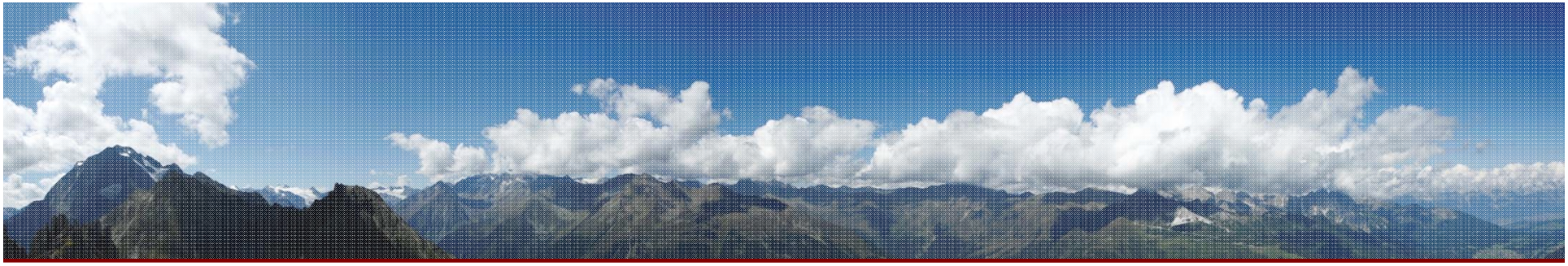
Web Services
WS 2017/18

Distributed Information Systems

Anna Fensel



What is the course about?

STI · INNSBRUCK

- Understanding of perspective in which Web services emerged
  - Distributed information systems and middleware solutions
- A stack of service-based technologies
  - Web technologies
  - Web service technologies
  - Web2.0/RESTful service technologies
- Security in Web services

STI · INNSBRUCK

- The lecturer is:

  Ass.-Prof. Dr. Anna Fensel (anna.fensel@sti2.at)

- Tutor (practicum):

  Mag. Zaenal Akbar (zaenal.akbar@sti2.at)

The lectures are bi-weekly, and practicum is weekly.

STI · INNSBRUCK

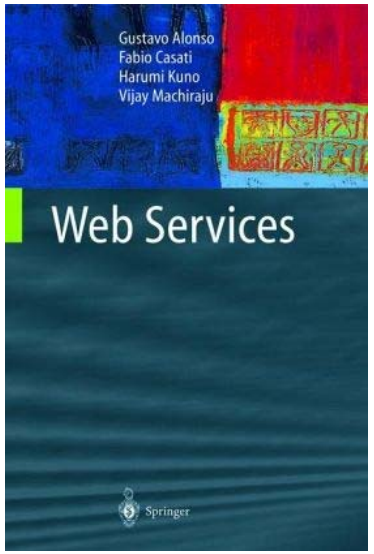
- Web site:

  https://www.sti-innsbruck.at/teaching/course-schedule/ws-201718/web-services-ws-20172018

  – Slides available online after each lecture

- Book

  – Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. Web Services – Concepts, Architectures and Applications. Springer-Verlag, 2004.



Where are we?

STI · INNSBRUCK

#	Title
1	**Distributed Information Systems**
2	Middleware
3	Web Technologies
4	Web Services
5	Basic Web Service Technologies
6	Web 2.0 Services
7	Web Service Security

Outline

STI · INNSBRUCK

- Motivation
- Technical solution
  - Distributed Information Systems
    - Design of an Information System
    - Architecture of an Information System
    - Communication in an Information System
- Illustration by a larger example
- Extensions
- Summary
- Resources

Motivation

Motivation

STI · INNSBRUCK

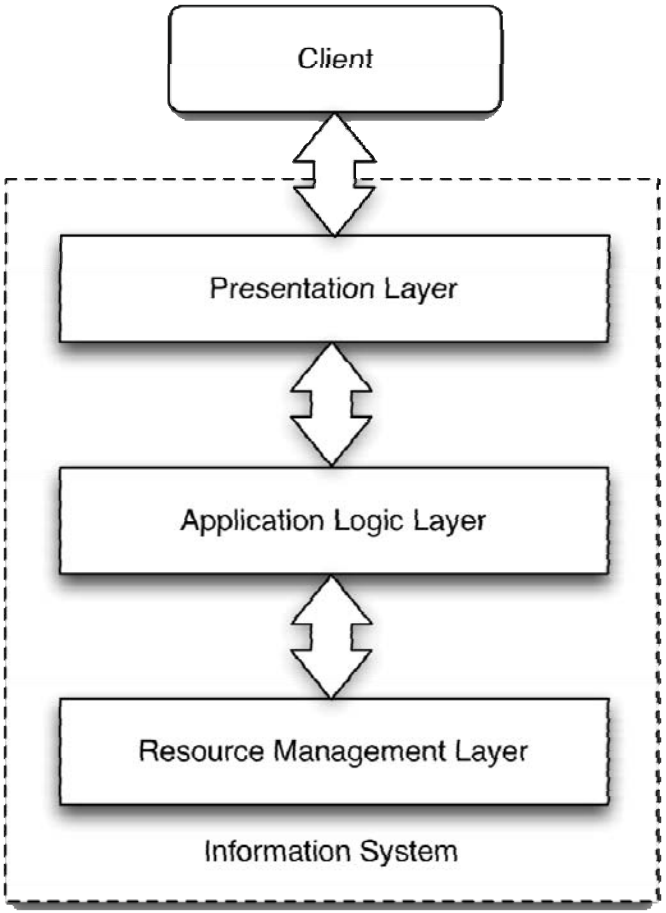
- Web services are a form of distributed information systems.
- Web services are the result of the natural evolution of middleware and enterprise application integration platforms as they try to rely on Internet technologies.
- In order to understand Web services, we need to understand how we got there and what is the relation of Web services with existing technologies to build distributed information systems.
- While the technology has changed, the problems that need to be solved are to a large extent the same.
- The aim of this lecture is to establish the correct perspective of Web services in the context of distributed information systems.

