Linear versus non-linear Boolean functions in Network Flow.

Sϕren Riis

December 2003

Abstract

It is shown that there exist network topologies such that congestion can only be avoided if messages are sent using non-linear Boolean functions. This can be achieved for any word length that is fixed in advance. To the best of my knowledge, this is the first published example where the use of non-linear Boolean functions is needed in *any* solution that maximises the total network flow.

The paper also highlights a new link between network flow and error correcting codes: There exist networks where the optimal flow is achieved by essentially selecting the worst error correcting code (rather than the best). The dichotomy between good and bad error correcting codes is highlighted in the paper.

Our constructions are for delay-free (and error-free) networks, but they can easily be adapted to more general models of network flow.

The major application of the construction is to illustrate some of the difficulties in proving lower bounds for the famous matrix transposition problem as well as sorting related problems.

1 Introduction

We show that there exist network topologies where any attempt to transmit data using ordinary routing techniques or even more general methods based on linear Boolean functions, will lead to deadlocks and congestion. Yet - and this is the surprising point - *it is possible to transmit all data to their destinations without congestion if the network uses non-linear Boolean functions.* Apparently this result contradicts the widespread belief that non-linear Boolean functions are never useful, and that data itself are best transmitted in a manner where no major calculations take place at different network nodes (except maybe in the source/target node where non-linear encryption or data-compression might be applied). One of the central and common assumptions in network flow is that data are being sent in a data-oblivious way [1]. That means that data are treated as a black box where processors can copy it, but cannot modify it.

Recently, a number of papers have challenged the 'conventional wisdom' [7, 11, 10, 9, 6, 4, 8] that there is little - if any - point in network coding. Instead these papers show that information is sometimes better transmitted as being 'diffused' throughout the network from the source to the sinks by means of network coding. In [4], some of the published examples use non-linear flows on the bit level. The use of non-linear functions can however be eliminated from these examples. Actually, by a minor modification of the elegant main theorem of [11, 10], we will show that any single source multicast problem has a bit-wise linear solution (on blocks of a suitable size) whenever it has a solution at all. Thus the non-linear Boolean functions underlying the constructions in [4] can be eliminated (which essentially was also shown in [11, 10]). Our example is - to the best of my knowledge - the first published example where the use of non-linear Boolean functions is proved to be necessary for delay-free (error-free) networks. It is also the first example that shows that there exist networks where certain messages are best transmitted in a fashion where they all have pairwise small hamming distance! All published examples of network flow essentially try to maximise the error correction capability of the network coding. In our construction the greatest network performance is achieved by minimising (rather than maximising) the error correction of certain signals!

The suggestion of sending data in a non-linear fashion has already been seriously discussed are in the context of the famous matrix transposition problem. In the seminal paper [3] the authors raise this challenging open problem. They write:

> *Intuitively, the lower bound should still hold in this more general model, since it is unlikely that these operations [arbitrary bit manipulations] are of any great help, but no proof is known. Such a proof would no doubt provide great insight into the nature of information transfer and sorting-related computations.*

The matrix transposition problem is still open. In fact it was when trying to resolve the matrix transposition problem, that I managed to produce the 'counter examples' showing that non-linear Boolean functions are sometimes needed in the solution of flow problems in delay-free networks. It still seems highly plausible that non-linear Boolean functions [i.e. arbitrary bit manipulations] are of little help for the matrix transposition problem. However the constructions in this paper suggest that that this might be very difficult to prove.

We leave it as an interesting open question, whether the use of non-linear flow can be eliminated by allowing sufficiently large block computations (see also [5] for a discussion of this conjecture).

We will not attempt to actually design protocols for finding non-linear solutions when such solutions exist. In practice the flow has to be controlled by, for example, using special header bits which might contain routing information as well as information which insures that packets are received in the correct order when they are delivered. When huge and massive data-flows are involved, the 'price' for consulting a somewhat centralised "traffic control" becomes very small, while the potential increase in the overall network performance could be huge. *We do not prove - or disprove - whether there could be any practical applications of allowing a (semi)central "traffic control" the use of non-linear Boolean functions.*

Before, we consider non-linear flows we show a few straightforward results concerning linear flows.

1.1 Linear flows are sometimes better than pushing bits

Consider the following network topology in figure 1 (a):

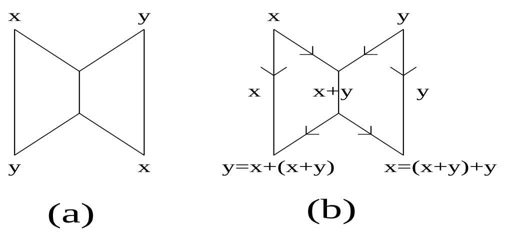


Figure 1

In this network (for a formal definition of delay-free network see [11, 10]) the task is to transmit a single bit x from the upper left node to the lower right node, and at the same time transmit a single bit y from the upper right node to the lower left node. In this network each channel can only carry one bit. If we try to transmit the bits x and y without interference, a deadlock is created in the middle channel where the messages x and y block the way for each other. If however we follow the approach illustrated in the network in figure 1 (b) and send the exclusive OR i.e. $x+y$, along the middle wire, we can send the bit x and y to their destinations without any deadlock. Thus it is possible to solve the broadcasting task using the linear Boolean functions $(x, y) \to x + y$,$(x, x + y) \to x + (x + y)$ and $(x + y, y) \to (x + y) + y$.

