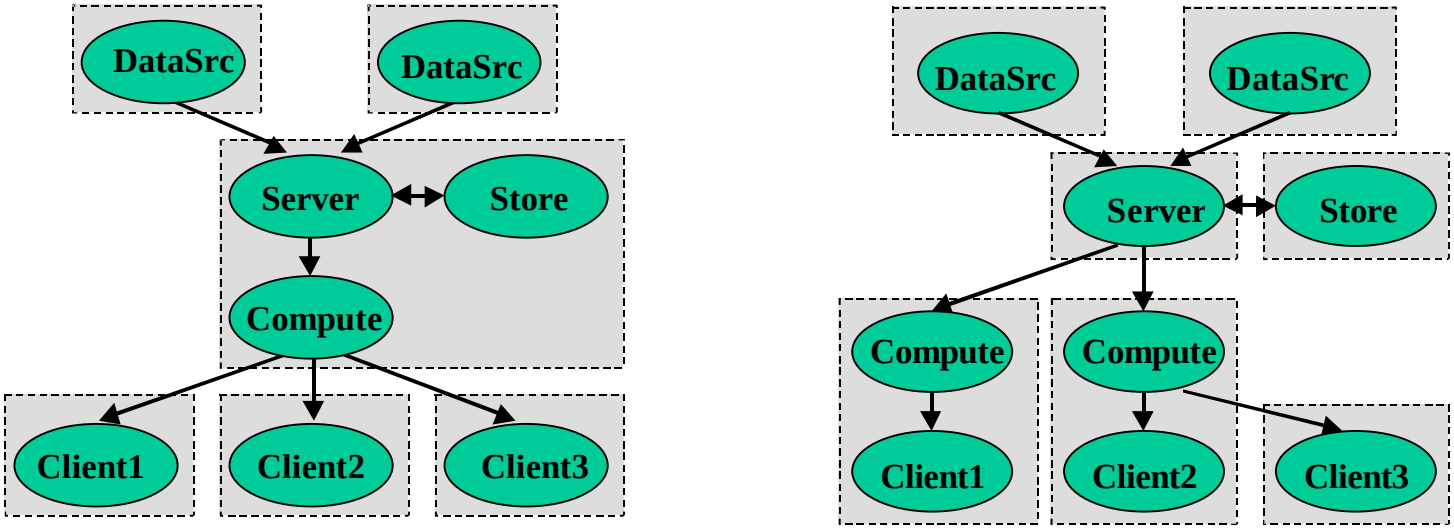
DACIA:
A Mobile Component Framework for Building Adaptive Distributed Applications

Radu Litiu and Atul Prakash
University of Michigan, EECS

Outline

- Motivation and Goals
- DACIA Architecture
- Performance Measurements
- Related Work
- Conclusions and Future Work

Need for Adaptation - SPARC Collaboratory



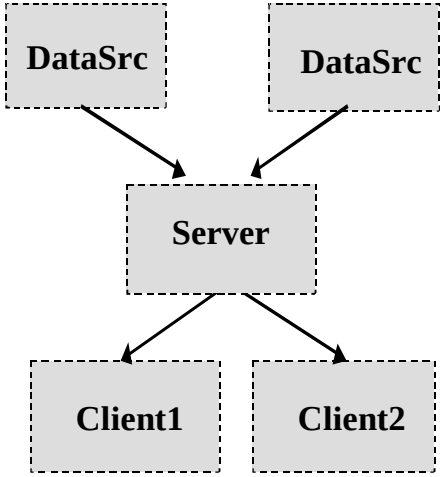
An application is a graph of connected components.

