CSci 5105

Introduction to Distributed Systems

Administrivia, Intro

Today

- Introductions
- Administrivia
- Distributed Systems Introduction

Welcome to 5105!

- Me:
  - Jon Weissman
  - CS Professor: Distributed Systems at UMn since 1999
  - Call me “Jon”
- TA:
  - Kartik Ramkrishnan
- You
  - Highly motivated CS or ECE student

Logistics

- Lecture
  - T/Th 2:30 – 3:45, KH 3-111
- Jon’s office hrs
  - 1-2 T/Th, KH 4-225D
  - Also come by when door is open
  - Can email for appointment other times

- Kartik’s office hrs
  - TBD, KH 2-209

Introduction

- CSCi 5105 is a graduate class
  - strong upper-level undergraduates as well
- Expected background
  - CSCi 5103 (inside OS), CSCi 4061 (systems)
    - fluent in: virt memory, synch, concurrency, ...
  - Know how to edit, program, debug on preferably Linux systems
  - Strong programmer in C/C++ or Java
  - BUT can also program in the other
  - Can go off and figure stuff out from docs

Goals

- Expose you to the principles and practice of Distributed Systems
  - concepts, techniques, algorithms, systems
  - examine real distributed applications and systems that use the above
- Acquire basic distributed programming skills
- Learn how to analyze and assess conflicting issues and designs

Class Materials

Web site:

http://www-users.cselabs.umn.edu/classes/Spring2018/csci5105

- Submit on line via moodle
- Forum on line via moodle

Book: Distributed Systems Principles and Paradigms, Tannenbaum and Van Steen, 2nd edition 2007 (TVS), Papers on website, handouts

Code: on website

Lecture

- Presentation of key ideas based on book and/or paper readings
  - do these ahead of class!

- Discussion periods

Class Etiquitte

1. Be attentive in class
2. Talk occasionally (to us)
3. Do not distract me or your classmates
4. Disagreeing with me is par for the course

Grading

- In class midterm: 15%
- In class final: 15%
- 3 Programming projects: 45%
  - allow C/C++ or Java for some; others may mandate a single language
  - group of 2-3
  - **not Plan C projects!**
- 3 Written homeworks: 15%
- Participation: 10%
  - "make yourself known"

Policies

- Late work

  24 hrs with a 10% penalty

  afterwards, not accepted
- Work your interview schedule around my class
- Cheating? 0 tolerance.
- Re-grading window
  - 1 week from time graded work is returned
- Use of the Web
  - Without citation ... see cheating above

Topics

- Week 1: Distributed Architectures
- Week 2: Communication: RPC
- Week 3: Advanced Communication: MoM, MPI, multicast, streaming
- Week 4: Naming
- Week 5: Synchronization, Mutual Exclusion
- Week 6: Replication and Consistency
- Week 7: Fault Tolerance
- Week 8: Exam, TBD
- Week 9: Consensus

Topics (cont’d)

- Week 10: Classic Design, Distributed Scheduling
- Week 11: Distributed File Systems
- Week 12: Case Studies: Grids, P2P
- Week 13: Case Studies: Cloud, Data-Intensive-1
  Week 14: Case Studies: Data-Intensive-2, Security
- Week 15: “Cool” Techniques, Wrapup

Questions?

