Material and some slide content from:

- Emerson Murphy-Hill
- Reid Holmes
- Mehdi Mirakhorli

- Software Architecture: Foundations, Theory, and Practice
- Essential Software Architecture

SE2: Introduction to Software Architecture

Mei Nagappan

What is Architecture?

- Encyclopedia Britannica defines it as
  - both the process and the product of planning, designing, and constructing buildings or any other structures

The three original principles

- Roman architect Vitruvius (early 1st century AD) in his book De Architectura
  - Durability – a building should stand up robustly and remain in good condition.
  - Utility – it should be suitable for the purposes for which it is used.
  - Beauty – it should be aesthetically pleasing.

[TAILOR ET AL.]

The architect

- Distinctive role.
- Broadly trained.
  - Requirements, design, implementation, & use.
- Has a keen sense of aesthetics.
- Strong understanding of the domain.
  - What are these for buildings?
  - What are these for software?

What common benefits can software gain from an architect that a building gets from its architect?

Analogy to Buildings

- Arch focuses on client's needs
- Iteration on a set of blueprints, refining as req
  - Intermediate plans, mockups, prototypes
- Created by specialists, not end users
- Structure induces properties (e.g., in a castle)
- Architects require broad training
  - Leverage lessons from past generations

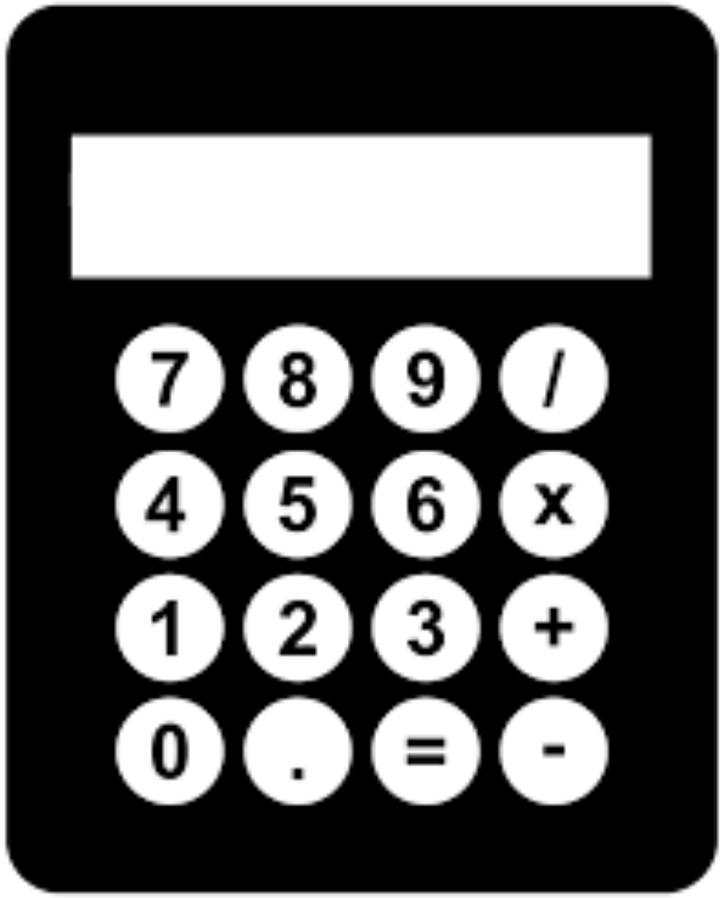
How is building architecture different from software architecture?

Shortcomings of Analogy

- We have much more experience with buildings
- Buildings are physical artifacts; SW is intangible
- Software industry is less differentiated (e.g, no ‘exception specialists’)
- Anyone can write software
- Deployment and Ops are different
- Nature of dynamic load is different
- Changes are expected

Why do we need Architecture?

The Software Equivalent



Architecture

- Architecture is:
  - All about communication.

Architecture

- Architecture is:
  - All about communication.
  - What ‘parts’ are there?

Architecture

- Architecture is:
  - All about communication.
  - What ‘parts’ are there?
  - How do the ‘parts’ fit together?

Architecture

- Architecture is:
  - All about communication.
  - What ‘parts’ are there?
  - How do the ‘parts’ fit together?
- Architecture is not:
  - About development.
  - About algorithms.
  - About data structures.

What is Software Architecture?

What is Software Architecture?

- The conceptual fabric that defines a system
  - All architecture is design but not all design is architecture.

