Communication Complexity With Synchronized Clocks

Russell Impagliazzo
Institute for Advanced Study (Princeton, NJ)
& University of California, San Diego
russell@cs.ucsd.edu

Ryan Williams
IBM Almaden Research Center
San Jose, CA
ryanwill@us.ibm.com

***Abstract*—We consider two natural extensions of the communication complexity model that are inspired by distributed computing. In both models, two parties are equipped with synchronized discrete clocks, and we assume that a bit can be sent from one party to another in one step of time. Both models allow implicit communication, by allowing the parties to choose whether to send a bit during each step. We examine trade-offs between time (total number of possible time steps elapsed) and communication (total number of bits actually sent).**

In the *synchronized bit model*, we measure the total number of bits sent between the two parties (e.g., email). We show that, in this model, communication costs can differ from the usual communication complexity by a factor roughly logarithmic in the number of time steps, and no more than such a factor.

In the *synchronized connection model*, both parties choose whether or not to open their end of the communication channel at each time step. An exchange of bits takes place only when both ends of the channel are open (e.g., instant messaging), in which case we say that a *connection* has occurred. If a party does not open its end, it does not learn whether the other party opened its channel. When we restrict the number of time steps to be polynomial in the input length, and the number of connections to be polylogarithmic in the input length, the class of problems solved with this model turns out to be roughly equivalent to the communication complexity analogue of P^{NP} ([BFS86]). Using our new model, we give what we believe to be the first lower bounds for this class, separating P^{NP} from $\Sigma_2 \cap \Pi_2$ in the communication complexity setting.

Although these models are both quite natural, they have unexpected power, and lead to a refinement of problem classifications in communication complexity.

I. Introduction

Sometimes silence speaks. Not receiving a message can convey information. Here we consider different interpretations of this implicit communication in a communication complexity framework. We assume two parties, Alice and Bob, with synchronized discrete clocks. At each step of time, each party chooses to send a bit to the other party, or not. In such synchronized communication models, we can distinguish between two resources for a protocol: the total bits exchanged between parties, and the total number of time steps available for communication. In the normal communication complexity model, these are considered identical. In our models, it is natural to ask about trade-offs between the two.

This material is based on work supported by NSF grants CCF-0832797 and DMS-0835373.

We consider two versions of *synchronized communication complexity*. In the first version of the model, called the *synchronized bit model*, we simply count all bits sent between the two parties. In the second version, called the *synchronized connection model*, a transfer of bits between parties takes place during a time step only when both parties try to send a bit during that step. (We do not allow participants to learn whether the other party was attempting communication if no communication occurs. Allowing such knowledge would also be natural, e.g., modeling telephone charge policies where an unanswered telephone ring is free. However, this would also be sufficiently powerful that communication would be essentially free; one player could code their input into the sequence of rings and silences.) That is, in the synchronized connection model, we count the number of successful "connections" between the parties.

The two models, although related in spirit, are very different in nature. We show that the synchronized bit model is equivalent to normal communication complexity up to factors logarithmic in the time, but can help by such logarithmic factors. (In fact, in constant round protocols, it always saves such a factor.)

The synchronized connection model is interesting in itself, but also as a tool to characterize the communication classes corresponding to the polynomial time hierarchy ([BFS86]). Separating such classes has become more important recently because such separations imply that the corresponding time classes cannot be proved equal using algebrizing methods ([AW08], [IKK09]). In particular, these papers give general techniques for converting any communication complexity separation into an oracle construction separating the corresponding complexity classes and having additional algebraic structure. In the case of [AW08], this involves separating the larger class relative to an oracle from the smaller class relative to an algebraic extension of the oracle. In [IKK09], the authors give constructions of separating oracles O that also satisfy an axiom allowing interpolation of NP^O witnesses to low-degree polynomials. (See Appendix B for a sketch of how our communication complexity lower bounds yield such oracle constructions.)

