


Distributed software
Introduction

CSCI E 45a: The Cyber World—part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

Learning goals



- Understand how collaborating pieces of software make today's most complex systems possible
- Understand the history and evolution of the technologies involved


2 Copyright © Scott Bradner & Ben Gaucherin 2015

Distributed software systems

- Multiple hardware/software components
- Working together (using the network)
- Appearing to the user as a single coherent system
- To distributed software systems, a network is assumed

3 Copyright © Scott Bradner & Ben Gaucherin 2015


Examples of distributed systems



- Large enterprise systems
HR and financial systems, Student Information System, supply chain management systems, etc.
- Complex websites
e-commerce sites, travel booking, banking, etc.
- The Internet, the Web
- Distributed people systems
Amazon's mechanical turk, distributed call centers, etc.

4 Copyright © Scott Badner & Ben Gaucherin 2015

Examples of distributed systems, contd.



- Very large computing problems to solve
- And the problem can be broken into many small, simple, like problems
Computer graphics for movie effects, financial markets simulations, weather models/simulations, encryption cracking, finding aliens – SETI@Home

5 Copyright © Scott Badner & Ben Gaucherin 2015

Big data, distributed processing



- Recent term describing incredibly large data sets and the ability to infer new information/insight from these enormous sets of data
- The 3 "V's":
Volume, Velocity, Variety

6 Copyright © Scott Badner & Ben Gaucherin 2015

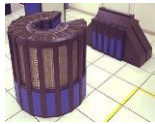
Big data, distributed processing, contd.

- Tools for big data
 - Google's MapReduce
 - Map – queries are split and distributed across parallel nodes
 - Reduce – results are gathered
 - Apache's Hadoop

7

Copyright © Scott Bradner & Ben Gaucherin 2015

Why distributed systems?



- When a single (even if very powerful) machine can't meet all of the requirements

8

Copyright © Scott Bradner & Ben Gaucherin 2015

Why distributed systems?




- Scalability & performance
 - To support large/dynamic populations of concurrent users
 - To manage large/complex processes, computations, and sets of data
 - To better adjust resources to what is needed
- Resiliency, Availability
- And a few other things:
 - Modularity, sharing of resources, incremental change, etc.

9

Copyright © Scott Bradner & Ben Gaucherin 2015

Topics


- Key terms and concepts - R
What are the key elements and characteristics of distributed systems
- Basic mechanics - R
How does it work "under the hood"
- Evolution of architectures - R
From mainframes, to client/server, to web applications and services



10 Copyright © Scott Bradner & Ben Gaucherin 2015

Topics

- Sample website - R
Using a fictitious sample website to illustrate some of the common elements of today's large distributed systems



11 Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
4	Oracle, SAP, PeopleSoft, Expedia, Amazon Mechanical Turks, eBay, Bank of America, Netflix logos
5	http://2.bp.blogspot.com/YHM1c1nlgkw/UM325Y_uj/AAAAAAABrQ/ALvRIEkY3Ys1600/jt-r-crossword-10-donation_de_sign.png
5	"Setiathomeversion5point15". Licensed under LGPL via Commons https://commons.wikimedia.org/wiki/File:Setiathomeversion5point15.png#/media/File:Setiathomeversion5point15.png
6	Big Data word cloud http://billterrain.net/wp-content/uploads/2014/04/big-data.jpg
8	"Cray2" by NASA - http://gimp-savvy.com/cgi-bin/img.cgi?alawE7kml1216740 . Licensed under Public Domain via Commons https://commons.wikimedia.org/wiki/File:Cray2.jpeg#/media/File:Cray2.jpeg
9	Oracle, SAP, PeopleSoft, Expedia, Amazon Mechanical Turks, eBay, Bank of America, Netflix logos

12 Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

10 Napster and JSON logos

10 "Televideo925Terminal". Licensed under Public Domain via

Commons -

<https://commons.wikimedia.org/wiki/File:Televideo925Terminal.jpg#/media/File:Televideo925Terminal.jpg>

11 healthcare.gov logo

13

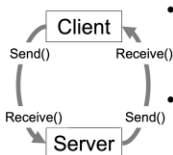
Copyright © Scott Bradner & Ben Gaucherin 2015

Distributed software
Key terms and concepts

CSCI E 45a: The Cyber World—part A

Copyright © Scott Bradner & Ben Gaucherin 2015 1


Client, Server, or both?