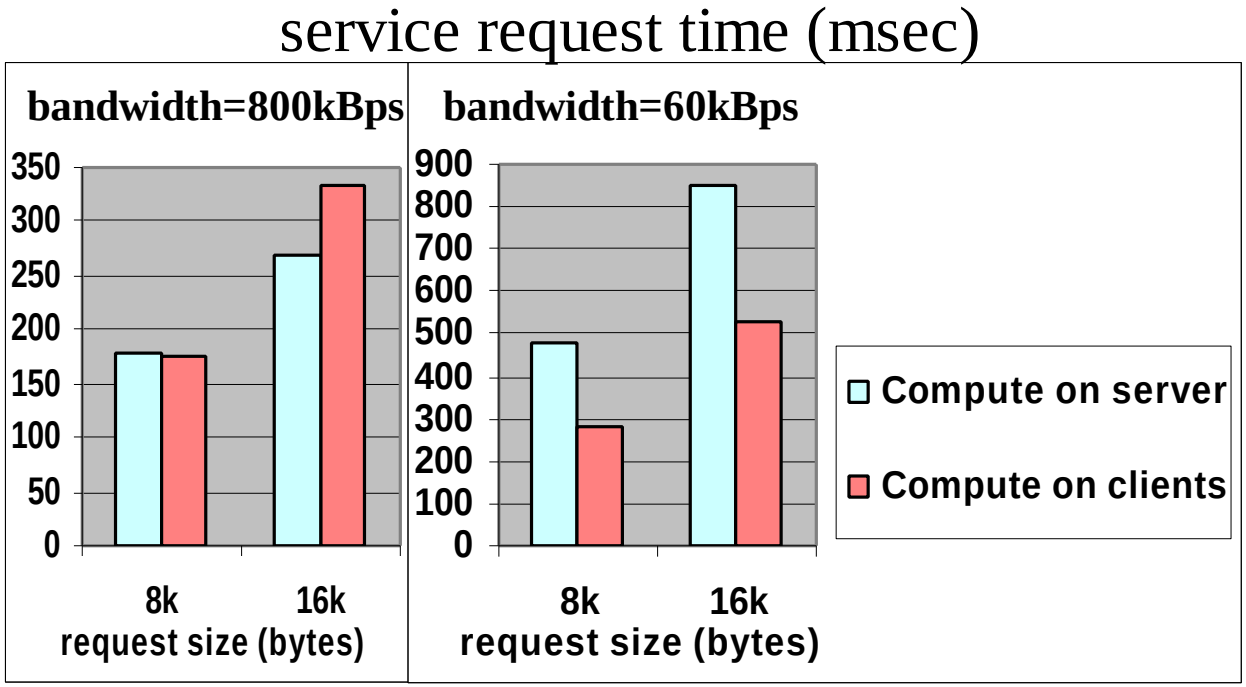
Possible changes:

- Execute the computation on the client machine
- Store computed images instead of raw data
- Add/remove modules

Why Reconfiguration?