Technical Solution

Design of an Information System

Design of an Information System
Three Layers

STI · INNSBRUCK



- Responsibilities of the presentation layer
  - Communication with the external entities (both humans and systems).
  - Information presentation.
  - Interaction with other systems by accepting operations and getting responses.
- Presentation layer can take various forms
  - Graphical User Interface (GUI).
  - Module which formats a data set into a given syntactical representation.
- Should be distinguished from the notion of client
  - Clients are consuming services provided by an information system (e.g., Web browser).
  - Client and presentation layer may be merged into one (typical in Client/Server systems).

- Responsibilities of the application logic layer
  - Data processing behind the delivered results.
  - Implements actual operations submitted by clients through the presentation layer.
  - The operations may be referred to as the *services*.
- Example of money transfer operation in a banking operation.
- Depending on the complexity of the logic involved and of the selected implementation technique we can refer to this layer as
  - Business processes,
  - Business rules, and
  - Sometimes simply server.

- Responsibilities of the resource management layer
  - Data management (databases, file systems, other repositories)
  - Implements different data sources of an information system independently of the nature of these data sources.
  - Also known as data layer
- RML may include as a part any external system that provides information
  - Other information systems with all three layers
  - This allows for a recursive building of an information system

STI · INNSBRUCK

- Typical workflow of top-down information system design:
  1. Starts from the functionality driven by clients and their interactions with the system.
  2. Application logic is developed to fulfill the required functionality,
  3. Supporting resources are defined to support application logic operations.

- Focuses first on the high-level goals of the problem and does everything to achieve the goals
  - Specifies also how the system will be distributed across different computing nodes.
  - Usually created to run on homogeneous nodes, thus tightly coupled (components cannot be used independently).

- Advantages
  - Emphasizes the final system goals.
  - Tailored to address functional (system operations) and non-functional (e.g., performance and availability) system properties.
- Drawbacks
  - Can be applied only to the systems developed from scratch (rarely a case today).

Design of an Information System
Bottom-Up Design

STI · INNSBRUCK

- Typical workflow of bottom-up information system design
  1. Starting with the high-level goals as in the previous case.
  2. Evaluating RML to see whether something can be fulfilled
     - What are the costs, feasibility to obtain the information
  3. RML components are wrapped to enable proper interfaces for app. logic layer
  4. Designing application logic.
- Occurs from necessity rather than choice
  - Integration of existing systems (e.g., legacy apps).
  - Everything is predefined and cannot be easily modified.
  - How to integrate such a system in a coherent whole?
- Yields loosely-coupled systems by design
  - Legacy systems are used as components and maintained as stand-alone systems
- No sense to talk about advantages/disadvantages
  - Usually there is no other choice
  - Web services are making bottom-up design more efficient, cost-effective, and simple.

