A. Arpaci-Dusseau **Department of Computer Science**
CS739: Distributed Systems **University of Wisconsin, Madison**

Transparent Process Migration: Design Alternatives and the Sprite Implementation – SP&E 1991

1 Motivation

- In what environment did the designers think Sprite migration would be used? How did these assumptions impact their design?
- What was the primary motivation for migration in Sprite?
- What are some of the drawbacks of simple *remote invocation* (e.g., using rsh) (or non-preemptive load balancing) compared to migration? Which can be fixed relatively easily?
- Process migration systems need to make compromises between four competing goals: transparency, no residual dependencies, performance, and low complexity. For each goal: what does each term mean? why is it desirable? Which did the authors emphasize?

2 The Overall Problem: Managing State

- Migration is hard because of process state, including virtual memory, open files, message channels, execution state, and other kernel state. In terms of transparency, residual dependencies, performance, and complexity, what are the pros and cons of transferring state? forwarding requests? ignoring old state? Is there a fourth approach for dealing with state?

3 Mechanics of Migration

- Numerous approaches have been investigated in the literature for transferring virtual memory. What are the pros and cons of stop-and-copy? Pre-copy? Lazy copying? How does Sprite migrate memory?
- Do you think the approach taken by Sprite makes sense given their assumptions?
- Interactions between file system state and migration are surprisingly complex. Why was forwarding not used? Sprite instead chose to transfer the state (ref count, dirty blocks, and access position) associated with open files. What went wrong when they tried to close the file on the source and then reopen on the target? What went wrong when they tried to open the file on the target before closing the file on the source? What did Sprite need to do to fix these problems? How do they handle access position when it is shared between the source and target?

- How is the process control block handled? Do you think there is something to generalize here?

4 Migration Policies

- When does Sprite migrate jobs? How does Sprite determine where to migrate a job to? When is a machine eligible to be a target?
- What do you think are the drawbacks of having your home machine (when idle) selected as a target?

5 Performance

- What type of processes are expensive to migrate?
- Why does compile and link get far from linear speedup?
- What do you think of the threshold of 30 seconds of idle time?

6 History and Experience

- Anything interesting here?
- Do you think that a general, transparent migration service warranted?

7 Comparison to Remote Execution in the V-System

- Were the assumptions and goals of the two systems similar or not?
- How did the system architecture of V impact migration?
- What was most interesting about V migration?