Let's Begin

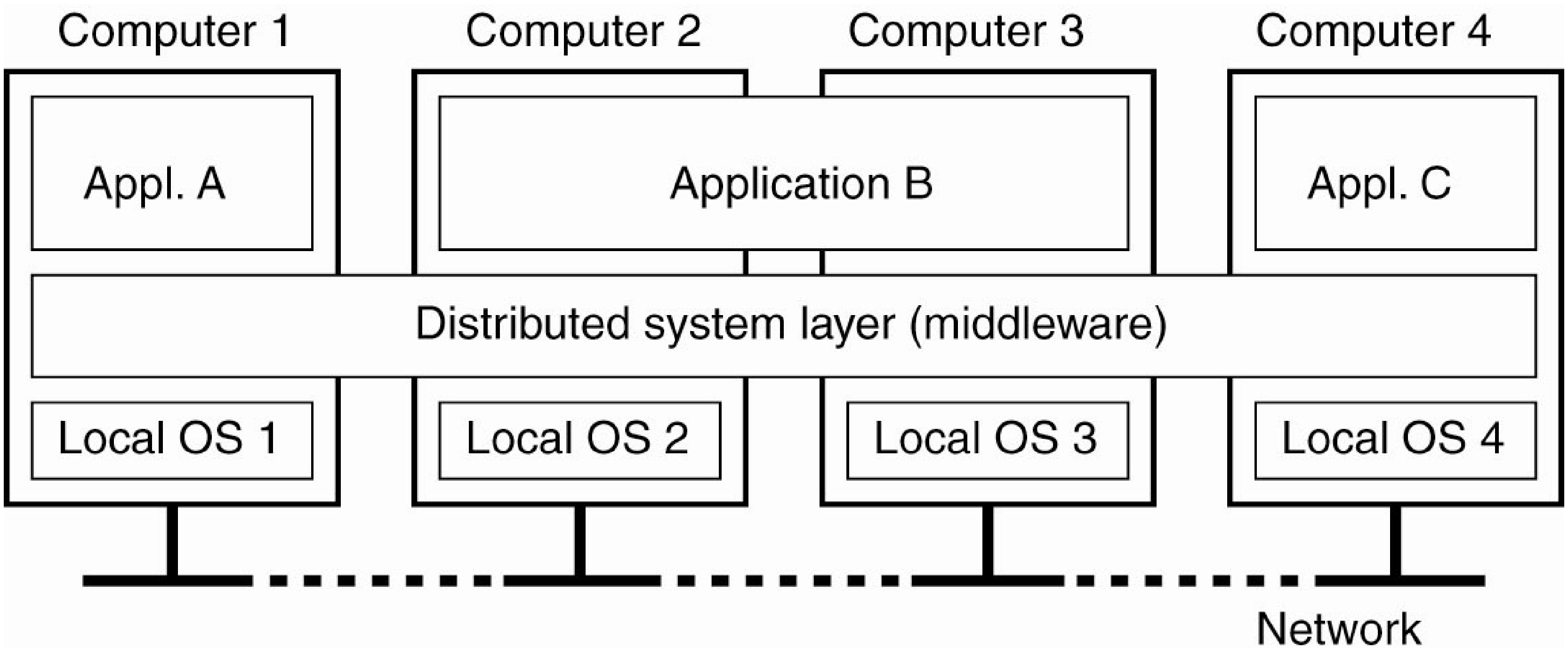
Distributed System Definition:

A collection of independent computers that appears to its users as a single coherent system

Examples?

Not Example?

Distributed System Middleware



Software that provides services needed to achieve "single coherent system" (buzzword: transparency)

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Openness

Can I build an equivalent implementation of a system: are protocols and dependent interfaces public?

e.g. I can implement TCP

Related to interoperability: can I replace a component of a system with my own?

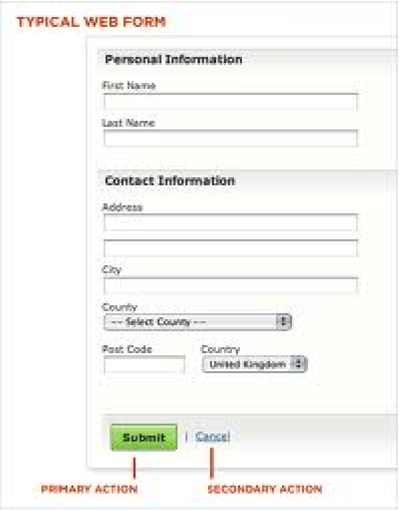
Scalability

- Important metric for distributed systems
- How do we want our systems to scale?
  - Size: easy to add resources, users, ...
  - Geographically
  - Management: across domains

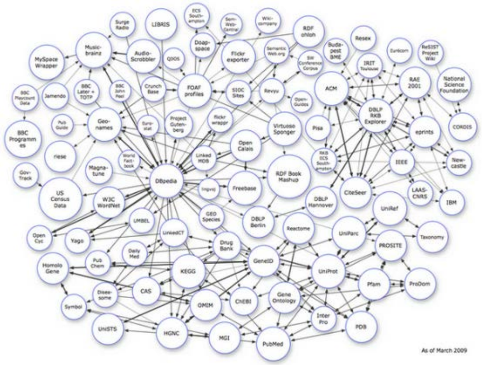
Scalability (cont'd)

