



Java IPC

And the CLIP Library

Clark N. Hobbie
Long Term Software, LLC
clark.hobbie@ltsllc.com

Who Gives a Damn?

You Should Care Because...

- If you are in a corner
- Java sync is one VM only
- Others require JNI
- Platform differences



What Are the Options?

- Sockets
- Message Queues
- Semaphores
- Shared Memory

Which Option Should I Use?

Shared Memory

- JRE support
- Highest bandwidth
- Decent synchronization
- Naming support

Why Should I Care about CLIP?

- New primitives
 - Semaphores
 - Message Queues
- Simplifies existing primitives
 - Shared Memory

Java

```
RandomAccessFile raf = new RandomAccessFile("/temp/smfile", "rw");  
FileChannel chan = raf.getChannel();  
MappedByteBuffer buf = chan.map(MapMode.READ_WRITE, 0, size);  
byte[] other = new byte[1024];  
buf.position(0);  
buf.get(other, 0, 1024);
```

CLIP

```
SharedMemory smem =  
    new SharedMemory( "/temp/smfile" );
```

What is IPC?

Inter-Process Communication (IPC)

- Multiple processes
- Naming
- Synchronization
- Bandwidth

Naming



Naming

How do you...

- Connect processes together?
- Determine who is allowed to connect?
- Examples
 - TCP/IP?
 - Email?
 - Telephones?

File Naming

Many IPC methods use the file system because

- Visible to all processes
- Has access control built in

File Naming Examples

- Memory Mapped Files
- Named Pipes/FIFOs

Synchronization

- Event ordering
- Mutual exclusion

Example: Online Purchase

1. Get user credit information
2. Decision purchase
3. Update credit information

Without Event Ordering

Client A

- 1 Get credit info
- 2 Decision purchase
- 3 Update info

Client B

- 2 Get credit info
- 4 Decision purchase
- 5 Update info

With Event Ordering

Client A

- 1 Get credit info
- 2 Decision purchase
- 3 Update info

Client B

- 2 Wait for A
- 4 Get credit info
- 5 Decision purchase
- 6 Update info

Synchronization is Event Ordering

Instead of this order:

- A gets credit info
- B gets credit info
- A decisions purchase
- A updates info
- B decisions purchase
- B updates credit info

We want this order:

- A gets credit info
- A decisions purchase
- A updates info
- B gets credit info
- B decisions purchase
- B updates credit info

Which Type of IPC is Appropriate?



IPC Types

- Shared Memory
- Semaphores
- Sockets
- Message Queues

Shared Memory



Shared Memory

- Preferred approach
- Any to any
- get/put semantics
- Synchronization support
- JRE support
- Naming support

Shared Memory Details

- Synchronization
 - File Locking
 - Lock/unlock ~ 25 usec
- Naming
 - File naming
- Bandwidth
 - Synchronized 250MB/sec
 - Unsynchronized 1000MB/sec

Memory Mapped Files

- Start with a file
 - Appears the same to everyone
 - Reads/writes appear to everyone
- Now speed it up
 - Until its as fast as memory
 - Like having the OS buffer

Why Dear God, Why?!!

- Originally Unix
 - Where *everything* is a file
 - mmap system call
 - CreateFileMapping system call
- Solves the naming problem
- Solves the access problem

Example: World of Warcraft!



WoW: Requirements

- One client process per player
- One server process for all
- Server periodically reads all orders
- Server issues results

WoW: Design

Client
(EvilOne)

Client
(Lancelot)

Client
(Martha)

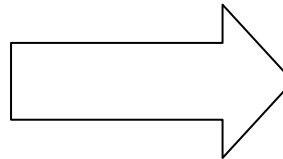
| Orders | |
|----------------|--------------|
| Player | Order |
| EvilOne | kill |
| Lancelot | kill |
| Martha Stewart | kill |
| Conan | make cookies |
| ... | |
| | |
| | |
| | |

Client
(Conan)

WoW: Design

| Orders | |
|----------------|--------------|
| Player | Order |
| EvilOne | kill |
| Lancelot | kill |
| Martha Stewart | kill |
| Conan | make cookies |
| ... | |
| | |
| | |
| | |

