

CS134c: Computing Systems Lab

- Course outline
 - *Projects*
 - *Topics*



Administrivia

- Instructor: Jason Hickey, jyh@cs.caltech.edu
 - *Office hours: MW 3-4pm*
- Course times
 - *April 1, M 2-3pm, 74JRG*
 - *April 13, F 2-3pm, 74JRG*
 - *April 26, F 2-3pm, 74JRG*
 - *May 10, F 2-3pm, 74JRG*
 - *May 24, F 2-3pm, 74JRG*
 - *May 31, F 2-3pm, 74JRG*



Online info

- Course web site
 - <http://www.cs.caltech.edu/courses/cs134/cs134c/>
- Mailing lists
 - *General discussions*
 - cs134-labs@metaprl.org
 - *Private correspondence*
 - cs134-admin@metaprl.org



Course info

- Prerequisites
 - CS20
 - CS134ab are recommended, not required
- Grading
 - Course is either 9 or 12 units
 - 5 presentations, 10% each
 - Final project, 50%



Projects

- Must have some relation to operating systems, compilers or both
- Examples
 - *Build an operating system*
 - *Implement a distributed filesystem*
 - *Design and implement a new network protocol*
 - *New language + compiler*
 - *Process mobility*



Teams

- Projects are in teams of 1-3 people
- Larger groups are expected to propose a larger project
- Presentations
 - *10 minutes each*
 - *Five times during the term*



Schedule

- Class discussions
 - *April 13: project proposals due*
 - *April 23: mid-term progress report*
 - *May 10: second presentations*
 - *May 24: final project discussions*
 - *May 31: final project discussion*
- Weekly 15 minute meetings scheduled with jyh
 - *Sign up with Betta, bettad@cs.caltech.edu*



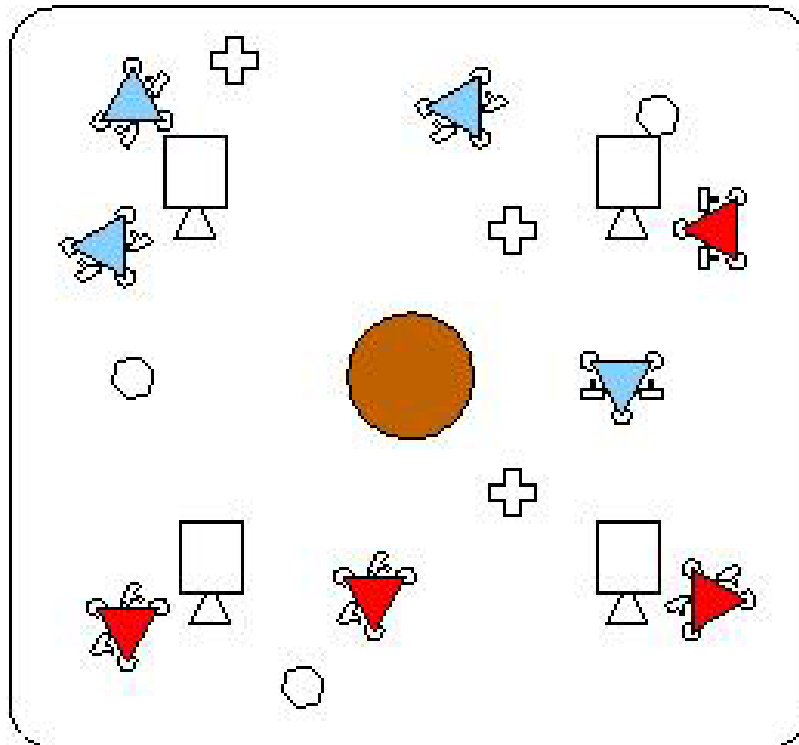
Course topics

- Projects
- Distributed operating systems
 - *Distributed communication protocols*
 - *Group membership*
- Languages for distributed systems
 - *Type theory, abstraction, polymorphism*
 - *Shared state, non-shared state model*
 - *Security*



