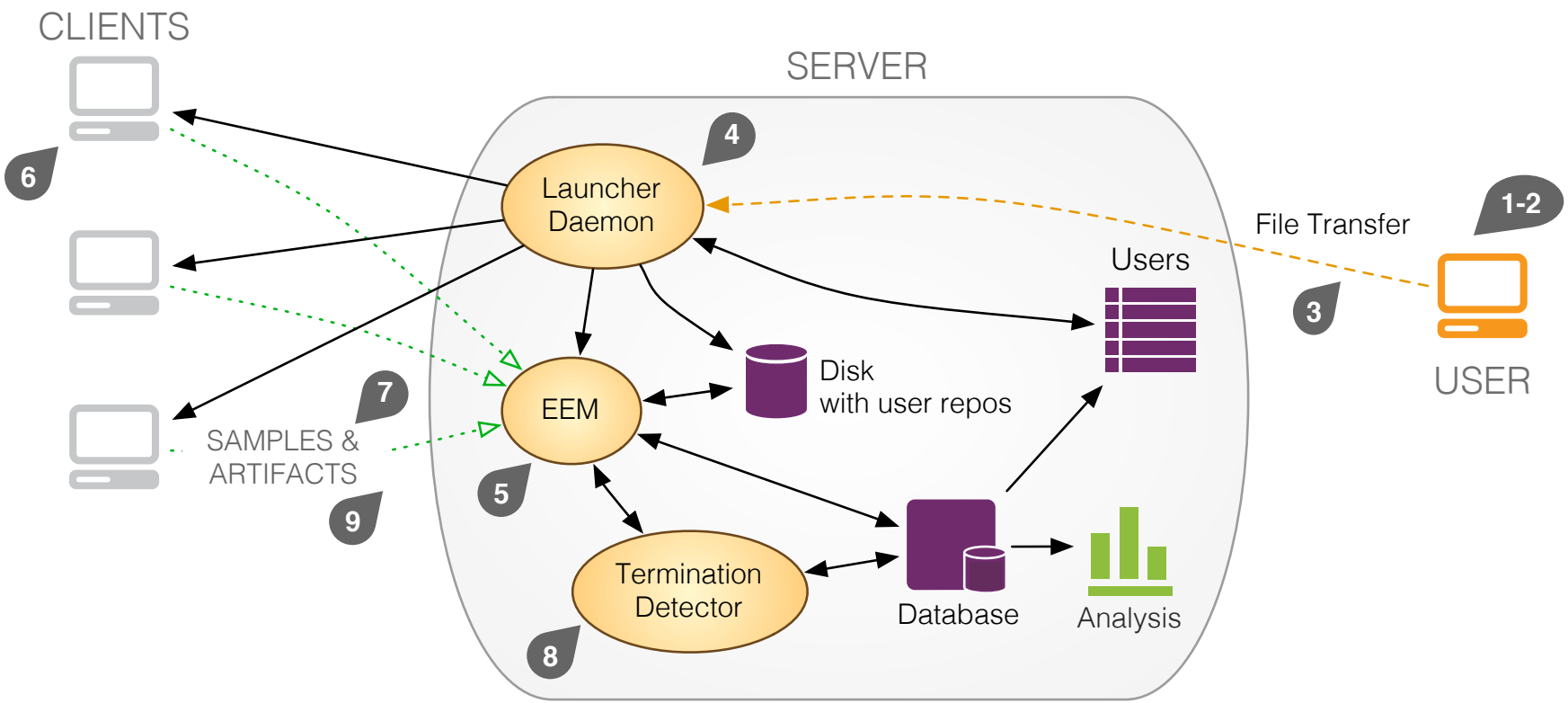
7) EEM receives samples and stores in database.

Workflow (8)