The synchronized connection model turns out to be roughly equivalent to the deterministic second level of the communication hierarchy, i.e., the equivalent of P^{NP} for communication complexity [BFS86]. This connection shows (via known lower bounds) that the connection model can be arbitrarily more efficient than standard communication complexity, even when restricted to linear total time. (For example, the set disjointness problem is solvable with one connection in linear time, but requires linear communication in the standard model.) We show lower bounds in this model for a variety of explicit functions of the form $T > 2^{\Omega(n/c)}$ where T is the total time the protocol takes, and c is the maximum number of synchronized connections allowed. This gives (to the best of our knowledge) the first lower bound for P^{NP}-style communication complexity: any protocol that makes q queries to languages with nondeterministic complexities at most t satisfies $qt = \Omega(n)$. (In particular, either q or t is $\Omega(\sqrt{n})$.) We can use the same method to prove lower bounds for functions in $\Sigma_2^{cc} \cap \Pi_2^{cc}$, the communication complexity analogues of the second levels of the polynomial time hierarchy. This shows that there is no algebrizing technique that collapses the second level of the hierarchy, or even proves equivalence between the above classes.

A. Prior Work

Similar models where there is a trade-off between bits and time, as well as models that take advantage of synchronization, have been studied in distributed computing for years, e.g. [GHS83], [Lam84], [Awe85], [FL87], [Lis93], [Lyn96]. This work is concentrated on studying a large number of parties, some of whom may be faulty, computing functions on relatively small inputs and outputs. Another recent and related model is the *pulse communication model* [FO05], [DFO06] from sensor network research, which allows players to be silent under a clocked model but still focuses on computing very simple functions with many participants. That is, in these models, problems scale with respect to the number of participants, rather than with the size of inputs.

In this paper we consider time/bit tradeoffs in the setting of communication complexity, where the emphasis is on few players with large inputs, and the problems scale with respect to the input size. We have not yet found applications of the many techniques in distributed computing to our communication complexity models. However we do expect that further study will lead to interesting connections between distributed computing and our models.

There are a few brief remarks in the complexity literature that discuss problems related to our synchronized bit complexity. (For example, see Papadimitriou and Sipser [PS84], pp.262–263, and Arora and Barak [AB09], pp.271–272.) However, none of them explore these problems in any depth, to our knowledge.

Finally, we note that in work of [HIKNV04], the authors study a time-communication tradeoff where the *runtimes of the computational entities* are measured against the amount of communication necessary to solve problems. Here we do not bound the computational power of the parties (just as in classical communication).

II. Background

We assume some familiarity with communication complexity. Here we briefly recall some key definitions.

We recall the definition of a *protocol tree* from two-party communication complexity, following [KN97]. A protocol tree for n-bit inputs is a binary tree where each node has either zero or two children. The nodes correspond to all the possible "histories" or states of the communication. The two child edges from a parent node are distinctly labelled by 0 and 1, respectively. A node with no children (a leaf) has a label 0 or 1. Every inner node v is labelled by x or y, along with a function $f_v : \{0,1\}^n \to \{0,1\}$. The *communication complexity* of a protocol tree is its depth. The *number of rounds* of a protocol tree is 1 plus the maximum number of times that the node label changes (from x to y or from y to x), along any path from the root to a leaf. (Intuitively, this is the number of times that the speaking party switches from one player to the other, plus one for the first speaking party.)

A protocol is executed as follows. When one party is given an input $x \in \{0,1\}^n$ and the other is given $y \in \{0,1\}^n$, the *current node* v is initialized to be the root node. If v has label x, then $f_v(x)$ is computed (otherwise, $f_v(y)$ is computed); this corresponds to a local computation done by the party holding x (or the party holding y, respectively). Let b be the output of f_v. The child edge out of v labelled b is followed, and the protocol continues with the new current node. If the current node is a leaf, then the final output of the protocol is the label of the leaf.

A *nondeterministic protocol tree* is defined similarly, except that the f_v at a node v may not be a function: it may allow a choice of either the 0 or 1 child edge out of v. We say that a nondeterministic protocol outputs 1 if there exists a set of such choices that lead from the root node to a leaf labelled 1.