Projects: multi-vehicle wireless testbed

- Active research
 - <http://www.cds.caltech.edu/~murray>

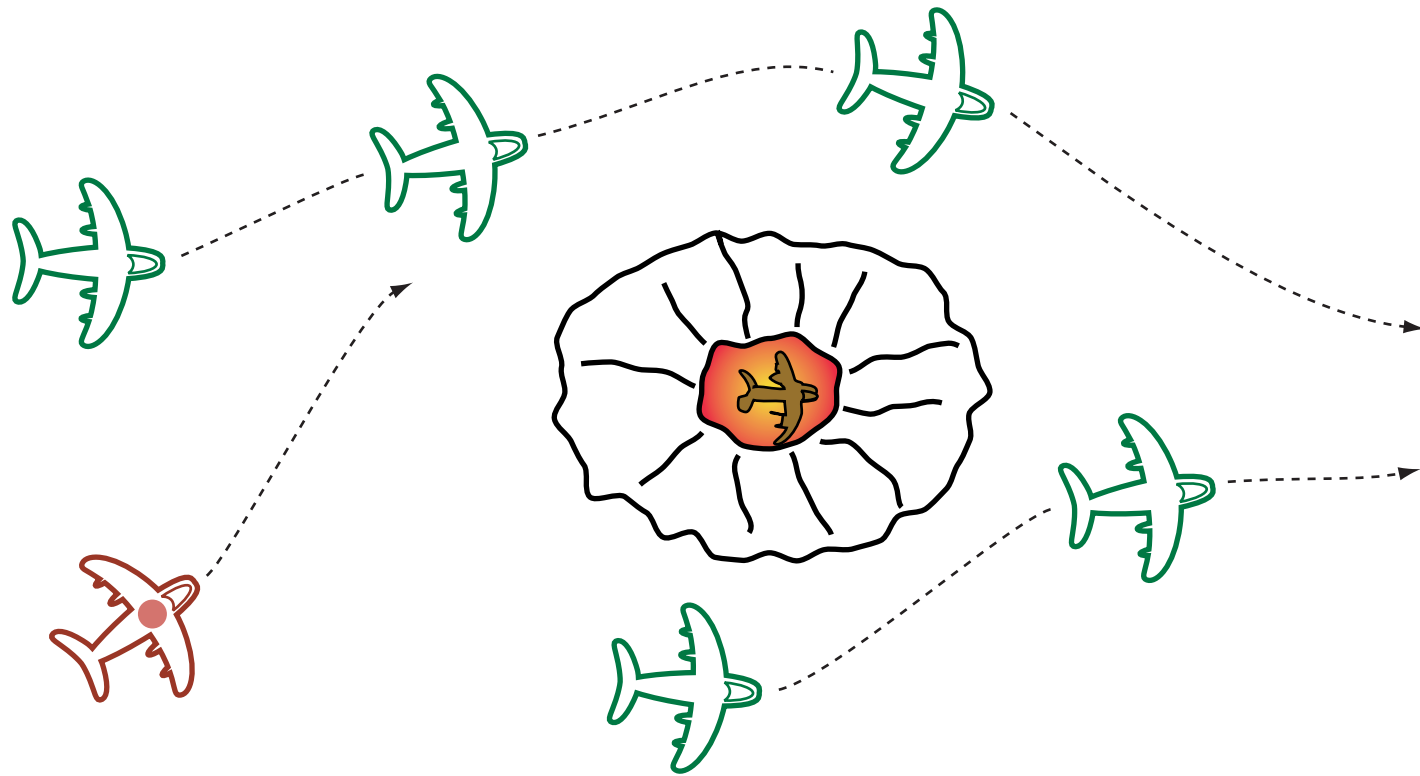


Multi-vehicle wireless testbed

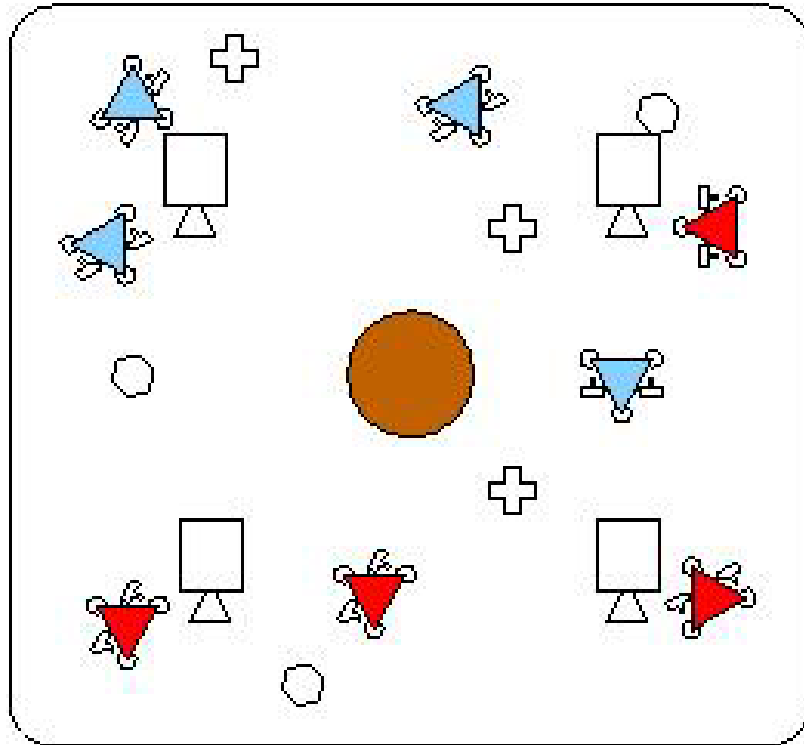
- Two teams of vehicles in a dynamic environment
- Vehicle are autonomous, each has
 - *some degree of compute power*
 - *a communication system*
- Communication is dynamic
- Control is real-time