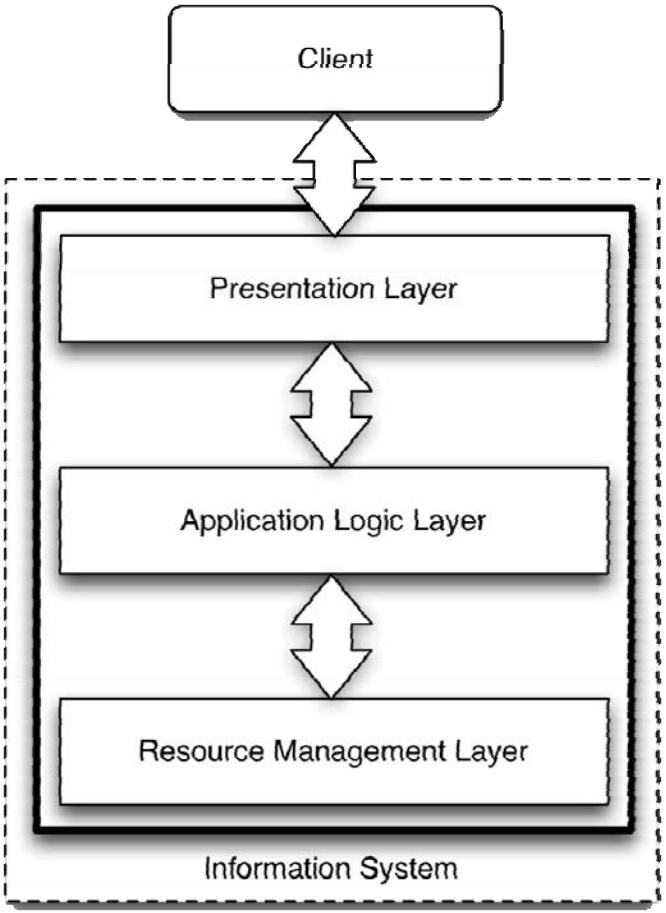
Technical Solution

Architecture of an Information System

STI · INNSBRUCK

- Three layers are logically separating functionality of an information system
- The layers may be combined and distributed in different ways
  - Not anymore layers but TIERS.
- There exist 4 basic types of information systems depending on how tiers are organized:
  - 1-tier,
  - 2-tier,
  - 3-tier, and
  - N-tier.

Architecture of an Information System
One-tier Architectures

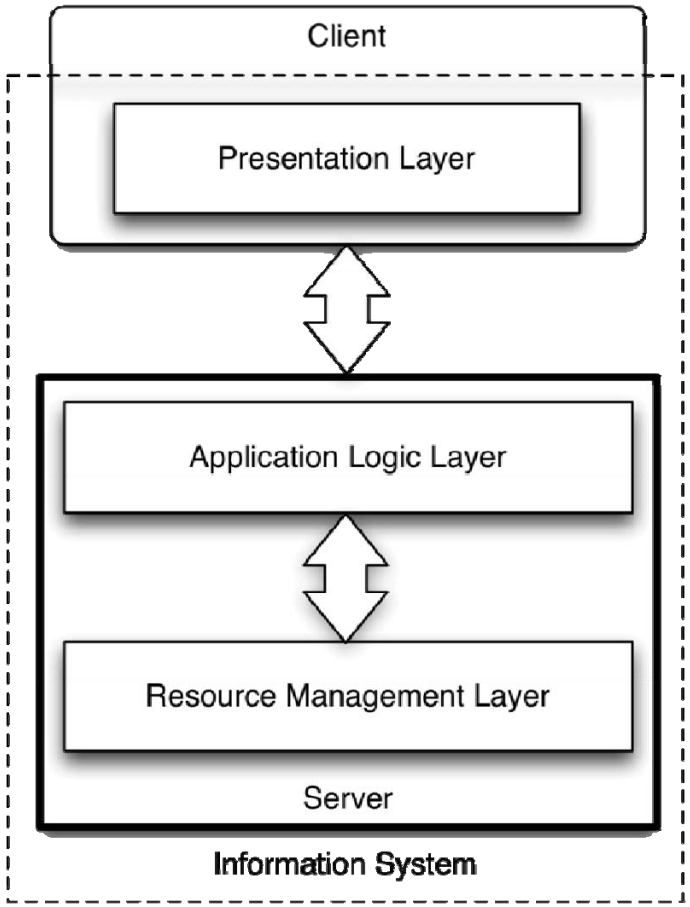


- Direct result of the computer architectures used several decades ago
  - Mainframe-based systems,
  - Interaction through dumb terminals (keyboard + computer screen) - clients,
  - Efficient usage of CPU.
- They represent monolithic systems
  - All layers are merged into the single tier
- Canonical example of legacy systems
  - No entry-points from outside,
  - No API provided to access the system functionality and resources,
  - Most popular method of integration is screen-scraping (simulating user and parsing the screen outputs) – expensive, ad-hoc, not elegant, not efficient.

STI · INNSBRUCK

- Advantages of 1-tier systems
  - Merging of layers to optimize performance (use of low level languages, no context-switching, no complex data transformations, no networking overhead).
  - Zero client development, deployment and maintenance costs.
- Drawbacks of 1-tier systems
  - Monolithic piece of code – difficult and expensive to maintain (lack of documentation and qualified programmers).
- Software industry is moving from them since many years
  - Modern mainframe software is no more monolithic.
  - P2P networks of inexpensive small computers (PC) can provide equal power.

