Network and Web Basics.
Architectures of Distributed Applications.
Java Platforms Editions

Vladimir Vlassov and Leif Lindbäck
KTH/ICT/SCS
HT 2015
Outline

• Basic network concepts
  – IP stack, TCP, UDP, IP address, DNS
  – Sockets, ports, socket connection

• Basic WWW technologies
  – URL, HTTP, HTML
  – Client side: Forms, other client-side technologies
  – Server side: Servlets, Beans, Server side scripting

• Architectures of distributed applications
  – Client-server
  – Three-tier
  – P2P

• Networking technologies in JavaSE
• Java Platform editions: JavaSE, JavaEE, JavaME
Network. Host. Internet

- A **network** is a hardware and software data communication system that provides interconnection of computers and other devices.
- A **node (host)** is an addressable device (computer) attached to a computer network.
- An **internet** is a set of networks connected with routers.
- The **Internet** is the largest internet that includes commercial, military, university and other networks with different physical links and various protocols including IP (Internet Protocol).
Multi-Layered Network Architecture

- The seven-layer OSI (Open System Interconnect) model
- The IP networking stack includes 5 layers

Application (FTP, HTTP)
Transport (TCP, UDP)
Network (IP)
Datalink (Ethernet frames)
Physical (e.g. Ethernet, FDDI)
1. Transport Protocols: TCP

- **TCP**, Transmission Control Protocol, is a reliable connection-oriented stream-based transport protocol.
  - Allows sending data in a continuous stream.
  - Guarantees delivering in proper order.

- **Phases of TCP communication:**
  - Establish a connection (open a TCP session)
  - Transfer data over the connection
  - Release the connection

- **Applications using TCP:** file transfer, email, WWW

- **TCP is used on Ethernet and the Internet:** TCP/IP
  - See: the standard STD 7, and the Request For Comments RFC 793
    - Standards and RFCs  [http://www.faqs.org/rfcs/](http://www.faqs.org/rfcs/)
2. Transport Protocols: UDP

- **UDP**, the User Datagram Protocol
  - For pure message passing (datagram send/receive).
  - Neither guarantees delivery nor requires a connection.
  - **Connectionless:**
    - UDP datagrams are sent between two hosts with no previous setup.
    - The datagrams contain the destination address, may take different routes.
    - Lightweight and efficient. Low overhead compare to TCP

- Phases of UDP communication:
  - Sending: create a UDP socket; create a datagram with the message and specified destination (IP address & port); send the datagram over the UDP socket.
  - Receiving: receive a datagram from the UDP socket; get data and source from the datagram

- Applications using UDP: DNS, streaming media (IPTV, VoIP, video-conferencing), online games

- UDP is layered on top of IP: UDP/IP; See STD 6, RFC 768
Network Protocol: IP

- **IP**, Internet Protocol, is a network layer protocol used for routing.
  - IP is connectionless
    - IP datagrams fragmented into IP packets
    - IP header includes destination, source, and time-to-live (TTL)
  - See STD 5, RFC 791
Protocol Encapsulation

- To get the size of a Maximum Transmission Unit (MTU) on a Linux machine: `ifconfig -a`
Addressing a Node on the Internet.

**IP Address**

- An Internet address (**IP address**) of a node on the Internet is a four-byte (32-bit) unsigned integer number (IPv4).
  - **Dot decimal notation**: four unsigned integers, each ranging from 0 to 255, separated by periods.
  - Example: 130.237.214.84
  - 127.0.0.1 - the local loopback interface, localhost
  - Addresses beginning with 0.0 refer to hosts on the same LAN.
    - 0.0.0.0 is used as a source address of the originating host.
  - Addresses beginning with 10. and 192.168. are non-routable and can be used on internal (private) networks.
  - Addresses beginning with 224. are multicast addresses.
    - 224.0.0.1 - multicast address on the LAN
IP Address Classes

- Internet addresses are assigned by Internet Corporation for Assigned Names and Numbers (ICANN) through Internet Service Providers (ISPs).
- Internet address classes
  - A (1-126.x.x.x) – 126 address blocks, each of 16,000,000 addresses.
  - B (128-191.x.x.x) – one address block contains ~65,000 addresses.
  - C (192-223.x.x.x) – one address block contains 254 addresses.
  - D (224-239.x.x.x) – multicast addresses.
  - E (240-255.x.x.x) – reserved.

