Interconnected Networks Assignment

Networks and their Structure IV,
Michaelmas Term
Ben Napier

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.

   Solution: When prebuilding routes, we can only deal with faults at the time of building routes. After this point, if a node fails it may still be included in a route as when the routes were built it was assumed to be working. This makes prebuilding routes a typically ineffective method for fault-tolerant routing, unless we frequently recalculate the routes.

   When calculating routes on-the-fly, we are able to work around faults and route regardless. However, this requires a solid description of the network; that is, each node needs to have an understanding of its neighbours and whether they are faulty. Additionally, this may add significantly to the computation time when sending a message.

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?

   [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.]

   Solution: We assume dimension-ordered routing. We have

   $$b_{\text{total}} = \sum_{n \in N \setminus \{u\}} D(u,n) \cdot b$$

   where $D(u,n)$ denotes the distance between u and n, and b is the number of bits required to store a node. We have 4 vertices of distance 1 away, 6 of distance 2 away, and 4 of distance 3 away, and 1 of distance 4 away. Thus

   $$b_{\text{total}} = (4)(1b) + (6)(2b) + (4)(3b) + (1)(4b) = 32b.$$

   As a node is an element of $\{0,1\}^4$, then $b = 4$. Thus

   $$b_{\text{total}} = 128\,\text{bit}.$$

   (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?

   Solution: Following a similar reasoning to part (a), we get the following.

   $$b_{\text{total}} = (2^4 - 1)(b) = 60\,\text{bit}$$

   (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.

Solution: For each node, we store a 2 digit binary number $b \in \{0,1\}^2$ which corresponds to the bit to flip to get to the next node in the path. For example, if $b = (0,0)$ we flip the first bit to get to the next neighbour. If $b = (1,1)$, we flip the fourth bit to get to the next neighbour. This requires $(2^4 - 1) \cdot 2 = 30$ bits per node.

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.

Solution: Circulent graph:

$$
\begin{aligned}
V &= \{v_0, \dots, v_{n-1}\} \\
E &= \{(v_{i \bmod n}, v_{i+1 \bmod n}) : i \in \{0, \dots, n-1\}\} \cup && (1) \\
&\quad \{(v_{i \bmod n}, v_{i+\frac{n}{2} \bmod n}) : i \in \{0, \dots, n-1\}\}. && (2)
\end{aligned}
$$

Here the edges defined in (1) are the outside cycle and (2) are the edges between antipodal points. Peterson graph:

$$
\begin{aligned}
V &= \{v_0, \dots, v_{n-1}\} \\
E &= \{(v_{2i \bmod n}, v_{2i+2 \bmod n}) : i \in \{0, \dots, n/2-1\}\} \cup && (3) \\
&\quad \{(v_{2i+1 \bmod n}, v_{2i+5 \bmod n}) : i \in \{0, \dots, n/2-1\}\} \cup && (4) \\
&\quad \{(v_{2i \bmod n}, v_{2i+1 \bmod n}) : i \in \{0, \dots, n/2-1\}\} && (5)
\end{aligned}
$$

(3) defines the cycle on the outside of the graph, (4) defines the graph contained within the cycle, and (5) corresponds to the connections between the cycle and the center.

(b) Suppose that in the two interconnection network topologies in Fig. 1, every channel has bandwidth b bit s^{-1}. 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.

Solution: This is possible in both graphs, broadcast schedule below.

(i) Let m_i be the message sent by 0 to node $i \in \{1, \dots, 9\}$. The packets follow the following routes:

Message	Route
m_1	$0 \to 1$
m_2	$0 \to 1 \to 2$
m_3	$0 \to 1 \to 2 \to 3$
m_4	$0 \to 5 \to 4$
m_5	$0 \to 5$
m_6	$0 \to 5 \to 6$
m_7	$0 \to 9 \to 8 \to 7$
m_8	$0 \to 9 \to 8$
m_9	$0 \to 9$

Then we have the following schedule.

(a) At time step 1, we send:

i. m_3 from 0 to 1,

ii. m_6 from 0 to 5,

iii. m_7 from 0 to 9.

(b) At time step 2, we send

i. m_2 from 0 to 1,

ii. m_3 from 1 to 2,

iii. m_4 from 0 to 5,

iv. m_6 from 5 to 6 (arrived),

v. m_7 from 9 to 8,

vi. m_8 from 0 to 9.

(c) At time step 3, we send

i. m_1 from 0 to 1 (arrived),

ii. m_2 from 1 to 2 (arrived),

iii. m_3 from 2 to 3 (arrived,

iv. m_4 from 5 to 4 (arrived,

v. m_5 from 0 to 5 (arrived),

vi. m_7 from 8 to 7 (arrived,

vii. m_8 from 9 to 8 (arrived),

viii. m_9 from 0 to 9 (arrived).

(ii) Let m_i be the message sent by 0 to node $i \in \{1, \ldots, 9\}$. We construct the following routes.

Message	Route
m_1	$0 \to 1$
m_2	$0 \to 2$
m_3	$0 \to 2 \to 3$
m_4	$0 \to 2 \to 4$
m_5	$0 \to 1 \to 5$
m_6	$0 \to 8 \to 6$
m_7	$0 \to 1 \to 7$
m_8	$0 \to 8$
m_9	$0 \to 8 \to 9$

We have the following schedule.

(a) At time step 1, we send:

i. m_4 from 0 to 2,

ii. m_6 from 0 to 8,

iii. m_7 from 0 to 1.

(b) At time step 2, we send:

i. m_3 from 0 to 2,

ii. m_4 from 2 to 4 (arrived),

iii. m_5 from 0 to 1,

iv. m_6 from 8 to 6 (arrived),

v. m_7 from 1 to 7 (arrived),

vi. m_9 from 0 to 8.

(c) At time step 3, we send:

i. m_1 from 0 to 1 (arrived),

ii. m_2 from 0 to 2 (arrived),

iii. m_3 from 2 to 3 (arrived),

iv. m_5 from 1 to 5 (arrived),

v. m_8 from 0 to 8 (arrived).

(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.

Solution: The diameter of the circulent graph is 3 (path $0 \to 5 \to 4 \to 3$ is minimal), thus a single-node broadcast is not possible. However, it is possible for the Peterson graph. Take the routes mentioned in (b).

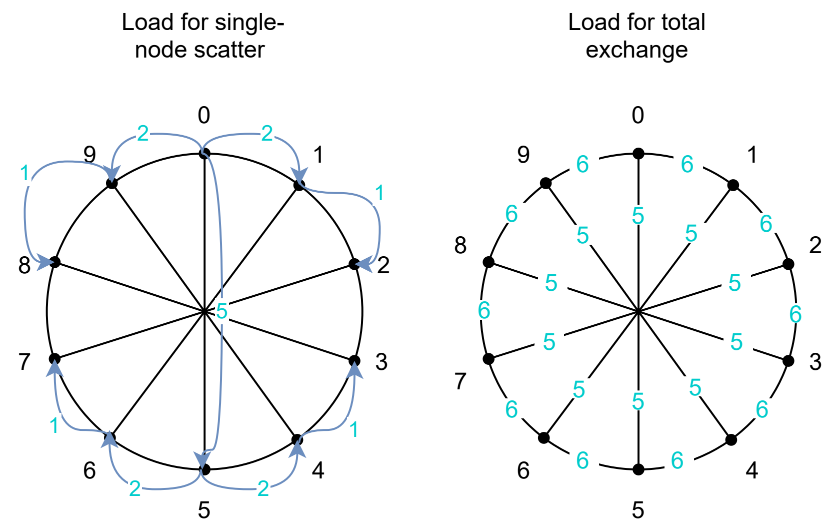
Message	Route
m_1	$0 \to 1$
m_2	$0 \to 2$
m_3	$0 \to 2 \to 3$
m_4	$0 \to 2 \to 4$
m_5	$0 \to 1 \to 5$
m_6	$0 \to 8 \to 6$
m_7	$0 \to 1 \to 7$
m_8	$0 \to 8$
m_9	$0 \to 8 \to 9$

All routes have length of at most 2, so we can just flood the packet at each node and ignore the incoming packet if you have already received it.

