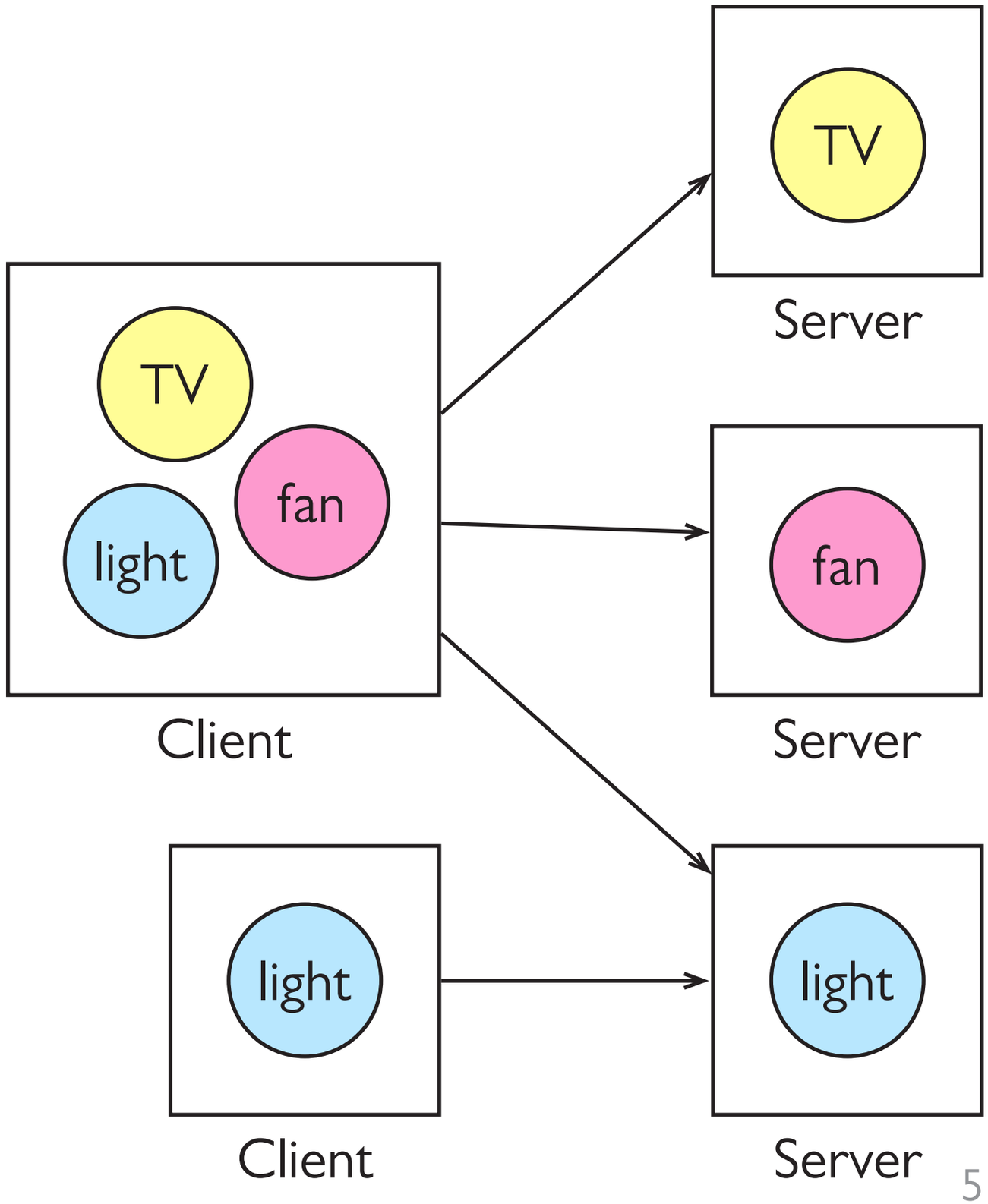
Hypermedia for IoT

Klaus Hartke

2017-03-10

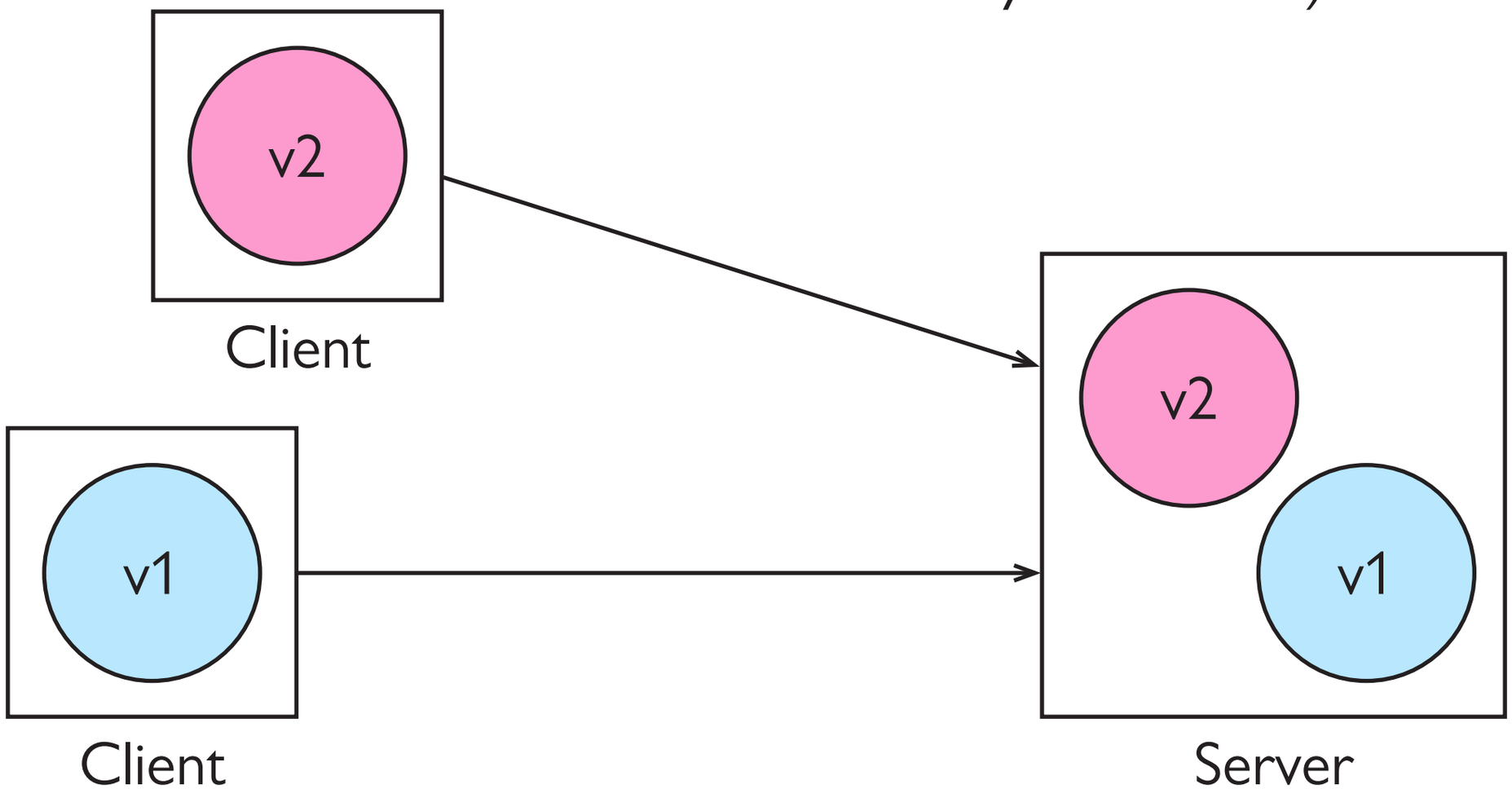
Resource Types

- **strategy:** modular API that allows clients to interact with servers that support the same modules
- shared *pre-knowledge* is identified by module names (e.g., resource types)
- supported modules are **discovered** using in-band information: clients have no pre-knowledge of what pre-knowledge each server has



