



Distributed Computing
Laboratory
(DisCo Lab)

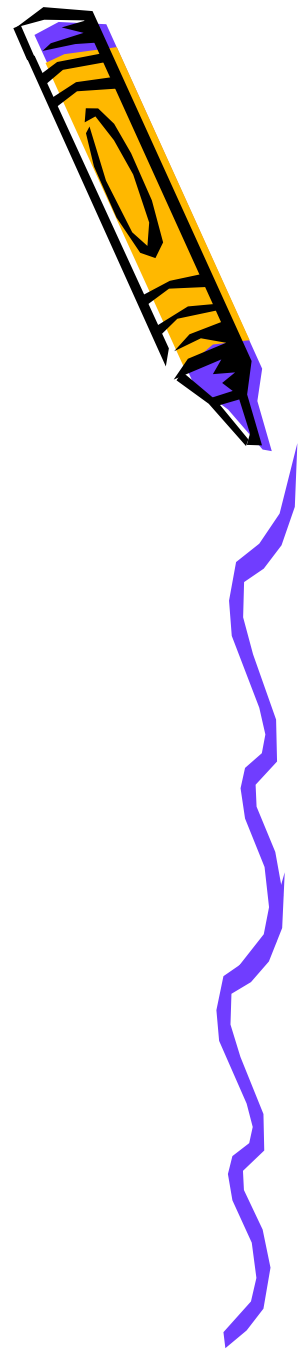
Liviu Iftode

Department of Computer Science
Rutgers University

<http://discolab.rutgers.edu>

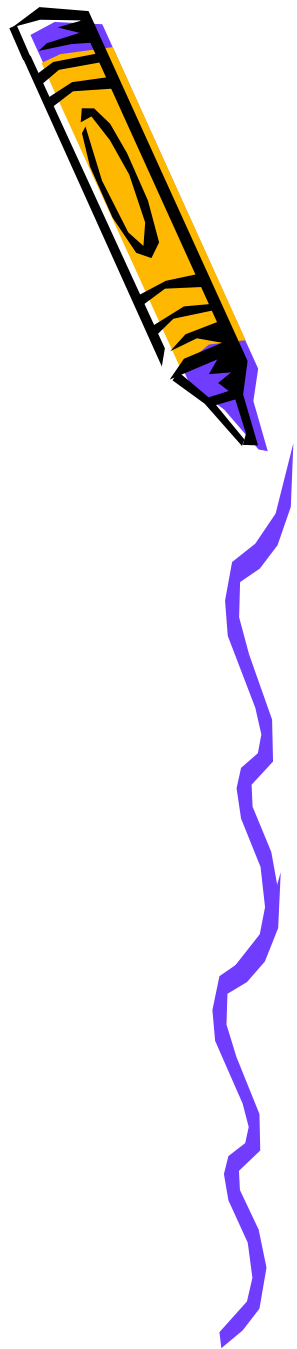
Disco Lab

- Created in 1998
- Research focus: Distributed Computing
- 8 Ph.D. students
- 2 undergraduate students/year
- 2 visitors/year
- 8-10 conference/journal papers/year
- Funding: NSF: >2 Million \$(2001-2005)
- Industrial collaborations
- International collaborations



Graduate Students

- Arati Baliga
- Aniruddha Bohra
- Vivek Pathak
- Murali Rangarajan
- Nishkam Ravi
- Pravin Shankar
- Steve Smaldone
- Gang Xu



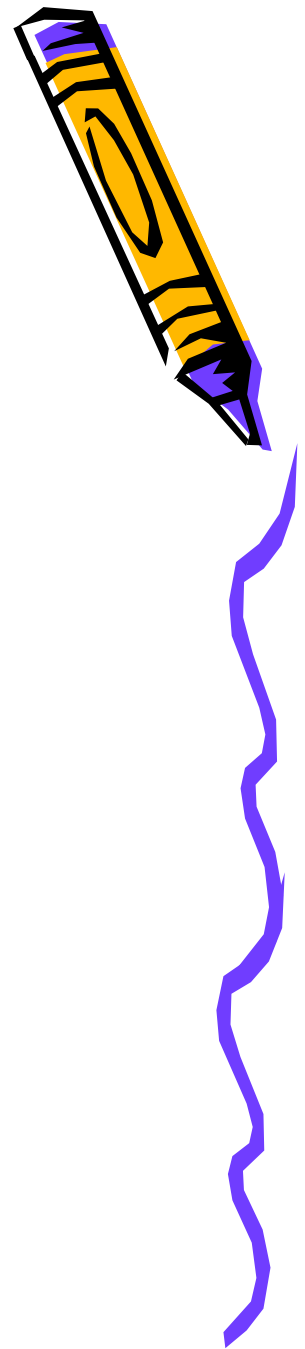
Rutgers University



- State University of New Jersey
- created in 1766 as the Queen's College
- Close to 50,000 students
- Three campuses: New Brunswick, Newark, Camden
- Located on the Northeastern Corridor, about one hour south of New York City

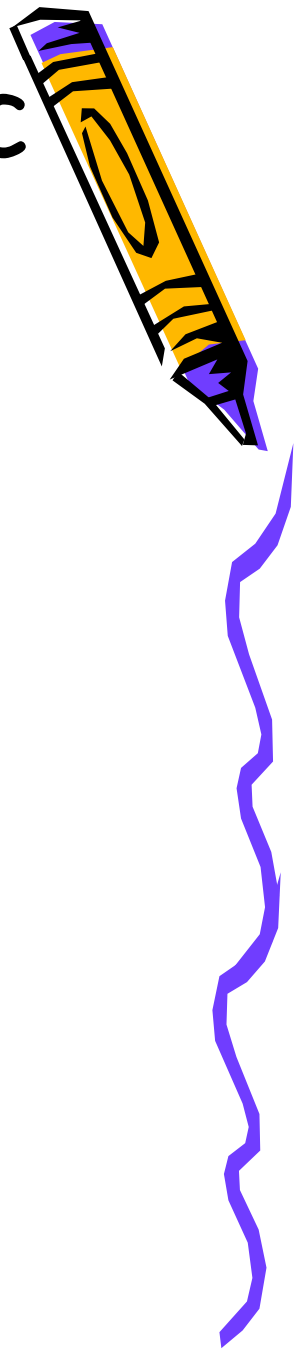
Department of Computer Science at Rutgers

- Faculty: 40
- Graduate students: 200
- Undergraduate CS majors: 200
- Major strengths: Theory, Systems, Networking, Vision, Bioinformatics
- Center for Biomedical Engineering
- Research Labs: Prolang, Dataman, Dark, Panic, Disco Lab, etc



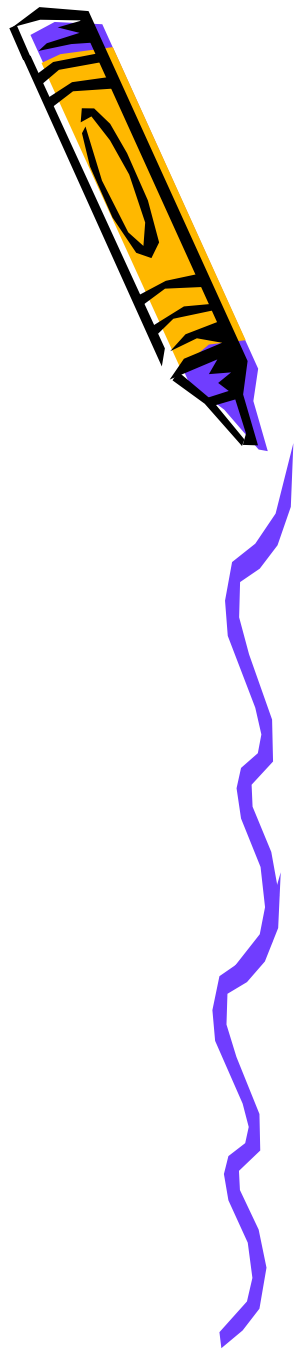
Our focus: Network-Centric Computing

- Networking is no longer a peripheral
- Progresses in computer systems are increasingly driven by networking
 - Storage networking
 - Peer-to-Peer systems
 - Pervasive computing
 - Security
 - Fault-tolerance
 - Recovery-Oriented Computing

