Corso di Laurea in Informatica — Università di Bologna
Progetto per il Corso di Sistemi Complessi
AA 2014/2015

Ozalp Babaoglu Stefano Ferretti
Dipartimento di Informatica — Scienza e Ingegneria
Università di Bologna

General Information, how and what to submit

The student must select one among the three proposals reported in this document.
The project must be carried out individually.
The project must be handed in electronically, by e-mail to s.ferretti@unibo.it. The deadline for submission is 1 July 2015, at 23:59. Please use your official university email address (@studio.unibo.it); the subject of your mail must be "**CS Project 2014-2015**"; you will receive a confirmation message (this may take a couple of days, please be patient).

Your mail must contain a compressed archive (only .tar.gz and .zip formats are accepted; avoid .rar and other formats) containing the following items:

- Source code you have developed (it is not necessary to include the source code of PeerSim or NetLogo);
- A short paper, in pdf format, describing the experiments that have been performed and a detailed discussion of the results.

The source code must be adequately commented. The paper can be written either in Italian or English, and should be organized like a technical paper with a title, abstract and (short) bibliography. You can use any document editor to write the paper, but you must send only the pdf version to us.
This project must be done individually. No sharing of source code or technical papers is allowed, either in part or in full. However you can — and are actually encouraged to — discuss any issues with your fellow students and/or with the instructors.
We suggest you write no more than 16 pages using the LaTeX or Word/OpenOffice template from Springer LNCS.

- Doc template: http://www.moreno.marzolla.name/teaching/CS2012/word_template.zip
- LaTex template: http://www.moreno.marzolla.name/teaching/CS2012/latex_template.zip

You are required to put your full name, e-mail address and personal ID number (numero di matricola) on the paper, on each source file and in your mail.

Grading policy

These are the minimum requirements of your project:

- You must define a model for one of the projects described in this document. Extensions to the basic model are encouraged.
- You must implement the model. In case you select PeerSim as the simulation tool, you should use the cycle-driven simulation engine of PeerSim; your simulator must be configurable by means of the standard PeerSim configuration file.
- The paper must describe the model (and the extensions you have implemented, if any), the simulation experiments and the results. You need to *explain* the results, not just present them. The paper might describe implementation details if necessary, but keep in mind that we can look at the source code, so be concise.

Write a simple model, focus on a limited set of experiments and demonstrate that you actually understand the results. You have limited time, so use it wisely.

If you are interested in these topics (e.g., you want to build a better model, or you want to study similar systems), you are encouraged to talk to us when you are looking for a thesis topic.

Project Proposal 1: Percolation Thresholds

Introduction

Percolation theory describes the behavior of connected clusters in a graph. There are two kinds of percolation models: *site percolation* and *bond percolation*. In *site percolation*, a site (a vertex of the graph) is "occupied" with probability p or "empty" with probability $(1 - p)$; it is possible to go from an empty site to another neighbor open site. In *bond percolation*, sites are always empty, but their *bonds* (graph edges) are open with probability p, or closed with probability $(1 - p)$; in this case, it is possible to go from a site to a neighbor when their related bond (link) is open.
Regardless of the model, the question is the same: for a given p, what is the probability that a path exists between top and bottom of the graph (i.e., a connected cluster exists that allows to cover the entire graph)? In percolation theory, a threshold p_c exists such that an abrupt change in the percolation probability occurs for p values lower than p_c ($p < p_c$) and those higher than p_c ($p > p_c$). The p_c value depends on the type of the problem (*site percolatio*n or *bond percolation*), and on the graph topology.

Project Goal

Write a program to establish the percolation threshold (both *site percolatio*n and *bond percolation*) of any graph that is given as an input. The program should be structured as two modules: a graph generator, and a percolation threshold calculator.
You need to test your software with random graphs generated according to different models discussed during the course (Erdos-Renyi, Watts-Strogatz, Barabasi-Albert, etc.). Other topologies described in the literature or those corresponding to real networks should also be considered. Critical percolation thresholds for some example topologies can be found in the references.