(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.

Solution:

(i) Circulent graph: consider the following image.

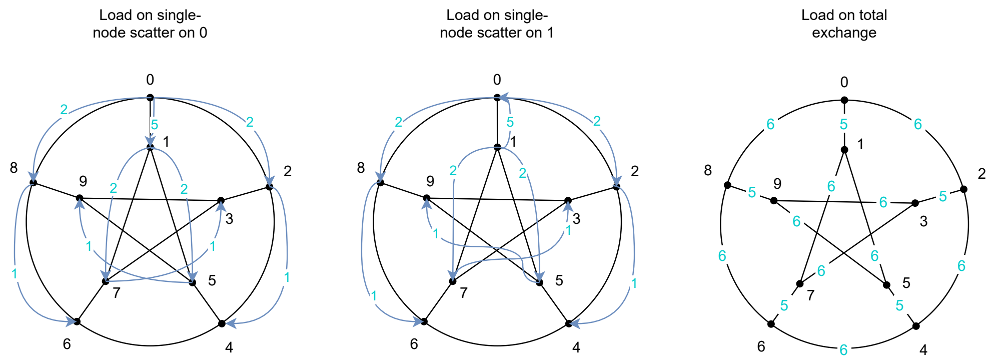


This depicts the following routes from node 0 to every other node.

Destination	Route
1	$0 \to 1$
2	$0 \to 1 \to 2$
3	$0 \to 5 \to 4 \to 3$
4	$0 \to 5 \to 4$
5	$0 \to 5$
6	$0 \to 5 \to 6$
7	$0 \to 5 \to 6 \to 7$
8	$0 \to 9 \to 8$
9	$0 \to 9$

As the circulent is node-symmetric by rotational symmetry of order 10, we can just rotate our routes to every node and get the loads as displayed in the image above.

(ii) Peterson graph: consider the following image.


Load on single-node scatter on 0
Load on single-node scatter on 1
Load on total exchange

Now, the Peterson graph is not node-symmetric; however, it does has rotational symmetry of order 5. Through rotation of $2\pi/5$, we have 2 *connected components* that we can reach. Namely, the odd nodes and the even nodes. Thus, finding a route for 0 and 1 is enough to generate both components.

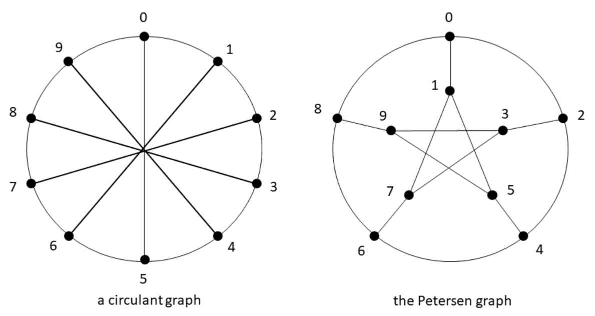


Figure 1: A circulant graph and the Petersen graph.

Source	Destination	Route
0	1	$0 \to 1$
0	2	$0 \to 2$
0	3	$0 \to 1 \to 7 \to 3$
0	4	$0 \to 2 \to 4$
0	5	$0 \to 1 \to 5$
0	6	$0 \to 8 \to 6$
0	7	$0 \to 1 \to 7$
0	8	$0 \to 8$
0	9	$0 \to 1 \to 5 \to 9$
1	0	$1 \to 0$
1	2	$1 \to 0 \to 2$
1	3	$1 \to 7 \to 3$
1	4	$1 \to 0 \to 2 \to 4$
1	5	$1 \to 5$
1	6	$1 \to 0 \to 8 \to 6$
1	7	$1 \to 7$
1	8	$1 \to 0 \to 8$
1	9	$1 \to 5 \to 9$

We can similarly rotate the routes for the odd numbered nodes and the routes for the even numbered nodes around the graph to get our final load, as displayed in the image above.

4. The *bubble-sort* interconnection network B_4 is pictured in Fig. 2. In parts (*b*)-(*c*), you may assume that B_4 is node-symmetric.

   (a) Give an algebraic definition of B_4.

Solution:

$$\begin{aligned} V &= \{(a_1, a_2, a_3, a_4) \in \{1,2,3,4\}^4 : a_i \neq a_j \ \forall i = j\}, \\ E &= \{((a_1, a_2, a_3, a_4), (a_2, a_1, a_3, a_4)) : (a_1, a_2, a_3, a_4) \in V\} \cup \\ &\quad \{((a_1, a_2, a_3, a_4), (a_1, a_3, a_2, a_4)) : (a_1, a_2, a_3, a_4) \in V \cup \\ &\quad \{((a_1, a_2, a_3, a_4), (a_1, a_2, a_4, a_3)) : (a_1, a_2, a_3, a_4) \in V\}. \end{aligned}$$

   (b) What is the diameter of B_4? You should justify your answer.

Solution: First, we complete a breadth-first search on this graph starting at 4321 to get the following spanning tree.

$$\begin{tikzcd}
 & & & 4321 \arrow[dl] \arrow[d] \arrow[dr] & & \\
 & & 4312 \arrow[dl] \arrow[d] & 3421 \arrow[d] & 4231 \arrow[d] \arrow[dr] & \\
 & 4132 \arrow[dl] \arrow[d] & 3412 \arrow[d] & 3241 \arrow[d] \arrow[dr] & 2431 \arrow[dr] & 4213 \\
4123 & 1432 \arrow[dl] \arrow[d] & 3142 \arrow[d] & 3214 \arrow[d] & 2341 & 2413 \arrow[dl] \\
1423 & 1342 & 3124 \arrow[d] & 2314 & 2143 \arrow[dl] \arrow[d] & \\
 & & 1324 & 2134 & 1243 \arrow[dl] & \\
 & & & 1234 & &
\end{tikzcd}$$

From this, we assert that a minimum path from $(4, 3, 2, 1)$ to any other node has length at most 6. We now claim this is true for any $y \in V$. Indeed, let $y \in N \setminus \{(4, 3, 2, 1)\}$ and $\sigma : V \to V$ such that $\sigma((4, 3, 2, 1)) = y$ (which exists as B_4 is node-symmetric). Now let $z \in V \setminus \{y\}$. Consider the route in the spanning from $(1, 2, 3, 4)$ to $\sigma^{-1}(z)$,

$$P = ((4, 3, 2, 1), v_1, \ldots, v_n, \sigma^{-1}(z))$$

where $n \leq 4$, and we take the image of this path:

$$\sigma(P) = P' = (y, \sigma(v_1), \ldots, \sigma(v_n), z).$$

As σ is an automorphism, this must be a valid path from y to z of length 6. Thus far, we have shown that there always exists a path of size 6 between any two nodes. We observe that (from our BFS search), there is no path from $(4, 3, 2, 1)$ to $(1, 2, 3, 4)$ that is shorter than 6. With this, we assert that the diameter of B_4 is 6.

(c) Is B_4 a moore graph? You should justify your answer.

Solution: $B_4 = (V, E)$ is a moore graph if and only if

$$|V| = 1 + d \sum_{i=0}^{k-1} (d-1)^i$$

where d is the degree of B_4 and k is the diameter. From part (b), we have that $k = 6$ and one can reason that $d = 3$. Furthermore, the number of nodes in B_4 is just the number of ways you can permute 4 elements, that is $|V| = 4! = 24$. We see

$$24 = |V| \neq 1 + d \sum_{i=0}^{k-1} (d-1)^i = 190.$$

Thus B_4 is not moore graph.

(d) Show as many node-disjoint paths as possible joining the node $(4, 3, 1, 2)$ to $(2, 3, 1, 4)$. What can you say about the numbers of node-disjoint paths joining the node $(3, 2, 1, 4)$ to $(4, 2, 1, 3)$ and the node $(1, 2, 3, 4)$ to $(4, 2, 3, 1)$?

Solution: We have the following node-disjoint paths between $(4,3,1,2)$ and $(2,3,1,4)$.