Let $\Pi = \{\Pi_n\}$ be a communication problem, where each Π_n is a function from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$. (We sometimes identify Π_n with the subset of $\{0,1\}^n \times \{0,1\}^n$ on which the function is 1.) The deterministic communication complexity of a problem Π is the function $f : \mathbb{N} \to \mathbb{N}$ such that $f(n)$ is the minimum depth of any protocol tree that computes Π on n-bit inputs (i.e., computes Π_n). Nondeterministic communication complexity is defined analogously, but with nondeterministic protocol trees. $D(\Pi)$ denotes the deterministic communication complexity of Π as a function of the input length n, and $N(\Pi)$ denotes nondeterministic communication complexity. P^{cc} is the class of communication problems Π such that $D(\Pi) \leq \text{poly}(\log n)$. The class NP^{cc} consist of those Π that satisfy $N(\Pi) \leq \text{poly}(\log n)$. Similar extensions of the protocol concept lend to definitions of $coNP^{cc}$, Σ_k^{cc}, and so on [BFS86].

III. The Synchronized Bit Model

We begin with the *synchronized bit model*, where two communicating parties have the option to send either a bit or nothing at all during a time step. We model this as communication with a ternary alphabet $\{0, 1, \star\}$ where $\star$ corresponds to *don't send* and the parties are only 'charged' for the number of bits sent.

A *synchronized protocol tree* in the synchronized bit model is defined analogously to protocol trees in the usual two-party communication model, except that every inner node may have up to three child edges, distinctly labeled 0, 1, and $\star$. Define the *running time* of a synchronized protocol to be the depth of the tree, and define the *synchronized bit complexity* of a protocol to be the maximum number of 0-edges and 1-edges on any path from the root to the leaf. By definition, the best possible "running time" for a communication problem is lower bounded by its deterministic communication complexity. In our framework, we allow the running time of our protocols to be polynomials in n (where n is the total length of the inputs), and focus on minimizing the cost of communication.

It is now appropriate to define complexity classes for problems computed efficiently in our model.

Definition 3.1: $TB[t(n), b(n)]$ is the class of communication problems Π with the property that there is a constant $c \geq 1$ such that for all n, there is a synchronized protocol tree for Π_n with running time at most $c \cdot t(n) + c$, and synchronized bit complexity at most $c \cdot b(n) + c$.

Definition 3.2: $PB(\Pi)$, the *polytime bit complexity* of Π, is the set of functions $\{b_k(n)\}$, where $b_k(n)$ is the minimum synchronized bit complexity of any protocol for Π_n with running time at most $n^k + k$. For a function $f : \mathbb{N} \to \mathbb{N}$, we say $PB(\Pi) \leq O(f(n))$ if there is a k such that $b_k(n) \leq O(f(n))$. We say that $PB(\Pi) \geq \Omega(f(n))$ if for all k, $b_k(n) \geq \Omega(f(n))$.

Note that the sequence $\{b_k\}$ is non-decreasing in the sense that for all k and n, $b_{k+1}(n) \leq b_k(n)$.

A. Relationship with Classical Communication

We now show how the synchronized bit model relates to the classical deterministic model. We first show how to reduce the message complexity of deterministic communication protocols using the synchronized bit model, by taking more time. Most of our proofs have a divide-and-conquer flavor: we split the communication history into pieces and perform some simulation on each piece.

Theorem 3.1: Suppose Π can be solved by a deterministic protocol with r rounds of messages, and a message has length at most ℓ_i in the ith round. Then for all $t(n) \geq 2$, there is a synchronized protocol for Π with bit complexity $O\left(r + \sum_{i=1}^{r} \lceil \ell_i/(\log t) \rceil\right)$ and time $O\left(t \cdot (r + \sum_{i=1}^{r} \lceil \ell_i/(\log t) \rceil)\right)$.

Proof: The synchronized protocol works as follows. During each round, the speaking party divides its ℓ-bit message into blocks $b_1, \ldots, b_k$ of length $\log t$, where $k = \max\{1, \lceil \ell/(\log t) \rceil\}$. For $i = 1, \ldots, k$, a $(\log t)$-bit counter c (initially set to 0) is incremented for t time steps. While $c \neq b_i$, the speaking party sends $\star$. During the step when $c = b_i$, the speaking party sends 1 if $i < k$, and 0 if $i = k$. (Note the 0 communicates that the end of the message has been reached, so the other party need not know the length of a message.) Thus an ℓ-bit message is communicated with at most k bits, taking at most $O(t \lceil \ell/(\log t) \rceil)$ time in each round. ■

Hence the $O\left(\sum_{i=1}^{r} \ell_i\right)$ communication complexity of Π can be reduced to $O(r)$ synchronized bit complexity.