- **Client**
The machine/software from which requests originate
- **Server**
The machine/software fulfilling the request
- A machine can be (and often is) both client and server
The role is context dependent

Copyright © Scott Bradner & Ben Gaucherin 2015 2

Client, Server, or both?



- **Peer to Peer** – machines are both client and server
No central server system in the normal operations, although one may be needed to facilitate session setup

Napster, BitTorrent, etc.

Copyright © Scott Bradner & Ben Gaucherin 2015 3

Characteristics

- Support **physical separation of components**
Components interconnected by a network
- Support **scalability**
By distributing the load on multiple components, and allowing the dynamic addition of components to handle a bigger load

Copyright © Scott Bradner & Ben Gaucherin 2015

4

Characteristics, contd.

- Support **administrative autonomy**
Multiple boxes, multiple administrators
- Support **heterogeneity**
No requirement to use consistent hardware/software

Copyright © Scott Bradner & Ben Gaucherin 2015

5

Challenges

- **Security**
Ensuring confidentiality, integrity, and availability
What is the trust model between components?
- **Manageability**
Getting a “system view”, and configuration consistency
Consistent management of individual components, by different people

Copyright © Scott Bradner & Ben Gaucherin 2015

6

Challenges, contd.

- **Indeterminacy**
Things will break (network, services, etc.) and the components need to handle failure (e.g. time outs, retries, etc.)

Copyright © Scott Bradner & Ben Gaucherin 2015

7

The 8 fallacies of distributed computing



Peter Deutsch



James Gosling

- The network is reliable
 - Latency is zero
 - Bandwidth is infinite
 - The network is secure
 - Topology doesn't change
 - There is one administrator
 - Transport cost is zero
 - The network is homogeneous
- 1994 Peter Deutsch, 1997 James Gosling

Copyright © Scott Bradner & Ben Gaucherin 2015

8

Language matters!



- Offensive:
- Master/Slave
 - White list/black list

- Not offensive:
- Master of Arts
 - Mastery

- Some language used in technology is finally being recognized as offensive
- Time to shift to more inclusive language
Terminology, Power, and Inclusive Language in Internet-Drafts and RFCs draft-knodel-terminology-03

Copyright © Scott Bradner & Ben Gaucherin 2020

9

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

3 Napster, BitTorrent logos

8 <http://www.wheels.org/spacewar/stone/deutsch.jpg>

8 "James Gosling 2008" by Peter Campbell - self-made, Nikon D80.

Licensed under GFDL via Commons -

https://commons.wikimedia.org/wiki/File:James_Gosling_2008.jpg#/media/File:James_Gosling_2008.jpg

9 rootsofjusticetraining.org/2014/04/linguistic-racism/

Copyright © Scott Bradner & Ben Gaucherin 2020

10

Distributed software
Basic mechanics

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

Remote Procedure Call (RPC)

```
graph TD; Client[Client] -- Send() --> Server[Server]; Server -- Receive() --> Client;
```

- Your code calls a function that appears to be “local” – **client stub**
- The function call is “marshaled” into a **message**
Marshaling: package up for transmission
- The message is routed to the actual location of the code for the function called – the **server**

2 Copyright © Scott Bradner & Ben Gaucherin 2015

Remote Procedure Call (RPC), contd.

```
graph TD; Client[Client] -- Send() --> Server[Server]; Server -- Receive() --> Client;
```

- The message is “**un-marshaled**” and the function executed
- The result is “**marshaled**” into a response message and sent back to the client
- The client receives the response message which is “**un-marshaled**” processed as the response to the function call

3 Copyright © Scott Bradner & Ben Gaucherin 2015

Remote Procedure Call (RPC), contd.

Different types of RPC:
Synchronous, asynchronous, callbacks

4 Copyright © Scott Bradner & Ben Gaucherin 2015

Latency of remote calls

- **Latency**: time delay experienced by a system
- Trade off between:
 - local processing latency
 - Local + network + remote processing latency
- The further you go, the bigger the time penalty
 - Intra-process
 - Inter-process – same host
 - Other host on LAN – interrupt latency on target machine
 - Other host on WAN – network buffering

5 Copyright © Scott Bradner & Ben Gaucherin 2015

A couple of different approaches

- **#1** - Involving a client stub created at the time the interface is defined/programmed – using binary protocols
 - MS-RPC, DCE RPC, CORBA, DCOM, Java's RMI, etc.

