CSCE 455/855 Distributed Operating Systems

Research Issues in Distributed Systems

Steve Goddard
goddard@cse.unl.edu

based on slides from Dr. Douglas Niehaus

http://www.cse.unl.edu/~goddard/Courses/CSCE855

1

Overview

- Design, implementation, and use of distributed systems raises a number of issues that can be broadly related to the basic issues of *research*
  - » Design is Hard
  - » Critical Thinking is Required
  - » Common Technical Paper Structure
  - » Experiment Plan Formulation
  - » Interpreting Results
- These issues apply to the topics of the class, to the semester projects, to research assistant work, and to most distributed software projects in industry

2

CSCE 455
Steve God

Design is Hard

- Design problems typically exhibit a wide range of options
  - » There is no single “right” answer
  - » *Right* and *wrong* may be only vaguely relevant terms
  - » *Effective* and *ineffective* are often more relevant
- Design problems also typically exhibit *several* constraints that must be satisfied
  - » Often the constraints compete or even conflict
  - » Designs must compromise among competing issues

3

Design is Hard

- Distributed systems are usually complex enough that it is not possible to fully predict their properties
  - » *Experimental measurement* should be part of design, development, and delivery methods

4

Critical Thinking is Required

- Design issues are often complex, interacting, and the most important issue in a given situation is often subtle
  - » Since appearances can be deceiving you must stay alert
- Regularly ask yourself
  - » What is the *real problem* and the *world view* within which it exists
  - » What properties should a *good solution* exhibit
  - » What questions will help evaluate how well my current *hypothesis* matches my current *world view* and *reality*
  - » How well do results from different tests match each other, my current hypothesis, world view, and reality
  - » What kinds of mistakes would currently be invisible

5

Critical Thinking

- The goal is to constantly improve the accuracy and level of your understanding
  - » Iterative refinement through hypothesis formation and testing designed to probe all aspects of the world view
- What is the *real problem*
  - » Distinguish *policy* and *mechanism*
  - » Always start with the most succinct statement of *policy* before worrying about *mechanism*
  - » As your understanding evolves, periodically revisit your problem description
- The simplest effective solution is usually the best
- BUT: you also have to allow for future extension

6

Critical Thinking

- Diagnostic Questions
  - » Designed to test the validity of your current hypothesis with your world view and against reality
  - » If A is true, then B should be true
  - » If you currently believe A, then testing B may be revealing
- Positive Questions
  - » Give answers that are useful in and of themselves
- Negative Questions
  - » Expected answers do not change your understanding
  - » Unexpected answers would signal something important
- Compare and contrast different answers for consistency

7

Reading and Writing Technical Documentation

- Technical writing is meant to fulfill a fairly narrow range of purposes, all of which fall under
  - » Communicate technical information for use by another
- Learn what to look for in documentation you are reading
- Learn what to put into documentation you write
- Narrow range of document types
  - » Project Proposals
  - » Conference and Journal papers
  - » Technical Reports
  - » Product Manuals

8

Reading and Writing Technical Documentation

- Project Proposals
  - » Convince people your idea is worth funding
- Conference and Journal papers
  - » Convince people what you did is worth telling others about through publication
  - » Conference papers are more current, shorter, and generally on a more narrow topic
- Technical Reports
  - » Detailed description concentrating on system design, implementation, and use for those extending the system
- Product Manuals
  - » Detailed description of how to use the system

9

Technical Paper Structure

- Most technical papers, proposals, and documentation follow, or *should*, a standard outline resembling:
  - » *Introduction*: Sets the conceptual framework and establishes why you should care about our system
  - » *Related Work*: What else is being or has been done in this area and how it compares to our work
  - » *Implementation*: How our stuff is built and works
  - » *Evaluation*: Why you should *believe* what we told you about how our stuff works and why you should *care*
  - » *Conclusions and Future Work*: Summarizes what you should now believe , why you should care, and what else we plan to do in the future

10

Technical Paper Structure
Introduction

- What is the problem
  - » Concentrate on answering this question and *only* this question in a succinct set of 1 to 3 paragraphs
- Why is it not already solved or other solutions are inferior in one or more important ways
  - » Your new idea need not solve every problem but it should solve at least 1 that is not already solved
- Why is our solution worth considering and why is it superior in some way
  - » A succinct statement of why the reader should care enough to read further
- How the rest of the paper is structured

11

Technical Paper Structure
Related Work

- What other efforts to solve this problem exist and why do they solve it less well than we do
  - » Resist the urge to point out only flaws in other work. Do your best to point out the strengths and weaknesses to provide as well rounded a view of how your idea relates to other work as possible
- What other efforts to solve related problems exist which are relevant, how they are relevant, and why they are less good than our solution for this problem
  - » Many times no one has solved your exact problem before, but others have solved closely related problems or problems with aspects that are strongly analogous to aspects of your problem