Architecture of an Information System
Two-tier Architectures



STI · INNSBRUCK

- Came with the emergence of PC (i.e., inexpensive hardware)
- Presentation layer can be detached from the layers below and deployed on the cheaper and smaller machines
  - Presentation layer can utilize the computational power available at PC,
  - Presentation layer can be tailored for different purposes without increasing the complexity of the remaining system.
- The best example of two-tier architectures is client/server architecture
  - Client – presentation layer.
  - Server – application layer and resource management layer.
- Clients can be
  - **Thin** – provide only a minimum of functionality, and
    - Easier to port/install/maintain, require less processing capacity (wider range of computers)
  - **Thick** – provide wide range of functionality
    - More sophisticated, offer richer functionality, large footprint and use considerable resources.

- Client/server is associated with many key developments in software for distributed systems
  - Remote Procedure Calls (RPC)
    - interaction between client and server,
  - Application Programming Interface (API)
    - stable published interfaces which can be used by clients.
    - specify how to invoke a service, expected responses, effects on internal server state.

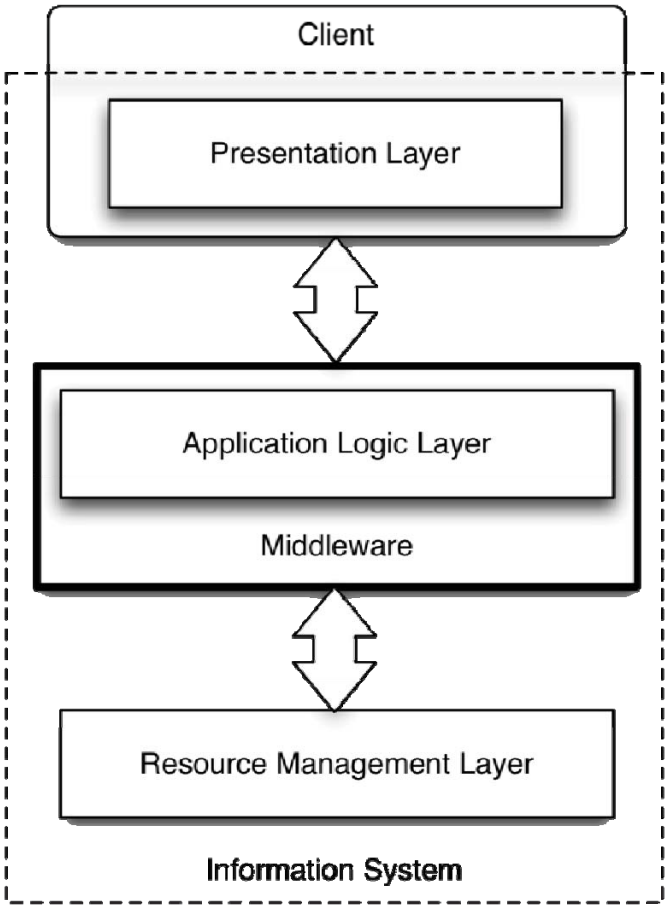
- C/S architectures became starting point for many crucial aspects of modern information systems
  - Services – individual server programs responsible for app logics,
  - Abstraction of service interface from the implementation,
  - Standardization to address need for services (the last example are Web services).

STI · INNSBRUCK

- Advantages of two-tier architectures
  - Key operations can be executed faster because ALL and RML are together.
  - Portability through separated presentation layer.
  - Provision of multiple presentation layers.

- Drawbacks of two-tier architectures
  - Server issues
    - Single server can serve only limited number of clients (connections, authentication, context maintenance, resources allocated to ALL and RML).
    - Typically they are running on a less expensive machines – limited scalability.
  - Client issues
    - Clients cannot be easily made to connect to the different servers,
    - They need to integrate different communication services
    - Understanding of different APIs makes clients bigger and more complex, dependent on various systems which reduces their useful lifetime.
    - Client is responsible for integration of two or more servers – extra app layer is embedded in client
    - Ad-hoc procedure of customizing the client to a new server must be repeated from scratch for every possible combination of servers.

