UiO-1: On the topology consistency in MANET

Contact person: Ovidiu Valentin Drugan ovidiu@ifi.uio.no

Description:

Identifying the clusters in Mobile Ad Hoc Networks (MANETs) is a challenge, especially because of the network dynamics (i.e., connected nodes, connections between nodes). To perform clustering in Mobile Ad Hoc Networks one needs to know the number of clusters and the nodes which can be cluster heads. One can use the routing protocol as information source for clustering; OLSR is a routing protocol which is maintains the topology information at each node. It is important to investigate the consistency of the topology information available at each node.

Task:

Design and implement a mechanism to extract topology information from the routing protocol form OLSR of a node at equal time intervals during a simulation.

1. Extract topology information at different time intervals
2. Use mobility traces from different mobility models
3. Perform measurements in static networks
4. Use different settings for the routing protocol
5. Build some mechanisms to perform the comparison of the topologies (e.g., compute the Hamming Distance and Levenstein Distance between the topologies at the different nodes).

Required skills:

Java language or C is required and some experience in using Linux is of advantage.

Note: There is the possibility to use *Esper* (http://esper.codehaus.org) to implement the required functionalities.

References:

[1]http://www.olsr.org

[2]NEMAN, http://www.ifi.uio.no/forskning/grupper/dmms/papers/138.pdf

[3] R. W. Hamming, “Error detecting and error correcting codes,” Bell System Technical Journal, vol. 26, no. 2, pp. 147–160, 1950.

[4] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, vol. 10, pp. 707–710, 1966.

[5] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” Commun. ACM, vol. 7, no. 3, pp. 171–176, 1964.

UiO-2: On the community consistency in MANET

Contact person: Ovidiu Valentin Drugan ovidiu@ifi.uio.no

Description:

Identifying the clusters in Mobile Ad Hoc Networks (MANETs) is a challenge, especially because of the network dynamics (i.e., connected nodes, connections between nodes). To perform clustering in Mobile Ad Hoc Networks one needs to know the number of clusters and the nodes which can be cluster heads. The method proposed by Newman and Grivan in 2004 is a simple divisive method which identifies the communities of the network, by removing the links with the highest importance from the network. It applies recursively the following steps on the network: 1) compute for each link its importance score in the network (e.g., its betweenness score, defined by the number of geodesics or shortest paths going through a link); 2) remove from the network the link with the highest importance score; 3) update the modularity value of the current network(i.e., measure the fraction of the links in the network that connect nodes within the same community minus the expected value of the same quantity in a network with the same community divisions but random connections between the nodes).The algorithm is applied until there are no more links left. The communities in the network are given for the highest modularity value. Since, there may be possible inconsistencies in the topologies at the nodes in the network, it is important to investigate the consistency of the detected communities at each node.

Task:

Implement the Newman and Grivan method and use it to detect the communities in a network based on the topology information from the routing protocol OLSR

1. Extract topology information at different time intervals
2. Use mobility traces from different mobility models
3. Perform measurements in static networks
4. Use different settings for the routing protocol
5. Build some mechanisms to perform the comparison of the topologies (e.g., compute the Levenshtein Distance between the communities at the different nodes).

Required skills:

Java language or C is required and some experience in using Linux is of advantage.

Note: There is the possibility to use *Esper* (http://esper.codehaus.org) to implement the required functionalities.

References:

[1]http://www.olsr.org

[2]NEMAN, http://www.ifi.uio.no/forskning/grupper/dmms/papers/138.pdf

[3] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” Physical Review E, vol. 69, no. 2.

[4] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, vol. 10, pp. 707–710, 1966.

[5] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” Commun. ACM, vol. 7, no. 3, pp. 171–176, 1964.

UiO-3: Custom aggregate functions for Clustering in MANETs

Contact person: Morten Lindeberg (mglindeb@ifi.uio.no)

Description:

Most data stream management systems (DSMSs) supports aggregate functions like MAX(), MIN(), and AVG() that can be applied over sliding windows over a data stream. Custom aggregate functions can be added to the DSMS engine so that users can perform their own operations on the stream. Sparse Mobile Ad-hoc Networks (MANETs) often suffers frequent topology changes. For the table driven proactive routing protocol OLSR [1], it is possible to monitor the node topology by constantly polling data from the routing tables.

In this assignment, student are required to implement custom aggregate functions for monitoring OLSR routing table data using the Java-based open source DSMS Esper [2]. The custom aggregate functions should contain implementation of a clustering algorithm. For more details of such an algorithm, we refer to [3]. For emulating nodes in a sparse MANET, the students will use the network emulator Neman [4].