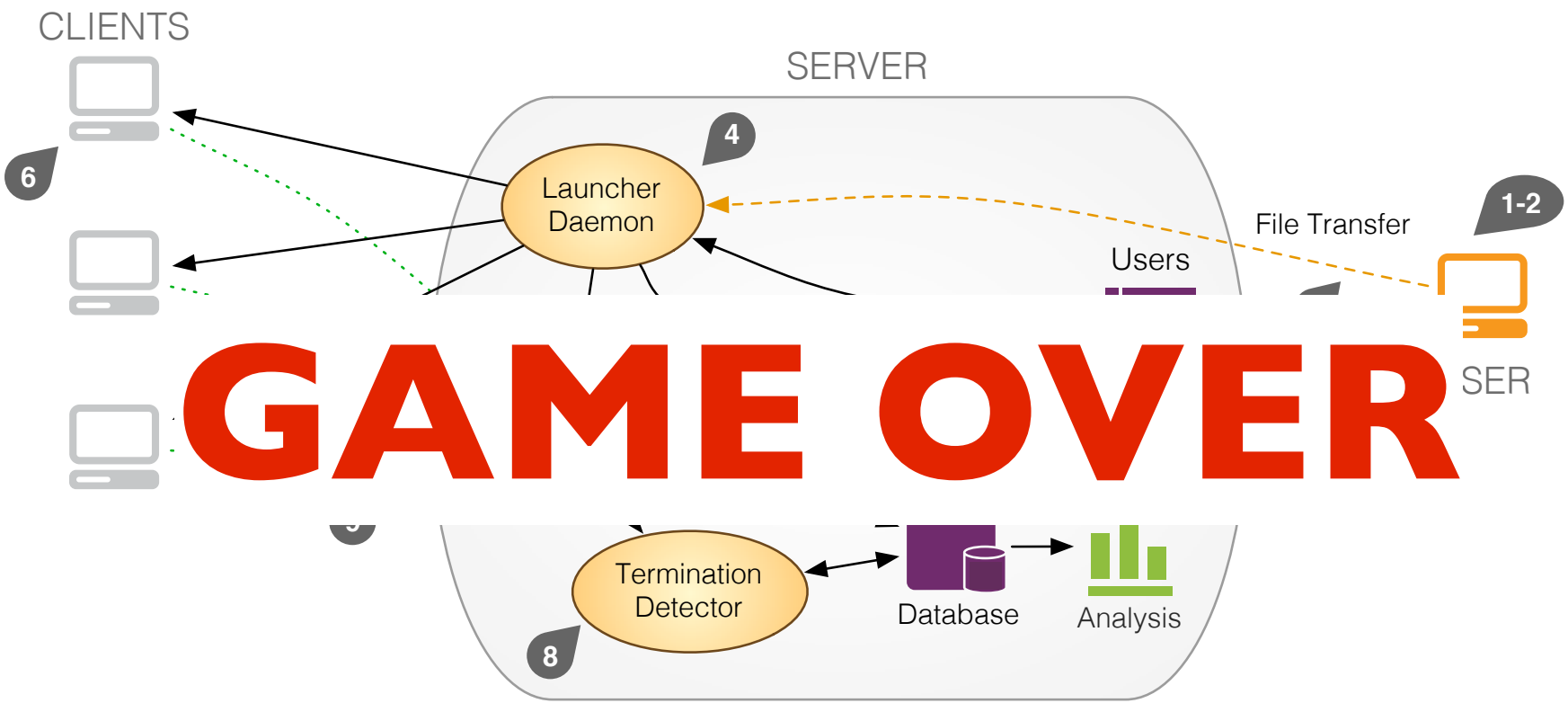
8) Termination Detector evaluates body of samples; when termination condition reached, tells EEM to send shutdown message to SCs.

Workflow (9)



9) SCs receive shutdown message: archive simulation artifacts generated locally and send to EEM.

Workflow


CLIENTS
SERVER
4
Launcher
Daemon
6
File Transfer
1-2
Users
SER
GAME OVER
Termination
Detector
Database
Analysis
8

