Distributed Systems
Principles and Paradigms

Maarten van Steen

VU Amsterdam, Dept. Computer Science
steen@cs.vu.nl

Chapter 02: Architectures

Version: September 3, 2012

Architectures

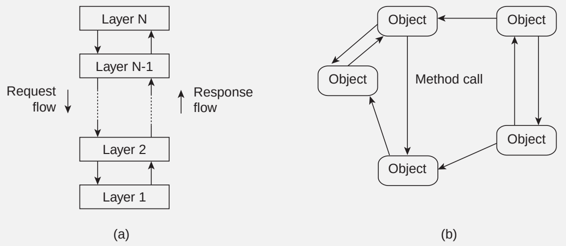
- Architectural styles
- Software architectures
- Architectures versus middleware
- Self-management in distributed systems

2 / 29

Architectural styles

Basic idea

Organize into logically different components, and distribute those components over the various machines.



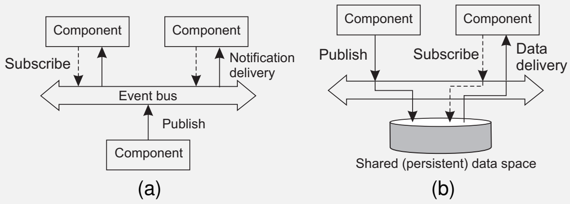
(a) Layered style is used for client-server system
(b) Object-based style for distributed object systems.

3 / 29

Architectural Styles

Observation

Decoupling processes in space ("anonymous") and also time ("asynchronous") has led to alternative styles.



(a) Publish/subscribe [decoupled in space]
(b) Shared dataspace [decoupled in space and time]

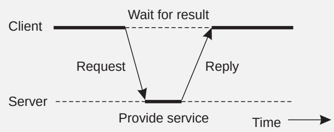
Architectures 2.1 Architectural styles

Centralized Architectures

Basic Client–Server Model

Characteristics:

- There are processes offering services (servers)
- There are processes that use services (clients)
- Clients and servers can be on different machines
- Clients follow request/reply model wrt to using services



Architectures 2.2 System Architectures

Application Layering

Traditional three-layered view

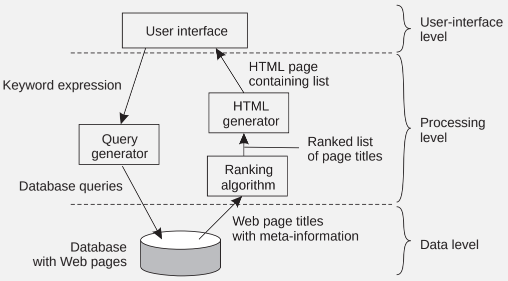
- User-interface layer contains units for an application's user interface
- Processing layer contains the functions of an application, i.e. without specific data
- Data layer contains the data that a client wants to manipulate through the application components

Observation

This layering is found in many distributed information systems, using traditional database technology and accompanying applications.

Architectures 2.2 System Architectures

Application Layering



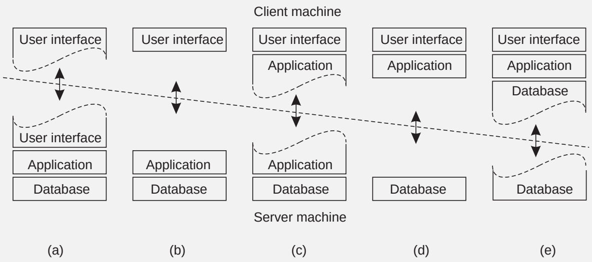
Architectures 2.2 System Architectures

7 / 29

Multi-Tiered Architectures

Single-tiered: dumb terminal/mainframe configuration
Two-tiered: client/single server configuration
Three-tiered: each layer on separate machine

Traditional two-tiered configurations:



Architectures 2.2 System Architectures

8 / 29

Decentralized Architectures

Observation

In the last couple of years we have been seeing a tremendous growth in peer-to-peer systems.

