Xerox PARC Technical Report

Secure Group Communication over Partially Connected Networks

Michel Abdalla *
Dept. of Computer Science & Engineering
University of California at San Diego
9500 Gilman Drive
La Jolla, California 92093-0123
michel.abdalla@ens.fr

Matthew Franklin **
Dept. of Computer Science
University of California at Davis
3021 Engineering II
Davis, CA 95616-8562
franklin@cs.ucdavis.edu

October 2000

Abstract

In this paper, we consider secure group communication protocols over partially connected networks. These protocols enable a user to robustly and privately transmit a message to any subset of users in the network, even in the presence of some malicious users. Our main goal is to design protocols for which we can give guarantees on the maximum number of rounds it will take. To do so, we first design a simple and efficient point-to-point protocol and then extend it to point-to-multipoint case in a straightforward way. Despite its simplicity, the resulting protocol is highly parallelizable and very efficient in terms of rounds of communication.

The protocols we construct require communication networks with enough short disjoint paths between any pair of users. We show both empirically and analytically how to construct these networks. In the empirical constructions, this is done by experimenting with random regular graphs. We show by means of extensive simulation that these graphs are rich in short disjoint paths for reasonable values of node degree. In the analytical constructions, we give several constructions for which we can give guarantees on the number and length of disjoint paths between any two nodes.

Keywords: multicast, privacy, reliability, secure multi-party group communication, network design.

* Work done while author was visiting Xerox PARC.
** Work done while author was at Xerox PARC.

Table of Contents

1 Introduction

In this technical paper, we consider secure group communication protocols over partially connected networks. These protocols enable a user to transmit a message to any subset of users with privacy and reliability, even in the presence of some malicious parties. We also call these protocols "targeted", because the sender can target the communication to any group of receivers within the network.

Our protocols are multi-party, in the sense that they require the participation of the sender, the receivers, and nodes in the network that are outside the targeted receiver set. The non-receivers participate by merely relaying information that is sent to them during the protocol. There is no need to trust the non-receivers completely. Instead, we assume that a sufficient fraction of all of the parties are behaving properly, while tolerating any form of malicious behavior by the rest of the parties. Our protocols are multi-round as well. The number of rounds depends on the particular topology of the communication network.

Our main motivation for this work is to provide an alternative to existing methods for handling secure communication within dynamic coalitions. In a dynamic coalition setting, receivers may need to be kicked out of the group, so that they cannot learn future communication. With targeted group communication, this does not require any rekeying, because senders merely exclude the kicked-out parties from the designated receiver set.

Broadcast encryption methods [10] are single-round, single-party protocols for targeted group communication. Unfortunately, they are inefficient when the pool of potential receivers is large, due to the explosion of secret keys that must be maintained. Secure multicast methods (e.g., [7]) are single-round, single-party protocols for broadcast without targeting. They require some rekeying whenever a party needs to be kicked out of the group. Despite clever efficiencies of the rekeying protocol itself, the result can be quite inefficient when the frequency of rekeying is high.

We model our communication network by an undirected graph in which each node represents a user. An edge is present between two nodes whenever these nodes can communicate securely to each other. We do not care how these basic secure channels are implemented. They could be lower-layer protocols between parties who share secret keys, or they could be physically secure communication links.

There are several properties that we may want from our protocols, such as low storage, high resilience, and scalability. In addition, we also want efficiency in communication. We are particularly focused on the round complexity of our protocols, although the total number of bits transmitted is important as well. Measures of traffic congestion may be important, too, although we do not consider them in this study. Of course, traffic congestion becomes much less important when targeted secure group communication is used to transmit a session key, and ordinary multicast is used to propagate bulk encrypted data.

We begin by describing simple protocols that will work over any network. The fault-tolerance of these protocols depends critically on the connectivity of the network, while the round complexity depends critically on the lengths of the disjoint paths between nodes[1]. This second measure has not received much attention in the literature. We go on to study this measure both empirically and analytically. We suggest specific network designs for which connectivity and path lengths are particularly well-balanced, and thus particularly well-suited for running our protocols.

1.1 Related work

MENGER'S THEOREM. Menger's theorem [16] shows that if a graph is k-connected, then there exists k disjoint paths between any two nodes in the graph. However, nothing is said about the length of these paths. In [15], the length of these paths is taken into consideration and the relation between the size of vertex-cuts that destroys all bounded paths between two nodes and the number of bounded disjoint paths between these nodes is investigated. In [12], Galil and Yu give bounds on the length of these paths when these paths are edge-disjoint. More precisely, they prove that in a undirected graph with N nodes and k edge-disjoint paths

[1] Throughout this paper, we will use "disjoint paths" to mean "node-disjoint paths", i.e., paths with only their endpoints in common. There is an analogous notion of edge-disjoint paths, which we will occasionally refer to explicitly.

between any two nodes, the average length of these paths is $O(N/\sqrt{k})$. Moreover, if all vertices have degree at least k, then the upper bound is even tighter and equal to $O(N/k)$.

The Bounded Disjoint Path problem. In the Bounded Disjoint Path (BDP), the goal is to find the maximum number of disjoint paths between the sender and receiver whose length is bounded by some value k. As it will become clear later, this problem plays an important role in our protocols. In [13], it is shown that, except for some small values of k, the BDP problem and several of its variants are NP-hard. Therefore, we should not hope to find efficient algorithms to solve the BDP problem. For all those cases in which a polynomial-time solution can exist, one such algorithm is provided in [13]. It is worth mentioning here that, as shown by [22], the related problem of finding the maximum number of disjoint paths with minimum total length can be solved efficiently. This metric can be of importance in some cases in which we are interested in the performance of the communication protocol as a whole.

In [20], an approximation algorithm for the BDP problem is proposed and proved to be optimal for $k < 5$. They also give support by means of simulation that the algorithm performs well for larger values of k. In [19], other heuristic algorithms for the BDP problem are given, which are more efficient than that of [20] both in terms of time and space. They also support the efficacy of their algorithms by extensive simulation.

Connectivity. In [9], the problem of point-to-point secure communication over partially connected networks is considered. They show, among several other results, that one can achieve perfect privacy and perfect reliability in the presence of up to t malicious faults if there are at least $2t + 1$ disjoint paths between the sender and receiver. They also show that this condition is not only sufficient but in fact necessary. The communication model assumes that only single channels are available, i.e. the user can feed each channel with totally independent messages.

In [2], Beimel and Franklin consider the problem of reliable communication when some pair of nodes in the network share authentication keys. They showed that in this case, the total number of faults that can be tolerated increases with respect to the same total number of nodes. These authentication keys can be viewed as an alternative to increasing the fault tolerance without adding more communication channels. Nevertheless, this comes at the cost of not having perfect reliability. That is, there exists a very small probability that a message being transmitted is not received correctly by the receiver node.

[11] extends the work of [9] to consider cases where the protocol may be unreliable or not private with some negligible probability and give a complete characterization of when secure communication is possible. They also consider a more general model for communication in which multicast "lines" are available. These lines (channels) allow the user to send a message in a single round to all its neighbors. The same message will be received by all its neighbors. The user cannot, however, use these lines and send different messages to different neighbors. They show that, in single line model, $t+1$ disjoint paths between sender and receiver are necessary and sufficient for almost perfect reliability when we have at most t malicious nodes. They also show that, in the single line model, $2t + 1$ disjoint paths between sender and receiver are sufficient and necessary for almost perfect privacy, which does not improve over the case of perfect privacy. However, they show that an improvement is possible in the multicast line model. More specifically, they prove that $t+1$ disjoint paths between sender and receiver are necessary and sufficient to achieve almost perfect privacy when at most t malicious nodes are present.

In [17], Nikoletseas et al investigate the multi-connectivity properties of random regular graphs with edge faults. Among other things, they show that a random regular graph of average degree d, in which some of its edges fails independently with probability f, remains d-connected except for $O(1)$ vertices with very high probability for all failure probabilities $f \leq N^{\epsilon}$ with for fixed $\epsilon > 0$. In a related paper [18], Nikoletseas and Spirakis study expander properties of random regular graphs with edge faults. More precisely, they determine the value of $p = 1 - f$ for which the the giant component of the remaining graph remains a *certified efficient expander* with high probability.

