Problems in Network Coding and Error Correcting Codes Appended by a Draft Version of S. Riis "Utilising Public Information in Network Coding"

S. Riis and R. Ahlswede

1 Introduction

In most of todays information networks messages are send in packets of information that is not modified or mixed with the content of other packets during transmission. This holds on macro level (e.g. the internet, wireless communications) as well as on micro level (e.g. communication within processors, communication between a processor and external devises).

Today messages in wireless communication are sent in a manner where each active communication channel carries exactly one "conversation". This approach can be improved considerably by a cleverly designed but sometimes rather complicated channel sharing scheme (network coding). The approach is very new and is still in its pioneering phase. Worldwide only a handful of papers in network coding were published year 2001 or before, 8 papers in 2002, 23 papers in 2003 and over 25 papers already in the first half of 2004; (according to the database developed by R. Koetters). The first conference on Network Coding and applications is scheduled for Trento, Italy April 2005. Research into network coding is growing fast, and Microsoft, IBM and other companies have research teams who are researching this new field. A few American universities (Princeton, MIT, Caltech and Berkeley) have also established research groups in network coding.

The holy grail in network coding is to plan and organize (in an automated fashion) network flow (that is allowed to utilise network coding) in a feasible manner. With a few recent exceptions [5] most current research does not yet address this difficult problem.

The main contribution of this paper is to provide new links between Network Coding and combinatorics. In this paper we will elaborate on some remarks in [8], [9]. We will show that the task of designing efficient strategies for information network flow (network coding) is closely linked to designing error correcting codes. This link is surprising since it appears even in networks where transmission mistakes never happen! Recall that traditionally error correction, is mainly used to reconstruct messages that have been scrambled due to unknown (random) errors. Thus error correcting codes can be used to solve network flow problems even in a setting where errors are assumed to be insignificant or irrelevant.

Our paper is the first paper that use error correcting codes when channels are assumed to be error-free. The idea of linking Network Coding and Error Correcting Codes when channels are not error-free was already presented in [4].

R. Ahlswede et al. (Eds.): Information Transfer and Combinatorics, LNCS 4123, pp. 861–897, 2006.

In this paper Cai and Yeung obtained network generalizations of the Hamming bound, the Gilbert-Varshamov bound, as well as the singleton bound for classical error-correcting codes.

2 The Basic Idea and Its Link to Work by Euler

The aim of the section is to illustrate some of the basic ideas in network coding. To illustrate the richness of these ideas we will show that solving the flow problem for certain simple networks, mathematically is equivalent to a problem that puzzled Euler and was first solved fully almost 200 years later! First consider the network in figure 1.

The task is to send the message x from the upper left node, to the lower right node labelled $r : x$ (indicating that the node is required to receive x) as well as to send the message y from the upper right node, to the lower left node labelled $r : y$. Suppose the messages belong to a finite alphabet $A = \{1, 2, \ldots, n\}$. If the two messages are sent as in ordinary routing (as used on the world wide web or in an ordinary wireless network) there is a dead lock along the middle channel where message x and message y will clash. If instead we send the message $s_{x,y} = S(x, y) \in A$ through the middle channel, it is not hard to show that the problem is solvable if and only if the matrix $(s_{i,j})_{i,j\in A}$ forms a latin square (recall that a latin square of order n is an $n \times n$ matrix with entries $1, 2, \ldots, n$ appearing exactly once in each row and in each column). We can now link this observation to work by Euler! Consider the extension of the previous flow problem in figure 2.

Now the task is to send the message x and the message y to each of the five nodes at the bottom. To do this each of the matrices $\{s_{x,y}\}$ and $\{t_{x,y}\}$ must, according to the previous observation, be latin squares. However, the latin squares must also be orthogonal i.e. if we are given the value $s \in A$ of the entry $s_{x,y}$ and the value $t \in A$ of the entry $t_{x,y}$, the values of x and y must be uniquely determined. Thus, we notice that:

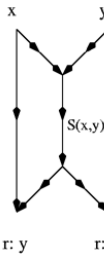

Fig. 1.

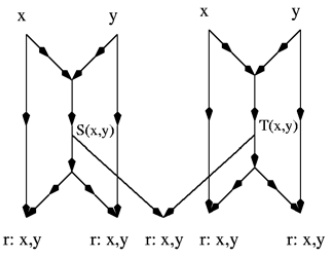

Fig. 2.

Proposition 1. *There is a one-to-one correspondence between solutions to the flow problem in figure 2 with alphabet A and pairs of orthogonal latin squares of order* $|A|$.

The problem of deciding when there exist such two orthogonal latin squares has an interesting history. Euler knew (c.1780) that there was no orthogonal

Latin square of order 2 and he knew constructions when n is odd or divisible by 4. Based on much experimentation, Euler conjectured that orthogonal Latin squares did not exist for orders of the form $4k + 2, k = 0, 1, 2, \ldots$. In 1901, Gaston Tarry proved (by exhaustive enumeration of the possible cases) that there are no pairs of orthogonal Latin squares of order 6 - adding evidence to Euler's conjecture. However, in 1959, Parker, Bose and Shrikhande were able to construct two orthogonal latin squares of order 10 and provided a construction for the remaining even values of n that are not divisible by 4 (of course, excepting $n = 2$ and $n = 6$). From this it follows:

Proposition 2 ((corollary to the solution to Euler's question)). *The flow problem in figure 2 has a solution if and only if the underlying alphabet does not have 2 or 6 elements.*

The flow problem in figure 2 might be considered somewhat 'unnatural' however the link to orthogonal latin squares is also valid for very natural families of networks. The multicast problem $N_{2,4,2}$ defined below has for example recently been shown to be essentially equivalent to Eulers question [6].

3 Network Coding and Its Links to Error Correcting Codes

The task of constructing orthogonal latin squares can be seen as a special case of constructing error correcting codes. There is, for example, a one-to-one correspondence between orthogonal latin squares of order $|A|$ and $(4, |A|2, 3)$ $|A|$-ary error correcting codes.[1]

Next consider the flow problem in figure 3.

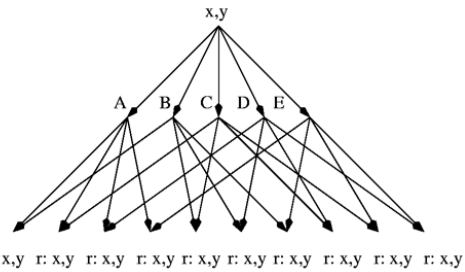

Fig. 3.

Assume each channel in this multi-cast network has the capacity to carry one message (pr. unit time). Assume that the task is to send two messages

[1] Recall that a (n, c, d) r -ary error correcting code C consists of c words of length n over an alphabet containing r letters. The number d is the minimal hamming distance between distinct words $w, w' \in C$.

$x, y \in A$ from the top nodes to each of the 10 bottom nodes. It can be shown that this flow problem has a solution over the alphabet A if and only if there exist an $(5, |A|^2, 4)$ $|A|$-ary error correcting code. It has been shown that there exit such codes if and only if $|A| \notin \{2, 3, 6\}$. The flow-problem in figure 3 can be generalized. Consider a network $N_{k,r,s}$ such that it consists of k messages $x_1, x_2, \ldots, x_k \in A$, that are transmitted from a source node. The source node is connected to a layer containing r nodes, and for each s element subset of r (there are $\binom{r}{s} = \frac{r!}{(r-s)!r!}$ such) we have a terminal node. The task is to insure that each message $x_1, x_2, \ldots, x_k \in A$ can be reconstructed in each of the terminal nodes. Notice the previous network flow problem is $N_{2,5,2}$. In general it can be shown [9], [8]:

Proposition 3. *The flow problem $N_{k,r,s}$ has a solution if and only if there exists an $(r, |A|k, r - s + 1)$ $|A|$-ary error correcting code.*[2]

Essentially, there is a one-to-one correspondence between solutions to the network flow problem $N_{2,4,2}$ and $(4, 4, 3)$ binary error correcting codes, i.e. orthogonal latin squares. Thus despite of the fact that the flow problem in figure 2 has a topology very different from the $N_{2,4,2}$ problem, the two problems essentially have the same solutions!

Next, consider the famous Nordstrom-Robinson code: This code is now known to be the unique binary code of length 16, minimal distance 6 containing 256 words. The point about this code is that it is non-linear, and is the only $(16, 256, 6)$ binary code. Again we can apply the proposition to show that the multi-cast problem $N_{8,16,11}$ has no linear solution over the field F_2, while it has a non-linear solution. Are phenomena like this just rare isolated incidences or much more widespread?

4 The Classical Theory for Error Correcting Needs Extensions

The previous sections indicate how it is possible to recast and translate network flow problems into the theory of error correcting codes (thus, using standard results in coding theory, it is possible to translate network flow problems into questions about finite geometries). Another approach is outlined in [7].

In [9], [8] the first example with only non-linear solutions was constructed. Unlike other examples this construction seems to go beyond standard results in error correcting codes. The construction is based on the network in figure 4. The network N in figure 4 has the curious property (like $N_{8,16,11}$) that the maximal through-put can only be achieved if non-linear flows are allowed (i.e non-linear boolean functions are needed in any solution). Furthermore it turns out that any code optimizing the vertical flows has to be a "minimal distance code" [9], [8]. This phenomena is interesting since a minimal distance code from a traditional perspective is very bad (as it essentially has the worst possible error correcting

[2] The fact that known bounds on maximum distance separable codes can be applied to bound the required alphabet-size was shown in [10].

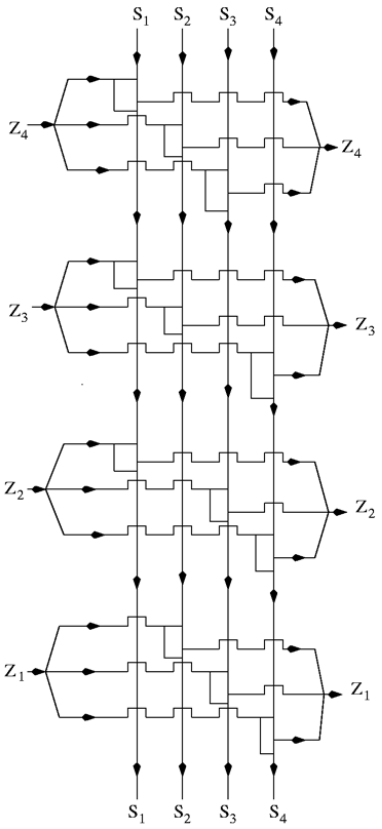

Fig. 4.

capability). This example is one of a collection of examples that suggests that the classical theory of error correcting codes needs to be extended and developed in order to serve as a basis for network coding. See also [3], [1], [2] more results pointing in this direction.

References

1. R. Ahlswede, Remarks on Shannon's secrecy systems, Probl. of Control and Inf. Theory, 11, 4, 301308, 1982.
2. R. Ahlswede and G. Dueck, Bad codes are good ciphers, Probl. of Control and Inf. Theory, 11, 5, 337351, 1982.
3. R Ahlswede and L.H Khachatrian, The diametric theorem in hamming spaces optimal anticodes, Advances in Applied Mathematics, 20, 429 449, 1996.
4. N. Cai and R.W. Yeung, Network coding and error correction, in ITW 2002 Bangalore, 119122, 2002.
5. S. Deb, C. Choute, M. Medard, and R. Koetter, Data harvesting: A random coding approach to rapid dissemination and efficient storage of data, in INFOCOM, submitted.

6. R. Dougherty, C. Freiling, and K. Zeger, Linearity and solvability in multicast networks, in Proceeding of CISS, 2004.
7. C. Fragouli and E. Soljanin, A connection between network coding and convolutional codes, in IEEE International Conference on Communications, 2004.
8. S. Riis, Linear versus non-linear boolean functions in network flow, in Proceeding of CISS 2004.
9. S Riis, Linear versus non-linear boolean functions in network flow (draft version), Technical report, November 2003.
10. M. Tavory, A. Feder, and D. Ron, Bounds on linear codes for network multicast, Technical Report 33, Electronic Colloquium on Computational Complexity, 2003.

Appendix

Utilising Public Information in Network Coding Draft Version

S. Riis

Abstract. We show that an information network flow problem N in which n messages have to be sent to n destination nodes has a solution (that might utilize Network Coding) if and only if the directed graph G_N (that appears by identifying each output node with its corresponding input node) has *guessing number* $\geq n$. The *guessing number* of a (directed) graph G is a new concept defined in terms of a simple cooperative game. We generalize this result so it applies to general information flow networks.