(i) $(4,3,2,1),(4,3,1,2),(4,1,3,2),(1,4,3,2),(1,3,4,2),(1,3,2,4),(1,2,3,4),(2,1,3,4),$ $(2,3,1,4)$;

(ii) $(4,3,2,1),(4,2,3,1),(2,3,4,1),(2,3,1,4)$; and

(iii) $(4,3,2,1),(3,4,2,1),(3,2,4,1),(3,2,1,4),(2,3,1,4)$.

We now introduce the automorphisms:

(i) $\sigma_1(a_1,a_2,a_3,a_4)=(a_4,a_3,a_2,a_1)$;

(ii) σ_2: reflection along the centered vertical axis; and

(iii) σ_3: anticlockwise rotation by 120°.

σ_2 and σ_3 are automorphisms as the geometric realisation of the graph in the question clearly has rotational symmetry of order 3 with respect to the center point of the image, and it also has reflection symmetry along the center vertical axis. Let $\boldsymbol{a}=(a_1,a_2,a_3,a_4),\boldsymbol{b}=(b_1,b_2,b_3,b_4)\in V$ be distinct nodes. Suppose $\boldsymbol{a}=\boldsymbol{b}$. Then

$$(a_4,a_3,a_2,a_1)=(b_4,b_3,b_2,b_1).$$

That is, $\boldsymbol{a}=\boldsymbol{b}$; a contradiction. Thus σ_1 is a permutation. Now we show that σ_1 is also channel perserving. Let $\boldsymbol{a}=(a_1,a_2,a_3,a_4)\in V$ and consider its neighbours

$$(a_2,a_1,a_3,a_4),(a_1,a_3,a_2,a_4),(a_1,a_2,a_4,a_3).$$

We consider all outgoing edges of $\boldsymbol{a}$ and verify that they are perserved.

$$\begin{aligned}(\sigma_1(\boldsymbol{a}),\sigma_1(a_2,a_1,a_3,a_4))&=((a_4,a_3,a_2,a_1),(a_4,a_3,a_1,a_2))\in E,\\(\sigma_1(\boldsymbol{a}),\sigma_1(a_1,a_3,a_2,a_4))&=((a_4,a_3,a_2,a_1),(a_4,a_2,a_3,a_1))\in E,\\(\sigma_1(\boldsymbol{a}),\sigma_1(a_1,a_2,a_4,a_3))&=((a_4,a_3,a_2,a_1),(a_3,a_4,a_2,a_1))\in E.\end{aligned}$$

These edges are infact contained within E as: both vertices within the edges are in V (as none of the a_i are equal and all lie within $\{1,2,3,4\}$) and each vertex can be obtained from the other in its pair by flipping an adjacent pair. Thus σ_1 indeed defines an automorphism. We now consider the number of disjoint graphs between the other pairs of nodes.

(i) $(3,2,1,4)$ to $(4,2,1,3)$: we note that $(\sigma_3^{-1}\circ\sigma_1)(4,3,1,2)=(\sigma_3^{-1})(2,1,3,4)=(3,2,1,4)$ and $(\sigma_3^{-1}\circ\sigma_1)(2,3,1,4)=(\sigma_3^{-1})(4,1,3,2)=(4,2,1,3)$. Thus the image of the three node-disjoint paths from $(4,3,1,2)$ to $(2,3,1,4)$ must also be node-disjoint paths from $(3,2,1,4)$ to $(4,2,1,3)$, as $\sigma_3^{-1}\circ\sigma_1$ is also an automorphism.

(ii) $(1,2,3,4)$ to $(4,2,3,1)$: similarly, we observe that $(\sigma_3^{-1}\circ\sigma_2\circ\sigma_1)(4,3,1,2)=(1,2,3,4)$ and $(\sigma_3^{-1}\circ\sigma_2\circ\sigma_1)(2,3,1,4)=(4,2,3,1)$. Thus, by similar argument to before, there must be three node-disjoint paths connecting $(1,2,3,4)$ and $(4,2,3,1)$, as $\sigma_3^{-1}\circ\sigma_2\circ\sigma_1$ is an automorphism.

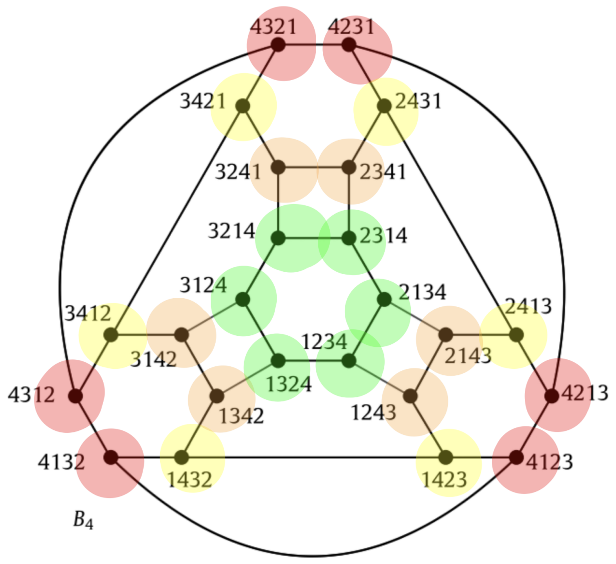
(e) Prove that B_4 is node-symmetric.

Solution: We first introduce the following geometric automorphisms:

(i) anticlockwise rotation by 120°; and

(ii) reflection along the vertical.

These are both automorphisms as the geometric realisation of B_4 presented in the question has rotational symmetry of order 3 with respect to the center point of the image, and reflection symmetry along the centered vertical axis. The following image illustrates the classes for which we can find automorphisms between any two nodes in the class (the highlight colour represents distinct classes) using the two automorphisms introduced so far.



We now consider the algebraically-defined automorphism (we will prove that this is an automorphism):

$$\sigma : V \to V$$
$$\sigma(a_1, a_2, a_3, a_4) = ((a_1 \bmod 4) + 1, (a_2 \bmod 4) + 1, (a_3 \bmod 4) + 1, (a_4 \bmod 4) + 1).$$

σ is an automorphism if it is a permutation and is channel-preserving. We let $\boldsymbol{a} = (a_1, a_2, a_3, a_4), \boldsymbol{b} = (b_1, b_2, b_3, b_4) \in V$ be distinct nodes of B_4. Suppose that $\sigma(\boldsymbol{a}) = \sigma(\boldsymbol{b})$. But then

$$(a_2, a_1, a_3, a_4) = (b_2, b_1, b_3, b_4);$$

that is, $\boldsymbol{a} = \boldsymbol{b}$; a contradiction as $\boldsymbol{a}$ and $\boldsymbol{b}$ were assumed to be distinct. Thus σ indeed defines a permutation. Now we consider $\boldsymbol{a} = (a_1, a_2, a_3, \dots, a_4) \in V$ and its neighbours:

$$(a_2, a_1, a_3, a_4), (a_1, a_3, a_2, a_4), (a_1, a_2, a_4, a_3) \in E.$$

We let $\sigma(\boldsymbol{a}) = (b_1, b_2, b_3, b_4) \in V$ and consider the image of the edges from $\boldsymbol{a}$.

$$(\sigma(\boldsymbol{a}), \sigma(a_2, a_1, a_3, a_4)) = (\boldsymbol{b}, (b_2, b_1, b_3, b_4)) \in E,$$
$$(\sigma(\boldsymbol{a}), \sigma(a_1, a_3, a_2, a_4)) = (\boldsymbol{b}, (b_1, b_3, b_2, b_4)) \in E,$$
$$(\sigma(\boldsymbol{a}), \sigma(a_1, a_2, a_4, a_3)) = (\boldsymbol{b}, (b_1, b_2, b_4, b_3)) \in E.$$

These edges are contained within E as $\sigma : V \to V$ is bijective and each endpoint in the edges can be obtained by flipping adjacent numbers in the other endpoint in the edge. Thus for every vertex its edges are all preserved; thus σ is channel preserving and thus an automorphism.

