

Final Project: Policies and Suggested Topics

1 Final project policies

Each student is required to work on a final project for this course. The project can be either a programming project or a literature survey, in both cases related to the course topics. Students are allowed to work on their project either individually or by groups, although teamwork is advised since it typically gives better results. The following rules apply:

- groups of up to 3 students are allowed for programming projects
- groups of up to 2 students are allowed for literature surveys
- each group member must be actively and equally involved in the project work
- the project must be the work of the group, students are not allowed to copy someone else's work
- a progress report, a final report and a presentation are required from each group or individual project
- students are allowed and encouraged to collect information from any kind of source (books, papers, web sites, etc.) that is useful for the project development, but everything must be explicitly referenced in the project report in order to put the students' original contribution in good evidence
- students are allowed to consult with the instructor and the GSA for assistance related to specific details

Each project must focus on a topic related to the course content. A list of suitable project topics is provided in the following sections of this document. Additional topics can be proposed by the students, but must be approved by the instructor. Project submission steps and deadlines are:

Feb 26, 1 PM - group composition and topic proposal, to be sent by email to the instructor

Mar 25, 1 PM - progress report including task assignment to group members, to be sent by email to the instructor

Apr 15, 1 PM - final report, to be sent by email to the instructor and TA (grader)

Progress and final reports should be structured as articles, including an abstract, an introductory section describing the project general topic, the project details and results, the conclusions and a list of references. The IEEE double column template must be used, available for both LATEX and MS-Word at the following URL: <http://www.ieee.org/web/publications/authors/transjnl/index.html>.

A project presentation will be scheduled either on April 15 or April 22. Each group member must take part to the presentation, illustrating her/his contribution to the project development.

2 Literature surveys

The purpose of the project is to provide an exhaustive survey of the topic chosen, including details from seminal articles and reviews of specific applications/extensions. Topics can be related to either very recent research fields or well-established solutions. A good way to start is to gather information from the most important publication and citation repositories (e.g. IEEE, ACM, Google Scholar, CiteSeer) using advanced search features where available. If the topic chosen is too wide, the scope should be narrowed to something more specific and this should be properly justified (e.g. because such particular field is/was getting more interest from the research community). The following are suggested topics for literature surveys.

1. QoS provisioning for emerging network technologies: the case of optical networking
2. QoS provisioning for emerging network technologies: the case of ad-hoc networks
3. Virtual output queuing in router/switch architectures
4. Fair queuing scheduling policies for QoS provisioning
5. Experiments and performance evaluation of routers based on standard PC architectures and open source software (software routers)
6. Recent developments in interdomain routing protocols
7. Routing in peer-to-peer networks
8. Congestion control for high bandwidth-delay product networks
9. Performance issues of TCP over satellite links
10. Recent developments in differentiated services (DiffServ) architecture

Suggestions on how to effectively read papers and write surveys can be found on several website. The following are two good starting points:

<http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf>

http://www.unc.edu/depts/wcweb/handouts/literature_review.html

3 Programming projects

The following are suggested topics for programming projects.

3.1 Simulation of switching paradigms

The purpose of the project is to design and implement network level simulations of different switching paradigms, namely circuit switching, datagram packet switching and virtual circuit packet switching. The topology to be simulated should be a star topology, with at least 4 edge nodes connected to a central core node. Each edge node should be connected to at least one traffic source and one destination. Sources should generate data directed to destinations such that traffic exists between any pair of edge nodes, in both directions. The core node uses a finite FCFS queue for each output channel in case of packet switching. The queue size is a variable system parameter.

A policy for connection admission control in circuit and virtual circuit switching should be specified and used. A suitable packet size (either fixed or variable) in virtual circuit and datagram packet switching

should be defined. Simulations should be run at critical load levels and considering three different kinds of ON/OFF data sources, with increasing burstiness: a source with fixed ON/OFF periods, a source with exponential ON/OFF periods, a self similar source implemented by aggregating multiple ON/OFF sub-sources (5 at least) with Pareto-distributed ON/OFF periods using shape parameter $1 < \alpha < 2$.