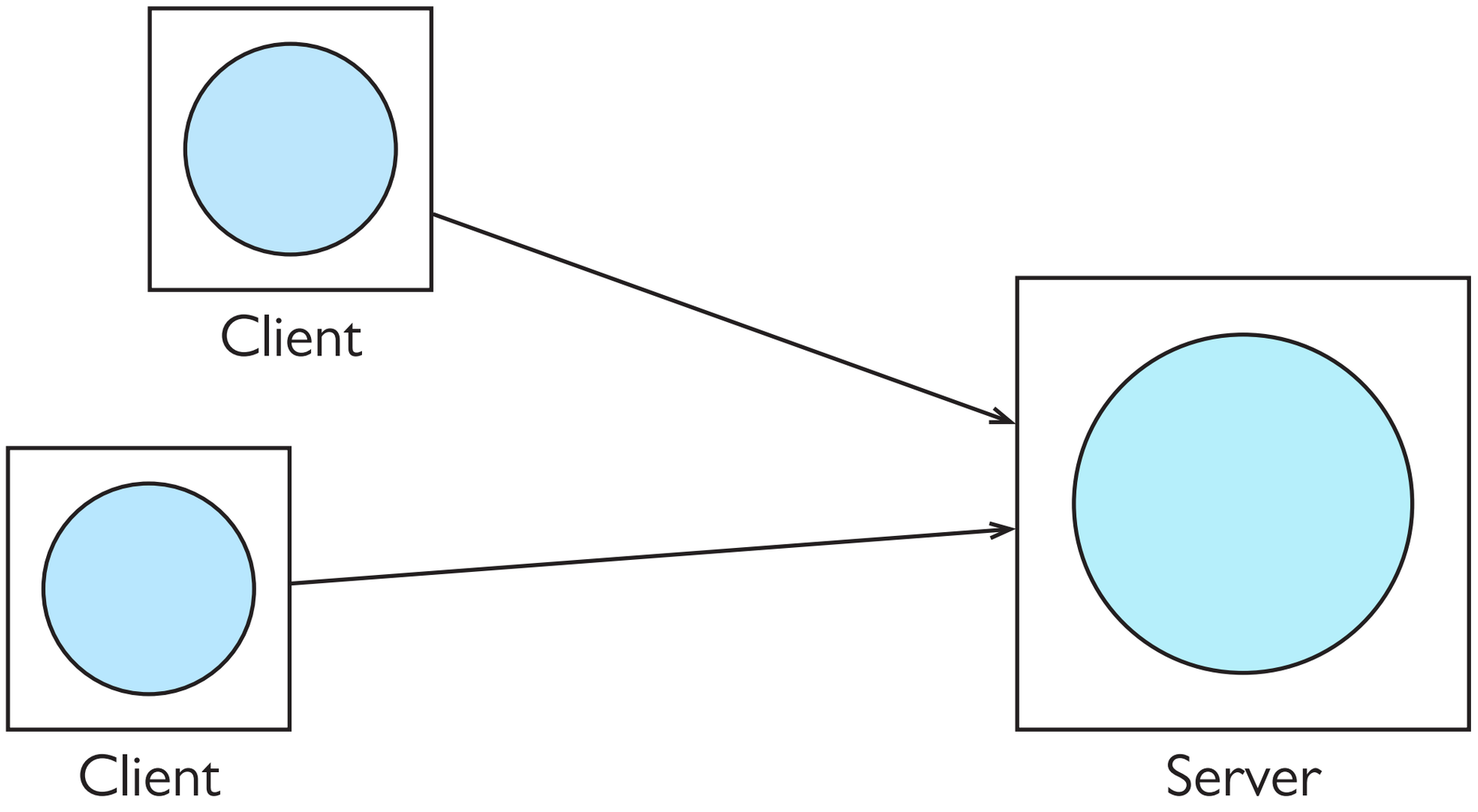
Versioning

- **strategy**: versioned API that allows older clients to run side-by-side with newer clients
- shared *pre-knowledge* is identified by version numbers
- works great if there is one server instance and multiple client implementations (e.g., Twitter API, GitHub API, …every REST API)