This example, is of course, highly unrealistic. However the example remains valid if x and y instead of representing single bits, represent general data flows. Suppose that $\overrightarrow{x}$ and $\overrightarrow{y}$ are two data-flows given by $\overrightarrow{x} := \ldots x_{-2}x_{-1}x_0x_1x_2\ldots$ and $\overrightarrow{y} := \ldots y_{-2}y_{-1}y_0y_1y_2\ldots$. Assume the task is to transmit the data-flow $\overrightarrow{x}$ (rather than the single bit x) from the upper left node to the lower right node, and at the same time to transmit the data flow $\overrightarrow{y}$ from the upper right node to the lower left node. In this scenario we assume that the network can do computations for words of arbitrary length (so-called block computations which are defined in [11, 10, 17, 4]). In this case there is no need for block computations and the task can be solved using words of length 1 by sending the data-flow $\overrightarrow{x} + \overrightarrow{y} := \ldots (x_{-2}+y_{-2}), (x_{-1}+y_{-1}), (x_0 + y_0), (x_1+y_1), (x_2+y_2), \ldots$. through the middle channel. There is no solution which just 'pushes' blocks (of any given length).

1.2 When doubling the bandwidth is more than Twice as good!

The purpose of this section is to illustrate the potential role of block computations as well as to show a somewhat counter intuitive phenomena that occur even when only a linear flow is involved. Yet, we will show (by a minor modification of [11]) that single source multi-cast problems can never be truly pathological and that linear Boolean functions (applied to blocks of a fixed but suitable size) always suffice.

At first the following seems quite absurd! We claim that for a fixed buffer size i.e. word length (=block length), doubling the bandwidth sometimes makes it possible to more than double the overall performance of the network!

To illustrate this consider the single source multicast problem in figure 2. Two bits (data-streams) x and y have to be sent from a single source S to a set of targets (nodes $1, 2, 3, 4, 5, 6$) at the bottom of the network.

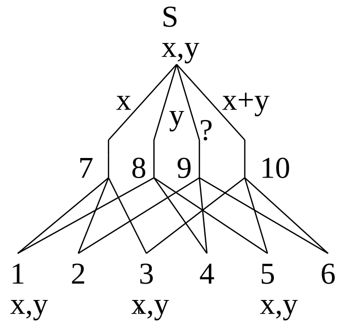


Figure 2

We claim that at least one of the six target nodes will fail to receive both x and y. To illustrate the problem assume, for example, that we decide to send x through channel $(s, 7)$, decide to send y through channel $(s, 8)$ and decide to send $x+y$ through channel $(s, 10)$. Which Boolean function $f(x, y)$ should we send through channel $(s, 9)$? If we send x through $(s, 9)$ node 2 is not able to reconstruct x and y. If we send y through $(s, 9)$ node 4 is not able to reconstruct x and y. It is no better if we send $x+y$ through $(s, 9)$ since then node 6 is not able to output x as well as y. It is not hard to show that in this example the use of non-linear Boolean functions is of no help. Thus *if the bandwidth is one, the bits x, y can only be transmitted correctly to five of the six nodes (i.e. at most 11 out of 12 bits can be guaranteed to arrive correctly).*

Now let us double the bandwidth, while keeping the buffer size, i.e. the word length, at one. The task of sending bits x_1, x_2, x_3, x_4 to the six nodes at the bottom can be solved as shown in figure 3.

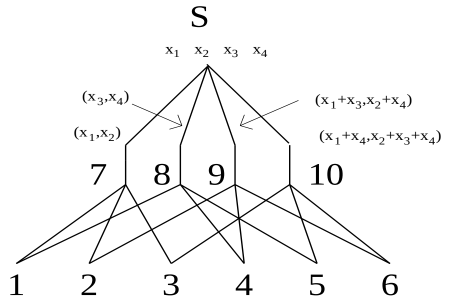


Figure 3

Notice that each of the six nodes is now able to receive the bits x_1, x_2, x_3, x_4 if we select appropriate linear functions at the bottom nodes. As an example node 6 receives the bit values $x_1 + x_3$, $x_2 + x_4$ from node 9 and receive the bit values x_1+x_4 and $x_2+x_3+x_4$ from node 10. From these bit values it is possible to reconstruct the original bits as follows: $x_1 = (x_1 + x_3) + (x_2 + x_4) + (x_2 + x_3 + x_4)$, $x_2 = (x_2 + x_3 + x_4) + (x_1+x_3)+(x_1+x_4)$, $x_3 = (x_2+x_4)+(x_2+x_3+x_4)$ and $x_4 = (x_1 + x_4) + (x_1 + x_3) + (x_2 + x_4) + (x_2 + x_3 + x_4)$. Similar, but even simpler, constructions are possible for nodes $1 - 5$.

Thus *if the bandwidth is two (and the word length remains 1), the bits x_1, x_2, x_3 and x_4 can be transmitted correctly to all the six nodes (i.e. 24 out of 24 bits can be guaranteed to arrive correctly).* Voila! By doubling the bandwidth from 1 to 2 we managed to increase the performance from 11 to 24 bits!