Corollary 3.1: A b-bit string can be sent from one party to the other in $O(2^{b/c})$ time with c bits of communication.

That is, a b-bit string can be sent in 2^b time and 1 bit, or in $\text{poly}(b)$ time and $O(b/\log b)$ bits.

Corollary 3.2: For every Π, $PB(\Pi) \leq O(n/\log n)$. [1]

The fact that every communication problem has $O(n/\log n)$ polytime bit complexity raises an interesting question. How does the deterministic communication complexity $D(\Pi)$ relate to the bit complexity of Π over polynomial time protocols? Is $PB(\Pi) \leq O(D(\Pi)/\log D(\Pi))$ for all Π? If the optimal protocol for Π can be implemented within at most $O(D(\Pi)/\log D(\Pi))$ rounds, then the answer is yes.

Theorem 3.2: Let $D_r(\Pi)$ denote the minimum communication complexity required to solve Π that uses at most r rounds of messages. Then $PB(\Pi) \leq O(D_r(\Pi)/\log n + r)$.

Proof: Fix the inputs of the two parties in an optimal protocol. Suppose that, in round i, the parties send messages of length a_i and b_i, respectively. By Corollary 3.1, the two parties can exchange these messages with a synchronized protocol using $\text{poly}(n)$ steps and $O(a_i/\log n + b_i/\log n)$ bits. The running time of this simulation is clearly polynomial in n, and the bit complexity is $\sum_{i=1}^{r} O(a_i/\log n + b_i/\log n) \leq O(D_r(\Pi)/\log n + r)$. ■

[1]These input-sending protocols can be easily shown to be optimal with respect to the bits they send. In Appendix A, we discuss these protocols in further detail, showing (for example) that an n-bit string can be sent in n steps and less than $n/4$ bits.

We can also prove a general upper bound, independent of the number of rounds:

Theorem 3.3: For every problem Π, $PB(\Pi) \leq O(D(\Pi)/\log\log n + 1)$.

Proof: The idea is to reduce an optimal protocol to another classical protocol that takes only $O(D(\Pi)/\log\log n)$ rounds, where $O(\log n)$ bits are exchanged in each round. Theorem 3.2 is then applied.

To get the round-efficient protocol, the parties divide the communication history of a protocol for Π into $D(\Pi)/(\log\log n)$ rounds of $\log\log n$ bits each. Each round starts at some node v of the protocol tree, and the goal is to find (with $O(1)$ synchronized bits) the unique node that will be reached by the parties after $\log\log n$ bits of normal communication. There are $\log n$ possible nodes of the tree that could be reached from v. Assuming communication alternates between the players, each player (say, Alice and Bob) can independently compute the subset of $\sqrt{\log n}$ nodes S_{Alice} and S_{Bob} that are consistent with their own inputs, i.e., reachable via paths where their own bits sent are determined by the protocol on their input. The actual node of the protocol tree reached at the end of the round will be the only node in $S_{\text{Alice}} \cap S_{\text{Bob}}$. Each player can describe their set for the current round as a string of length $\log n$, specifying which nodes are consistent. They repeat this process until the simulated protocol halts.

The above is a protocol for Π with round complexity $r = D(\Pi)/\log\log n$, and communication complexity $D(\Pi)\log n/\log\log n$. Applying Theorem 3.2, we obtain a polytime synchronized protocol with $O(D(\Pi)/\log\log n{+}1)$ bit complexity. ■

Complementing the above results, a synchronized bit protocol can also be simulated with a deterministic communication protocol with some overhead.

Theorem 3.4: Suppose $\Pi \in TB[t, b]$. Then $D(\Pi) \leq O(b(1 + \log \frac{t}{b}))$. Furthermore the protocol achieving this bound takes $O(b)$ rounds of messages with $O(\log \frac{t}{b})$ length.

Proof: Consider a synchronized protocol for Π satisfying the hypothesis. We partition the running time of the protocol into b blocks of t/b steps each.