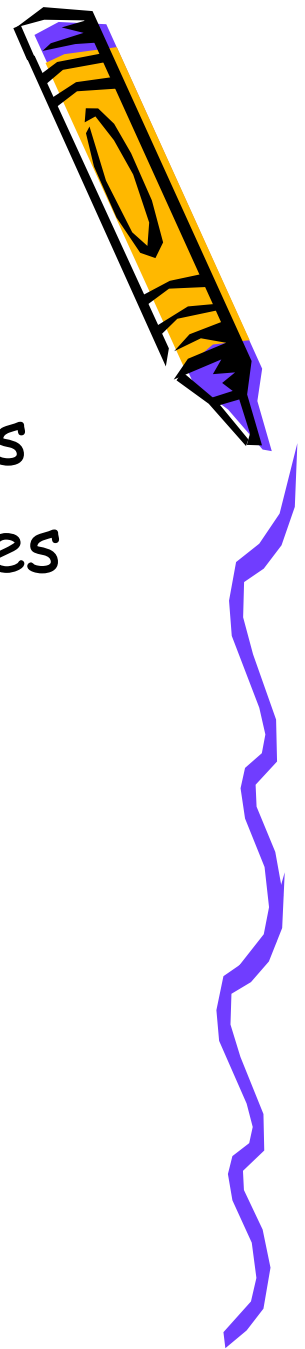


Past Projects

- Split-OS
 - TCP-Servers
 - Storage Networking using RDMA
- High-availability
 - Fault-tolerant Software DSM
 - Migratory-TCP
 - Service Continuations
- Outdoor Distributed Computing
 - Smart Messages
 - Spatial Programming
 - Pervasive computing using Smart Phones



Split-OS



- Servers as cluster of intelligent devices
- OS distributed across intelligent devices
- TCP- Servers executed on
 - dedicated intelligent NICs
 - dedicated processors in an SMP
- Direct device-to-device communication

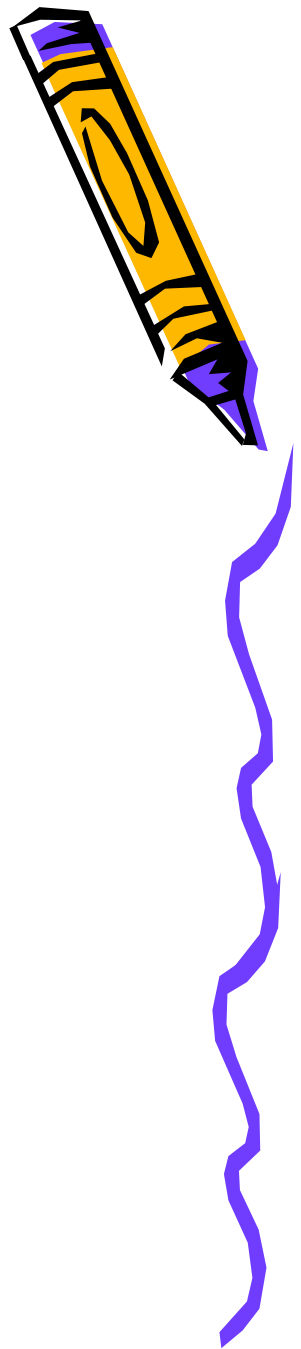
Memory-to-Memory Communication



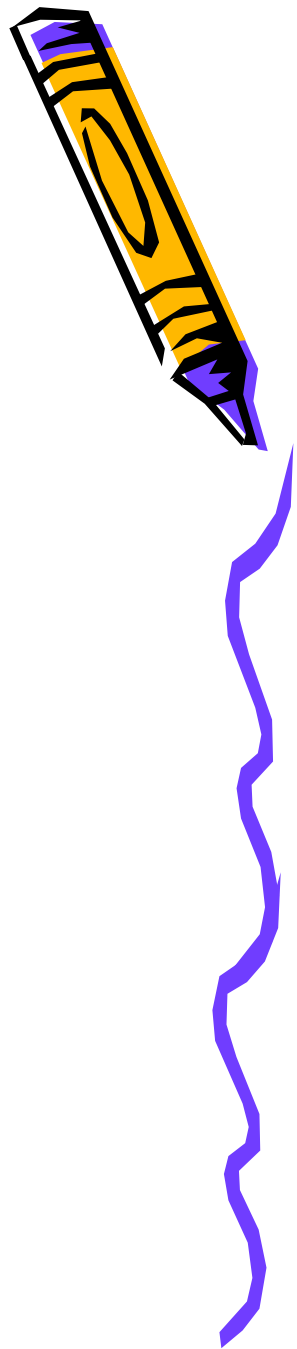
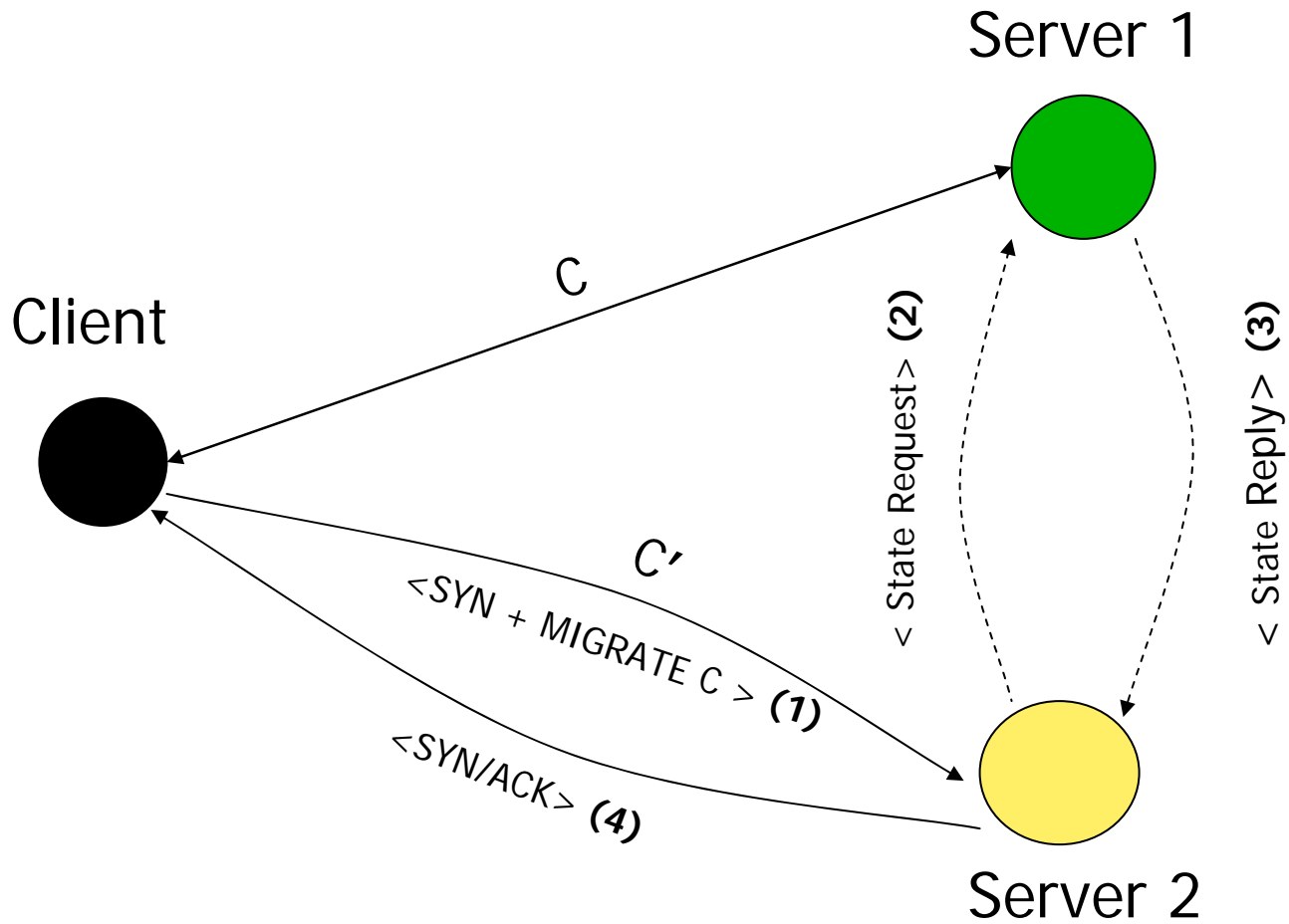
- Remote DMA (Read/Write)
 - read and write operations performed on registered remote memory without remote CPU involvement
- Standards: VIA, Infiniband, DAFS
- Started as university projects to improve performance
 - reduce cluster communication latency and overhead
 - improve performance of software DSM
- RDMA can be also used for remote state recovery from a failed system (backdoor idea)

