521266S

Distributed Systems (spring 2012)

Course work specification

Overview

- The task is to design a distributed system/application running on 3 or more nodes
  - Node: a computer with a local web server and a display
- More specifically, design and implement:
  - A service utilizing a local web server in each node running the service
  - Distributed communication between the application peers, i.e. each application instance can exchange data with other (2 or more) instances
- <u>The course work is to be implemented in groups which you have already formed</u>
- No distinct topics or themes, but technical requirements
  - All groups must have their own topic
- Deadlines for submitting the course work
  - 4.6.2012: Early bird deadline, gives 1 point towards point total
  - 1.10.2012: Final deadline (FIRM - no exceptions)

Technical requirements

1. On each node, the system must utilize a local web server and a web technology of choice on the browser side (UI)
2. There must be at least three nodes / peers running the service
3. The participating peers must:
   - Be able to exchange messages using a well-known, common protocol (messaging)
   - Be able to express their state and readiness for sessions towards other peers (discovery)
   - All peers must log all important events of their activity
     - Locally or to a central location (filesystem or database)

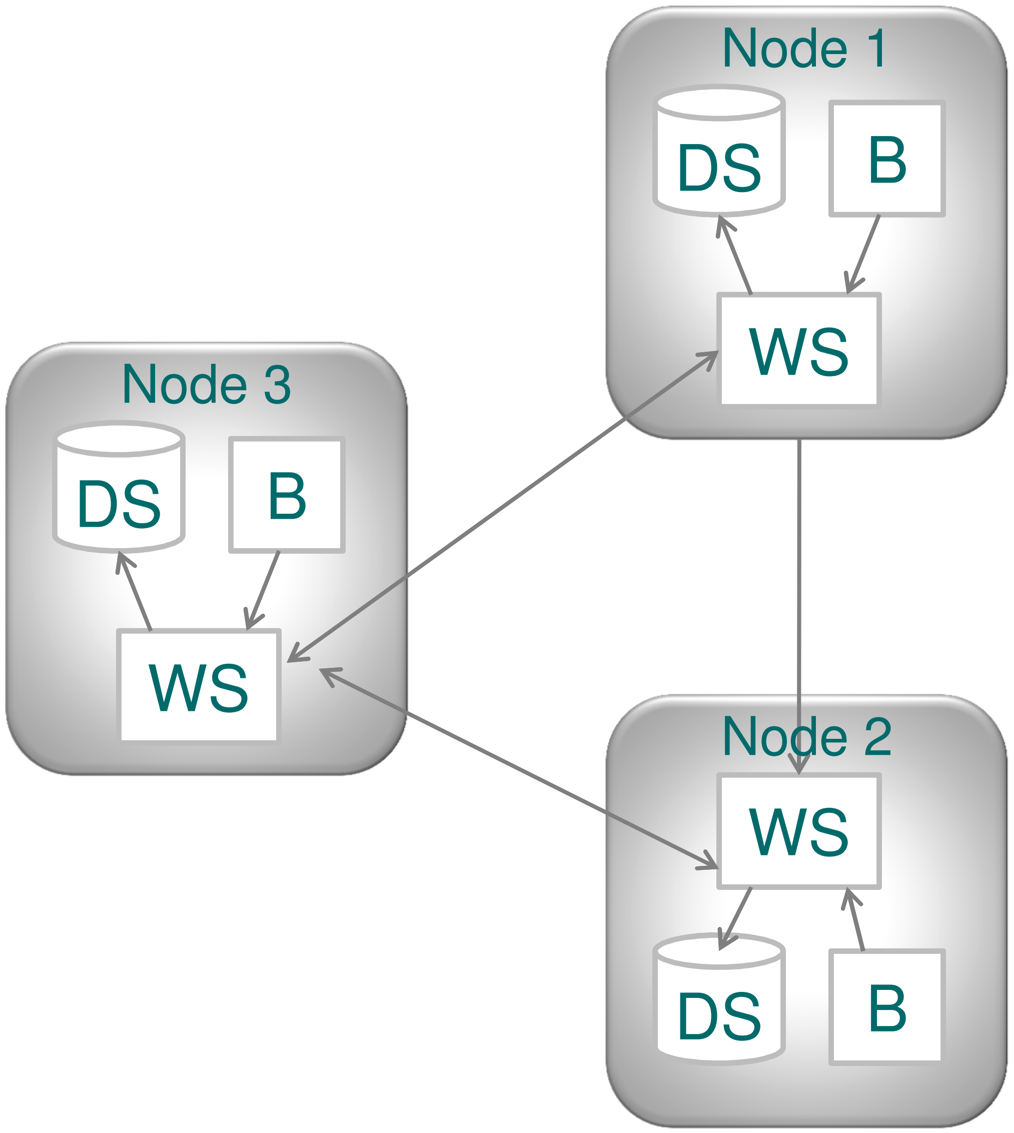
Finally, peers must work together, forming a functional distributed application running on 3 or more nodes

Example technologies to use

(feel free to use any you can, though...)