We now claim that appropriate compositions of the three introduced automorphisms (and their inverses) are adequate in showing that B_4 is node-symmetric. First, we note that the composition of two automorphisms also defines an automorphism and that the inverse of an automorphism is also an automorphism. From the earlier image, we have the following *classes* in which we can construct an automorphism between any two nodes of the same

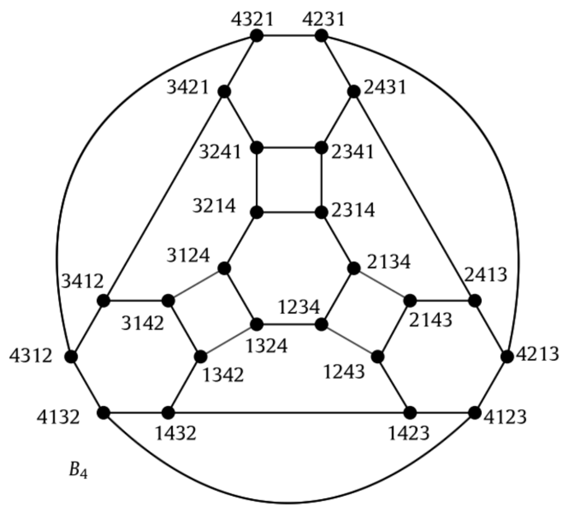


Figure 2: The bubble-sort interconnection network B_4.

class:

$$
\begin{aligned}
C_1 &= \{(4,3,2,1),(4,3,1,2),(4,1,3,2),(4,1,2,3),(4,2,1,3),(4,2,3,1)\},\\
C_2 &= \{(3,4,2,1),(3,4,1,2),(1,4,3,2),(1,4,2,3),(2,4,1,3),(2,4,3,1)\},\\
C_3 &= \{(3,2,4,1),(3,1,4,2),(1,3,4,2),(1,2,4,3),(2,1,4,3),(2,3,4,1)\},\\
C_4 &= \{(3,2,1,4),(3,1,2,4),(1,3,2,4),(1,2,3,4),(2,1,3,4),(2,3,1,4)\}.
\end{aligned}
$$

Using σ, we can connect these classes together. Indeed, consider the following:

$$\sigma(4,3,2,1) = (1,4,3,2), \tag{1}$$
$$\sigma(1,4,3,2) = (2,1,4,3), \tag{2}$$
$$\sigma(4,2,1,3) = (1,3,2,4). \tag{3}$$

We see that (1) connects C_1 to C_2, (2) connects C_2 and C_3, and (3) connects C_1 and C_4. Thus, we have shown that using composition of the two geometric automorphisms and σ (as well as their inverses), we can construct an automorphism between any two nodes of the graph.

5. Consider a supercomputer $\mathcal{K}$ whose interconnection network is a 4-ary 3-cube Q_3^4 whose node set is $\{(u_1,u_2,u_3) : 0 \le u_1,u_2,u_3 \le 3\}$ and whose edge set is

$$
\begin{aligned}
\{((u_1,u_2,u_3),(v_1,v_2,v_3)) : 0 \le u_1,u_2,u_3,v_1,v_2,v_3 \le 3,&\\
v_1 = u_1 \pm 1, u_2 = v_2, u_3 = v_3 \text{ or}&\\
v_1 = u_1, u_2 = v_2 \pm 1, u_3 = v_3 \text{ or}&\\
v_1 = u_1, u_2 = v_2, u_3 = v_3 \pm 1 \text{ with addition modulo } 4\}.&
\end{aligned}
$$

Suppose, for simplicity, that $\mathcal{K}$ operates synchronously in that there is a global clock and at each tick of the click: some computation is undertaken by every node (the computation phase); at most one new message is generated for each of the neighbours of a node (the generation phase); and at most one message is sent to each neighbour of a node (the message phase). Of course, a node might receive messages, in the message phase, from neighbours that are intended for other nodes and that need to be passed on in an upcoming message phase. Every message includes in its header the full route of that message and so any node can immediately detect the neighbour to which the message is to be sent. For every output channel, there is a buffer that holds the messages that need to be sent down the channel in an upcoming message phase to the corresponding neighbour and any generated or received message is automatically placed in the correct buffer (at the end of the corresponding generation or message phase, respectively). All messages have size one unit and buffer sizes are measured in units.

There is a Hamiltonian cycle H in Q_3^4 defined as follows:

$$\begin{gathered}(0,0,0),(1,0,0),(2,0,0),(3,0,0),(3,1,0),(2,1,0),(1,1,0),(0,1,0),\\(0,2,0),(1,2,0),(2,2,0),(3,2,0),(3,3,0),(2,3,0),(1,3,0),(0,3,0),\\(0,3,1),(1,3,1),(2,3,1),(3,3,1),(3,2,1),(2,2,1),(1,2,1),(0,2,1),\\(0,1,1),(1,1,1),(2,1,1),(3,1,1),(3,0,1),(2,0,1),(1,0,1),(0,0,1),\\(0,0,2),(1,0,2),(2,0,2),(3,0,2),(3,1,2),(2,1,2),(1,1,2),(0,1,2),\\(0,2,2),(1,2,2),(2,2,2),(3,2,2),(3,3,2),(2,3,2),(1,3,2),(0,3,2),\\(0,3,3),(1,3,3),(2,3,3),(3,3,3),(3,2,3),(2,2,3),(1,2,3),(0,2,3),\\(0,1,3),(1,1,3),(2,1,3),(3,1,3),(3,0,3),(2,0,3),(1,0,3),(0,0,3),(0,0,0).\end{gathered}$$

(a) Explain how we can use H to undertake a single-node scatter from $(0,0,0)$ in $\mathcal{K}$ (as in slide 8 of Lecture 4). Be sure to detail the number of clock-ticks required and the maximal size of any channel buffer.

Solution: Let n_i denote the ith node in the Hamiltonian cycle H ($n_0 = (0,0,0)$). On the first clock cycle, we generate the message intended for $n_{63} = (0,0,3)$. Then on the kth clock cycle ($k \in \{2,3,\dots,63\}$), we generate the message for node n_{64-k} and send the message for node n_{64-k+1} (which is generated in the $(k-1)$th clock cycle). Then on the 64th clock cycle, we send the message to $(1,0,0)$ (which was generated on the 63rd clock cycle). Thus, by the end of the 64th cycle all nodes have received their message. This requires a maximal buffer size of 1 and 64 clock cycles until all nodes have received their message.

(b) There is another supercomputer $\mathcal{Q}$ where the interconnection network is the 3-dimensional hypercube Q_3. We wish to simulate the single-node scatter of $\mathcal{K}$ in $\mathcal{Q}$. Explain how we might do this and comment on the overheads arising from your simulation.

Solution: We describe an embedding from $\mathcal{K}$ to $\mathcal{Q}$: the first 8 nodes of the Hamiltonian cycle H is mapped to $(0,0,0)$. The next 8 nodes are mapped to $(1,0,0)$ and so on. Then we proceed with the simulation by timesharing each of the virtual processors that are mapped to each node of $\mathcal{Q}$. If φ is the embedding we described above, if a message was to be sent between x and y in $\mathcal{K}$ it is instead sent between $\varphi(x)$ and $\varphi(y)$ (in the case that $\varphi(x) = \varphi(y)$, we place this message in the buffer). This embedding has a load of 8, congestion 8, and dilation 1 and will require a buffer of 8 at each node of $\mathcal{Q}$.

6. Suppose you have a supercomputer $\mathcal{C}$ whose interconnection network is the 3-dimensional hypercube Q_3. However, you have a fault-tolerant software application S that never terminates, is designed for a supercomputer $\mathcal{D}$ whose interconnection network is Q_4 and can tolerate one faulty processor.

(a) Explain how you might make use of an embedding to run S on $\mathcal{C}$ and describe any overheads accruing.

Solution: We consider $V(Q_4) = \{0,1\}^4$ and $V(Q_3) = \{0,1\}^3$. With our embedding, we still want our software to be fault tolerant. However, if we have multiple nodes (in the guest network) mapped to a single node (in the host network), then a failure of this node will cause the software to be unable to run, as we can only deal with a single faulty processor. To mitigate this, we can maximise the number of one-to-one mappings in our embedding.

By doing this, we may get an embedding φ like the following:

$$(a_1, a_2, a_3, 0) \mapsto (0, 0, 0),$$
$$(a_1, a_2, a_3, 1) \mapsto (a_1, a_2, a_3)$$

for all $a_1, a_2, a_3 \in \{0, 1\}$. Lets describe how the channels may get mapped, here we are trying to map channels to paths such as to minimise the congestion.