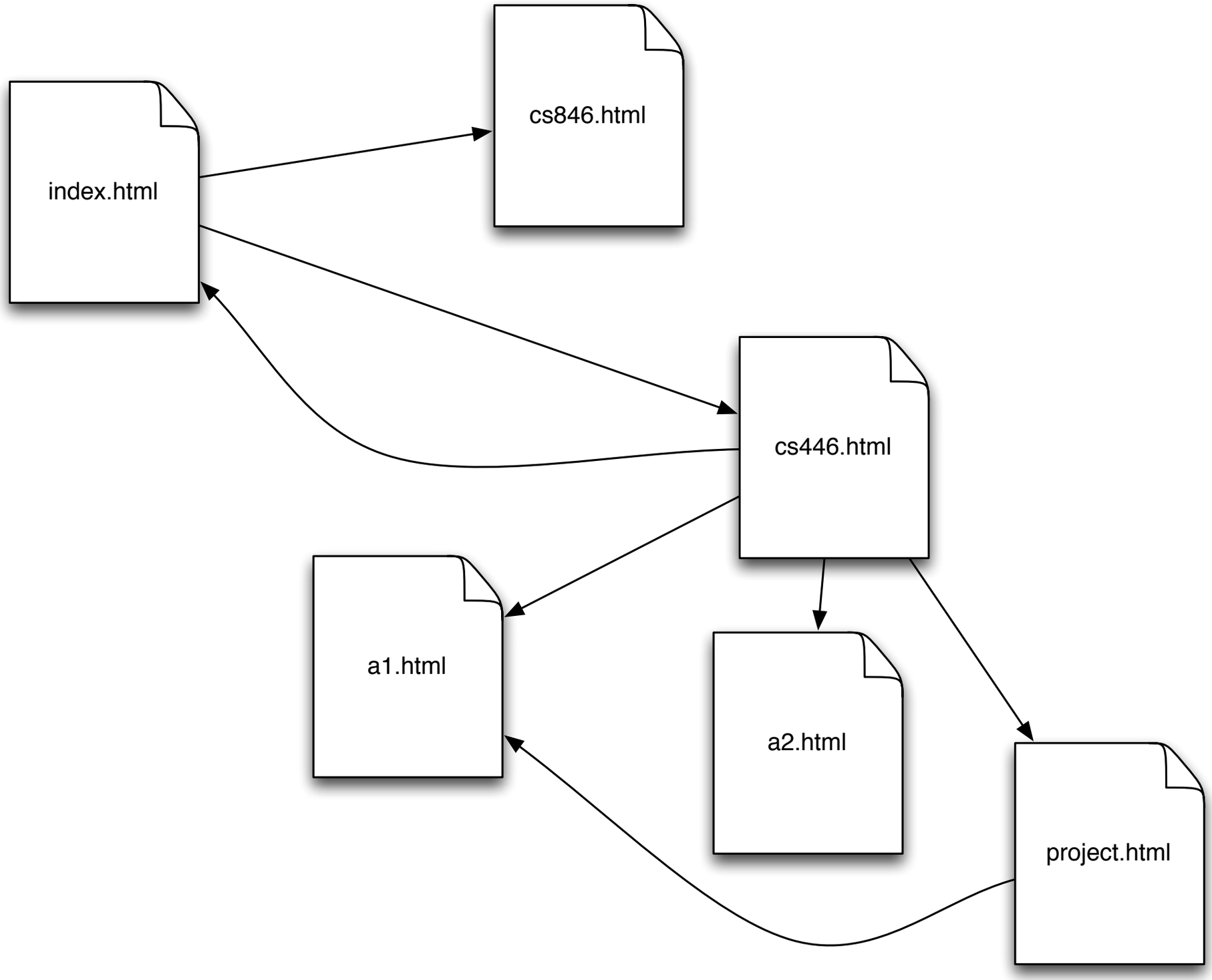
What is Software Architecture?

- The conceptual fabric that defines a system
  - All architecture is design but not all design is architecture.
- Architectures capture three primary dimensions:
  - Structure
  - Communication
  - Nonfunctional requirements