Expander graphs. In general terms, we say a graph is an (vertex) α-expander if, for each subset of vertices X including at most half of the vertices, the total number of nodes in the neighborhood of X is at least $\alpha|X|$. In [14], Kleinberg and Rubinfeld consider bounded degree (edge) expander graphs and build upon previous work asserting that expander graphs are rich in short disjoint paths. In particular, they provide a

greedy algorithm for approximating the maximum number of disjoint paths in expander graphs for which they provide performance guarantees.

1.2 Outline

In Section 2, we present our communication model as well as some of the definitions we use in other sections. Section 3 presents a simple and secure group communication protocol. The protocol is quite efficient in terms of communication and works well for multicast group of small sizes. These protocols require communication networks with enough short disjoint paths between every pair of nodes. In Section 4 and Section 5, we present two different approaches for designing such networks. Section 4 presents some empirical designs based on random regular graphs, where we analyze their behavior in terms of short disjoint paths when we vary the average degree of a node. We show by means of extensive simulation that the disjoint paths between any two nodes in such graphs are in general very short even for small degrees. However, we cannot give any guarantees on the maximum length of these paths. Section 5, on the other hand, presents several constructions of graphs for which we can give guarantees on the number and length of disjoint paths between any pair of nodes. These guarantees, however, are usually conservative and in some cases very far from what can be achieved in practice.

2 Model and definitions

Model. We represent the network by an undirected graph $G = (V, E)$, where the set of vertices $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ represents the set of nodes in the network and $(\mathbf{v}_i, \mathbf{v}_j) \in E$ whenever nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ can communicate reliably and privately between themselves. This can be achieved, for example, by having shared keys between these nodes for both encryption and authenticity.

Connectivity. Let $G = (V, E)$ be a graph and let $X \subseteq V$. A graph G is said to be *connected* if any two nodes in V is connected by a path in G. For any two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in V, we say that X *separates* nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in G if any path from $\mathbf{v}_i$ and $\mathbf{v}_j$ contains at least one node in X. Or more generally we say that X separates G if it separates any two nodes of $V - X$ in G. Now if $G - X$ is connected for every set $X \subset V$ of size less than δ, then we say that G is δ-connected. That is, no two nodes in V are separated by fewer than δ other nodes. By Menger's theorem [16], the above definition is equivalent to saying that any two nodes in V can be connected by δ node-disjoint paths in G [8]. Hence, the following definition.

Definition 1. *Let $G = (V, E)$ be a graph on N elements. We say G is δ-connected if any two nodes in G can be connected by δ node-disjoint paths in G.*

The above definition does not take into consideration the length of the disjoint paths. For this reason, we extend here the above definition to capture the notion of maximum length of disjoint paths between any two nodes.

Definition 2. *Let $G = (V, E)$ be a graph on N elements. We say G is (δ, k)-connected if any two nodes in G can be connected by δ disjoint paths in G, each of length at most k.*

Reliability. Following [11], we have two variations for the definition of reliability in protocols. In the first one, we say a protocol is almost-perfectly reliable if a message transmitted by a node $\mathbf{v}_i$ to a node $\mathbf{v}_j$ is received by the latter with very high probability even in the presence of faults. In this variation, there is a very small probability that the receiver does not get the message. We, however, are more interested in the second variation which does not allow for the possibility of the receiver not obtaining the message. In this second variation, we say that a protocol is (perfectly) reliable if a message transmitted by a node $\mathbf{v}_i$ to a node $\mathbf{v}_j$ is always received by the latter even in the presence of faults. In the following, we give a more precise definition which also takes into account the size of the subset of nodes controlled by the adversary.

Definition 3. *Let $G = (V, E)$ be a graph on N elements and let $\mathbf{v}_i$ and $\mathbf{v}_j$ be any two nodes in V. A protocol is said to be t-reliable if any message sent by $\mathbf{v}_i$ to $\mathbf{v}_j$ is received by the latter even if the adversary can control any subset $T \subset V \setminus \{\mathbf{v}_i, \mathbf{v}_j\}$ of size at most t.*

PRIVACY. Similarly to the case of reliability, we can have two variations for the definition of privacy in protocols, depending on whether we allow a very small probability of some information about the message being transmitted to be leaked (almost perfect privacy) or not (perfect privacy). Since we are interested in the case of perfect privacy, we only give a formal definition for this variant.

Definition 4. *Let $G = (V, E)$ be a graph on N elements and let $\mathbf{v}_i$ and $\mathbf{v}_j$ be any two nodes in V. A protocol is said to be t-private if, for any pair of equal-length messages m_0 and m_1, no subset $T \subset V \setminus \{\mathbf{v}_i, \mathbf{v}_j\}$ of size at most t can distinguish which message was sent from $\mathbf{v}_i$ to $\mathbf{v}_j$.*

3 Protocol

In this section, we assume we have networks with enough short disjoint paths between any two nodes. We show in later sections how one can design such networks. So, let us turn our attention to the problem of designing secure group communication protocols for which we can give bounds on the total number of rounds it can take.

The protocol we present is based on an efficient point-to-point protocol and uses a naïve approach of transmitting the message to each member in the multicast group separately. Despite its simplicity, our protocol is highly parallelizable and very efficient in the total number of rounds it can take. Of course, this simple approach has its limitations and can only be applied to small groups, since the amount of information being transmitted grows linearly with the size of the group. But it is our opinion that such approach, yet limited, still meets the requirements of several applications. It remains as future work to design group communications protocols which are still round efficient whose communication costs are sub-linear in the size of the multicast group.

3.1 Point-to-point protocol

Let $G = (V, E)$ be a $(3t + 1, k)$-connected graph, where t is the maximum number of faults the protocol tolerates. Let $\mathbf{v}_i$ and $\mathbf{v}_j$ in V be respectively the source and target nodes. The goal is to transmit a message msg securely from $\mathbf{v}_i$ to $\mathbf{v}_j$. We assume v_i and v_j are not connected by an edge in E, else no protocol is needed for them to communicate securely. We assume the message msg to be an element of a finite field of prime order, say Z_p^*. The protocol works as follows.

TRANSMISSION. The main idea is to use standard polynomial secret sharing[21]. Let p_x, for $l = 1, \ldots, 3t+1$, denote a disjoint path in G between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. The sender node $\mathbf{v}_i$ first picks a random polynomial $f(x) = \mathrm{msg} + c_1 x + \ldots + c_t x^t$ in Z_p^* of degree t, by first making its 0-coefficient equal to msg and then picking all the other coefficients c_j for $l = 1, \ldots, t$ at random from Z_p^*. Then, for each disjoint path $l = 1, \ldots, t$, $\mathbf{v}_i$ sends the value $f(l)$ through path p_l to node $\mathbf{v}_j$. For a pictorial representation of our point-to-point protocol, please refer to Fig. 1.

RECONSTRUCTION. The goal of the receiver node $\mathbf{v}_j$ is to reconstruct the message msg from its shares even when some of these shares are not correct. Note that this is possible since we allow for up to t malicious adversaries. In order to do so, we opted for using Berlekamp's reconstruction algorithm [3]. The algorithm is very simple, requiring first solving a system of linear equations and then dividing one polynomial by another, and yet quite efficient, taking $O(t^3)$ by naïve methods (for a concise description of this algorithm, please refer to [11]. The main property of this algorithm in which we are interested is that the correct reconstruction of the secret is possible whenever less than a third of these shares are wrong. More precisely, it allows us to reconstruct msg from the $3t + 1$ shares if at most t of them are not correct. For completeness, $f(l)$ should be set to a predefined value, say $\perp$, whenever the point $f(l)$ transmitted through path p_x is not received by node $\mathbf{v}_j$.