6 Copyright © Scott Bradner & Ben Gaucherin 2015

A couple of different approaches

- #2 - Internet applications, Web-RPC – distributed interfaces are available and “query-able” - using text or binary Internet application protocols
HTTP, SMTP, HTTP web services (SOAP, XML-RPC, REST, Ajax, etc.), etc.
- Latency gap between these two approaches is shrinking

7

Copyright © Scott Bradner & Ben Gaucherin 2015

Interface Description Language (IDL)

- Language to define the distributed interface
Objects, functions, data types, etc.
- File created by the developers of a distributed interface
- Interface definition file is compiled to generate:
Client stub
Server stub/skeleton

8

Copyright © Scott Bradner & Ben Gaucherin 2015

Interface Description Language (IDL)

- Example for a simple bank account

```
interface BankAccount {  
    attribute float balance;  
    readonly attribute string customer_id;  
  
    void makeDeposit(in float amount, out float newBalance);  
  
    void makeWithdrawal(in float amount, out float newBalance);  
};
```

9

Copyright © Scott Bradner & Ben Gaucherin 2015

Message formats: XML

```
<?xml version="1.0" >
<person id="000" >
  <name>Kru
  <address>
    <street>
    <city>
```

- Originally, no broadly accepted standard for structuring messages
 - Mix of binary and text based approaches
- eXtensible Markup Language (XML)
 - Angled bracket tags to mark beginning/end of structure
 - Structures can be nested

10

Copyright © Scott Bradner & Ben Gaucherin 2015

Message formats: XML

- Example

```
<person first="Ben"
last="Adams" gender="M">
  <interest>Books</interest>
  <interest>Poker</interest>
  <interest>Computers</interest>
</person>
```

11

Copyright © Scott Bradner & Ben Gaucherin 2015

Message formats: XML

- Intended to focus on simplicity, usability
 - Text based/legible – in theory
- Powerful tool to navigate XML tree structures
 - Document Object Model (DOM)

12

Copyright © Scott Bradner & Ben Gaucherin 2015

Message formats: XML



- But...
Everybody wanted to come up with their specialized schema of XML
In many cases tended to be verbose and complex (e.g. SOAP, XSLT, etc.)
Mapping of XML tree structure to programming language type systems

13

Copyright © Scott Bradner & Ben Gaucherin 2015

Message formats: JSON



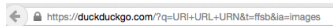
- JavaScript Object Notation
Collection of name/value pairs (objects)
Ordered list of values (array)
e.g.:

```
{  
  "first": "Ben",  
  "last": "Adams",  
  "gender": "M",  
  "interests": [ "Books",  
                "Poker", "Computers" ]  
}
```
- Because browsers support JavaScript, this is a better format to use than XML

14

Copyright © Scott Bradner & Ben Gaucherin 2015

Using named resources



- Use names for resource instead of more explicit references/locators (e.g., IP address)
- Names make location, relocation, fail-over, etc. transparent to user (and parts of the system)

15

Copyright © Scott Bradner & Ben Gaucherin 2015

Using named resources

- **URIs** – Uniform Resource Identifiers
 - URLs – Locator for the resource
 - e.g. `http://www.example.com/index.html`)
 - URNs – Name without specifics of where/how to access
 - e.g. `urn:isbn:978-0471316152`

16

Copyright © Scott Bradner & Ben Gaucherin 2015

Using state

- **State** - Data maintained in the various parts of the system
 - Client side – cookies
 - Server side – (volatile) in memory session data, (persistent) cloud state storage and databases
- When servers maintain state, they need to have a way to identify the client
 - Retrieving cookies, or device fingerprint

17

Copyright © Scott Bradner & Ben Gaucherin 2015

Using state

- Resiliency concerns
 - What happens if you don't hit the same server?
 - Session state stored in a way that all other servers can access
 - What level of degradation is experienced when state is lost?
- Security concerns
 - Who has access to these bits of information?

18

Copyright © Scott Bradner & Ben Gaucherin 2015

Using transactions

- Example
 - Bank transfer:
 - Debit account #1
 - Credit account #2
 - Both need to happen!
 - SABRE (Semi-automated Business Research Environment) flight reservation system
 - With people booking seats around the globe, ensuring we do not end up with more than one person in each seat
 - Resource locking can be an issue!