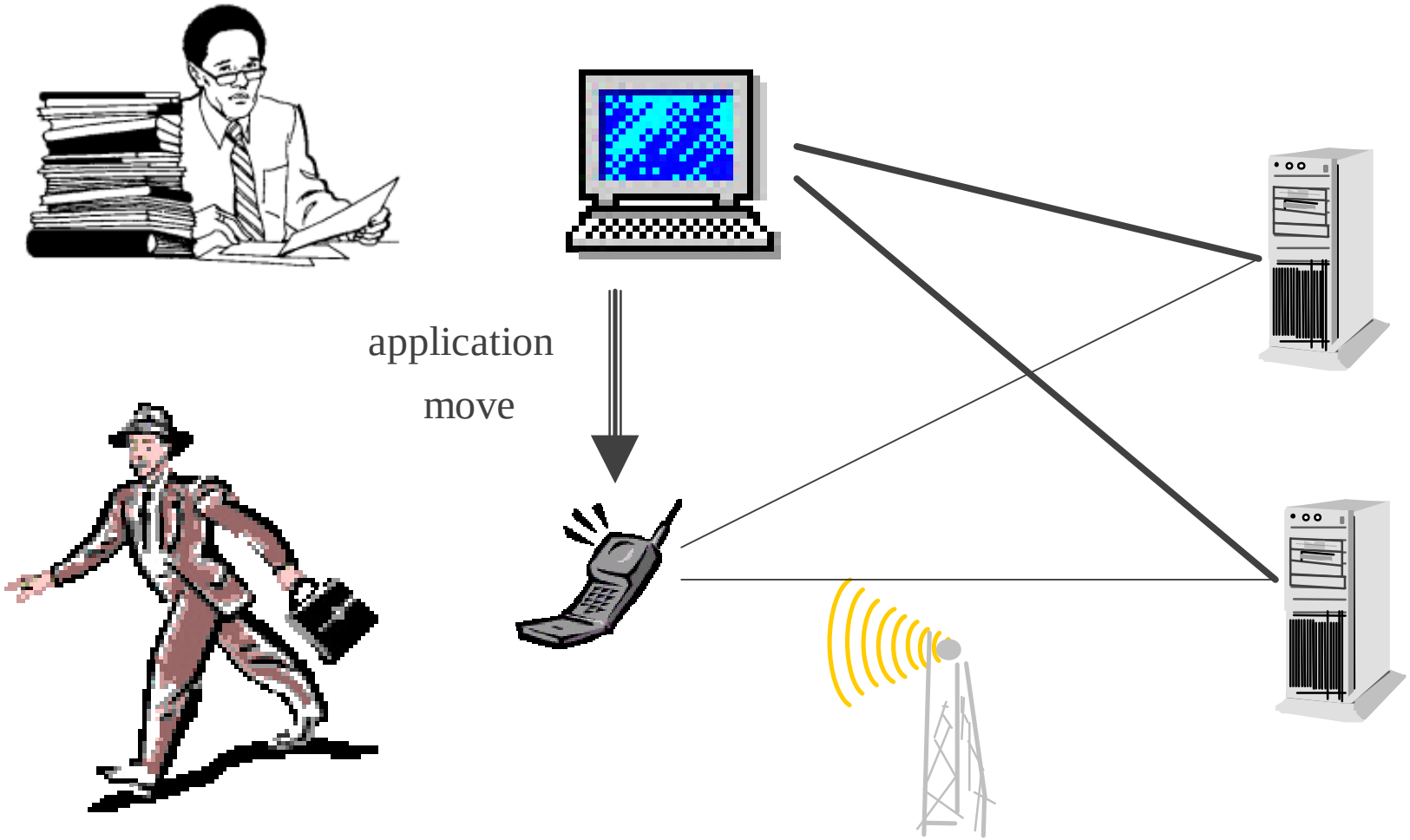
- Environment: Pentium II 200, Ultra SPARC 1
- raw data size / computed image size = 1/2
- compute time = 5 msec/kB (fast machine)
  15 msec/kB (slow machine)