Server
Process



| Results | |
|----------------|------------|
| Player | Result |
| EvilOne | dead |
| Lancelot | dead |
| Martha Stewart | dead |
| Conan | overweight |
| ... | |
| | |
| | |
| | |

Shared Memory: Summary

- Preferred IPC
- 250 to 1000MB/sec
- File naming
- Synchronization through file locking
- JRE support

Sockets



Sockets

- TCP/IP
- Point to point
- Stream oriented
- Client/Server
- Synchronized
- Java support
- Naming support

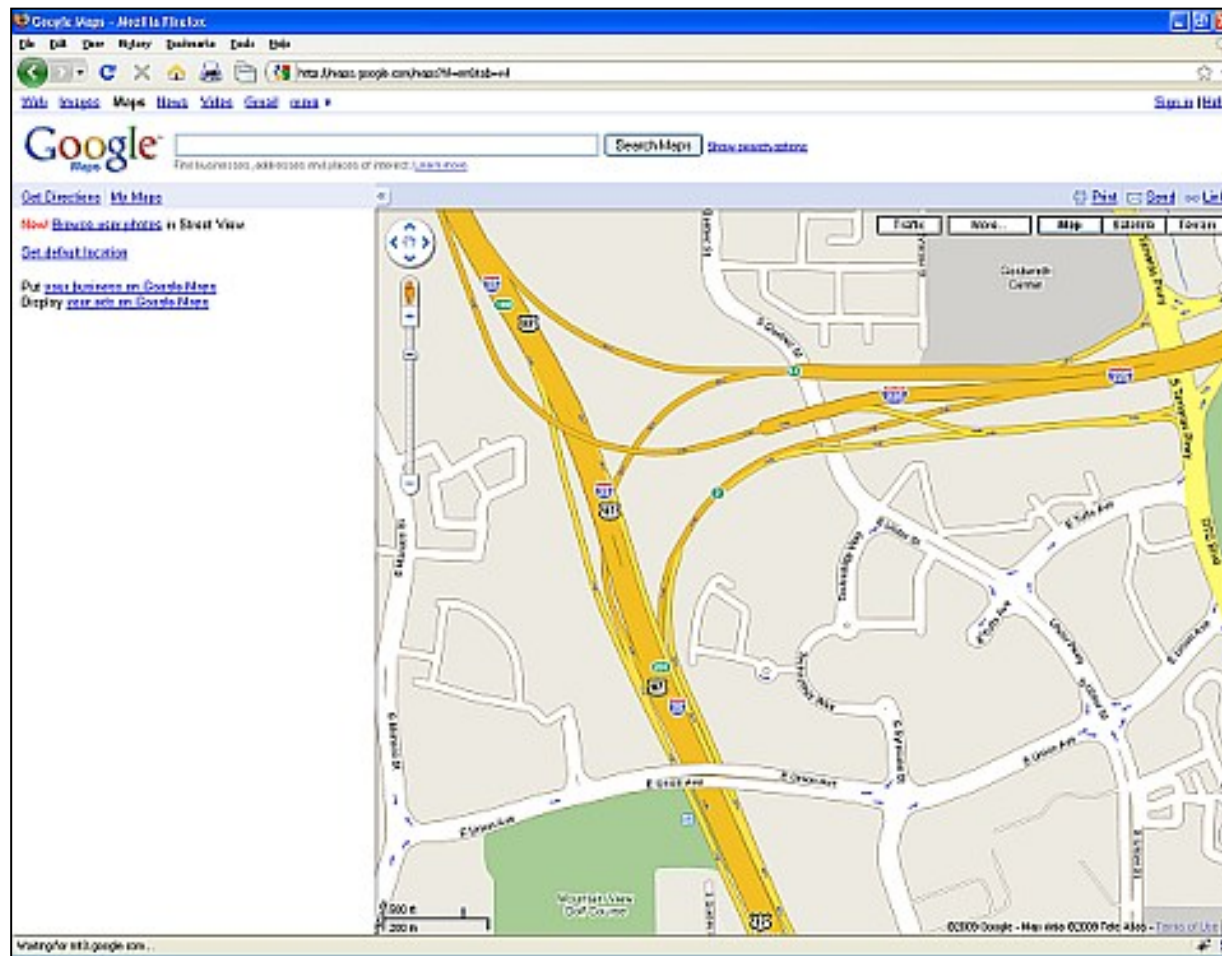
Sockets Details

- Naming
 - 127.0.0.1 port 7777
- Synchronization
 - Accept, read, write
 - 70 usec
- Bandwidth
 - 15 MB/sec