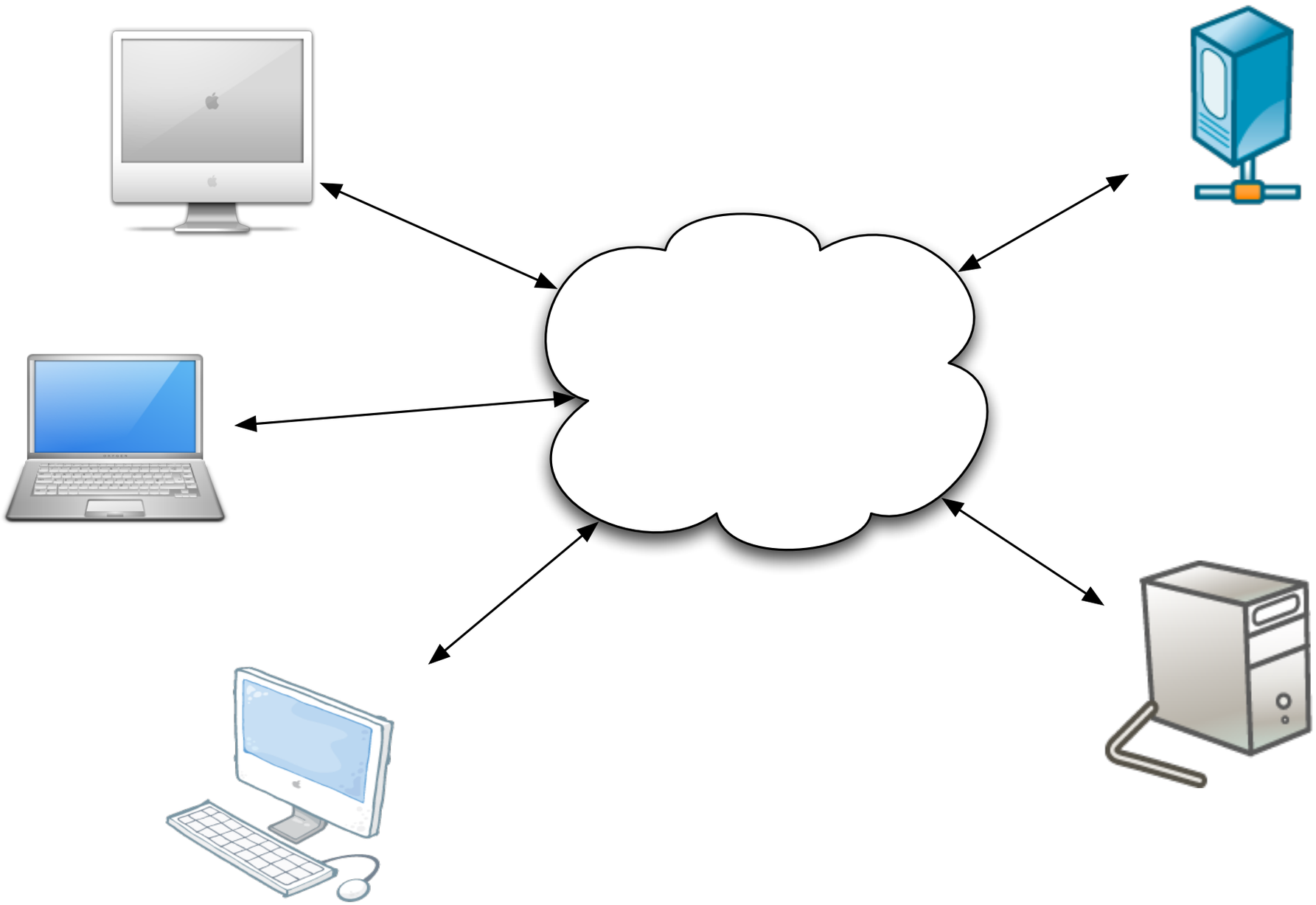
ANSI/IEEE 1471-200

"Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution"