- Adaptability and reconfiguration can be useful

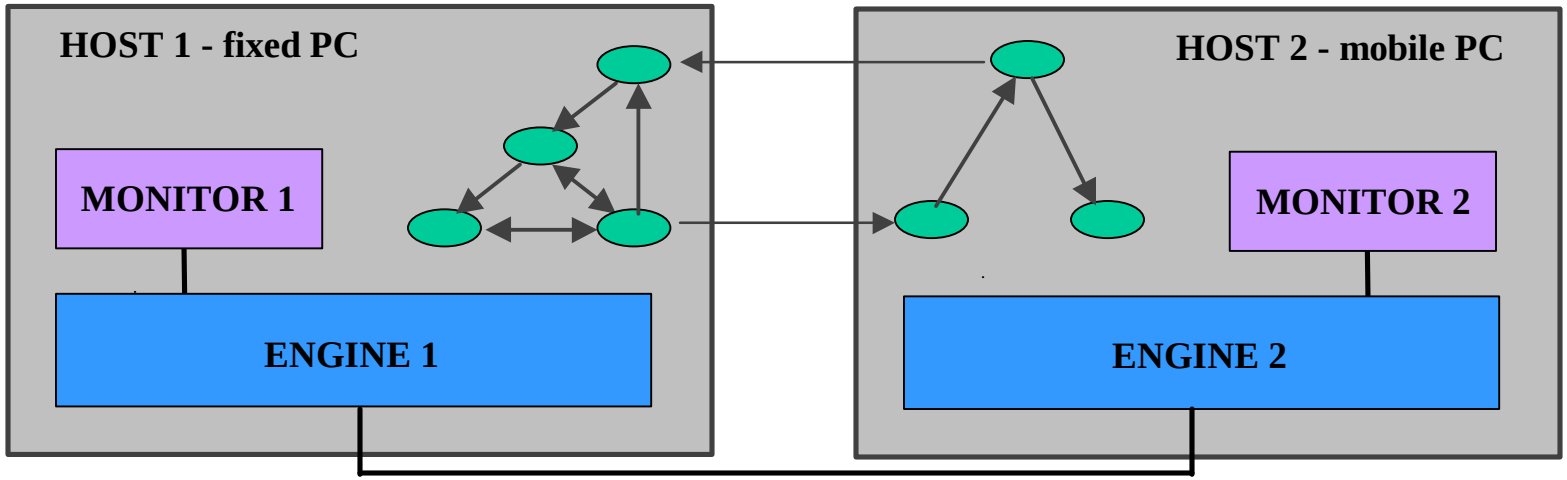
Mobility



Design Goals

- Adapt to variability
- Runtime reconfiguration
- Application and user mobility
- Persistent connectivity between components
- Low communication overhead
- Ease of use

DACIA* Architecture



Engine (mechanism)

- Communicate between hosts
- Manage connections between components
- Relocate components
- Reconfigure the application

Monitor (policy)

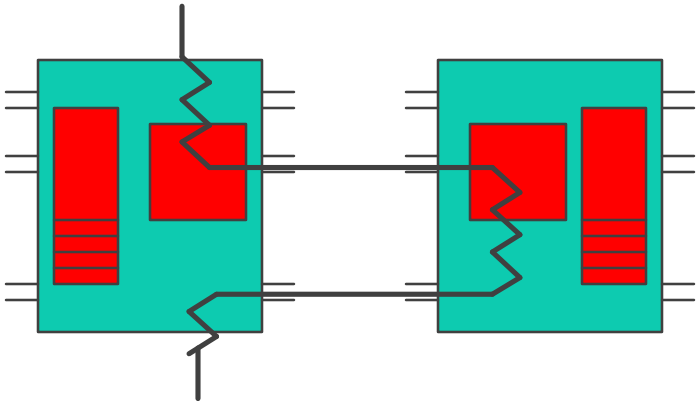
- Monitor performance
- Make reconfiguration decisions
- Implement application-specific reconfiguration policies

PROCs : Basic Model

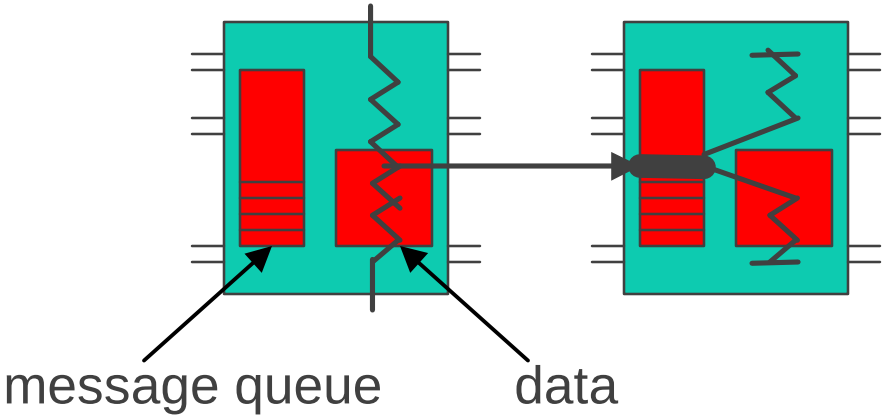
PROC - Processing and Routing Component

- Communication through ports
- Key goal: low communication costs

synchronous communication



asynchronous communication



Communication Performance

- Environment: Pentium II 200, Ultra SPARC 1, 10 Mbps LAN
- Latencies (μsec) for inter-PROC communication and raw TCP

message size (bytes)	local PROCs synchronous	local PROCs asynchronous	local procedure call	local TCP	Remote PROCs	Remote TCP
0	6.6	44	6.4	370	2040	990
1000	6.6	44	6.4	400	3900	2600

- Throughput (message size = 1-5 kB)
  - DACIA: 4.78-5.33 Mbps
  - TCP (Java): 5.35-6.61 Mbps