Storage Networking Dilemma

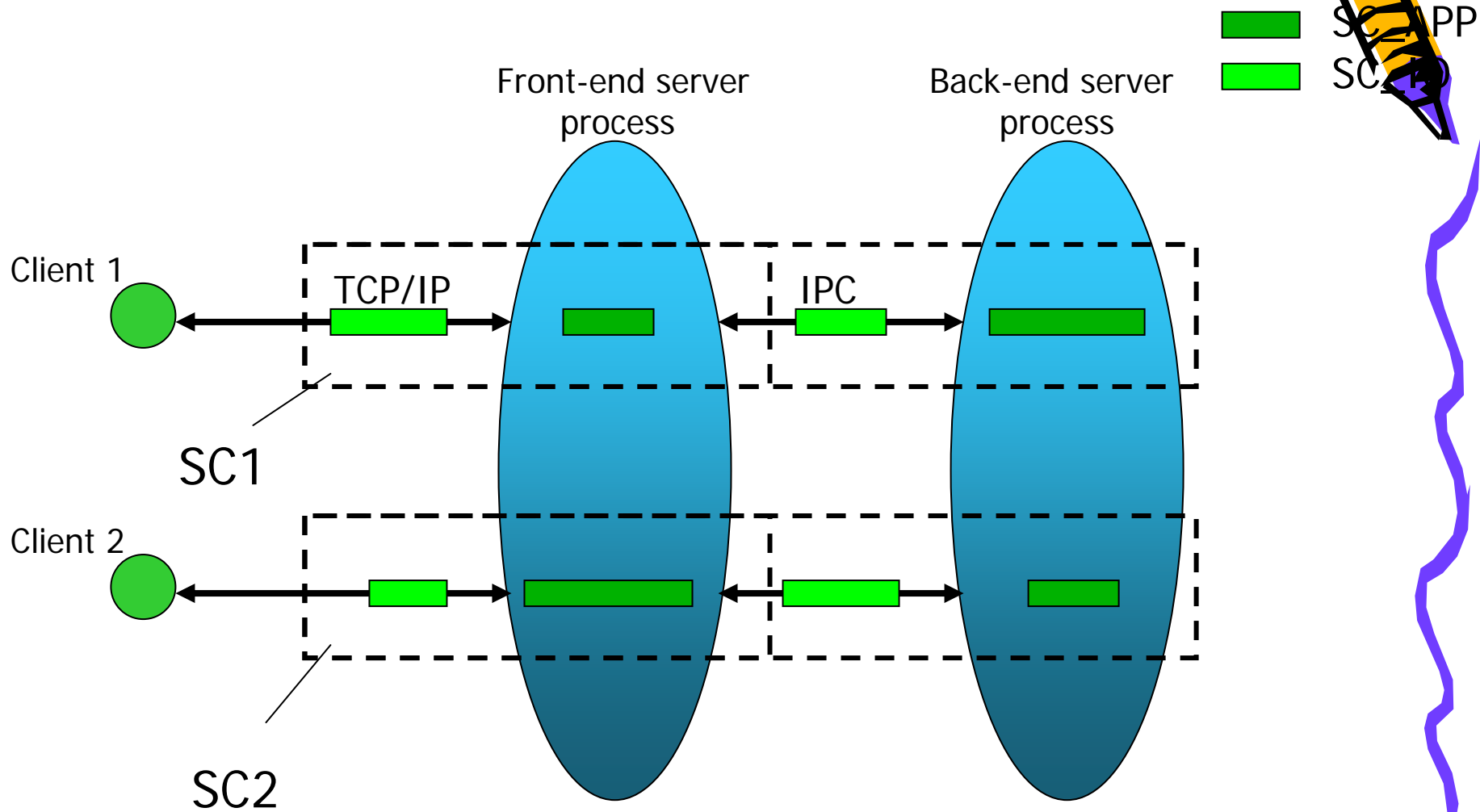
- TCP/IP overhead is unnecessarily high
- Two solutions:
 - Lightweight Transport (RDMA)
 - TCP Offloading (TOE, TCP-Servers)
- Two targets
 - File level protocols: NFS
 - Block level protocols: SCSI
- Ongoing project supported by Network Appliance Inc.



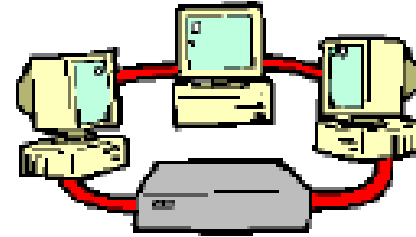
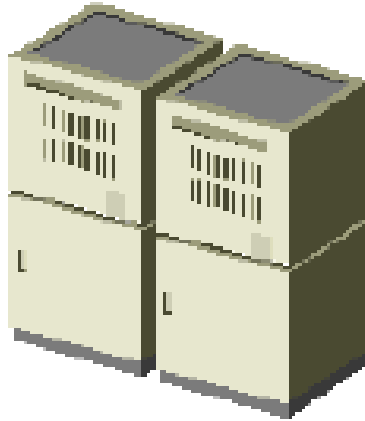
Migratory TCP (M-TCP)