Task:

- Create data streams by polling OLSR routing tables from nodes in an emulated MANET (running Neman)
- Analyze the data streams using Esper DSMS running on either an emulated or external node
- Implement neighborhood prediction algorithm, clustering prediction algorithm or both into custom aggregate functions in Esper
- Analyze and describe your results

Required skills:

Java Programming, and Linux and Networking

References:

[1] OLSR Website, http://www.olsr.org

[2] Esper Website, http://esper.codehaus.org

[3] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” Physical Review E, vol. 69, no. 2. http://arxiv.org/abs/cond-mat/0308217

[4]Matija Puzar, Thomas Plagemann . “NEMAN: A Network Emulator for Mobile Ad-Hoc Networks (tech. report)”, Technical Report #321, University of Oslo, Department of Informatics, March, 2005, http://www.ifi.uio.no/forskning/grupper/dmms/papers/138.pdf

UiO-4: DENS application

Contact person: Katrine Stemland Skjelsvik katrins@ifi.uio.no

Description:

DENS (Distributed Event Notification Service) is a layer of software that facilitates communication between nodes in a highly unstable network. The communication is done using the publish-subscribe scheme. DENS provides user applications with an interface to subscribe to events and to notify of events. The delivery of subscriptions to publisher nodes and the notifications to the subscribing nodes is done internally. One important feature of DENS is that it is subscription language-independent and that language-specific plug-ins are used for parsing the subscription to be able to find the correct publisher nodes, filtering of events at the publisher nodes and matching of subscriptions and notifications.

The assignment is to write an application that uses DENS for subscribing to changes in the routing table and thus topology changes in a network, agree on the subscription syntax and customize the filtering and matching plug-ins (you can assume that all nodes are potential publishers and can therefore omit the parsing plug-in). The filtering plug-in, or the "watchdog" gets information from the routing daemon.

For more information on DENS you can look at the DENS presentation and the DENS article on your syllabus. Information on the OLSR daemon is available on http://www.olsr.org/ and OLSR RFC 3626: http://www.ietf.org/rfc/rfc3626.txt

Tasks:

- Understand DENS
- Understand the routing protocol OLSR and its implementation OLSRd
- Design/describe the subscription language syntax
- Find meaningful events of interest – what kind of topology changes could be of interest?
- Program the subscribing application and the "watchdog"
- Make a matching-plug-in for DENS for matching of subscriptions and notifications according to the subscription language you have defined
- Design some small test-setups and show how it works.

Required skills:

C language is required and some experience in programming on small devices is an advantage.

References:

[1] Supporting Multiple Subscription Languages by a Single Event Notification Overlay in Sparse MANETs Skjelsvik, K.S., Lekova, A., Goebel, V., Munthe.-Kaas, E., Plagemann, T. Sanderson, N., Proceeding of the ACM MobiDE 2006 Workshop, June, 2006

[2] OLSR RFC 3626: http://www.ietf.org/rfc/rfc3626.txt

UiO-5: MDS-Optimization

Contact person: Katrine Stemland Skjelsvik katrins@ifi.uio.no

Description:

MDS stands for MIDAS Data Space. It is a middleware service for mobile, wireless, dynamic networks and provides applications possibilities of storing, retrieving and querying of data like a traditional relational database. However, MDS chooses on which node or nodes a table is stored and directs the query to the correct node. Due to network dynamicity and the possibility of long-lasting partitions, MDS can replicate tables and thus increase data availability. This means that data also needs to be synchronized. In MDS, no data is actually deleted or changed: deleted records are only marked as deleted and updated records are kept as is, only marked as updated, and a new record is inserted. MDS can leave it to the application to perform conflict resolution, i.e., to choose between several versions of a record – in many cases this will be the most recent version.

Synchronization is done in two stages:

1. *EAGER*: Immediately ending INSERT, UPDATE-statements to all known nodes having a replica of the table in question
2. *LAZY* (Due to partitions, messages lost, nodes being temporarily shut down): Synchronize periodically and Synchronize when a new node enters the network (partition) by getting notified by the routing table

The lazy synchronization protocol itself is quite simple – each pair of node looks to see if they have any tables in common, and if so, the node having the highest node ID initiates the synchronization by sending a list of book-keeping fields of the records for each of these tables. By looking at this list, the nodes know which records are missing and send updates as queries that are handled in the same manner as for the so-called eager synchronization.

Tasks:

- First of all, look into the code and understand it
- Look into ways of optimizing it
    - What is actually sent – marshal/un-marshal the content in some ways
    - Check whether every node needs to synchronize with a new node
- Perform testing on PDAs – Nokia 770 and/or NEMAN Emulator. Make a small application and set up test scenarios and figure out a way of quantifying the improvements of your optimization

Required skills:

Java language is required and some experience in programming on small devices is an advantage.

