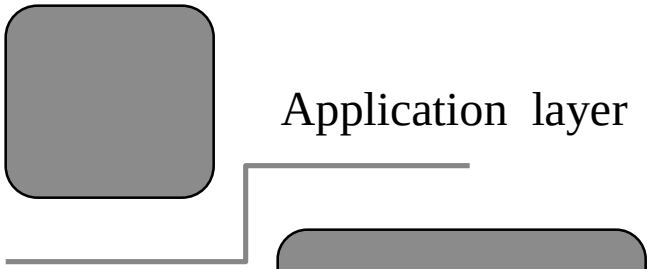
HTTP - Next Generation

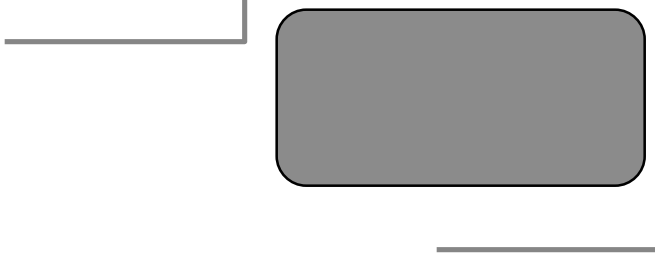
Architecture

Bill Janssen
Xerox PARC ISTL
25 August 1998

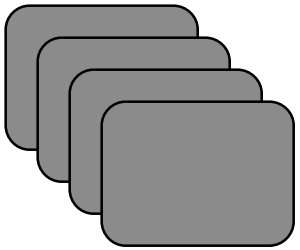
Three-layer Architecture



Application layer



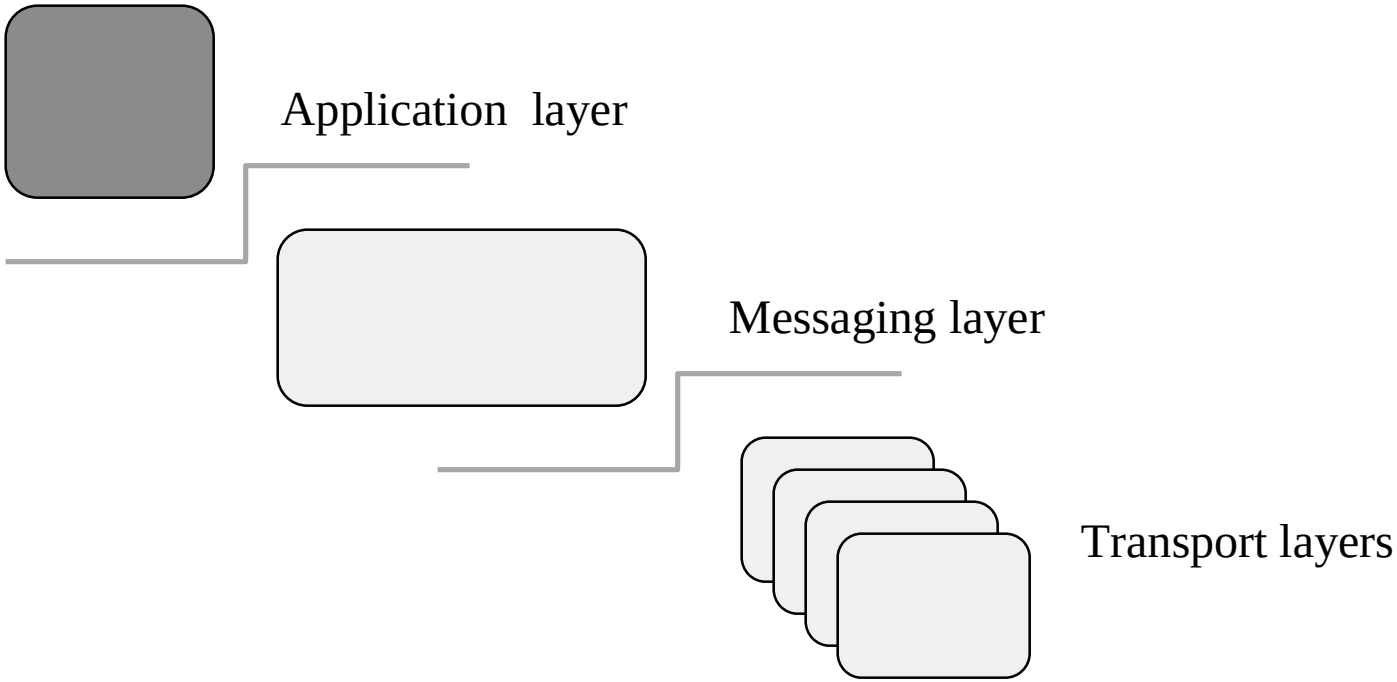
Messaging layer



Transport layers

Each layer is ***isolated*** from the other layers by small well-defined API

Application Layer



Application Layer

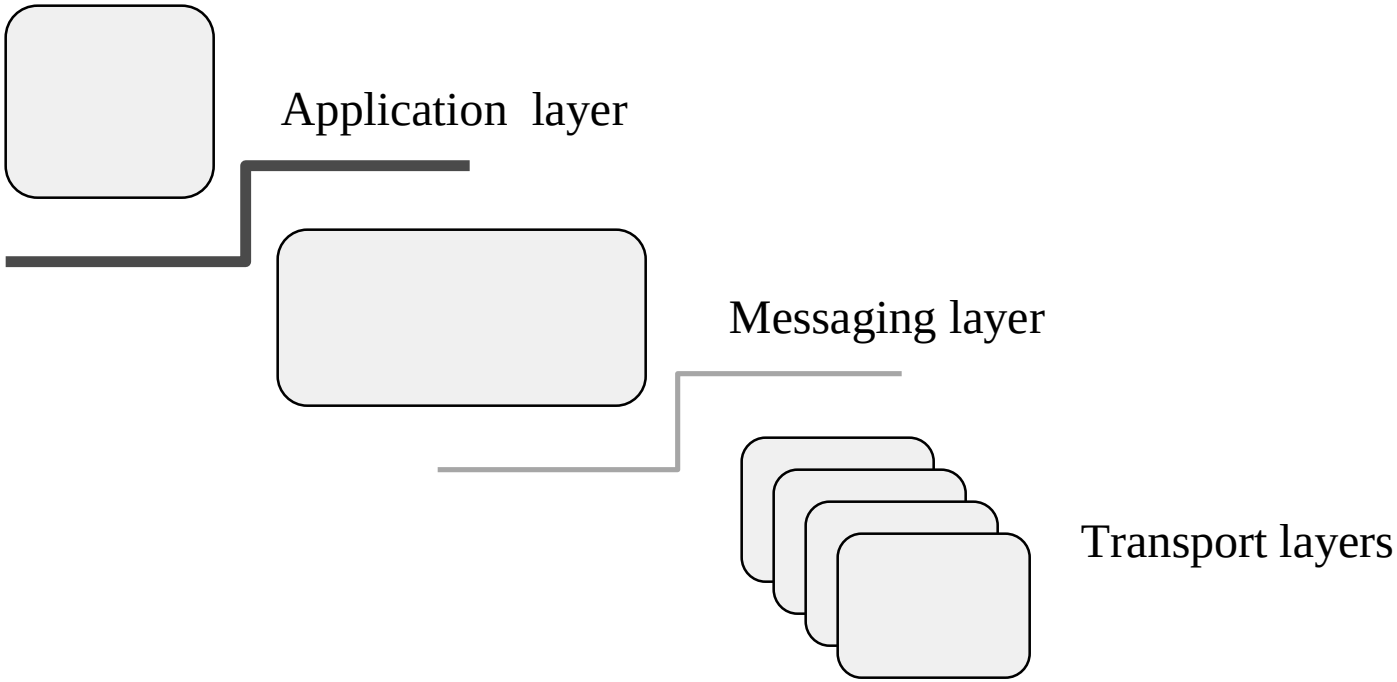
- Simple way of using HTTP-NG for any application (TCWA, IPP, WebDAV, …)
- Applications don’t interfere with each other
- Applications don’t modify wire protocol
- Operation set inheritance for backwards compatible extensibility
- Operation inheritance for anarchic evolution

The Web as Application

In some non-real IDL:

```
the-operation-set the-classic-web-app {
  (metadata h, entity-body eb, …) GET (opt-uri uri, …);
  (metadata h, …) HEAD (opt-uri uri, …);
  (metadata h, entity-body result, …) POST (…);
  …
  } inherits (HTTP-ng.RemoteObjectBase)
  id uuid:d340-a349c…-3340;
```

Application-to-Messaging API



Application-to-Messaging API

- Applications characterized by defining types and operations
- Standard messaging-layer type system used for definitions
- All applications appear to messaging layer as small set of messages and types