Classes

<table>
<thead>
<tr>
<th></th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>Network</td>
<td>Host</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1 0</td>
<td>Network</td>
<td>Host</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1 1 0</td>
<td>Network</td>
<td>Host</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1 1 1 0</td>
<td></td>
<td>Multicast Group</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1 1 1 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Addressing a Node on Ethernet.

**MAC Address**

- **MAC** (Media Access Control) address
  - The HW address of a device connected to a shared network medium, e.g. Ethernet.
  - MAC address is used by the link layer.

- **ARP** (Address Resolution Protocol) is used for conversion of an IP address into the corresponding MAC address.
  - The sender broadcasts an ARP packet with the Internet destination address and waits for the destination host to send back its Ethernet address.
  - If no reply, the “unreachable host” ICMP message is generated
  - Each host maintains a cache of address translations.

- `arp -a`
  - Display the Internet-to-Ethernet address translation tables.
Host Names

- A *hostname* is a unique name of a computer on the Internet. It consists of a local name and a domain name.
  - For example: oyster.it.kth.se
- A machine may have multiple names, for example:
  - mail.it.kth.se - an e-mail server
  - ftp.it.kth.se - an FTP server
  - piraya.it.kth.se - a host on the Internet.
- One name can be mapped to multiple IP addresses
  - Web site with multiple hosts
- See [http://www.iana.org/domain-names.htm](http://www.iana.org/domain-names.htm)
DNS: Domain Name System

• **DNS** is a distributed service on the Internet that translates host names into IP addresses.

• Search for a host information:
  – Lookup in the local cache: the `/etc/hosts` file
  – *optional*: NIS (Network Information Service)
  – Lookup in DNS

• **nslookup** - lookup IP-address by name (or visa versa)

  ```
  C:\>nslookup www.oracle.com
  Server:  res2.ns.kth.se
  Address:  130.237.72.200

  Non-authoritative answer:
  Name:  e7075.x.akamaiedge.net
  Address:  23.61.230.140
  Aliases:  www.oracle.com
           www.oracle.com.edgekey.net
  ```
Sockets

- **Socket** is an end-point of a virtual network connection between processes – much like a full-duplex channel
  - A socket address: IP address and a port number
  - A socket type: distinguished by the transport protocol used for communication over the socket
    - TCP socket - stream-based, connection-oriented
    - UDP socket - datagram-based, connectionless
- The socket API in C, a.k.a. Berkeley sockets, was introduced in 1981 as the Unix BSD 4.2 generic API for inter-process communication
  - Initially was a part of the kernel (BSD Unix)
  - Today is a library (Solaris, MS-DOS, Windows, OS/2, MacOS)
Ports

- **Port** is an entry point to a process that resides on a host.
- 65,535 logical ports with integer numbers 1 - 65,535
- A port can be allocated to a particular service:
  - A server listens the port for incoming requests
  - A client connects to the port and requests the service
  - The server replies via the port.
- Ports with numbers 1-1023 are reserved for well-known services.
  - A list of services and allocated ports is stored in
    - /etc/services (Linux)
    - C:\WINDOWS\system32\drivers\etc\services (Windows)
### Some Assigned Ports (RFC 1060)

