OS Research
(CSE 422S)

Ken Wong
Washington University

kenw@wustl.edu
www.arl.wustl.edu/~kenw

New OSes at SOSP

What Do I Want?

- **Don’t bother me** unless I want to be bothered
  - Universal connectivity, YES
  - Constantly connected, NO
- A **good assistant** in my pocket
  - An idea amplifier
  - A database of facts and ideas
  - Helpers
- Miscellaneous
  - Quick access
  - Large screen and large fonts
- A **development environment** on my desk
- No more than needed

Systems Software Research

- **Systems**
  - OS, networking, languages
- **Software**
  - You know what it is
- **Research**
  - Academic research
- **Is**
  - Now (circa 2000-????)
- **Irrelevant**
  - Does not influence industry
What Does My Brother Want?

- Dow Jones from anywhere
  - “I don’t invest when people are crazy ... just when they are stupid”
- Email
- No hassles
- Dow Jones from anywhere
- Email
- No hassles
- Dow Jones from anywhere
- Email
- No hassles

What Does Grandma Want?

G: “Remind me. What do you do for a living?”

“I study software for advanced computers ... multicore computers ... your next PC will have 2, 4, 8, 16 computers inside.”

G: “I only need one to send email. Can I buy one for 1/2, 1/4, 1/8, or 1/16th the price?”

“Well, ...”

adapted from Patrick Crowley, “The Future in Your Pocket”

Experimental Router Testbed

Architecture
**Intel IXP 2800 (Data Plane)**

- No OS
- Hardware multithreading
  - No preemption
- No caches
  - Memory latency hiding instructions
- Pipelined MEs
- 12,000 bits @ 10 Gb/s → 1200 nsec

**Control Plane**

- User Requests
- Routes & Filters
- Statistics
- TCAM
- XScale CP
- SRAM
- Stats
- Data Plane Config.
- Data Plane MEs
- Plugins

**General-Purpose Multicore Processors**

- Technical Motivations
  - CPU clock frequencies will not continue to increase significantly
    - Laws of physics
  - Increasing clock frequency → Increasing heat
- Vendors
  - Intel Desk/Laptop: Dual-core
  - Intel Servers: Quad-core
  - Sun DB Servers: 16 cores + transactional memory
- Do we need them?
  - Cisco uses a 192-core processor in its high-end router
  - What does Grandma want?

**Alternative: Integration**

- ARM CPU
- 2D/3D GPU
- DSP
- Crypto
- Codec
- Display
- Camera
- Touch
- Speech
- Radio
- DRAM
- Flash
- USB
- Radio

**Software Development**

- Battery life, networking, displays, hw accelerators
Synchronization

Bank Account Example (1)
- See Simon Peyton Jones, “Beautiful Concurrency”
- Procedure to transfer money from account 1 to account 2
- Using Locks

```java
class Account {
    int bal;
    synchronized void withdraw( int n ) {
        bal = bal - n;
    }
    void deposit( int n ) {
        withdraw( -n );
    }
}
```

```
from.lock();
to.lock();
from.withdraw( x );
to.deposit( x );
from.lock();
to.lock();
```

Deadlock-Free Transfer
- What if you want to block until there are sufficient funds in the `from` account?
  » wait on a condition variable

```
if ( from.num() < to.num() )
    { from.lock(); to.lock(); }
else    { to.lock(); from.lock(); }
from.withdraw( x );
to.deposit( x );
from.lock();
to.lock();
```

Difficulties With Locks and CVs
- Forget to lock
  » incorrect update
- Too many locks
  » less concurrency or deadlock
- Wrong lock
- Wrong locking order
  » deadlock
- Error recovery
  » avoid inconsistent state or indefinite locking
- Lost wake-ups
  » indefinite waiting
- Difficult to compose modules (unchanged)
**Transactional Memory**

- Read/write memory locations atomically
  - mutual exclusion without using locks
  - like a database transaction
    - not finished until successful commit.
- Transaction
  - sequence of memory loads and stores
  - unlike databases, not concerned with failures
  - if a transaction commits, all loads and stores appear to have run atomically with respect to other transactions
  - if a transaction aborts, none of the stores take effect
    - transaction can be restarted
  - programmer specifies transaction begin and end

**No Futz Computing**

- See “Research Issues in No-Futz Computing”
  - David Holland, William Josephson, Kostas Magoutis, Margo I. Seltzer, Christopher A. Stein and Ada Lim
  - context: System administration
- What is futzing?
  - tinkering, fiddling with something
  - examples
    - blue screen of death for 3 months
    - getting ucontext to work on grid.cec Linux
    - getting my HP printer to work on my Windows XP
- Goals
  - Reduce cost
    - sys admin salaries, user time
  - Reduce frustration of users and sys admins

**Current Approaches to Reducing Futz**

- Limit functionality
  - e.g., server only handles email or NFS
- Homogenize
  - Install one master copy
  - Push out master copy to all nodes
  - Flaws
    - users aren’t allowed to customize
    - if master has a virus, then all nodes will have the same virus
    - organizations grow their nodes ➔ more than 1 kind of node
- Centralize
  - Futz only on replicas of node types
    - e.g., initial configuration
  - Flaw: no reduction in futz
Some Sources of Futz

- **Policy state**
  - Configuration parameters that reflect site/user policy
- **Autoconfiguration data**
  - Data used to autoconfigure other systems
- **Manual configuration state**
  - Setup of the operating environment or hardware
- **Application file state**
  - Files and meta-data used by applications

Declarative Configuration Example

- **Declarative**
  - Host X should use host M as the mail server
- **Implementation**
  - Run script on host X to edit the sendmail config file
- **What’s wrong with implementation-oriented?**
  - Intent is not clear
  - Cannot reason about desired configuration
  - Cannot validate policy
- **Why declarative?**
  - Simpler specifications
    - unordered rules
  - More portable
  - Easier to compose (just rules)

Anderson and Couch, "What is This Thing Called System Configuration?"