Architecture of an Information System
Three-tier Architectures



STI · INNSBRUCK

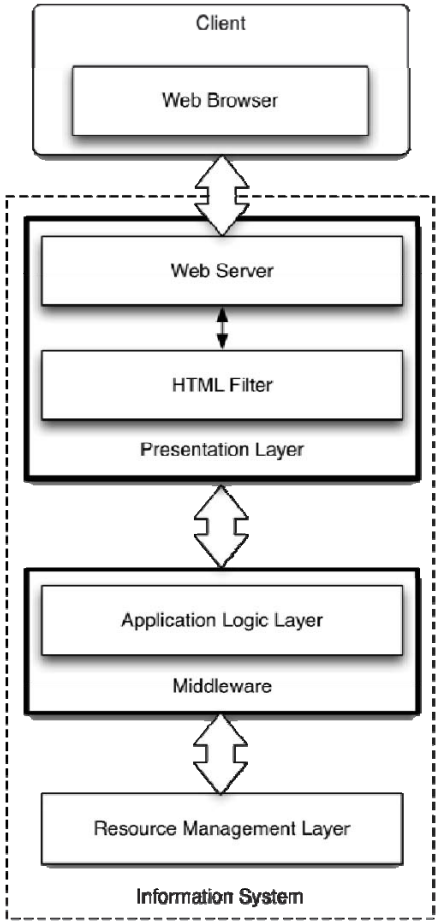
- Problems of two-tier architectures
  - Proliferation of servers with stable and published interfaces.
    - Creation of islands of information – clients can communicate only to the particular servers.
  - LAN-based increase of network bandwidth.
    - Opening a possibility to integrate different servers
- Three-tier architectures are a solution to this problem
  - The additional layer between clients and servers is used for the integration of underlying systems – application logic supporting integration resides in it.
- Finally we have clear separation between each of the three layers
  - Presentation layer resides at the client
  - Application logic resides at the middle tier – MIDDLEWARE.
  - Resource management layer is composed of all servers integrated by the 3-tier architecture.
    - Each of these servers may have its own ALL and RML.
    - Programs running on ALL are clients working in client/server setting.

- Scalability advantages of 3-tier over 2-tier architectures
  - In 2-tier architectures scalability is achieved by increasing the server power – very expensive.
  - In 3-tier architectures scalability can be accomplished by running each layer in a different server
    - Application logic can be distributed across several nodes; it is less tied to RML, more portable and reusable.
    - Communication becomes much more expensive.
- Concepts introduced in 3-tier systems are:
  - Clear interfaces of resource managers (e.g., ODBC and JDBC).
  - Standardized interfaces for application logic code.
- Three-tier systems are targeted to be used for integration issues
  - Middleware supporting services which can be used by application logic
  - This created another standardization wave (X/Open for transactions, CORBA).

- Advantages of 3-tier architectures
  - Additional tier where the integration logic resides.
  - Performance loss is compensated by the flexibility provided by the additional tier and the logistics services.
  - Distributing functionality across nodes – boosting scalability and reliability.

- Disadvantages of 3-tier architectures
  - Legacy problems – when integration must happen between different 3-tier systems or over the Internet.
  - They are not designed for inter-network integration neither have standards to bridge the gap between 3 tier systems.

Architecture of an Information System
N-tier Architectures



- Applies the three-tier model to its full generality and takes into account the relevance of the Web as access channel.

- Resolves some problems of pure 3-tier systems
  - Inter-system linking and integration.
    - RML may include also full-fledged 2-tier and 3-tier systems.
  - Connectivity over the Internet.
    - Web servers as part of the presentation layer – an additional layer.

- The architecture of N-tier systems can be very complex with many different tiers
  - They are build for future building.

- Disadvantages
  - Too much middleware involved, often redundant functionality
  - Difficulty and cost of developing, tuning, maintaining and evolving increases exponentially with the number of tiers.
  - It may be difficult event to identify where the system begins and where it ends.

- With each layer architecture gains flexibility, functionality and possibilities for distribution.
- Each layer introduces performance issues, cost of communication and complexity in terms of management, tuning and maintenance.
- Web services are yet another tier on top of the existing ones.

