Distributed Applications

- Automated Banking Systems
- Tracking Roaming Cellular Telephones
- Air-Traffic Control Systems
- Retail Point-of-Sale Terminals
- Global Positioning Systems
- Fly-by-Wire Systems
- The World-Wide Web?

Coverage of this Course

Theory and Practice of

- Organization of Distributed Systems
- Communication in Distributed Systems
- Replication and other techniques for Fault Tolerance

We will *not* cover issues specific to *real-time systems*; we recommend reading Chapters 16 – 19 instead.

Ingredients of a Distributed System

- Multiple computers
- Interconnections
- Shared state

Issues to be addressed

- Unreliable Computers and Communication
- Independent Failure

Example: Banking

- Central office
- ATMs

Clients at different ATMs may access same account simultaneously

Example: Banking

- Two central offices
- Many ATMs

ATMs use nearest working central¡office. Each central office acts as abackup for the other.

What about outstanding transactions? What if the network fails?

Security

In practice, neither centralized computers nor distributed ones are secure.

Nevertheless, achieving security in distributed systems is a fundamentally different problem from that in centralized ones:

- Centralized systems can rely on physical security
- Users understand what trust to assign to the system
- Systems administrator is responsible

None of this is true in a distributed system

It is hard to know what is being trusted and what can be trusted

Networked vs. Centralized Systems

- Cycles always available
- Incremental growth
- Independent failure
- Increased Autonomy
  - purchasing
  - managament
  - software

- Harder to manage

State-of-the-Art Distributed System

No distributed system exists today that combines the accessibility, coherence, and manageability advantages of centralized systems with the sharing, growth, cost, and autonomy advantages of networked systems.

A “Best-of-Both-Worlds” Distributed System

We can describe a “Best-of-Both-Worlds” (*BOB*) system as a *properties and services model*.

Ingredients of BOB

- Heterogeneous set of hardware, software and data components
- Large size and geographic extent
- Connected by a network
- Uniform set of services
- Well-defined global properties

Global Properties of BOB

- Global Names
- Global Access
- Global Security
- Global Availability
- Global Management

Global Names

The same name works everywhere.

Machines, users, files, distribution lists, access-control groups, services have full names that mean the same thing regardless of where in the system these names are used.

Are Global Names Desirable?

It is useful that `sape@huygens.org` uniquely names one mailbox and that `http://www.Telefoongids.ptt-telecom.nl/` uniquely identifies a particular service.

But we also wish `/bin/sort` to refer to the sorting application, regardless of whether we run it on a MIPS, Pentium, or 68040 machine.

At Bell Labs we argue that global *conventions* are much more important than global names: `/dev/tty` refers to the controlling terminal on all Unix systems and `/etc/passwd` to the user administration file.

Global Access

The same functions are usable everywhere.

Functions, such as printing, mail, storage, database must be accessible in a similar manner everywhere. Accessibility also implies data consistency.

Global Availability

Services continue to work despite failures.

The level of replication of each service can be chosen to achieve any desirable degree of fault tolerance.

Global Security

The same user authentication and access control apply everywhere.

Users can authenticate themselves to the system anywhere. Any person can be put on any object's access control list. Communication between any two entities can be made private and authenticated.

This requires global naming of *principals* and global conventions for authentication.

Global Management

The same user can manage system components anywhere.

There is a common interface to management tools. Managing an entity does not require physical access to that entity.

Remote Invocation

A standard invocation mechanism allows any command interpreter, or any application to use any service.

The most widespread invocation mechanism used to be Remote Procedure Call. Now, http is rapidly taking over, even though it is only operationally defined (by implementations such as Netscape and Explorer).

Standardization encompasses parameter-passing mechanism, encoding of parameters on the network and protocols for transport of data and reporting of failures.

Blocking vs. non-blocking is a local issue.

Time

Global, standardized time allows clocks to be synchronized which is necessary for some functions and useful for many more.

Secure time is important for security functions.

Interface

An interface serves as a contract between a service and its clients.

Essential for achieving the global properties of BOB is that its interfaces are well defined.

An interface defines the operations provided, their parameters and the semantics of the operations.