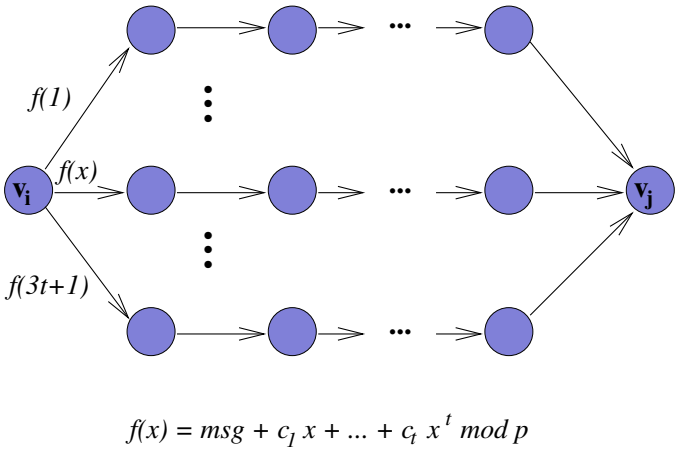


Fig. 1: Pictorial representation of our point-to-point protocol. Each path is labeled a value i between 1 and $3t+1$.

Efficiency. If we assume that transmission of each point $f(l)$ can be done in parallel and that the given graph G is $(3t+1,k)$-connected, then the above protocol takes at most k rounds since the length of each of these paths is bounded by k.

Privacy. One can easily see that no information about msg is leaked to any adversary controlling up to t nodes. This adversary has access to at most t shares in the worst case. However, we know that Shamir secret sharing is information theoretically secure and all sets of t shares have the same probability.

Reliability. From Berlekamp's algorithm, we know that the receiver can reconstruct msg from $3t+1$ shares even if up to t of these shares contains incorrect values. Therefore, the protocol can tolerate up to t faults. This corresponds to the worst possible case in which we have at most 1 faulty node in each disjoint path.

3.2 Point-to-multi-point protocol

The targeted point-to-multi-point protocol we present here is a simple extension of the point-to-point protocol in Section 3.1, in which we transmit the message msg from the sender to each member of the multicast group separately. Care should be taken, though, when picking a random polynomial during the transmission of the message. For each user, we need to choose a new random polynomial so as to avoid having any node in the graph, other than the sender or any of the receivers, obtaining more than one point per polynomial, which of course would lower the privacy threshold. One can easily see that this protocol inherits all the security of the underlying point-to-point protocol.

4 Empirical designs

We want to design specific communication networks for which the simple protocols from the previous section will be secure and efficient. If every pair of nodes is $(3t+1,k)$-connected or connected by a single edge, then our targeted group communication protocol will take k rounds and be t-reliable and t-private. We would like to understand the tradeoffs among n, t, and k. We focus our attention in this section on the realistic setting where the size of n is in the low 100's.

In order to better understand the trade-offs between the number of disjoint paths and their length as a function of the several parameters, such as average degree, in graph design, we run several experiments with random regular graphs. The reason for picking regular graphs is that they seem to be the best possible choice for our purposes since, in this case, all pairs of nodes are connected by the same number of disjoint paths

with very high probability [5], where this number is given by the average degree of a node. More specifically, in this section, we present several experiments with random regular graphs and analyze their behavior with respect to the number and length of disjoint paths between any pair of nodes when we vary the average degree of a each node.

Before proceeding any further, we should also mention that we did make experiments with other types of random graphs. In particular, we worked with graphs of the form $G_{N,p} = (V, E)$, in which $N = |V|$ denotes the total number of nodes and p denotes the probability of having an edge between any two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$, i.e. $\Pr[\,(\mathbf{v}_i, \mathbf{v}_j) \in E\,] = p$ for $i, j = 1, \ldots, N$. Such graphs are most likely not regular and connected with high probability for $p \geq \log N/N$ [1]. The average degree of a node is given by $p(N-1)$, but can vary significantly. Because of this unbalance, we did not expect such graphs to be very useful for our purposes as we are concerned with the worst possible case for connectivity. That is, the connectivity of a graph will be given by the pair with fewer number of disjoint paths connecting them. Indeed, our simulations with this type of random graphs showed that such unbalance plays an important role in the overall connectivity of a graph, significantly reducing its value in relation to that of a random regular graph with the same average degree.

4.1 Generating random regular graphs

Our generation of random regular graphs is not exact in that it may generate graphs which are "slightly" irregular. By that, we mean a graph where most of the nodes will have the same degree, the desired degree, but some may have a slightly higher degree. This approximation works fine for our purposes and does not have a significant effect on our results. In most cases, we noticed that the total number of edges which are in excess is less than 0.5 percent of the total number of edges.

The algorithm takes as input two parameters: N, the total number of nodes in the graph; and d, the average degree. It outputs a random graph $G_{N,d}$ where most nodes have degree d and a possibly few other nodes have a slightly higher degree. It works as follows. Let $V = \{1, 2, \ldots, N\}$ denote the set of nodes and let $N(i)$ denote the set of nodes which are neighbors of node i. For each node $i = 1, \ldots, N$, the algorithms first picks a node j at random from the set $V' = V - N(i)$, then checks if its degree is smaller than d and, if so, adds j to $N(i)$ and i to $N(j)$. If node j has degree greater than or equal to d, then we simply repeat the process by picking another node at random. In both cases, however, we remove node j from set V' to avoid picking the same node over and over. We iterate until the size of $N(i)$ reaches d or until the size of V' reaches 0, in which case we simply pick $d - |N(i)|$ nodes at random from the set $V - N(i)$ without worrying about their degrees.

4.2 Computing bounded disjoint paths

The problem of computing bounded disjoint paths is of extreme relevance and plays an important part in our protocols, as it will be seen later. In this problem, we are a given graph $G = (V, E)$, two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in V, and a bound k, and we want to compute the maximum number of vertex-disjoint paths between $\mathbf{v}_i$ and $\mathbf{v}_j$ of length at most k. As it was shown in [13], this problem as well as several of its variants are NP-complete except for small values of k. If we assume $P \neq NP$, then we can not hope for an efficient way to compute the exact solution to this problem. Instead, we opted for computing an approximate solution to this problem by using a greedy approach with the following heuristic.

The heuristic is to pick the shortest paths first. That is, for each pair of nodes, we first compute a shortest path between them, then verify if its length is smaller than the bound k, remove all the edges and nodes in it from the original graph, and repeat the process until no path of size at most k is found. The intuition behind this heuristic is that by removing the shortest paths first, we would be removing the least number of nodes from the graph and hence maximizing the total number of paths that we can find. One of the main advantages of using this heuristic is the efficiency of its computation. Each disjoint path can be computed in time $O(E + V)$, by using a breadth-first search algorithm [6]. On the other hand, this method is not guaranteed to perform well in all situations and can sometimes compute values which are very far from the

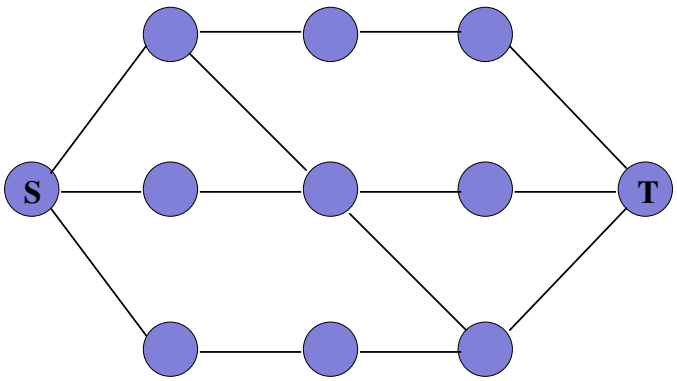


Fig. 2: An counter-example for the heuristic of picking shortest paths first.

exact ones. For instance, in the example in Fig. 2, depending on which shortest path we select first between nodes s and t, the total number of disjoint path can vary between 1 and 3.