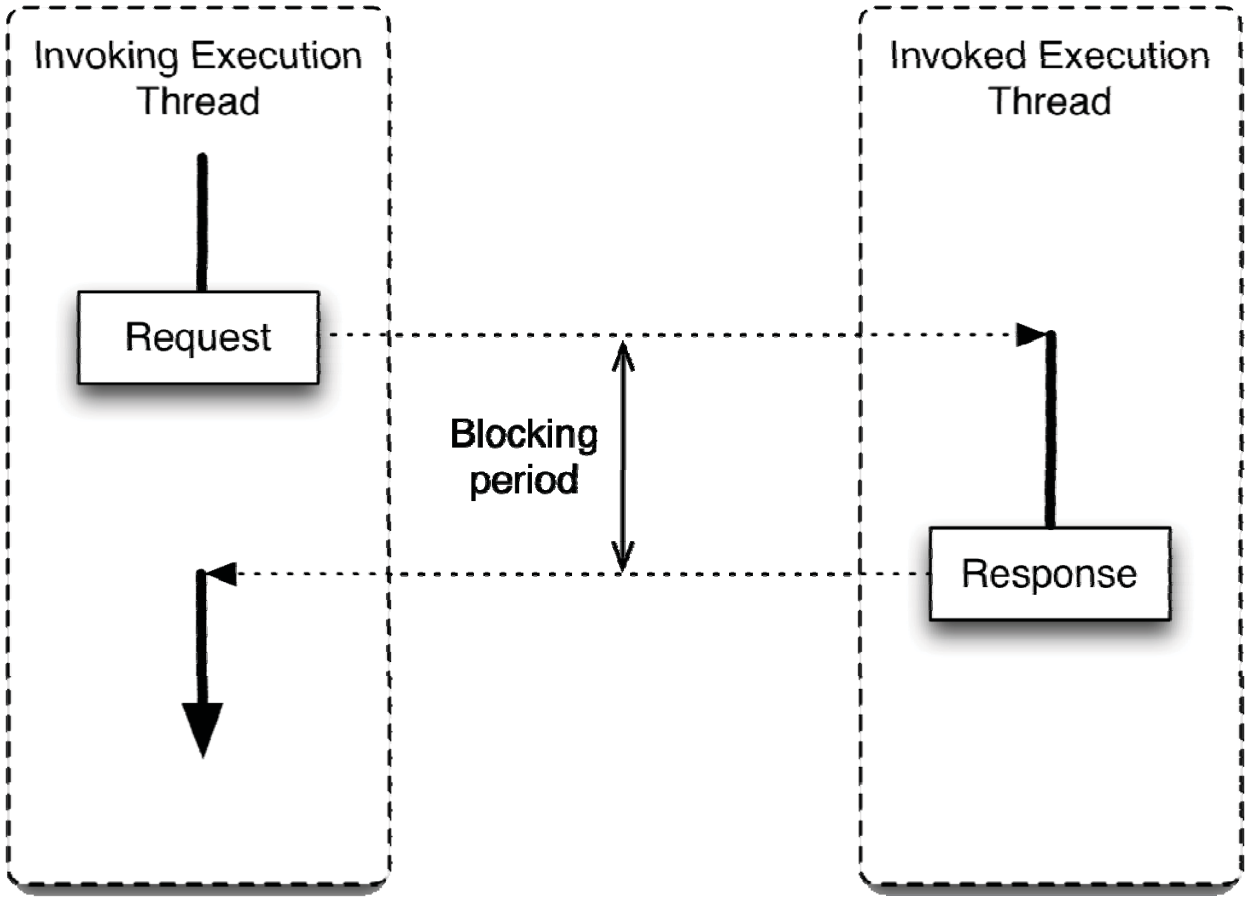
Technical Solution

Communication in an Information System

- Tiers introduce necessity of software interaction and communication between them.

- The interaction can be characterized as
  - Synchronous (Blocking) interaction
    - Parties in the communication are waiting for the interaction to conclude before doing anything else.
  - Asynchronous (Non Blocking) interaction
    - Parties are not waiting for the interaction to conclude.