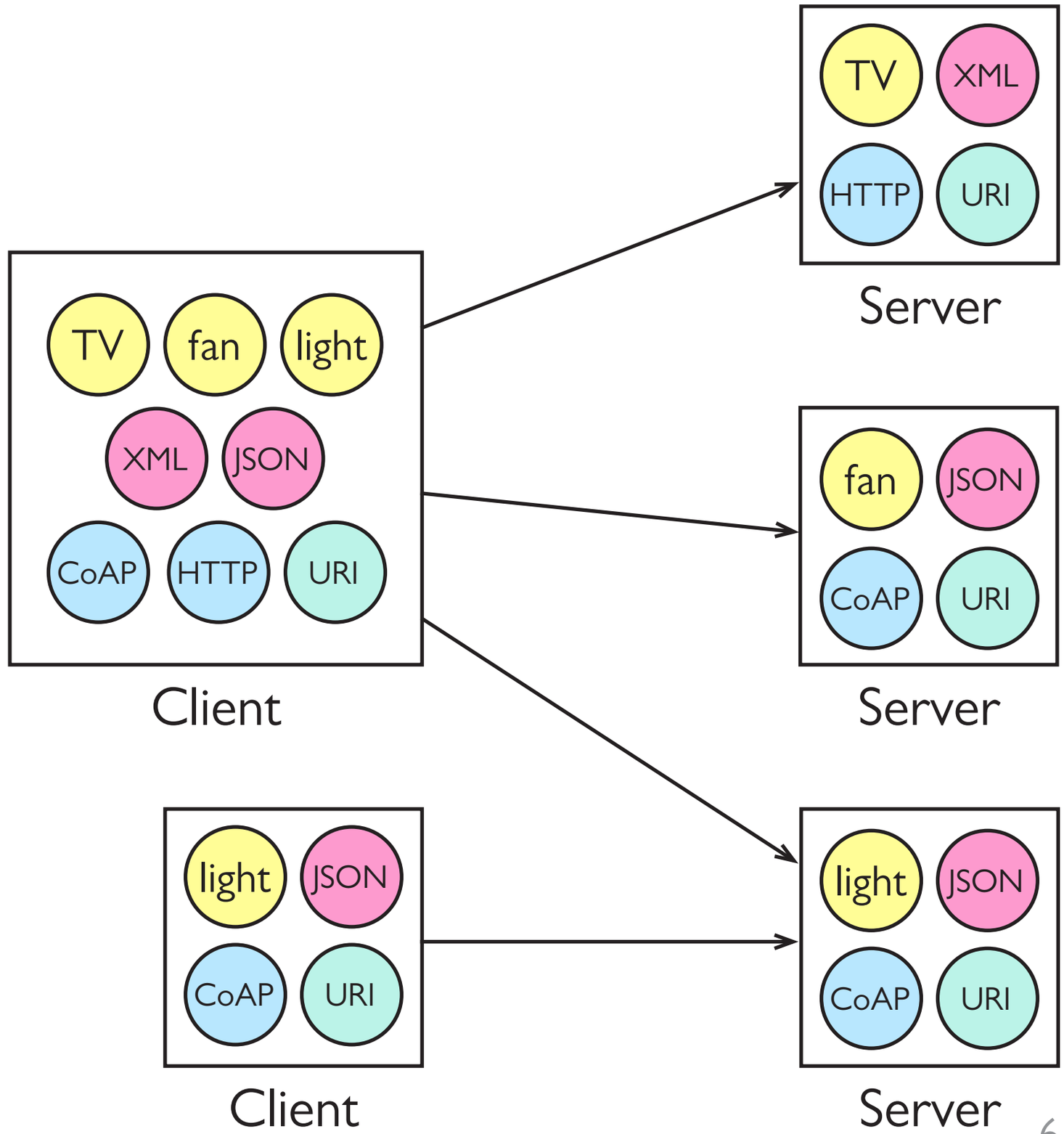
Code on Demand

- **strategy:** always keep all servers and clients updated
- single *pre-knowledge* shared by clients and servers
- works great if there is one server instance and one client implementation (e.g., online games, browser applications)