- Structured P2P: nodes are organized following a specific distributed data structure
- Unstructured P2P: nodes have randomly selected neighbors
- Hybrid P2P: some nodes are appointed special functions in a well-organized fashion

Note

In virtually all cases, we are dealing with overlay networks: data is routed over connections setup between the nodes (cf. application-level multicasting)

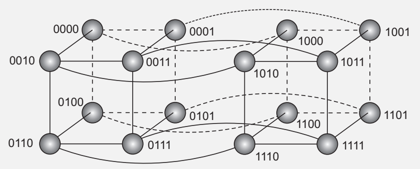
Architectures 2.2 System Architectures

9 / 29

Structured P2P Systems

Basic idea

Organize the nodes in a structured overlay network such as a logical ring, or a hypercube, and make specific nodes responsible for services based only on their ID.



Note

The system provides an operation *LOOKUP(key)* that will efficiently route the lookup request to the associated node.

Architectures 2.2 System Architectures

10 / 29

Unstructured P2P Systems

Essence

Many unstructured P2P systems are organized as a random overlay: two nodes are linked with probability *p*.

Observation

We can no longer look up information deterministically, but will have to resort to searching:

- Flooding: node *u* sends a lookup query to all of its neighbors. A neighbor responds, or forwards (floods) the request. There are many variations:
  - Limited flooding (maximal number of forwarding)
  - Probabilistic flooding (flood only with a certain probability).
- Random walk: Randomly select a neighbor *v*. If *v* has the answer, it replies, otherwise *v* randomly selects one of *its* neighbors. Variation: parallel random walk. Works well with replicated data.

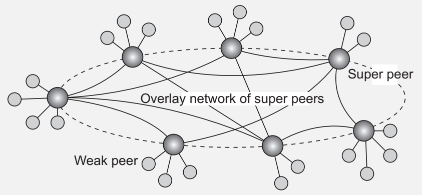
Architectures 2.2 System Architectures

11 / 29

Superpeers

Observation

Sometimes it helps to select a few nodes to do specific work: superpeer.



Examples

- Peers maintaining an index (for search)
- Peers monitoring the state of the network
- Peers being able to setup connections

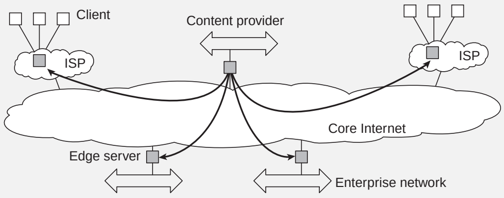
Architectures 2.2 System Architectures

12 / 29

Hybrid Architectures: Client-server combined with P2P

Example

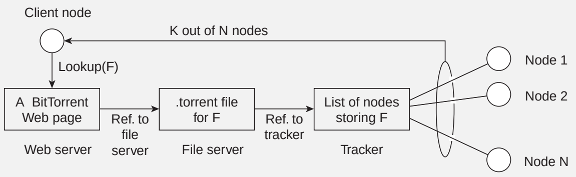
Edge-server architectures, which are often used for Content Delivery Networks



Architectures 2.2 System Architectures

13 / 29

Hybrid Architectures: C/S with P2P – BitTorrent



Basic idea

Once a node has identified where to download a file from, it joins a swarm of downloaders who in parallel get file chunks from the source, but also distribute these chunks amongst each other.

Architectures 2.2 System Architectures

14 / 29

Architectures versus Middleware

Problem

In many cases, distributed systems/applications are developed according to a specific architectural style. The chosen style may not be optimal in all cases $\Rightarrow$ need to (dynamically) adapt the behavior of the middleware.

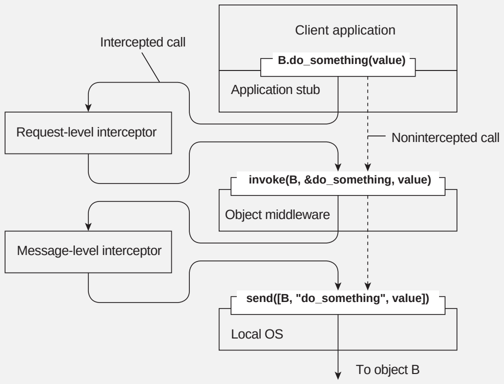
Interceptors

Intercept the usual flow of control when invoking a remote object.