<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>echo</td>
<td>7</td>
<td>tcp/udp</td>
<td>Echo back the input</td>
</tr>
<tr>
<td>discard</td>
<td>9</td>
<td>tcp/udp</td>
<td>Discard the input</td>
</tr>
<tr>
<td>daytime</td>
<td>13</td>
<td>tcp/udp</td>
<td>Output an ASCII string with the current time</td>
</tr>
<tr>
<td>ftp-data</td>
<td>20</td>
<td>tcp</td>
<td>Data port of ftp: transfer file</td>
</tr>
<tr>
<td>ftp</td>
<td>21</td>
<td>tcp</td>
<td>Command port of ftp: send ftp command</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
<td>tcp</td>
<td>Interactive remote command-line sessions</td>
</tr>
<tr>
<td>smtp</td>
<td>25</td>
<td>tcp</td>
<td>“Simple Mail Transfer Protocol”: send email</td>
</tr>
<tr>
<td>time</td>
<td>37</td>
<td>tcp/udp</td>
<td>The number of seconds since Jan. 1, 1990</td>
</tr>
<tr>
<td>whois</td>
<td>43</td>
<td>tcp</td>
<td>Directory service for Internet administrators</td>
</tr>
<tr>
<td>finger</td>
<td>79</td>
<td>tcp</td>
<td>Information about a user or users</td>
</tr>
<tr>
<td>http</td>
<td>80</td>
<td>tcp</td>
<td>HyperText Transfer Protocol of WWW</td>
</tr>
<tr>
<td>pop3</td>
<td>110</td>
<td>tcp</td>
<td>Post Office Protocol for server-to-client mail</td>
</tr>
<tr>
<td>nntp</td>
<td>119</td>
<td>tcp</td>
<td>Network News Transfer Protocol</td>
</tr>
<tr>
<td>RMI</td>
<td>1019</td>
<td>tcp</td>
<td>Java RMI registry service</td>
</tr>
</tbody>
</table>
The Berkeley Socket API for the Client-Server Architecture

Client

- `socket()`
- `connect()`
- `write()`
- `read()`
- `close()`

Connection is established

Server

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- Start a thread

Start a thread

- `accept()`

HTTP/1.0 200 OK … </html>

0.1/PTTH lmth.xedni/ TEG

Lecture 1: Network and Web Basics. Architectures of Distributed Applications
Some Basic Web Technologies

– URL, HTTP, HTML, XML, SOAP
– Forms, Servlets, Beans, JSF, Sever-side processing
World-Wide Web. URLs

- **World-Wide Web (WWW, the Web)** is distributed client-server information system on the Internet
  - allows to locate and to access resources (files, services) on the Internet pointed on by URLs via servers by using Web protocols such as HTTP.
- **Uniform Resource Locator (URL)** is the address of a resource on the Internet. See RFC 1738.
  - Common URL syntax:
    
    `<scheme>://<user>:<password>@<host>:<port>/<url-path>`
  - For example:
    
    ftp://anonymous@ftp.sunet.se/
    mailto:jnp-adm@it.kth.se
    http://www.it.kth.se/index.html
    telnet://vlad@octopus/
    http://student:nescafe@vvv.it.kth.se/edu/gru/Java/assignments/`
Some Web Protocols

- **HTTP**, Hyper Text Transfer Protocol
  - A client-server TCP/IP protocol. Stateless. RFC 2086 (1.1)
  - The most implemented requests are GET, HEAD and POST
  - URL format: http://user:password@<host>:<port>/<URL-path>
  - Server process: httpd; Default port: 80
- **FTP**, File Transfer Protocol
  - A session-oriented TCP/IP protocol. See STD 9, RFC 959
  - URL format: ftp://<user>:<password>@<host>:<port>/<URL-path>
  - Ports: 20 (data), 21 (commands)
- **SMTP**, Simple Mail Transfer Protocol
  - A server-to-server protocol for e-mail transfer. See STD 10, RFC 821
  - SMTP port: 25
- **SOAP**, Simple Object Access Protocol,
  - A protocol for exchanging structured information in the implementation of Web Services.
  - Relies on XML for message formats, and HTTP (SMTP) as a transport protocol for message transmission.
Telnet

- **Telnet protocol** is the Internet standard protocol for remote login that runs on top of TCP/IP (see: STD 8, RFC 854)
- **telnet** is a program that uses the Telnet protocol and acts as a terminal emulator for the remote login session

  `avril:~>telnet www.ict.kth.se 80`
  Trying 130.237.216.36...
  Connected to web.ict.kth.se.
  Escape character is `^[`.
  GET /index.html HTTP/1.0

  HTTP/1.1 302 Found
  Date: Mon, 25 Oct 2013 09:34:23 GMT
  Server: Apache/2.2.6 (Unix) mod_ssl/2.2.6 ...
  ...
  Connection closed by foreign host.
Markup Languages