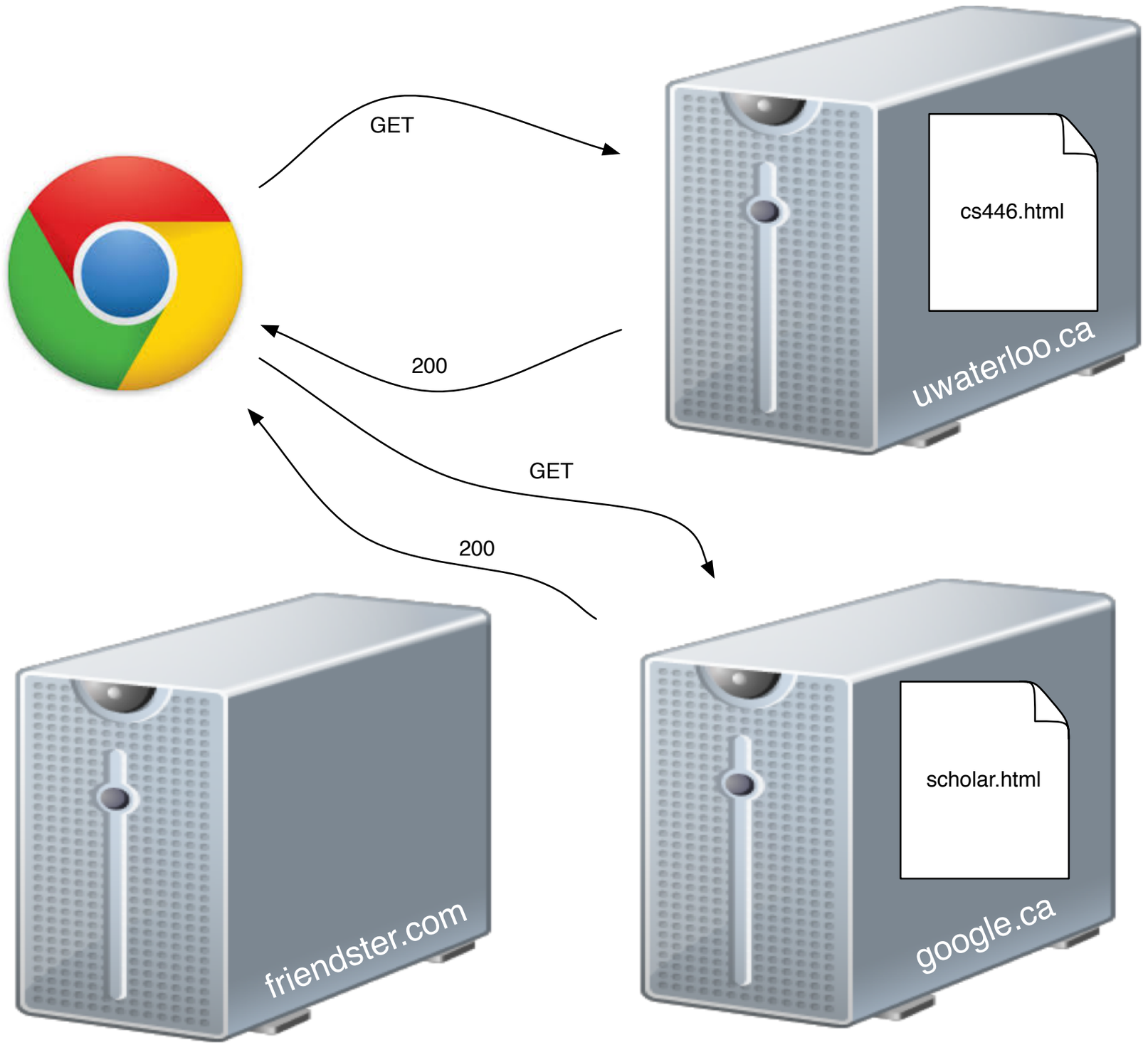
Logical Web Architecture



Physical Web Architecture



Dynamic Web Architecture



Non-functional requirements

- Technical constraints: restrictions made for technical reasons
- Business constraints: restrictions made for business reasons
- Quality attributes: e.g., the *'ilities'*
  - Scalability
  - Security
  - Performance
  - Maintainability
  - Evolvability
  - Reliability/Dependability
  - Deployability

Why is Software Architecture important?

Eoin Woods

"Software architecture is the set of design decisions which, if made incorrectly, may cause you project to be cancelled."

Why is Software Architecture important?

- Architecture focuses on those aspects of a system that would be difficult to change once the system is built.

Why is Software Architecture Difficult?

Philippe Krutchen

"The life of a software architect is long (and sometimes painful) succession of sub-optimal decisions made partly in the dark.

[TAILOR ET AL.]

What makes building systems so hard?

- Young field.
- High user expectations.
- Software cannot execute independently.

Difficulties Classified

- Incidental difficulties [Brooks MMM].
  - Problems that can be overcome.
- Essential difficulties [Brooks MMM].
  - Those problems that cannot be easily overcome.

[TAILOR ET AL.]

Essential Difficulties

- Complexity
  - Grows non-linearly with program size.
- Conformity
  - System is dependent on its environment.
- Changeability
  - Perception that software is easily modified.
- Intangibility
  - Not constrained by physical laws.

Attacks on Essential Difficulties

- High-level languages.
- Development tools & environments.
- Component-based reuse.
- Development strategies.
  - Incremental, evolutionary, spiral models.
- Emphasis on design.
  - Design-centric approach taken from outset.