Hot Topics in Operating Systems Research

CS 239, Spring 1996

June 5, 1996

Some Hot Areas

- Mobile computing
- Multimedia support
- Extensibility
- Highly optimized message handling
- OS support for electronic commerce
- Cryptography for the masses

Mobile Computing

An increasing fraction of the computers sold are portable computers

Wireless networks are beginning to appear

Portable computers are becoming more powerful

Systems Software Problems for Mobile Computing

- Power management
- Handling limited local resources
- Varying communications capabilities
- Security

Power Management for Portable Computers

Typical portable computers have battery lives of 2-6 hours

Shorter if used like a normal machine

Battery life isn't getting that much better quickly

So what do you do to make batteries last longer?

Power Management Issues

What uses power?

- Display
- Disk
- CPU
- Memory
- Communications devices

Different approaches required for each

Basic Approach for Power Management

Turn it off when you don't need it

One simple case - if user shuts the machine's lid, power nearly all down

One step further - if the user hasn't been entering anything, turn off screen

And another step - if the disk hasn't been used recently, spin it down

And another step - if there's little activity, slow down the CPU

Where's the Research in Power Management?

- When is it wise to turn components off?
- How do you assure state is preserved when they come back on?
- How do you make necessary performance guarantees?

Handling Limited Resources in Mobile Computing

- Disk space
- Communications bandwidth

Disk Space Limitations in Mobile Computing

Typically only one hard disk per mobile computer

Of limited capacity

When machine is disconnected, it's totally dependent on that disk

So it's important to fill the disk wisely

How To Fill the Disk

What should be on the disk?

The files a user needs while disconnected

How do you determine which those are?

- Ask the user
- LRU
- Intelligent hoarding

Why Not Ask the User?

He probably doesn't know

Modern software uses lots of strange stuff

E.g., running latex requires file decompression programs

Too heavy a burden on the user

Why Not LRU?

LRU only deals with recent history

Doesn't preserve information about relationships among files

So if you didn't follow a link in your document the last time, what's at the other end won't be on the disk

File Hoarding for Portable Computers

Watch user's file usage patterns over a long period of time

Based on information gathered, cluster together files that are part of same task

Figure out which tasks will be needed

Hoard entire clusters on the portable computer's disk

Hard Questions for This Approach

- How do you determine relationships between files?
- How do you build clusters?
- How do you determine which clusters to hoard?
- How little user intervention do you need?
- How do you measure success?

Determining File Relationships

Lots of possible ways

- Directory membership
- File name relationships
- Internal linking information
- Semantic distance measurements
- Or some or all of the above

Dealing With Varying Communications Capabilities

A mobile machine will have communications bandwidths varying from megabytes per second to nothing at all

And it changes as the machine moves

Other communications characteristics also change

Problems With Varying Communications

- Applications need to "hand off" their communications from one device to another
- Applications need to adapt to lower/higher communications rates
- Applications may want to split their communications between different channels

Multimedia Support

Users increasingly want audio and video delivered to their machines

Some from saved files

Some from local devices

Some over the network

What must the OS do to support these needs?

Multimedia and Real Time

Audio and video delivery requires real-time-like guarantees

Delays in delivery of this kind of data lead to undesirable effects

But most machines are not used exclusively for these tasks

What Problems Arise in Multimedia Support?

- Scheduling
- Buffer management
- Graceful degradation
- Smoothing jitter in the delivery

Scheduling for Multimedia Support

Audio and video require periodic service

If you don't schedule them in time, the output suffers

But you don't want to spend all system resources supporting a video stream

And what about handling multiple simultaneous video streams?

Overloading Multimedia Systems

When a system supporting multimedia gets overloaded, what should you do?

- Degrade video before audio
- Degrade incoming before outgoing
- Degrade streams opened the longest

But answers vary by system and desired result

Buffer Management For Multimedia

Multimedia data may arrive while the display process is swapped out

Or may be coming in too fast to deal with right now

So you must buffer it

Need buffers fast, and possibly lots of them

Need intelligence about handling full buffers

Graceful Degradation in Multimedia Systems

Don't freeze up when you can't keep up

Especially in video, relatively easy to offer slightly degraded services

- Color goes to B/W
- Frame rate goes down
- Resolution goes down

Partially the job of the application

Smoothing Jitter for Multimedia

Data may be delivered at different intervals than that at which it's displayed

Special buffering is required to ensure that display doesn't have jitter

Should buffer close to actual displaying device

Multimedia for Wireless/Mobile Computing

Must have good way to hand off data transmission between wireless cells

- Can't afford to drop frames when receiver moves

Mobile computer doesn't have power to customize its format

- So server should store/send data in multiple formats

Extensibility in Operating Systems

Widely regarded as key issue in designing modern operating systems

Microkernels provide it by allowing out-of-kernel servers

Stackable services provide it by allowing addition of stacked layers/protocols

Some approaches allow user code to execute in privileged mode

Why Is Extensibility So Important?

Experience shows system designers never get it completely right

And individual users/applications have special needs

Example - multimedia needs high bandwidth

- So it's bad to pass bits through the kernel

Why Not Just Microkernels?