The second heuristic we present is due to Reiter and Stubblebine [19] and is based on the idea of first picking paths with smallest degrees. By degree of a path, we mean the sum of the degree of all the nodes in the path. The intuition behind this heuristic is that paths with small degrees will most likely intercept a smaller number of paths, hence maximizing the total number of disjoint paths we can compute. As pointed out by [19], this method can be easily implemented by using a variant of Dijkstra's shortest path algorithm [**?**,6]. As one can see, this method works well, for example, for the graph given in Fig. 2, since now we would definitely not pick the path between s and t that goes from top to bottom. Nevertheless, its computation is somewhat more expensive than that of the previous method and can take time $O(V^2)$ per path.

It turns out that in the case we are interested in, which is that of regular graphs, both heuristic will be equally effective. Therefore, for performance purposes, we opted for using the first one in our simulations.

4.3 Some results on small graphs

In order to better understand how the number and length of disjoint paths vary as a function of the average degree and total number of nodes, we ran several experiments on graphs with up to 200 nodes. The main reason for choosing such small graphs is mostly due to the constraints involved in the simulation, which is very time-consuming. However, it is our hope that the knowledge gained with such small graphs can still be applied to larger ones. But even if this is not the case, we expect that results on small graphs can still be useful for several different applications where the total number of nodes is small, such as multicast protocols within small virtual private networks.

When working with random regular graphs, we are mainly interested in analyzing their connectivity. More specifically, we want to know for a given graph the minimum number of disjoint paths that we have between any two nodes and how long or short they are. For that purpose, we generate several random instances of regular graphs and analyze their average connectivity. In doing so, we also keep track of the graph with best connectivity among all the graphs we generated. These graphs are actually the most useful to us as they provide us with best network design (connectivity) for a given cost (degree). Unless stated otherwise, the total number of trials (instances) in our experiments is 100.

Fig. 3 describes the best (dotted line) and average (full line) cases for the connectivity in our simulation of regular graphs with 100 nodes for different upper bounds on the path length and for several average degrees. That is, for a given upper bound b and average degree d, its gives both the best and average cases for the minimum number of disjoint paths of length at most b between any pair of nodes in graphs with 100 nodes and average degree d. For example, we can see that for graphs with average degree $d = 10$, there are at least 8 disjoint paths of length at most 4 between any two nodes in the best case and at least about 7 disjoint

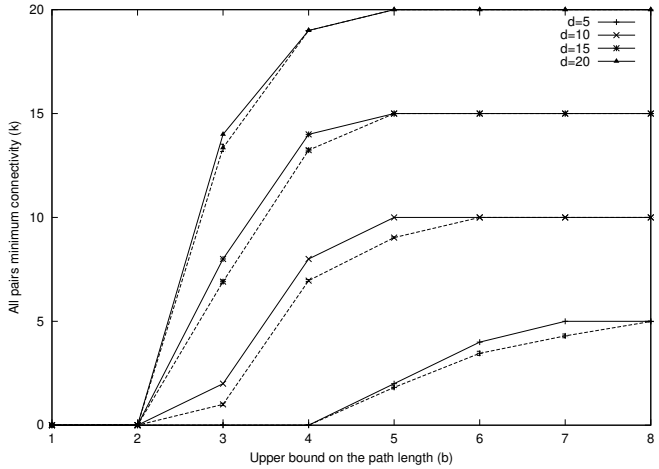


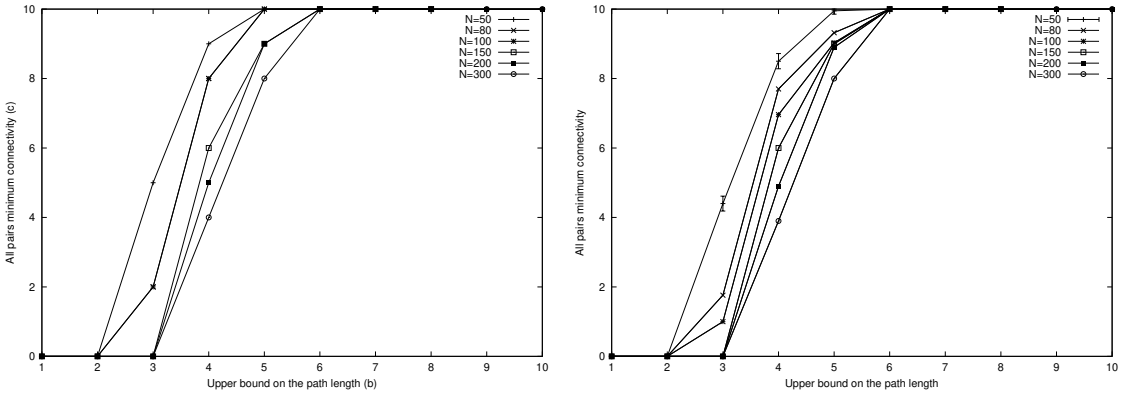
Fig. 3: Best (dotted line) and average (full line) cases for connectivity in regular graphs with 100 nodes for different upper bounds on the length of a path.

paths of length at most 4 on average (see Table 1). As one can see, the graph with best connectivity does not differ significantly from the average random graph and they occur quite often. This seems to be the case even when a larger number of instances is considered in the simulation. In fact, in almost all of our experiments, this difference was no greater than 2. As a result, even if we pick the graph with best connectivity from a smaller number of samples, say 10, we do not expect to be far from the optimum value for the connectivity. Therefore, from now on, we only consider the best case for connectivity as they are not hard to find.

Best and average cases for connectivity in regular graphs with 100 nodes and average degree 10			
Upper Bound	Best Connectivity	Average connectivity	Confidence Interval
1	0	0.00	0.0000
2	0	0.00	0.0000
3	2	1.00	0.0554
4	8	6.96	0.0474
5	10	9.03	0.0334
6	10	10.00	0.0000
7	10	10.00	0.0000
8	10	10.00	0.0000

Table 1: Best and average cases for the connectivity in regular graphs with 100 nodes for different upper bounds on the length of a path. A total of 100 random instances were considered in the simulation.

Fig. 4a and Fig. 4b show the effect of varying the total number of nodes on the connectivity of a graph while keeping the average degree d fixed and equal to 10. As expected, the larger the graph is, the longer the disjoint paths connecting their nodes are. However, this increase is not as pronounced as one may expect and, sometimes, not even existent. Note that as we increase the total number of nodes from 50 to 300, the curve for the connectivity only slightly shifts to the right. In fact, when the total number of nodes in the graph goes from 50 to 80 in Fig. 4a, there is no change in the connectivity and both curves are exactly the same. It is true that these values are just approximations and that the exact values for both of these curves



(a) Best case (b) Average case

Fig. 4: Best and average cases for connectivity in regular graphs of degree 10 for different upper bounds on the length of a path.

may actually present some discrepancy. Nevertheless, we do not expect any significant change in the overall picture. That is, we still expect the connectivity to increase very slowly with the total number of nodes in the graph.

Table 2 summarizes some of our results on graphs with 100 nodes. More specifically, for a given lower bound on the number of disjoint paths and upper bound on the length of these paths, it describes the minimum average degree for which we could find a regular graph meeting these criteria. For example, the entry 8 in the second column ($k = 5$) of the second row ($c = 7$) indicates that minimum degree for which we could find a graph, in which every two nodes is connected by at least 7 disjoint paths of length at most 5, is 8. In this particular case, our simple group communication protocol presented in Section 3 would be 2-resilient and 2-private and each transmission would take at most 5 rounds. Likewise, in order to have a secure protocol that tolerates up to 4 faults (last row) and runs in at most 4 rounds (third column), we only need a regular graph with average degree 14.

Lowest average degree achieved				
	Maximum length			
Connectivity	6	5	4	3
4	6	6	8	12
7	7	8	10	15
10	10	10	12	17
13	13	13	14	20

Table 2: Minimum degree achieved on regular graphs with 100 nodes for different bounds on the length and number of disjoint paths.

5 Analytical designs

In this section, we give a few specific constructions of graphs for which we can provide bounds on the number and length of disjoint paths between every pair of nodes.

