

CS 598IG: Advanced Topics in Distributed Systems Spring 2006

Indranil Gupta
Structuring Project Code:
“The 1 Line Solution”
© November 11, 2004

Background

Discussion – Studying *Your* Protocol

- How accurate are mathematical analyses?
 - Often simplistic, so we resort to simulations, often trace-based...
- Simulations easy to do – implement, and run on your machine (or a small cluster)
- How accurately can simulations model real-world stresses?
- How do we know that we’re accounting for all possible kinds of failure?
- All possible kinds of stresses? All possible kinds of traces?

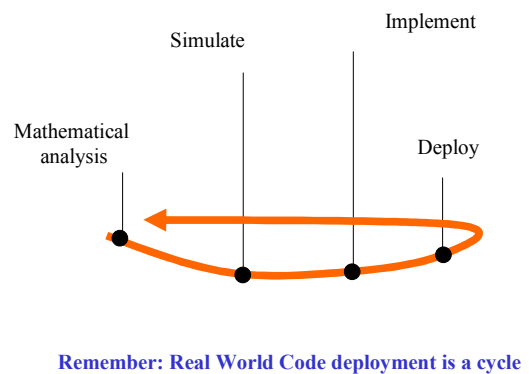
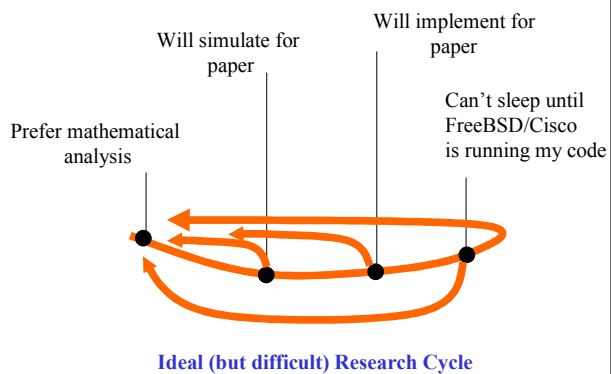
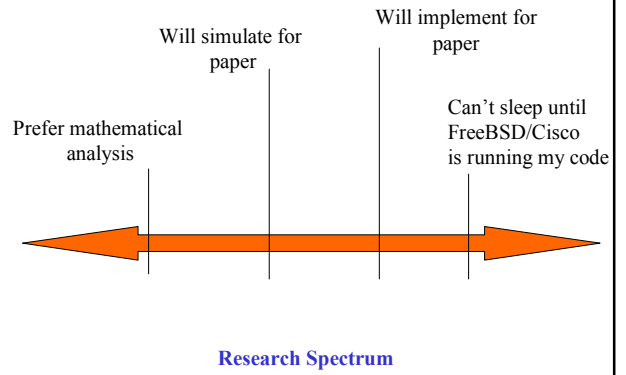
Discussion – Studying *Your* Protocol

- *Can* simulations ever model reality accurately?
- Is deployment the ultimate test?
- Have you seen any papers that match simulation and real-world running experimental numbers?
- Why?

- Unfortunately, often “The paper is the system” in research

As a Result

- Rare for someone else to pick up your idea, implement it and run it in the real world (although it does happen, there are too many ideas out there...)



Presumption

- Assumption: Rare for someone else to pick up your idea, implement it and run it in the real world (although it does happen, there are too many ideas out there...)
- Deployment is mostly your responsibility
- **Problem:** Design your simulation code so that you can convert your code from simulation → deployable version by changing a single line of code
- “1 Line Solution”

For Your Project

“How do I write code for my Distributed Protocol XYZ so that I can evaluate it with 100, 000 nodes?”

The 1 Line Solution

Writing The Code

- Simulation engines (ns2, glomosim) etc. are one option
- A required standard in some research communities (e.g., ad-hoc networking)
- Not so in the p2p or (largely) the sensor net communities (yet)

Writing The Code

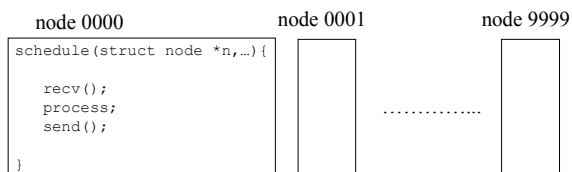
- Let's talk about a second option - **Basic Custom Evaluation...**
- **Threads – a bad idea!** (100K threads on Linux? Try it!)
- Ultimate goal – write **real deployable code** that can run on a socket API/your favorite OS
- But also generate numbers for 1000, 10K, 100K nodes
- **Simulation** → structure it so it's easy to do both of above by changing just one line of code
- How?

```

struct node{
  char nodeid[6]; // ip(4),port(2)
  .
  .
  .
}
    
```

```

struct node{
  char nodeid[6]; // nodeid[0] assigned int value
}
    
```

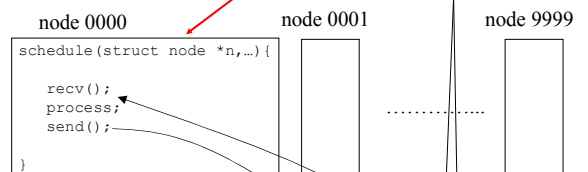


```

struct node allnodes[10000];
for (i=0;i<=9999;i++)
  schedule(allnodes[i]);
    
```

```

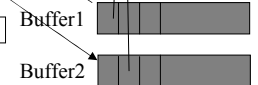
struct node{
  char nodeid[6];
}
struct msg{
  char src[6];
  char dest[6];
}
    
```

