**6.033 Lecture 1**
Sam Madden
Dina Katabi

What are computer systems?

(Show slides)

What makes building computer systems hard?

Complexity

What does that mean?  (A bit vague)

- lots of components
- lots of connections
- irregular
- difficult to describe
- requires many people to maintain

(Show Kernel Lines of Code)

Complexity limits what we can build;  need to mitigate it — more complex things are
more expensive, harder to maintain, break more, etc.

Other issues  w/ complexity

        - emergent properties  — surprises
        - difficulty scaling — small designs don't work at scale
        ( not just problems in computers)
        - propagation of effects — no small changes


So how do we manage complexity?

Learn from design principles — Ch 1 gives a bunch, e.g., "Avoid Excessive Generality"

"**Modularity**"

Divide system into components, consider implementation and interface of components
separately.

Different organizations of modules — e..g, a hierarchy, as in domain names, or a layers,
as in network protocols

Cloud —> Modules


**"Abstraction"**

Should be able to specify their interfaces without specifying their implementation
— a familiar concept from software engineering.

E.g., I don't need to know how a car works to drive it

More relevantly to computers , I don't need to know how a transistor works to program a computer,  or how assembly language works to write a Java program, or Java works to use a Word Processor implemented in it

So, designing systems is about deciding on the set of components for a system, and their interfaces

Let's start with an familiar system



        Web App <——> Web Server  <——> Database

Later in this course we will bust open these components — in particular the database and the network links between the components — to understand how they work, but not today.



What else would I like from these components?

"Enforced **Modularity**"

Want to bound the interactions between the components in some way — really don't want to have these modules interfere w/ each other

Procedure calls / modules / classes
————————App ——————————
Class WebBrowser          Class WebServer  |
main                      loadURL          |
…                         …                |
————————————————————————————

Does this enforce modularity?

Not really - compiler / language only protects components from each other a bit

E.g., in a C program, a misbehaving function can overwrite memory some other module is using

What about Java?  Can't overwrite memory, but even there, a component that crashes, or throws an uncaught exception, or goes into an infinite loop can cause the whole program to stop working.

Might be fine in some settings, but not OK to have database crash for misbehaving web apps, or web server to crash because user requested some crazy URL

**Enforcing Modularity**

So how do we enforce modularity?

Lots of different ways.  Over the next few lectures are going to study in depth how a computer operating system enforces boundaries between itself and applications running on it, and between applications running on the same machine.
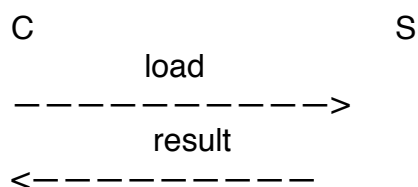
But for today let's look at an even simpler method:  **client / server**.

Let's put each component on a different machine:


M1                                    M2

Web Client           <———>            Web Server
                     messages

Only interaction is via messages



Specifically:

C                             S
        load
————————————>
          result
<————————

How does that help?

- Messages don't allow components to overwrite the memory of other components.

- If one component crashes, it doesn't cause the other component to crash.

- Client can time out, or retry.

No **fate sharing**.

Of course, messages can be malformed/garbled/malicious, and such messages can cause a component that is not careful to do bad stuff.  So we need to carefully validate messages before processing them.

Not always easy, e.g., many well known computer systems are vulnerable to attacks like buffer overflows, where carefully crafted messages that exceed the size of input arrays cause the receiving component to execute code the attacker sends in.   We will talk much more about this later in the course.

Client server used everywhere —
      to share data — e.g., on the web
      to allow remote access — e.g., to a database, or a bank account
      to create a trusted 3rd party — e.g., ebay runs auctions all parties trues

Can we make client server look like procedure call?

**Remote procedure Call**

(show slides)

Idea — can make client server look like a regular procedure call

Problem:

RPC is not a procedure call!

messages can be lost

server can crash