(i) Edges of the form $((a_1, a_2, a_3, 1), (b_1, b_2, b_3, 1))$ get mapped to $((a_1, a_2, a_3), (b_1, b_2, b_3))$.

(ii) Edges of the form $((a_1, a_2, a_3, 0), (b_1, b_2, b_3, 0))$ get mapped to the empty path (as they get contracted into $(0, 0, 0)$).

(iii) The edge $((0, 0, 0, 0), (0, 0, 0, 1))$ also gets mapped to the empty path as it is also contracted into $(0, 0, 0)$.

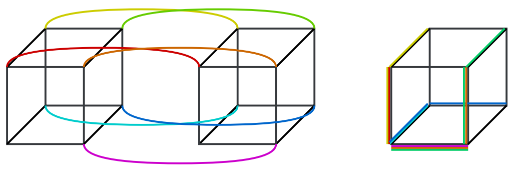
(iv) Finally, we need to pick a suitable mapping for edges of the form

$$((a_1, a_2, a_3, 0), (a_1, a_2, a_3, 1))$$

(excluding the case in (iii)). Observe that $\varphi(a_1, a_2, a_3, 0) = (0, 0, 0)$ and $\varphi(a_1, a_2, a_3, 1) = (a_1, a_2, a_3)$. So we are looking paths from $(0, 0, 0)$ to every other point in Q_3 that minimises congestion and dilation. We observe that, in Q_3, $(0, 0, 0)$ has 3 outgoing channels, thus as we have 7 paths to form, the congestion must be at least 3. Furthermore, the shortest path from $(0, 0, 0)$ to $(1, 1, 1)$ is 3, and so the dilation must be at least 3. We can indeed attain both of these minimums with the following assignments.

$$\begin{aligned}
((1,0,0,0),(1,0,0,1)) &\mapsto ((0,0,0),(1,0,0)),\\
((0,1,0,0),(0,1,0,1)) &\mapsto ((0,0,0),(0,1,0)),\\
((0,0,1,0),(0,0,1,1)) &\mapsto ((0,0,0),(0,0,1))\\
((1,1,0,0),(1,1,0,1)) &\mapsto ((0,0,0),(1,0,0),(1,1,0)),\\
((1,0,1,0),(1,0,1,1)) &\mapsto ((0,0,0),(0,0,1),(1,0,1)),\\
((0,1,1,0),(0,1,1,1)) &\mapsto ((0,0,0),(0,1,0),(0,1,1)),\\
((1,1,1,0),(1,1,1,1)) &\mapsto ((0,0,0),(1,0,0),(1,1,0),(1,1,1)).
\end{aligned}$$

I have displayed the graphically below.



It is clear to see that our load is 9, our congestion is 3, and our dilation is also 3.

To run S on $\mathcal{C}$, for each processor $p \in \{0, 1\}^3$ of $\mathcal{C}$, we simulate each processor in $\varphi^{-1}(\{p\})$; that is, each processor mapped to p. This can be done by time-sharing, and if a message is originally sent along the channel (x, y) in Q_4, it is instead sent along the described path from $\varphi(x)$ to $\varphi(y)$ which is described above. In this situation, the simulation requires significant buffer space as 9 processors have been mapped to a single processor. So any messages being sent between these 9 nodes will have to be stored in a buffer. The reason for the high load is to ensure that the program is still fault tolerant. One may argue that increasing the load on processor $(0, 0, 0)$ to 9 times its original load may cause it to be more likely to fail, but it may be the case that the expected chance of failure is still lower than in the situation where we may spread our mapping more evenly throughout Q_3.

(b) Suppose that from time to time you obtain the resources to buy an additional processor. Explain how you might incrementally scale $\mathcal{C}$ by incorporating these processors one at a time and so that there is never any need to pause the execution of S.

[You may assume that you have access to an unlimited supply of processor-to-processor cables.]

Solution: We can utilise the fault-tolerance of S to upgrade $\mathcal{C}$ without causing a pause in execution. There would be no need to expand $\mathcal{C}$ past the size of the original interconnection network, so we assume we have $k \in \{8, \dots, 15\}$ processors (think of this as the step number) and we wish to add a $(k+1)$th processor. Consider the following path (in Q_4):

$$P = ((0,0,0,0),(1,0,0,0),(1,1,0,0),(0,1,0,0),(0,1,1,0),(1,1,1,0),(1,0,1,0),(0,0,1,0))$$

and let P_i denote the ith entry in P ($P_1 = (0,0,0,0)$). To add this $(k+1)$th processor, we disable the (logical) processor P_{k-7} (which is currently being mapped to node $(0,0,0)$). As S is fault-tolerant, it can handle this processor being disabled. We then add our $(k+1)$th processor to our network, joining it to $(0,0,0)$ and also to every other processor we have added beyond Q_3 (i.e. the processors added in step $8, 9, \dots, k-1$). By doing this, we remove all previous mappings we had in our embedding for P_{k-7}. The only changes to software which would be needed is in the routing tables at $(0,0,0)$. All other nodes should route to nodes mapped to $(0,0,0)$ as normal, but as $(0,0,0)$ is either adjacent to one of the nodes that were originally mapped to it or is itself simulating that node then it can either place it in the internal buffer, or directly send it to the new node added for said node. To discuss the overheads here, we would indeed have a large number of processor-to-processor cables as we are effectively building a complete graph off of a single node in Q_3. Our load would decrease by 1 for every node we add, our congestion we remain constant, and our dilation would increase to 4 after the first processor addition, but stay constant with further additions.

7. Explain the reasoning behind the definition of the ideal throughput. Which sort of traffic scenarios is it best suited to?

Solution: First, we define ideal throughput. Let $G = (N, C)$ be a topology, where N is a set of nodes and C is a set of channels. To start we fix some particular routing $R = \{R_n\}_{n \in N}$, that is, for each $n \in N$, R_n denotes a set of routes starting at n (a route is a list of channels that defines a path). We further suppose all channels have bandwidth b bit s^{-1}. We define the *load* γ_c on a particular channel c to be the number of routes using channel c. Precisely,

$$\gamma_c = |\{r \in R_n : c \in r, n \in N\}|.$$

We define the *maximum channel load*, $\gamma_{\max}$ to be maximum load over all channels, that is

$$\gamma_{\max} = \max_{c \in C}\{\gamma_c\}.$$

A channel $c \in C$ such that $\gamma_c = \gamma_{\max}$ is said to be a *bottleneck channel.* We define the ideal throughput to be the number of bits that can be sent over a bottleneck channel in the scenario that every route that utilises that channel contests for it at the same time. Formally,

$$\theta_{\text{ideal}} = \frac{b}{\gamma_{\max}}.$$

Ideal throughput gives us the notion of the maximum size (in bits) of a message such that we can send it along a route in the worst case scenario without contest. It is optimal for traffic scenarios where $\gamma_{\max} - \overline{\gamma}$ is small, where $\overline{\gamma}$ denotes the mean channel load for C. To illustrate why, consider a traffic scenario on a large network where all channels have bandwidth 1024 bit s^{-1} with one channel c' such that $\gamma_{c'} = 16$ and for all $c \in C \setminus \{c'\}$ we have $\gamma_c = 1$. Then, $\gamma_{\max} = 16$, thus $\theta_{\text{ideal}} = \frac{1024}{16} = 64$. Thus, to ensure that we do not cause performance issues each message must have size 64 bit. As this is a large network, we see that this is causing lots of channels to have 960 bits of bandwidth unused, and so our ideal throughput is not giving us a good picture in this scenario.

8. Suppose that we have 260 processors, each with 60 pins and so that the signal frequency is 1 GHz; moreover, the router delay of any router is 15 ns. Our intention is to design an interconnection network that is a hypercube, a k-ary n-cube, or a cube-connected cycles so that: for the hypercube and the cube-connected cycles, we always use as many processors as possible but never less that 210 processors; and for the k-ary n-cube, for each possible n we always use as many processors as possible but never less than 210 processors (we always assume that $k \geq 3$ and $n \geq 2$). Additionally, our applications will be such that packets consist of 512 bits and the traffic pattern is random.