Architectures 2.3 Architectures versus Middleware

15 / 29

Interceptors



Architectures 2.3 Architectures versus Middleware

Self-managing Distributed Systems

Observation

Distinction between system and software architectures blurs when automatic adaptivity needs to be taken into account:

- Self-configuration
- Self-managing
- Self-healing
- Self-optimizing
- Self-*

Warning

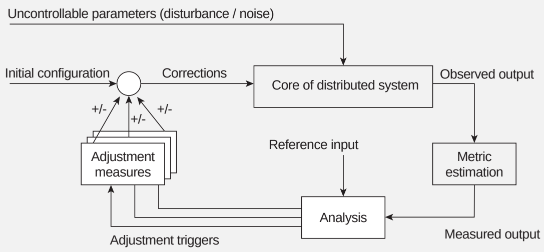
There is a lot of hype going on in this field of autonomic computing.

Architectures 2.4 Self-management in Distributed Systems

Feedback Control Model

Observation

In many cases, self-* systems are organized as a feedback control system.



Architectures 2.4 Self-management in Distributed Systems

Example: Globule

Globule

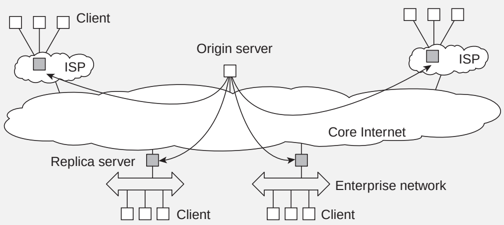
Collaborative CDN that analyzes traces to decide where replicas of Web content should be placed. Decisions are driven by a general cost model:

$$cost = (w_1 \times m_1) + (w_2 \times m_2) + \cdots + (w_n \times m_n)$$

Architectures 2.4 Self-management in Distributed Systems

19 / 29

Example: Globule



- Globule origin server collects traces and does what-if analysis by checking what would have happened if page P would have been placed at edge server S.
- Many strategies are evaluated, and the best one is chosen.

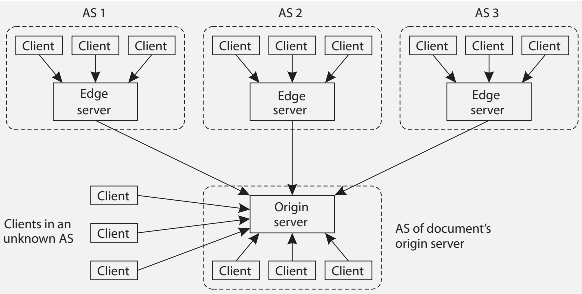
Architectures 2.4 Self-management in Distributed Systems

20 / 29

An experiment

Research question

Does it make sense to distribute each Web page according to its own best strategy, instead of applying a single, overall distribution strategy to all Web pages?



Architectures Extra: Strategy evaluation in Globule

21 / 29

An experiment

- We collected traces on requests and updates for all Web pages from two different servers (in Amsterdam and Erlangen)
- For each request, we checked:
  - From which autonomous system it came
  - What the average delay was to that client
  - What the average bandwidth was to the client's AS (randomly taking 5 clients from that AS)
- Pages that were requested less than 10 times were removed from the experiment.
- We replayed the trace file for many different system configurations, and many different distribution scenarios.

Architectures Extra: Strategy evaluation in Globule

An experiment

Issue	Site 1	Site 2
Start date	13/9/1999	20/3/2000
End date	18/12/1999	11/9/2000
Duration (days)	96	175
Number of documents	33,266	22,637
Number of requests	4,858,369	1,599,777
Number of updates	11,612	3338
Number of ASes	2567	1480

Architectures Extra: Strategy evaluation in Globule

Distinguished strategies: Caching

Abbr.	Name	Description
NR	No replication	No replication or caching takes place. All clients forward their requests directly to the origin server.
CV	Verification	Edge servers cache documents. At each subsequent request, the origin server is contacted for revalidation.
CLV	Limited validity	Edge servers cache documents. A cached document has an associated expire time before it becomes invalid and is removed from the cache.
CDV	Delayed verification	Edge servers cache documents. A cached document has an associated expire time after which the origin server is contacted for revalidation.