We notice that the theoretical advantage of Network Coding is as high as one could have possibly hoped for: for each $n \in N$ we define a network flow problem N_n with n input nodes and n output nodes for which the optimal through-put using Network Coding is n times as large as what can be obtained by vector routing or any other technique that does not allow interference (between messages) . In the paper we obtain a characterisation of the set of solutions for each flow problem N_n.

1 Network Coding

1.1 A. The Wave Approach to Information Network Flow

In recent years a new area called Network Coding has evolved. Like many fundamental concepts, Network Coding is based on a simple mathematical model of network flow and communication first explicitly stated in its simplicity in [3]. Recently, ideas related to Network Coding have been proposed in a number of distinct areas of Computer Science (e.g. broadcasting in wireless networks [25,**?**,23], data security [4], distributed network storage [6,**?**] and wireless sensor networks

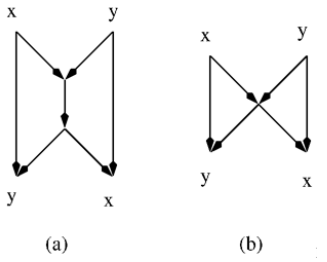

Fig. 1.

[15]). Network Coding has also a broad interface with various Mathematical disciplines (error correcting codes [18,5,10], circuit complexity [17], information theory [11], algebra [13,12] and graph theory).

The basic idea underlying Network Coding has been explained in numerous papers e.g. [13,3,17,7].The idea can be illustrated by considering the "butterfly" network in figure 1a.

The task is to send the message x from the upper left corner to the lower right corner and to send the message y from the upper right corner to the lower left corner. The messages $x, y \in A$ are selected from some finite alphabet A. Assume each information channel can carry at most one message at a time. If the messages x and y are sent simultaneously there is a bottleneck in the middle information channel. On the other hand if we, for example, send $x \oplus y \in A$ through the middle channel, the messages x and y can be recovered at 'output' nodes at the bottom of the network.

The network in figure 1a can be represented as the network in figure 1b. In this representation (which we will call the 'circuit representation') each node in the network computes a function $f : A \times A \to A$ of its inputs, and sends the function value along *each* outgoing edge. Historically, it is interesting to note that in this slightly different (but mathematically equivalent) form, the idea behind Network Coding (i.e. the power of using non-trivial boolean functions rather than "pushing bit") was acknowledged already in the 70s (though never emphasized or highlighted) in research papers in Circuit Complexity (see e.g. [22,20,16,21,2]). It is also worth mentioning that in Complexity Theory many lower bounds are proved under the assumption that the algorithm is *conservative* or can be treated as such. Conservative means that the input elements of the algorithm are atomic unchangeable elements that can be compared or copied but can not be used to synthesize new elements during the course of the algorithm. From a perspective of Circuit Complexity, Network Coding is an interesting theory of information flows since it corresponds to unrestricted models of computation.

Information flow in networks falls naturally within a number of distinct paradigms. Information flow can, for example, be treated in a fashion similar to traffic of cars in a road system. In this view each message is treated as **a packet** (e.g. a car) with a certain destination. Messages (cars!) cannot be copied, or divided. This way of treating messages is almost universally adopted in today's

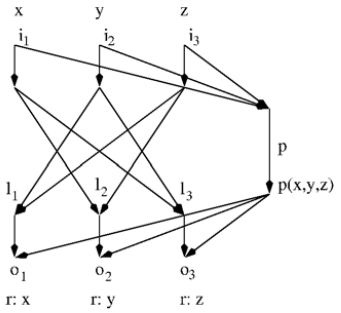

Fig. 2.

information networks (e.g. wireless communication, communication on the web, communication within processors or communication between processors and external devices). Another, less used possibility, is to treat messages in some sense as **a liquid** that can be divided and sent along different routes before they reach their destination. This approach (like, for example, in vector routing [9]) allows messages to be spread out and distributed over large parts of the network. Another and more radical approach is to treat messages as **"waves"**. Recall that the signals carrying the messages are digital (discrete) and thus certainly do not behave like waves. It is, however, possible to transmit and handle the digital (discrete) signals in a fashion where the messages (not the bits carrying the messages) behave like waves subject to interference and super position. More specifically, assume A is a (finite) alphabet of distinct (wave) signals that can be sent through a channel. The superposition of (wave) signals $w_1, w_2 \in A$ creates a new (wave) signal $w = w_1 \oplus w_2 \in A$. Thus mathematically, in the wave picture the set A of wave signals forms a (finite) commutative group with neutral element $0 \in A$ representing the zero-signal.

The network in figure 2 illustrates the point that in specific network topologies there can be quite a large advantage of treating messages as waves. The task of the network is to send messages x, y and z from the source (input) nodes i_1, i_2 and i_3 to the three output nodes o_1, o_2 and o_3. The receiver (output) node o_1 requires x, node o_2 requires y and node o_3 requires z. We assume that channels are one way and that the messages are only sent downwards in the figure. All crossings in the figure are 'bridges' and it is, for example, only possible to move from i_1 to o_1 by moving through channel p.

If messages are treated as packets (cars) like in traditional routing, or if messages are treated as a liquid, there is no point in sending information through l_1, l_2 or l_3. All messages x, y and z must pass through the channel labelled with p (for 'public'). This clearly creates a bottleneck in channel p if we assume that only one message can pass at a time.

If, however, messages are treated as waves we can send $p(x, y, z) := x \oplus y \oplus z$, the superposition of the messages x, y and z, through channel p. And we can send superpositions $l_1 := -(y \oplus z)$, $l_2 := -(x \oplus z)$ and $l_3 := -(x \oplus y)$ through the nodes with these labels. Node o_1 can take the superposition of l_1 and $p(x, y, z)$

and then reconstruct the message x $= -(y \oplus z) \oplus (x \oplus y \oplus z)$. Similarly, node o_2 (or o_3) can take the superposition of l_2 (or l_3) and $p(x, y, z)$ and then reconstruct the message $y = -(x \oplus z) \oplus (x \oplus y \oplus z)$ (or $z = -(x \oplus y) \oplus (x \oplus y \oplus z)$. This shows that the wave approach allows us to eliminate the bottleneck in channel p in figure 2. Notice also that the wave approach increases the overall network performance (of the network in figure 1) by a factor 3. [3]

In general the advantage of the wave approach (compared to any approach that does not allow interference) can be as large as one could have possibly hoped for. We will later notice that there exists information flow networks (with n source nodes and n receiver nodes) for which the optimal throughput is n times larger using the wave approach. Actually, there are even networks where the success rate for each active channel using the wave approach are close (as close as we wish) to n times the success rate for each active channel in a routing solution. The wave approach usually requires more information channels to be involved than traditional routing (or other methods that do not allow interference). Yet, by allowing interference, the total network performance divided by number of active information channels can for some network topologies be close to n times higher than any approach that is unable to utilise interference.

Network Coding allows messages to be sent within the wave paradigm. In fact superpositioning of signals (described above) represents an important type of Network Coding we will refer to as *Linear Network Coding* (see also [14]). Although Linear Network Coding represents a very important subclass of network coding, in general network coding involves methods that go beyond linear network coding. Certain network problems have no linear solutions, but require the application of non-linear boolean functions [17,7]. Non-Linear Network Coding has no obvious physical analogue. Rather general Network Coding represents a paradigm of information flow based on a mathematical model where 'everything goes' and where there are no a priory restrictions on how information is treated. Thus in Network Coding packets might be copied, opened and mixed. And sets of packets might be subject to highly complex non-linear boolean transformations.

2 Coherence: Utilising Apparently Useless Information

2.1 A. Guessing Game with Dice

While I was researching various flow problems related to Circuit Complexity it became clear that a key problem is to characterize and formalism what pieces of information are "useful" and what pieces of information are genuinely "useless" . It became clear that this distinction can be very deceptive. A piece of information that is useless in one context, can sometime be very valuable in a slightly different context [17].

[3] Notice that this increase of a factor 3 comes at a certain expense. In the routing approach only 7 channels are active (namely, $(i_1, p), (i_2, p), (i_3, p), (p, o_1), (p, o_2), (p, o_3)$ and channel p), while in the Network Coding solution all 19 channels are active. The success rate $\frac{3}{19}$ for each active channel is higher in the Network Coding solution than in the ordinary solution $\frac{1}{7}$.

To illustrate the problem, consider the following situation [19]: Assume that n players each has a fair s-sided dice (each dice has its sides labelled as $1, 2, \ldots, s$). Imagine that all players (simultaneously) throws their dice in such a manner that no player knows the value of their own dice.

1. What is the probability that each of the n players is able to guess correctly the value of their own dice?
2. Assume that each player knows the values of all other dice, but has no information about the value of their own dice. What is the probability that each of the n players correctly guesses the value of their own dice? (**Hint:** *The probability is NOT* $(\frac{1}{s})^n$*- The players can do much better than uncoordinated guessing!!*)
3. Assume the i^{th} player receives a value $v_i = v_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \{1, 2, \ldots, s\}$ that is allowed to depend on all dice values except the i'th player's own dice. What is the probability that each of the n players correctly manages to guess the value of their own dice?

In question 1 the probability that each player is right is $\frac{1}{s}$ and thus with probability $(\frac{1}{s})^n$ all n players successfully manage to guess correctly their own dice' value simultaneously. Maybe somewhat surprisingly in question 2, the answer depends on the 'protocol' adopted by the players! An optimal protocol appears, for example, if the players agree in advance to assume that the sum of all n dice' values is divisible by s. *This protocol ensures that all players simultaneously 'guess' the value of their own dice with probability* $\frac{1}{s}$.

Question 3, can be answered using a minor modification of the protocol just discussed. Let v_i be defined as the sum $x_1 \oplus x_2 \oplus \ldots, \oplus x_{i-1} \oplus x_{i+1} \oplus \ldots \oplus x_n$ modulo s. Each player then 'guesses' that $x_i = -v_i$ modulo s. Again, the probability that all n players simultaneously guess the correct value of their own dice is $\frac{1}{s}$.

2.2 B. Playing the Guessing Game on a Graph

We will now define a generalisation of the dice guessing game that is (surprisingly?) directly related to a certain type (the so called multiple-unicast type) of information flow problems.

Definition

The Guessing Game (G, s) is a cooperative game defined as follows: Assume that we are given a directed graph $G = (V, E)$ on a vertex set $V = \{1, 2, \ldots, n\}$ representing n players. Each player $v \in \{1, 2, \ldots, n\}$ sends the value of their dice $\in \{1, 2, \ldots, s\}$ to each player $w \in \{1, 2, \ldots, n\}$ with $(v, w) \in E$. Or in other words each node w receives dice' values from a set $A_w := \{v \in V : (v, w) \in E\}$. Each player has to guess the value of their own die. We want to calculate (assuming the players in advance have agreed on an optimal protocol) the probability that all the players (nodes) simultaneously guess their dice values. Question 2 (in Section 1 (A)) corresponded to the case where G is the complete graph on n nodes.

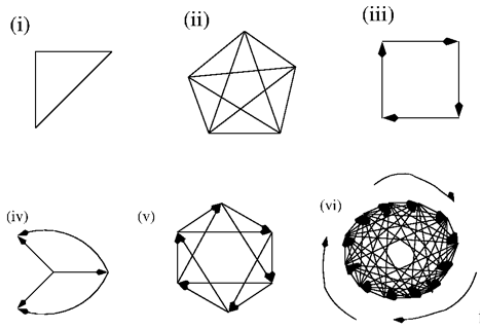

Fig. 3.

Definition

A (cooperative) guessing strategy for the Guessing Game (G, s) is a set of functions $f_\omega : \{1, 2, \ldots, s\}^{A_w} \to \{1, 2, \ldots, s\}$ with $\omega \in \{1, 2, \ldots, n\}$. Notice that each player (node) ω is assigned exactly one function f_ω.