Sockets: CLIP

- Some utility classes
 - ThreadedSocketServer
- JRE already has excellent support

Example: Google Maps

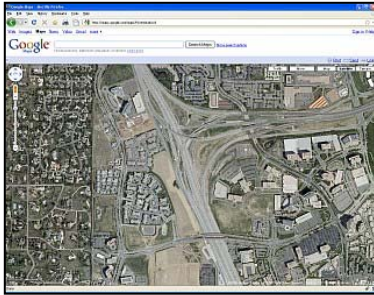


Copyright © Google

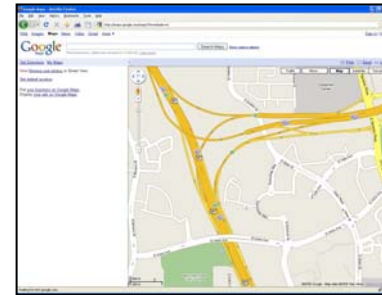
GMaps

Combining Data

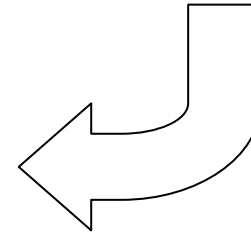
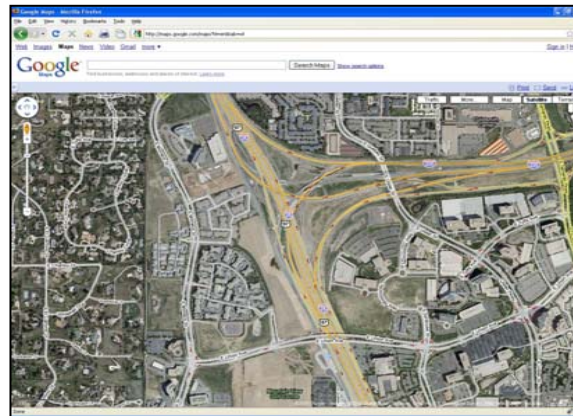
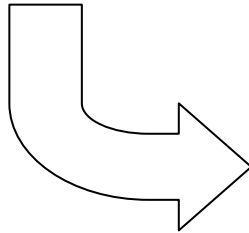
Image