Network performance should be analyzed under critical load levels and increasing traffic burstiness. Performance parameters to be measured are: average connection request blocking rate in circuit and virtual circuit switching, average packet delay and loss at the core node in virtual circuit and datagram switching, average channel utilization in each case. Simulation results should be statistically valid. Simulations may be performed using any existing simulation environment or by building an ad-hoc program (graphical interface is not required).

3.2 Simulation of data-link layer flow and error control

The purpose of the project is to design and implement simulations of different data-link layer flow and error control schemes, namely stop-and-wait ARQ and sliding window ARQ. For the sliding window case, both the go-back-N and selective repeat schemes should be provided. The source should be modeled as to always have data waiting to be transmitted.

The sender should set a timeout for each transmitted frame, starting immediately after frame transmission is completed by the sender. In the stop-and-wait case, when the timeout expires before the ACK has been received, the frame is retransmitted. In the sliding window case, when the timeout expires and no ACKs, NACKs, SREJs have been received, the sender transmits a POLL control frame.

The receiver should be configured to send an acknowledgment *every two frames correctly received* during regular operations. A NACK or SREJ should be sent for each frame received out of sequence, while an ACK should be sent as soon as a retransmitted frame is correctly received. In addition, a ACK, NACK or SREJ should be re-sent immediately after a POLL has been received.

After a frame has been retransmitted, the sender should ignore duplicate NACKs or SREJs related to the same frame if they are received before the retransmitted frame actually reaches the receiver. This is to avoid unnecessary duplicate retransmissions. A technique to handle this situation should be implemented.

Random error events should be generated in order to test the ARQ schemes. Performance should be analyzed, both without and with errors, by measuring the average throughput obtained through simulations. The error-free case should be compared with an approximate formula whenever possible. Simulation results should be statistically valid. Simulations may be performed using any existing simulation environment or by building an ad-hoc program (graphical interface is not required).

3.3 Implementation of a simple FTP protocol over UDP

The purpose of the project is to design and implement a reliable Simple File Transfer Protocol (SFTP) over UDP using datagram socket programming. The protocol should be a uni-directional transaction-oriented protocol implementing a sliding window flow and error control scheme. The stop-and-wait scheme should also be implemented by simply setting the window size to 1. The transaction should be driven by the sender, while the receiver should simply acknowledge each packet it gets.

Each transaction should begin when the sender transmits an initiation control packet, used to synchronize the sender and the receiver. This is followed by a series of data packets containing data from the file to be transferred. An ending control packet should be used to signal the end of the file transmission. Positive and negative acknowledgment control packets as well as timeout counters should be used, depending on the ARQ scheme adopted. The protocol should be designed so that deadlocks are avoided.

The software design should include the possibility to force errors in both data and control packets, in order to test recovery procedures. To simulate damaged transmission, random errors could be inserted in a given packet before it is transmitted. The software may be implemented using any language supporting socket programming (e.g. C, C++, Java).

3.4 Simulation of a traffic regulator

The purpose of the project is to design and implement simulations of different traffic policing/shaping mechanisms, namely leaky bucket and token bucket. The behavior of a single traffic regulator should be simulated. This regulator should be able to implement both a leaky bucket and a token bucket algorithm, depending on the user's choice. The user should also be allowed to choose whether traffic policing or shaping must be performed. All parameters involved should be variable and set by the user.

Simulations should be run considering three different kinds of data source, with increasing burstiness: a constant bit rate source, an ON/OFF source with exponential ON/OFF periods, a self similar source implemented by aggregating multiple ON/OFF sub-sources (5 at least) with Pareto-distributed ON/OFF periods using shape parameter $1 < \alpha < 2$.