Instead of doubling the bandwidth we could double the word length (i.e. the buffer size). Alternatively, we can solve the problem using the elegant method of diversity coding [6] by working over a field F_p with $p \geq 3$ being a prime number and letting $f(x,y) := x+2y$ be the flow through channel $(s,9)$ in the first diagram. This solution would essentially work since words of length l (for l large), can be converted into numbers in base p with only an insignificant reduction of performance. Notice, however that this solution - though linear on the semantic level - is non-linear on the bit level. As we already noticed the problem has actually a linear solution (for block size 2). This is not a coincidence since:

Theorem(1): Any single source multi-cast problem has a solution[1] if and only if it has a linear solution[1]. Furthermore, a single source multi-cast problem always has a solution[1], except when this can be ruled out for the obvious reason that the max flow to one of the targets is too small .

The fact that the mincut bound is achievable by linear codes over sufficiently large fields was first shown in [12]. Later (in [11]) this result was improved to smaller fields with size being a power of 2.

From this theorem it follows that our multi-cast problem has a solution (as we have already seen), since the max flow from the source to each of the targets is 2. It also follows that this solution can be chosen to be linear (even bit-wise linear) for a suitable choice of word length.

Theorem(1) follows from an easy modification of Theorem 2, in [11]. According to this theorem if a single source multi-cast problem has a solution, then there is a linear solution over the field F_{2^l} for a suitable choice of l. To prove Theorem(1) it suffices to show that the linear solution (over F_{2^l}) can be converted to a bit-wise linear solution of block length l. Since F_2 is a subfield of F_{2^l} we can view F_{2^l} as a l dimensional vector space over F_2. Notice that any F_{2^l}-linear map $\psi : F_{2^l} \oplus \ldots \oplus F_{2^l} \to F_{2^l}$ is also a F_2-linear map. Let $e_1, e_2, \ldots e_l \in F_{2^l}$ be a basis for this vector space and let us identify the string $(\lambda_1, \lambda_2, \ldots \lambda_l) \in F_2^l$ with the vector $\Sigma_j \lambda_j e_j \in F_{2^l}$ (here an inappropriate identification would lead to non-linear Boolean functions). Let $b : F_2^l \to F_{2^l}$ be the bijection given by $b(\lambda_1, \lambda_2, \ldots \lambda_l) = \Sigma_j \lambda_j e_j$. The induced map $\psi' : F_2^l \oplus \ldots \oplus F_2^l \to F_2^l$ defined by $\psi'(r_1, r_2, \ldots r_s) := b^{-1} \circ \psi(b(r_1), b(r_2), \ldots b(r_s))$ is clearly F_2-linear since the bijection $b : F_2^l \to F_{2^l}$ is F_2-linear. Thus any linear solution (over F_{2^l}) can be converted to a bit-wise linear solution of block length l.

[1]Solutions are allowed to work on blocks of any fixed length

Rudolf Ahlswede has pointed out (personal communication) that the construction in this section is directly related to the construction of a special type of error correcting codes. More specifically the example above (with bandwidth 1) is equivalent to the question whether there is a $(4,4,3)$ 2-ary MDS code (there is none). The same question for bandwidth 2 is equivalent to the question whether there is a $(4,16,3)$ 4-ary MDS code. In effect there is a linear $(4,16,3)$ 4-ary MDS code, which also shows that there exists a linear boolean solution for bandwidth 2.

2 The main results

2.1 Non-linear flow is sometimes better than linear flow

Consider the network T in figure 4:

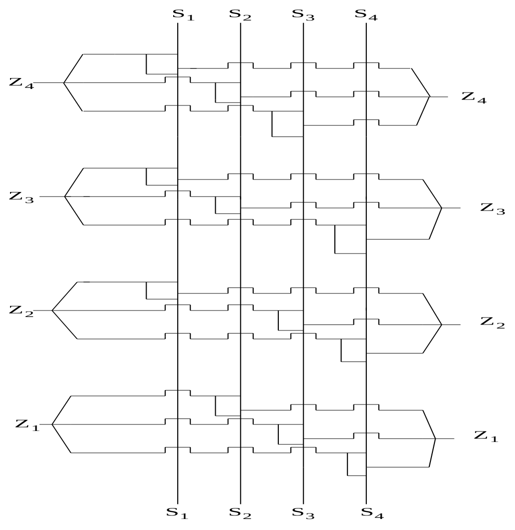


Figure 4

This network, which will serve as a building block in our construction, has the following curious property:

Proposition(2): *Assume we want to ensure all 4 bits z_1, z_2, z_3 and z_4 are successfully transmitted from the left side to the right side in the diagram. If only linear signals are allowed it is possible to send at most four distinct messages through the vertical channels. If however non-linear flows are allowed it is possible to send five distinct messages (and this is optimal) through the vertical channels.*