19

Copyright © Scott Bradner & Ben Gaucherin 2015

Using transactions

- Transaction syntax—begin, commit, rollback
- Two phase commit—distributed transaction with a coordinator node
 - 1) Commit-request
 - 2) Commit
- Transaction log - ability to restore to a known state
- Early transaction managers: ENCINA, Tuxedo

20

Copyright © Scott Bradner & Ben Gaucherin 2015

Transactions are ACID



21

Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
--------	--------

22

Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
4	https://i-technet.sec.s-mft.com/dynimg/IC196578.gif
5	http://www.storynory.com/wp-content/uploads/2013/09/white-rabbit.jpg
10	http://www.inc.com/uploaded_files/image/using-xml-pop_3712.jpg
14	JSON logo
21	http://cdn.softwaretestinghelp.com/wp-content/uploads/2013/08/DB-Testing.jpg

23


Copyright © Scott Bradner & Ben Gaucherin 2015

Distributed software
Evolution of architectures

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

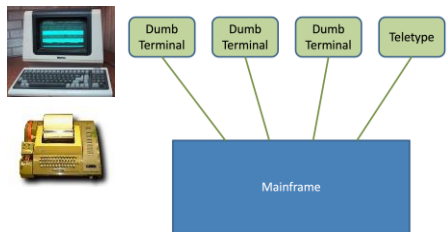
In the beginning



- Few people could afford large scale computing resources
 - Large research institutions
 - Large enterprises
 - Federal government

2 Copyright © Scott Bradner & Ben Gaucherin 2015

Mainframes and dumb terminals



3 Copyright © Scott Bradner & Ben Gaucherin 2015

And then came the network, but...



- No framework for building distributed systems
- Developers would use the network directly
 - Simple client/server
 - Raw socket, no standard for messages, etc.
- UCLA LOCUS operating system – early 80's
 - Mobile VMs – Back to the future...

4

Copyright © Scott Bradner & Ben Gaucherin 2015

Open Software Foundation's DCE

DCE

- Distributed Computing Environment
 - Early 90's
- Comprehensive software framework to develop distributed computing solutions
 - Remote Procedure Call (RPC),
 - Distributed file system, Threads,
 - Distributed Time Service,
 - Directory Name Service, Security Services

5

Copyright © Scott Bradner & Ben Gaucherin 2015

DCE, why did it fail?

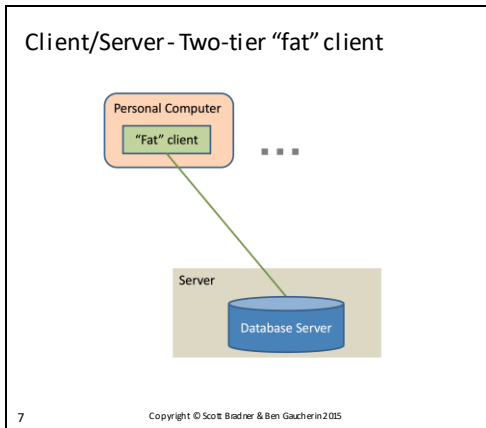
DCE

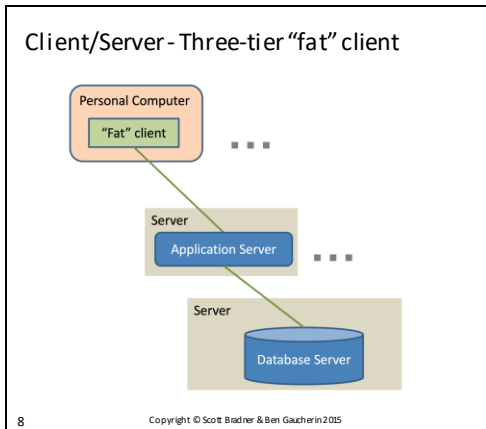
FAIL

- Big and complex
 - For a given system, people only really needed a small subset
- A collection of technologies developed by different vendors
- DCE assumed that building/managing a distributed system was less complicated/costly than building/managing a local system

6

Copyright © Scott Bradner & Ben Gaucherin 2015





A better RPC - distributed objects

- Instantiate objects that appear local, but may be remote
- Common Object Request Broker Architecture – (CORBA)
- Microsoft's DCOM using MSRPC
- Java's Object Serialization/RMI/EJBs