Streets



Overlay

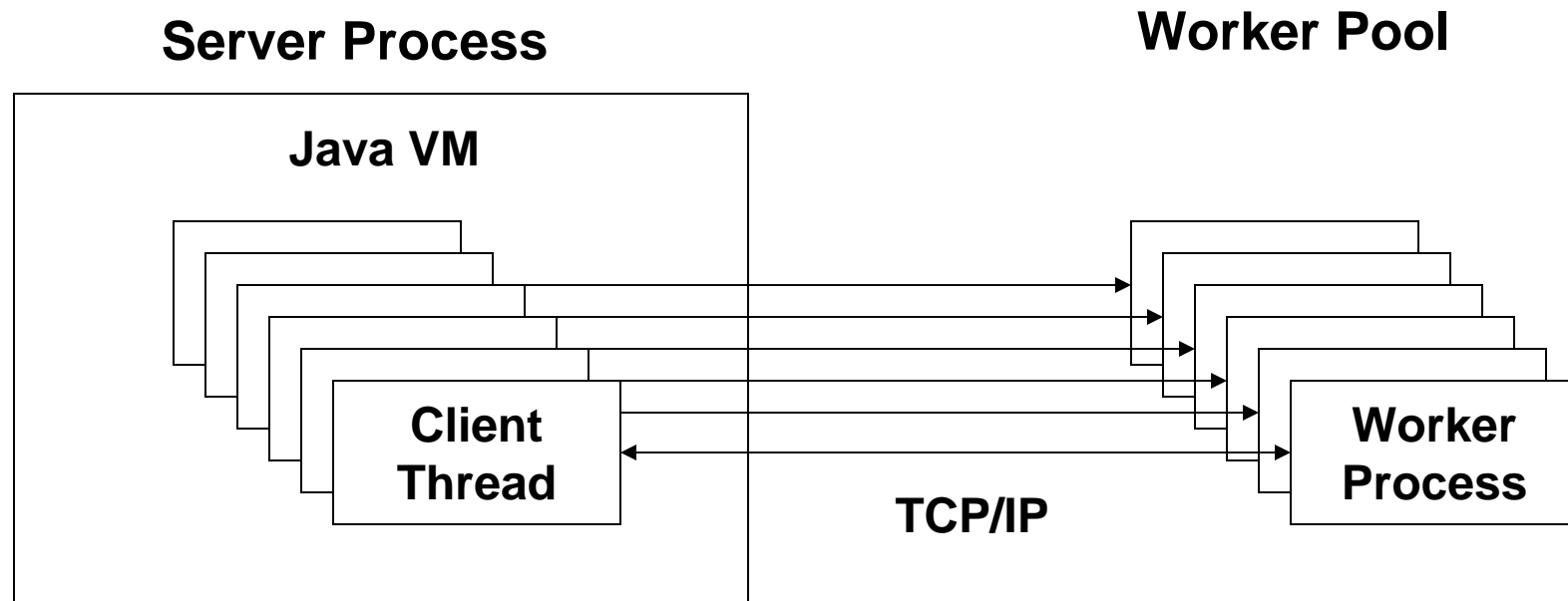


Images are Copyright ©
www.Google.com

GMaps: Requirements

- C legacy code
- One instance/process
- Receive file name
- Process for 1 to 10 sec
- Respond with new file name

GMaps: Design



Sockets vs Shared Memory

- Faster synchronization
 - 25 usec vs. 70 usec
- More bandwidth
 - 250 MB/sec vs. 15 MB/sec

Message Queues



Message Queues

- Point to point
- Message or stream
- Client/Server
- Synchronization support
- No Java Support
- No naming support

Message Queue Details

- Synchronization
 - Accept, read, write
 - 25 usec
- Bandwidth
 - 167 MB/sec

Platform Differences

- Direction
 - Linux is one-way
 - Windows is two-way
- Naming
 - Linux: any
 - Windows: must be `\\.\\pipe\\name`
- Misc
 - Windows pipes are networkable

Message Queues: CLIP

- MessageQueue class
- One direction
- File naming
- JNI under the hood

Message Queues vs. Shared Memory

- Less bandwidth
 - Synchronized: 250MB vs. 167MB
 - Unsynchronized: 1000MB vs. 167MB
- No Java support
- Platform differences

Semaphores



3/3/2009

<http://ltsllc.com/slides/ipc.html>

46

Semaphores

- Synchronization only
- Any to any
- Access via increment/decrement
- No Java support
- No naming support

Semaphore Details

- Integer value
- Decrement reserves
 - Blocks if the value is 0 or less
- Increment releases
 - May wake a blocked process
- N-ary semaphores
 - Values other than 0 or 1

Semaphore Details

- Synchronization
 - 25 usec
- Platform naming differences
 - Linux: /somename
 - Windows: somename
 - Ad hoc access control