- **HTML**: HyperText Markup Language
  - A Hypertext document format used on WWW.
  - "Tags" are used to mark text elements:
    < directive (case insensitive), zero or more parameters > text element </ directive>
  - Links to other documents:
    <A HREF="http://machine.edu/subdir/file.html">foo</A>

- **XML**: Extensible Markup Language
  - A language for exchange of a wide variety of data on the Web and elsewhere.
The APPLET Tag

**Java Applet** - a downloadable Java component executed on the browser’s JVM

```xml
<APPLET
    CODEBASE = codebaseURL
    ARCHIVE = archiveList
    CODE = appletFile ...or...
          OBJECT = serializedApplet
    ALT = alternateText
    NAME = appletInstanceName
    WIDTH = pixels
    HEIGHT = pixels
    ALIGN = alignment, e.g."baseline"
    VSPACE = pixels
    HSPACE = pixels
>
    <PARAM NAME = appletAttribute1 VALUE = value>
    <PARAM NAME = appletAttribute2 VALUE = value>
    ...
</APPLET>
```
Java Applets are not yet Obsolete

• Some real world applications of Java applets
  – *ThinkFree Online* – an office suit using Java Applets and Ajax.
  – *JPC Emulator* – an x86 emulator.
  – *Yahoo Games*
  – Android apps
The EMBED Tag

<EMBED
  TYPE = “application/x-java-applet;version=1.1.2”
  CODEBASE = codebaseURL
  ARCHIVE = archiveList
  CODE = appletFile ...or... OBJECT = serializedApplet
  ALT = alternateText
  NAME = appletInstanceName
  WIDTH = pixels  HEIGHT = pixels
  ALIGN = alignment
  VSPACE = pixels  HSPACE = pixels
  PLUGINSPAGE="http://java.sun.com/products/plugin/1.2/
  plugin-install.html"
  appletAttribute1 = value
  appletAttribute2 = value
  ...
>
<NOEMBED> No JDK 1.2 support for APPLET!! </NOEMBED>
</EMBED>
HTML Forms

• Allow constructing a simple GUI embedded in an HTML document for a Web client.
  – To input a user request and submit it to a CGI program or a servlet (JSP/JSF).

• Example:

```html
<FORM method="POST"
  ACTION="/bin/javacourse/ReportCheck.exe">
  <P>Enter your personal number (YYMMDD-xxxx):
  <P><INPUT Type="text" Name="personalNumber"
    Value="" Size="11" >
  <P><INPUT Type="submit" value="Submit">
  <INPUT Type="reset" value="Reset"></P>
</FORM>
```
Dynamic Web Content: Server Side Processing

- Provides dynamically generated contents: dynamic web sites, web applications, web services
  - The content is generated when requested (on a HTTP request).
- A CGI program
  - Executed in a separate process
  - An old obsolete technology;
- Java Servlets
  - Live in server-side JVM, process HTTP requests and generate content
  - Methods doGet, doPost, doPut, doDelete, init, destroy
- Enterprise JavaBeans
- Server-Side Scripting
  - Embedding program code in HTML documents, parsing and executing the code by the Web server; the result is included in the place of the code.
  - Examples:
    - Active Server Pages (ASP.NET) from Microsoft
    - Java Server Faces (JSF) from Oracle
    - Hypertext Preprocessor (PHP)
GET and POST Requests

Two ways an HTTP request is presented to the server and passed to the target Java servlet, JSP or JSF:

- **GET method**
  
  GET /Adder?username=Vladimir+Vlassov&email=vlad%40it%2ekth%2ese HTTP/1.0
  
  – The parameters values are sent as a query string along with the URI.

- **POST method**
  
  POST /Adder HTTP/1.0
  
  Content-type: application/x-www-form-urlencoded
  
  Content-length: 65
  
  username=Vladimir+Vlassov&email=vlad%40it%2ekth%2ese
  
  – The parameters values are sent in the request body, in the format that the content type specifies.
  