9 Copyright © Scott Bradner & Ben Gaucherin 2015

A better RPC - distributed objects

- Better than remote functions/procedure
Objects contain both functionality and state
- But state-full elements are problematic
Make scalability and resiliency harder
Make security harder
- And object RPC protocols were difficult to use over firewalls

10

Copyright © Scott Bradner & Ben Gaucherin 2015

And then came the Web

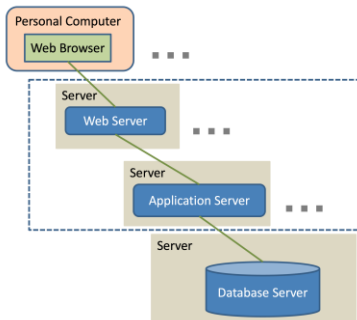


- Mid-90's the World Wide Web starts to be widely deployed
- **Web browsers** became generic clients
Available on all types of operating systems
Ability to create application specific interfaces
No more need to deploy application specific clients
No more fat clients

11

Copyright © Scott Bradner & Ben Gaucherin 2015

Web architectures – web application



12

Copyright © Scott Bradner & Ben Gaucherin 2015

Service Oriented Architecture (SOA)

- Software without a UI
- Not a new concept/construct, but lots of hype
- Use software services
 - Logical groupings of remotely accessible functions
 - e.g. Bank Account service – balance, credit, debit
 - Not just web services

13

Copyright © Scott Bradner & Ben Gaucherin 2015

Service Oriented Architecture (SOA)

- Making possible a new form of systems development: “composable apps”
 - Coarse services by combining granular services
 - e.g. Bank Fund Transfer service – debit from one account, credit to another
 - Business processes using workflow software

14

Copyright © Scott Bradner & Ben Gaucherin 2015

Service Oriented Architecture (SOA)

- Brings out some really complicated business issues
 - Acceptable time span of processes/activities?
 - Where/how are humans involved?
 - How to manage transactions and failures?
 - Active management or compensating transactions

15

Copyright © Scott Bradner & Ben Gaucherin 2015

Web services/Web APIs

- Use HTTP as the base protocol
- Use XML, or JSON for data formats
Early standards: XML-RPC, SOAP, JSON-RPC, RSS, etc.

16

Copyright © Scott Bradner & Ben Gaucherin 2015

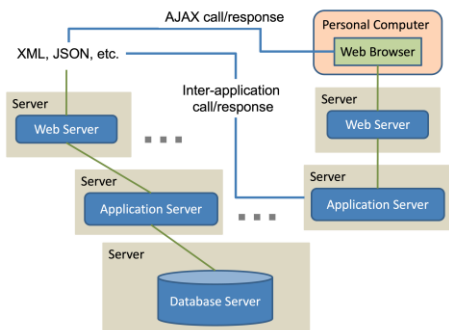
Web services/Web APIs

- Often built as RESTful web APIs
REpresentational State Transfer – Roy Fielding (2000)
URL includes path into server data/function structure
`http://www.mywebservice.org/books/`
`http://www.mywebservice.org/books/9780471316152`

17

Copyright © Scott Bradner & Ben Gaucherin 2015

Web architectures – web services



18

Copyright © Scott Bradner & Ben Gaucherin 2015

Web services/Web APIs



- Used to create mashups
Creating composite views by grabbing information from multiple places
- Broad web APIs to major web platforms
Amazon, Facebook, FedEx, Google, UPS, USPS, Wikipedia, etc.

19

Copyright © Scott Bradner & Ben Gaucherin 2015

Evolution of architectures

- Underlying themes:
 - From proprietary to open standards
 - From expensive to commodity items
 - From few components to many specialized & redundant components

