Introducing 8th Grade Girls to Fault Tolerant Computing (An Experience Report)

Paul A.G. Sivilotti
Murat Demirbas

Dept. of Computer & Info. Science
The Ohio State University

{paolo,demirbas}@cis.ohio-state.edu

The Context

- "Future Engineers' Summer Camp"
  - Piloted at OSU Summer 2002
- Workshop for 8th grade girls
  - 30 participants
  - 1 week (days only) on campus
- Theme: Introduction to "engineering and science"
  - Mechanical, chemical, civil, astronomy, environmental, industrial, …
  - Lectures, lab tours, activities

The Challenge

- Design a 3-hour module for CS
- Goals:
  - Fun
  - Educational
  - Reflection of CS as a discipline
- Requirements:
  - No CS background assumed

Approach 1: Logo

- Use a simple imperative programming environment
  - E.g. "Darwin's World" exercise in CS 1/2
    - Simple programming language to control bug movement, replication
    - Bugs interact, infect, thrive, die
    - Ref: SIGCSE '99 panel on nifty assignments
- Appeal:
  - Conditionals, iteration, recursion,…
- Problems:
  - Syntax is a distraction
  - Low engagement for this audience

Approach 2: Using App's

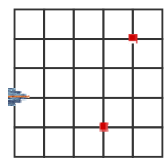
- Use engaging applications
  - E.g. tool for designing web pages
- Appeal:
  - Gender-appropriate applications can be chosen
  - Clear, identifiable skill is learned
  - Sense of accomplishment from an impressive final product
- Problem:
  - Not CS!

Our Approach

- Teach:
  1. Software engineering principles
  2. Parallel programming
  3. Self-stabilizing distributed algorithms
- Three graduate-level CS topics!
  - Each builds on the previous
  - Each consists of lecture + activity (1 hr)
  - Consistent theme:
    - Programs as recipes
    - Computers as chefs

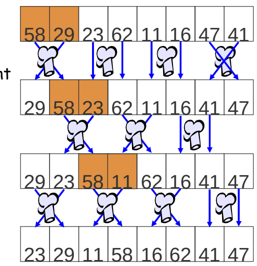
Topic 1: Programs

- Lego Mindstorm robots
  - Light sensors
  - Follow grid lines
- Instructions
  - Turn, forward, take sample
  - Printed on cards
- Cards stacked to form programs
  - Straight-line
  - Simple requirements
  - Uploaded to robots



Topic 2: Parallel Programs

- Each person holds a number
  - Physically move to represent data movement
- Sequential sort
  - Bubble sort
- Parallel sorts
  - Even-odd transposition
  - Radix
- Differences apparent
  - Execution time
  - Multiple threads
- SIGCSE '94 paper



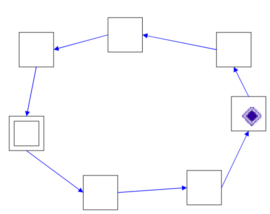
Topic 3: Fault Tolerance

- Lecture
  - Nature of faults (chef analogy)
  - Easy answer: redundancy
    - Follows directly from parallel algorithm unit
  - Self-stabilizing token ring algorithm
    - Correct state: 1 token (mutual exclusion)
    - Possible faults: token loss or duplication
    - Converges to correct state
    - Distributed control

Fault Tolerance Activity

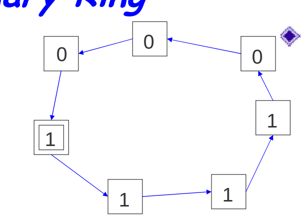
- Design goals
  - *Simple* rules
  - Reinforce *distributed* nature of algorithm
  - *Dramatic* difference between correct & incorrect states
  - *Satisfaction* in re-establishing correct state
- Solution:
  - Use music!
  - Students are in a ring, each with a chime
  - When they have the token, they play their note
  - Correct (1 token) = melody
  - Incorrect (multiple/none) = chaos / silence

Fault Intolerant Token Ring



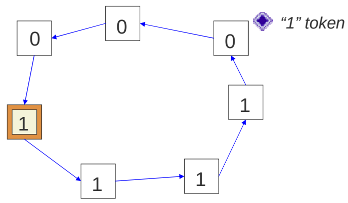
- Problem: What about faults?
  - What happens if token is lost?
  - One fault means disaster!

Prevent Loss of Token: Binary Ring



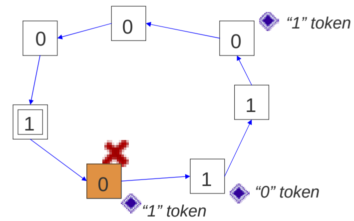
- Rule (for most people):
  if left neighbor is different from me, then I have the token
  - Make my number equal to that neighbor's

Completing the Ring



- One person is special:
  if left neighbor is same as me,
  then I have the token
  - Make my number differ from that neighbor's

Fault: Corruption of Values



- Problem: multiple tokens in ring
  - Tokens chase each other around ring
  - One fault means disaster

k-State Token Ring

- Solution: use more values than people!
- Same rule
  - If left neighbor different from me:
    - I have the token! (use it)
    - Change my value to be equal to neighbor
- Again, one person is special
  - If left neighbor *same* as me
    - I have the token (use it)
    - Change my value to be *one bigger*

Activity

- Form a ring
  - Each person has number cards
  - Each person has a chime
- When you get the token:
  - *Play your chime*
  - Then change your number
- We'll run different versions
  - I'll introduce "faults" and see if you can recover!

FT Demo: Tips for Success

- Recognizable tune with equal note lengths
  - *TTLS*, a scale, *Frère Jaques*, *Carmen Ohio*…
  - Use a large group (14 notes worked well)
- Do *not* align tune with processor 0
  - Supervise the "special processor"
- Binary ring:
  - Allow tune to emerge, then disrupt
  - After fault, make sure all 3 tokens appear
  - Change tune after binary ring
- K-state ring:
  - Disrupt *before* tune emerges
  - Start from random state for effect

Participant Evaluation

- "How much did you know about CS before?"
  - 2.8 (1 = none, 5 = a lot)
- "Is CS now more or less interesting?"
  - 4.0 (1 = less, 5 = more)
- "Most important thing learned?"
  - "It's really fun"
  - "Computers need specific instructions"
  - "Sequential programs are slow"
  - "How a program can recover from faults"
- "In which activity did you learn the most?"
  - Most popular selection: CS

Conclusions

- Effectiveness of anthropomorphism
  - Caveat: encourages operational reasoning
- Try the fault tolerance activity!
  - Works best as a 3-part series
  - But each part can work individually too
- Age neutral
  - Middle school, HS, UG, Grad
- Slides, notes, and code available:
  http://www.cis.ohio-state.edu/~paolo/FESC02

19

Acknowledgements

- Graduate student assistants
  - Sandip Bapat
  - Florina Comanescu, Scott Pike, Nigamanth Sridhar, Hilary Stock
- School of Music
  - Amy Giles, Prof. Ken Williams
- FESC Workshop
  - Prof. Linda Weavers
- Funding:
  - National Science Foundation
  - Ameritech

20

Introducing 8th Grade Girls to Fault Tolerant Computing (An Experience Report)

Paul A.G. Sivilotti
Murat Demirbas

Dept. of Computer & Info. Science
The Ohio State University

http://www.cis.ohio-state.edu/~paolo/FESC02