Service Continuation



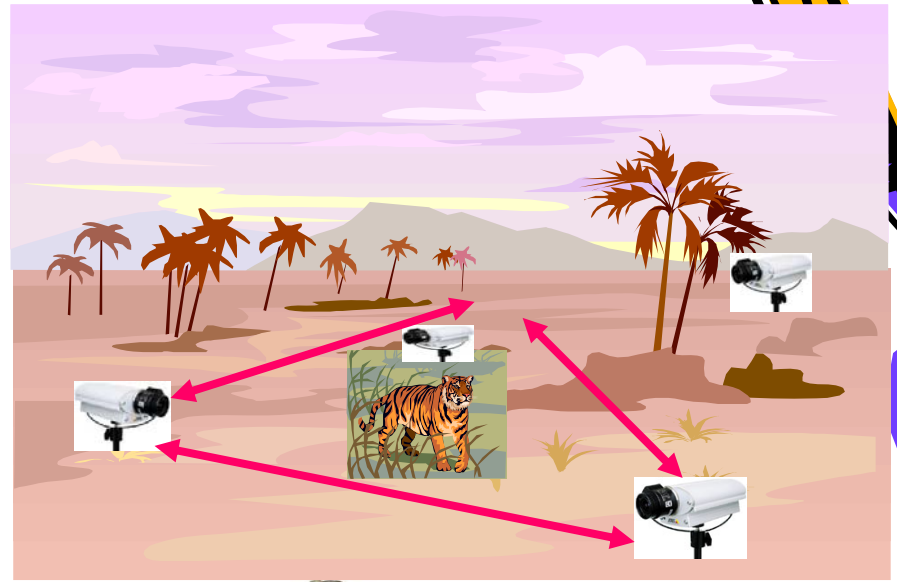
"Indoor" Distributed Computing



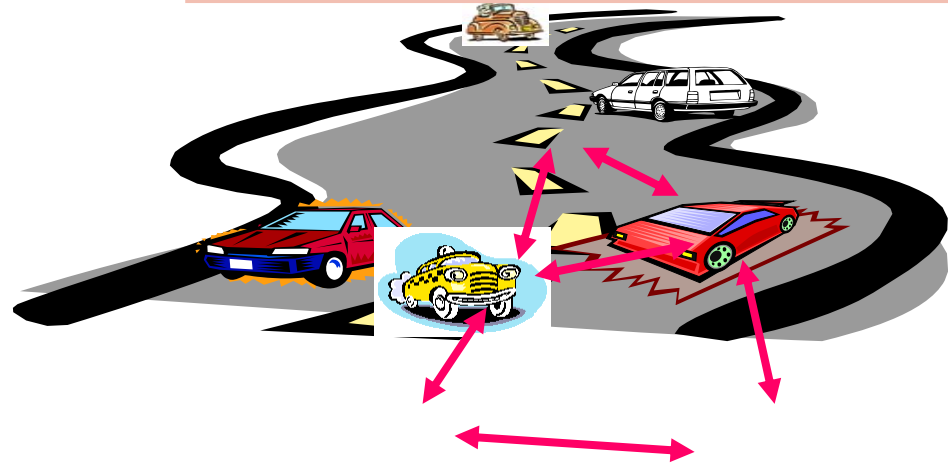
- Computing distributed for performance or fault tolerance
- Computing nodes are computationally equivalent
- Configuration is stable (failures are exceptions)
- Inter-node communication is robust and has acceptable delays
- Relatively easy to program

Computers "Move" Outdoors

Distributed object tracking over a large geographical area



Cars collaborating for a safer and more fluent traffic



Outdoor Application

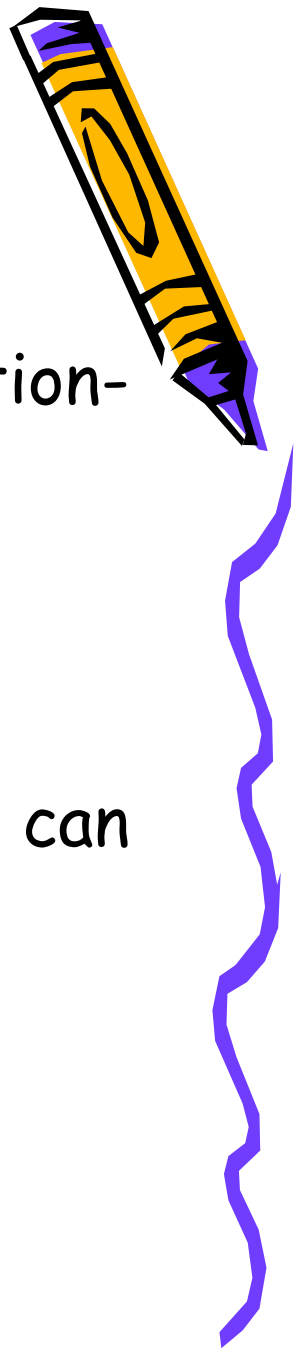


Mobile sprinkler
with temperature
sensor

Hot spot

- "Water the hottest three spots of the Left Hill"
- Configuration and environment changes continuously
 - sprinklers move, may disappear or go out of reach
 - temperature changes

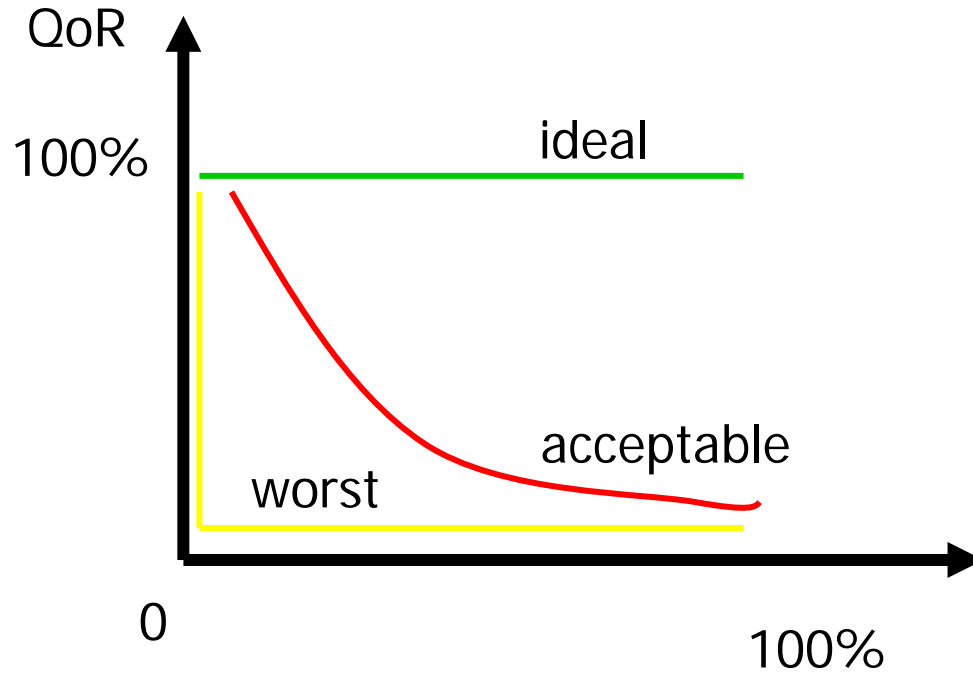
Outdoors is Different



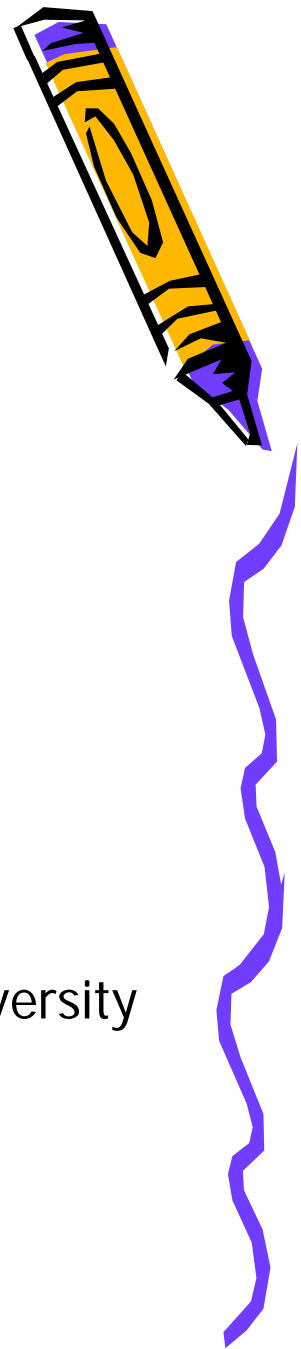
- Computation is distributed because it is location-dependent
- Embedded nodes are mobile and volatile
- Networking is ad-hoc and wireless
- End-to-end is hard/impossible to achieve
- Configuration, node properties, even weather can change during outdoor program execution

- How to program outdoors?
- How to support "reliable" execution?
- How to control the quality of the result?