References:

[1]A Data Sharing Facility for Mobile Ad-Hoc Emergency and Rescue Application, Thomas Plagemann, E. Munthe-Kaas, Katrine S. Skjelsvik, Matija Puzar, Vera Goebel, Ulrik Johansen, Joe Gorman, Santiago P. Marin. In Distributed Computing Systems Workshops, ICDCSW'07. June, 2007. P.: 85-88, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04279093

UiO-6: Sampling rate adjustment wrappers for CEP systems

Responsible: Jarle Søberg jarleso@ifi.uio.no

Description:

In sensor networks it is important to save power. This is because sensors are usually wireless and powered by batteries. For instance, transmitting data is a very power consuming process, thus reducing transmissions to a minimum is essential for long living sensor networks. Complex event processing (CEP) systems can obtain data streams from sensor networks, which mean that these systems need to have optimization functionalities like sample rate adjustment.

One interesting application domain is automated health care, where patients wear heart sensors that investigate the pulse. If the pulse becomes irregular, e.g., suddenly increases, it might be interesting to investigate the signals in more detail, thus increase the sampling rate. And of course: This should happen in real time.

The Esper CEP engine gives possibilities for extensions using Java. The task is to design, implement, and evaluate a sample rate adjustment wrapper for Esper. The user should have the possibility to state a query, and when the triggering events happen, the wrapper should send a signal that increases the sample rate. Investigate both "traditional" filtering/aggregation queries, as well as pattern-based queries, e.g. queries that investigate consecutive events.

Task:

- Which similar approaches have other researchers done in adjusting sample rates?
- What should the low/high sampling rate values be?
- Should the rate increase instantly or gradually when interesting events happen?
- How can this wrapper be efficiently distributed to the sensor nodes, if a non-centralized solution is suggested?

Required skills:

Java

References:

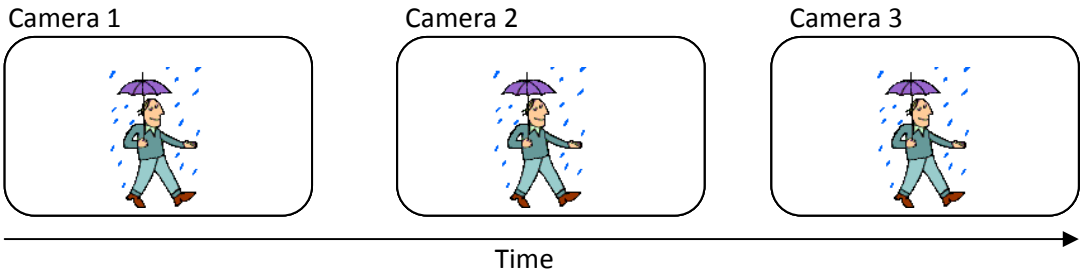
[1] Esper: http://esper.codehaus.org

UiO-7: Using complex event processing to capture movement over a multi-camera video stream.

Responsible: Jarle Søberg jarleso@ifi.uio.no

Descrition:

Esper is an open source complex event processing system library that provides functionality for capturing patterns of events. A pattern is a description of several events, for example that a person is first detected on one sensor, and afterwards on another sensor. This can be captured in Esper by stating a query. We want to use this functionality to read the data from three cameras and capture the movement of e.g. a person as shown in the figure below:



The time is shown as an arrow beneath the cameras. First, Camera 1 captures the person, and then Camera 2 captures this person. Finally, the person is captured by Camera 3. Obtaining the information from the cameras is done by using the OpenCV library. OpenCV is a library of programming functions mainly aimed at real-time computer vision. Two examples of actions that can be done using OpenCV is object identification and tracking, as well as motion detection.

Task:

- *Test movement over several cameras*. Use OpenCV to publish a data stream that consists of features identifying an object when it is detected. For instance, use the face recognition functionality. State a query in Esper that returns a notification if the object has been detected moving from Camera 1, via Camera 2, to Camera 3. Can your feature stream assure object equivalence, i.e., making sure that it is actually the same object that is moving in front of the cameras? If time, try to write a query so that a notification is only sent if an object is detected in Camera 2 after being detected in Camera 1 and before it is detected in Camera 3.
- *Investigate multi-modality*. Use Esper to verify that it is a person who moves in the video stream, by combining information from motion detection and object detection. Verify by investigating objects that are not persons.

Required skills:

Java/C/C++

References:

[1] Esper: http://esper.codehaus.org

[2] OpenCV: http://opencv.willowgarage.com/wiki/

UiO-8: A ns-2 Model of CPU Utilization at Wireless Nodes

Responsible: Stein Kristiansen steikr@ifi.uio.no

Description:

Mobile ad hoc networks (MANETs) are attractive in scenarios where computer network communication is desirable but where a proper supporting infrastructure is unavailable. In recent years, there has been a lot of research on techniques to make it feasible to perform multimedia streaming across such networks, most of which assume the wireless medium as the main bottleneck. However, often such networks largely consist of small, portable devices with stringent resource constraints. Practical real-world experiments with video streaming over such devices [1, 2] indicate that resources on the devices themselves (particularly the CPU) can pose as a bottleneck in an end-to-end path rather than the wireless links between them.