The first construction, based on hypercubes, is very simple and yet able to produce graphs with a high number of disjoint paths between any pair of nodes. Moreover, these paths are also reasonably short and,

in the worst case, have length proportional to logarithm of total number of nodes in the graph. The main drawback of this method for constructing graphs resides in its lack of flexibility. The only freedom we have is to choose the dimension of the hypercube. See Section 5.1.

To overcome the lack of flexibility in hypercube-based designs, we suggest three other methods of construction. The second and third methods (resp. parallel hypercubes and M-ary hypercubes) are extensions to the basic hypercube construction. Both extensions are based on the idea of adding extra neighbors to each node in the graph, hence increasing the number of disjoint paths between nodes, while keeping the upper bound on the length of these paths the same. See Section 5.2 and Section 5.3.

The last method we present is based on balanced incomplete block designs and is somewhat more complex, but more flexible than the previous ones. For example, by increasing the average degree of a node, one can obtain graphs with much shorter disjoint paths. It is interesting to mention here, as it will be shown later, that the graph generated using this method is not necessarily regular. See Section 5.4.

5.1 Hypercubes.

A n-dimensional hypercube is a graph $G = (V, E)$, in which each node $\mathbf{v}_i \in V$ can be viewed as n-dimensional binary vector $(\mathrm{v}_i^{(1)}, \ldots, \mathrm{v}_i^{(n)})$ where $\mathrm{v}_i^{(j)} \in \{0, 1\}$, such that two nodes are connected by an edge if and only if their representing vectors differ in exactly 1 position.

Some basic properties of n-dimensional hypercubes are: (i) the total number of vertices is $N = |V| = 2^n$; (ii) the total number of edges is $2^{n-1} \times n$; and (iii) each node has exactly n neighbors.

The following theorem, whose proof is given in **??**, presents a more interesting property of n-dimensional hypercubes, which is related to what we want. More specifically, it says that in hypercubic graphs, the number and length of disjoint paths between any two nodes are both logarithmic in the total number of nodes in the graph. Corollary 1 follows easily from Theorem 1 by considering the worst possible case for the length of a path, which in this case occurs when $m = n - 1$.

Theorem 1. *Let $G = (V, E)$ be a n-dimensional hypercube and let $\mathbf{v}_i$ and $\mathbf{v}_j$ in V be any two nodes with Hamming distance m. Then, there exists n disjoint paths connecting these nodes, m of which have length m and $n - m$ of which have length $m + 2$.*

Corollary 1. *Let $G = (V, E)$ be a n-dimensional hypercube. Then, for any two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in V, there exists n disjoint paths connecting these nodes whose length are at most $n + 1$.*

As seen above, this construction is very simple and yet able to produce graphs with a high number of disjoint paths between any pair of nodes. Moreover, these paths are also reasonably short and, in the worst case, have length proportional to logarithm of total number of nodes in the graph. The main drawback of this method for constructing graphs resides in its lack of flexibility. The only freedom we have is to choose the dimension of the hypercube.

Proof. The proof is constructive in that we give a specific algorithm for computing all the n disjoint paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. From the definition of Hamming distance between two vectors, we know that the representing vectors of $\mathbf{v}_i$ and $\mathbf{v}_j$ differ in exactly m positions. We consider two cases in our construction: $m = n$ and $m < n$.

Let us start by considering the case where $m = n$. Without loss of generality, let us assume that $\mathbf{v}_i = (0, \ldots, 0)$ and $\mathbf{v}_j = (1, \ldots, 1)$. We can do so by simply changing the representation of every node $\mathbf{v}_k$ from $(\mathrm{v}_k^{(1)}, \ldots, \ldots, \mathrm{v}_k^{(n)})$ to $(\mathrm{v}_{k'}^{(1)}, \ldots, \ldots, \mathrm{v}_{k'}^{(n)})$ where $\mathrm{v}_{k'}^{(l)} = 1 - \mathrm{v}_k^{(l)}$ when $\mathrm{v}_i^{(l)} = 1$, and $\mathrm{v}_{k'}^{(l)} = \mathrm{v}_k^{(l)}$ when $\mathrm{v}_i^{(l)} = 0$. For simplicity, we slightly change the notation to represent a node as a binary string of length n. Hence, we have $\mathbf{v}_i = 0 \ldots 0 = 0^n$ and $\mathbf{v}_j = 1 \ldots 1 = 1^n$.

The algorithm takes as input a value n, the dimension of the hyper-cube graph, and outputs n disjoint paths between $\mathbf{v}_i = 0^n$ and $\mathbf{v}_j = 1^n$, It works as follows. Let $k = 1, \ldots, n$. The k-th path between $\mathbf{v}_i$ and $\mathbf{v}_j$ is given by $\mathbf{v}_i = 0^n \to 0^{k-1}10^{n-k} \to 0^{k-1}110^{n-k-1} \to \cdots \to 0^{k-1}1^{n-k+1} \to 10^{k-2}1^{n-k+1} \to 110^{k-3}1^{n-k+1} \to \cdots \to \mathbf{v}_j = 1^n$. Note that, except for nodes $\mathbf{v}_i$ and $\mathbf{v}_j$, no other node appears in more than one path. Moreover, all paths have length n.

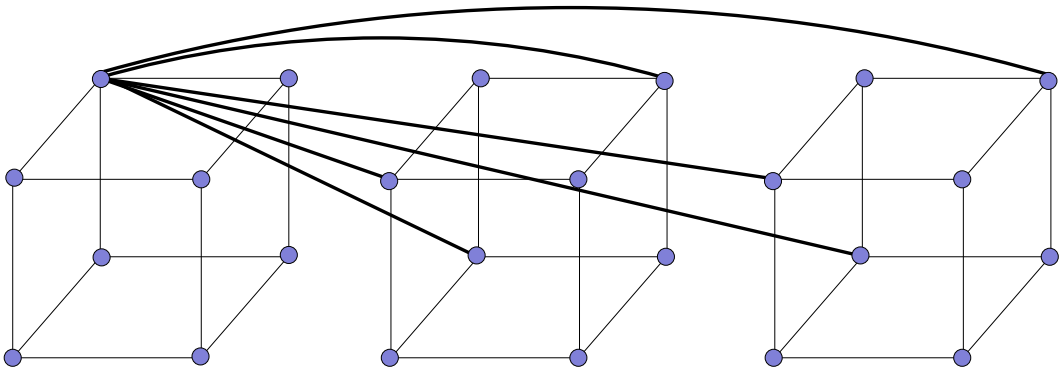


Fig. 5: An example of a parallel hypercube construction for $n = 3$ and $B = 3$. Only some cross-edges are shown.

Now, let us consider the case where $m < n$. Without loss of generality, we assume that $\mathbf{v}_i$ and $\mathbf{v}_j$ differ in the first m positions (dimensions). We can do so by simply permuting the positions in the representing vectors. By an argument similar to the one in the previous case, we can also assume that $\mathbf{v}_i = 0^n$ and $\mathbf{v}_j = 1^m 0^{n-m}$. The algorithm works as follows.

The algorithm takes as input two values: n, the dimension of the hyper-cube graph; and m, the Hamming distance between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. It outputs n disjoint between $\mathbf{v}_i = 0^n$ and $\mathbf{v}_j = 1^m 0^{n-m}$.

The algorithm works in two stages. In the first stage, m paths of length m are computed in a way similar to the previous case by simply ignoring the higher $n - m$ dimensions. Let $k = 1, \ldots, m$. The k-th path between $\mathbf{v}_i$ and $\mathbf{v}_j$ is given by $\mathbf{v}_i = 0^n \to 0^{k-1}10^{n-k} \to 0^{k-1}110^{n-k-1} \to \cdots \to 0^{k-1}1^{m-k+1}0^{n-m} \to 10^{k-2}1^{n-k+1}0^{n-m} \to 110^{k-3}1^{n-k+1}0^{n-m} \to \cdots \to \mathbf{v}_j = 1^m 0^{n-m}$. Note once again that, except for nodes $\mathbf{v}_i$ and $\mathbf{v}_j$, no other node appears in more than one path and that each path has length m.