Proof: Along the four vertical channels we send signals s_1, s_2, s_3 and s_4. And horizontally the aim is to send the signals z_1, z_2, z_3 and z_4. Assume first that all signals are sent in a linear fashion. More specifically let ρ be any assignment of linear functions to each node in the network. Under this assignment the signals $z_i + L_i(s_1, s_2, s_3, s_4)$, $i = 1, 2, 3, 4$, (where L_1, L_2, L_3 and L_4 are linear Boolean functions) are transmitted to the right hand nodes z_i, $i = 1, 2, 3, 4$. Without loss of generality we can assume that the signals s_1, s_2, s_3 and s_4 arrive unscrambled at the bottom destinations. We will view each message $(s_1, s_2, s_3, s_4) \in \{0,1\}^4$ as a vector in the four dimensional vector space F_2^4 over the field F_2. Let $V_\rho \subseteq F_2^4$ denote the linear subspace spanned by all vectors for which $L_i(s_1, s_2, s_3, s_4) = 0$ for $i = 1, 2, 3, 4$ (notice that this is indeed a linear subspace since it is the intersection of the kernel of the linear maps L_i, $i = 1, 2, 3, 4$). We claim that the vector space dimension $dim(V_\rho)$ of V_ρ is at most 2 (i.e. $dim(V_\rho) \leq 2$). To see this, assume that $dim(V_\rho) \geq 3$. Let $pr_i : F_2^4 \to F_2^3$,$i = 1, 2, 3, 4$ be the projections given by $pr_1(s_1, s_2, s_3, s_4) = (s_2, s_3, s_4)$, $pr_2(s_1, s_2, s_3, s_4) = (s_1, s_3, s_4)$,$pr_3(s_1, s_2, s_3, s_4) = (s_1, s_2, s_4)$ and $pr_4(s_1, s_2, s_3, s_4) = (s_1, s_2, s_3)$. Since we assumed $dim(V_\rho) \geq 3$ at least one of the projections pr_i must be onto i.e. for at least one of the projections we must have $pr_i(V_\rho) = F_2^3$. If however $pr_i(V_\rho) = F_2^3$ it is not hard to show that the variable z_i cannot be sent correctly i.e. that $L_i(s_1, s_2, s_3, s_4) \neq 0$ for certain $(s_1, s_2, s_3, s_4) \in V_\rho$. This contradicts the definition of V_ρ. Thus $dim(V_\rho) \leq 2$ as we claimed. Thus the number of distinct messages we can send along the vertical channels is at most $2^{dim(V_\rho)} \leq 4$. Clearly it is possible to send 4 messages (let for example $s_3 = s_4 = 0$) through the vertical channels. Thus the optimal number of messages we can send through the vertical wires is 4 in the case where all flows are linear.

Next, consider non-linear flows. Consider the five messages $\overrightarrow{m_0} := (0,0,0,0)$, $\overrightarrow{m_1} := (1,0,0,0)$, $\overrightarrow{m_2} := (0,1,0,0)$, $\overrightarrow{m_3} := (0,0,1,0)$ and $\overrightarrow{m_4} := (0,0,0,1)$. If we only send these messages through the vertical wires we can ensure that the signal z_4 arrives unscrambled, if we use the non-linear construction in figure 5:

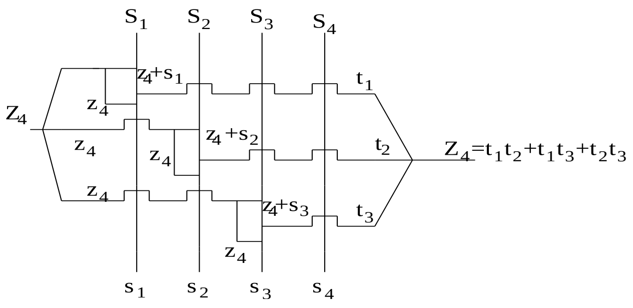


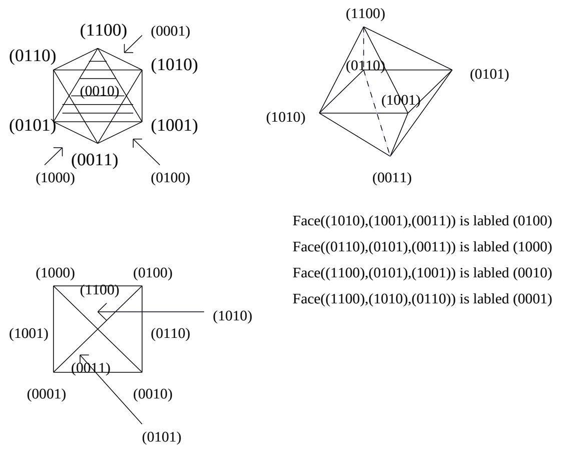
Figure 5

Here $t_1 := s_1 + z_4$, $t_2 := s_2 + z_4$ and $t_3 := s_3 + z_4$. We have to check that $z_4 = (s_1 + z_4)(s_2 + z_4) + (s_1 + z_4)(s_3 + z_4) + (s_2 + z_4)(s_3 + z_4)$ whenever (s_1, s_2, s_3, s_4) is one of the five messages $\overrightarrow{m_0} := (0,0,0,0)$, $\overrightarrow{m_1} := (1,0,0,0)$, $\overrightarrow{m_2} := (0,1,0,0)$, $\overrightarrow{m_3} := (0,0,1,0)$ and $\overrightarrow{m_4} := (0,0,0,1)$. To do this notice that $z_4 = (1 + z_4)(0 + z_4) + (1 + z_4)(0 + z_4) + (0 + z_4)(0 + z_4)$ and that $z_4 = (0 + z_4)(0 + z_4) + (0 + z_4)(0 + z_4) + (0 + z_4)(0 + z_4)$. The cases of $\overrightarrow{m_2} := (0,1,0,0)$, $\overrightarrow{m_3} := (0,0,1,0)$ are treated similarly.