20

Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
2	"IBM Electronic Data Processing Machine - GPN-2000-001881" by NASA - Great Images in NASA Description. Licensed under Public Domain via Commons - https://commons.wikimedia.org/wiki/File:IBM_Electronic_Data_Processing_Machine_-_GPN-2000-001881.jpg#/media/File:IBM_Electronic_Data_Processing_Machine_-_GPN-2000-001881.jpg
3	"Televideo925Terminal". Licensed under Public Domain via Commons - https://commons.wikimedia.org/wiki/File:Televideo925Terminal.jpg#/media/File:Televideo925Terminal.jpg
3	"ASR-33 1" by Dominic Alves from Brighton, England - ASR-33 Teletype. Licensed under CC BY 2.0 via Commons - https://commons.wikimedia.org/wiki/File:ASR-33_1.jpg#/media/File:ASR-33_1.jpg
4	http://i.ytimg.com/vi/T2z4YmFApl/hqdefault.jpg
5,6	DCE logo
11	http://www.w3.org/Press/Stock/BernersLee/small/TimBL-bw.jpg-notle-400.gif_small.jpg

21

Copyright © Scott Bradner & Ben Gaucherin 2015

Distributed software

Sample website

CSCI E 45a: The Cyber World – part A

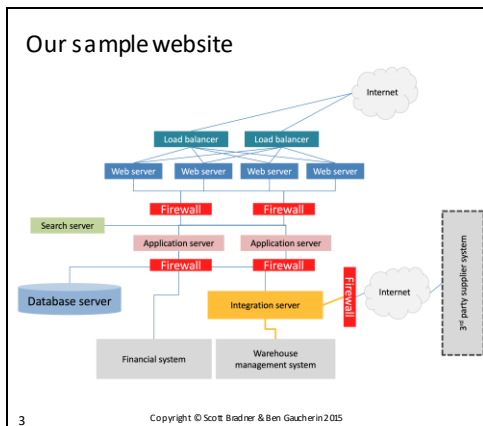
1 Copyright © Scott Bradner & Ben Gaucherin 2015

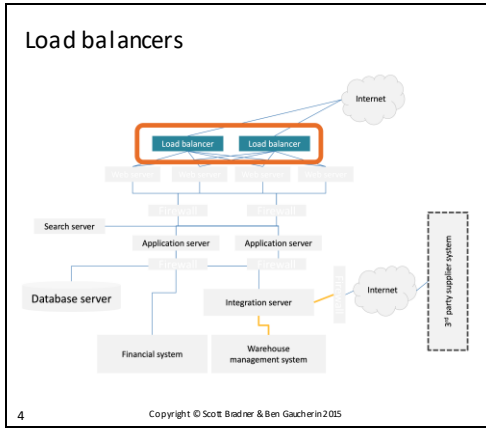
Our sample website



- Let's imagine we have to build a large, and complex website
- Serving large numbers of customers
- In a business requiring complex workflows/processes
- And requiring exchange of information with 3rd party partners

2 Copyright © Scott Bradner & Ben Gaucherin 2015



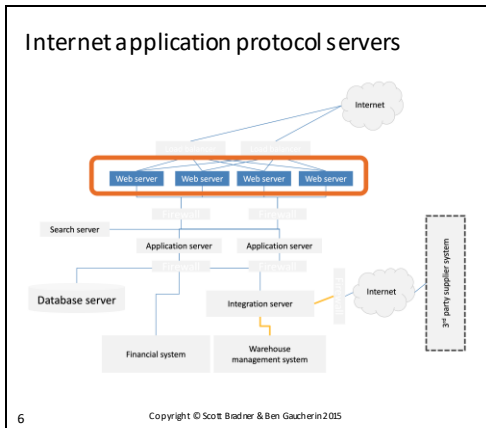


Load balancers

Load balancer

- **Load balancing:** spreading the load between resources
Also helps with resiliency
- Local load balancing
Spreading the load between local servers
- More global load balancing
Domain Name Server (DNS)
Content Delivery Networks (CDNs)

5 Copyright © Scott Badner & Ben Gaucherin 2015



Internet application protocol servers

Web server

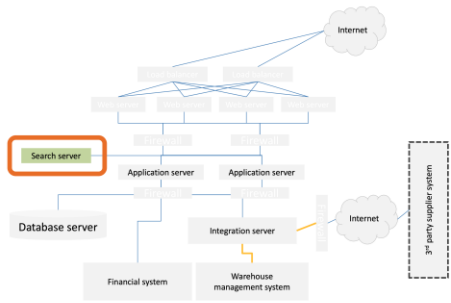


- HTTP or Web server
 - Basic commands: HEAD, GET, POST
 - Common Gateway Interface (CGI)
 - delegate generation of web resource to an executable
- SMTP or Mail server
- Etc.