  – Can be used for update/upload any content of a specified MIME type.
Java Servlets

- **Java Servlet** is a Java object in the server’s JVM
  - Provides extra functionality on the server side (extends web-server).
  - Given a name on deployment; addressed by the corresponding URL
  - Accepts and processes user’s requests from HTML forms and applets
  - For example, provides access to corporate databases and information services in the third tier of a 3-tier application

- A servlet lives in the server’s JVM (application server) much like an applet lives in the client’s JVM (in Web browser).
  - Once created, a servlet is alive as long as the server (JVM) is alive
  - The servlet can keep state between requests
    - New state and response is a function of old state and request
  - A servlet can be multithreaded – Scalability
  - A servlet may use EJBs – Extendibility
Servlets vs CGI

CGI

Server
/cgi-bin/...
/cgi-bin/...
state

Native server

Server
JVM
servlet
servlet
state

Java server

JVM
Server
servlet
servlet
state
Some Solutions for Server-Side Scripting

• **ASP.NET from Microsoft**
  • Languages: C#, VBScript based on Visual Basic
  • Tags `<% dynamic code %>`

• **JavaServer Faces (JSF) from Oracle**
  • Language: JSF markup and Java
  • To be studied later in the course

• **Hypertext Preprocessor (PHP) – open source software**
  • A server-side, cross platform HTML-embedded scripting language
Servlet API and Servlet Enabled Servers

• The Java Servlet API is available as a part of Java Platform, Enterprise Edition (javax.servlet)

• The Java Servlet API and JSFs are supported on many web servers (application servers), see
  • http://www.oracle.com/technetwork/java/javaee/compatibility-1-138385.html
  • http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html
Architectures of Distributed Applications

- Distributed applications
- Architectures of distributed applications
- Java networking technologies
Distributed Applications

• Motivation:
  – Data, computers and resources, users (clients) are geographically distributed;
  – Improve performance or/and scalability or/and robustness of applications by means of distributed execution.

• Distributed applications on a network of computers (LAN, WAN, the Internet):
  – Print servers, distributed file systems (DFS), DNS, rlogin;
  – WWW: web servers and browsers, ftp and mail servers, ftp and mail clients, instance messaging, on-line games, content delivery networks, streaming media applications, web-services, etc.;
  – Financial and commercial applications: E-commerce, banking (OLTP);
  – Remote control and monitoring of networked devices;
  – Scientific and engineering computing;
  – Cloud computing environments;
  – Content delivery (or distribution) networks (CDN)
Basic Architectures of Distributed Applications

• **Two-tier architecture** (a.k.a. *client-server* architecture):
  – Clients (with UI, GUI)
  – Servers

• **Three-tier architecture**
  – Clients (with UI, GUI) – in the 1\textsuperscript{st} tier
  – Business logic – in 2\textsuperscript{nd} tier
  – System services (databases) – in the 3\textsuperscript{rd} tier

• **Peer-to-peer (P2P) architecture**
  – Formed of peers– processes running on networked nodes
  – On structured or unstructured overlay networks
  – All peers are equal, being both clients and servers

• **Service-Oriented Architecture (SOA)**
  – Builds on web-services with well defined interfaces, which can be described, deployed, discovered, bound, composed, invoked.
  – Based on WS technologies and standards
  – Studied in ID2208 Programming Web-Services, period 3
2-Tier Client-Server Architecture

- The most commonly used model for distributed applications
  - Can be applied for a particular request-response interaction
- The **client** is the entity (process) accessing the remote resource and the **server** provides access to the resource.
- Request / response protocols

![Diagram of 2-Tier Client-Server Architecture](attachment:image.png)
Problems of 2-Tier Client-Server on the Internet

• Portability
  – No control over the client operating system and hardware.
  – Challenging to upload anything to the client if it does not accept.

• Efficiency
  – A “fat” client may require too much resources on a client machine
    • Also slow to download (applets)
  – Direct SQL access can generate lots of network requests

• Security – the most important
  – DBAs do not accept the risks of putting the database on the Internet
  – Internet security should be at the service level, not at the data level
3-Tiered Architecture

- **User-Interface Tier**
  - The layer of user interaction.
  - A “thin” client of the business logic servers

- **Business Logic Middle-Tier**
  - The business logic layer. It is made up of business objects: inventory control, budget, transaction monitors, ORBs, authentication, etc.