The other parts of the network (corresponding to z_1, z_2 and z_3) are treated similarly by the obvious modifications. We ensure that the signals z_1, z_2 and z_3 arrive unscrambled by letting $z_1 := (s_2 + z_1)(s_3 + z_1) + (s_2 + z_1)(s_4 + z_1) + (s_3 + z_1)(s_4 + z_1)$, $z_2 := (s_1 + z_2)(s_3 + z_2) + (s_1 + z_2)(s_4 + z_2) + (s_3 + z_2)(s_4 + z_2)$ and by letting $z_3 := (s_1 + z_3)(s_2 + z_3) + (s_1 + z_3)(s_4 + z_3) + (s_2 + z_3)(s_4 + z_3)$.

To complete the proof it suffices to show that it is impossible to send more than five messages through the vertical channels. Consider the part of T corresponding to the variable z_4. In general the signals t_1, t_2 and t_3 are of the form $t_i := \alpha_i z_4 s_i + \beta_i z_4 + \gamma_i s_i + \delta_i$ with $i = 1, 2, 3$ and $\alpha_i, \beta_i, \gamma_i, \delta_i \in \{0,1\}$. Notice however that the signal s_i can be reconstructed only if $\alpha_i = 0$ and $\gamma_i = 1$. With out loss of generality we can assume $\delta_i = 0$ and thus we can assume that $t_i := \beta_i z_4 + s_i.$, $i = 1, 2, 3$. Clearly at least one of $\beta_1, \beta_2, \beta_3$ must have value 1. The most 'clean' case appears when $\beta_1 = \beta_2 = \beta_3 = 1$. In this case the variable z_4 can be transmitted without error if and only if for any pair of code words (s_1, s_2, s_3, s_4) and (t_1, t_2, t_3, t_4) the hamming distance between (s_1, s_2, s_3) and (t_1, t_2, t_3) is ≤ 2. If $\beta_1 = \beta_2 = 1$ and $\beta_3 = 0$ the variable z_4 can be transmitted without error if and only if for any pair of code words (s_1, s_2, s_3, s_4) and (t_1, t_2, t_3, t_4) we have $(s_1 + 1, s_2 + 1, s_3) \neq (t_1, t_2, t_3)$. Finally, if $\beta_1 = 1$ and $\beta_2 = \beta_3 = 0$ the variable z_4 can be transmitted without error if and only if for any pair of code words (s_1, s_2, s_3, s_4) and (t_1, t_2, t_3, t_4) we have $(s_1 + 1, s_2, s_3) \neq (t_1, t_2, t_3)$.

Now consider the network T. We want to maximise the number of messages (code words) we can send through the vertical channels. We leave most cases to the reader. Here we only check the case where $\beta_1 = \beta_2 = \beta_3 = 1$ not just for z_4 but also for z_1, z_2 and z_3. In this case we notice that the variables z_1, z_2, z_3 and z_4 can be transmitted without error if and only if any pair of code words (s_1, s_2, s_3, s_4) and (t_1, t_2, t_3, t_4) has hamming distance ≤ 2.



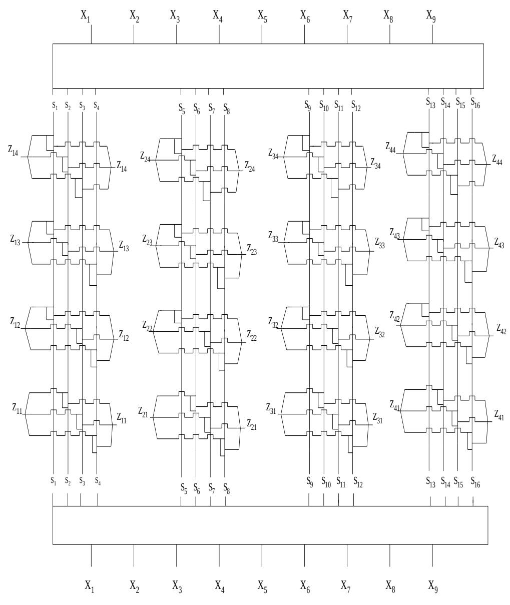
Face((1010),(1001),(0011)) is labled (0100)
Face((0110),(0101),(0011)) is labled (1000)
Face((1100),(0101),(1001)) is labled (0010)
Face((1100),(1010),(0110)) is labled (0001)

Figure 6

Figure 6 contains a few nice geometric interpretations of the problem of constructing codes with a hamming 'diameter' ≤ 2 (in each display, we assume that $(0, 0, 0, 0)$ is a code word). Whatever way we look at it we notice that there is only one way to select five code words having diameter 2 and containing $(0, 0, 0, 0)$. ♣

Given the "devise" T it is not hard to construct examples where non-linear flow makes it possible to send more bits (rather than just more messages). The smallest example we have been able to construct this way contain 25 sources/targets.

