03/04: Data oriented transfer

Decoupling of functionality
- allows revisiting of fn. split in traditional n/w.
- separation of content negotiation from content transfer.
- generic service design. with clearly defined interfaces.

Key motivation / observation.

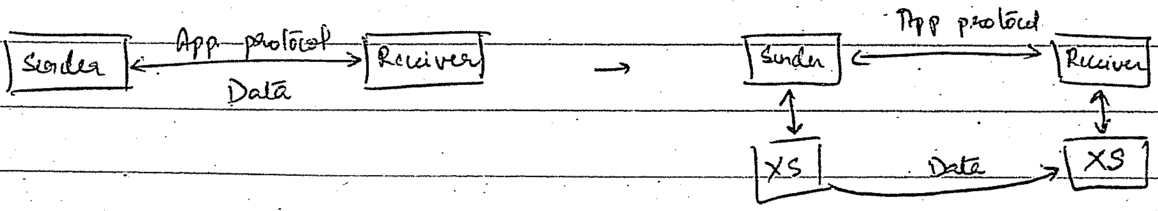
- Innovation in data transfer techniques is ~~past~~ hard
- Imagine you have a novel technique that you want to deploy and use
  - Modify HTTP / SMTP; talk to IETF; change appls.
  - long and painstaking process.

- Why? Applications bundled data transfer with application-specific content negotiation
  - Naming schemes
    ↳ URL, directories
  - Encodings etc.

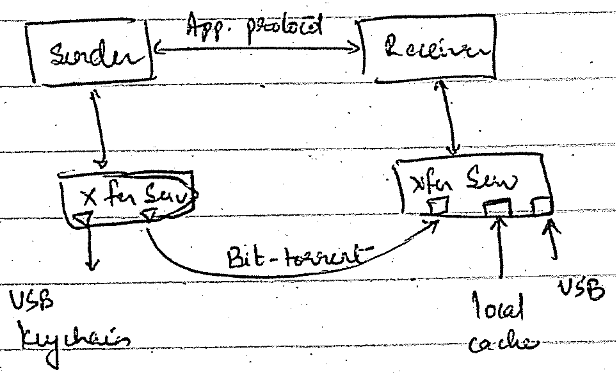
Data transfer itself is just the function of moving bits and it is common to many applications.

Tight coupling
- Hinders inno services
- Hinder new transport techniques.

Solution: A separate, generic data-transfer service that implements the app-independent parts as a separate service.



This enables an extensible transfer architecture.



① App - independent local cache

② New transports and n/ws.

③ Non- networked transfers.

Benefits → Apps. can reuse availble transfer techniques

→ Easy deployment of new " "

→ cross- app. sharing

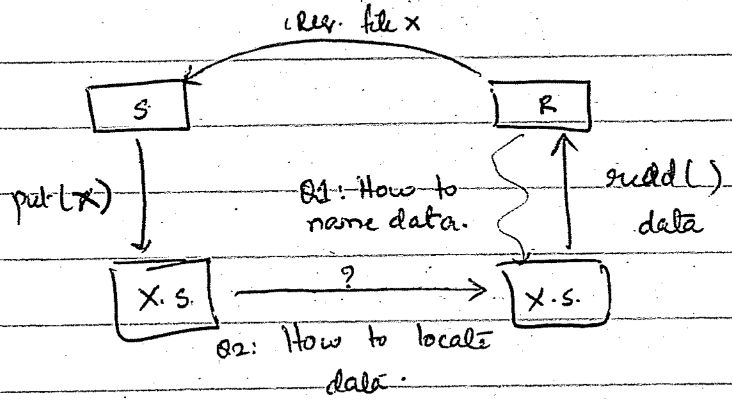
→ Handles transparent disconnection → multihoming → data oriented nature

→ Content delivery → data delivery accleration in a protocol indep. fashion.

→ Use any n/w technology

→ cross - application data processors, such as virus checkers.

10,000 ft view.



Q1: How to name data?

Host and app-independent content name

→ OID

Objects can be further sub-divided into chunks.