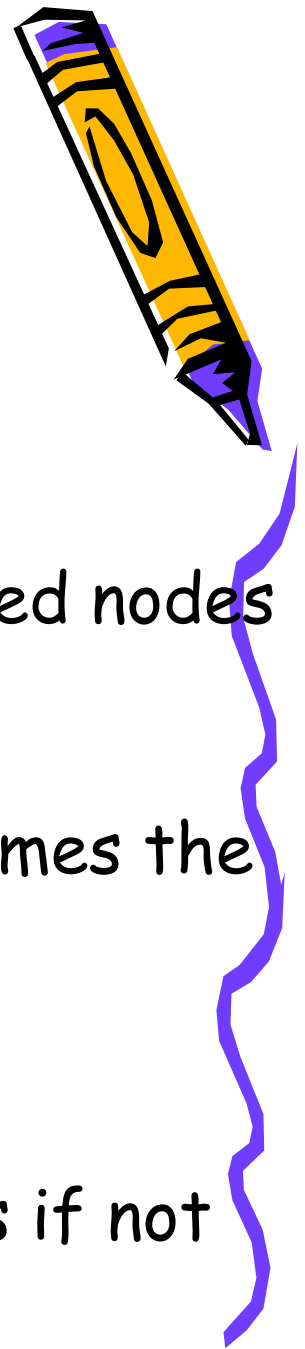
Quality of Result



Outdoors Adversity

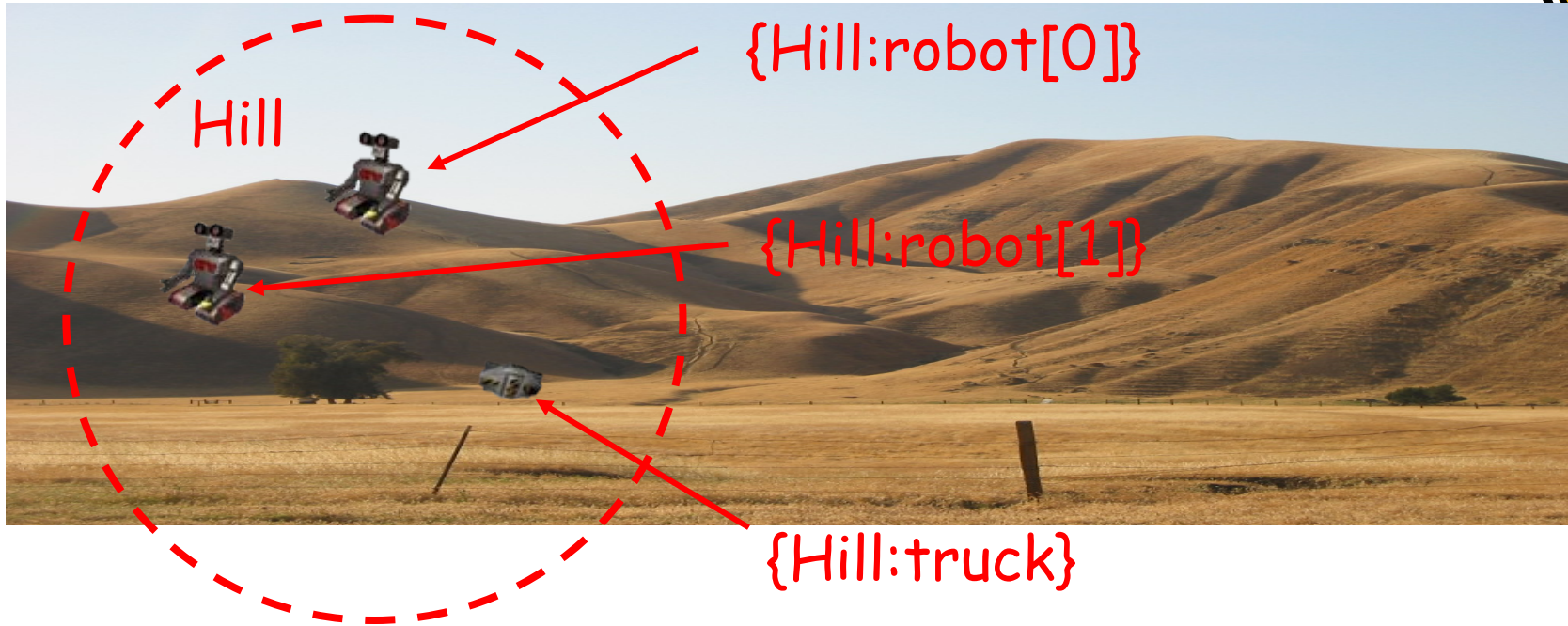


Spatial Programming Idea



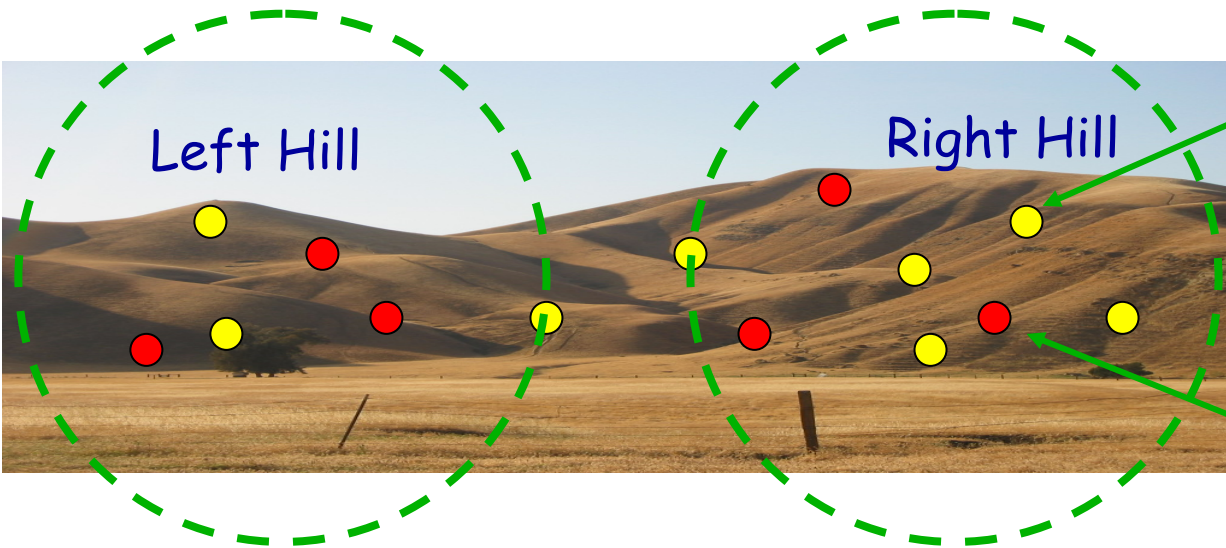
- **Space regions**: virtual address spaces mapped physical space
- **Spatial references**: virtual names for embedded nodes located in a particular space region
- **Reference consistency**: a spatial reference names the same embedded node throughout a program's execution
- **Access timeout**: spatial reference access fails if not completed before timeout