Communication in an Information System
Synchronous or Blocking Calls

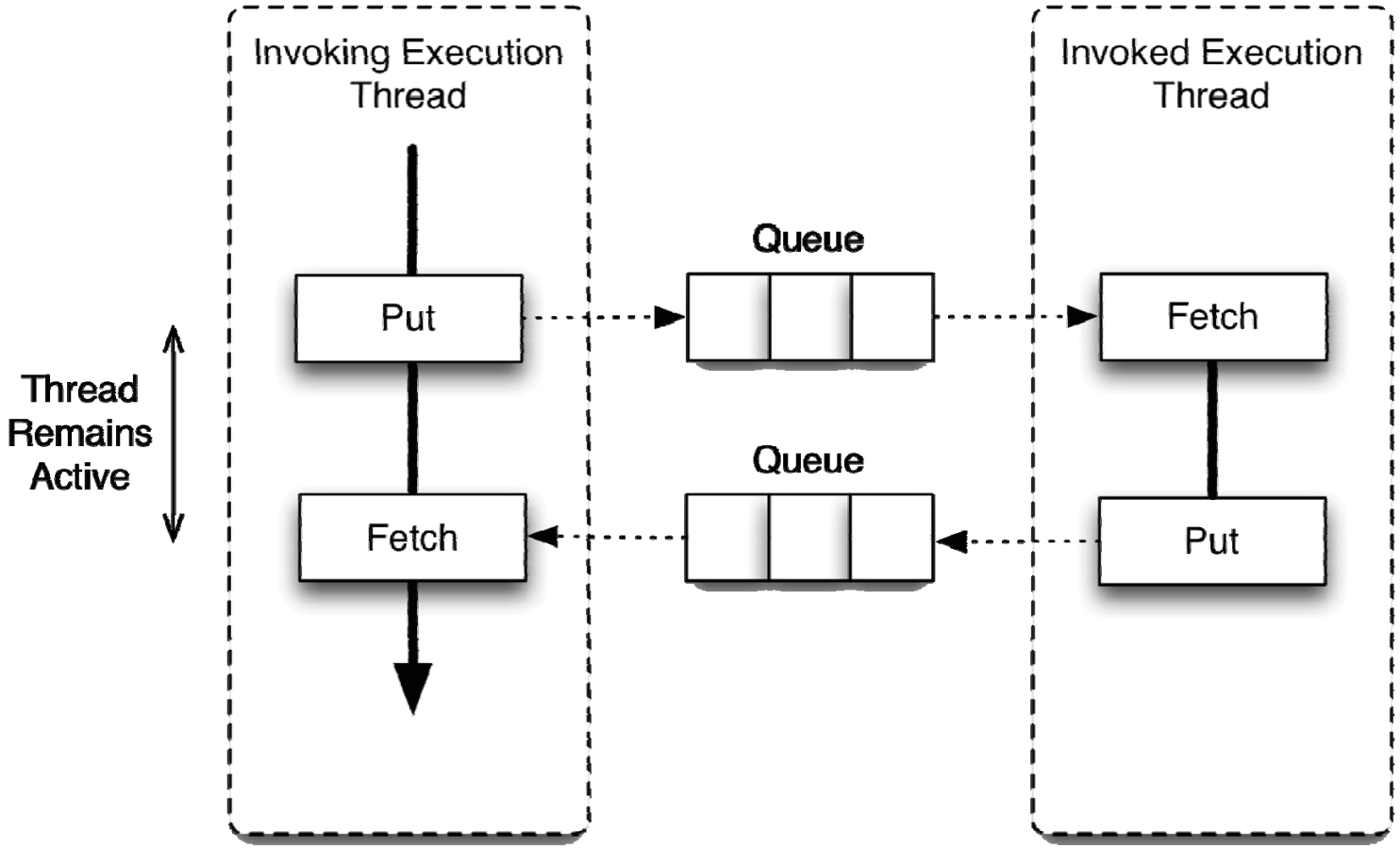


- Calling thread is blocked until the response arrives.
- Design is much simpler, easier to understand, and the state of the calling thread is preserved during the call.
- Code blocks to make the call and process the response are strictly sequential.
- Debugging and performance analysis are simpler.
- This is a dominant way of communication between tiers.

- Disadvantages of synchronous calls
  - Significant waste of time and resources.
  - Waiting process can be even swapped out of memory which increases even more the time to respond.
  - Danger of running out of connections.
  - Components are highly tied (coupled) which hinders distribution.
  - Complex to use when there is many tiers involved.
  - It requires both the caller and the called to be present and online during the interaction time
    - Reduces fault tolerance.
    - Makes maintenance procedures more complex (during the system upgrade everything goes down).

Communication in an Information System
Asynchronous or Non Blocking Calls

STI · INNSBRUCK



- The message is sent and then at some later time the program checks whether an answer has arrived.
  - The caller is not blocked, it can do some other tasks thus saving time and resources.
  - No need to coordinate the interaction.

- It can be used as an underlying mechanism to support Remote Procedure Calls (RPC) under design constrains (e.g., limited number of connections opened).

Communication in an Information System
Asynchronous or Non Blocking Calls

STI · INNSBRUCK

- Advantages
  - Reduces problems with connection management, dependencies management, fault tolerance, format representation.
  - Especially useful when the required communication pattern is not request/response by nature
    - Information dissemination.
    - Event notification.
    - Publish/subscribe solutions.

- Disadvantages
  - Messages must be stored at some intermediate place
    - But additional functionality can be added to it to promote them as full message brokers
      - Filtering and controlling message queue and flow of messages,
      - Implementing complex distribution strategies,
      - Manipulating format/content of the messages