- Obstacle?
  - Centralization (service, data, algorithm)
  - Centralization makes things easy
    - management, consistency, security

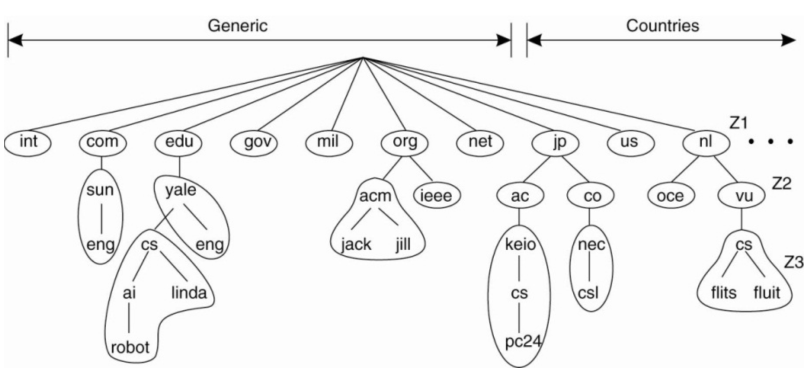
Scaling Examples

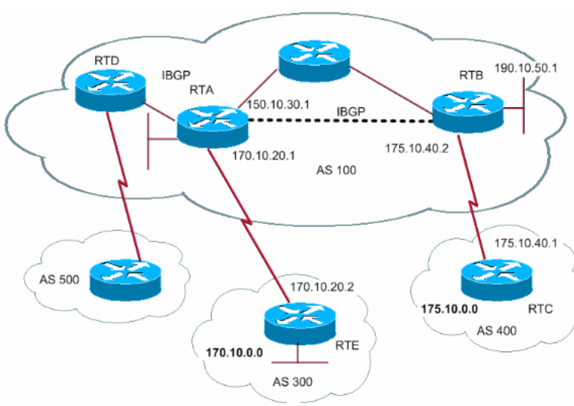


- Partitioning (~ service)
  - Local web form vs. server processing



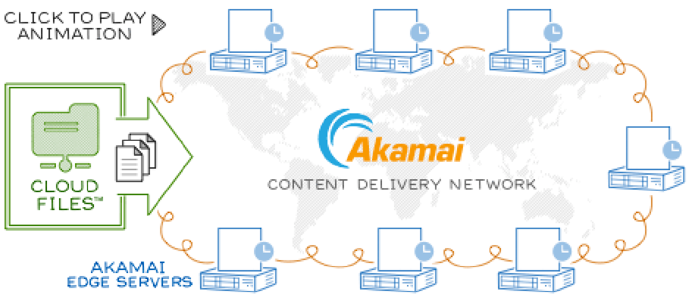
- Distribution (~ data)
  - Web
  - DNS, BGP routing (~ algorithm)





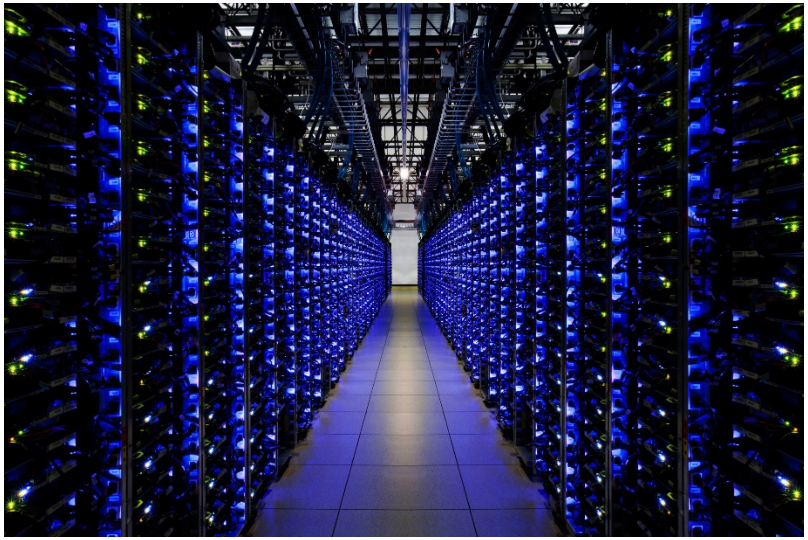
Scaling Examples (cont'd)