7

Copyright © Scott Bradner & Ben Gaucherin 2015

Indexing/search servers



8

Copyright © Scott Bradner & Ben Gaucherin 2015

Indexing/search servers

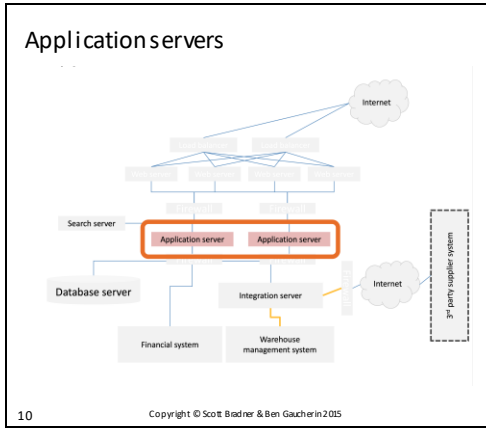
Search server



- Ingest massive quantities of content, to make it easily searchable
- Two basic models:
 - Inverted index
 - faceted index
- Interface to the indexing/search service is often HTTP based

9

Copyright © Scott Bradner & Ben Gaucherin 2015



Application servers

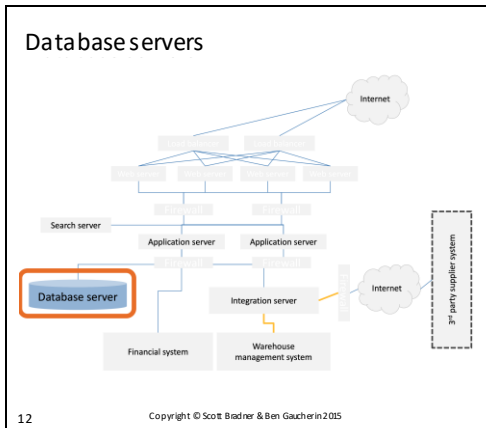
Application server

- Software dedicated to host applications
- Provides a lot of supporting functionality

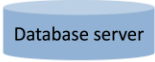
Security, load balancing, state management, transaction management, database connections, etc.

Also provide the “remoting” infrastructure

The logos for JBoss by Red Hat, Microsoft .NET, and Apache Tomcat are displayed on the left side of the slide. The number '11' is in the bottom left corner, and the copyright notice 'Copyright © Scott Badner & Ben Gaucherin 2015' is at the bottom.



Database servers



- Make databases available over the network using standardized query/response format
- Much evolution in database architectures and capabilities
Flat files, ISAM/VSAM, SQL, no-SQL

13

Copyright © Scott Bradner & Ben Gaucherin 2015

Relational/SQL databases



- The most popular architecture today
- Data stored in tables
Columns are fields
Rows are records
- Data manipulated using Structured Query Language (SQL)
CREATE, INSERT, UPDATE, SELECT
Example - `SELECT * FROM customer WHERE customer_age > 18`

14

Copyright © Scott Bradner & Ben Gaucherin 2015

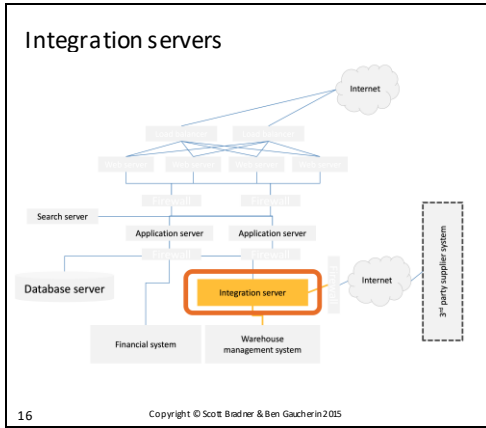
New DB models emerging



- No SQL databases
Built around less rigid structures
Many have a tight affinity to JavaScript
And make their data available as JSON
- Search indexes are also databases of sorts

15

Copyright © Scott Bradner & Ben Gaucherin 2015



Integration servers

- Integration server**
- Allow applications to exchange information
- Exist to...
 - Reduce the number of point to point connections
 - Provide better management of flow of data/integration
- Combine adapters and message based structures
 - Adapters – provide standardized interfaces into diverse systems (file systems, enterprise applications, etc.), like a “message” outlet into a system

17 Copyright © Scott Badner & Ben Gaucherin 2015

Integration servers, contd.