- Speed
  - Still controversy over whether generally usable microkernel systems can be fast enough
- Safety
  - Hard to guarantee safe operations among several users/services sharing a service

One Solution - Run User Code in Kernel Space

SPIN (at U. of Washington) does this

Key issues -

Providing safety of overall system

High performance

Safety provided by object oriented techniques and explicit guards

Performance provided by run-time compilation of code into OS

Highly Optimized Message Handling

Many systems built on RPC, client/server, or object oriented model

Which means lots of messages between entities in different contexts

Success of such systems depends on extremely fast message delivery

Messages also important in parallel computing

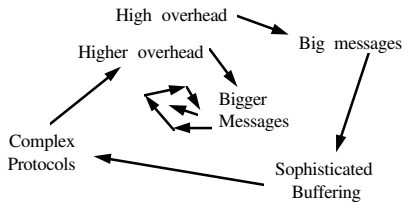
How To Handle Problem

Fundamental problem is too many cycles spent in message overhead

- Use bigger messages
  - Fewer instances of overhead
- Cut down heavily on per-message overhead
  - Less overhead per instance

Why Not Bigger Messages?

Because of the big message spiral -



Basic Ideas in Speeding Up Message Passing

- Avoid buffering and copying
  - Buffering requires management
  - Copying takes time
- Avoid context switching
  - Context switching is expensive
  - Naive microkernels do it a lot
- Use any available hardware assistance

One Approach - Protocol Service Decomposition

Done in context of particular communications protocols (e.g., IP)

Separate application's interface to the network from its interface to OS

In other words, hook up the application directly to network for common ops

OS takes care of heavyweight network management

Architectural Components of Protocol Decomposition

- Operating system server to provide heavyweight services
  - connection establishment, shutdown, routing info, etc.
- Multithreaded application library that implements basic protocol (send, receive)
- Network interface - thin layer of software controlling network HW

How Protocol Decomposition Fits Together

Application (containing multithreaded library) asks OS server to start network session

OS provides session that allows application to send certain operations directly to network interface

Normal reads/writes don't go through OS

Another Approach - Active Messages

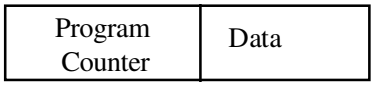
Usually in context of parallel processor and cooperating threads within task

Have handlers for messages compiled into threads

Handlers know precisely what to do with specific message types

Message contain PC for its handler

Contents of Active Message



Receiving an Active Message

- An interrupt comes in on a communication device
- Read message off device
- Don't queue it, buffer it, pass it through OS at all

  Just jump to the specified program counter with the provided data

Active Message Handlers

Handler program must deal with active message very, very quickly

Because it's like an interrupt handler

Typically, perform quick operation and return short result

Used for lightweight operations, not heavyweight ones

Electronic Commerce

How can we do business over computer networks?

A fundamental issue is how to charge for services

- Bill the user
- Credit card transactions
- Electronic checks
- Electronic cash

Authentication for Electronic Commerce

Billing, credit card transactions, electronic checks require strong authentication

Servers that have never heard of users before may need to authenticate them

Networks of trusted authentication servers often proposed

Electronic Cash

Different properties than other forms of electronic commerce

- Anonymous
- Transferable
- Non-traceable
- Non-forgeable

Issued by electronic banks

Solutions rely heavily on cryptography

Other Issues in Electronic Commerce

Can we provide services free from eavesdropping?

Can we provide one-time-per-payment services?

Can we provide "micropayments?"

Can we prevent dissemination of valuable data?

Cryptography For the Masses

What do we do if millions of machines with millions of users need to communicate securely with each other?

Anyone can communicate with anyone else at any time

How to build a practical system to protect these communications?

Technical Problems in Large-Scale Cryptography

- Key distribution
- Enforcing proper use of keys
- Large scale authentication

But social problems are harder

Social Problems in Large-Scale Cryptography

Should terrorists be able to conduct untappable phone conversations?

Should child pornographers be allowed to hide their wares behind encryption?

Should the government be able to read any and every encrypted message?

Should anyone else be able to read them?

Key Escrow Solutions

Allow ordinary users to use cryptographically strong cipher

But make it possible for the government to obtain the key

Without assuming users will willingly provide it

How To Make Key Escrow Work?

1). Divide key into parts

2). Store key with different, trusted agencies

3). Require legal warrant to obtain all parts of keys

Issues in Key Escrow

How do we force the keys to be stored?

Hardware-based solution - cryptography only through particular HW device

Manufacturer gives keys to agencies

But what about danger of overuse of keys?

Handling Key Overuse

Create new key for each communications

Encrypt that key with super-key

Escrow the super-key

Glue the encrypted key to all communications that use it

Key Escrow Is Tricky

Proposed schemes have had several vulnerabilities discovered

- Making communications insecure
- Or making escrowing ineffective

If it doesn't work right, it will be broken

Social Issues of Key Escrow

Do we trust the "separate" government agencies?

Do we trust the government-created encryption algorithm?

Why on earth would other countries want their encryption readable by the US?