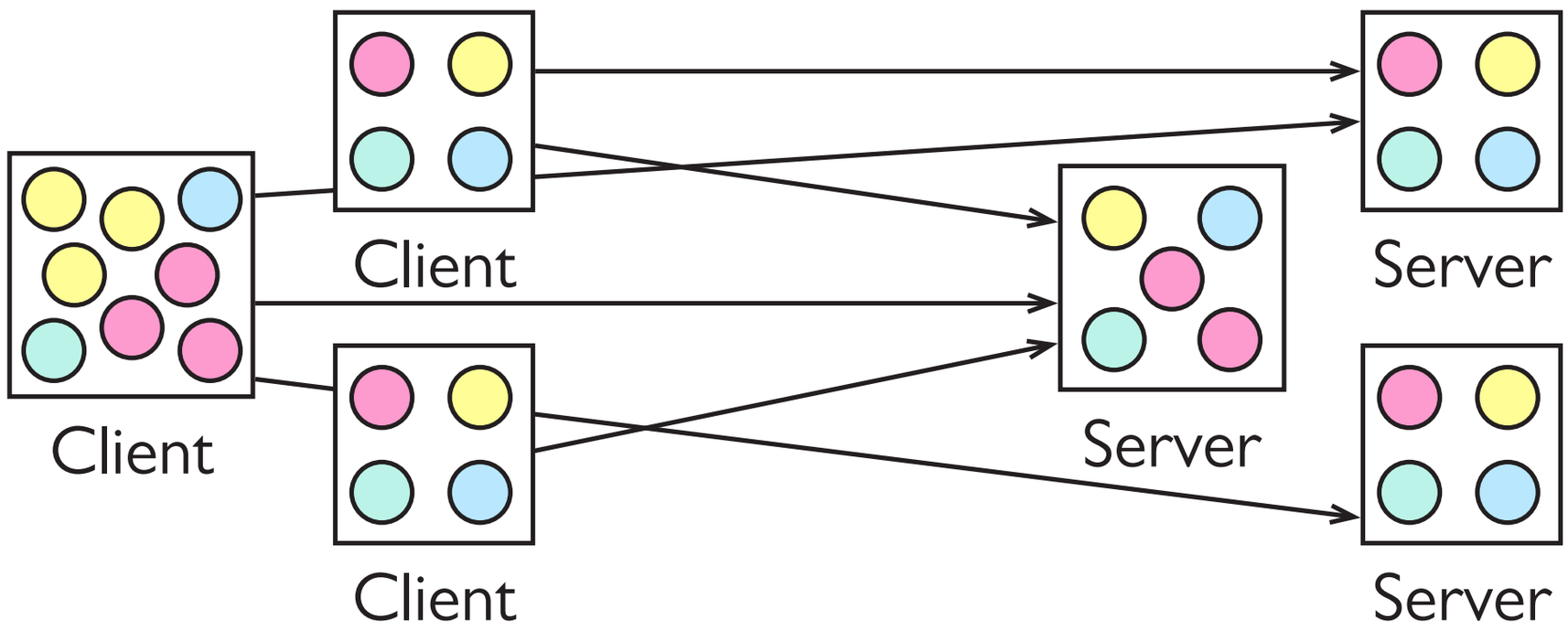
Orthogonal Specifications

- **strategy**: orthogonal components for identification, interaction and representation
- components may evolve independently without negatively impacting other components