In figure 3, we consider six simple examples:

In (i) and (ii) corresponds to the dice guessing game we already considered (with 3 and 5 players). The players have a guessing strategy that succeeds with probability $\frac{1}{s}$. In the guessing game based on (iii) (or in general the cyclic graph on n points) an optimal protocol appears if each node 'guesses' that its own dice value is the same as the value as it receives. This strategy succeeds if each of the four dice has the same value i.e. with probability $(\frac{1}{s})^3$ (or in general $(\frac{1}{s})^{n-1}$). Though this probability is low, it is s times higher than if the players just make uncoordinated random guesses.

In (iv) the graph contains no cycles so the players cannot do any better than just guessing i.e. the players can achieve probability at most $(\frac{1}{s})^4$.

In (v) it can be shown that there are a number of distinct guessing strategies that guarantee the players' success with probability $(\frac{1}{s})^4$ (one, optimal strategy appears by dividing the graph into two disjoint cycles (triangles)).

Finally, in (vi) we consider a graph with 12 nodes (one for each hour on a clock) and edges from (i, j) if the 'time' from i to j is at most 5 hours. We will show (and this will be fairly simple given the general methods we develop) that the players in the Guessing Game (G, s) have an optimal guessing strategy that ensures that the players with probability $(\frac{1}{s})^7$ (i.e. with a factor s^5 better than pure uncoordinated guessing) all simultanously guess the value of their own dice.

Definition

A graph $G = (V, E)$ has for $s \in N$ *guessing number* $k = k(G, s)$ if the players in the Guessing Game (G, s) can choose a protocol that guarantees success with probability $(\frac{1}{s})^{|V|-k}$.

Thus the guessing number of a directed graph is a measure of how much better than pure guessing the players can perform. If the players can achieve a factor s^k better than pure random uncoordinated guessing, the graph has guessing

number $k = k(G, s)$. Notice that a directed graph has a guessing number for each $s = 2, 3, 4, \ldots$.

For many directed graphs (though not all) the guessing number is independent of s. The directed graphs in figure 3 have guessing numbers $4, 2, 1, 0, 2$ and 5 (independently of $s \geq 2$). From the definition there is no reason to believe that the guessing number of a directed graph is in general an integer. Yet remarkably many graphs have integer guessing numbers. Later we will show that there exist directed graphs for which the guessing number $k = k(G, s)$ (for alphabet of size $s \in N$) of a graph is not an integer. We will show that there exist graphs where the guessing number $k(G, s)$ even fails to be an integer for each $s \in \{2, 3, 4, \ldots, \}$.

Observation(A)

In the Guessing Game (G, s) the graph G allows the players to do better than pure uncoordinated guessing if and only if G contains a cycle.

Observation(B)

A graph $G = (V, E)$ contains a cycle if and only if its guessing number is ≥ 1. If a graph contains k disjoint cycles its guessing number $\geq k$ (for each $s \geq 2$). A graph is reflexive if and only it has guessing number $|V|$. Assume that the set of nodes V in the graph G can be divided in r disjoint subsets $V_1, V_2, \ldots, V_r$ of nodes such that the restriction of G to each subset V_j is a clique. Then the graph G has guessing number $\geq |V| - r$ (for each $s \geq 2$).

From Observation (A), we notice the curious fact that, *the players have a "good" strategy that ensures that they all succeed with higher probability than uncoordinated random guessing if and only if the players have a "bad" strategy that insures they never succeed.*

Sometimes it is convenient to focus on certain more limited guessing strategies.

Definition Let B be a class of functions $f : A^d \to A$ for $d = 1, 2, 3, \ldots$. An important class appears if we let A denote a fixed algebraic structure (e.g. a group, a ring or a vector space) of $s = |A|$ elements, and let the class $B = LIN$ consist of all homomorphisms (linear maps) $A^d \to A$ for $d = 1, 2, 3, \ldots$. If all the functions f_w belong to the class B we say the players have chosen a guessing strategy in B. If $B = LIN$ we say that the players use a linear guessing strategy.

Definition A graph $G = (V, E)$ has *guessing number* $k = k_B(G, s)$ with respect to the functions in B if the players in the Guessing Game (G, s) have a protocol with all guessing functions in B that guarantee success with probability $(\frac{1}{s})^{|V|-k}$. We say G has (special) linear guessing number $k_{lin} = k_{lin}(G, s)$ if the players have a linear guessing strategy that guarantee success with probability $\geq (\frac{1}{s})^{|V|-k}$.

3 Network Coding and Guessing Games

In this section we show that mathematically there is a very close link between Network Coding and the guessing games we just defined. We will show that each information flow problem is equivalent to a problem about directed graphs.

The translation between information networks and directed graphs is most clean if we represent information networks such that we place all computations (Network Codings) in the nodes of the network. As already indicated, we refer to this representation as the Circuit representation. This representation is slightly more economical (usually save a few nodes) than the standard representation in Network Coding. The representation is more in line with circuit complexity, where the task of the network in general is a computational task. Formally, each source node is associated with a variable. Each node computes a function of incoming edges signals. Each outgoing edge from a node transmits the same signal (function value of node). Each receiver node is required to produce a specific input variable.

In general given an information flow problem N (in the Circuit representation) we obtain a directed graph G_N by identifying each source node with the corresponding receiver node.

In figure 4 we see a few examples of simple information networks together with their corresponding directed graphs.

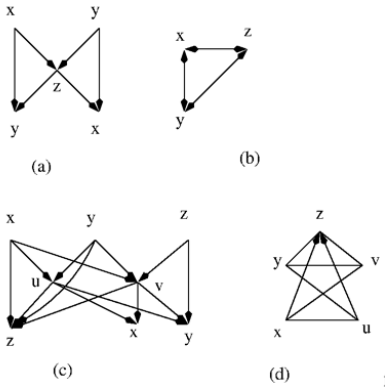

Fig. 4.

The information network N in figure 4a (or figure 1b) is the usual 'butterfly' network (presented in circuit represention). If we identify the input node (source node) x with the output node (receiver node) x, and identify input node (source node) y with the output node (receiver node) y, we get the graph in figure 4b.

The information network in figure 4c does not have any obvious symmetries, but when input and output nodes are identified we get the directed graph in figure 4d that clearly contains a number of symmetries. The translation shows that nodes x and u (as well as y and v) are equivalent points. The guessing number of the graph in (b) as well as the graph in (d) can be shown to have the value 2.

In general we let $C_{multiple-unicast}$ (the class of multiple-unicast directed information networks) consist of information networks N for which for some $n \in N$, n messages $m_1, m_2, \ldots, m_n \in A$ (selected from some alphabet A) has to be sent

from input (source) nodes $i_1, i_2, \ldots, i_n$ to output nodes $o_1, o_2, \ldots, o_n$. Somewhat formally, to each source node i_j is associated a variable x_j and each node w (except the source nodes) are assigned a function symbol f_w representing a function f_w that is mapping all incoming signals $a_1, a_2, \ldots, a_{k_w}$ to an element $a = f(a_1, a_2, \ldots, a_{k_w}) \in A$. Each outgoing edge from a node transmits the same signal (the function value a of the node). Each receiver node is required to produce a specific input variable.

For an information network $N \in C_{multiple-unicast}$ we associate a directed graph G_N that appears by identifying each source (input) node i_j in N with its corresponding receiver (output) node o_j. If N has n input nodes, n output nodes and m inner nodes ($2n + m$ nodes in total) the graph G_N has $n + m$ nodes.

We are now ready to state the surprising link that shows that each information flow problem is equivalent to a problem about directed graphs.

Theorem 1

An information Network flow problem $N \in C_{multiple-unicast}$ with n input/output nodes has a solution over alphabet A with $|A| = s$ elements if and only if the graph G_N has guessing number $\geq n$.

The main point of the theorem is that it replaces the flow problem - a problem that mathematically speaking involves slightly complicated concepts like *set of source nodes*, *set of receiver nodes* as well as set *of requirements (demands)* that specifies the destination of each input - with an equivalent problem that can be expressed in pure graph theoretic terms (no special input or output nodes). Actually we show the theorem in a slightly stronger form:

Theorem 2

The solutions (over alphabet A with $|A| = s$) of an information network flow problem $N \in C_{multiple-unicast}$ with n input/output nodes are in one-to-one correspondence with the guessing strategies (over alphabet A with $|A| = s$) that ensure that the players in the guessing game played on G_N have success with probability $(\frac{1}{s})^{|G_N|-n}$ (where $|G_N|$ is the number of nodes in G_N).

The following simple observation highlights (in a quite geometric fashion) the difference between Network coding and traditional routing:

Observation(C)

An information flow network $N \in C$ has through put k using ordinary routing (i.e. pushing each message along a unique path) if and only the graph G_N contains k disjoint cycles.

Consider the three information flow problems in figure 5(i-iii). They are in circuit representation (i.e. all functions are placed in their nodes and each outgoing edge from a node transmits the same function value). The three information networks in 5(i)-(iii) are non-isomorphic and are clearly distinct. However if we identify the sauce nodes and the receiver nodes in each of the networks we get the *same* directed graph in figure 5 (iv).

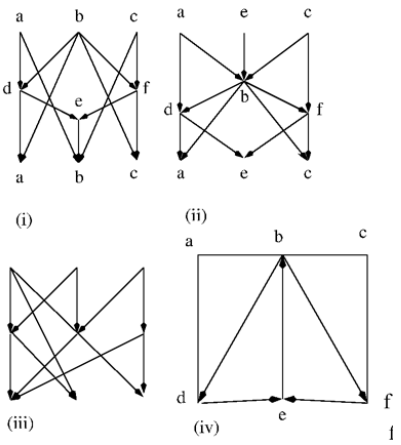

Fig. 5.

According to Theorem 2 there is a one-to-one correspondence between solutions of each of the three information networks 5(i)-5(iii) and the successful strategies in the Guessing Game (G, s). Thus, the set of solutions to each of the three information networks 5(i)-5(iii) are in a natural one-to-one correspondence. Before we prove Theorem 1 and Theorem 2, let us have a closer look at the networks in figure 5. A (cooperative) strategy for the players in the guessing game with the directed graph in figure 5 (iv) consists of 6 functions $g_1, g_2, \ldots, g_6$ such that:

$$
\begin{aligned}
a^{guess} &= g_1(b, d) \\
b^{guess} &= g_2(a, c, e) \\
c^{guess} &= g_3(b, f) \\
d^{guess} &= g_4(a, b) \\
e^{guess} &= g_5(d, f) \\
f^{guess} &= g_6(b, c)
\end{aligned}
$$

For all players to guess their own message correctly we must have $a^{guess} = a$ i.e. we must have $a = g_1(b, d)$. Thus assuming that we work under the conditional situation with $a^{guess} = a$, we can substitute a with $g_1(b, d)$ leading to the equations:

$$
\begin{aligned}
b^{guess} &= g_2(g_1(b, d), c, e) \\
c^{guess} &= g_3(b, f) \\
d^{guess} &= g_4(g_1(b, d), b) \\
e^{guess} &= g_5(d, f) \\
f^{guess} &= g_6(b, c)
\end{aligned}
$$

Now pick any equation of the form $x^{guess} = h$ where x does not appear in the expression h. We might for example assume $c = g_3(b, f)$ (i.e. the $c^{guess} = c$). Substituting c with $g_3(b, f)$ in the equations we get:

$b^{guess} = g_2(g_1(b,d), g_3(b,f), e)$
$d^{guess} = g_4(g_1(b,d), b)$
$e^{guess} = g_5(d,f)$
$f^{guess} = g_6(b, g_3(b,f)$
This system of equations contains still one equation of the form $x^{guess} = h$ where x does not appear in the expression h. Let $e = g_5(d,f)$ (assuming $e^{guess} = g_5(d,f)$) and substitute this into the equations we get:
$b^{guess} = g_2(g_1(b,d), g_3(b,f), g_5(d,f))$
$d^{guess} = g_4(g_1(b,d), b)$
$f^{guess} = g_6(b, g_3(b,f))$
For any fixed choice of functions g_1, g_2, g_3, g_4, g_5 and g_6 let $0 \leq p \leq 1$ denote the probability that a random choice of b, d and f satisfies the equations:
$b = g_2(g_1(b,d), g_3(b,f), g_(d,f))$
$d = g_4(g_1(b,d), b)$
$f = g_6(b, g_3(b,f))$
It is not hard to show that the probability that $a^{guess} = a$, $c^{guess} = c$ and $e^{guess} = e$ with probability $(\frac{1}{s})^3$ (for a more general argument see the proof of Theorem 2). Thus the conditional probability that the remaining players all guess correctly their own dice value is p and the probability that all players are correct is $p(\frac{1}{s})^3$. Thus - in agreement with Theorem 1 - the guessing number of the graph in figure 4 (iv) is 3 if and only if there exist functions $g_1, g_2, \ldots, g_6$ such that the equations hold for all b, d and f (i.e. hold with probability 1).