Component Mobility

- Transfer the PROC's state
  - data
  - message queue
- Implicit/explicit state capture
- Movement at well-defined times
- Cost of PROC movement - 121 msec (size = 788 bytes)
  - Java serialization cost
- Component mobility more effective for long-term environment changes

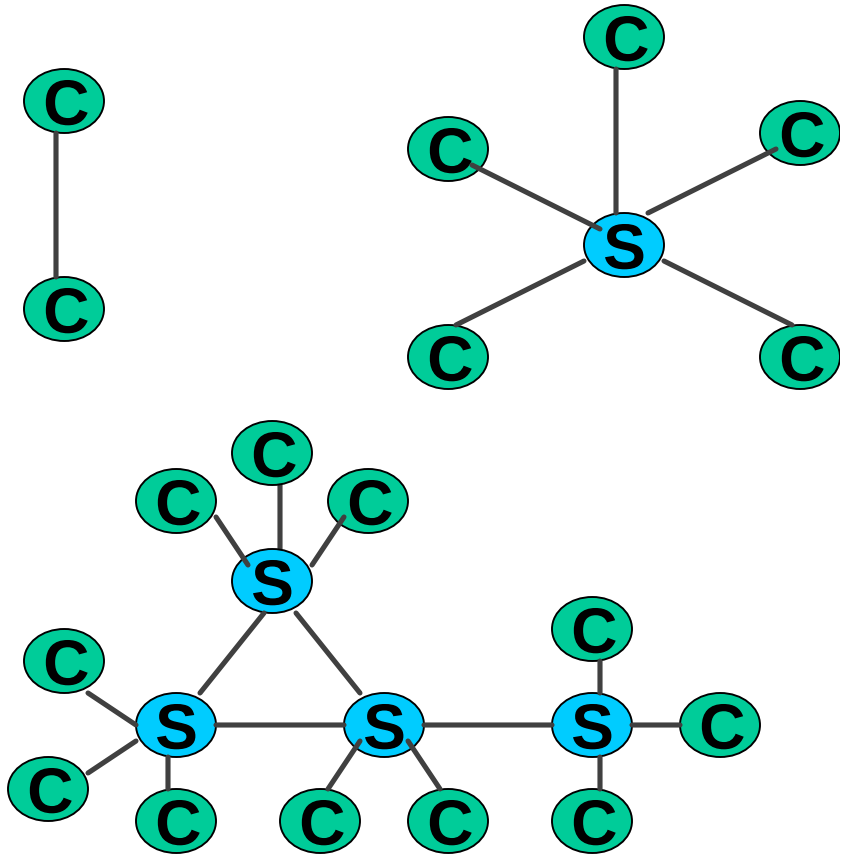
Connectivity

- Multiplex virtual connections between PROCs
  - Low cost to establish connections
  - Hide temporary network failures
- Persistent connectivity between moving PROCs
  - Messages buffered or forwarded
  - Dissemination of PROC location information

Dynamic Application Reconfiguration

- Change connections between components
- Change components' location
- Load new components

An adaptive application: multi-party communication



Reconfiguration Mechanisms

- Specialized monitors
- Command-line interface :
  - *connect [hostname] [portnumber]*
  - *connectProcs [sourceProcID] [sourcePortNo] [destProcID] [destPortNo]*
  - *disconnectProcs [sourceProcID] [sourcePortNo]*
  - *move [procID] [hostname]*
  - *start [procID]*
  - *startMonitor*

Related Work

- Distributed component architectures: CORBA, Globus, Darwin, Scout
- Code mobility & mobile agents: Telescript, Obliq, Sumatra, Tacoma, Aglets, FarGo
- Mobile environments: Rover, Daedalus/Barwan, GloMop
- Adaptive systems: Odyssey, Conductor

Conclusions

- DACIA - a framework for building adaptive distributed applications
- Dynamic reconfiguration can improve the performance of the application
- Low-cost connectivity
- Application and user mobility
- Persistent connectivity between mobile components

Current and Future Work

- Policies and algorithms for application reconfiguration
- Formalism for specifying components and composition rules
- Deployment and experimentation
- Security infrastructure

http://www.eecs.umich.edu/~radu