Semaphores CLIP

- Semaphore class
- File system naming
- JNI under the hood

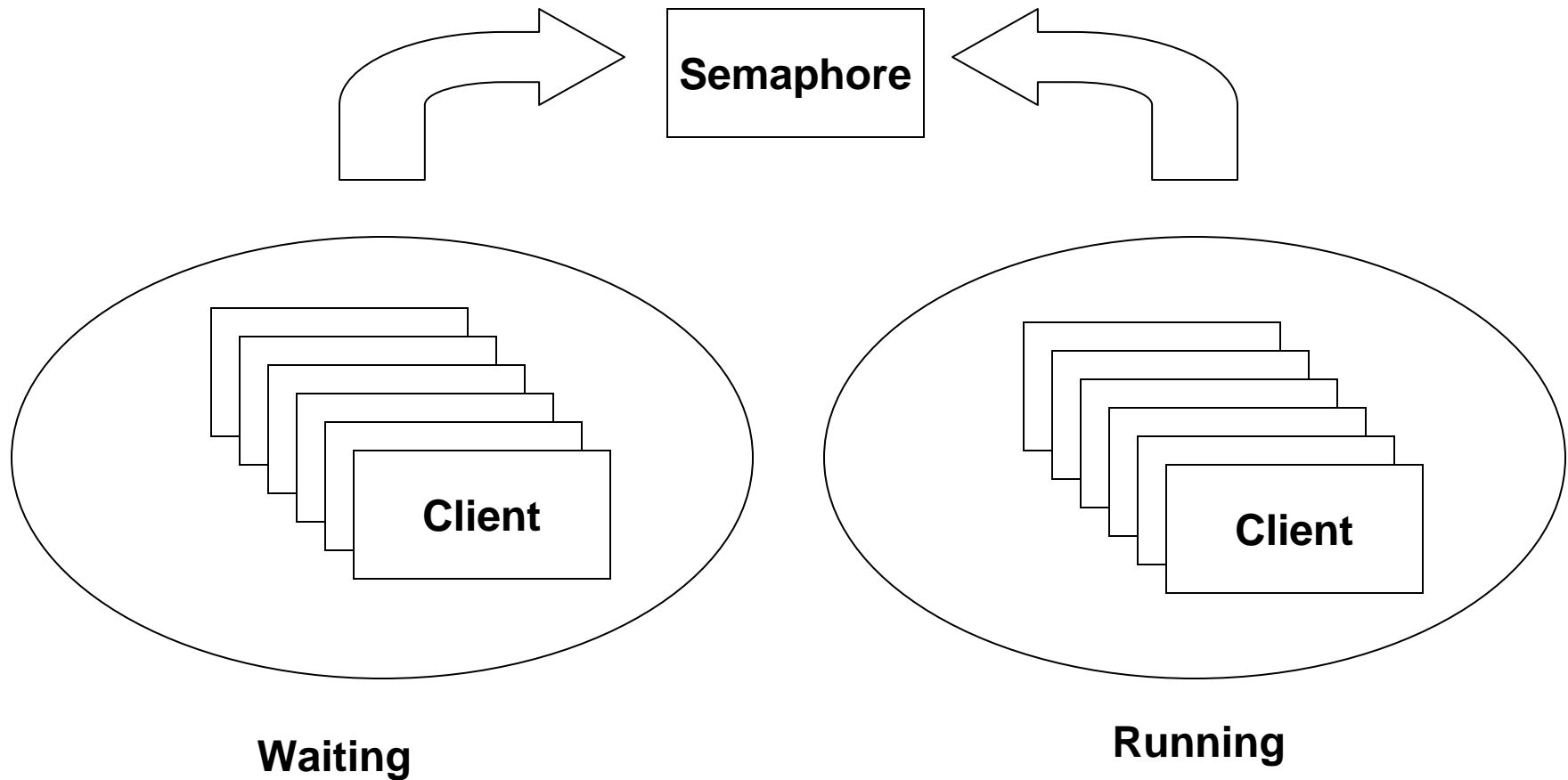
Example: The Liminator



The Liminator: Requirements

- Start with Google Maps
- Too many processes == poor performance
- Limit processes with a semaphore
 - Initial value = max number of processes
 - Reserve when trying to spawn
 - Release when complete

Liminator: Design



Semaphores vs Shared Memory

- About the same speed
- No naming support
- No JRE support

Summary



Summary

- IPC
 - Multiple processes
 - Naming
 - Bandwidth
- CLIP
 - Open source Linux & Windows
 - New primitives via JNI
 - Simplify others

Summary of IPC Types

| IPC Type | Sync | Band | Java Support? |
|----------------|---------|-------|---------------|
| Semaphores | 25 usec | N/A | No |
| Sockets | 70 usec | 15MB | Yes |
| Message Queues | 25 usec | 167MB | No |
| Shared Memory | 25 usec | 250MB | Yes |

Resources

| | |
|---------------|---|
| Slides & code | ltsllc.com/talks/ipc |
| Windows | msdn.microsoft.com |
| Linux | Advanced UNIX Programming basepath.com/aup/index.htm |
| Java | JTUX basepath.com/aup/jtux |
| Sun Forums | java.sun.com |
| Code Project | codeproject.com |

The End



Which Option is Best?