As it happens we can solve the equations by turning the alphabet A into a commutative group $(A, \oplus)$ and the by letting $g_1(b,d) = b \oplus d$, $g_2(\alpha, \beta, \gamma) = \alpha \ominus \gamma$, $g_3(b,f) = b \oplus f$, $g_4(\alpha, \beta) = \alpha \ominus \beta$, $g_5(d,f) = d$ and $g_6(\alpha, \beta) = \ominus\alpha \oplus \beta$. Thus the players have a (cooperative) guessing strategy that ensures that all players simultaneously are able to guess their own message correctly with probability $(\frac{1}{s})^3$. One strategy is given by:
$a^{guess} = b \oplus d$
$b^{guess} = a \ominus e$
$c^{guess} = b \oplus f$
$d^{guess} = a \ominus b$
$e^{guess} = d$
$f^{guess} = c \ominus b$
Figure 6 (i)-(iii) shows how this strategy naturally corresponds to network codings in the three information flow problems in figure 5(i)-(iii). Figure 6 (iv) shows the strategy as a guessing strategy.

4 Proof of Theorems

Before we prove Theorem 1 and Theorem 2 we need a few formal definitions of information networks. As already pointed out, the translation between information networks and directed graphs is most clean if we represent information networks such that we place all computations (Network Codings) in the nodes of the network. An information flow network N (in circuit representation) is an

acyclic directed graph with all source nodes (input nodes) having in-degree 0 and all receiver nodes (output nodes) having outdegree 0. Each source node is associated with a variable from a set Γ_{var} of variables. In the receiver node there is assigned a demand i.e. variable from Γ_{var}. In each node w that is not a source, there is assigned a function symbol f_w. The function symbols in the network are all distinct.

Messages are assumed to belong to an alphabet A. Sometimes we assume A has additional structure (e.g. a group, a ring or a vector space). Each outgoing edge from a node transmits the same signal (function value of node).

An actual information flow is given by letting each function symbol f represent an actual function $\tilde{f} : A^d \to A$ where d is the number of incoming edges to the node that is associated the function symbol f. The information flow is a solution, if the functions compose such that each demand always is met.

We let $C_{multiple-unicast}$ denote the class of information networks N for which for some $n \in N$, n messages $m_1, m_2, \ldots, m_n \in A$ (selected from some alphabet A) have to be sent from nodes $i_1, i_2, \ldots, i_n$ to output nodes $o_1, o_2, \ldots, o_n$.

Let $C_{multiple-unicast}$ be an information network in this model. We define the graph G_N by identifying node i_1 with o_1, node i_2 with o_2, ... and node i_j with o_j in general for $j = 1, 2, \ldots, n$.

Theorem 1 follows directly from Theorem 2. So to prove the theorems it suffices to prove Theorem 2.

Proof of Theorem 2: Let N be an information network with input (source) nodes $i_1, i_2, \ldots, i_n$, output (receiver) nodes $o_1, o_2, \ldots, o_n$ and inner nodes n_1, $n_2, \ldots, n_m$. The network N is acyclic so we can assume that we have ordered the nodes as $i_1 < i_2 < \ldots < i_n < n_1 < n_2 < \ldots < n_m < o_1 < o_2 < \ldots < o_n$ such that any edge (i, j) in N has $i < j$ in the ordering. Any selection of coding functions (whether they form a solution or not) can then be written as:

$$
\begin{aligned}
z_1 &= f_1(x_1, x_2, \ldots, x_n) \\
z_2 &= f_2(x_1, x_2, \ldots, x_n, z_1) \\
z_3 &= f_3(x_1, x_2, \ldots, x_n, z_1, z_2) \\
&\ldots\ldots\ldots\ldots \\
z_m &= f_m(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{m-1}) \\
x_1^o &= g_1(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m) \\
x_2^o &= g_2(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m) \\
&\ldots\ldots\ldots\ldots \\
x_n^o &= g_n(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m)
\end{aligned}
$$

where x_j is the variable denoting the value assigned to the input node i_j, z_j is the variable denoting the value computed by the inner node n_j and x_j^o is the variable denoting the output value computed by the node o_j for $j = 1, 2, \ldots, n$ for a given choice of values of $x_1, x_2, \ldots, x_n \in \{1, 2, \ldots, s\}$.

Next consider the corresponding graph G_N we get by identifying nodes i_r and o_r for $r = 1, 2, \ldots, n$. We consider the guessing strategy given by the functions above i.e. the strategy given by:

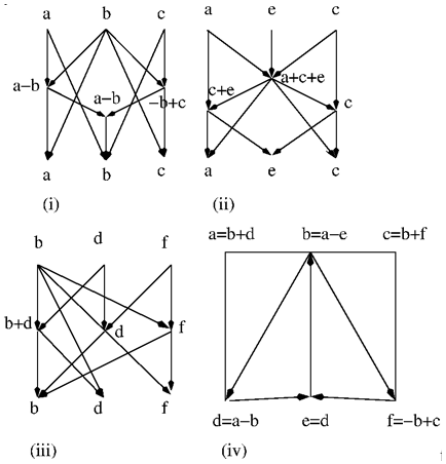

Fig. 6.

$$
\begin{aligned}
z_1^{guess} &= f_1(x_1^{real}, x_2^{real}, \ldots, x_n^{real}) \\
z_2^{guess} &= f_2(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}) \\
z_3^{guess} &= f_3(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}) \\
&\ldots\ldots\ldots\ldots \\
z_m^{guess} &= f_m(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_{m-1}^{real}) \\
x_1^{guess} &= g_1(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_m^{real}) \\
x_2^{guess} &= g_2(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_m^{real}) \\
&\ldots\ldots\ldots\ldots \\
x_n^{guess} &= g_n(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_m^{real})
\end{aligned}
$$

Conversely each guessing strategy for G_N can be written on this form and can thus be viewed as an attempt to solve the information flow problem N. To prove Theorem 2 we show that the guessing strategy succeeds with probability $(\frac{1}{s})^m$ if and only if the corresponding information flow functions solves the information Network problem. This boils down to showing that the probability that all inner nodes guess their own dice values correctly is $(\frac{1}{s})^m$. To see this assume we have shown this. Then the probability all players guess correctly is at most as large as the probability all players corresponding to inner nodes $n_1, n_2, \ldots, n_m$ guess correctly. Thus all the players guess simultaneously their own die values correctly with probability $\leq (\frac{1}{s})^m$. Equality holds if and only if the conditional probability x_j^{guess}, $j = 1, 2, \ldots, n$ takes the correct value with probability 1. This happens if and only if the functions $f_1, f_2, \ldots, f_m, g_1, \ldots, g_n$ form a solution for the information flow problem. So to complete the proof of Theorem 2 it suffices to show:

Lemma 3. For any set of functions $f_1, \ldots, f_m$ and $g_1, \ldots, g_n$ the probability that players $n_1, n_2, \ldots, n_m$ (i.e. players in nodes corresponding to inner nodes in the information Network) guess their own die values correctly is $(\frac{1}{s})^m$ (i.e. independent of the chosen guessing functions).

Proof: We are asking for the probability $z_j^{guess} = z_j^{real}$ for $j = 1, 2, \ldots, m$ where
$z_1^{guess} = f_1(x_1^{real}, x_2^{real}, \ldots, x_n^{real})$
$z_2^{guess} = f_2(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real})$
$z_3^{guess} = f_3(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real})$
.....
$z_m^{guess} = f_m(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_{m-1}^{real})$
The number of choices of $x_1^{real}, x_2^{real}, \ldots, x_n^{real}$ and $z_1^{real}, z_2^{real}, \ldots, z_m^{real}$ is s^{n+m}. We want to count the number of "successful" choices for which $z_j^{guess} = z_j^{real}$ for $j = 1, 2, \ldots, m$. That is the number of choices for which:
$z_1^{real} = f_1(x_1^{real}, x_2^{real}, \ldots, x_n^{real})$
$z_2^{real} = f_2(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real})$
$z_3^{real} = f_3(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real})$
.....
$z_m^{real} = f_m(x_1^{real}, x_2^{real}, \ldots, x_n^{real}, z_1^{real}, z_2^{real}, \ldots, z_{m-1}^{real})$
But for each choice of $x_1^{real}, x_2^{real}, \ldots, x_n^{real}$ there is exactly one choice of $z_1^{real}, z_2^{real}, \ldots, z_m^{real}$. Thus the number of successful choices is s^n. The probability is $\frac{\text{number of successful choices}}{\text{number of choices}} = \frac{s^n}{s^{n+m}} = \frac{1}{s^m}$. ♣

4.1 A. Standard Representation

There are a few slightly different ways to represent flows in information networks. In the previous section we considered the Circuit representation. We call the standard (and "correct") way of representing information flows in Network Coding the **Standard representation**. If we use the standard representation we get slightly different versions of Theorem 1 and Theorem 2. The actual theorems can be stated in the same way (verbatim)! The Theorems are modified to fit the standard representation in the way the graph G_N is defined.

An information network N is a directed acyclic multi-graph. Each source node has indegree 0, while each receiver node has outdegree 0. Associated with each source node is a variable from a set Γ_{var} of variables. Each outgoing edge is associated with a distinct function symbol with an argument for each incoming edge. Each receiver node has a list of demands which is a subset of variables from Γ_{var}. In the receiver node there is assigned a function symbol for each demand. All function symbols are distinct.

Messages are assumed to belong to an alphabet A. An actual flow (using Network Coding) is given by letting each function symbol f represents an actual function $\tilde{f} : A^d \to A$ where d is the number of incoming edges to the node that is associated with the function symbol f. The flow (that might utilise Network Coding) is a solution if the functions compose such that each demand is given by the functional expression of the involved terms.

We let $C_{multiple-unicast}$ denote the class of information networks N for which for some $n \in N$, n messages $m_1, m_2, \ldots, m_n \in A$ (selected from some alphabet A) has to be send from nodes $i_1, i_2, \ldots, i_n$ to output nodes $o_1, o_2, \ldots, o_n$.

We convert a given information network $N \in C_{multiple-unicast}$ to a directed graph G_N as follows:

Step 1: For each variable or function symbol assigned to an edge or a node we introduce a node in the new graph G_N.

Step 2: We identify nodes i_1 with o_1, i_2 with o_2, ... and i_j with o_j in general for $j = 1, 2, \ldots, n$.

With this translation of N to G_N Theorem 1 and Theorem 2 remain valid (verbatim).

5 General Results for Information Networks

Theorem 1 and Theorem 2 only apply for information networks $N \in C_{multiple-unicast}$. In this section we generalize the results so they essentially cover all (!) instantaneous information networks.

Let N be an information network and let A be an (finite) alphabet with s elements. For a selection of fixed network functions $\bar{f}$ we define the networks N's *global success rate* $p(N, s, \bar{f})$ of a specific network coding flow as the probability that *all* outputs produce the required outputs if all inputs are selected randomly with independent probability distribution. The *maximal global success rate* $p(N, s)$ of the information flow network N (over alphabet of size s) is defined as the supremum of all global success rates $p(N, s, \bar{f})$ that can be achieved by any choice of coding functions $\bar{f}$. Since the set of functions $\bar{f}$ (for a fixed finite alphabet A) is finite $p(N, s)$ is the maximal global success rate of $p(N, s, \bar{f})$.

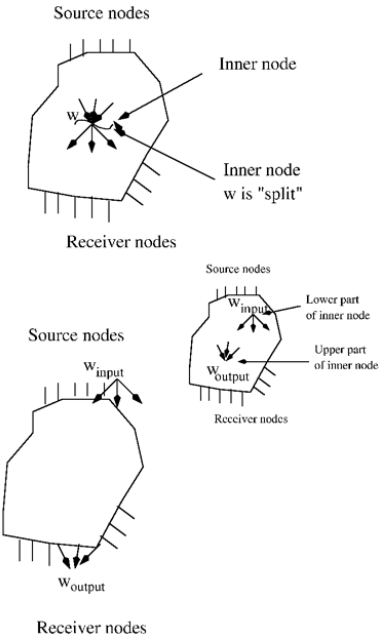

Fig. 7.