In the second stage, the remaining $n - m$ paths are computed by moving further away from the target node in a first step (to maintain the disjointness property), and then getting closer to the target node in each subsequent step. Let $k' = 1, \ldots, n - m$. The $k' + m$-th path from $\mathbf{v}_i$ to $\mathbf{v}_j$ is as follows: $\mathbf{v}_i = 0^n \to 0^{m+k'-1}10^{n-m-k'} \to 10^{m+k'-2}10^{n-m-k'} \to 110^{m+k'-3}10^{n-m-k'} \to \cdots \to 1^m 0^{k'-1}10^{n-m-k'} \to \mathbf{v}_j = 1^m 0^{n-m}$. One can easily see that no node appears in more than one path and that each path has length exactly $m + 2$. This concludes the proof.

5.2 Parallel hypercubes

Parallel hypercubes are a collection of parallel hypercubes connected to each other in a special way. They are described by two parameters: B, the number of parallel hypercubes; and n, the dimension of each hypercube. Let $\mathbf{v}_{i,2}$ represent node i in hypercube j. The construction is as follows. First, each hypercube j itself is constructed like in the previous section. That is, if we represent a node $\mathbf{v}_{i,2}$ by a n-dimensional vector $(\mathbf{v}_{i,j}^{(0)}, \ldots, \mathbf{v}_{i,j}^{(n-1)})$, This node will have exactly n neighbors, those ones whose vectors differ from its vector in exactly one position. Hence, each hypercube will have a total of 2^n nodes. Last, if a node $\mathbf{v}_{i,2}$ is connected to a node $\mathbf{v}_{i',2}$ in hypercube j, then it is also connected to node $\mathbf{v}_{i',2}$ in hypercube $j' \neq j$. We call these edges *cross-edges*. Fig. 5 gives an example of a parallel hypercube construction for $n = 3$ and $B = 2$. For clarity, we only shoe the cross-edges (bolder lines) for one of the nodes in the left-most hypercube.

Let (B, n)-hypercube denote a graph $G = (V, E)$ consisting of B n-dimensional hypercubes. Some of its basic properties are: (i) the total number of vertices is $N = |V| = B \times 2^n$; (ii) the total number of edges is $B \times n \times 2^{n-1}$; and (iii) each node has exactly $B \times n$ neighbors.

The following theorem, whose proof can be easily derived from that of Theorem 1, describes the number and length of disjoint paths between any two nodes within Hamming distance m of each other in a (B, n)-

hypercube graph. Corollary 2 follows easily from it by considering the worst possible case for the length of a path, which occurs when $m = n - 1$.

Theorem 2. *Let $G = (V, E)$ be a (B, n)-hypercube and $\mathbf{v}_i$ and $\mathbf{v}_j$ in V be any two nodes with Hamming distance m. Then, there exists $B \times n$ disjoint paths connecting these nodes, $B \times m$ of which have length m and $B \times (n - m)$ of which have length $m + 2$.*

Corollary 2. *Let $G = (V, E)$ be a (M, n)-hypercube. Then, for any two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in V, there exists $(M - 1)n$ disjoint paths connecting these nodes of length at most $n + 2$.*

5.3 M-ary hypercubes

To overcome the lack of flexibility in the hypercube-based design in Section 5.1, we suggest here an extension to that construction. This extension is based on the idea of adding extra neighbors to each node in the graph, hence increasing the number of disjoint paths between nodes, while keeping the upper bound on the length of these paths the same.

As seen in Section 5.1, each node $\mathbf{v}_i$ in a n-dimensional hypercube can be seen as n-dimensional *binary* vector $(\mathrm{v}_i^{(1)}, \ldots, \mathrm{v}_i^{(n)})$, where $\mathrm{v}_i^{(j)} \in \{0, 1\}$. To allow for more neighbors, while maintaining the maximum length of each disjoint path the same, the idea is to replace binary vectors by M-ary vectors. That is, each element $\mathrm{v}_i^{(j)}$ of a node $\mathbf{v}_i$ is now a member of the set $\{0, \ldots, M-1\}$. As before, we still have an edge between two nodes if and only if their representing vectors differ in exactly 1 position.

Let (M, n)-hypercube denote a M-ary n-dimensional hypercube graph $G = (V, E)$. Some of its basic properties are: (i) the total number of vertices is $N = |V| = M^n$; (ii) the total number of edges is $(M - 1) \times n \times M^n \div 2$; and (iii) each node has exactly $(M - 1) \times n$ neighbors.

The following theorem states precisely the number and length of disjoint paths between any pair of nodes in the graph. Its proof is given in **??**. Corollary 3 follows easily from Theorem 1 by considering the worst possible case for the length of a path, which is $m = n - 1$ in this case.

Theorem 3. *Let $G = (V, E)$ be a (M, n)-hypercube and let $\mathbf{v}_i$ and $\mathbf{v}_j$ in V be any two nodes with Hamming distance m. Then, there exists $(M - 1)n$ disjoint paths connecting these nodes, m of which have length m and $(M - 1)n - m$ of which have length $m + 2$.*

Corollary 3. *Let $G = (V, E)$ be a (M, n)-hypercube. Then, for any two nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ in V, there exists $(M - 1)n$ disjoint paths connecting these nodes of length at most $n + 2$.*

Proof. As in the proof of Theorem 1, we give a specific algorithm for computing $(M - 1)n$ disjoint paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ with the specific length property stated in the theorem. As in that construction, we also consider two possible cases in our construction: $m = n$ and $m < n$.

Let us start by considering the case where $m = n$. Without loss of generality, let us assume that $\mathbf{v}_i = (0, \ldots, 0)$ and $\mathbf{v}_j = (1, \ldots, 1)$, which can be achieved by a simple change of representation. For simplicity, instead of using the vector notation for a node, we use a M-ary string of length n to represent a node. Hence, $\mathbf{v}_i = 0 \ldots 0 = 0^n$ and $\mathbf{v}_j = 1 \ldots 1 = 1^n$.

The algorithm takes as input two values: n, the dimension of the M-ary hyper-cube graph; and M, the size of the alphabet from which each point coordinate is drawn; It outputs $(M - 1)n$ disjoint paths between $\mathbf{v}_i = 0^n$ and $\mathbf{v}_j = 1^n$, It works as follows.

The construction of the first n paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ can be described in a straight forward manner. The idea is to start with node $\mathbf{v}_i$ and each round move one step closer to node $\mathbf{v}_j$ by selecting a node whose hamming distance to node $\mathbf{v}_j$ is smaller than that of the current node. Let $k = 1, \ldots, n$. Then the first n paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ are as follows: $\mathbf{v}_i = 0^n \to 0^{k-1}10^{n-k} \to 0^{k-1}110^{n-k-1} \to \cdots \to 0^{k-1}1^{n-k+1} \to 10^{k-2}1^{n-k+1} \to 110^{k-3}1^{n-k+1} \to \cdots \to \mathbf{v}_j = 1^n$. Note that, except for nodes $\mathbf{v}_i$ and $\mathbf{v}_j$, no other node appears in more than one path. Moreover, all paths have length n.

The construction of the last n paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ is also very similar to that of the first n ones. The main difference is that in the first step we select a node whose Hamming distance to node $\mathbf{v}_j$ is

still n. The reason for doing so is that all other nodes closer to node $\mathbf{v}_j$ were already used in the construction of the first n paths. After that, the idea is the same and we try to move one step closer to node $\mathbf{v}_j$ in each round. Let $k = 1, \ldots, n$ and let $\delta = 2, \ldots, M-1$. Then, the last $(M-2)n$ paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ are as follows: $\mathbf{v}_i = 0^n \to 0^{k-1}\delta 0^{n-k} \to 0^{k-1}\delta 10^{n-k-1} \to 0^{k-1}\delta 110^{n-k-2} \to \cdots \to 0^{k-1}\delta 1^{n-k} \to 10^{k-2}\delta 1^{n-k} \to 110^{k-3}\delta 1^{n-k} \to \cdots \to 1^{k-1}\delta 1^{n-k} \to \mathbf{v}_j = 1^n$. Again, note that, except for nodes $\mathbf{v}_i$ and $\mathbf{v}_j$, no other node appears in more than one path. Moreover, all paths have length $n+2$.