Idea



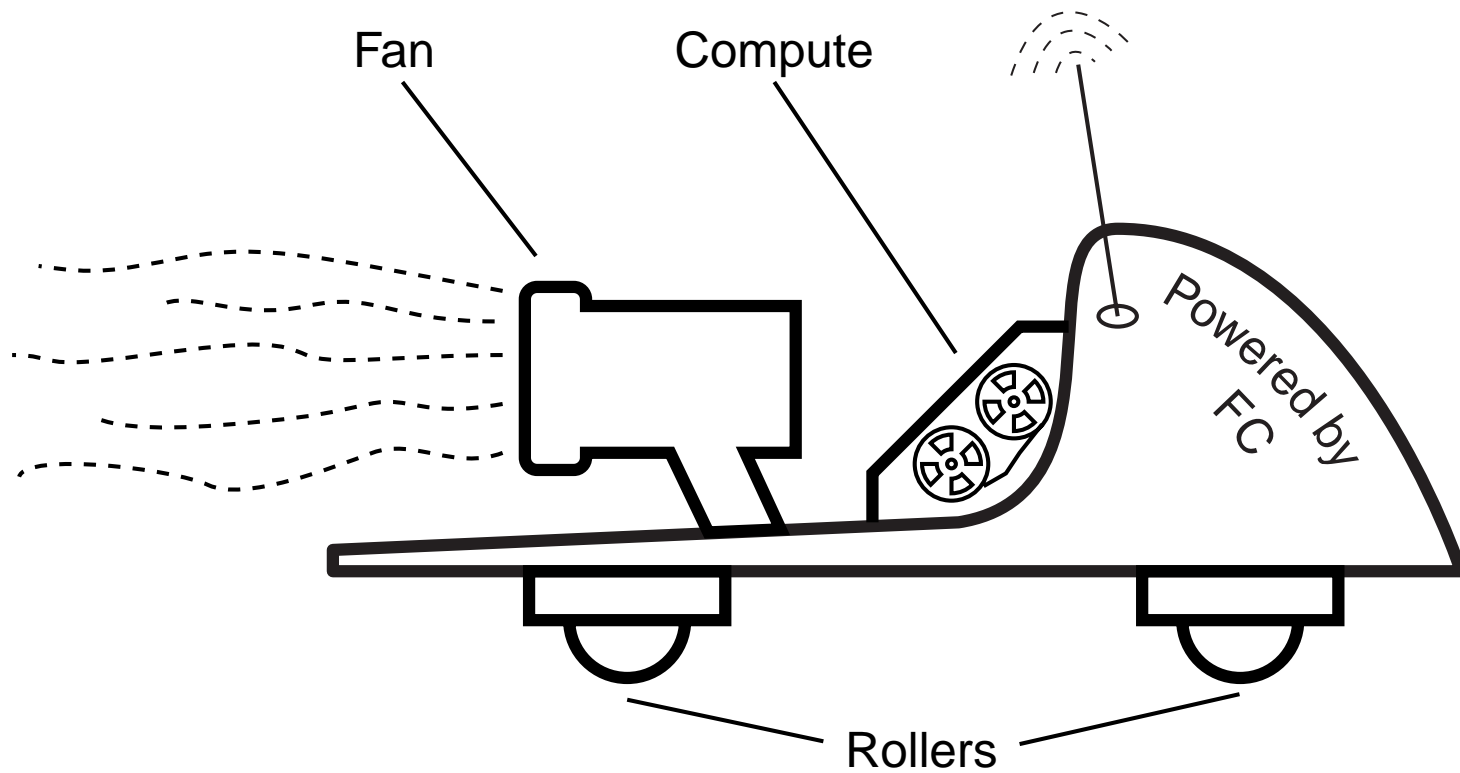
Testbed



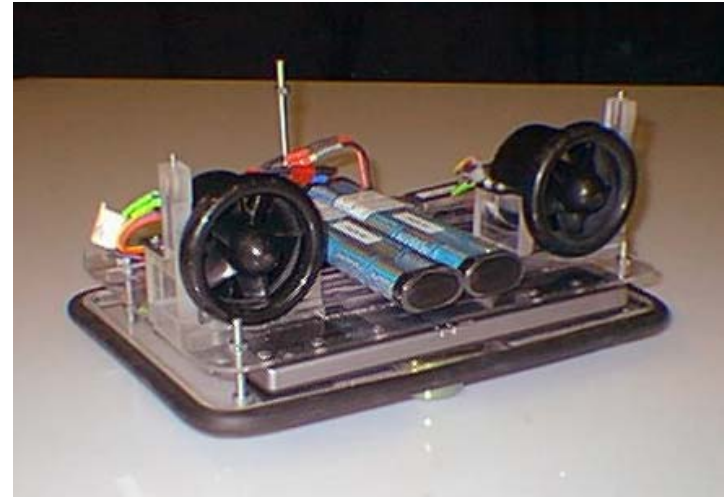
- Multiple vehicles
- Fan-driven
- Wireless communication
- Approximate positioning
 - *Vision*
 - *Sensors*