Granularity

- **strategy**: fine-grained components



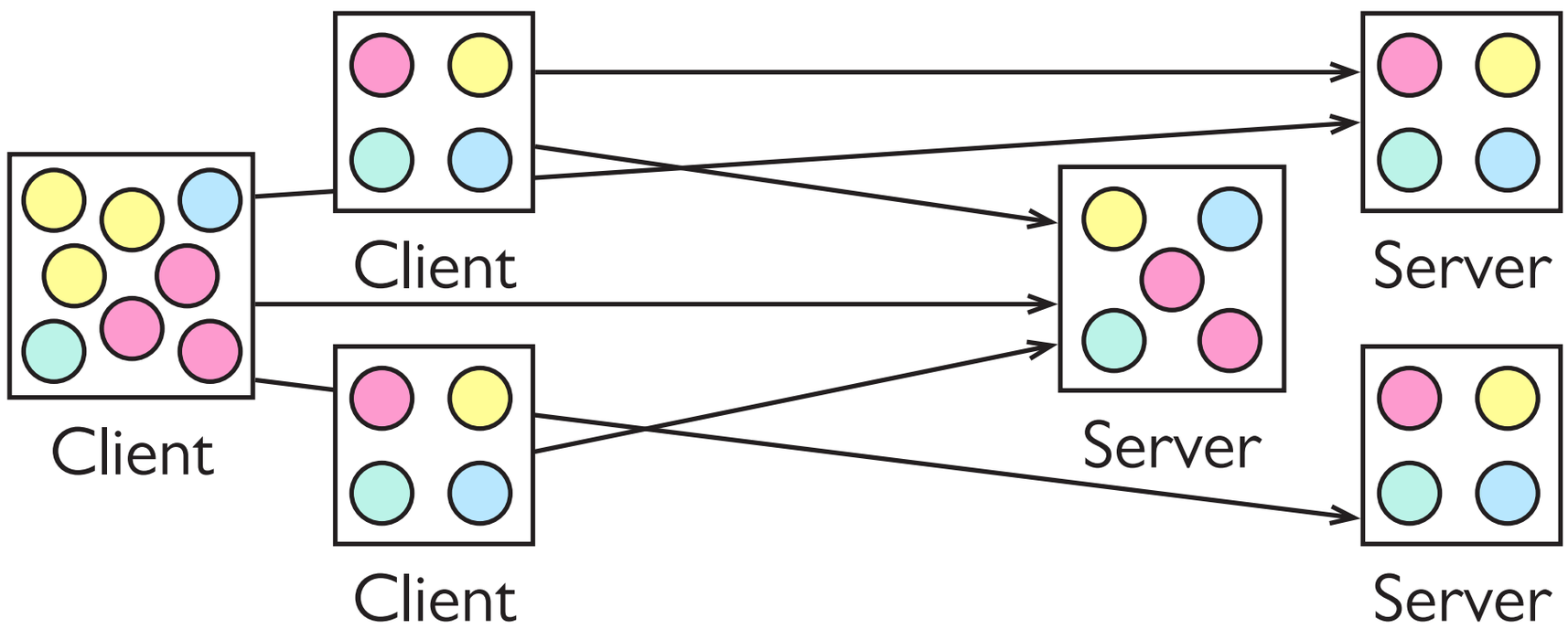
- make coffee
- water plants
- illuminate room
- …

- PUT
- POST
- PATCH
- …

- Celsius
- Fahrenheit
- RGB
- …

Granularity

- **strategy**: fine-grained components



- make coffee
  - ☐ with sugar
  - ☐ with milk
  - ☐ with rum

- illuminate room
  - ☐ romantic
  - ☐ arctic
  - ☐ sunset

How to encode this in-band information?

There's an RFC for that!

- URI schemes to identify protocols — RFC 7252
- URIs to identify resources — RFC 3986
- Links to discover the URIs of resources — RFC 5988
  (e.g., .well-known/core)
- Link attributes to indicate resource semantics
  (e.g., resource type, link relation type)
- Internet media types & CoAP content formats
  to identify data formats
- Link attributes to indicate supported data formats
- Hypermedia formats to encode links
  (e.g., CoRE Link Format) — RFC 6690

Formats for Hypermedia-driven APIs

- CoRE Link Format
  RFC 6690
- Hypertext Application Language (HAL)
  draft-kelly-json-hal-08
- Home Documents for HTTP APIs
  draft-nottingham-json-home-06
- Constrained RESTful Application Language (CoRAL)
  draft-hartke-core-apps-07 &
  draft-hartke-t2trg-coral-01

Differences:

- attaching semantics
- granularity
- link attributes, hints, metadata
- embedded data

CoRAL

Expressive

- Links
  - discover resources
  - semantics of resources (link relation type)
  - available data formats
  - metadata
- Forms
  - discover interactions
  - semantics of interactions ("form relation type")
  - supported parameters
  - accepted data formats
- Embedded data

Compact

- binary format based on CBOR
- clever default values
- CoAP-style URI encoding

Links and Forms can very often be expressed in a few bytes!

Additionally:

- extensible
- supports (almost) all of CoRE Link Format
- smaller than Link Format
- suitable for constrained environments

https://tools.ietf.org/html/draft-hartke-core-apps-07
https://tools.ietf.org/html/draft-hartke-t2trg-coral-01
https://tools.ietf.org/html/draft-hartke-t2trg-cbor-forms-00
https://tools.ietf.org/html/draft-hartke-t2trg-data-hub-00
https://github.com/ektrah/coral.net
https://github.com/ektrah/coral-c
https://github.com/ektrah/cirilla