   (a) Which topologies are candidates for our interconnection network?

Solution: So for each of the topology types, we have that $p \in \{210, 211, \dots, 260\}$ where p is the number of processors used. For hypercube, we require the topology of maximum p, which is Q_8: $|V(Q_8)| = 2^8 = 256$. For the cube connected cycle, there is no n such that $p = |V(\text{CCC}_n)| = n2^n$ for the constraints above, thus there are no candidates. Finally, for the k-ary n-cube, we have the following candidate topologies (note $|V(Q_n^k)| = k^n$).

n	k	$\lvert V(Q_n^k)\rvert$
2	16	256
3	6	216
4	4	256
5	3	243

   (b) Analyse and compare each of the possible candidate designs with respect to the maximum possible ideal throughout.

Solution: Let d be the degree of any of the candidates above, then we observe that

$$b = \left\lfloor \frac{60}{d} \right\rfloor \times 10^9.$$

We let θ^1_{ideal} and θ^2_{ideal} denote our two upper bounds for the idea throughput, where

$$\theta^1_{\text{ideal}} = \frac{2bB_C}{|N|} = \frac{2B_C \left\lfloor \frac{60}{d} \right\rfloor \times 10^9}{|N|},$$
$$\theta^2_{\text{ideal}} = \frac{|C|b}{H_{\text{ave}}|N|} = \frac{|C| \left\lfloor \frac{60}{d} \right\rfloor \times 10^9}{H_{\text{ave}}|N|}.$$

Graph	$\lvert C\rvert$	$\lfloor 60/d \rfloor$	H_{ave}	$\lvert N\rvert$	B_C	θ^1_{ideal}	θ^2_{ideal}	$\min\{\theta^1_{\text{ideal}}, \theta^2_{\text{ideal}}\}$
Q_8	2048	3	4	256	256	6.0×10^9	6.0×10^9	6.0×10^9
Q_2^{16}	1024	7	8	256	64	3.5×10^9	3.5×10^9	3.5×10^9
Q_3^6	1296	5	9/2	216	144	6.7×10^9	6.7×10^9	6.7×10^9
Q_4^4	2048	3	4	256	256	6.0×10^9	6.0×10^9	6.0×10^9
Q_5^3	2430	3	10/3	243	324	8.0×10^9	9.0×10^9	9.0×10^9

So we see that, to maximise possible ideal throughput, Q_5^3 seems to be the best option. This also coincides with having the lowest hop count, suffering from router delay the least, but also having the most number of channels.

9. Let G be an interconnection network that is node-symmetric. Derive an upper bound on the ideal throughput for the tornado traffic pattern.

Solution: Let $G = (N, C)$ be a node-symmetric interconnection network with diameter d. We have derived the upper bound

$$\theta_{\text{ideal}} \le \frac{2bB_c}{|N|}$$

where b is the bandwidth and B_C is the bisection width; however, this assumes a random traffic pattern and as such is not suitable. Furthermore, we can make almost no comment on the bisection width of a graph given only that it is node-symmetric. We have also seen the upper bound

$$\theta_{\text{ideal}} \le \frac{|C|b}{H_{\text{ave}}|N|},$$

which is more suitable as it makes no assumption upon the random traffic model, and we can calculate $H_{\text{ave}}|N|$ with a bit more reasoning.

Lemma. For every $x \in N$, $\max_{y\in N} D_{\min}(x, y) = d$.

Proof. As d is the diameter of G, there is a path $(x_1, \dots, x_{d+1})$ that is minimal. We reason that any path with a larger length is not minimal. Thus, as we are consider the tornado traffic pattern, x_1 with be sending a packet some node y_1 such that $D_{\min}(x_1, y_1) = d$. Let $x \in N\setminus\{x_1\}$. As G is node-symmetric, there is an automorphism on G, $\varphi_x : \mathbb{N} \to \mathbb{N}$, such that $\varphi_x(x_1) = x$. We claim that $(x, \dots, \varphi_x(x_{d+1}))$ is a minimal path from x to $\varphi_x(x_{d+1})$. Indeed, assume otherwise: let $(x, y_1, \dots, y_m, \varphi_x(x_{d+1}))$ such that $m < d - 1$. Then, as φ_x^{-1} is also an automorphism, $(x_1, \varphi_x^{-1}(y_1), \dots, \varphi_x^{-1}(y_m), x_{d+1})$ is a path of length $l < d$ which contradicts the minimality of $(x_1, \dots, x_{d+1})$. There is no node further than distance d from x as our diameter is d. Thus, for every node the node furthest away is of distance d (which we constructed). $\square$

Now, we replace $H_{\text{ave}}|N|$ with the sum of all the hop counts to get the following upper bound:

$$\theta_{\text{ideal}} \le \frac{|C|b}{d|N|}.$$

10. When we constructed the cube-connected cycles CCC_n from the hypercube Q_n, we performed a local operation at every node of Q_n that amounted to 'replacing' the node of Q_n with a cycle of length n and joining corresponding cycle nodes. We could perform this construction at *any* node of *any* graph. Consider the complete graph K_n where there is a link joining every pair of nodes.

(a) Build the graph KKK_n by 'replacing' every node of K_n with a cycle of length $n - 1$ and give a concise algebraic description of KKK_n.

Solution: Let $KKK_n = (N, C)$.

We define the node set as

$$N = \mathbb{Z}/n \times \mathbb{Z}/(n-1).$$

Lets explore some intuition behind this definition. To build KKK_n, we are performing local operations on each of nodes in K_n, so we will define the node set K_n as $N_{K_n} = \mathbb{Z}/n$. Each point $\overline{a} \in N_{k_n}$ is replaced with the nodes $\{(\overline{a}, \overline{b}) : \overline{b} \in \mathbb{Z}/(n-1)\}$: our cycle. So for $(\overline{a}, \overline{b}) \in N$, $\overline{a}$ corresponds to the initial node that we replaced, and $\overline{b}$ gives us a *position* within the cycle.

Now for our channel set. There is a link from $(\overline{x}, \overline{u})$ to $(\overline{y}, \overline{v})$ if

(i) (the cycle edges) $\overline{x} = \overline{y}$, $\overline{u} = \overline{v \pm 1}$; or

(ii) (the hypercube edges) $\overline{y} = \overline{x + u + 1}$, $\overline{v} = \overline{-u - 1}$.

Now we justify this definition. Firstly, condition (i) accounts for all the cycle edges clearly. Next, condition (ii) connects distinct cycles together. Precisely, in a given cycle $\overline{x}$, the node with index $\overline{u}$ links to the cycle $\overline{u + 1}$ away, that is cycle $\overline{x + u + 1}$. Thus $(\overline{x}, \overline{u})$ would connect to cycle $\overline{x+u+1}$. But in cycle $\overline{x+u+1}$, cycle $\overline{x}$ is $\overline{(n-1)-(u+1)} = \overline{-u-1}$, thus $(\overline{x},\overline{u})$ and $(\overline{x+u+1},\overline{-u-1})$ must have a link between them.

(b) Give an upper bound on the diameter of KKK_n.

Solution: Let $(\overline{x},\overline{u}),(\overline{y},\overline{v}) \in N$. First, we note that the diameter of a cycle is $\lfloor n/2 \rfloor$. If $\overline{x} = \overline{y}$, then we can find a route of length at most $\lfloor n/2 \rfloor$. If $\overline{x} \neq \overline{y}$, then we construct the following route. First, we route along cycle channels from $(\overline{x},\overline{u})$ to $(\overline{x},\overline{y-x-1})$ (in at most $\lfloor n/2 \rfloor$ hops). Then we move along the hypercube channel from $(\overline{x},\overline{y-x-1})$ to $(\overline{y},\overline{x-y})$ (one hop). Finally, we route along cycle channels again from $(\overline{y},\overline{x-y})$ to $(\overline{y},\overline{v})$ (in at most $\lfloor n/2 \rfloor$ hops). In total, this route takes at most $2\lfloor n/2 \rfloor + 1$ hops and as it can be constructed for arbitrary nodes in N, is an upper bound of the diameter of KKK_n.