Architectures Extra: Strategy evaluation in Globule

Distinguished strategies: Replication

Abbr.	Name	Description
SI	Server invalidation	Edge servers cache documents, but the origin server invalidates cached copies when the document is updated.
SU*x*	Server updates	The origin server maintains copies at the *x* most relevant edge servers; *x* = 10, 25 or 50
SU50 + CLV	Hybrid SU50 & CLV	The origin server maintains copies at the 50 most relevant edge servers; the other intermediate servers follow the CLV strategy.
SU50 + CDV	Hybrid SU50 & CDV	The origin server maintains copies at the 50 most relevant edge servers; the other edge servers follow the CDV strategy.

Architectures Extra: Strategy evaluation in Globule

25 / 29

Trace results: One global strategy

Turnaround time (TaT) and bandwidth (BW) in relative measures; stale documents as fraction of total requested documents.

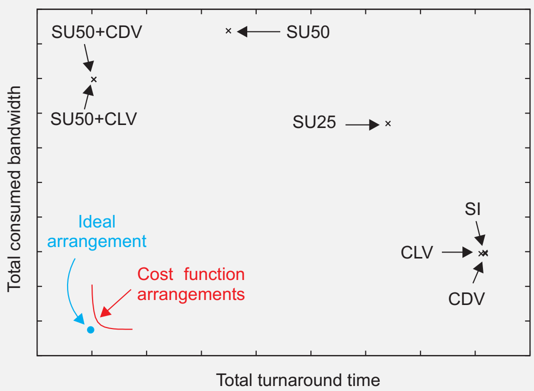
	Site 1			Site 2		
Strategy	**TaT**	**Stale docs**	**BW**	**TaT**	**Stale docs**	**BW**
NR	203	**0**	118	183	**0**	115
CV	227	**0**	113	190	**0**	**100**
CLV	182	0.0061	113	142	0.0060	**100**
CDV	182	0.0059	113	142	0.0057	**100**
SI	182	**0**	113	141	**0**	**100**
SU10	128	**0**	**100**	160	**0**	114
SU25	114	**0**	123	132	**0**	119
SU50	102	**0**	165	114	**0**	132
SU50+CLV	**100**	0.0011	165	**100**	0.0019	125
SU50+CDV	**100**	0.0011	165	**100**	0.0017	125

Conclusion: No single global strategy is best

Architectures Extra: Strategy evaluation in Globule

26 / 29

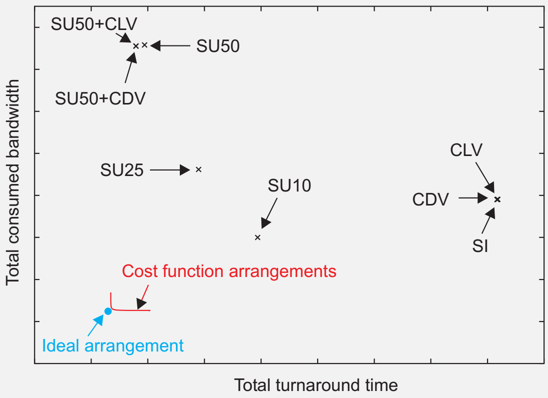
Assigning an optimal strategy per document: Site 1



Architectures Extra: Strategy evaluation in Globule

27 / 29

Assigning an optimal strategy per document: Site 2



Architectures Extra: Strategy evaluation in Globule

28 / 29

Useful strategies

Fraction of documents to which a strategy is assigned.

Strategy	Site 1	Site 2
NR	0.0973	0.0597
CV	0.0001	0.0000
CLV	0.0131	0.0029
CDV	0.0000	0.0000
SI	0.0089	0.0061
SU10	0.1321	0.6087
SU25	0.1615	0.1433
SU50	0.4620	0.1490
SU50+CLV	0.1232	0.0301
SU50+CDV	0.0017	0.0002

Conclusion: It makes sense to differentiate strategies

Architectures Extra: Strategy evaluation in Globule

29 / 29