References

- https://en.wikipedia.org/wiki/Percolation_threshold
- https://en.wikipedia.org/wiki/Percolation_theory

Project Proposal 2: Dynamically adaptive DHTs

Introduction

Typical Distributed Hash Tables (DHTs) assign to nodes a key space that does not take into consideration the popularity of keys. This might result in a higher overhead at some nodes. Possible causes of unbalanced load are:

- A node has a bigger portion of the keyspace (this is usually solved during the design of a DHT);
- Uniform partitioning of the keyspace among nodes, but an interval contains more data items than other intervals (this issue is usually solved by choosing a good hash function that uniformly distributed keys in the hash table);
- Uniform partitioning and uniform distribution of the data items, but there are certain items that are more requested than others;

The problem is that an unbalanced load might lead to a lower robustness and a lower scalability.

Project Goal

The student is asked to create a DHT model that is able to dynamically adapt the key space management based on the key popularity. The student should assume a sort of popularity model (keys that are popular for a certain amount of time, with flash crowd requests and with popularity that decreases exponentially with time); then, the study should assess if it is feasible to redistribute in some way the key space or devise some temporary strategy, where neighbour nodes (in the DHT) "help" those that are overloaded.
The student should choose a specific DHT (e.g. Chord, CAN, Pastry), take (or implement) a model for this DHT (see the references, where a list of DHT models is available), make some first tests by simulating flash crowd requests for some popular keys. Then, the student should try to adapt/modify the model to simulate the possibility of distributing/replicating popular keys among nodes in the DHT. Note that there are several possible alternatives here:

- an overloaded node *n* might "ask" their neighbors to store a popular key *k*; thus, if one of these nodes receives a request for *k,* instead of routing the request to *n*, it answers directly;
- use some different caching strategy.

Note that depending on the assumed flash crowd model, the costs for doing this adaptation might result high, making it unfeasible.
Suggestion: keep the model simple, since there are several possible alternatives, and one should focus also on different models for the flash crowd requests.

References and Notes

- See the slides of the lectures on DHTs and the related references
- There are some available implementations of DHT models in PeerSim. http://peersim.sourceforge.net/

Project Proposal 3: Study of connection patterns in Opportunistic Networks

Introduction

A *mobile ad hoc network* (MANET) is a continuously self-configuring, infrastructure-less network of mobile devices connected without wires. Each device/node in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. The network of connections creates a graph that changes continuously. Thus, the primary challenge in a MANET is to maintain connectivity.
Links between nodes are created based on some constraints, such as physical proximity (due to limited communication range) and maximum degree (i.e., number of active connections). Thus, assuming that all nodes can communicate with other nodes within a given coverage area represented as a circle with radius *r*, a link between two nodes is established if their distance is less than *r*, and both nodes have degrees that are less than the maximum possible degree.
When two connected nodes move beyond their coverage areas, the communication link is removed.
A naive strategy for handling connections might be as follows: nodes create links until reaching their maximum degree. When such a limit is reached, upon the arrival of a new node an existing connection is replaced with the new one. The connection to replace could be either the oldest one or one selected randomly and uniformly among the existing connections.

Project Goals

The student must simulate a MANET composed of a set of nodes that move randomly within a given area (i.e., an open rectangle that wraps at the borders, as a torus). The student should study how the obtained networks change when varying:

- the radius of the coverage area (with respect to the dimensions of the area where nodes move);
- the maximum degree;
- the speed of node movement;
- the replacement strategy.

An evaluation metric might be whether the network is fully connected or, alternatively, how the size of the giant component varies depending on the parameter settings.

References

- A simplified model of a MANET is available in NetLogo: http://ccl.northwestern.edu/netlogo/models/community/Lattice-Walking%20Turtles_15
- Wireless Ad Hoc Networks Bibliography: http://w3.antd.nist.gov/wctg/manet/manet_bibliog.html