Assume that N is an information network over an alphabet A with s elements. Assume that N has n source nodes (input nodes) and that r of these are required by at least one receiver node (output node). Usually, $n = r$ since in most information networks each source message is required by at least one receiver node.

Definition

We define the *source transmission bandwidth* $k = k(N, s)$ of the information network N (over alphabet of size s) as $k(N, s) = \log_s(p(N, s)) + r$.

The notion is motivated by the Theorem 4 below, and can be viewed as a generalisation of the guessing number of a graph.

Notice, that a network has source transmission bandwidth k if all output nodes simultaneously can calculate their required messages with probability s^k higher that what can be achieved by the "channel free" network. An information network N that sends n distinct source messages (each message is required at one or more receiver nodes), has source transmission bandwidth $k(N, s) = n$ if and only if it has is solvable (in the sense of network coding) over an alphabet of size s.

For each directed graph $G = (V, E)$ we want to define an information flow problem $N_G = (W, F)$ with $|V|$ source nodes (input nodes) and $|V|$ receiver nodes (output nodes). Expressed slightly informally, we define N_G by splitting each node $w \in V$ into two nodes w_{input} and w_{output} (thus the vertex set W consists of two copies of V). For each edge $(w, v) \in E$ we add an edge $(w_{input}, v_{output}) \in F$. Let $N_G = (W, F)$ denote the flow problem that appears through this transformation where each output node v_{output} requires the message assigned to v_{input}. Notice that the information network N_G usually is very far from being solvable since most source (input) nodes have no path to its corresponding receiver (output) node.

Observation

Let G be a directed graph. Then $N_G \in C_{multiple-unicast}$ and G has guessing number $k = k(G, s)$ if and only if N_G has source transmission bandwidth $k = k(N_G, s)$.

For each $p \in [0, 1]$ there is a one-to-one correspondence between guessing strategies $\bar{f}$ in the Guessing Game (G, s) that achieve success with probability p and information flows $\bar{f}$ in N_G that have global success rate p.

The Observation is too trivial to deserve to be called a theorem. It is however quite interesting since it shows that the notion of source transmission bandwidth generalizes the guessing number of a directed graph.

We now introduce a simple move we call "split". Given an information network $N = (V, E)$ (with E being a multiset) the move "split" can be applied to any inner node $w \in V$ in N (a node is an inner node if it is not a source or a receiver node). The move "split" copy the inner node w into two nodes w_{input} and w_{output}. In other words the move convert the vertex set V to the set $V' = V \cup \{w_{input}, w_{output}\} \setminus \{w\}$ containing all point in V but with two copies (w_{input} and

w_{output}) of w. For each outgoing edge $(w, u) \in E$ from w we introduce an edge $(w_{input}, u) \in E'$ (with the same multiplicity as (w, u)). For each incoming edge $(u, w) \in V$ we introduce an edge $(u, w_{input}) \in E'$ (with the same multiplicity as (w, u)).

The information network $N' = (V', E')$ has as source (input) nodes all source (input) nodes in V together with $\{w_{input}\}$. The set of receiver (output) nodes consists of the receiver (output) nodes in V together with $\{w_{output}\}$. We associate a new variable z to the node w_{input} and node w_{output} demands (requires) z. All other nodes keep their demands.

In figure 7, we see how the split move can be applied. We say that the information network N' appear from the information network N by a "reverse split move", if N appears from N' using a split move.

The split move always result in an information network that have no solution (since there is no path from the source node w_{input} to the receiver node w_{output}).

The next Theorem can be considered as a generalization of Theorem 1 and Theorem 2.

Theorem 4

Let N and N' be two information networks that appear from each other by a sequence of split and inverse split moves (in any order). The network N and N' has the same source bandwidth (i.e. $k(N, s) = k(N', s)$)

More specifically let N be an information flow network, let A be an alphabet with $|A| = s$ letters and assume $\bar{f}$ is a selection of coding functions over this alphabet. Assume N has source messages $x_1, x_2, \ldots, x_r$ (they might be transmitted from more than one input edge). Assume that the coding functions have global success rate $p = p(N, s, \bar{f}) \in [0, 1]$. Let N' be any information network that appears from N by application of the split move. Then N' with the coding functions $\bar{f}$) has global success rate $p(N', s, \bar{f}) = \frac{p}{s}$.

In general if N has global success rate p (over alphabet A) any network N' that appears from N by application of r split moves and t reverse split moves (in any order) has global success rate $p \times s^{t-r}$.

Proof: The first part follows from the more detailed second part since each application of the split rule increase the value of r by one and each application of the inverse split rule decrease the value of r by one.

Assume that the information network $N = (V, E)$ has global success rate $p = p(N, s, \bar{f}) \in [0, 1]$ with respect to the coding functions $\bar{f}$. Let $w \in V$ be any inner node in N. Replace (split) w into two nodes w_{input} and w_{output} as already explained. The incoming coding function to node w_{output} is the same function as the inner coding function to node w in the network N. Each outgoing coding function of w_{input} is the same as each outgoing function for node w. The network N' has got a new input node. Let us calculate the probability $p(N', s, \bar{f})$ that all output nodes produce the correct outputs. The probability that node w_{output} produce the correct output is exactly $\frac{1}{s}$. Assume now $w_{output} = w_{input}$. The conditional probability (i.e. the probability given $z_{output} = z_{input}$) that all output nodes in the network N produce the correct output is $p = p(N, s, \bar{f})$.

But, then the probability all output nodes in N' produce the correct output is exactly $\frac{p}{s}$.

The second part of the theorem follows from the first part. Assume $\bar{f}$ is a selection of coding functions such that $p(N, s, \bar{f}) = p(N, s)$ (the alphabet is finite so there are only finitely many functions $\bar{f}$ and thus there exists functions that achieve the maximum value $p(N, s)$). We already showed that $p(N', s, \bar{f}) = \frac{p(N,s,)}{s}$. We claim that $p(N', s) = p(N', s, \bar{f})$. Assume that $p(N', s, \bar{g}) > p(N', s, \bar{f})$. But then $p(N, s, \bar{g}) = s \times p(N', s, \bar{g}) > s \times p(N', s, \bar{f}) = p(N, s, \bar{f})$ which contradicts the assumption that $\bar{f}$ was an optimal coding function for the information network N (over alphabet of size s). ♣

6 Utilising Public Information

6.1 A. Another Game

Consider the directed graph in figure 8(i) (introduced in 4 (iv)). Each node has to derive their own message. This is, of course, impossible and we know that the best the players can hope for (if they use a suitable coordinated guessing strategy) is that they are all correct on s^3 distinct inputs (out of the s^6 different input). If the players have access to s^3 public messages and these are carefully chosen, it is possible for the players (through a cooperative strategy) to ensure that each player can derive his/her own message.

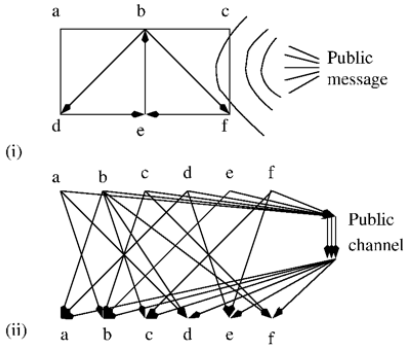

Fig. 8.

If, for example, the values $a \ominus b \ominus d \in A, c \ominus b \ominus f \in A$ as well as $e \ominus d \in A$ are common knowledge (broadcast through public messages) each node can derive its own message (since $a = (a \ominus b \ominus d) \oplus b \oplus d, b = (e \ominus d) \ominus (a \ominus b \ominus d) \oplus (a \ominus e), c = (c \ominus b \ominus f) \oplus b \oplus f, d = (a \ominus b) \oplus (a \ominus b \ominus d), e = (e \ominus d) \oplus d$ and $f = (c \ominus b) \ominus (c \ominus b \ominus f)$).

Another equivalent way of stating this is to consider the bipartite flow problem in figure 8 (ii), with public channel of bandwidth 3. Notice that figure 8 (i) and figure 8 (ii) are different representations of the problems that are mathematically equivalent.

Are the solutions (public messages $a \ominus b \ominus d \in A, c \ominus b \ominus f \in A$ as well as $e \ominus d \in A$) in figure 8 (i) and figure 8 (ii) optimal? Is it possible to send fewer than s^3 message through the public channel (and still have all players being able to deduce their own message)? From the analysis of the guessing game in figure 5 (iv) we know that the probability that the players in nodes a, c and e guess their own messages is independent (for any guessing strategy) and thus nodes a, c and e guess correctly their own message with probability $(\frac{1}{s})^3$. We claim that if node a, c and e in general are able to derive their own message they must have access to at least s^3 distinct messages in the public channel. To see this assume that it were possible for the players in figure 8 (i) to deduce their own messages from a public channel that sends $< s^3$. The players could then all agree to guess *if* the public channel is broadcasting a specific message m they agreed on in advance. Since there are less than s^3 public messages there is a message m that is broadcast with probability $> (\frac{1}{s})^3$). This contradicts the fact that the players (especially the players in nodes a, b and c) cannot do better than $(\frac{1}{s})^3$. Thus the solutions in figure 8 (i) (and in figure 8 (ii)) are optimal.

Let $G = (V, E)$ be a directed graph. Assume like before that each node is being assigned a message x randomly chosen from a fixed finite alphabet A containing $s = |A|$ elements. Like in the guessing game each node transmits its message (dice value) along all outgoing edges. In other words each node j knows the messages (dice values) of exactly all nodes i with $(i, j) \in E$.

The task of the players is to deduce their own message. This is of course impossible (unless the graph is reflexive) since in general the players have no direct access to their own message (dice values). The task of the players is to cooperate and agree on a protocol and a behaviour of a public channel that ensure that all players are always able to derive their own messages.

Definition

Let $G = (V, E)$ be a directed graph and let A denote an alphabet with s letters. Let P be a finite set of public messages. Consider the following Public Channel Game (G, A, P). The game is played as follows. Each node $j \in V$ is assigned to a message $x_j \in A$. A public message $p = p(x_1, x_2, \ldots, x_n) \in P$ (given by a function $p : A^n \to P$) is broadcast to all nodes. Each node j has access to the message $p \in P$ as well as x_i for each i with $(i, j) \in E$. In the game each player j needs to deduce the content of their own message x_j.

Each player (node) $v \in \{1, 2, \ldots, n\}$ sends its message to each person $w \in \{1, 2, \ldots, n\}$ with $(v, w) \in E$. Or in other words each node w receives messages from a set $A_w := \{v \in V : (v, w) \in E\}$. The task is to design the function $p(x_1, x_2, \ldots, x_n)$ such that each player always (i.e. for any choice of $x_1, x_2, \ldots, x_n \in S$) can deduce their own message. If this is possible, we say that the Public Channel Game (G, A, P) has a solution.

Definition

A directed graph $G = (V, E)$ has *(general) linear guessing number* $k = k_s$ if the Public Channel Game (G, A, P) has solution for some A with $|A| = s$ and with $P = s^{|V|-k}$.

In the case of figure 3(iii) each player would be able to calculate his own dice value if, for example, $x_1 \oplus x_4, x_2 \oplus x_4$ and $x_3 \oplus x_4$ modulo s were known public information. [To see this, notice that node 1 receives x_4 from which it can calculate $x_1 = (x_1 \oplus x_4) \ominus x_4$, node $i = 2, 3$ receives x_{i-1} from which it can calculate $x_i = (x_i \oplus x_4) \ominus (x_{i-1} \oplus x_4) \oplus x_{i-1}$. Finally, node 4 receives x_3 from which it can calculate $x_4 = (x_3 \oplus x_4) \ominus x_3$].

For any information network N we can apply the split move until all inner nodes have been spilt. In this case N becomes a bipartite graph B_N with no inner nodes. Notice that B_N is uniquely determined by N.

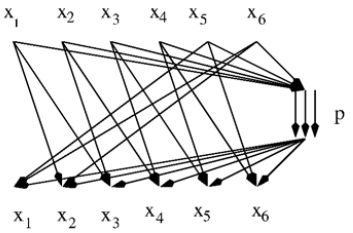

Fig. 9.