11. Consider Q_2^3. We can partition the channels as follows.

    (i) $pos_1 = \{((u,v),(u+1,v)) : 0 \leq u,v \leq 2\}$

    (ii) $pos_2 = \{((u,v),(u,v+1)) : 0 \leq u,v \leq 2\}$

    (iii) $neg_1 = \{((u,v),(u-1,v)) : 0 \leq u,v \leq 2\}$

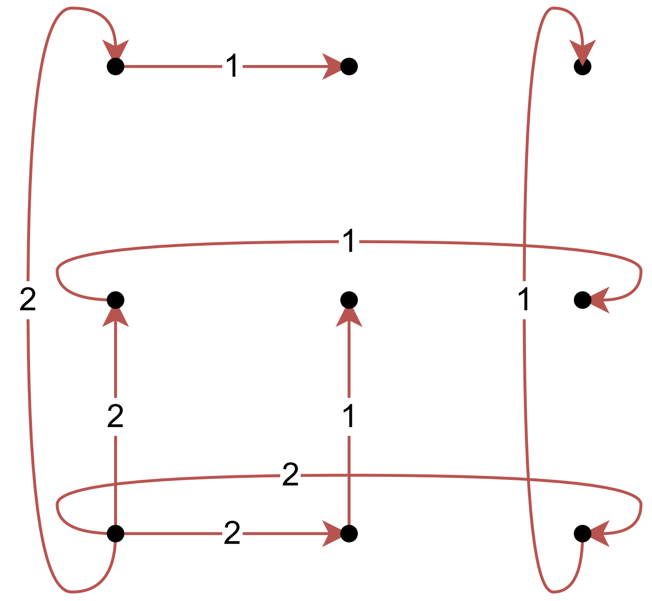
    (iv) $neg_2 = \{((u,v),(u,v-1)) : 0 \leq u,v \leq 2\}$

    with addition and subtraction modulo 3. For any spanning tree T rooted at node $(0,0)$, denote the channels directed away from the root by $E^+(T)$. We can think of such a spanning tree T as detailing 8 paths from $(0,0)$ to every other node.

    (a) Describe a spanning tree T as above so that

    - every channel of $E^+(T)$ is labelled with the number of paths on which it features, which we call the load of the channel
    - the sums of the loads of the channels in $E^+(T) \cap pos_1$, $E^+(T) \cap pos_2$, $E^+(T) \cap neg_1$, and $E^+(T) \cap neg_2$, respectively, are identical.

Solution: Consider the following spanning tree.



(b) Explain carefully how to use your spanning tree T so as to obtain a routing algorithm in Q_2^3 that under the all-to-all traffic pattern results in every channel of Q_2^3 having the same load.

Solution: To utilise our spanning tree T, we first show that Q_2^3 is node-symmetric. Consider the following.

(i) $\varphi_1(\overline{x}_1, \overline{x}_2) = (\overline{x_1 + 1}, \overline{x}_2)$; and

(ii) $\varphi_2(\overline{x}_1, \overline{x}_2) = (\overline{x}_1, \overline{x_2 + 1})$.

We claim that these both define automorphisms. For any $\boldsymbol{x}, \boldsymbol{y}$, it is clear that if $\varphi_1(\boldsymbol{x}) = \varphi_1(\boldsymbol{y})$ then $\boldsymbol{x} = \boldsymbol{y}$ and similarly for φ_2; thus both define permutations. Now, let $(\boldsymbol{x}, \boldsymbol{y})$ be a channel. Then $(\varphi_1(\boldsymbol{x}), \varphi_1(\boldsymbol{y}))$ is clearly also a channel (think about moving horizontally along the 3-ary 2-cube). A similar logic holds for φ_2. Now, we see by using φ_1 and φ_2, we can traverse every node on the network. Thus, Q_2^3 is node-symmetric. Therefore, by moving T (using compositions of φ_1 and φ_2), we can obtain a routing algorithm for any source to any destination in which *every channel has the same node* (this can be shown by simply studying the spanning tree, and by using the node-symmetry of Q_2^3).

12. The augmented cube AQ_n is constructed as follows. First, AQ_1 consists of a link joining node 0 and node 1. Given AQ_n, where $n \geq 1$, we construct AQ_{n+1} by taking two copies of AQ_n, call them AQ_n^0 and AQ_n^1, and renaming the nodes of AQ_n^0 (resp. AQ_n^1) by tagging 0 (resp. 1) onto their names. A simple induction yields that for all $n \geq 1$, the node set of AQ_n is $\{0,1\}^n$. Next, we join the node $(u_1, u_2, \ldots, u_n, 0)$ of AQ_n^0 with the node $(u_1, u_2, \ldots, u_n, 1)$ of AQ_n^1 and also the node $(\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_n, 1)$ of AQ_n^1 where for any $b \in \{0,1\}$, $b = 0$ if and only if $\overline{b} = 1$.

(a) What are the degrees of the nodes of AQ_n for $n \geq 1$? [You should justify your claim.]

Solution: We look at $n = 1$, we have AQ_1 being isomorphic to Q_1, and thus $\deg(AQ_1) = 1$. Next, we assume $\deg(AQ_k) = 2k - 1$ for some k. Now, we look at AQ_{k+1}. We can build this graph from two copies of AQ_k by associating each node in one copy with two nodes in the other. Thus, we get $\deg(AQ_{k+1}) = \deg AQ_k + 2 = 2(k+1) - 1$. Thus, by induction, we have $\deg(AQ_n) = 2n - 1$.

(b) Suppose that for some $n \geq 1$, any two nodes x and y of AQ_n have $\min\{\deg_n(x), \deg_n(y)\}$ mutually node-disjoint paths joining them, where $\deg_n(x)$ (resp. $\deg_n(y)$) is the degree of x (resp. y) in AQ_n. Suppose that $u = (u_1, u_2, \ldots, u_n, 0)$ and $v = (v_1, v_2, \ldots, v_n, 0)$ are two distinct nodes of AQ_{n+1}. Prove that u and v have $\min\{\deg_{n+1}(u), \deg_{n+1}(v)\}$ mutually node-disjoint paths joining them.

Solution: Let $\boldsymbol{u} = (u_1, \ldots, u_n, 0)$ and $\boldsymbol{v} = (v_1, \ldots, v_n, 0)$ be nodes in AQ_{n+1}. Firstly, by viewing the first n components of $\boldsymbol{u}$ and $\boldsymbol{v}$ in AQ_n, we can construct $2n - 1$ node-disjoint paths (by assumption). So, to complete our proof we must find 2 more. Let

$$\begin{aligned} \boldsymbol{v}' &= (v_1, \ldots, v_n, 1), \\ \boldsymbol{v}'' &= (\overline{v}_1, \ldots, \overline{v}_n, 1), \\ \boldsymbol{u}' &= (u_1, \ldots, u_n, 1), \\ \boldsymbol{u}'' &= (\overline{u}_1, \ldots, \overline{u}_n, 1). \end{aligned}$$

First, we assume that $n \geq 2$. Then we have two cases to look at.

(i) Suppose that for all $i \in \{1, \ldots, n\}$, $u_i \neq v_i$. Then we must have $\boldsymbol{u}' = \boldsymbol{v}''$ (and $\boldsymbol{u}'' = \boldsymbol{v}'$). Thus we construct the addition (node-disjoint) routes:

$$\begin{aligned} \boldsymbol{u} \to \boldsymbol{u}' = \boldsymbol{v}'' \to \boldsymbol{v}, \\ \boldsymbol{u} \to \boldsymbol{u}'' = \boldsymbol{v}' \to \boldsymbol{v}. \end{aligned}$$

(ii) Suppose otherwise. That is, there is $i \in \{1, \ldots, n\}$ such that $u_i = v_i$. We now consider the edges only in Q_n (not that added edges from our construction), and partition this into two copies of Q_{n-1} over dimension i. We see that $\boldsymbol{u'}$ and $\boldsymbol{v'}$ lie in the same partition, as do $\boldsymbol{u''}$ and $\boldsymbol{v''}$. Thus, as Q_{n-1} is connected, we can construct two node-disjoint routes:

$$\boldsymbol{u} \to \boldsymbol{u'} \to \ldots \to \boldsymbol{v'} \to \boldsymbol{v},$$
$$\boldsymbol{u} \to \boldsymbol{u''} \to \ldots \to \boldsymbol{v''} \to \boldsymbol{v}$$

as required.