- Replication (~ service + data)
  - content-delivery network (CDN)

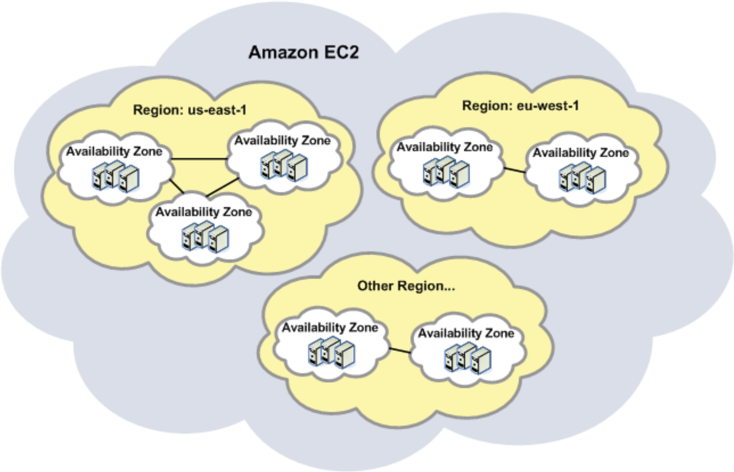


Types of Scaling

- Scale-up

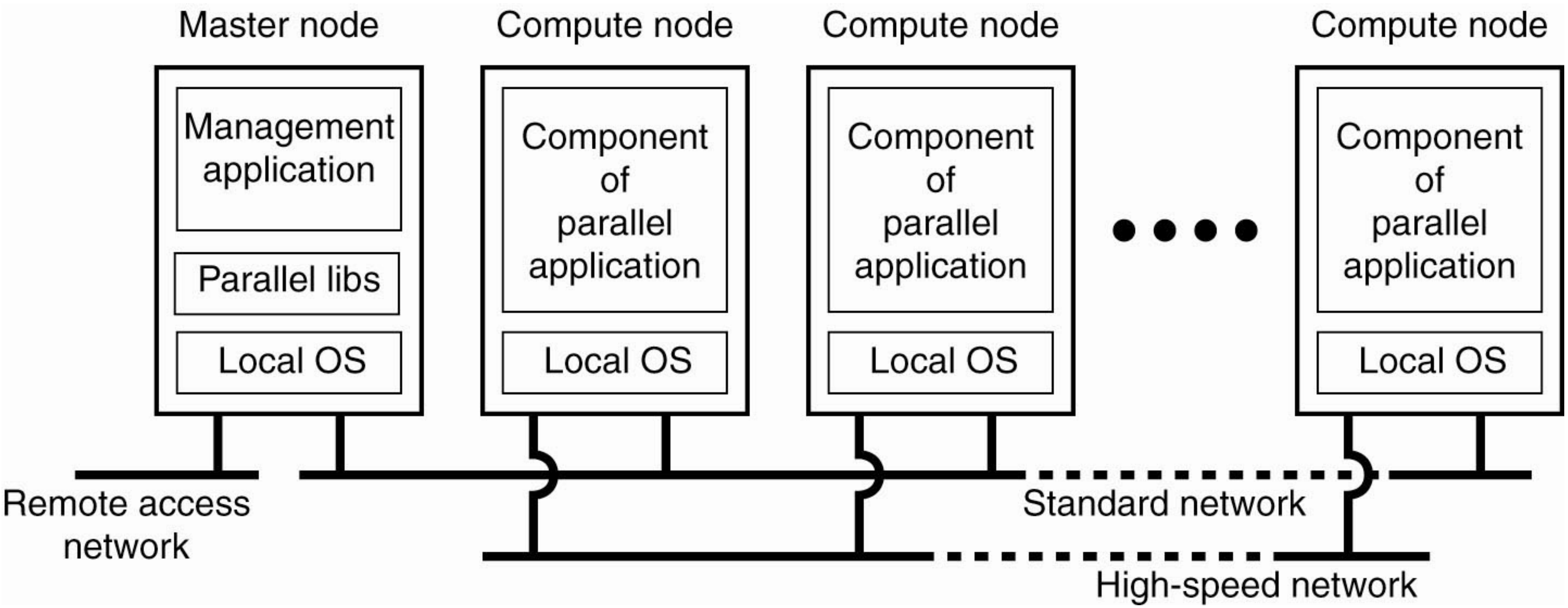


- Scale-out



Distributed Computing Systems

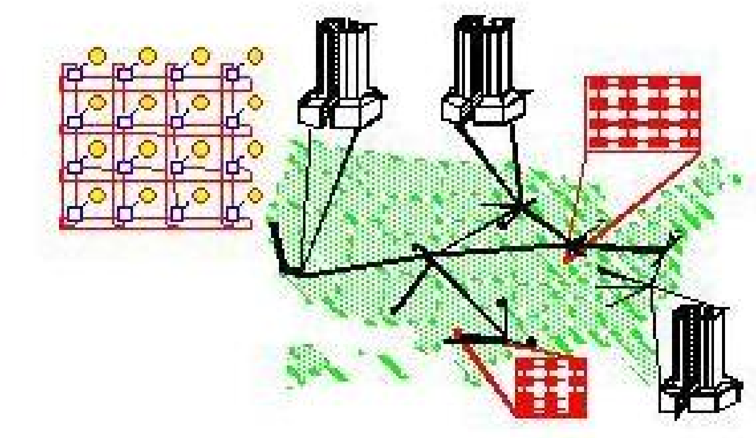
- Tightly-coupled: cluster