Spatial References



- `{Hill:truck}`, `{Hill:robot[0]}`, `{Hill:robot[1]}` are spatial references that name a truck and two robots located in **Hill** virtual space
- Defined as `{space:property}` pairs
- Indexes distinguish similar systems located in the same space

Spatial Programming Example



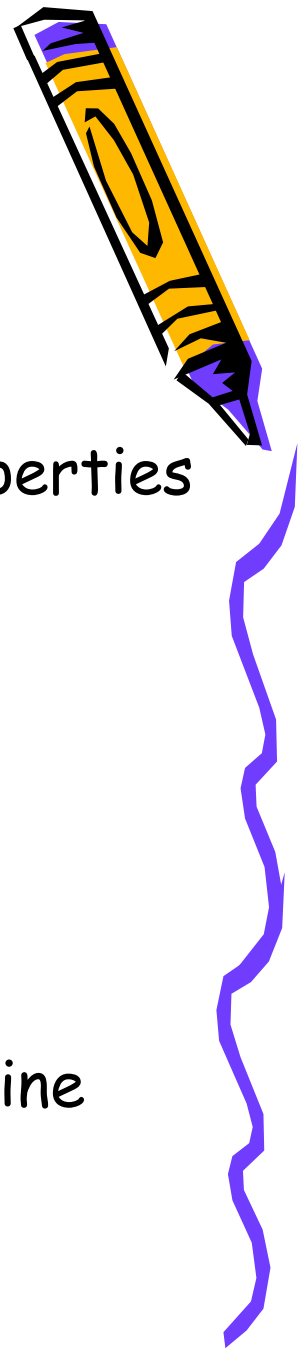
Mobile sprinkler
with temperature
sensor

Hot spot

“Water the hottest spot on the Left Hill”

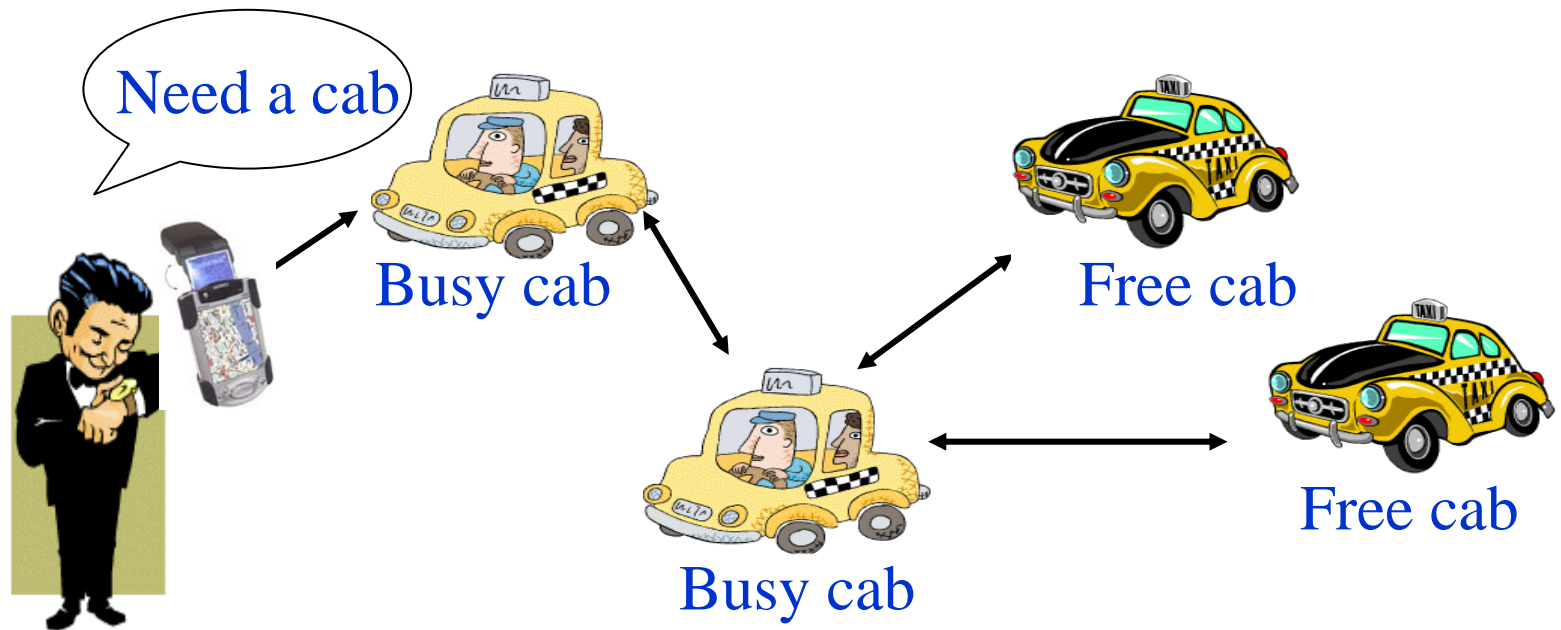
```
for(i=0;i<1000;i++)  
  try{  
    if ({Left_Hill:Hot[i], timeout}.temp > Max_temp)  
      Max_temp = {Left_Hill:Hot[i], timeout}.temp;  
      Max_id = i;  
    }catch(TimeoutException e)  
    break;  
  {Left_Hill:Hot[Max_id]}.water = ON;
```

Smart Messages



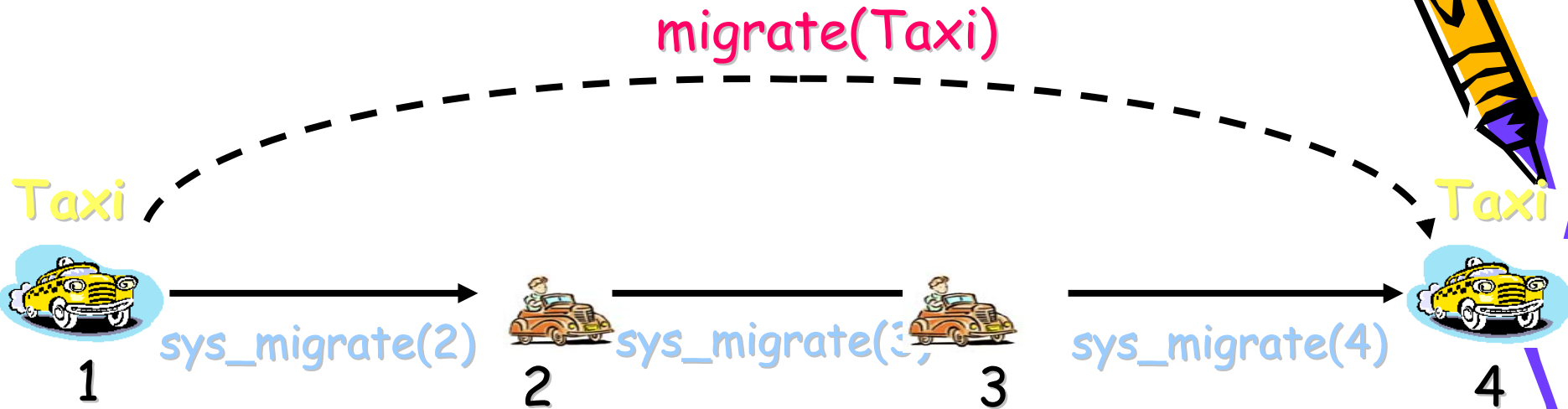
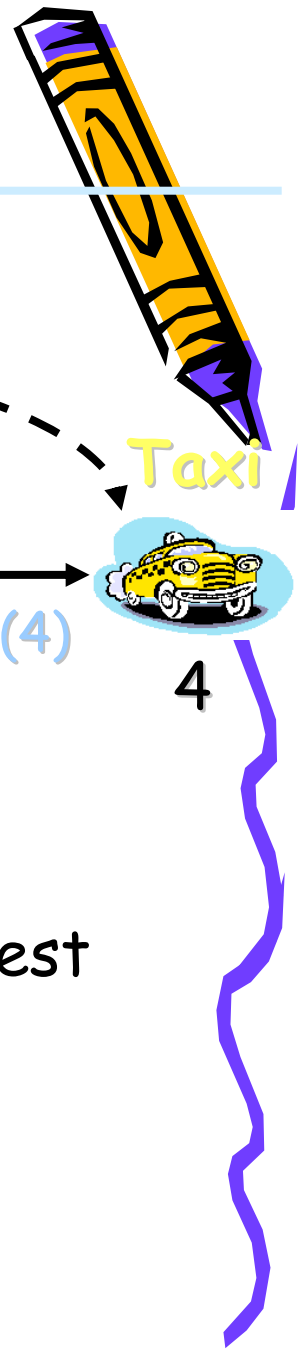
- **Smart Message (SM)**
 - User-defined migratory application
 - Executes on **nodes of interest** named by properties
 - Migrates between nodes of interest
 - Migration is lightweight
- **Self-Routing**
 - SM-specific routing used for SM migration
 - Executed on every node on the path
- **Cooperative Embedded Nodes**
 - Execution environment for SM: Virtual Machine
 - Local Memory accessible to SM: Tag Space
 - Do not provide routing