This example suggests that it is always possible to replace the guessing part of guessing game, and instead let all players have access to a suitable public channel of information. We will show (Corollary 10) that this is possible for linear solutions (also sometimes called matrix linear) for the guessing game, but it is never possible if only non-linear solutions exists. Notice, that this analysis is only meaningful when the alphabet (i.e. the dice values) can be organized as a vector space U (of dimension d) over a finite field F (with a number q of elements being a prime power). The number $|U|$ of element of U is given by $s := q^d$.

Theorem 5

Assume that the alphabet U is a vector space of dimension d over a finite field F with q elements (i.e. q is a prime power). Then the following statements are equivalent:

(1) The players have a linear guessing strategy in the Guessing Game (G, U) that succeeds with probability $(\frac{1}{q^d})^k$
(2) G has a special linear guessing number $k = k_{lin}(G, q^d)$.
(3) The Public Channel Game (G, U, U^k) has a solution (possible non-linear).
(4) The Bipartite information flow problem B_G associated to G has a solution (over U and possible non-linear) that uses a public channel P of bandwidth k.
(5) The Bipartite information flow problem associated to G has a linear solution (over U) that uses a public channel of bandwidth k.
(6) The Public Channel Game (G, U, U^k) has a linear solution.

From this we get:

Theorem 6

Assume that the alphabet U is a finite dimensional vector space over a finite field F. The nodes in a directed graph G can calculate their messages (selected from U) if they have access to a public channel of bandwidth $\leq k$ if and only if the (special) linear guessing number of G is $\geq |V| - k$.

Theorem 6 explain the terminology of the *(special) linear guessing number*. In the case where the alphabet is a vectorspace the linear guessing number (in sense of linear maps) agree with the *(general) linear guessing number*. The two notions of linear guessing numbers agree when they are both defined. The general linear guessing number is, however, defined for all $s \in \{2, 3, 4, \ldots, \}$, while the special linear guessing number only is defined when s is a prime power (since a finite dimensional vector space always has a number of elements being a prime power).

6.2 B. Proof of Theorem 5

First notice that (1) and (2) are equivalent (by definition). We claim:

Lemma 7

(1) implies (3):

Proof: We are given a graph $G = (V, E)$ and we consider the Guessing Game (G, U, U^k), for U being a vector space of dimension d over a field F with q elements (q being a prime power). The number k is given by (1). We assume that the players have a linear guessing strategy, i.e. a strategy where all functions $f_w : U^{r_\omega} \to U$ are linear (i.e. given by a $r_\omega d \otimes d$ matrix with entries in F). Furthermore we assume this linear guessing strategy makes it possible for the players to guess correctly all their own dice values with probability $(\frac{1}{q^d})^k$.

Consider $\tilde{U} := U^{|V|}$, the linear subspace of vectors $(v_1, v_2, \ldots, v_{|V|}) \in U^{|V|}$ with $v_j \in U$ for $j = 1, 2, \ldots, |V|$. Let $W \subseteq \tilde{U}$ denote the linear subspace of dice values for which the players all successfully guess their own dice value (while using the linear guessing strategy for which we assume that it exists). Since the strategy is successful with probability $(\frac{1}{q})^{dk}$ and since the number of points in $\tilde{U}$ is $q^{d|V|}$ the number of points in W is $q^{d|V|-dk}$. Since W is a linear subspace with $q^{d|V|-kd}$ points, its dimension $d|V| - dk$ must be an integer (thus dk must be an integer while k might not be an integer).

For each vector $u \in \tilde{U}$ we consider the linear "side" space $u + W$. Let $u_1, u_2, \ldots, u_l$ denote a maximal family of vectors with $W, u_1 + W, u_2 + W, \ldots, u_l + W$ all being disjoint. It follows that $l = q^{dk} - 1$, i.e. that there are q^{dk} disjoint side spaces of W and that $W \cup_j (u_j + W) = U$.

We can now convert this into a solution to the Public Channel Game (G, U, U^k). We do this by broadcasting a public message as follows: Assume each node in V has been assigned a value from U. The information of all dice values are contained in a vector $u \in \tilde{U}$. There exists exactly one index $j \in \{1, 2, \ldots, l\}$ such that $u \in$

$u_j + W$. Broadcast the index $j \in \{0, 1, 2, \ldots, q^{dk} - 1\}$ by selecting a bijection from $\{0, 1, \ldots, q^{dk} - 1\}$ to U. Now each node can calculate its own message by correcting their guess (they could have played the Guessing Game) by the suitable projection of u_j.

This shows that the Public Channel Game (G, U, U^k) has a solution (possible non-linear) with the public message being selected from the set of public messages U^k. ♣

In this construction, the public channel broadcasts different messages for each index $j \in \{1, 2, \ldots, l\}$. In general this map is not linear. We will show that any non-linear strategy can be turned into a linear strategy.

Lemma 8

(4) implies (5)

Before we prove this implication we make a few general observations and definitions. Assume the flow problem has a solution with the public channel broadcasting

$$p_1(x_1, x_2, \ldots, x_n), \ldots, p_w(x_1, x_2, \ldots, x_n).$$

Since $p_j : A^n \to A$ and A is a field, each function p_j can be expressed as a polynomial $p_j \in A[x_1, x_2, \ldots, x_n]$. Each output node o_j receives $p_1, p_2, \ldots, p_w \in A$ as well as $x_{j_1}, x_{j_2}, \ldots, x_{j_v} \in A$. The task of output node o_j is to calculate $x_j \in A$. For any $q \in A[x_1, x_2, \ldots, x_n]$ let $L(q) \in A[x_1, x_2, \ldots, x_n]$ denote the sum of all monomials (with the original coefficients) of q that only contains one variable (e.g. x_j, x^3, or x_j^7). In other words $L(q)$ consists of q where the constant term, as well as all monomials containing more than one variable, have been removed. If for example $q = 5x_1x_3 - 7x_1x_2 + 3x_1 - 5x_2 + 1$, then $L(q) = 3x_1 - 5x_2$.

We first consider the special case where the alphabet U is a one dimensional vector space (i.e a finite field) rather than a general finite (dimensional) vector space.

Lemma 9

A bipartite information flow problem B has a solution with public information given by polynomials $p_1, p_2, \ldots, p_w \in A[x_1, x_2, \ldots, x_n]$ then B has a solution with public information given by linear expressions $l_1, l_2, \ldots, l_w \in A[x_1, x_2, \ldots, x_n]$.

Remark: In general non-linear flows cannot be eliminated from information networks. In a general network a non-linear solution might for example involve that two nodes send messages $(x + y)$ and $(y + z)$ to a node r where their product $(x + y)(y + z) = xy + xz + yz + y^2 = xy + xz + yz + y$ is being calculated. Removing mixed monomials would lead to $L(x + y) = x + y$ and $L(y + z) = y + z$ to be sent to node r where $L((x + y)(y + z)) = y^2$ must be calculated. Since it is not possible to derive y^2 (or y) from $x + y$ and $y + z$ the process of removing monomials with mixed variables fails in general. The networks in [17] and [7] show that certain flow problems only have non-linear solutions. For such networks any attempt of removing non-linear terms (not just

using local linearisation) will fail. The point of the lemma is that the network B together with any public channel is structured in such a fashion that allows us to remove mixed terms and then replace the resulting function with linear functions. Information networks in which only two messages are transmitted provide another case where linearisation is always possible [8].

Proof of Lemma 9: We apply the operator L that removes all monomials with two or more distinct variables. The public information then becomes $L(p_1), L(p_2), \ldots, L(p_w)$. These functions can be realized (since there are no restrictions on the public channel and all functions $A^n \to A^w$ can be calculated). Using the same argument we can remove all mixed terms and insure that each output node o_j receives a function of its inputs (the input from input nodes as well as from the public channel). This completes the proof of Lemma 9. ♣

In general, when A is a vector space the operator L removes all monomials that contain variables associated to distinct inputs.

Thus it is easy to prove Theorem 5. We have shown (1) → (3) (Lemma 7) , as well as (4) → (5) (Lemma 8).

The implications (5) → (6) → (1) as well as (3) ↔ (4) are all almost trivial and are left as easy exercises for the reader. This completes the proof of Theorem 5. Theorem 6 follows as an easy corollary.

7 Some Corollaries

In general the Guessing Game (G, s) might only have non-linear optimal guessing strategies. When this happens G has linear guessing number k_{lin} that is strictly smaller than G's guessing number k. We have the following characterisation:

Corollary 10

Let $G = (V, E)$ be a graph and let U be a finite vector space. The linear guessing number k_{lin} of G over U is smaller or equal to the guessing number k of G. Equality holds if and only if the Public Channel Game $(G, U, U^{|V|-k}))$ is solvable.

We have seen that the problem of solving information network flow problems (of class C) can be restated to that of calculating the guessing number of a graph. The linear guessing number of a graph is an important concept:

Corollary 11

The information flow problem $N \in C$ with n input/output nodes has a linear solution (i.e. a solution within the "wave paradigm") over an alphabet of size s if and only if G_N has its linear guessing number $k(G, s) \geq n$ (which happens if and only if $k(G, s) = n$).

8 Algebraic Solution to the Case Where $|A| = 2$

Consider a directed graph $G = (V, E)$. In this section we show that the linear guessing number (for an alphabet of size 2) has an algebraic definition. Assume

G has no edges $(v, v) \in E$ for $v \in V$. We say $G' = (V, E')$ is a subgraph of G if $E' \subseteq E$. The reflexive closure $ref(G)$ of $G' = (V, E')$ is the graph that appear if we add all the edges (v, v) to the edge set E'.

Theorem 12

Let G be a directed graph. Assume that the alphabet A contains two elements. Then G has linear guessing number $k = k_{lin,2}$ (for alphabet of size 2) if and only if there exists a subgraph G' of G such that the rank of the incident matrix for $ref(G') = k$.

Proof: We assume $A = \{0, 1\}$. Assume G has linear guessing number k. According the Theorem 5 G has linear guessing number k if and only if the Public Channel Game $(G, 2)$ has a linear solution S with a public channel of bandwidth k. We say an edge $(v_1, v_2) \in E$ in G is active (with respect to the solution S) if the message in v_1 affects the guessing function in v_2. Let $E' \subseteq E$ consists of all active edges in G. Let $G' = (V, E')$ be the subgraph of G that consists of all active edges in G.

Consider a node $w \in V$ such that $(v_1, w), (v_2, w), \ldots, (v_d, w)$ are all active incoming edges. The (linear) signal being send to node w is $s = m_{v_1} + m_{v_2} + \ldots + m_{v_d}$ i.e. the sum of all incoming signals, as well as the signals that are send from the public channel. Next we assume that the rank of $ref(G')$ is k for some $G' \subseteq G$. Let $l_1(x_1, x_2, \ldots, x_n), l_2(x_1, x_2, \ldots, x_n), \ldots, l_k(x_1, x_2, \ldots, x_n)$ denote the k linearly independent rows of $ref(G')$. Send these signals as public messages. Let w be an arbitrary node. The node receives a signal $m_{v_1} + m_{v_2} + \ldots + m_{v_r}$ from the channels in G'. The node w needs to derive m_w so it suffice to show that the node w can derive $m_{v_1} + m_{v_2} + \ldots + m_{v_d} + x_w$ from the public messages. But, the row $m_{v_1} + m_{v_2} + \ldots + m_{v_d} + m_w$ appears in $ref(G')$ and thus it belong to the span of the k vectors $l_1(x_1, x_2, \ldots, x_n), l_2(x_1, x_2, \ldots, x_n), \ldots, l_k(x_1, x_2, \ldots, x_n)$ that are send through the public channel. ♣

For a graph G let $Ref(G)$ denote the reflexive closure of G. Let $rank(G)$ denote the rank over the field $\{0, 1\}$ of the incident matrix of G.

Theorem 13

Assume the alphabet $A = \{0, 1\}$ only contains two elements. Let G be a graph. Then the Public Channel Game $(G, \{0, 1\}, \{0, 1\}^k)$ has a solution if and only if

$$k \geq min_{G' \subseteq G} rank(Ref(G'))$$

9 More Games

Suppose $N \in C_{multiple-unicasts}$ is an information network where some nodes have indegree > 2. For each node n with indegree $d > 2$ we can replace the incoming d edges with a tree with d leaves and a root in n (see figure 11).