- Key issue: Massive clusters: failure is prevalent

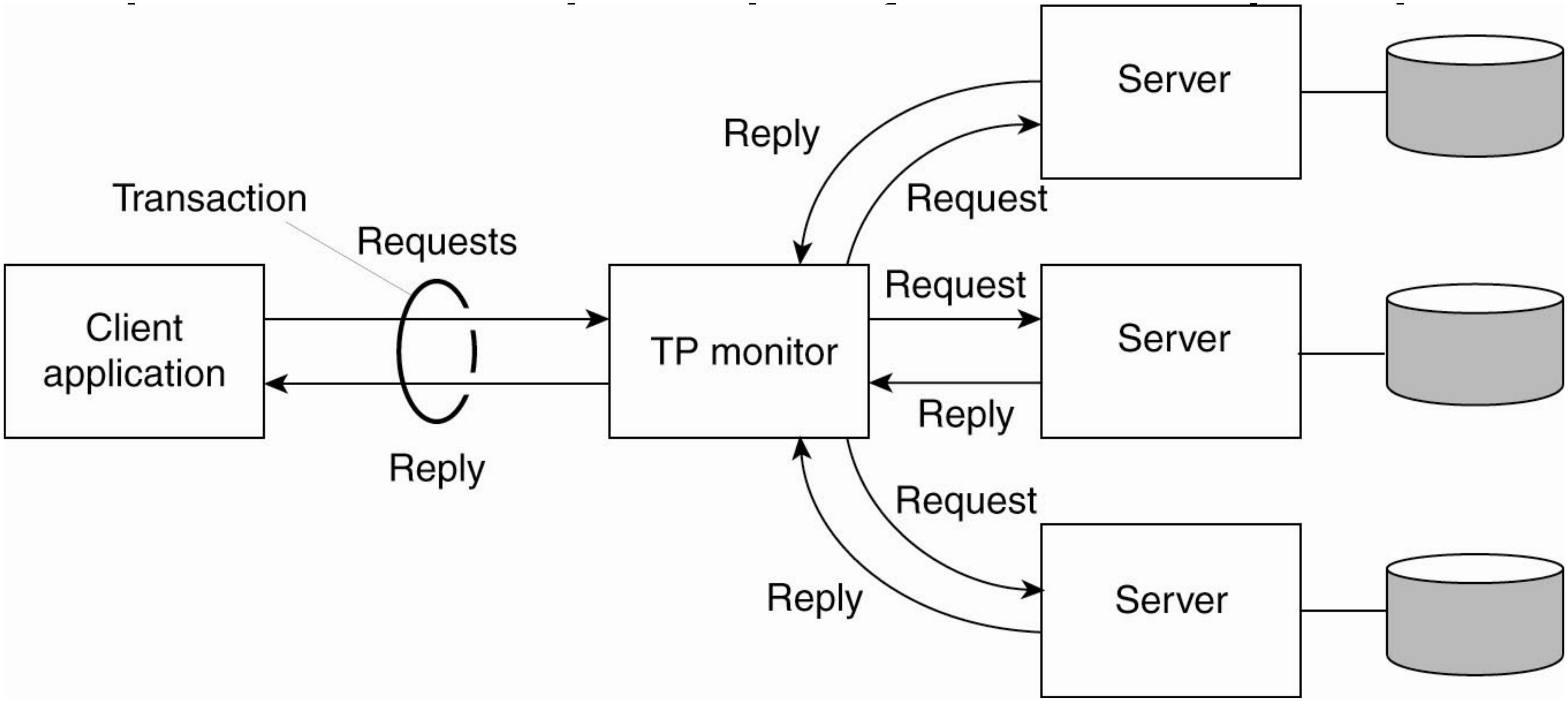
Distributed Computing Systems

- Loosely-coupled: Grid



- Key issue: Multiple Admin Domains: security

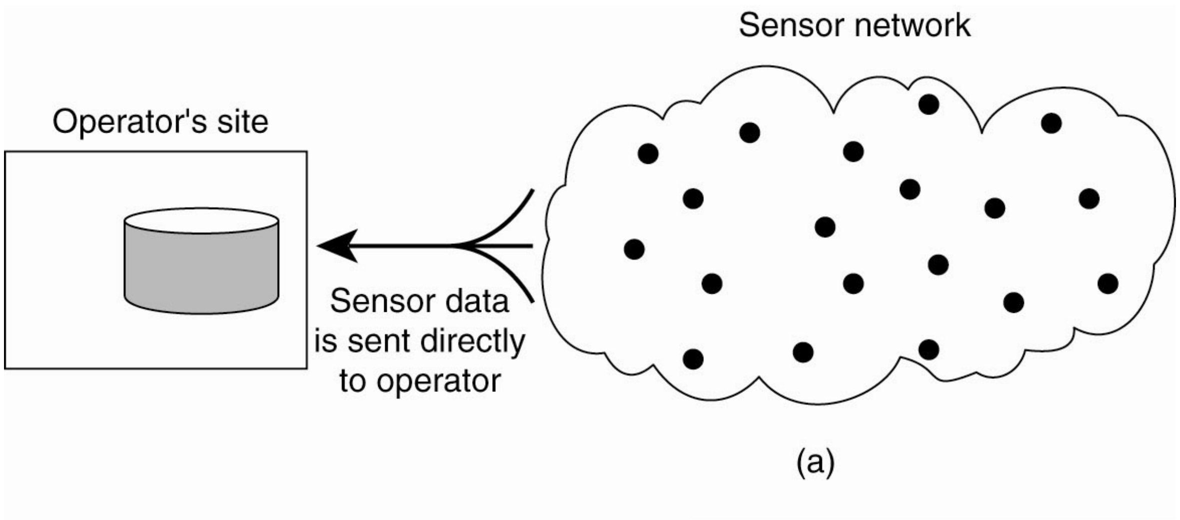
Distributed Information Systems: Distributed DBs

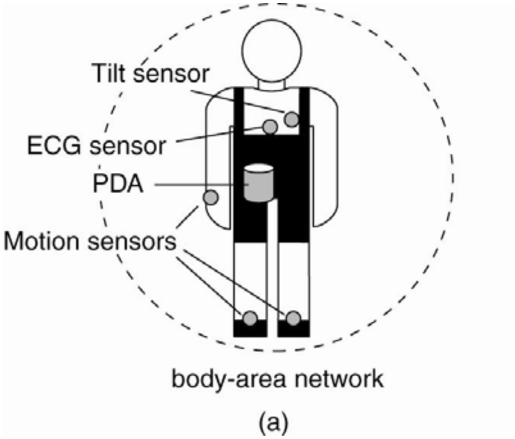


Key issue: ACID (atomic, consistent, isolated, durability)

Distributed Pervasive Sensor Systems

- Small unreliable elements
- Ad-hoc





Key issue: Resource constrained (e.g. battery)

Next Time

Read Chapters 1, 2 TVS

Explore website: schedule, dates, syllabus

Next topic: Architectural Styles