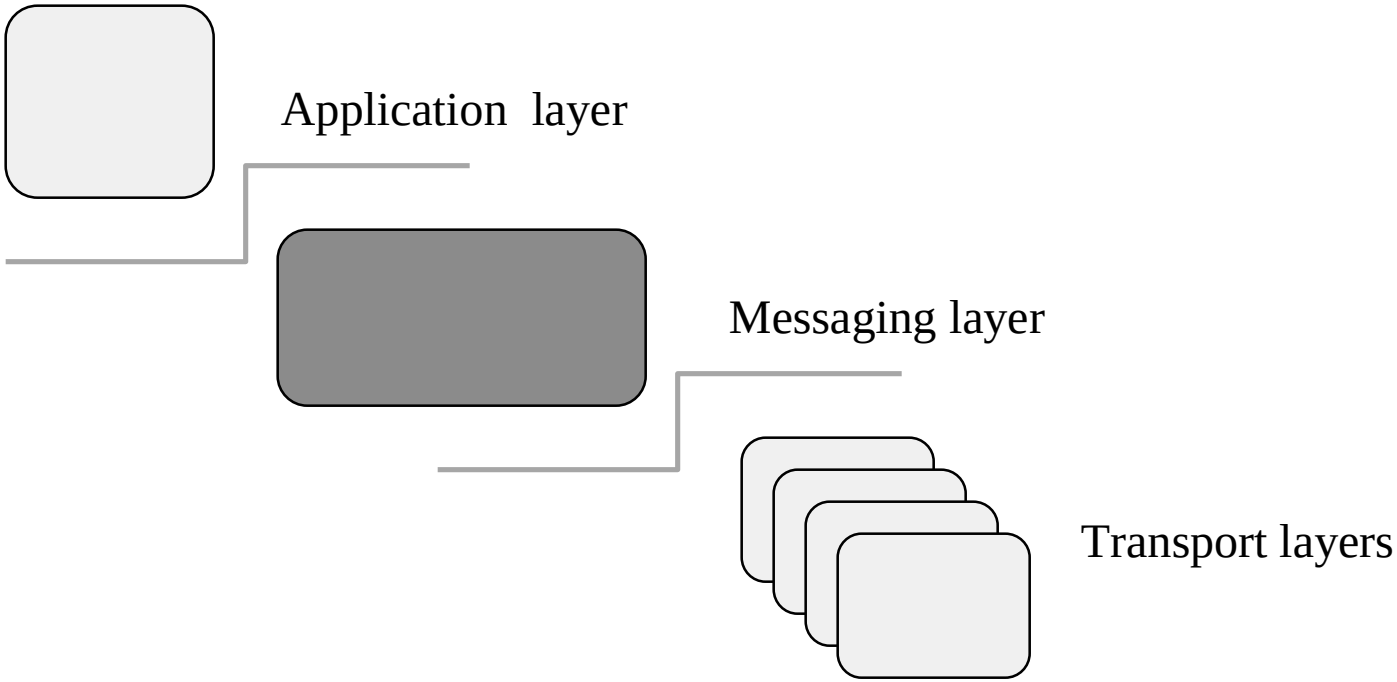
Type System

- Fixed-point & floating-point numerics
- Boolean, string types, enumeration types
- Sequences, arrays, unions (simplified)
- *Reference* (optional, aliased) pointer types
- *Pickle* type supports dynamic typing
- *Local object* (struct with local methods)
- *Remote object* (sets of remote methods)

Pre-defined Types

- HTTP-ng.TypeReference type provides description of any type, similar to CORBA typecode
- HTTP-ng.CollectibleObjectBase provides hooks for distributed garbage collection
- HTTP-ng.RemoteObjectBase distinguishes remote object types