- **System Service Tier (e.g. persistent storage)**
  - Objects that encapsulate database routines and interact with DBMS.
3-Tier Internet Architecture Benefits

• Improved performance
  – Use faster protocols than http or ODBC
  – Download the GUI (thin client), but leave the rest of the logic on
    the server or in the middle-tier

• Manage security
  – The middle-tier are not restricted by applet security rules
  – The middle-tier can control user authentication, access to resources
    in the third tier

• Manage user application context
  – The server can remember user data
  – The user can access his context from any Web client
3-Tier and Skills Partitioning

- Application Developers concentrate on the user’s needs: GUI, how to present business information, convenient front-ends
- Business Object Modelers work with the domain experts
- Architects manage technology integration
- DBAs focus on data storage, administration and optimization

Clients (applets, applications)

Business Object Framework

Database

Technical Framework
Peer-to-Peer (P2P) Architecture

• A P2P application runs on an overlay network
  – All peers are equal in terms of responsibility, capabilities and functionality: typically execute the same set of algorithms, participate in distributed algorithms

• An *overlay network* is a “virtual” network of nodes created on top of an existing network, e.g. the Internet.
  – Each node has an ID, knows neighbors, does not know the global topology, communicates as a source and a destination, and also *serves as a router* in sending data.
  – Can provides a **Distributed Hash-Table (DHT)** functionality

• Structured overlay (P2P) networks
  – E.g. Chord, Pastry, Tapestry, DKS

• Unstructured overlay networks
  – E.g. Gnutella
General Design Issues of Distributed Applications

Quality:

• Functional requirements
  – What functions the application must provide
  – Usage scenarios, use cases – to guide development and to test against
  – API (Application Programming Interfaces) specifications
  – Should be discussed with domain experts and end-users

• Non-functional requirements
  – Given the application fulfills functional requirements, how good is it?
    • “Goodness” has to be defined as measurable metrics;
    – Performance: short response time, low latency, high throughput;
    – Complexity: Message complexity; time complexity;
    – Scalability: ability to handle a growing workload in a capable manner, or ability to be enlarged to accommodate that growth;
    – High availability and dependability (trustworthiness)
    – Elasticity: ability to grow (scale out) or shrink depending on workload
    – Other requirements
General Design Issues (cont’d)

• First major problem: Communication latency
  – Affects response time, user experience with the applications
  – Issues at client side:
    • Responsive and informative UI (GUI)
    • Tolerate long communication latency by data caching and prefetching
    • Hide long communication latency by multithreading
  – Issues at server side:
    • Concurrency by multithreading: handle client requests in multiple threads

• Second major problem: Failures
  – Need to build reliable distributed applications and systems
  – Issues at server side:
    • (Transparent) Replication for robustness and/or performance

• Third major problem: Dynamicity
  – Nodes (resources) can un-predicatively join/leave/fail
  – The application/system can be evolving over time
General Design Issues (cont’d)

• **How to achieve good quality?**
  – Balanced distribution of functionality among distributed components – which component does what; loosely coupled
  – Efficient communication protocols – use as less as possible messages
  – Proper levels of location transparency and location awareness
  – Data replication and caching
    • Consistency and coherence issues
  – Data migration and prefetching
  – Multithreading, caching and prefetching allow to hide and / or to avoid long communication latencies
  – Scalability by concurrent execution – multithreading
    • Servicing of requests in parallel threads
    • Exploit multicore facilities
  – Fault tolerance, failure management
Basic Communication Mechanisms

- **Message passing** over sockets (TCP or UDP)
  - Application specific request/response protocols
- **Remote Procedure Calls (RPC) and rendezvous**
  - RPC spawns a new process (thread) to handle a request
  - Rendezvous – request is accepted (selected) and processed by an existing running server process
- **Remote Method Invocation (RMI)**
  - The object-oriented analog of RPC in a distributed object-oriented environment
  - Distributed object architecture
A Distributed Component Architecture (Platform)

- *A middleware* that provides ability to build an application of distributed components (objects, web-services), i.e.
  - To declare, create, name, locate and bind distributed components
  - To (transparently) invoke methods on the components
  - To migrate, replicate and keep consistent distributed copies of a component
  - To manage distributed memory: distributed garbage collection
  - To automate most of systems functions (deployment, runtime reconfiguration and upgrade, failure management, etc.)