12

Technical Paper Structure
Implementation

- What we (will do | did): *Our Solution*
  - » Generally a higher level description of the solution and how it addresses the problem described
- How our solution (will | does) work
  - » Richer level of detail about how the solution works as appropriate to the type of paper (conference, journal, technical report, manual)
  - » Proposals are necessarily a good deal more vague in this section
  - » Manuals and technical reports have a lot in common but different audiences and thus different emphases

13

Technical Paper Structure
Evaluation

- How we tested our solution
  - » Performance metrics
  - » Performance parameters
  - » Experimental design
- How our solution performed and how its performance compared to that of other solutions
  - » Presentation and Interpretation of experimental results
  - » Why, how, and to what degree our solution is better
  - » Why the reader should be impressed with our solution
- Context and limitations as required for summation
  - » What the results *do say*, *do not say*, and *why*

14

Technical Paper Structure
Conclusions and Future Work

- What is our problem
- What is our solution to the problem
- Why our solution is worthwhile in some significant way
- Why you should be impressed
- What we will (or could) do next
  - » Improve our solution
  - » Apply our solution to harder or more realistic versions of this problem
  - » Apply our solution or a related solution to a related problem

15

Technical Paper Structure
Use the Standard Outlines

- LaTeX and MS-Word versions of the standard outline exist
  - » Check my WWW home page and class pages
- Surprisingly useful way to concentrate and organize thoughts in an stepwise and orderly fashion
  - » Start by filling the template in in outline form
  - » Concentrate on each question separately
  - » Make sure what you write answers the current question
  - » Move anything that is out of place
- Adaptation obviously required for specific case
  - » Proposal, conference paper, technical report
- Use it for the class project proposal and report

16

Experiment Plan Formulation

- Experiments are the *mechanism* used to accomplish a *purpose*
  - » Answering a question about a system
- Crafting the right set of questions is probably the single most important job skill in almost any technical field
  - » Still very intuitive for me and thus difficult to teach
  - » BUT some basic guidelines and tactics exist

17

Experiment Plan Formulation

- Three basic considerations
  - » Relevant Performance Metrics
  - » System Parameters affecting Performance Metrics
  - » Experiments evaluate the relations between parameters and metrics

18

Experiment Plan
Performance Metrics

- Ways of describing the performance of a system which are *relevant* to its *effectiveness* for *your problem*
  - Relevant metrics for one problem will be irrelevant for another problem
  - No single metric ever captures *every* relevant factor
  - Choosing metrics is thus partly a *design problem*
  - Interpreting the results is also related to *design decisions*

19

Experiment Plan
Performance Metrics

- You must also consider how the metrics in your set relate to one another
  - Rarely without at least a modest correlation (+/-)
  - Make some priority ranking
  - Source of cross-reference sanity checks on results

20

Experiment Plan
Performance Metrics

- Consider evaluating an ORB supporting sets of objects
  - » What are the relevant performance metrics?
  - » *It Depends* on what aspects of performance are relevant to the application or class of applications in question
- Candidate metrics include
  - » Throughput
  - » Latency: message transmission and connection creation
  - » Scheduling: message response and real-time deadlines
  - » Interactive response for users
  - » Response time of various services
  - » Event delivery time

21

Experiment Plan
Parameter Identification

- When you have the set of metrics, then consider the set of *parameters* which can affect system performance as measured by the metrics
  - » Which parameters can you control
  - » Which can you measure but cannot control
  - » Which can you neither measure nor control
- For each metric, consider the set of values you will use
  - » Some are naturally quantized
  - » Some are continuous: choose a range and a set of values within it

22

Experiment Plan
Parameter Identification

- The set of possible experiments is

  {metrics ⊗ parameters}

- Each (metric, parameter) pair is a relation question
  - » How is the metric affected by changes in the parameter
  - » World view enters here as basis for *hypotheses* about *how* and *why* a parameter is correlated with a metric

23

Experiment Plan
Parameter Identification

- Some correlations and their relevance are more obvious than others
  - » All are grounded in your system model and are thus *hypotheses* subject to *experimental verification*
  - » Obvious correlations are those that are *causal* in the current system model
  - » Many can be ignored if your world view is correct
  - » *Any* can provide insight if your world view is not correct

24

Experiment Plan
Parameter Identification

- Parameters can be even more varied than metrics
  - » They define all *relevant* aspects of the system state
  - » Including irrelevant parameters wastes time and effort
  - » Excluding relevant parameters wastes time and effort
  - » Which is worse depends on how relevant an excluded parameter is, compared to how much effort it takes to run an experiment
- An experiment is conducted within a *context* defined by the set of all parameters, and evaluates a *metric*
  - » Reproducibility of the results depends on whether all relevant parameters have been identified and controlled
  - » Erratic results indicate an incomplete parameter set

25

Experiment Plan
Experiment Design

- An experiment is designed to answer a question
  - » Sometimes this requires a set of experiments
- Several types of questions
  - » Does this system work?
  - » How well does this system work?
  - » Do I have an adequate understanding of the system?
- Cast in terms of metrics and parameters
  - » How does the value of a performance metric vary with the value of various parameters
  - » Predict metric value from parameters using system model
  - » Compare to behavior measured by experiment