The Messaging Layer



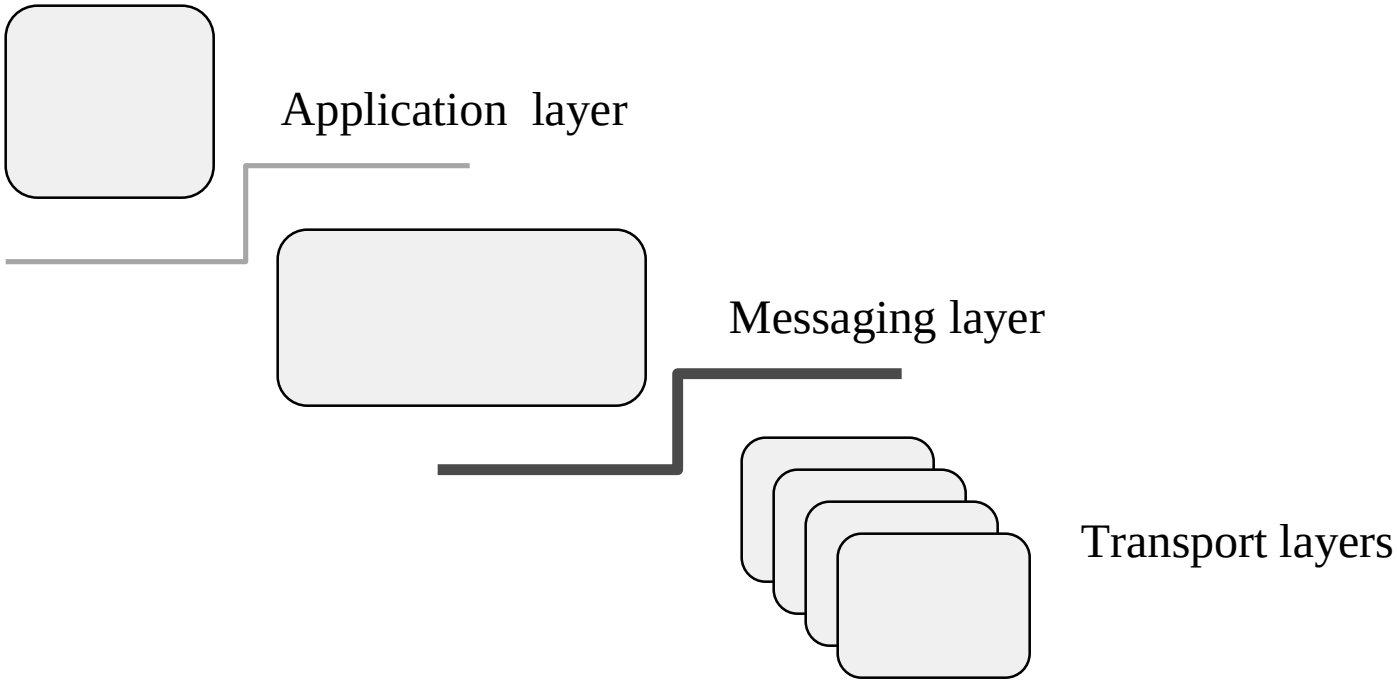
NG Messaging System

- Inherits lessons from previous systems:
  - X11, XNS Courier, ONC RPC, CORBA IIOP, DCOM, Java RMI, DCE RPC, TCP
- Exploits session context
  - persistent connections, pipelining, batching
  - caching of method and resource identifiers
  - caching of results (*functional* methods)
- Very efficient wire protocol

Wire Protocol

- Binary for efficiency
- Uses ONC RPC XDR for marshalling
  - dozens of implementations of XDR
  - IETF standards track protocol
- Discriminant & method caching yields 4-byte request & response messages
- hop-by-hop