Because of its ease of management and low cost, network simulation is a common approach for evaluation of new network solutions. ns-2 [3] is one of the most popular network simulators today, mainly because of its rich and detailed set of computer network models. Even though ns-2's wireless networks are adequately modeled with regard to the network resources (available bandwidth, queue sizes, etc.), its models of node resources (CPU, memory, remaining battery power, etc.) are very limited, and in particular lacks a model on available CPU resources. In light of what is mentioned above, such a model can be crucial for the accuracy of simulation of scenarios involving streaming multimedia over MANETs. Hence, the aim of this task is to define and implement in ns-2 a model of available CPU resources on a wireless node. This model should allow future ns-2 experiments to take into account throughput limitations caused by CPU saturation on intermediate nodes in an end-to-end path across a MANET.

Task:

- Perform real-world experiments by streaming multimedia content across a wireless MANET consisting of resource constrained devices. For this purpose, a set of Nokia 770/810s will be provided to the students. A “how-to” on configuring the devices for streaming in ad-hoc mode can be found in [3]. By measuring the CPU utilization at an intermediate node while varying the rate of the traffic it has to forward, one can obtain an indication of the relationship between CPU utilization and the number of packets forwarded per second.
- Analyze the obtained CPU-measurements and derive a model of CPU utilization.
- Implement the obtained model in ns-2. Keep in mind that this involves the preliminary effort of getting sufficiently acquainted with the ns-2 codebase [3] and the wireless extension [5-7].
- Compare results from real-world experiments and ns-2 simulations to obtain a measure on the accuracy of the new model.
- Write a report of 10 pages (IEEE Style Formatting), on design, implementation and performance evaluation results.

Required skills:

C++ programming language is required and basic ns-2 knowledge is an advantage (involving simple OTcl scripting) [8]

References:

[1] M. E. Halvorsen. Video Streaming over MANETs: Reality or Fiction?. MobiMedia 08, 2008

[2] M. E. Halvorsen. Quality of Service Monitoring for Video Streaming in Mobile Ad-Hoc Networks. Masters Thesis. Department of Informatics, University of Oslo, 2008

[3] The ns-2 Manual. Available online: http://www.isi.edu/nsnam/ns/doc/index.html

[4] M. E. Halvorsen. How to Set Up Video Streaming on the Nokia 770. Available by request from your supervisor.

[5] The CMU Monarch Project. The CMU Monarch Project's Wireless and Mobility Extensions to ns. Available online: http://www.monarch.cs.cmu.edu

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparision of multi-hop wireless ad hoc network routing protocols. In Proceedings of the Forth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom98), 1998

[7] Fancisco J. Ros and Pedro M. Ruiz. Implementing a New Manet Unicast Routing Protocol in NS2. Department of Information and Communications Engineering, University of Murcia, 2004

[8] Marc Greis. Tutorial for the Network Simulator "ns". Available online: http://www.isi.edu/nsnam/ns/ns-documentation.html

UiO-9: ANA Task – Monitoring Brick for Mobile Node

Responsible: Thomas Plagemann, University of Oslo, plageman@ifi.uio.no and Andreas Mauthe, Lancaster University, andreas@comp.lancs.ac.uk

Description:

The principles of Autonomic Communication have been introduced in the ANA lecture. One of the key elements is the autonomic formation of compartments and automatic configuration of the protocol stack according to network characteristics and the requirements of a specific use case. The aim of this coursework is investigate the principles of autonomic communication and to analyze how they can be deployed in an actual communication context.

In order to organize themselves into compartments network nodes take physical properties into account such as kind of network and vicinity of nodes. The goal is that a set of nodes with certain physical properties should be able to organize itself into compartments according to a rule set. The infrastructure is dynamic and should be able to adapt to changes.

Task

Your task is to work as a team to design and develop a simple monitoring brick for mobile nodes. The goal is to implement a monitoring brick in ANA that is able to monitor in minimum the following the local load on a node, i.e., CPU load, and the link quality of the wireless network. Additionally, it might also measure the error rate to another wireless node and the available bandwidth.

Write a short report in the form of a short academic paper (10 pages maximum, IEEE format), which documents the design and implementation of your task, the experiments you undertook, the results you measured, and evaluation of those results, concluding remarks, any planned future work and references.

Required skills:

C language is required and some ANA knowledge

References:

See articles for ANA lecture