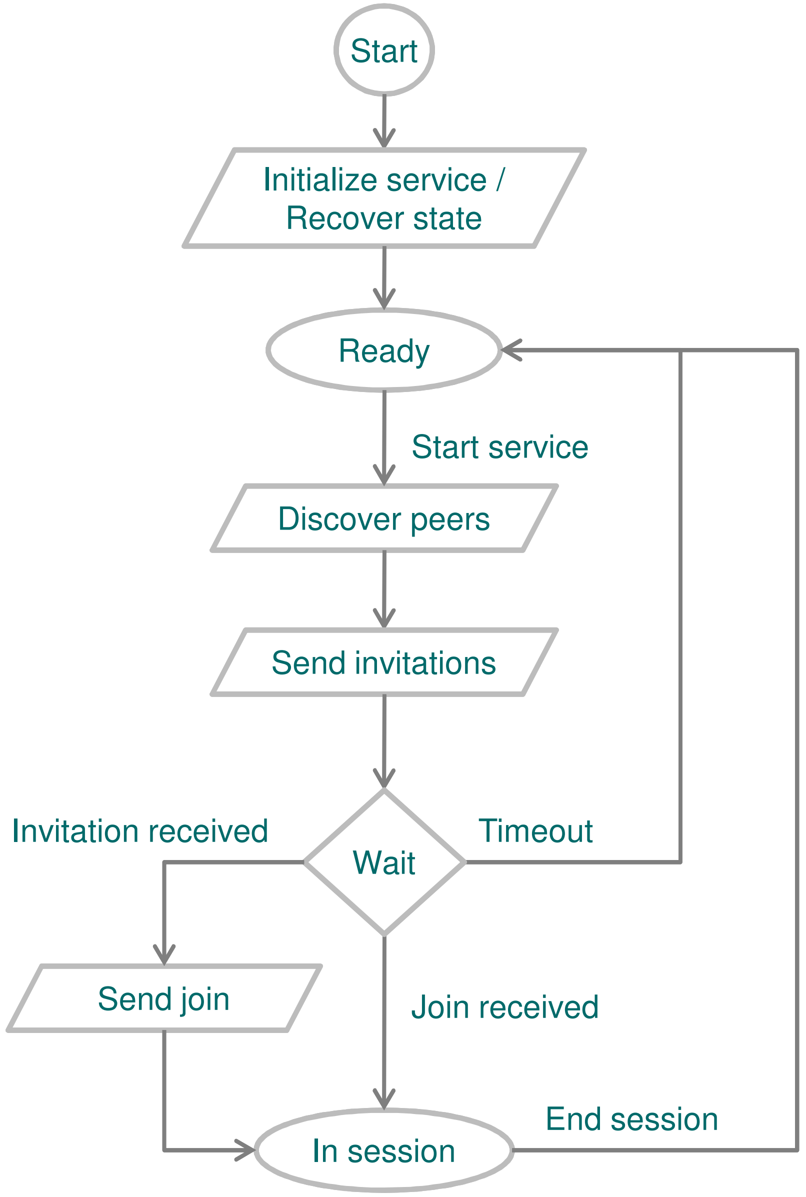
- Local web server / UI technologies
  - Apache HTTP Server Project or Apache Tomcat (Java Servlet, JavaServer Pages)
  - HTML (5), JavaScript, PHP, Flex, Flash, Applet, Servlet, etc.
- Communication between nodes:
  - RPC: e.g. REST-RPC, RPC-tunneling, SOAP, XML-RPC
  - Message-oriented: e.g. XMPP, JMS, MQTT
  - Any other mechanism, as long as it works
- Express state and ”readiness” for sessions towards other peers
  - Highly application specific
    - ”online”, ”ready”, ”in-game”, ”busy”, etc.
- Discovering other nodes
  - Hard-coded addressing in each app, shared database, UDP broadcast, DNS-SD, don’t use IP addressing, etc.

Example conceptual architecture



WS =	Web server
B =	Browser
DS=	Data Storage

Example of one-on-one peer state chart



General guidelines and tips

- **Focus of this course is not on the user interface**
- Feel free to use virtualization technologies for emulating several individual machines on a single host (e.g. VMWare, Virtual Box)
- Peers don’t have to be identical
  - e.g. one can act as a monitor/admin peer and another one as a sensor/actuator
- **Get your course work proposal approved by the TAs before starting implementation**
  - Mail a short proposal with initial design and ideas on implementation to ds@ee.oulu.fi
  - Cover letter containing your group name, student names and IDs
- Using mashups / online APIs is also possible, i.e. use public APIs of Google, flickr, youtube, facebook, etc. for your data or content needs
  - http://www.programmableweb.com/apis
- Don’t overcomplicate things – good initial design will help a lot

Topic examples

- Games
  - Othello, Five-in-a-row, Four-balls-straight, Memory game, Battleship, Yahtzee, etc.
- Shared whiteboards, workspaces, collaborative apps, image sharing, etc.
- Sensor applications, request data from other nodes, aggregate in other ones
- Groups are encouraged to propose their own ideas

Grading

- Pass (minimum requirements)
  - Implement the required funtionality (see Technical requirements slide)
  - Good documentation and readable, well commented code (in English)
- Extra points (up to seven)
  - Client UIs do not use polling to implement notifys (reverse AJAX, HTML5, etc.)
  - Communication between peers is encrypted
  - Fault tolerance: restarting the browser or web server is handled gracefully, e.g. state is recovered or other peers notify users accordingly (+1)
  - No hard-coded peer list for discovery or messaging
  - Provide a _fully_ functional installer (e.g. RPM, VM, batch file...)
  - Submit and have the course work approved before 4.6.2012
  - Implement the UI of your course work according to the ”UBI-hotspot media card” - http://www.ubioulu.fi/sites/default/files/UBI_Challenge_UBI-portal_Media_Card.pdf

Deliverables

1. Proposal of exercise work topic to TAs
   - Before the final submitted work
   - Grading: accepted / rejected
2. A thorough report about your system design
   - ”Final document”
3. All source codes and resource files
4. Installation instructions (or the installer as defined for an extra point)
5. Demonstration of a working system and any extra features to TA:s