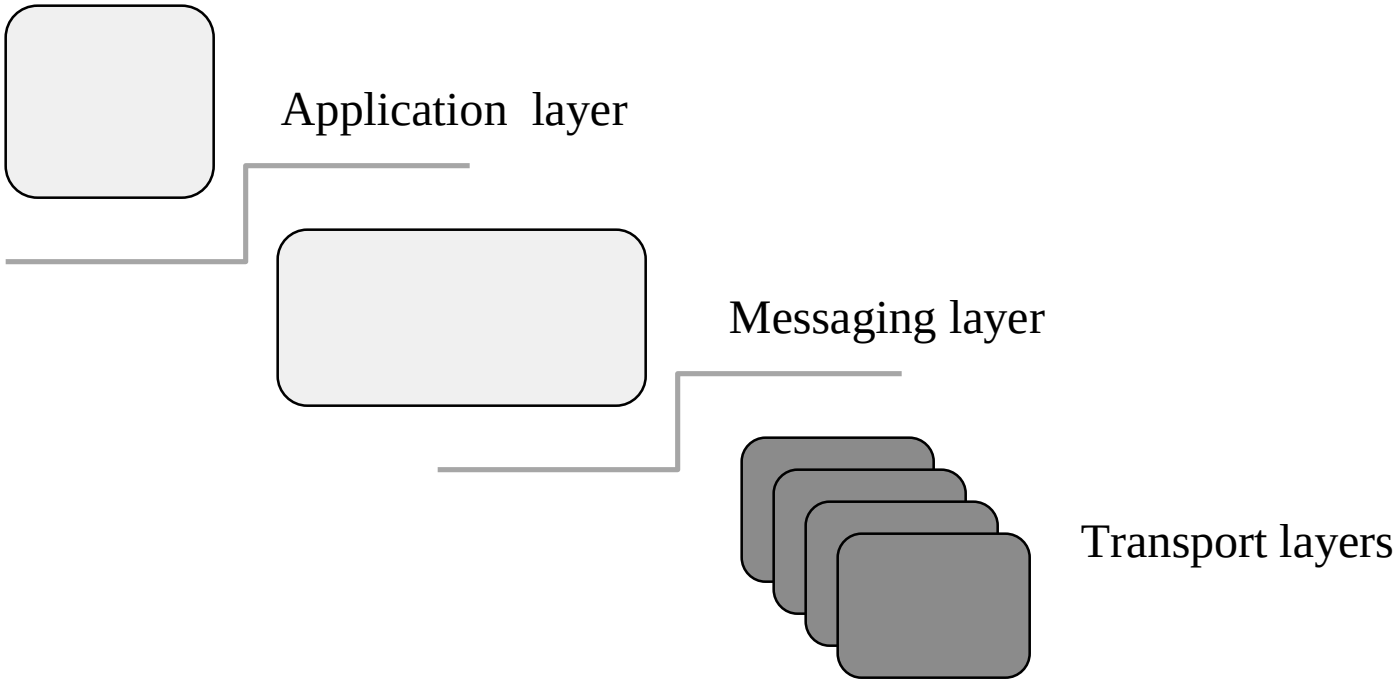
Messaging-to-Transport API



Messaging-to-Transport API

- Messages just blocks of data to transport layer
- Additional abstractions for connection control -- chunking, batching, security, proxying
- Identities and contexts important in transport layer

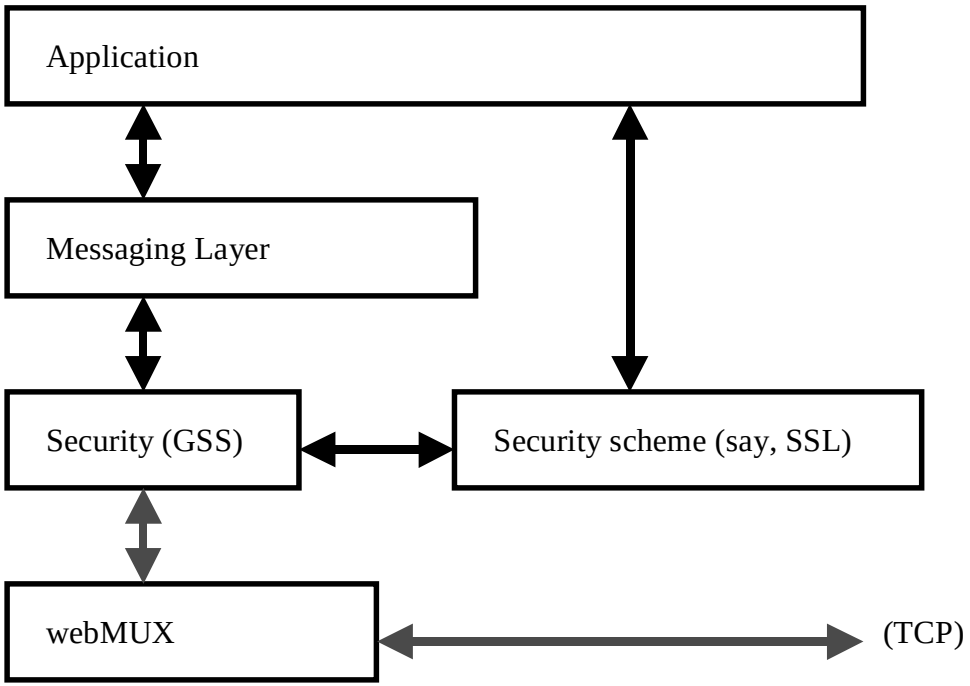
Transport Layers



Stacks of Transport Layers

- *Separate* layers instead of combining them
- Each layer transforms bytes received from next higher layer
- Examples
  - compression, security, record-marking, chunking, bi-directionality, multiplexing
- Run-time extensible

Security Transport Layer



WebMUX Transport

- Provides four transport layers in one:
  - record-marking
  - chunking
  - bi-directionality via endpoint identification
  - multiplexing of virtual connections
- Combining these layers provides more efficient headers