Suppose N is the number of bits sent during an execution of the synchronized protocol on some block (and some fixed inputs). We shall describe how to simulate this protocol classically, with only $N(2\log(t/b)+2)+2$ communication.

First, each party sends a 1 iff they will send a bit during the current block, assuming that the other party sends only $\star$ (i.e., the other party never sends a bit). Note that at least one party sends a 1, if at least one party sends a bit during the block. If neither send a 1, the parties move to simulate the next block. This takes two bits of communication.

Otherwise, the parties exchange up to $2\log(t/b) + 2$ bits for each bit sent during the block, in the following way. They maintain a $(\log t/b)$-bit counter encoding the current timestep in the block (which is initially all-zeros). Assuming the opposite party does not send a bit during the block (sends only $\star$), a party sends either $\log(t/b) + 1$ bits, encoding the first bit they send in the block and the timestep in which they send it, or nothing at all if they do not send a bit under the assumption. (Note they have already indicated whether or not they would send a bit under this assumption.) The two parties then update their counter to the smaller of the two timesteps (if only one party sent a timestep, the counter is updated to that), and both assume that the party that sent the smaller timestep sent its bit. The simulation continues from this point in the protocol: the two parties exchange two bits to see if either will speak further from that point in the block, and then exchange timesteps of when they will speak. This process continues until both parties state that they will not speak further during the block.

The total communication is at most $N(2\log(t/b)+2)+2$ to simulate a block of a synchronized bit protocol in which x bits are exchanged. Since there are b blocks, it follows that $D(\Pi) \leq O(b(1 + \log t/b))$. ■

Theorem 3.4 immediately implies the following relationship between classical communication complexity and polytime bit complexity:

Corollary 3.3: $D(\Pi) \leq O(PB(\Pi)\log n)$. That is, if $b_k(n)$ is the synchronized bit complexity of Π over protocols running in $n^k + k$ time, then $D(\Pi) \leq O(b_k(n)\log n)$ for all constants k.

Summarizing the above results, we have the bounds on polytime bit complexity:

Theorem 3.5: For all Π,

$$\Omega\left(\frac{D(\Pi)}{\log n}\right) \leq PB(\Pi) \leq O\left(\frac{D(\Pi)}{\log\log n}\right).$$

It would be interesting to determine if the upper bound on $PB(\Pi)$ can be improved to $O(D(\Pi)/\log n)$ in general. Note that for problems with maximum communication complexity, we can asymptotically determine the polytime bit complexity.

Corollary 3.4: If $D(\Pi) = \Theta(n)$, then $PB(\Pi) = \Theta(n/\log n)$.

Proof: The upper bound is Corollary 3.2. For the lower bound, note that $D(\Pi) \geq \Omega(n)$ implies that $PB(\Pi) \geq \Omega(n/\log n)$, by Corollary 3.3. ■

These relationships also help us easily establish *time hierarchies* for bit complexity. For example:

Theorem 3.6: Let $f(n) \leq n^{1+o(1)}$. Then for all $\varepsilon > 0$, $TB[f(n), n/\log n] \subsetneq TB[n^{1+\varepsilon}, n/\log n]$.

Proof: By Corollary 3.1, any problem can be solved in $n^{1+\varepsilon}$ time and $O(n/(\varepsilon \log n))$ bits of communication. On the other hand, Theorem 3.4 says that every $\Pi \in TB[f(n), n/\log n]$ satisfies $D(\Pi) \leq O(f(n)\log((f(n)\log n)/n)) \leq O(n/\log n \cdot \log(n^{o(1)})) \leq o(n)$. Hence any problem with $D(\Pi) = \Theta(n)$ is not in $TB[f(n), n/\log n]$. ∎

It would be interesting to investigate further which conditions are sufficient for such hierarchies.

IV. Connection Complexity

We now introduce the *synchronized connection model* for communication. The distinguishing feature of this model is that the parties have the ability to choose *whether or not to participate in communication* in each step. In particular, in every step the two parties Alice and Bob choose whether to open their end of the communication channel, then try to send a bit. An exchange of bits takes place if and only if Alice and Bob both open their ends, and they are only charged in those steps where communication takes place. That is, the parties are only charged for the number of successful *connections* made. This model is natural in situations where parties do not have a means to alert the other to open up their channel. Unlike the synchronized bit model, this model does not easily relate to the usual communication model. This is because messages from Alice to Bob depend not only on Alice, but also whether Bob chooses to communicate.