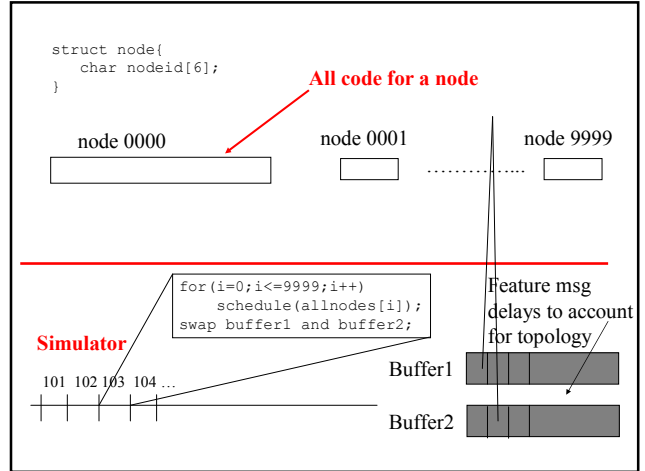
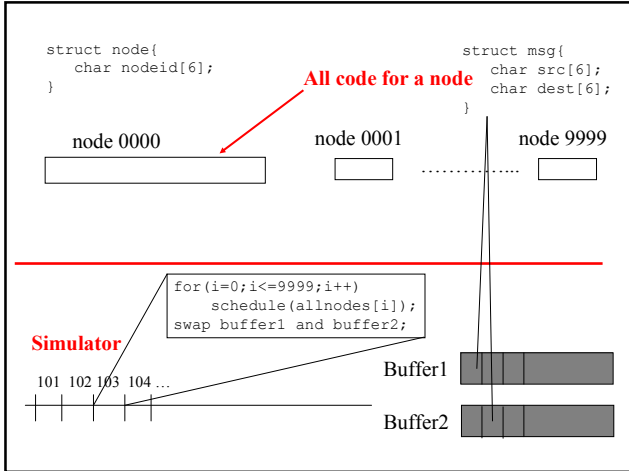


Simulator

```

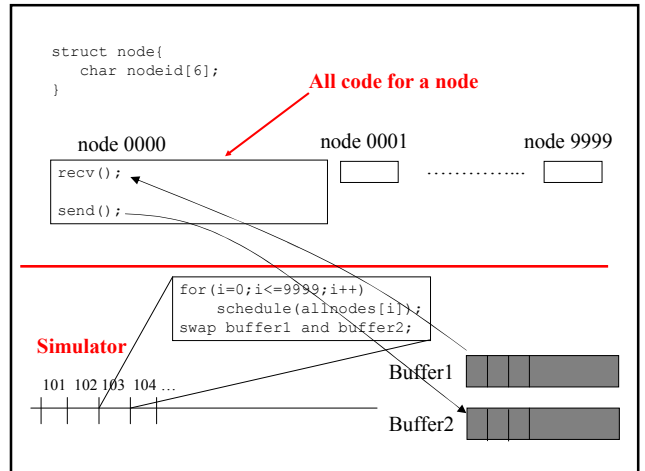
struct node allnodes[10000];
    
```

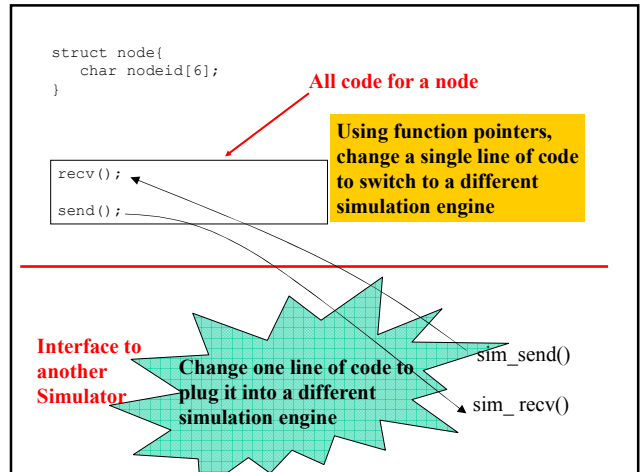
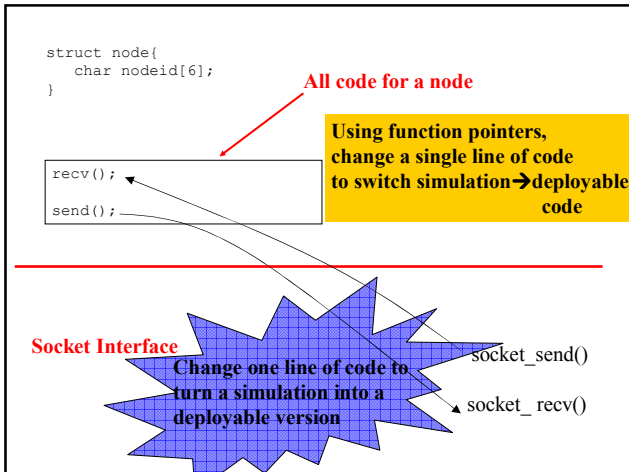




The advantage of such an elaborate spread?

- Layering gives clean separation of implementation from simulation
- Easy debugging (No global variables for the implementation, please!)
- And...





- Easier to do above with C or Java or C++
- Can put an “Application” layer on top of the “Real Code” layer
- Of course, you are free to structure your code in a different way should you so wish...

Questions