Vehicle



Actual vehicle

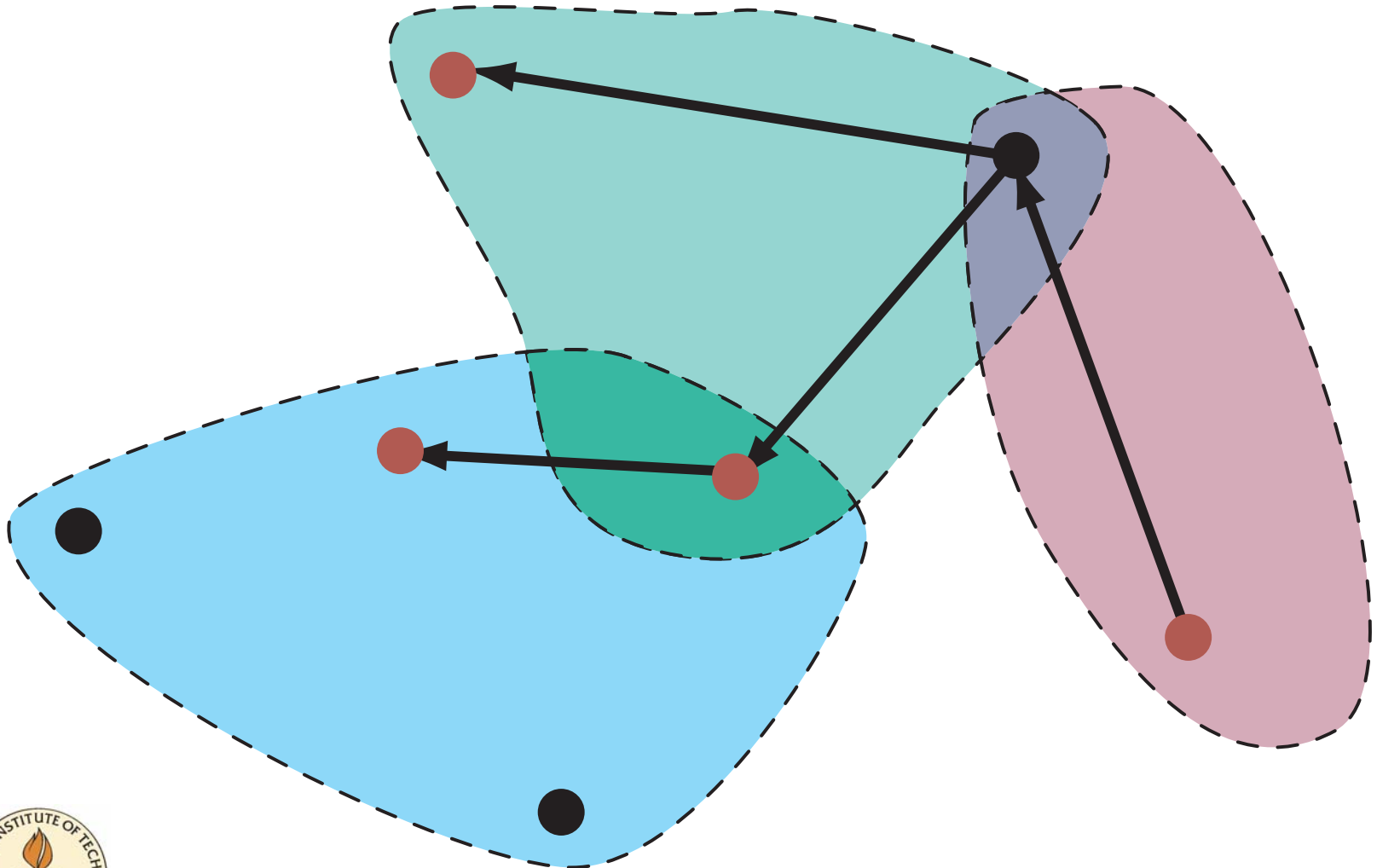


Project parts

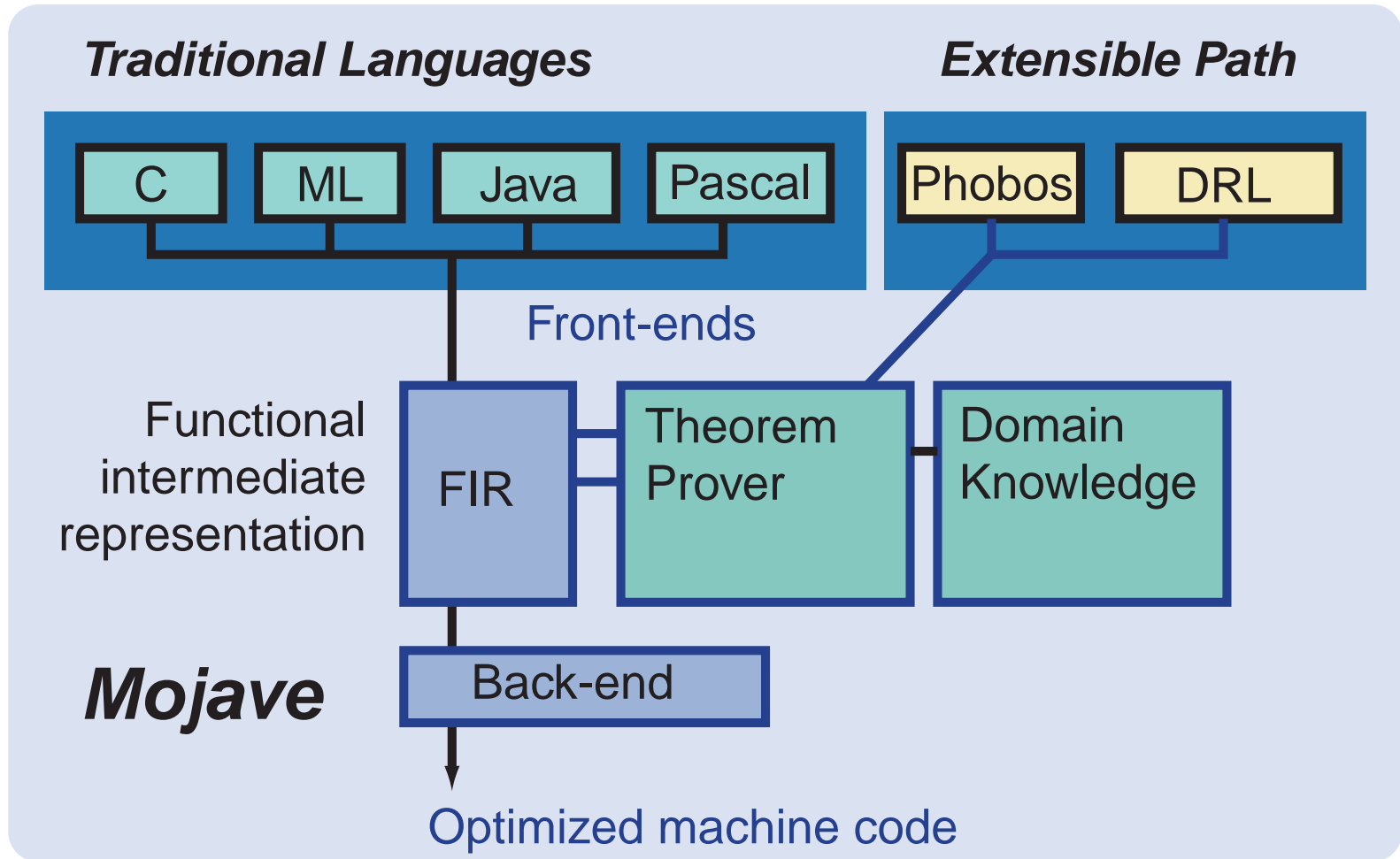
- Real-time scheduling
- Real-time planning
- Distributed sensing
- Distributed (fault-tolerant) communication protocols
- Event/agent languages
- Algorithm synthesis



Example: subgroup communication



Languages



Language topics

- Front-ends (ML, Pascal?, Java?)
- Back-ends (alpha?)
- Distributed operating system
 - *Thin-kernel (compiler guarantees safety)*
 - *Process mobility (code motion across machines)*
 - *Communication, where does it fit?*
- FC: extend to the full C language
 - *Adds: polymorphism, threads, exceptions, mobility*



Process mobility

- Back-end, when a process moves
 - *marshal code, state*
 - *send as a message*
 - *recipient re-activates process*
- Front-end
 - *What are the constructs for process mobility*
 - *How do we guarantee safety?*



Implementing process mobility

- Linux mobile process module
 - *Code migrates into kernel*
 - *Kernel links against “public” interface*
- Process server
 - *Receives requests from migrating processes*
 - *Invokes back-end, activate process*
- Load-balancing
- Resource management



Implement an OS

- Features:
 - *Processes*
 - *Virtual memory, swapping*
 - *Disk I/O*
- System calls
 - *gettimeofday()*
 - *semaphore operations*
 - *fork()*
 - *file operations*



Implementing an OS

- Platform
 - *SPARC?*
 - *VMware i386?*
- Boot block is easy
- Need to understand BIOS
 - *Interrupts*
 - *Device I/O*
 - *System/user mode*
- Should work in a team