- Integration server**
- Cover a broad spectrum of scenarios
 - infrequent transfer of large amounts of data – Extract Transform & Load (ETL)
 - High volumes of very small messages (queue based, pub/sub)

18 Copyright © Scott Badner & Ben Gaucherin 2015

Large scale supply chains and integration

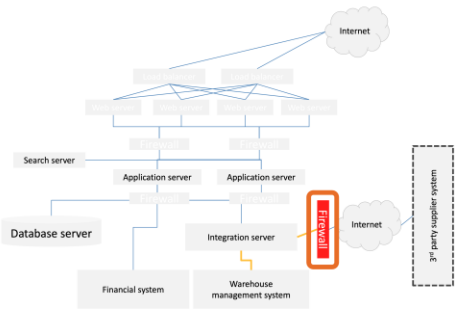
- Enterprise integration as a national imperative
 - Complex global scale supply chain integrations
 - The Enterprise Integration Act of 2002
 - Yet, federal government not showing strong abilities to manage complex integrations



19

Copyright © Scott Badner & Ben Gaucherin 2015

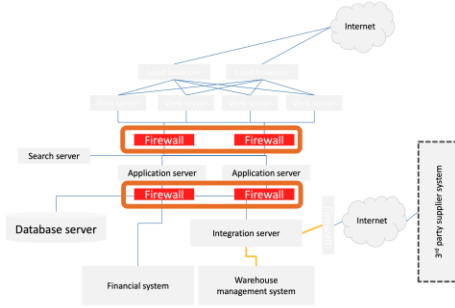
Perimeter firewalls



20

Copyright © Scott Badner & Ben Gaucherin 2015

Internal firewalls




21

Copyright © Scott Badner & Ben Gaucherin 2015

Firewalls

Firewall

- **Perimeter filtering**
Filtering the bits that come in and out of the perimeters of the organization
- **Internal filtering**
Filtering traffic between internal hosts/subnets and ensuring that only those hosts that are expected to talk to each other do



22 Copyright © Scott Bradner & Ben Gaucherin 2015

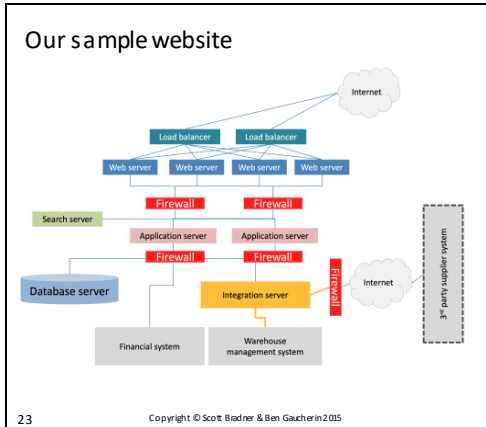


Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
2	Amazon, Netflix, eBay, Expedia, Bank of America
7	Apache webservers, Microsoft ISS 8 logos
9	Apache Solr, Elastic Search logos
11	JBoss, Apache Tomcat, Microsoft .NET logos
14	SQLite, MySQL, ODBC, PostgreSQL, Oracle, Microsoft SQL Server logos
15	MongoDB, CouchDB logos
17	Microsoft BizTalk 2013, Informatica, IBM Integration Bus v.9 logos
19	Healthcare.gov logo
22	Juniper, CISCO logos


24 Copyright © Scott Bradner & Ben Gaucherin 2015

Distributed software
Conclusion


CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

In summary...




- **Distributed systems allow**
The building of systems with requirements that a single (even powerful) machines could not meet
- **Building distributed systems is challenging**
Hard to reconcile the requirements and keep things simple
Components and network are not reliable



2 Copyright © Scott Bradner & Ben Gaucherin 2015

In summary...



- **Building/architecting a distributed system consists of**
Assembling pieces of software that use the network to communicate
Understanding the functional building blocks needed (web servers, database servers, etc.)
Understanding the scaling, performance, security, resiliency requirements

3 Copyright © Scott Bradner & Ben Gaucherin 2015

Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

2 SETI diploma <http://www.triskellon-ltd.com/psa.htm>

2 Bad bridge design <http://wonderfulengineering.com/31-engineering-mistakes-that-make-you-wonder-who-gave-them-engineering-degrees/>

3 Puzzle

https://en.wikipedia.org/wiki/File:A_Puzzle.JPG