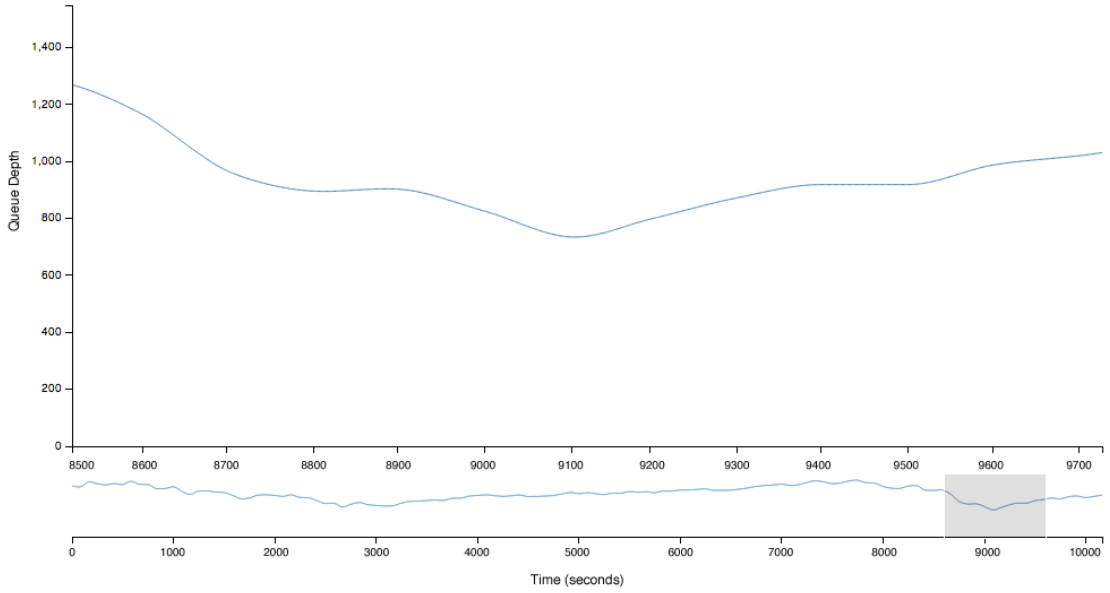
Ongoing and Future Work

Web Interfaces and Data Visualization

- Web interface for configuration and control.
- Exploratory data analysis through web browser.
- Generation of presentation quality graphs.