The network topology Y in figure 7 uses four copies of the network T. The two boxes in figure 7 represent the bipartitioned graphs $K_{9,16}$. Notice that the network $K_{9,16}$ at the top allows us to compute any Boolean function $f : \{0,1\}^9 \to \{0,1\}^{16}$ and that the network $K_{16,9}$ at the bottom allows us to compute any Boolean function $f : \{0,1\}^{16} \to \{0,1\}^9$.


X1 X2 X3 X4 X5 X6 X7 X8 X9
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16
Z14 Z24 Z34 Z44
Z13 Z23 Z33 Z43
Z12 Z22 Z32 Z42
Z11 Z21 Z31 Z41
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16
X1 X2 X3 X4 X5 X6 X7 X8 X9
Figure 7

Next we show that this network topology Y has the following property:

Theorem(3) *The problem of transmitting the* 25 *data-flows* $x_1, x_2, \dots x_9$ *and* z_{ij} $i, j \in \{1, 2, 3, 4\}$ *from their sources (top/left) to their destinations (bottom/right) has a non-linear solution. The problem has no linear solution (of block length* 1*).*

Proof: We can send $5^4 = 625$ distinct messages through the vertical channels (without messing up the 16 z variables). Hence it is possible to send the 9 bits ($2^9 = 512$ messages) through the vertical channels. Thus the transmission problem has a solution using non-linear functions. We have to show that the transmission problem has no linear solution. Assume that there is a linear solution ρ and consider the linear map: $\psi(x_1, x_2, \ldots x_9) : Z_2^9 \rightarrow Z_2^{16}$ defined by $\psi(x_1, \ldots x_9) := (s_1(x_1, \ldots x_9), s_2(x_1, \ldots x_9), \ldots s_{16}(x_1, \ldots x_9))$ with $s_j : F_2^9 \rightarrow F_2$. We claim that the rank of ψ is at most 8. Assume that it is strictly larger than 8. Consider the projections of ψ to the linear subspaces $V_1 := span\{s_1, s_2, s_3, s_4\}$, $V_2 := span\{s_5, s_6, s_7, s_8\}$, $V_3 := span\{s_9, s_{10}, s_{11}, s_{12}\}$ and $V_4 := \{s_{13}, s_{14}, s_{15}, s_{16}\}$. One of these must have rank > 2, which contradicts proposition 1. Thus we can broadcast at most 8 bits through the vertical channels if we are only allowed to use linear Boolean functions (if we let $s_3 = s_4 = s_7 = s_8 = s_{11} = s_{12} = s_{15} = s_{16} = \overrightarrow{0}$ we notice that it is actually possible to archive this and broadcast 8 bits along the vertical channels). Since any solution needs to transmit 9 bits, there is no linear solution (of block length 1) ♣

2.2 A linear solution of finite Block size

While I circulated [13] among leading experts in the network coding community, Ralf Koetter responded that the network coding problem for T (and thus Y) has a linear solution using block length 3:

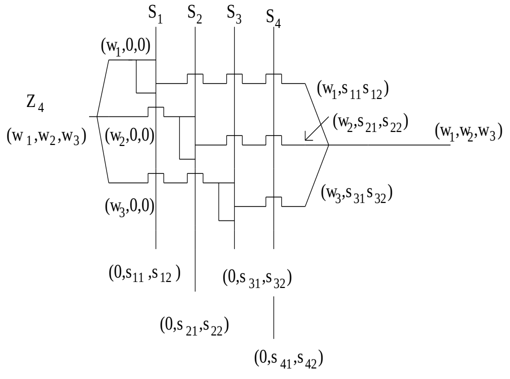


Figure 8

Ralf Koetter's solution outlined in figure 8, resolves the non-linear flow in figure 5. This can be explained as follows: We consider three time slots. In the first time slot, no information is transmitted on the vertical lines and hence we can transmit three bits of information for all the horizontal transmissions Z_1, Z_2, Z_3 and Z_4 (in figure 4) by using the three parallel connections for different bits. In both of the subsequent two time-slots, four bits of information are transmitted on the vertical transmissions, which is possible if no bits are transmitted across the network. Altogether, during three time intervals we transmit 1 bit per time slot for Z_1, Z_2, Z_3, Z_4, and $8/3$ bits per time slot through the vertical channels. Now since $8/3 > \log_2(5)$ this provides a linear (vector) solution that matches the rate of the non-linear solution (of block size 1).

We have already seen that there are network flow problems where there is no solution of block length 1, but where there exist linear solutions of block lengths > 1. Recently an example by R. Koetter (mentioned in [5]) shows that there exist a network flow problem with a linear solution of finite block length, yet the problem has no bit-wise linear solution over any field. The construction in figure 7 gives another such example. The flow problem Y has no bit-wise linear solution over any field, yet it has a linear solution of block length 3. The main point however is that the flow problem in Y has non-linear solutions (bit-wise).