Now, let us consider the case where $m < n$. Without loss of generality, we assume that $\mathbf{v}_i$ and $\mathbf{v}_j$ differ in the first m positions (dimensions) and are equal to 0^n and $1^m 0^{n-m}$, respectively. This can be accomplished by a simple change in the representation of a node. The algorithm works as follows.

The algorithm takes as input three values: n, the dimension of the M-ary hyper-cube graph; M, the size of the alphabet from which each point coordinate is drawn; and m, the Hamming distance between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. It outputs $(M-1)n$ disjoint between $\mathbf{v}_i = 0^n$ and $\mathbf{v}_j = 1^m 0^{n-m}$.

The algorithm works in two stages. In the first stage, $(M-1)m$ paths are constructed in a similar way to that of the previous case by simply ignoring the higher $n-m$ dimensions. Let $k = 1, \ldots, m$. The first m paths between $\mathbf{v}_i$ and $\mathbf{v}_j$ are given by $\mathbf{v}_i = 0^n \to 0^{k-1}10^{n-k} \to 0^{k-1}110^{n-k-1} \to \cdots \to 0^{k-1}1^{m-k+1}0^{n-m} \to 10^{k-2}1^{n-k+1}0^{n-m} \to 110^{k-3}1^{n-k+1}0^{n-m} \to \cdots \to \mathbf{v}_j = 1^m 0^{n-m}$. Note that each of these paths has length m. Let $k = 1, \ldots, m$ and let $\delta = 2, \ldots, M-1$. The last $(M-2)m$ paths between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$ are as follows: $\mathbf{v}_i = 0^n \to 0^{k-1}\delta 0^{n-k} \to 0^{k-1}\delta 10^{n-k-1} \to 0^{k-1}\delta 110^{n-k-2} \to \cdots \to 0^{k-1}\delta 1^{m-k}0^{n-m} \to 10^{k-2}\delta 1^{m-k}0^{n-m} \to 110^{k-3}\delta 1^{m-k}0^{n-m} \to \cdots \to 1^{k-1}\delta 1^{m-k}0^{n-m} \to \mathbf{v}_j = 1^m 0^{n-m}$. Note that all these paths have length $m+2$.

In the second stage, the remaining $(M-1)(n-m)$ paths are constructed by first moving further away from the target node in an initial step (using one of the higher $(n-m)$ dimensions) and then moving closer to the target node in each subsequent step. Let $k' = 1, \ldots, n-m$ and let $\delta = 1, \ldots, M-1$. The remaining paths from $\mathbf{v}_i$ to $\mathbf{v}_j$ are as follows: $\mathbf{v}_i = 0^n \to 0^{m+k'-1}\delta 0^{n-m-k'} \to 10^{m+k'-2}\delta 0^{n-m-k'} \to 110^{m+k'-3}\delta 0^{n-m-k'} \to \cdots \to 1^m 0^{k'-1}\delta 0^{n-m-k'} \to \mathbf{v}_j = 1^m 0^{n-m}$. One can easily see that each path has length exactly $m+2$. This concludes the proof.

5.4 Balanced incomplete block designs (BIBD)

Balanced Incomplete Block Designs (BIBD) are often used when it comes to the design of regular mathematical structures. Here, we are interested in using such designs to help us construct graphs for which we can give bounds on the number and length of disjoint paths between any two nodes in the graph. But before we proceed with the construction of graphs, let us first define BIBDs more formally.

Definition 5. *A Balanced Incomplete Block Design (BIBD) is a pair $(V, \mathcal{B})$ where V is a set of N elements and $\mathcal{B}$ is a collection of b k-subsets (blocks) of V such that each element of V appears in exactly r subsets and each pair of elements of V appears in exactly λ subsets.*

Note that these parameters are not totally independent. In fact, in order to guarantee the existence of a BIBD with parameters (N, b, r, k, λ), we require that

$$rN = b\ k; \quad \text{and} \tag{1}$$

$$r(k-1) = \lambda(N-1); \tag{2}$$

As a result, we can also refer to BIBDs with parameters (N, b, r, k, λ) as (N, k, λ)-designs since the parameters b and r can be easily derived from the other three parameters. For other conditions for the existence of BIBDs, please refer to [4].

Here is an example taken from from [4] of a $(11, 5, 2)$-design, i.e. a set of subsets of size 5 drawn from a set of 11 elements such that each pair of elements appears in exactly 2 subsets: $\{\{0,1,2,3,7\}, \{0,1,4,5,6\}, \{0,2,5,8,9\}, \{0,3,6,8,a\}, \{0,4,7,9,a\}, \{1,2,4,8,a\}, \{1,3,5,9,a\}, \{1,6,7,8,9\}, \{2,3,4,6,9\}, \{2,5,6,7,a\}, \{3,4,5,7,8\}\}$.

Now, let us go back to the construction of graphs using BIBDs. Remember that we want to ensure that every two nodes, not directly connected by an edge, are connected by a minimum number of disjoint paths,

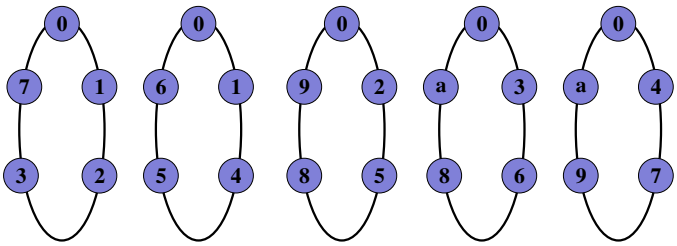


Fig. 6: Some of the virtual rings generated from the $(11, 5, 2)$-design in the text.

say t, each of length at most k. Each block (subset) in a BIBD will be mapped into a virtual ring over the network. Despite its simplicity, this construction has several advantages. First, each pair of nodes in a ring, not directly connected by an edge, will be connected by 2 disjoint paths of length at most $k-2$. Second, if each triple of nodes appears in exactly 1 block, then every pair of nodes, not directly connected by an edge, will be connected by at least 2λ paths. Such types of block designs can be obtained, for example, by using sub-designs of Steiner families [4]. Henceforth, we assume this is true in all block designs we consider. Third, the computation of disjoint paths is straight forward and does not depend on the total number of nodes in the graph.

We should mention here that a similar idea of constructing graphs from block designs was used in [23], but they were mainly interested in the case where $\lambda = 1$. In our case, however, we want to have as many disjoint paths as possible between any pair of nodes and λ will be always greater than 1.

Before going any further, let us take a look at a more concrete example. Consider the $(11, 5, 2)$-design given above as an example. Fig. 6 shows some of the rings we can obtain by mapping each block into a ring over a network with 11 nodes. Note that some edges appears in more than one virtual ring. Because the block size, k, is 5, each pair of nodes in the ring, not directly connected by an edge, will be connected by 2 disjoint paths of length at most $3 = k-2$. Because each pair of nodes appears in exactly $\lambda = 2$ blocks, there is at least $2\lambda = 4$ disjoint paths between any two nodes, not directly connected by a node.

Fig. 7 depicts the entire graph obtained by mapping all 11 blocks into virtual rings. Consider, for instance the pair of nodes 0 and 5. By using the virtual rings, we can easily draw the following 4 disjoint paths between nodes 0 and 5: $0 \to 1 \to 4 \to 5$; $0 \to 6 \to 5$; $0 \to 2 \to 5$; and $0 \to 9 \to 8 \to 5$. Note, however, that, by using the other remaining nodes, we can still draw 2 more disjoint paths: $0 \to 3 \to 5$; and $0 \to 7 \to 5$. Moreover, by not using node 8 in one of the paths above (we could have gone directly from node 9 to node 5), we could have obtained yet another disjoint path, $0 \to a \to 8 \to 5$, and achieved the maximum number of disjoint paths between 0 and 5.