26

Experiment Plan
Experiment Design

- Does this work?
  - » Build it, try it, measure it
- How well does this work?
  - » Cast in terms of metric values vis a vis parameter values
  - » How does throughput vary with bandwidth
  - » How does throughput vary with MTU size
  - » How does response time vary with system load
- Do I understand how the system works well enough?
  - » Are the relations I observe consistent with my model
  - » What further experiments might clarify my understanding of any inconsistencies

27

Experiment Plan
Experiment Design

- Positive experiments
  - » Evaluate the *interesting* (M,P_x) relations
  - » Expected results give valuable information
  - » Unexpected results indicate a flaw in the system model
- Negative experiments
  - » Evaluate a *less interesting* relation (M,P_x)
  - » Expected results are of no real use
  - » Unexpected results indicate a flaw in the system model
  - » Regression tests checking already “well known” relations are negative tests in this sense

28

Experiment Plan
Experiment Design

- Designing a single experiment evaluating *(M,P_x)* requires making several choices
  - » Metric for evaluation
  - » Parameter P_x which is the focus of the relation
  - » Set of all other parameter values *(P_w, P_y, P_z)* which define the unchanging aspects of the system context
  - » You also have to ask yourself if you have adequate control over all parameters influencing relevant behavior
- Conduct the experiment, collect the results, and consider how to look at them to determine features of interest

29

Experiment Plan
Experiment Design

- Evaluating (M,P_x) relations is even harder than implied because *most parameters are not independent*
- Consider one metric *M* and two parameters P_a and P_b
  - » We wish to consider relations *(M, P_a)* and *(M, P_b)*
  - » An experiment evaluates *M* in the context of several points in the parameter space *(P_a, P_b)*
  - » The the ranges of P_a and P_b define the parameter space
  - » Assume five values in each parameter range $P_x(1) \rightarrow P_x(5)$
  - » If P_a and P_b are independent then the relation *(M, P_a) does not* depend on the value of P_b
  - » Much of the time it *does* $\rightarrow$ 25 measurements in *(M,P_a,P_b)*
  - » Sometimes 3D plots are important and revealing

30

Experiment Plan
Sets of Experiments

- Most often the relation of interest *(M,P_x)* requires several experiments to accumulate enough data
  - » Graphs of how *M* varies across a range of P_x values
  - » Gather everything in one experiment if possible
  - » Automating the execution of several experiments is good when separate runs are required for meaningful data sets
- Ordering the sets of experiments you run is important
  - » Always choose the experiment that returns the most useful information *at that time*
- Information is useful in many ways

31

Experiment Plan
Sets of Experiments

- Forms of utility of results
  - » Contributions to understanding system behavior
  - » Cross checks on understanding system behavior
  - » Calibration and sanity checks of experimental methods
- Go from most fundamental and valuable to most subtle and least valuable
  1) Calibration of measurement methods
  2) Basic structure of relations *(M,P_x)*
  3) Cross reference relations and sanity checks
- Increase level of detail by iterating on Step 2 leavened with elements of Steps 1 and 3

32

Experiment Plan
Sets of Experiments

- Since the parameter space is large, iteratively refining your coverage can save a lot of time
  - » Each parameter P_x has a range of values
  - » Choose the first, last, and a few in between for a first level assessment of (M, P_x)
  - » As each predicted relation is roughly evaluated, consider the quality of the system model *wrt* the results
  - » Adjustments to the model require rerunning only these crude experiments
  - » Refine results by covering the full range of P_x at the full resolution when confidence in the system model is high

33

Interpreting Results

- It is important to employ critical thinking when analyzing data sets from experiments
  - » Perform the intended and obvious analysis first
  - » Compare results to expectations
  - » Evaluate the same relation (M,P_x) in more than one way when possible
  - » Apply sanity checks to raw data (e.g. make sure time stamps are recorded in *increasing* order)
  - » Keep raw data for further analysis when possible

34

Interpreting Results

- Good organization and record keeping is important
- Need a good description of points and regions in a large parameter space
- Results from earlier experiments suggest new questions
  - » These translate into further experiments

35

Interpreting Results

- Discovery of anomalies and inconsistencies can motivate changes in the system model
  - » Previous experiments may need to be rerun
  - » Recall the advice to iteratively refine coverage and evaluation of relations (M,P_x)
- Ultimately you are evaluating a system to *tell a story*
  - » How the system works
  - » Why it works that way
  - » Why your audience should believe you and care

36

Conclusions

- The goal of software design and implementation is to design systems that do things people care about
  - » Lots of people currently care about distributed systems
- In a general sense, *research* asks and answers questions
  - » All technical documentation considers solutions to problems in one of several basic ways
- A large percentage of technical documentation thus follows the same basic outline
  - » Define the problem, place it in context, describe a solution, evaluate the solution, conclude
- Doing this well is less common and harder to do than you might think

37