OID → list of chunk descriptors.

chunks → allow for partial xfers

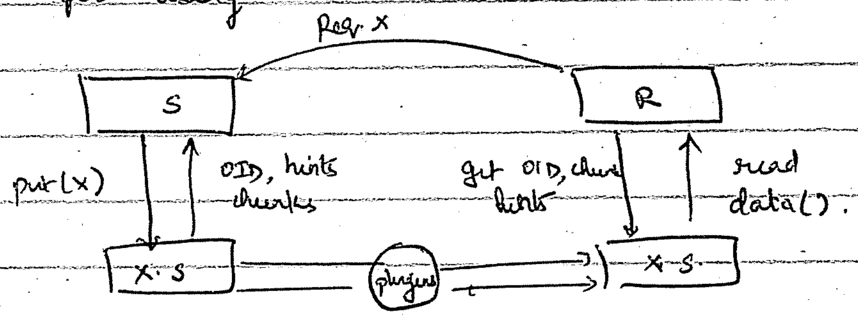
Object location: → ① data xfers are receiver driven

② sender provides hints

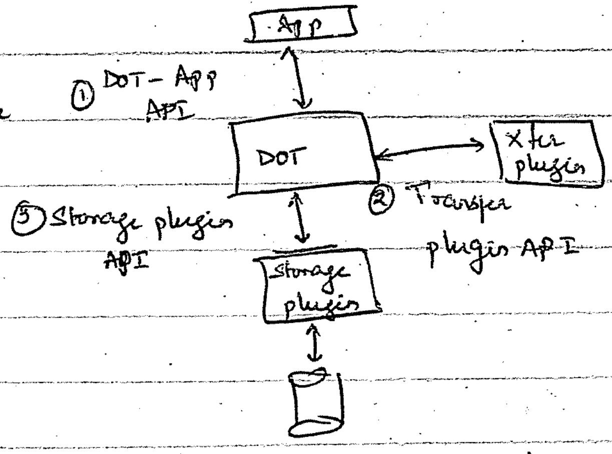
③ Receiver selects appropriate location(s) depending on local constraints

→ late binding; flexible adaptation; multipath; disconnection tolerance etc.

A transfer using DOT:



API and Modular architecture



① → put, get

② → get-descr. (hints) → URI (method) → plugin and params
get-chunks (hints)

priority → order of trying

weight → prob. of trying when priority is same

03/04: Data oriented Transfer

Decoupling of functionality
- allows revisiting of fn. split in traditional n/w.
- separation of content negotiation from content transfer
- generic service design. with clearly defined interfaces.

Key motivation / observation.

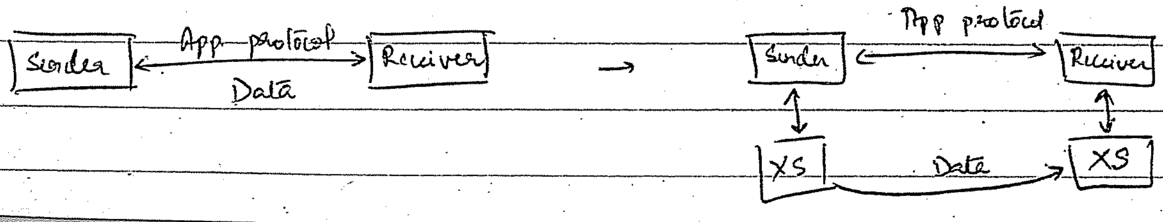
- Innovation in data transfer techniques is ~~past~~ hard
- Imagine you have a novel technique that you want to deploy and use
  - Modify HTTP / SMTP ; talk to IETF; change appls.
  - long and painstaking process.

- Why? Applications bundled data transfer with application-specific content negotiation
  - Naming schemes
    ↳ URL, directories
  - Encodings etc.

Data transfer itself is just the function of moving bits and it is common to many applications.

Tight coupling
- Hinders inno services
- Hinder new transport techniques

Solution : A separate, generic data-transfer service that implements the app-independent parts as a separate service.


Sender
App protocol
Data
Receiver
Sender
App protocol
Receiver
XS
Data
XS