The impact of the regulator on the traffic generated at the source should be analyzed using both schemes and varying the related parameters (i.e. allowed packet rate, bucket size and buffer size). Performance parameters to be measured and compared at different loads and burstiness levels are the policed/shaped packet rate as a function of the time and the average packet delay and loss. Simulation results should be statistically valid. An ad-hoc simulation software must be programmed (graphical interface is not required).

3.5 Implementation of a traffic regulator for UDP traffic

The purpose of the project is to design and implement an UDP traffic regulator using datagram socket programming. The regulator should be able to implement both a leaky bucket and a token bucket algorithm, depending on the user's choice. The user should also be allowed to choose whether traffic policing or shaping must be performed. All parameters involved should be variable and set by the user. The regulator should police/shape UDP packets with a specific destination port incoming on an input interface and forward them to an output interface.

The impact of the regulator on the relevant UDP traffic should be analyzed using both schemes and varying the related parameters (i.e. allowed packet rate, bucket size and buffer size). Performance parameters to be measured and compared at different loads and burstiness levels are the policed/shaped packet rate as a function of the time and the average packet delay and loss. Measurements should be statistically valid. The software may be implemented using any language supporting socket programming (e.g. C, C++, Java).

3.6 Simulation of a packet scheduler

The purpose of the project is to design and implement simulations of different packet scheduling policies, namely FIFO, Priority Queuing, Fair Queuing, Deficit Round Robin, Generalized Processor Sharing and Weighted Fair Queuing. The behavior of a single scheduler should be simulated. This scheduler should be able to implement all the policies listed above, depending on the user's choice. All parameters involved should be variable and set by the user.

Simulations should be run considering flows with Poisson arrivals and both cases of fixed and exponentially distributed packet sizes should be tested. Also the two cases of flows offering equal and different loads should be considered. The scheduler's behavior should be analyzed using all the implemented policies under different load conditions. In particular, the behavior of packet-based scheduling schemes should be compared to the ideal Generalized Processor Sharing policy. Performance parameters to be measured and compared are the service rate as a function of the time and the average service rate and packet delay, for each flow. Simulation results should be statistically valid. An ad-hoc simulation software must be programmed (graphical interface is not required).

3.7 Implementation of a packet scheduler for UDP traffic

The purpose of the project is to design and implement an UDP packet scheduler using datagram socket programming. The scheduler should be able to implement FIFO, Priority Queuing, Fair Queuing, Deficit Round Robin and Weighted Fair Queuing, depending on the user's choice. All parameters involved should be variable and set by the user. The scheduler should work over UDP flows identified by specific destination ports incoming on an input interface and forward them to an output interface.

The behavior of the scheduler on the relevant UDP traffic flows should be analyzed using all the implemented policies under different load conditions. Performance parameters to be measured and compared are the service rate as a function of the time and the average service rate and packet delay, for each flow. Measurements should be statistically valid. The software may be implemented using any language supporting socket programming (e.g. C, C++, Java).

3.8 Experiments with Linux traffic control

The purpose of the project is to explore and experiment with the Linux kernel traffic control subsystem. Linux systems provide a large number of networking tools, ranging from IP packet management and filtering to several application layer servers. One of the most powerful functionality of recent Linux kernels is the traffic control subsystem, that is able to implement different QoS-oriented techniques over TCP/IP traffic. In particular, operations at the packet level include queuing, policing, classifying, scheduling, shaping and dropping. Further details are available at the following URLs:

M. A. Brown, *Traffic Control HOWTO*, <http://linux-ip.net/articles/Traffic-Control-HOWTO/>

B. Hubert, *Linux Advanced Routing and Traffic Control HOWTO*, <http://lartc.org/howto/>

The different options available for traffic control under Linux should be explored and analyzed in detail. A set of targeted experiments should be designed and carried on to test Linux traffic control functions under different load conditions. Performance parameters to be measured are flow service rate, packet delay, packet loss rate. Measurements should be statistically valid.