Theoretically this replacement restricts the power of the information network since not all functions $f : A^d \to A$ can be written as a composition of d functions

$g_j : A^2 \to A$, with $j = 1, 2, \ldots, d$. Let S_d denote the class of d-ary functions $f : A^d \to A$ that can be written as a composition of d, 2-ary functions.

Given a directed graph $G = (V, E)$ and assume that each node with indegree d can only compute functions that belong to S_d. How does this affect the guessing number of the graph? How does it affect the set of solutions?

The network in figure 2 corresponds to a type of games that can be described as follows:

- Public Channel Game Variant(G, s): As before let $G = (V, E)$ be a graph on a vertex set $V = \{1, 2, \ldots, n\}$ of persons. The game is played by n players. Each player is assigned a message selected from some alphabet $\{1, 2, \ldots, s\}$. Each person $w \in \{1, 2, \ldots, n\}$ receive *the function value* (a value in $\{1, 2, \ldots, s\}$) from the set $A_w = \{v \in V : (v, w) \in E\} \subseteq V$. Each player also have access to a public information channel p. How many messages should the public channel p be able to broadcast for all players to be able to deduce there own message? Problem 3 (in section I(A)) corresponded to the case where G is the complete graph on n nodes.

As we already pointed out there exists graphs G for which the dice guessing game only can achieve maximal probability, if the players uses non-linear functions.

We will show (and this will follow as a corollary of Theorem 16) that:

Theorem 14

Assume that the public information is given by a function $p : A^n \to A$. Then the Public Channel Game Variant (K_n) played on the complete graph K_n has a solution if and only if there exists a commutative group $(A, \oplus)$ structure on the alphabet A and there exists n permutations $\pi_1, \pi_2, \ldots, \pi_n \in S_A$ of elements in A such that the public channel broadcast

$$p(x_1, x_2, \ldots, x_n) = \pi_1 x_1 \oplus \pi_2 x_2 \oplus \ldots \oplus \pi_n x_n$$

Roughly, Theorem 14 states that the set of solutions consists of all the "obvious" solutions (where $p(x_1, x_2, \ldots .x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n$for a commutative group), together with all "encryptions" $\pi : A \to A$ of these.

10 On the Power of Network Coding

In this section we show that the advantage of (linear) Network Coding compared to any method that does not allows "interference" is as high as one could possible have hoped for. Consider the information networks N in figure 10. The networks corresponds to the Guessing Game on the complete graph K_n.

Theorem 15

For each n there is a network N with n input nodes and n output nodes such that the through-put is n times higher than any method that does not allow interference.

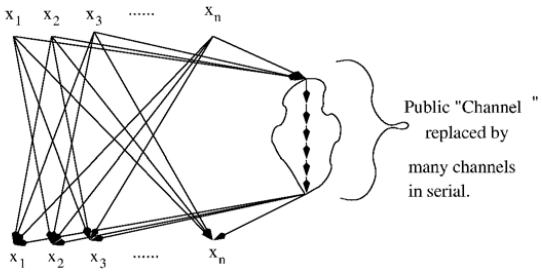

Fig. 10.

For any $n \in N$ and for any $\epsilon > 0$ there exists a network $N(n, \epsilon)$ such that the through-put divided by the number of active channel using Network Coding, is $n - \epsilon$ times as high as the maximal through-put divided by the number of active channels using methods that does not allow interference.

If each inner node is required to have in-degree (and out-degree) ≥ 2 the result remains valid.

Proof: For each $n \geq 2$ (and each $\epsilon > 0$ we base the construction on the network in figure 10. Assume that the public channel consists of m channels in serial. In any "solution" (operating at rate $\frac{1}{n}$) that does not allow mixture of datapackets all messages must go through these m channels. Thus the number of active channels is $m+2$. In the Network Coding solution (operating at rate 1) all $n(n-1)+(m+2)$ channels are active. We can choose m such that $n \times (\frac{(m+2)}{n(n-1)+(m+2)}) > n - \epsilon$. For this m the through-put divided by the number of active channel (in the Network Coding solution) is $n - \epsilon$ times as high as the maximal through-put divided by the number of active channels using methods that does not allow interference.

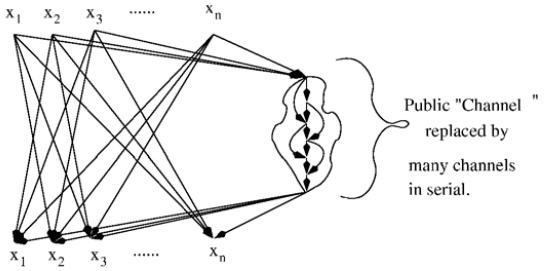

Fig. 11.

The serial construction in this proof might be considered unacceptable. It might be argued that the cost of using the serial channels ought to count as 1 rather than m. To overcome this criticism we can modify the serial channels as indicated in figure 11 and select m such that each path through the public channel still must involve $\geq m$ active channels (m chosen as before). ♣

11 Analysis of Specific Networks

Consider the information network N_n sketched in figure 12. The network $N(3)$ is displayed in in figure 2.

The networks N_n corresponds to the Public Channel Game Variant (K_n, s) played on the complete graph K_n.

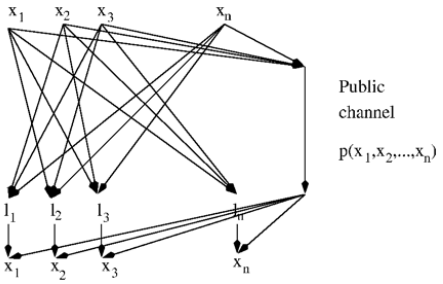

Fig. 12.

Consider, again the information network $N(3)$ in figure 2. The three output nodes receive the messages $l_1(x_2, x_3) \in A, l_2(x_1, x_3) \in A$ and $l_3(x_1, x_2) \in A$. Besides this, each output node has access to public message $p = p(x_1, x_2, x_3) \in A$. We notice that a solution to the flow problem associated with N_3 consists of six functions $l_1, l_2, l_3, r_1, r_2, r_3 : A \times A \to A$ as well as one function $p : A \times A \times A \to A$ such that $x_1 = r_1(p(x_1, x_2, x_3), l_1(x_2, x_3))$, $x_2 = r_2(p(x_1, x_2, x_3), l_2(x_1, x_3))$ and $x_3 = r_3(p(x_1, x_2, x_3), l_3(x_1, x_2))$.

The solution we already considered can be achieved (within the framework of linear Network Coding) as follows: Let $(A, \oplus)$ be an Abelian group, let $p(x_1, x_2, x_3) := x_1 \oplus x_2 \oplus x_3$, let $l_i(x, y) := x \oplus y$ for $i = 1, 2, 3$ and let $r_i(x, y) := \mathrm{x} \ominus \mathrm{y}$ for $i = 1, 2, 3$. We leave to the reader to check that this defines a solution to the flow problem associated with the network N_3.

Actually, for each Abelian group $(A, \oplus)$ and for any three permutations $\pi_1, \pi_2, \pi_3 : A \to A$ the network has a solution with $p(x_1, x_2, x_3) := \pi_1 x_1 \oplus \pi_2 x_2 \oplus \pi_3 x_3$, $l_1(x_2, x_3) := \pi_2 x_2 \oplus \pi_3 x_3$, $l_2(x_1, x_3) := \pi_1 x_1 \oplus \pi_3 x_3$ and $l_3(x_1, x_2) := \pi_1 x_1 \oplus \pi_2 x_2$. We will show that all solutions are essentially of this form. More generally let N_n denote the network:

The network N_n has n input nodes. These transmit messages $x_1, x_2, \ldots, x_n \in A$. The messages $x_1, x_2, \ldots, x_n$ are independent so we assume that the network cannot exploit hidden coherence in the data. The network N_n has n internal nodes $l_1, l_2, \ldots, l_n$. The node l_j is connected to each input node *except* the node that transmits message x_j. The network has n output nodes that are required to receive the messages $x_1, x_2, \ldots, x_{n-1}$ and x_n (one message for each output node). The node required to receive x_j is connected to l_j as well as to the public channel p. The public channel broadcasts one message $p = p(x_1, x_2, \ldots, x_n) \in A$ to all output nodes. First we notice that:

Observation

The network N_n has a solution over any (finite) alphabet A. Using routing only one message can be transmitted at a time. Thus the through-put using Network coding is n-times as large as the through-put using any type of routing method that does not allow interference. This is optimal since any network problem with n input nodes that is solvable using network coding can be solved using routing if the bandwidth is increased by a factor n.

The next Theorem gives a complete classification of the set of solutions (all utilising Network coding) to the network N_n.

Theorem 16

Consider the network flow problem N_n over a finite alphabet A. Assume $n \geq 3$. Let $p : A^n \to A$ be any function. The network flow problem N_n has a solution with public information p if and only if for some group composition $\oplus$ on A that makes $(A, \oplus)$ an Abelian group, there exist n permutations $\pi_1, \pi_2, \ldots, \pi_n : A \to A$ such that $p(x_1, x_2, \ldots, x_n) = \oplus_{j=1}^{n} \pi_j x_j$.

Proof: In general if Theorem 16 have been shown for N_r for some $r \geq 3$ the theorem is also valid for each N_s with $s \geq r$. Thus to prove the theorem it suffices to show that the theorem is valid for N_3.

Let $p : A^3 \to A$ be defined by $p(x_1, x_2, x_3)$. Assume that the network has a solution when the public signal is given by p. The function $p : A^3 \to A$ must be 'latin' (i.e. $f_{a,b}(z) := p(a, b, z)$, $g_{a,c}(y) := p(a, y, c)$ and $h_{b,c}(x) := p(x, b, c)$ for each $a, b, c \in A$ define bijections $f_{a,b}, g_{a,c}, h_{b,c} : A \to A$). Notice that p defines a latin cube of order $|A|$. The functions $l_1.l_2, l_3 : A^2 \to A$ are also forced to be latin i.e. they define three latin squares each of order $|A|$. In order to proceed we need to prove a number of lemmas.

Lemma 17

Denote one element in A by 1. The network N_3 has a solution for some functions $l_1, l_2, l_3 : A^2 \to A$ if and only if the network N_3 has a solution when $l_1(x_2, x_3) := p(1, x_2, x_3)$, $l_2(x_1, x_3) := p(x_1, 1, x_3)$ and $l_3(x_1, x_2) = p(x_1, x_2, 1)$.

Proof of Lemma 17: We introduce a new and interesting type of argument that might be useful when reasoning about 'latin' network flow in general. For each output node we draw a triangle with a coding function assigned to each corner. The triangle corresponding to the output node that required output x_1 has assigned $p(x_1, x_2, x_3), l_1(x_2, x_3)$ and x_1 to its corners. If p and l_1 are functions that produce a solution to the network flow problem, $x_1 \in A$ can uniquely be calculated from $p(x_1, x_2, x_3) \in A$ and $l_1(x_2, x_3) \in A$ (i.e. there exists a (latin) function $f : A^2 \to A$ such that $x_1 = f(p(x_1, x_2, x_3), l_1(x_2, x_3)))$. Notice, that any coding function assigned to one of the corners can be calculated uniquely from the two other functions. More specifically $l_1(x_2, x_3) \in A$ is uniquely determined by $x_1 \in A$ and $p(x_1, x_2, x_3) \in A$. And the value $p(x_1, x_2, x_3)$ is uniquely determined by x_1 and $l_1(x_2, x_3)$. We say that a triangle with a coding function

assigned to each corner is 'latin' if each of the three coding functions can be calculated from the two other functions. For any solution of the network flow problem N_3 each of the following three triangles are latin:

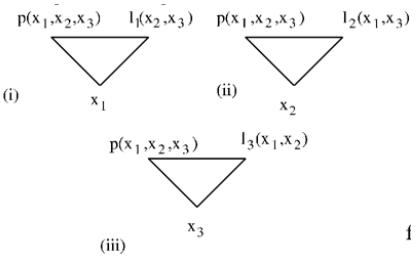

Fig. 13.