- Defines, specifies and provides services (and corresponding APIs) common for most applications, such as naming, deployment, lifetime management, transactions, etc.

- Typically includes:
  - A programming model,
  - A programming environment with APIs,
  - A runtime system (containers, services)
SOA: Service-Oriented Architecture

• Applications are built of services
  – Services are built of components
  – Components are bound to each other via client/server interfaces
    • A client interface of a (client) components is bound to a server interface of another (server) component

• Services are loosely-coupled
  – Expose interfaces (port types);
  – Can be described, discovered, bound, and invoked;
  – Service invocation: request-response interaction
Some Existing Approaches

- **CORBA**
  - Common Object Request Broker Architecture from OMG
  - Heterogeneous
  - Many implementations exist

- **.NET (DCOM)**
  - Distributed Component Object Model from Microsoft
  - Homogeneous (“MS-only”)

- **Java RMI**
  - Homogeneous
  - Enterprise JavaBeans – A component architecture for building integrated enterprise services based on RMI/IIOP

- **Web services**
  - SOAP (Simple Object Access Protocol) – a minimal set of conventions and standards for invoking code using XML over HTTP
Essential Networking Technologies in Java
(web-services technologies are not shown)

• Sockets (TCP, UDP, multicast) -- the substrate technology
  • URL connections
  • HTTP URL connections

java.net

• Applets - Downloadable clients
  java.applet

• Servlets - Web server extension
  javax.servlet

java.io

• Streams, Serialization

• JavaIDL (CORBA)
  org.omg.CORBA

java.rmi

• Remote Method Invocation
  – A native Java ORB
Three Java Editions (Platforms)

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
The Java Platform, Standard Edition (Java SE)


Lecture 1: Network and Web Basics. Architectures of Distributed Applications
Java Platform, Enterprise Edition (Java EE)

- **Enterprise Application Technologies**
  - Enterprise JavaBeans (EJB)
  - J2EE Connector Architecture
  - Java Message Service (JMS)
  - Java Persistence API (JPA)
    * Provides a persistence model for object-relational mapping. Developed and use for EJB, but can be used directly
  - Java Transaction API (JTA)
  - JavaMail

- **Web Application Technologies**
  - Java API for WebSocket
  - Java Servlet
  - JavaServer Pages (JSP)
  - JavaServer Faces (JSF)

- **Management and Security Technologies**
  - J2EE Application Deployment
  - J2EE Management
  - Java Authorization Contract for Containers

See the complete list of Java EE technologies at [http://www.oracle.com/technetwork/java/javaee/tech/index.html](http://www.oracle.com/technetwork/java/javaee/tech/index.html)
Java EE (cont’d)

- **Java EE Web Services Technologies**
  - Java API for RESTful Web Services (JAX-RS)
  - Java API for XML-Based Web Services (JAX-WS)
    - Replaces JAX-RPC
  - Java API for XML-Based RPC (JAX-RPC)
  - Java Architecture for XML Binding (JAXB)
    - Provides a convenient way to bind an XML schema to a representation in Java code.
  - SOAP with Attachments API for Java (SAAJ)
    - Provides a standard way to send XML documents over the Internet from the Java platform.
  - Streaming API for XML
    - Streaming Java-based, event-driven, pull-parsing API for reading and writing XML documents.
  - Web Service Metadata for the Java Platform

See the complete list of Java EE technologies at http://www.oracle.com/technetwork/java/javaee/tech/index.html
Java Platform Micro Edition (Java ME)

Components of Java ME technologies.
Java ME (cont’d)

- **Configurations**
  - functionalities (runtime, APIs) for a particular range of devices with similar characteristics
  - **CLDC**: The Connected Limited Device Configuration;
  - **CDC**: The Connected Device Configuration.
- **KVM**: Kilobyte Virtual Machine

- **Profiles**
  - complete runtime environments and APIs for a specific device category
  - **MIDP**: Mobile Information Device Profile;
  - **FP**: Foundation Profile;
  - **PDAP**: Personal Digital Assistant Profile