Finally, for the case when $n = 1$. We can trivially see that the condition holds.

13. Prove that destination-tag routing in 2-ary n-flies always yields a path from a given source to a given destination.

Solution: Let $\boldsymbol{s} = (s_1, \ldots, s_n)$ be the source and $\boldsymbol{d} = (d_1, \ldots, d_n)$ be the destination. We first note that there are n switch nodes. We see that when $\boldsymbol{s}$ reaches the switch node $i \in \{1, \ldots, n\}$, the edge that the route follows corresponds to the ith most significant bit of $\boldsymbol{s}$ (that is, s_{n+1-i}) being flipped or not flipped in order to match the ith most significant bit of $\boldsymbol{d}$. Thus, we can see that after moving through all n switch nodes, each bit of our source *must* equal our destination. So we have ensured our route ends at $\boldsymbol{d}$, but how can we be sure that the route is valid? By definition, the ith switch route has an edge in which no bits are flipped, or the ith most significant bit is flipped. Thus the route is well-defined.

14. *Data centre networks* (DCNs) are interconnection networks for data centres and consist of interconnected servers and switches. Describe the difference between server-centric DCNs and switch-centric DCNs; in particular, explain any interconnection restrictions that occur and how routing is handled in both types of DCN. State one relative advantage of each type of DCN in comparison with the other. Say what a dual-port DCN is and explain why dual-port DCNs are important. Explain the *stellar construction* of dual-port DCNs and how we can obtain routing algorithms in DCNs built using the stellar construction. Give an illustration of the stellar construction with a hypercube Q_3.

Solution: A server-centric data centre network is such that the servers handle the routing; that is, algorithm-based. We are restricted by the processing speed on our router for calculating the routes for each packet. Alternatively, in a switch-centric data centre network, the switches handle the routing; that is, table-based. Here, we are restricted by the table size (memory). Switch-centric routing is fast in comparison to server-centric while server-centric routing is much more scalable.

A dual-port data centre network is a data centre network in which each server has at most two ports. Dual-port data centres are important as most commercially available servers (as well as servers in existing data centres) have two ports (a primary and a backup). Thus, we can use reduce the building cost of our data centre by using cheaper and more basic equipment.

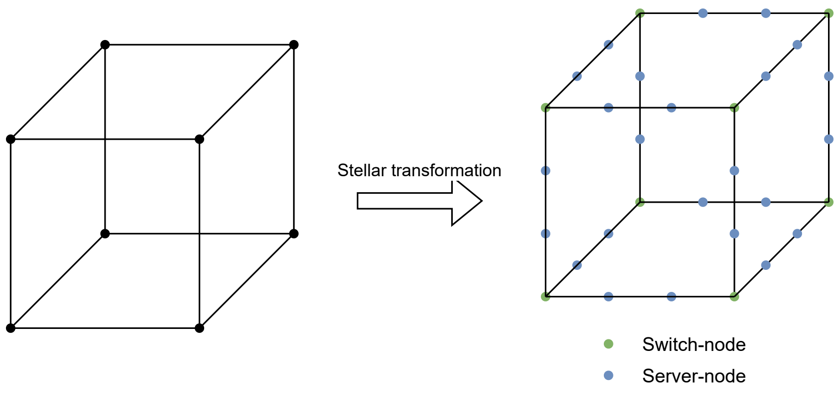
The *stellar construction* of dual-port data centre networks is a method of constructing a dual-port data centre network (or *stellar data centre network*) from any interconnection network. Let $G = (V, E)$ be an interconnected network where V is the set of nodes are $E \in \mathcal{P}(V^2)$ is a set of links. The stellar data centre network G^* is obtained by placing a switch-node v^* for each $v \in V$, then placing 2 server-nodes u^v, v^u for each $(u, v) \in E$ and adding a link between them (in G^*), then finally connecting u^* to u^v and v^* to v^u.

We now show how routes are transformed between a graph and its stellar construction. Given a route $P = (p_1, \ldots, p_n)$, we observe the route

$$P^* = (p_1', p_1^*, p_1^{p_2}, p_2^{p_1}, p_2^*, \ldots, p_{n-1}^*, p_{n-1}^n, p_n^{n-1}, p_n^*, p_n')$$

between p'_1 and p'_n, neighbours of the switch nodes p^*_1 and p^*_2 respectively. This path has size $2n+1$, but note, if $p'_1 = p_1^{p_2}$ or $p_n^{n-1} = p'_n$, we can reduce our path size by 1 (and by 2 if both hold). Thus, if we have an efficient routing algorithm in our base graph, our stellar construction may inherit it.

See below an illustration of the stellar construction with a hypercube Q_3.



15. Consider the bubble-sort interconnection network from Question 4.

(a) Generalize the definition in Question 4 and give an algebraic definition of the bubble-sort interconnection network B_n, where $n \geq 3$.

Solution: Let $B_n = (N, C)$. Then

$$\begin{aligned} N &= \{(a_1, \ldots, a_n) \in \{0, \ldots, n-1\}^n : a_i = a_j \implies i = j\}, \\ C &= \{((a_1, \ldots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \ldots, a_n), (a_1, \ldots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \ldots, a_n)) : \\ &\qquad i \in \{1, \ldots, n-1\}\}. \end{aligned}$$

(b) Devise a routing algorithm for B_n. Your algorithm should be described in pseudo-code so that the description is both precise and provides an intuitive understanding of how your algorithm works.

Solution:

```
function BubblesortRouting($n$, source, destination)
    function Compare($x$,$y$)
        return is $x$ before $y$ in destination?
    end function
    let path be an empty array
    for $i \in (0, \ldots, n-1)$ do
        for $j \in (0, \ldots, n-1-i)$ do
            if Compare(source[$j+1$], source[$j$]) then
                swap entry $j$ and $j+1$ in source
                append new value of source to path
                if source = destination then
                    return path
                end if
            end if
        end for
    end for
end function
```

The idea of this algorithm is to define a new strict total order (Compare function) on the set $\{0, \ldots, n-1\}$, and applying bubblesort as normal with this order. The path that is produced is a valid route in our bubblesort intersection graph.

(c) Implement your routing algorithm in Python. What are the loads of each node and each channel of B_6 in an all-to-all traffic pattern that result from an execution of your algorithm? Also, what is the maximal path-length and the average path-length produced by your algorithm? Comment on the loads on the various channel types. Both your algorithm and your analysis should be submitted.

Solution: For each node $(a_1, \ldots, a_6)$, the node load is 4681 and we have the following loads for the outgoing channels.

Channel	Load
$((a_1,a_2,a_3,a_4,a_5,a_6),(a_2,a_1,a_3,a_4,a_5,a_6))$	1044
$((a_1,a_2,a_3,a_4,a_5,a_6),(a_1,a_3,a_2,a_4,a_5,a_6))$	1368
$((a_1,a_2,a_3,a_4,a_5,a_6),(a_1,a_2,a_4,a_3,a_5,a_6))$	1332
$((a_1,a_2,a_3,a_4,a_5,a_6),(a_1,a_2,a_3,a_5,a_4,a_6))$	1056
$((a_1,a_2,a_3,a_4,a_5,a_6),(a_1,a_2,a_3,a_4,a_6,a_5))$	600

The maximal path-length is 15 and the average path length is 7.5. To give some more intuition on the channel types, for each node the channel corresponding to flipping the first two elements has load 1044. The channel corresponding to flipping the second and third element has load 1368 and so on with respective loads 1332, 1056, and 600. This lines up with, given an unordered list, the expected distribution of which adjacent pair of entries you'll have to switch. That is, you expect to switch the second and third the most when bubble sorting 6 elements.

16. Implement and submit a fault-tolerant routing algorithm for the hypercube Q_n. Given a set of channel faults, your algorithm should aim to tolerate these faults so as to come up with a path from a given source to a given destination that is as short as possible (given the faults).