3 Lifting the results to networks with higher bandwidth

The main constructions (figure 4 and figure 7) were proved to be valid only for networks with bandwidth (block size) 1. The constructions can, however, be lifted to situations of higher bandwidth ω using the following devise $A_{\omega,r}$:

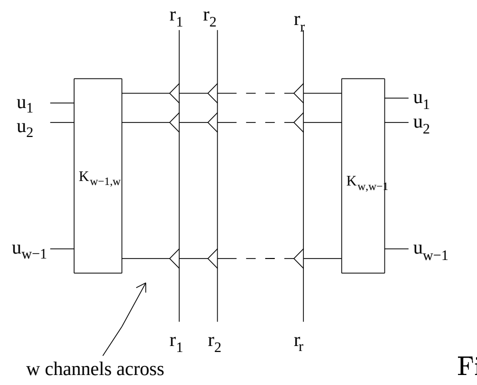


Figure 9

Proposition(4): *Assume all channels in figure 9 have bandwidth $\omega > 1$. Let $u_1, u_2, \dots u_{\omega-1}$ be distinct variables, each representing an element in $\{0,1\}^{\omega}$. There are ω horizontal channels across and any number $r \geq 1$ of vertical channels. If each of the u variables are guaranteed to be sent across unscrambled, there is a solution that transmit two distinct messages (i.e. one bit) along each vertical channel. There are no solutions that transmit more than two messages (i.e. more than one bit) through a vertical channel.*

Proof: All channels have bandwidth ω. First we show that there is a solution that transmits exactly one bit through each vertical channel. Assume that $u_j := (u_{j1}, u_{j2}, \dots u_{j\omega})$ and send $(u_{1j}, u_{2j}, \dots u_{\omega-1,j}, 0)$ $j = 1, 2, \dots \omega$ through the horizontal channels. Send $(0, 0, 0, \dots 0, r_k)$ through the kth vertical channel. Notice that this can be achieved without the horizontal (vertical) channels "contaminating" each other. To prove the last claim we have to show that there is no solution where one of the channels r_j sends more than two distinct signals. *Assume* that the channel r_j sends three (or possibly more) distinct signals m_1, m_2 and m_3. We need to be able to send $2^{\omega(\omega-1)}$ messages through the horizontal channels. The number of messages which can be sent through each horizontal channel is at most $2^{\omega}/3$. Thus the total number of messages that can be sent through the ω horizontal channels is at most $(2^{\omega}/3)^{\omega} < 2^{\omega(\omega-1)}$. This is a contradicts the assumption. ♣

Assume we modify the network Y by increasing the bandwidth of each channel to ω. For each channel c in Y we add a copy of the devise $A_{\omega,1}$ such that the channel c is identical to the vertical channel r_1 in $A_{\omega,1}$. Denote this new network Y_{ω} and notice that the number of nodes of Y_{ω} is bound by $c\omega$ for a suitable constant c. Notice that there is a one-to-one correspondence between the solutions to Y (bandwidth 1) and solutions to the modified network Y_{ω}(bandwidth ω). Furthermore for any $\omega \in N$ given in advance, the network Y_{ω} has only non-linear solutions of block size ω. Thus we have shown:

Theorem(5): There exists a constant $c \in N$, such that for any $\omega \in N$, there exists a network Y_{ω} such that:

- There is no linear solution of bandwidth $\leq \omega$.
- There is a non-linear solution of bandwidth $\leq \omega$.
- Y_{ω} contains less than $c\omega$ nodes.

Applications to the matrix transposition problem

Next we consider a famous open problem from network complexity theory: *the matrix transposition problem*. This problem was raised in [3] and discussed in more details in [16]. One version of the problem is also described explicitly in [2]. The problem can be described as follows: Consider an acyclic network which has n source nodes at the top, and where each node (except the source nodes) has fan-in 2. Besides having n input nodes at the top, the network has n output nodes at the bottom. The task of the network is to transpose any given $n \times n$ matrix A. More specifically the rows of the matrix A are input (as n, n-bit words) into the input nodes (each input node receiving a different row in A). The output nodes have to output the n columns of A. Each node can convert any two n-bit input words - using *any* possible Boolean function $f : \{0,1\}^{2n} \rightarrow \{0,1\}^{n}$- to an n-bit output word. The matrix transposition problem is to show that any such network which transposes any given $n \times n$ matrix input matrix, must contain at least $\Omega(n \log n)$ nodes.

The intuition behind this 'conjecture' is that it is hard to see that complex Boolean functions or even linear Boolean functions, should be of any help in solving this problem. Furthermore, the natural merge-sort approach which uses $n \log n$ nodes seems hard to improve. In the merge sort network each bit follows a predefined path, and is never even copied. Yet intuitively this network seems to be optimal.

A natural approach for solving the matrix transposition problem is first to show that non-linear Boolean functions are of no use, and that without loss of generality we can assume that all possible network coding is linear. To prove non-trivial lower bounds for this case still seems to be very hard, but somehow it appears much more tractable than the messy non-linear case. Our constructions suggest that it might be very hard to reduce the problem to the purely linear case.

Given the results in [2] we notice that any non-trivial lower bound for the matrix transposition problem (when arbitrary network coding is allowed) would give a non-trivial lower bounds for the fast Fourier transform as well as for sorting (in a computational model which is stronger than the comparison-based model considered in [2]). Thus our counter example also pinpoints to some of the difficulty for proving lower bounds for sorting related problems.

Open questions

Our constructions raise a number of natural and important questions. Maybe the most important question is whether any flow problem can be solved using linear coding. This interesting possibility was first suggested in [5]. Our construction shows that the linear coding in general needs to have block length $\Omega(n)$ where n denotes the number of nodes of the network. How much is it possible to amplify the advantage of non-linear Boolean functions over linear Boolean functions? Is it possible to increase the (bit-wise) performance by more than a logarithmic factor in the size of the network?