Letting $x_1 = 1$ in triangle (i) we notice that $p(1, x_2, x_3)$ can be calculated from $l_1(x_2, x_3)$ and conversely we notice that $l_1(x_2, x_3)$ can be calculated from $p(1, x_2, x_3)$. Thus we can replace the function $l_1(x_2, x_3)$ with the function $l_1(x_2, x_3) = p(1, x_2, x_3)$. Similarly, by letting $x_2 = 1$ in triangle (ii) and letting $x_3 = 1$ in triangle (iii) we obtain a solution with $l_2(x_1, x_3) = p(x_1, 1, x_3)$ and $l_3(x_1, x_2) = p(x_1, x_2, 1)$. This completes the proof. ♣

Lemma 18

Assume that there is a solution to the flow problem N_3 with public information given by $p : A^3 \to A$. Then the latin function $p(x_1, x_2, x_3)$ determines (uniquely) two latin functions (i.e two latin squares) $l : A^2 \to A$ (l stands for 'left') and $r : A^2 \to A$ (r stands for 'right') defined by the two equations:

- $p(1, l(x_1, x_2), x_3) = p(x_1, x_2, x_3)$
- $p(x_1, r(x_2, x_3), 1) = p(x_1, x_2, x_3)$

Proof of Lemma 18: Certainly (since p is latin), there exist uniquely defined functions $l', r' : A^3 \to A$ such that $p(1, l'(x_1, x_2, x_3), x_3) = p(x_1, x_2, x_3)$ and $p(x_1, r'(x_1, x_2, x_3), 1) = p(x_1, x_2, x_3)$. To show Lemma 18 it suffices to show that l' is independent of x_3 and that r' is independent of x_1. Consider the two latin triangles:

Fig. 14.

In each triangle (iv) and (v) each coding function is uniquely determined by the two other coding functions in the triangle. Thus there exists $f, g : A^2 \to A$ such that $p(x_1, x_2, x_3)$=$f(p(x_1, x_2, 1), x_3)$ and such that $p(x_1, x_2, x_3)$= $g(x_1, p(1, x_2, x_3))$. Let $l(x_1, x_2) = l'(x_1, x_2, 1)$ and let $r(x_2, x_3) = r'(1, x_2, x_3)$ and notice that $p(x_1, x_2, 1) = p(1, l(x_1, x_2), 1)$ and $p(1, x_2, x_3) = p(1, r(x_2, x_3), 1)$. But then $p(x_1, x_2, x_3) = f(p(x_1, x_2, 1), x_3) = f(p(1, l(x_1.x_2), 1), x_3) = p(1, l(x_1, x_2), x_3)$ and $p(x_1, x_2, x_3) = g(x_1, p(1, x_2, x_3)) = g(x_1, p(1, r(x_2, x_3), 1) = p(x_1, r(x_2, x_3), 1)$. Thus l and r satisfies the same equations that uniquely determined l' and r' and thus $l'(x_1, x_2, x_3) = l(x_1, x_2)$ and $r'(x_1, x_2, x_3) = r(x_2, x_3)$. This completes the proof. ♣

Lemma 19

Assume that $p : A^3 \to A$ has a solution and that $p(x_1, x_2, x_3) = p(1, l(x_1, x_2), x_3)$ and assume that $p(x_1, x_2, x_3) = p(x_1, r(x_2, x_3), 1)$. Then the functions $l, r : A^2 \to A$ satisfy the equation $r(l(x_1, x_2), x_3) = l(x_1, r(x_2, x_3))$.

Proof: Since p is latin and $p(x_1, x_2, x_3) = p(1, r(l(x_1, x_2), x_3), 1) = p(1, l(x_1, r(x_2, x_3)), 1)$. ♣

The next three lemma are straight forward to prove.

Lemma 20

Assume $p(x_1, x_2, x_3)$ allows a solution and that $l(x_1, x_2)$ and $r(x_2, x_3)$ are defined such that $p(1, l(x_1, x_2), x_3) = p(x_1, x_2, x_3)$ and $p(x_1, r(x_2, x_3), 1) = p(x_1, x_2, x_3)$. Then for each pair $\pi_1, \pi_3 : A \to A$ of permutations $p'(x_1, x_2, x_3) = p(\pi_1 x_1, x_2, \pi_3 x_3)$ allows a solution and $l'(x_1, x_2)$=$l(\pi_1 x_1, x_2)$ and $r'(x_2, x_3) = r(x_2, \pi_3 x_3)$ satisfies the equations

$$p'(1, l'(x_1, x_2), x_3) = p'(x_1, x_2, x_3) \text{ and } p'(x_1, r'(x_2, x_3), 1) = p'(x_1, x_2, x_3).$$

Lemma 21

There exists permutations $\pi_1, \pi_3 : A \to A$ such that $l(\pi_1 x_1, 1) = x_1$ and such that $r(1, \pi_3 x_3) = x_3$.

Lemma 22

If $p(x_1, x_2, x_3)$ is a solution, there is another solution $p'(x_1, x_2, x_3) = p(\pi_1 x_1, x_2, \pi_3 x_3)$ such that the two functions $l'(x_1, x_2)$ and $r'(x_2, x_3)$ that satisfy the equations $p'(1, l'(x_1, x_2), x_3) = p'(x_1, x_2, x_3)$, $p'(x_1, r'(x_2, x_3), 1) = p'(x_1, x_2, x_3)$ as well as $l'(x_1, 1) = x_1$ and $r'(1, x_3) = x_3$.

Without loss of generality (possibly after having replaced x_1 and x_3 by $\pi_1 x_1$ and $\pi_3 x_3$) we can assume that we are given a latin function $p(x_1, x_2, x_3)$ and two latin functions $l(x_1, x_2)$ and $r(x_2, x_3)$ that satisfies $l(x_1, 1) = x_1$, $r(1, x_3) = x_3$, and have $l(x_1, r(x_2, x_3)) = r(l(x_1, x_2), x_3)$ for all $x_1, x_2, x_3 \in A$. But, then $r(x_1, x_3) = r(l(x_1, 1), x_3) = l(x_1, r(1, x_3)) = l(x_1, x_3)$ and thus $l = r$. But then l is transitive i.e. $l(x_1, l(x_2, x_3)) = l(l(x_1, x_2), x_3)$. Furthermore since $l(x, 1) = x$ and $l(1, x) = r(l, x) = x$ we notice that l defines a group operation on A. Thus we have shown that for any function $p(x_1, x_2, x_3)$ that allows a solution to the

network flow problem N_3, there exist permutations $\pi_1, \pi_3 : A \to A$ such that if we let $p'(x_1, x_2, x_3) = p(\pi_1 x_1, x_2, \pi_3 x_3)$ then there is a group structure $*$ on A such that $p'(x_1, x_2, x_3) = p'(1, x_1 * x_2 * x_3, 1)$ for all x_1, x_2, x_3. But then there is a permutation $\pi : A \to A$ such that if we let $p''(x_1, x_2, x_3) = \pi(p'(x_1, x_2, x_3))$ then $p''(1, b, 1) = b$ for all $b \in A$. Notice, that $p''(x_1, x_2, x_3) = \pi(p'(x_1, x_2, x_3)) = \pi(p'(1, x_1 * x_2 * x_3, 1)) = p''(1, x_1 * x_2 * x_3, 1) = x_1 * x_2 * x_3$. This shows:

Lemma 23

Let $p : A^3 \to A$ be the public information in the network N_3. Then, if there is a solution to the network flow problem N_3, there exists a group composition $*$ on A such that 'essentially' $p(x_1, x_2, x_3) = x_1 * x_2 * x_3$ (modulo the application of suitable permutations to x_1, x_3 and p (or x_2)).

Lemma 24

Let $(A, *)$ be a group and let $p(x_1, x_2, x_3) = x_1 * x_2 * x_3$. Then the flow problem N_3 has a solution if and only if $(A, *)$ is a commutative group.

Proof: Assume that $p(x_1, x_2, x_3) = x_1 * x_2 * x_3$ (or just $x_1 x_2 x_3$ for short) allows a solution. Then we have the following 'derivation' from latin triangle with coding functions $p(a, b, c) = abc$, $p(a, 1, c) = ac$ and b.

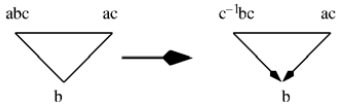

Fig. 15.

Figure 15, represents the fact that b can be uniquely determined from abc and ac. But, then given $c^{-1}bc$ and ac we can calculate $abc = (ac)c^{-1}bc$ and thus we can determine b. Now ac can take any value (depending on a) and thus this equation is useless in calculating b. This shows that b is uniquely determined from $c^{-1}bc$. The expression $c^{-1}bc$ must be independent of c and thus $c^{-1}bc = 1^{-1}b1 = b$. But, then $bc = cb$ for all $a, b, c \in A$ which shows that the group $(A, *)$ must be a commutative group. The converse is rather obvious, since if $(A, *)$ is an ablean group and $p(x_1, x_2, x_3) = x_1 x_2 x_3$, we get a solution by letting $l_1(x_1, x_2) = x_1 x_2$, $l_2(x_1, x_3) = x_1 x_3$ and $l_3(x_1, x_2) = x_1 x_2$. This completes the proof of Lemma 19 which in turn clearly implies the theorem for N_3. This in turn easily implies the validity of Theorem 2 for general N_n with $n \geq 3$. ♣

References

1. S. Medard, R.Koetter, M. Acedanski, and S. Deb, How good is random linear coding based distribued network storage? to appear.
2. A. Aggarwal and J.S. Vitter, The input/output complexity of sorting and related problems, Communications of the ACM, 31, 9, 1116–1126, 1988.
3. R Ahlswede, N Cai,S.Y.R. Li, and R.Yeung, Network information flow, IEEE Trans. Inf. Theory, Vo. 46, No. 4, 1204-1216, 2000.

4. K.R. Bhattad and K. Narayanan, Weakly secure network coding, to appear.
5. N. Cai and R.W. Yeung, Network coding and error correction, ITW Bangalore, 119–122, 2002.
6. S. Deb, C. Choute, M. Medard, and R. Koetter, Data harvesting: A random coding approach to rapid dissemination and efficient storage of data, INFOCOM, 2005, submitted.
7. R. Dougherty, C. Freiling, and K. Zeger, Insufficiency of linear coding in network information flow, Technical report, February 2004.
8. R. Dougherty, C. Freiling, and K. Zeger, Linearity and solvability in multicast networks, Proceeding of CISS, 2004.
9. R. Dougherty, C. Freiling, and K. Zeger, Network routing capacity, IEEE/ACM TRANSACTIONS ON NETWORKING, October 2004, submitted.
10. C. Fragouli and E. Soljanin, A connection between network coding and convolutional codes, IEEE International Conference on Communications, 2004.
11. T. Ho, M. Medard, and R. Koetter, An information theoretic view of network management, Proceeding of the 2003 IEEE Infocom, 2003.
12. R. Koetter and M. Medard. An algebraic approach to network coding, Proceedings of the 2001 IEEE International Symposium on Information Theory, 2001.
13. R. Koetter and M. Medard, Beyond routing: an algebraic approach to network coding, Proceedings of the 2002 IEEE Infocom, 2002.
14. S.-Y.R. Li, R.W.Yeung, and N. Cai, Linear network codes, IEEE Trans. Inform. Theory, Vol. 49 ,371-381, 2003.
15. K. Rabaey, J. Petrovic, and D. Ramchandran, Overcomming untuned radios in wireless networks with network coding, to appear.
16. L.G. Pippenger and N. Valiant, Shifting graphs and their applications, JACM, 23, 423–432, 1976.
17. S. Riis, Linear versus non-linear boolean functions in network flow, Proceeding of CISS, 2004.
18. S. Riis and R. Ahlswede, Problems in network coding and error correcting codes, NetCod 2005.
19. M. Thorup and S. Riis, Personal communication, 1997.
20. L. Valiant, Graph-theoretic arguments in low-level complexity LNCS, Springer Verlag, No. 53, 162-176, 1997.
21. L. Valiant, On non-linear lower bounds in computational complexity, Proc. 7th ACM Symp. on Theory of Computing, 45–53, 1975.
22. L. Valiant, Why is boolean circuit complexity theory difficult? In M.S. Patterson, editor, Springer Lecture Series, 84–94, 1992.
23. C. Boudec J.Y. Widmer, and J. Fragouli, Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding, to appear.
24. Y. Wu, P.A. Chou, and S.Y. Kung, Information exchange in wireless networks with network coding and physical-layer broadcast, Technical Report MSR-TR-2004-78, Microsoft Technical Report, Aug. 2004.
25. R. Yeung and Z. Zhang, Distributed source coding for satellite communications, IEEE Trans. Inform. Theory, Vol. 45, 1111–1120, 1999.