Politecnico di Milano

Middleware Technologies for Distributed Systems

13th of July 2015

Rules:

- **You are not allowed to use books, notes, or other material.**
- **You can answer in Italian or English.**
- **Total time for the test: 1.5 hours.**
- **Check the back of this page for any additional exercises.**

---

1. Use **Java** to implement a MsgBatcher class that holds a (finite) set of messages and sends them, on request, in batch. Suppose we have a Message class with a method void send(). MsgBatcher provides a method void enqueue(Message) to add a new message to the batch. It suspends the caller if the MsgBatcher is full (the maximum number of messages that can be enqueued is provided in the MsgBatcher constructor). A method void sendAll() is also provided to send all messages enqueued up to that moment (it empties the MsgBatcher).

   Organize synchronization that will take care of the fact that sending a message may take a long time.

   Optional: implement the sendAll method so that the sending is performed asynchronously w.r.t. the caller (i.e., in a separate thread, which should be started at MsgBatcher creation time and reused for each sending).

2. Write in **TinyOS** the Counter module, which provides a command to increment the counter and an event that is notified when the couter has been incremented 100 times. The event provides the total number of increments done since the last reset. The counter can be reset with an ad-hoc command, but it also resets automatically every 10 seconds. Introduce two additional commands, start and stop, to enable or disable this "auto-reset" feature.

   Write the interface and the implementation of the module. The following interfaces may help you:

   ```
   interface Timer<precision_tag> {
      command void startPeriodic(uint32_t dt);
      command void startOneShot(uint32_t dt);
      command void stop();
      event void fired();
   }
   ```

3. Use **RMI** to expose a remote method capable of returning a 7-day weather forecast. The forecast should contain, for each day, a lower and higher temperature estimate, as well as a rain probability. The input to the exposed method should be the postal code (e.g., 90210) of the place for which we want the forecast.

   Provide both the server-side code required to implement and publish the service, and the client-side code required to access the remote method correctly.

4. With reference to the **REST** architectural style explain the distinction between resource and representation. Why do we make this distinction? What advantages does this separation give us? Also what role do the representations play in advancing the overall application state?

5. The pavillions at EXPO2015 have installed innovative sensors that can register people entering and leaving each pavillion. Each reading produces a tuple of the type <timestamp, pavillionID, personID, direction>, where direction is 1 if the person entered the pavillion and -1 if he/she left the pavillion. Use **Hadoop** to solve the following problems.

   - Say we want to analyze the mexican pavillion, and that we want to know the top three days, between the 1st and the 31st of July, in terms of number of entrances. Describe the Hadoop job(s) needed to produce a list in which we have the three dates ordered by decreasing number of visitors.

   - Say we want to analyze the EXPO attendance numbers between the 1st and the 31st of July. Describe the Hadoop job(s) needed to produce, for each day in that time span, a tuple containing the date, the ID of the pavillion that received the most visits, and the number of visits it received.

   NOTE: you do not have to write code; simply describe the dataset structures and what you would do in the various job(s) steps.