It seems that the usefulness of non-linear network coding is somewhat 'atypical' and that 'most' network flow problems have linear solutions (of small block size) if they have solutions at all. Is it possible to formalise and prove that this intuition is correct? Is it possible to identify nice classes of network topologies where it is possible to show that non-linear boolean functions are not needed? The complexity of finding optimal (linear) solutions for single source multi-cast problems has been shown to be essentially quadratic in the size of the underlying network [15, 9]. Is there a fast algorithm that can detect if a given problem requires the use of non-linear Boolean functions? Is there a general method which allows us to modify a given network, (where channels might have capacity $\omega > 1$) or to obtain an 'equivalent' network which only has solutions if it has linear solutions?

The link between network coding and error correcting codes seems very interesting. According to Rudolf Ahlswede (personal communication) it should be possible to use MDS codes to construct multi-cast problems that have only non-linear solutions. It is possible to implement his suggestion using the Nordstrom-Robinson code [14], which is a non-linear $(12, 32, 5)$-2-ary MDS code with no linear solution. To do this, consider a network with source s and 12 nodes $a_1, a_2, \dots a_{12}$, each accessed by s with one edge. Furthermore for each 8-element subset of $\{a_1, a_2, \dots a_{12}\}$ there is a specified sink connected to all of its elements. So there are $\binom{12}{8}= 495$ sinks. Using non-linear flows it is possible to send 5 bits from s to each of the 495 sinks as each sink can reconstruct the original code word $a_1 a_2 a_3 \dots a_{12}$ since (at most) 4 bits of the 12 bits are unknown. It should be emphasised that this construction only seems to work for networks with bandwidth 1. Thus it is an open question whether there exist multi-cast problems (of band width ≥ 2) that only have non-linear solutions. It is also an open question if it is possible to use error correcting codes to construct examples like ours, where each source (sink) sends (receives) independent messages.

While analysing the network T we noticed that there is a distinct advantage in letting the class of code words send through the vertical channels having as diameter (in terms of hamming distance) as possible. Thus there seems to be a very interesting dichotomy between situations where the main strategy is to maximise the error correcting ability (like in the examples based on MDS-coding), and different situations (like in the network T) where the main strategy is rather to minimise the error correcting level. The trade-off between such different situations could lead to new research questions.

Acknowledgements:

I would like to thank Mikkel Thorup for many very rewarding discussions concerning the matrix transposition problem. Also I would like to thank him for helping me to knock down an earlier (and somewhat naive) attempt to reduce the matrix transposition problem to the linear case. I would also like to thank Micah Adler, Rudolf Ahlswede, Donald Preece, and Muriel Medard for their useful comments during this research. Finally, I would like to offer a special thank to Ralf Koetter for his highly valuable feedback concerning the existence of linear solutions of finite block size.

References

[1] M Adler, F Fich, L A Goldberg, and M Paterson. Tight size bounds for packet headers in narrow meshes.

[2] M Adler and T Leighton. Compression using efficient multicasting. *in Proceedings of Symposium on Theory of Computing (STOC)*, 32, 2000.

[3] A Aggarwal and J S Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1126, September 1988.

[4] R Ahlswede, N Cai, Li, and R Yeung. An algebraic approach to network coding. page 104, 2001.

[5] M Effros, M Medard, T Ho, and D Karger. On coding for non-multicast networks. In *41st Annual Allerton Conference on Communication Control and Computing, 2003*, Oct 2003.

[6] T Ho, D Karger, M Merard, and Koetter R. Network coding from a network flow perspective. In *Proocedings of the ISIT 2003, Yokohama, Japan*, July 2003.

[7] T Ho, Merard M, and Koetter R. A coding view of network recovery and management for single-receiver communications. In *2002 Conference on Information Sciences and Systems, Princeton University*, 2002.

[8] T Ho, M Medard, and R Koetter. An information theoretic view of network management. In *Prooceeding of the 2003 IEEE Infocom.*

[9] S Jaggi, P Chou, and K Jain. Low complexity algebraic multicast network codes. In *Proocedings of the ISIT 2003, Yokohama, Japan*, 2003.

[10] R Koetter and M Medard. An algebraic approach to network coding. In *Proocedings of the 2001 IEEE International Symposium on Information Theory.*

[11] R Koetter and M Medard. Beyond routing: An algebraic approach to network coding. In *Proceedings of the 2002 IEEE Infocom*, 2002.

[12] Li, Yeung, and Cai. Linear network codes. *IEEE Trans. v.49,371-381.*

[13] S Riis. Linear versus non-linear boolean functions in network flow (draft version). Technical report, November 2003.

[14] S. Roman. *Coding and Information Theory*, chapter 6. 134 Springer-Verlag, 1992.

[15] P Sanders, S Egner, and L Tolhuizen. Polynomial time algorithms for network information flow. Technical report, October 2002.

[16] J.S Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing Surveys*, 33(2):209–271, June 2001.

[17] R W Yeung. Multilevel diversity coding with distortion. *IEEE Trans. Inform. Theory*, 41:412–422, March 1995.