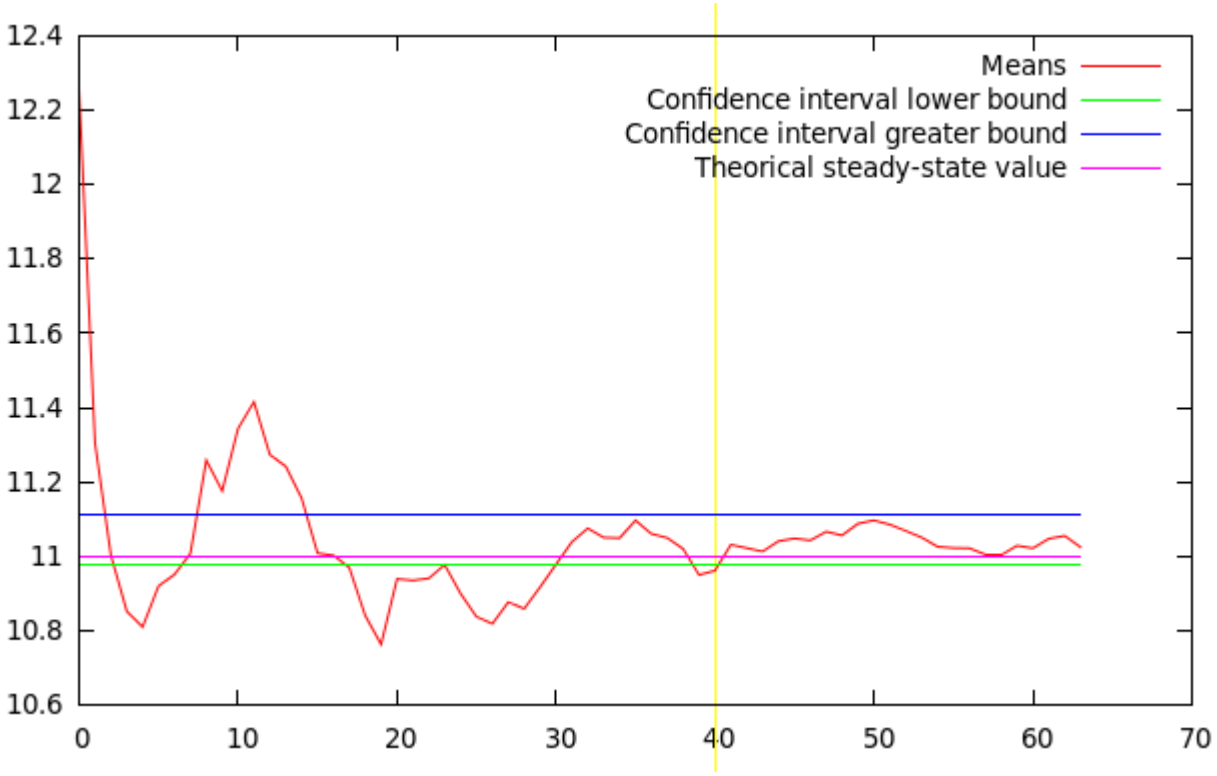
micro view

macro view



Steady-State Detection

- Batch Means
- MSER and variations
- MSER-5

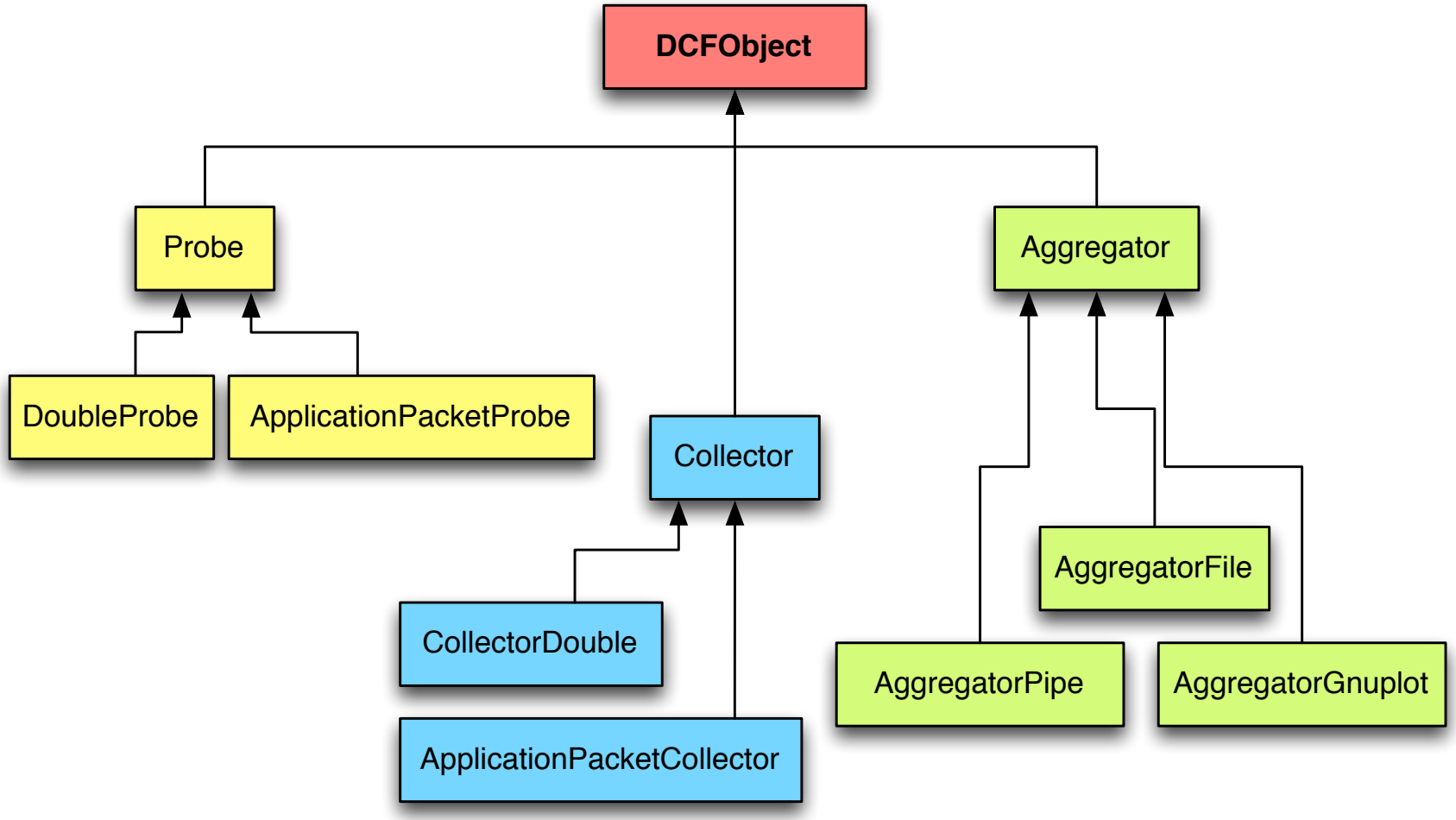


Data Collection Framework

Objective: extend TraceSource mechanism to facilitate output generation in ns-3

- **DCFObject:** base class for DCF elements
- **Probe:** extends TraceSources for controllability
- **Collector:** Arbitrary computations on sampled data
- **Aggregator:** Marshall data into various output formats

Existing DCF Classes



Thanks for your attention!

Questions?