We need to make two observations here. First, the graph obtained by this kind of transformation is not unique and depends on the way we arrange the nodes in a ring. In fact, even the total number of edges in a graph can be different depending on which heuristic we use for constructing the rings. As a result, the degree d of a node is also not unique and can vary between $2r$ (since we can have 2 new edges per node per ring and each node belongs to exactly r rings) and $2r/\lambda$ (if, whenever a node is paired with a node in some ring, the same are paired together in all the other $\lambda - 1$ rings in which they appear). Likewise, the total number of edges will vary between Nr and Nr/λ. Second, both the lower bound on the number of disjoint paths between any pair of nodes and the upper bound on the length of these paths are conservative. In fact, as in the example in Fig. 7, the lower bound on the number of disjoint paths, 2λ, can be much smaller than the maximum value we could achieve in practice, which is usually in the order of the minimum of the degrees of the source and target nodes. From the analysis above, we know this number is at least $2r/\lambda$.

Table 3 summarizes the range of values for a graph $G = (V, E)$ constructed from a (N, k, λ)-design.

In order to compare the construction based on hypercubes from Section 5.1 to the current one, let us consider the best possible scenario for BIBD-based constructions. That is, let us assume that the graph generated by the current construction is regular with average degree $d = 2r/\lambda$ and that $r = \lambda^2$ so that the

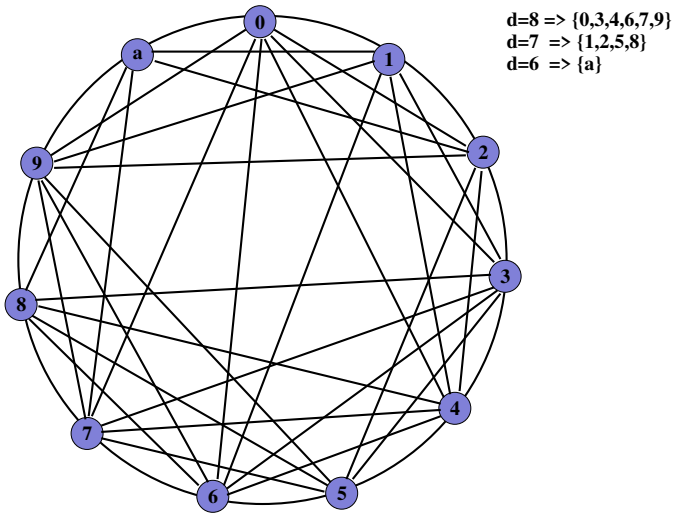


Fig. 7: Graph generated from the $(11, 5, 2)$-design in the text.

Parameter	Value range
Total number of nodes ($\|V\|$)	N
Total number of edges in the graph ($\|E\|$)	$N\ r/\lambda \leq \|E\| \leq N\ r$
Degree of a node (d)	$2\ r/\lambda \leq d \leq 2\ r$
Number of disjoint paths (t)	$t \geq 2\lambda$
Maximum length of disjoint path	$k - 2$

Table 3: Parameters of a graph $G = (V, E)$ based on a (N, k, λ)-design.

lower bound on the number of disjoint paths between any two nodes, 2λ, matches its upper bound, $d = 2r/\lambda$. Then, by (2), we get

$$d(k-1) = 2(N-1). \tag{3}$$

That means that the product of the degree of a node (which also determines the maximum number of disjoint paths) and the length of a disjoint path is in the order of N. For example, if we want disjoint paths with length bounded by a constant, we would need graphs where the average degree is linear in the total number of nodes.

Let us first consider the case where $k = \log N$ in (3). This is about the same guarantee we could get for the upper bound on the length of disjoint paths by using hypercube-based constructions. As one can see, the average degree of a node d for BIBD-based constructions is in the order of $N/\log N$, which is much larger than $d = \log N$ achieved by the hypercube-based construction of Section 5.1. On the other hand, the total number of disjoint paths between any two nodes would be much higher in BIBD-based constructions than in hypercube-based ones.

Now consider the case where $d = \log N$ in BIBD-based constructions. As one can see from (3), the upper bound on the path length, k, is in the order of $N/\log N$, which is much higher than that achieved by hypercube-based constructions.

From the comparison above, it seems that hypercube-based constructions outperforms BIBD-based constructions with respect to both the upper bound on the length of disjoint paths and the average degree of a node. The main reason for this, as showed by the specific example in Fig. 7, seems to be that the bounds on the path length and number of disjoint paths are very conservative. This suggests that in order to have a more realistic comparison between the two constructions, one would have to first compute, even if approxi-

mately, the bounded disjoint paths for a given BIBD-based construction and only then compare the values with those of a hypercube-based construction.

One main advantage of BIBD-based constructions over basic hypercube-based ones is the freedom of choosing the parameters in the network design in the former. As pointed out before, we only have the freedom to choose one parameter in the latter, which is the dimension of the hypercube. However, the same is not true for extended hypercubes constructions and is no longer clear in this case whether the gain in flexibility in BIBD-based constructions justifies the not-so-tight bounds on the number and length of disjoint paths.

6 Future Work

One important direction for future work is to design protocols for point-to-group communication that achieve sublinear bit complexity in the size of the targeted group. It seems intuitively plausible that there are ways to improve on our simple replication of point-to-point communication.

We would also like to have better tools for designing good communication networks on which to execute our protocols. Empirically, we may have only scratched the surface. The set of all possible networks is vast, and it is not clear where and how to search. Analytically, our designs based on hypercubes and BIBD's are promising, but much remains to be discovered.

Lastly, we remark that it is not clear that the search for better protocols and the search for better network designs should be treated separately. There may be a unified approach that will yield better results.

References

1. N. Alon and J. Spencer. *The Probabilistic Method*. Wiley-Interscience Publication, New York, 1992.
2. A. Beimel and M. Franklin. Reliable communication over partially authenticated networks. *Theoretical Computer Science*, 220(1):185–210, 1999.
3. E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470. Issued Dec. 1986.
4. C. Colborn and J. Dinitz, editors. *The CRC Handbook of Combinatorial Designs*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, fifth edition, 1996.
5. C. Cooper, A. Frieze, and B. Reed. Random regular graphs of non-constant degree. Pre-Print, June 2000.
6. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.
7. E. H. D. Wallner and R. Agee. Key management for multicast: Issues and architectures. *ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt*, 1998.
8. R. Diestel. *Graph theory*. Springer-Verlag, New York, 1997. Translated from the 1996 German original.
9. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993.
10. A. Fiat and M. Naor. Broadcast encryption. *Proc. Crypto '93*, pages 480–491, 1993.
11. M. Franklin and R. Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, 2000.
12. Z. Galil and X. Yu. Short length versions of menger's theorem. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing (STOC '95)*, pages 499–508, New York, May 1995. ACM.
13. A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–86, 1982.
14. J. Kleinberg and R. Rubinfeld. Short paths in expander graphs. In *37th Annual Symposium on Foundations of Computer Science*, pages 86–95, Burlington, Vermont, 14–16 Oct. 1996. IEEE.
15. L. Lovasz, V. Neumann-Lara, and M. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–76, 1978.
16. K. Menger. Allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.
17. S. Nikoletseas, K. Palem, P. Spirakis, and M. Yung. Short vertex disjoint paths and multiconnectivity in random graphs: Reliable network computing. *Lecture Notes in Computer Science*, 820:508–??, 1994.
18. S. E. Nikoletseas and P. G. Spirakis. Expander properties in random regular graphs with edge faults. In *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *lncs*, pages 421–432, Munich, Germany, Mar. 1995. Springer.

19. M. Reiter and S. Stubblebine. Path independence for authentication in large-scale systems. In *Proceedings of the Fourth ACM Conference on Computer and Communication Security*, pages 57–66, Apr. 1997.
20. D. Ronen and Y. Perl. Heuristics for finding a maximum number of disjoint bounded paths (telecommunication networks). *Networks*, 14(4):531–44, 1984.
21. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
22. J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–45, 1974.
23. B. Yener, Y. Ofek, and M. Yung. Combinatorial design of congestion-free networks. *IEEE/ACM Transactions on Networking*, 5(6):989–1000, Dec. 1997.