DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Chapter 2
ARCHITECTURES
Architectural Styles (1)

Important styles of architecture for distributed systems

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures
Architectural Styles (2)

Figure 2-1. The (a) layered architectural style and ...
Figure 2-1. (b) The object-based architectural style.
Architectural Styles (4)

Figure 2-2. (a) The event-based architectural style and ...
Figure 2-2. (b) The shared data-space architectural style.
Centralized Architectures

Figure 2-3. General interaction between a client and a server.
Application Layering (1)

Recall previously mentioned layers of architectural style

- The user-interface level
- The processing level
- The data level
Figure 2-4. The simplified organization of an Internet search engine into three different layers.
Multitiered Architectures (1)

The simplest organization is to have only two types of machines:

- A client machine containing only the programs implementing (part of) the user-interface level
- A server machine containing the rest,
  - the programs implementing the processing and data level
Multitiered Architectures (2)

Figure 2-5. Alternative client-server organizations (a)–(e).
Multitiered Architectures (3)

User interface (presentation) → Wait for result → Application server → Wait for data → Database server → Return data → Return result

Figure 2-6. An example of a server acting as client.
Structured Peer-to-Peer Architectures (1)

Figure 2-7. The mapping of data items onto nodes in Chord.
Structured Peer-to-Peer Architectures (2)

Figure 2-8. (a) The mapping of data items onto nodes in CAN.
Structured Peer-to-Peer Architectures (3)

Figure 2-8. (b) Splitting a region when a node joins.
Unstructured Peer-to-Peer Architectures (1)

Actions by active thread (periodically repeated):

- select a peer P from the current partial view;
- if PUSH_MODE {
  - mybuffer = [(MyAddress, 0)];
  - permute partial view;
  - move H oldest entries to the end;
  - append first c/2 entries to mybuffer;
  - send mybuffer to P;
} else {
  - send trigger to P;
}
- if PULL_MODE {
  - receive P's buffer;
}
- construct a new partial view from the current one and P's buffer;
- increment the age of every entry in the new partial view;

Figure 2-9. (a) The steps taken by the active thread.
Unstructured Peer-to-Peer Architectures (2)

Actions by passive thread:

- receive buffer from any process Q;
  - if PULL_MODE {
    - mybuffer = [(MyAddress, 0)];
    - permute partial view;
    - move H oldest entries to the end;
    - append first c/2 entries to mybuffer;
    - send mybuffer to P;
  }
- construct a new partial view from the current one and P's buffer;
- increment the age of every entry in the new partial view;

Figure 2-9. (b) The steps take by the passive thread
Figure 2-10. A two-layered approach for constructing and maintaining specific overlay topologies using techniques from unstructured peer-to-peer systems.
Topology Management of Overlay Networks (2)

Figure 2-11. Generating a specific overlay network using a two-layered unstructured peer-to-peer system [adapted with permission from Jelasity and Baooglu (2005)].
Superpeers

Figure 2-12. A hierarchical organization of nodes into a superpeer network.

Tannenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved.
Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.
Collaborative Distributed Systems (1)

Figure 2-14. The principal working architecture of BitTorrent [adapted with permission from Pouwelse et al. (2004)].
Collaborative Distributed Systems (2)

Components of Globule collaborative content distribution network:

- A component that can redirect client requests to other servers.
- A component for analyzing access patterns.
- A component for managing the replication of Web pages.
Interceptors

Figure 2-15. Using interceptors to handle remote-object invocations.
General Approaches to Adaptive Software

Three basic approaches to adaptive software:

- Separation of concerns
- Computational reflection
- Component-based design
The Feedback Control Model

Uncontrollable parameters (disturbance / noise)

Initial configuration

Corrections

Core of distributed system

Observed output

Adjustment measures

Reference input

Metric estimation

Analysis

Measured output

Adjustment triggers

Figure 2-16. The logical organization of a feedback control system.
Example: Systems Monitoring with Astrolabe

<table>
<thead>
<tr>
<th>avg_load</th>
<th>avg_mem</th>
<th>avg_procs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.06</td>
<td>0.55</td>
<td>47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP-addr</th>
<th>load</th>
<th>mem</th>
<th>procs</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.2</td>
<td>0.03</td>
<td>0.80</td>
<td>43</td>
</tr>
<tr>
<td>192.168.1.3</td>
<td>0.05</td>
<td>0.50</td>
<td>20</td>
</tr>
<tr>
<td>192.168.1.4</td>
<td>0.10</td>
<td>0.35</td>
<td>78</td>
</tr>
</tbody>
</table>

Figure 2-17. Data collection and information aggregation in Astrolabe.
Example: Differentiating Replication Strategies in Globule (1)

Figure 2-18. The edge-server model assumed by Globule.
Example: Differentiating Replication Strategies in Globule (2)

Figure 2-19. The dependency between prediction accuracy and trace length.
Example: Automatic Component Repair Management in Jade

Steps required in a repair procedure:
• Terminate every binding between a component on a nonfaulty node, and a component on the node that just failed.
• Request the node manager to start and add a new node to the domain.
• Configure the new node with exactly the same components as those on the crashed node.
• Re-establish all the bindings that were previously terminated.