EZCab



- A taxi booking protocol over ad-hoc networks

Migration



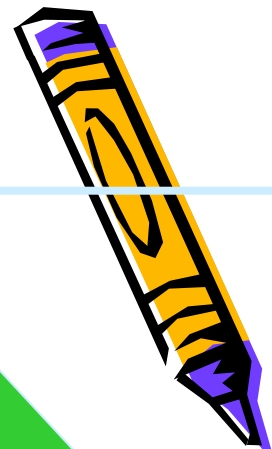
■ `migrate()`

- migrates application to next node of interest
- implements a self-routing algorithm

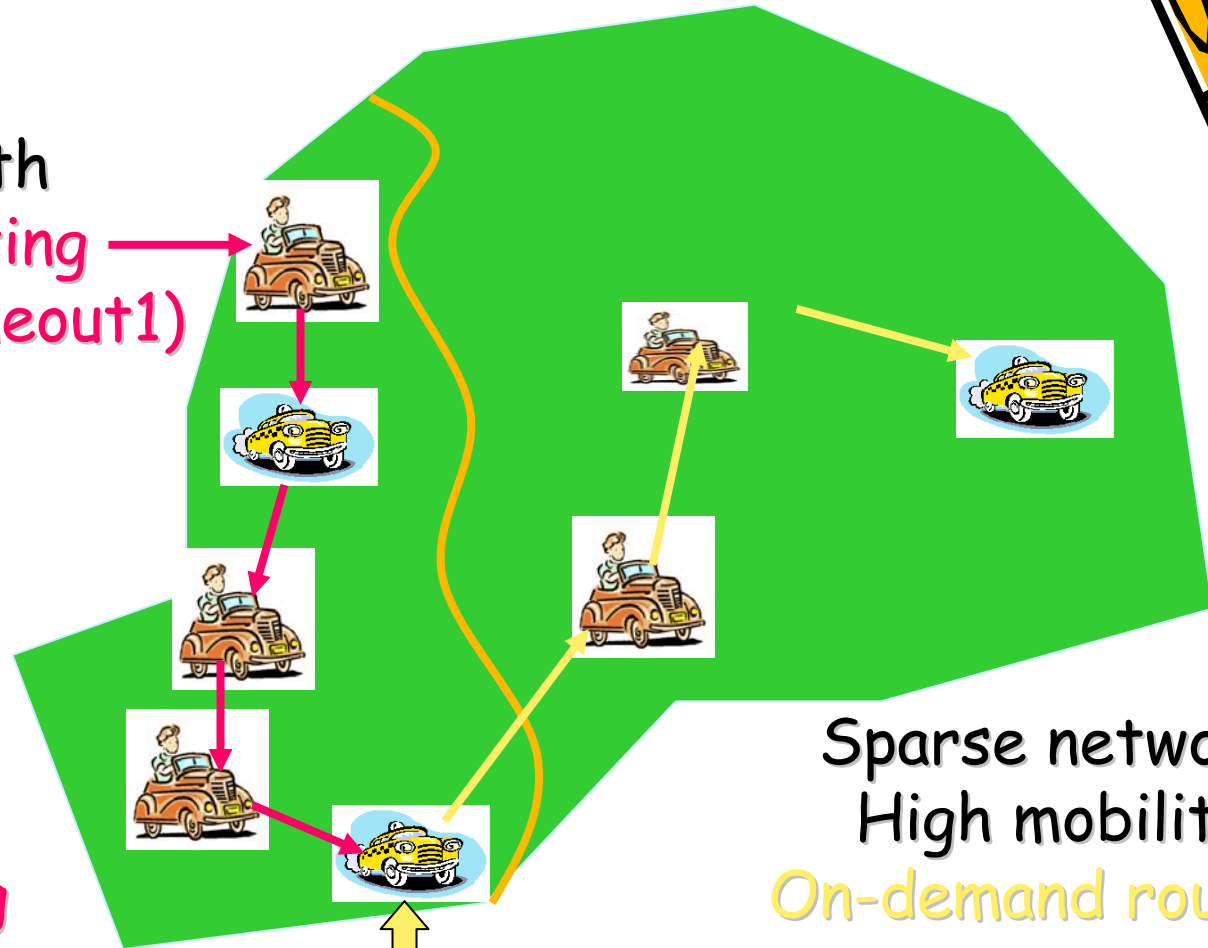
- `sys_migrate()`

- one hop migration

Dynamic Change of Routing



SM starts with
proactive routing
`migrate(Taxi, timeout1)`

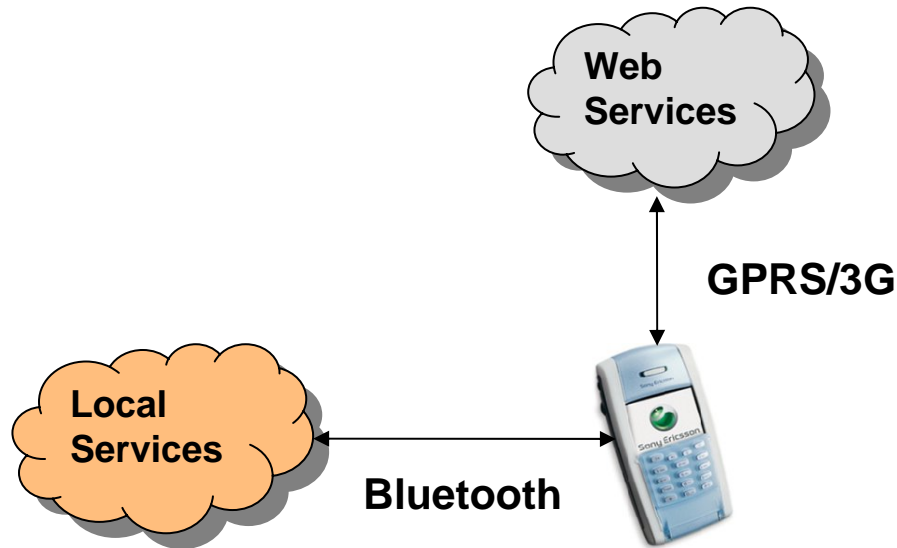


Dense network
Low mobility
Proactive routing

Sparse network
High mobility
On-demand routing

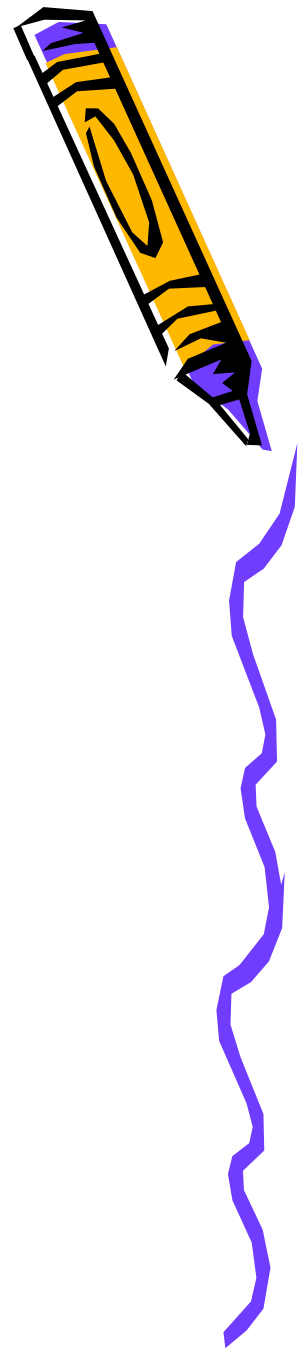
`migrate` timeouts
SM changes routing to on-demand
`migrate(Taxi, timeout2)`

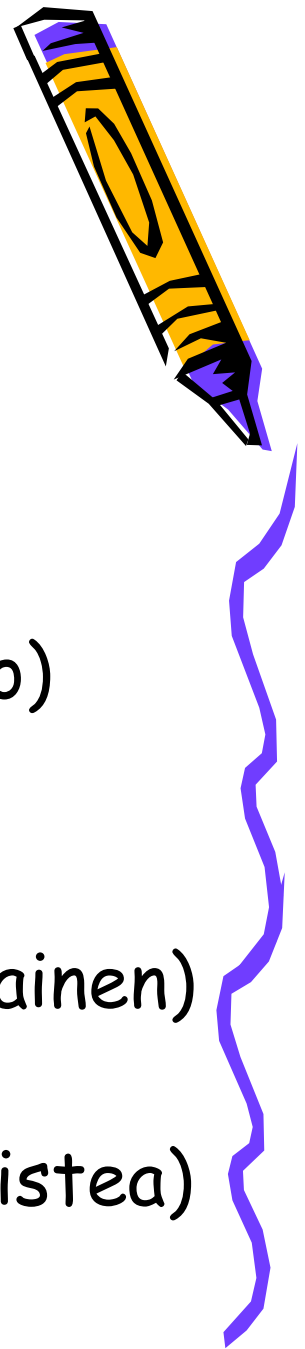
Pervasive Computing using Smart Phones



- Smart phones will “empty” our pocket: replace keys, money, business cards, etc
- Protocols for service discovery, interaction and payment

Student uses a smart phone to open the door to DiscoLab

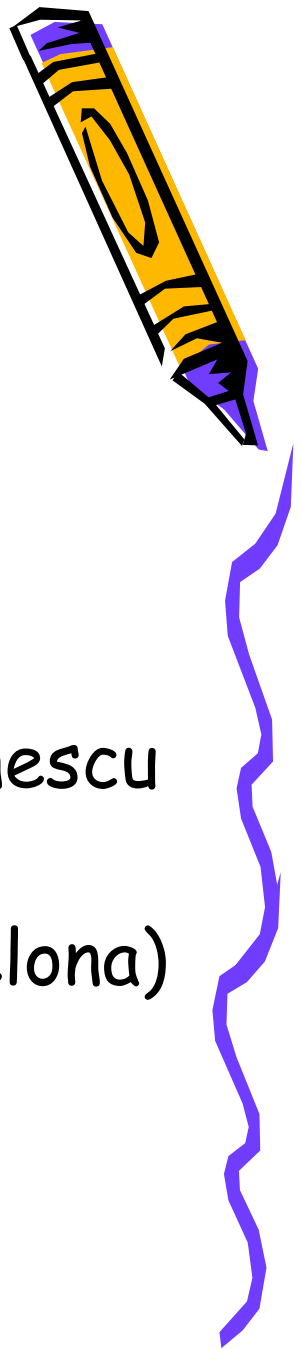




International Collaborations

- Existing
 - Paris Project (INRIA, C. Morin)
 - HiPEAC (UPC, Barcelona, Spain, M. Valero)
- New
 - LIP6 (Paris 6, F. Kordon)
