Interconnection Networks Assignment

SLAT hours and load: According to the Faculty Handbook, this sub-module has a load of 50 SLAT hours. Of these, 10 hours cover attending the 10 lectures. Consequently (as this module is coursework only), the remaining 40 hours cover the time taken to understand and digest the lecture material and to complete this assignment; this means devoting, on average, 4 hours per week to this sub-module during the first term (outside of the lectures).

Helping your time management: Questions are presented in roughly lecture-order and are released week by week in 5 batches so that you can work on the coursework throughout the whole of the term. However, when you answer any question, you should apply your *full course knowledge* (so, you might return to an early answer later in the term after you have learned more). Don't forget to revisit the videos of lectures as sometimes I might make a helpful comment verbally that does not appear in writing.

Difficulty: The batches of questions released vary in terms of difficulty and the time required to answer them. I have tried to help by detailing the difficulty of each question-part with α, β or γ where α denotes the hardest parts and γ the easiest (though this is just my perception). Where I write, e.g., α/β, I mean that at least half of the marks awarded are at α level and at most half at β level. Also, you are intended to spend roughly the same amount of time on each of the 5 batches of questions. Most questions require written answers but some are answered by Python codes that you need to hand in. Full details are given below. By the way, the total number of marks available is 350 with 40% at α-level, 52% at β-level and 8% at γ-level.

Format: The main document containing your written answers should be **in pdf form** and called `Int_Nets_text_abcd12.pdf` where 'abcd12' is your personal ID (I don't care whether the document is typeset or simply photos of legible handwritten pages but **it needs to be in pdf form**). You should also submit your implementations of routing algorithms, all defined as functions (in the form and named as explained below), in a single Python program entitled `Int_Nets_routing_abcd12.py` so that the functions are executable in Python 3.8. Moreover, there should be no non-standard imported modules. Both the answers to the questions and your Python code should be submitted in a zipped folder named `abcd12`.

If you do not follow these instructions then you run the risk of losing marks!

Questions *roughly* covered by Lectures 1 and 2

Question 1. Explain the overheads and difficulties relating to fault-tolerant routing in the two contexts of: pre-building routes and storing them in routing tables at the nodes; or building routes on-the-fly.

[10 marks at β/γ]

Question 2. The 4-dimensional hypercube Q_4 is detailed on slide 15 of Lecture 1. It takes 4 bits to store the name of a node.

(a) If there is a table at each node $u \in \{0,1\}^4$ with an entry for each destination and with the entry being a full path from u to the destination, how many bits are required to store all of the entries in the table?

[3 marks at β]

[You may assume that for each entry, the source need not be stored but the rest of the path, including the destination, needs to be stored.]

(b) If there is a table at each node $u \in \{0,1\}^4$ with an entry for each destination and with the entry being only the next node on a path from u to the destination, how many bits does it take to store all of the entries in the table?

[2 marks at γ]

(c) Develop a coding scheme so as to reduce the total number of bits required to store the table in (b) as much as you can and state how many storage bits you require.

[10 marks at α/β]

Question 3. (a) Any graph can always be described by its adjacency matrix. However, in the context of interconnection networks where the number of nodes might be a million, a concise algebraic description is necessary. Give concise algebraic descriptions of the two interconnection network topologies that are natural generalizations of those in Fig. 1 to n nodes where n is divisible by 2 but not by 4.

[6 marks at β/γ]

[Full credit will be given only for precise *mathematically-defined* descriptions of the edge sets. Note that for the generalization of the Petersen graph, there are edges joining consecutive nodes in $1, 5, 9, 13, \ldots$]

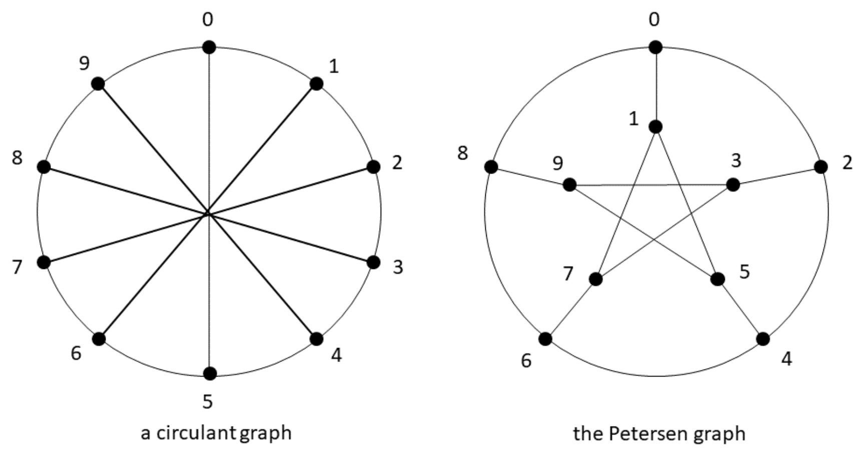

Figure 1: A circulant graph and the Petersen graph.

(*b*) Suppose that in the two interconnection network topologies in Fig. 1, every channel has bandwidth *b* bit/sec. Suppose that node 0 in each topology needs to send a message of *b* bits to every other node where these messages are all distinct; this is called a *single-node scatter*. For each interconnection network, explain whether this can be accomplished in 3 seconds.

[14 marks at β]

[You may assume that any node can simultaneously send (resp. receive) messages to (resp. from) adjacent nodes (with message sending and receiving simultaneous too). If you think that the answer is 'no' then you need to *prove* this; and if you think that the answer is 'yes' then you need to describe a *full* broadcast schedule. In essence, you can think of time as synchronous with one message being able to be sent along some channel in one second.]

(*c*) Suppose that in case (*b*) all messages are identical; that is, node 0 wishes to send the same message of *b* bits to every other node. This is called a *single-node broadcast*. For each interconnection network in Fig. 1, explain whether this single-node broadcast can be accomplished in 2 seconds.

[8 marks at β]

(*d*) Suppose that a *total exchange* is required; that is, every node needs to send a message to every other node and all messages are different. For each interconnection network in Fig. 1, devise a routing whereby there is a path from every node to every other node and calculate the load on every channel; that is, for each channel, the number of paths using the channel.

[14 marks at α/β]

[Your aim should be to reduce the load on the maximally loaded channel as much as you can and also to balance loads if possible.]