Lecture 33 — Software Communication Patterns

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

December 29, 2014

Software Communication Patterns

Software Architecture, Continued.

We will now look at communication architectures:
how messages and data are passed.

Pipes and Filters Architecture

Two major components: pipes and filters.

Pipe: Takes input, performs some action, and outputs result.

Take one input, do one thing, produce one output.

Filter: Examines input and will either accept it or reject it.
- Accept: let the data pass and appear in the output.
- Reject: block the data; no output appears.

Pipes and Filters Architecture

Example: input messages are formatted with CRLF.

Data will be sent to a system that recognizes only LF as the end of line and considers CR to be an error.

Solution: create a pipe that removes the CR characters.

Alternative: create a filter. Reject messages with CR.

Pipes and Filters Architecture

Pipes do not have to transform data.

Might have a logging pipe.

Filters are read-only; they do not change data.

Pipes and Filters Architecture

Pipes and filters can be assembled into a *Pipeline*.

Data enters at the start of the pipeline and passes through the pipes and filters until it gets to the end.

Filters can be at any point.

If a filter rejects a piece of data, it does not advance.

Build a complex system by assembling smaller components.

Pipes and Filters Architecture

Individual elements are small and simple.

Many components in a system - hidden dependencies?

Not much room for decision-making in processing.

Pipes and Filters Architecture

Sometimes a third element: *valve*.

Valves are 1 input : 1+ outputs.
    (Pipes are 1 input : 1 output.)

A way to decide farther down how to proceed.

Example: send the message via e-mail or instant message.

Pipes and Filters Example

We have a system that accepts XML data within a ZIP file and delivers the extracted XML to a server.

A pipeline for such a system might be written like this:

validZIPFileFilter/extractFromZIPFilePipe/
loggingPipe/removeCRPipe/syntaxFilter/deliverToServerPipe

A slash indicates separation between segments of the pipeline.

Pipes and Filters Example

This example has four pipes and two filters.

- **validZIPFileFilter**
- **extractFromZIPFilePipe**
- **loggingPipe**
- **removeCRPipe**
- **syntaxFilter**
- **deliverToServerPipe**

Pipes and Filters Example

UNIX example:

The system utilities like `ls`, `cat`, and so on are all small and do exactly one thing (like pipes).

On the commandline you can create a pipeline connecting these two or more items.

```
cat lectures.txt | grep blackboard
```

Blackboard Architecture

Blackboard: Common knowledge base.

Independent agents work on the knowledge base.

Agents co-operate to solve the problem.

Blackboard Architecture Analogy

Imagine a difficult math problem.
To solve it, assemble a group of math professors.

The processors will collaborate using a physical blackboard.

Each watches to see if he/she can apply his/her expertise.

When a professor can contribute, he/she writes on the board.

Blackboard Architecture Analogy

After writing something, hopefully someone else can now act.

Professors continue contributing until the problem is solved.

But perhaps nobody can proceed; then professors have failed.

Blackboard Architecture

Three major components in a Blackboard system.

Software agents or *Knowledge Sources* (professors).

Blackboard: shared repository of problems & partial solutions.

Moderator: controls access so agents do not interfere.
(Moderator stops profs from fighting over the chalk.)

Blackboard Architecture

Unusual architecture; few domains where it is best choice.

Usually one agent is best for a problem.

Example: Optical Character Recognition (OCR).

Example: Artificial Intelligence.

Blackboard Architecture

Advantage: can solve problems that are otherwise intractable.

Advantage: might be more efficient or faster.

Disadvantage: complex system, difficult to understand.

Disadvantage: limited applicability.

Publish-Subscribe Architecture

Message passing system.

The sender of a message is called a *Publisher*.

The publisher does not know whom the reciever(s) of the message is (are).

Receivers are called *Subscribers*.

Subscribers register interest in certain kids of messages without knowledge of any publishers.

Publish-Subscribe Architecture

There may be a third party involved, called a *Broker*.

Brokers connect the publishers and subscribers indirectly.

If there is no Broker, any subscriber who is not “listening” at the time the message is published will not receive the message.

Broker is a place where the publisher can publish the message so subscribers can collect the message at their convenience.

Publish-Subscribe Architecture

Subscribers are usual not interested in all messages.

Two ways to filter, not mutually exclusive:

Topic-Based: topics/channels.
  Example: ECE 155 topic.

Content-Based content-matching.
  Example: any message with the text “ECE 155”.

Note publisher vs. subscriber filtering responsibility.

Publish-Subscribe Architecture Advantages

- No dependency between publishers & subscribers.
- Scales well; we can have m publishers & n subscribers.
- Broker allows asynchronous communication.

Publish-Subscribe Architecture Disdvantages

- No certainty a message was delivered.
- Publishers might assume subscribers are listening even if they are not.
- Broker can be a central point of failure.

Publish-Subscribe Example

RSS (Really Simple Syndication) feeds use publish-subscribe.

The author publishes his writing on the blog.

Subscribers to the RSS feed get a notification.

This new message contains the published content.

Peer-to-Peer Architecture

Different nodes (computers or programs) in the network both supply and receive resources.

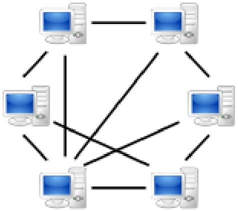
Contrast with client-server model where the client requests resources that the server provides.

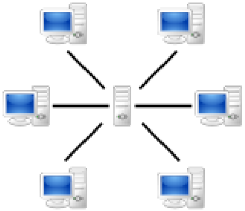
Tasks like searching for files or streaming audio/video are shared amongst the interconnected peers.

Peers contribute some resources (e.g., processing power).

No centralized server.

Peer-to-Peer Architecture





Left side: A P2P network with a central admin system;
Right side: a network based on the client-server model.
`http://en.wikipedia.org/wiki/Peer-to-peer`

Peer-to-Peer Example: Napster

Napster was a P2P file transfer system, used to share music.

There was no central repository of music files;
each user shared what files he had locally;
could browse the files others made available.

P2P systems are commonly associated with sharing copyrighted material (music, movies);
the P2P network is just the form of communication.

Linux distributions like Ubuntu provide an option to download their installation CDs via BitTorrent.

P2P Advantages & Disadvantages

Advantages:

- Ability to serve content increases with users.
- Decentralized; no single point of failure.

Disadvantages:

- Communication and search overhead is high.
- Co-ordination as peers enter and leave the network.
- No guarantee of finding what is sought, even if available.

P2P: Distributed Systems

If peer-to-peer networks are a subject of interest for you, take the class “Distributed Systems”