Illustration By a Larger Example

Google Search Engine
Architecture

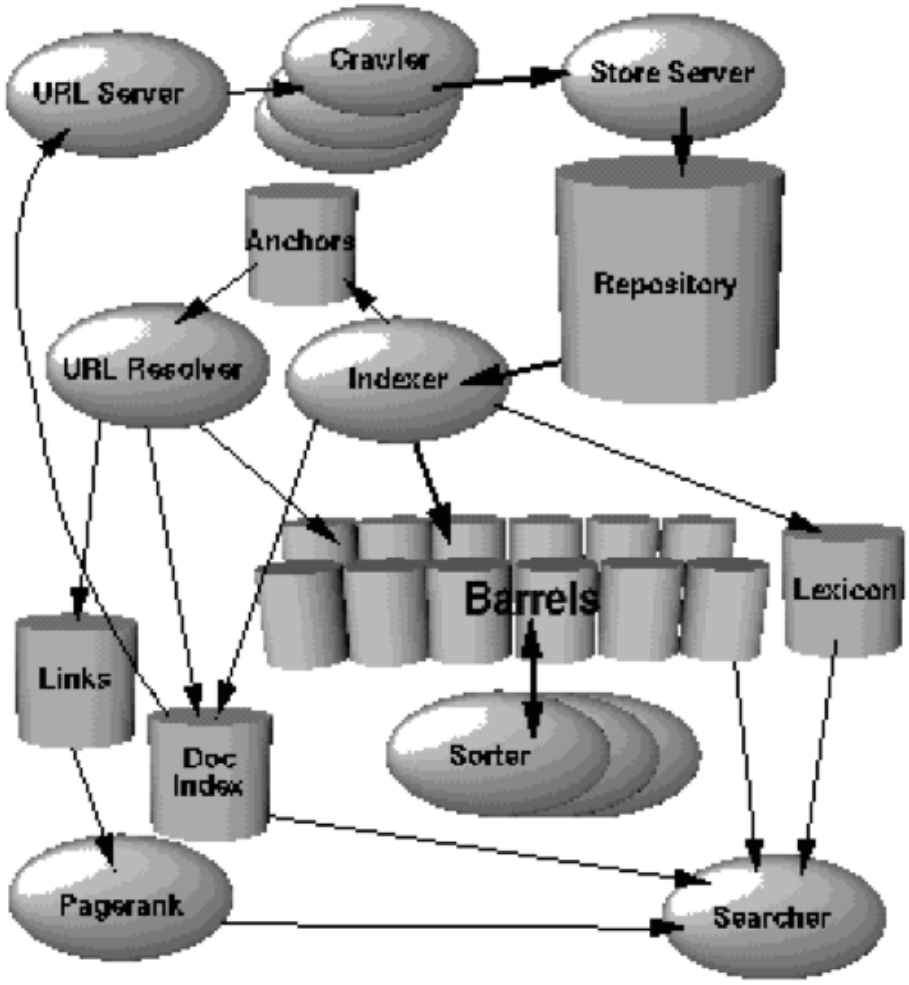


Figure taken from Brin, S. and Page, L. (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

Google Search Engine
Functionality

STI · INNSBRUCK

- Distributed crawlers are feed by URLServer (maintaining lists of URLs to be fetched) and they store fetched pages into the Repository over StoreServer which compresses them.
  - Every page has associated docID
- Indexing functionality is provided by indexer and sorter
  - Indexer functionality
    - Parsing of the stored documents
    - Converting documents into a set of word occurrences called hits.
    - Hits are distributed into the barrels.
      - Partially sorted forward index (document and associated words).
    - Parses links and stores them into anchor file.
  - Sorter functionality
    - Resorts data in barrels to generate inverted index (words and associated documents).
    - Generates list of words and offsets in the inverted index.
  - Lexicon – generates lexicon to be used by searcher
- URL Resolver responsibility is to
  - convert relative URLs into absolute URLs and then into docIDs
  - It puts the anchor text into forward index associated with docID where URL points to.
  - Generates a database of links which are pairs of docIDs (used by PageRank).

Google Search Engine
Division by layers

STI · INNSBRUCK

- Presentation Layer
  - Crawler
  - Searcher
- Application Logic Layer
  - URL Server
  - Store Server
  - Indexer
  - URL Resolver
  - Sorter
  - PageRank
- Resource Management Layer
  - Repository
  - Anchors
  - Links
  - Doc Index
  - Barrels
  - Lexicons

Extensions

Extensions

STI · INNSBRUCK

- No real extensions beyond N-tier architectures and presented communication paradigms.
- It would be worth seeing how the recent trends are aligning with the established architectural and communication solutions.
- Web 2.0
  - The emerging trend of Web2.0 applications requires the appropriate solutions in the area of Web services and how they reflect through the established concepts and architectures.
  - RESTful services and Web APIs.
- Semantic Web
  - The current Web service technologies are lacking solutions which would allow scalable automation and data and process heterogeneities reconciliation
  - Heavy-weight (WSMO, OWL-S) and light-weight (Micro-WSMO, hRESTS) SWS solutions can contribute to the solution of the problem.

Summary

Summary

STI · INNSBRUCK

- This lecture gave an overview of the design, architecture and communication issues in distributed information systems
- Design of an information system
  - Three layers: presentation, application logics and resource management.
  - Top-down vs. bottom-up design approaches.
- Architecture of an information system
  - One-tier, two-tier, three-tier and N-tier architectures.
- Communication in an information system
  - Blocking vs. non blocking communication paradigms.
- The concepts introduced in this lecture are a foundation for understanding, judging and reasoning about different Web service solutions.

References

References

STI · INNSBRUCK

- Mandatory reading
  - Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. Web Services - Concepts, Architectures and Applications. Springer-Verlag, 2004. **first chapter**
- Optional reading
  - Ariel Ortiz Ramirez. Three-Tier Architecture, Linux Journal, July 1st, 2000, http://www.linuxjournal.com/article/3508
- Wiki and Web references
  - Multitier architecture http://en.wikipedia.org/wiki/Multitier_architecture
  - Middleware http://en.wikipedia.org/wiki/Middleware
  - Client/Server http://en.wikipedia.org/wiki/Client-server_architecture

Next Lecture

STI · INNSBRUCK

#	Title
1	Distributed Information Systems
2	**Middleware**
3	Web Technologies
4	Web Services
5	Basic Web Service Technologies
6	Web 2.0 Services
7	Web Service Security

Questions?

STI · INNSBRUCK