 - University of Helsinki, Finland (K. Raatikainen)
 - University of Cyprus (M. Dikaiakos)
 - Technical University of Bucharest (V. Cristea)

Disco Lab International Visitors



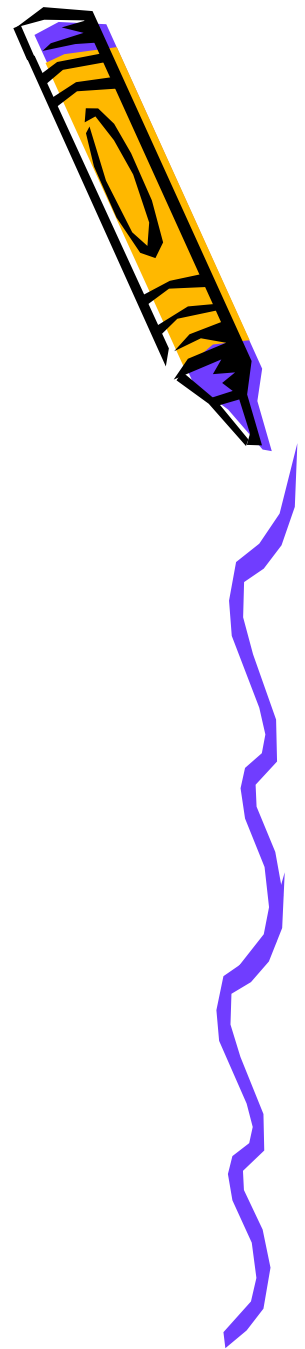
- 2003: P. Gallard (INRIA/IRISA)
- 2004: O. Riva (Univ Helsinki)
- 2005: M. Dikaiakos (Univ Cyprus)
- 2005: C. Gorgorin, R. Diaconescu, V. Gradinescu (Tech Univ, Bucharest)
- 2005: Carlos Villavieja Prados (UPC, Barcelona)

Current Projects



- **Backdoor**: healing computer systems remotely using dedicated hardware or virtual machines
- **E-Road**: protocol stack for vehicular ad-hoc network
- **Security-related projects**
 - Privacy control in location-based services
 - Intrusion detection and containment
 - Byzantine fault-tolerant authentication in P2P
 - Trusted network services
- **File system virtualization** using file system ad-hoc federations

Disco Lab talks



- Steve Smaldone: Backdoors
- Arati Baliga: Intrusion Detection
- Guest: Cristian Tapus (Caltech):
Recovery through Speculations
- Nishkam Ravi: Privacy control
- Pravin Shankar: E-Road

DiscoLab students on Route 1