Here we model synchronized connections with communication over the alphabet $\{0, 1, \star\}$, where $\star$ corresponds to *don't connect*.[2] The definition of a *connection protocol tree on n-bits* changes slightly from before. Every inner node v of the tree is now labeled with *two* functions, one function $f_v : \{0,1\}^n \rightarrow \{0, 1, \star\}$ for Alice and one function $g_v : \{0,1\}^n \rightarrow \{0, 1, \star\}$ for Bob. For every inner node, there are five child edges, with labels $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$, and $\star$. Evaluation of a node v has two rules: if $f_v(H_A)$ or $g_v(H_B)$ evaluates to $\star$, the $\star$-edge is traversed; otherwise, the edge labeled $(f_v(H_A), g_v(H_B))$ is traversed. The latter type of edge is called a *connection*. Intuitively, if either Alice or Bob choose not to connect, then no communication takes place, otherwise their messages are exchanged.

The *running time of a connection protocol* in this model is just the depth of the protocol tree. The *synchronized connection complexity* of a protocol is defined to be the maximum number of non-$\star$ edges on any path from root to leaf. (That is, we only count successful connections between Alice and Bob.) We can define complexity classes analogously to the synchronized bit complexity model:

[2]The complexity of protocols will only change by a constant factor when defined over larger alphabets.

Definition 4.1: $TC[t(n), c(n)]$ is the class of communication problems Π with the property that there is a constant $k \geq 1$ such that for all n, there is a connection protocol tree for Π_n with running time at most $k \cdot t(n) + k$, and synchronized connection complexity at most $k \cdot c(n) + k$.

Definition 4.2: $PC(\Pi)$, the *polytime connection complexity* of Π, is the set of functions $\{c_k(n)\}$, where $c_k(n)$ is the minimum synchronized connection complexity of any protocol for Π_n with running time at most $n^k + k$. For a function $f : \mathbb{N} \rightarrow \mathbb{N}$, we say $PC(\Pi) \leq O(f(n))$ if there is a k such that $c_k(n) \leq O(f(n))$. We say that $PC(\Pi) \geq \Omega(f(n))$ if for all k, $c_k(n) \geq \Omega(f(n))$.

We remark that synchronized connection complexity at least as powerful as bit complexity.

Proposition 1: $TB[t(n), b(n)] \subseteq TC[t(n), b(n)]$.

Proof: Consider a protocol with bit complexity $b(n)$ and running time $t(n)$. Without loss of generality, we can allocate even-numbered timesteps for Alice to (possibly) send a bit, and odd-numbered timesteps for Bob, in the synchronized bit protocol. To get an equivalent connection protocol, have Bob always send 1 during even-numbered timesteps, and Alice always send 1 in odd numbered timesteps. That is, when one party sends a bit during its allotted timestep, the other party always has its channel open, so the communication of that bit always takes place. ∎

Corollary 4.1: For all Π, $PC(\Pi) \leq PB(\Pi)$.

A. Relationship with Classical Communication

First we note that connections can be exchanged for more time, in any communication problem. We start with a useful definition for (normal) communication protocols. We may assume (with only a constant factor loss) that Alice speaks only at odd numbered time steps and Bob speaks at even numbered ones.

Definition 4.3: Let $y \in \{0, 1\}^t$ and let P be a protocol. We say that y is a *plausible communication history* for Alice (Bob) on P when the bits in odd positions of y (even positions of y) are precisely those that Alice (Bob) would respond on their input to the protocol P, assuming the opposite party responded with the bits in even positions of y (odd positions of y, respectively).

Theorem 4.1: Let Π a communication problem. Then for every $k \geq 1$, $\Pi \in TC[\frac{D(\Pi)2^k}{k}, \frac{D(\Pi)}{k}]$.

Proof: Start with a normal deterministic communication protocol P. To get a synchronized protocol with lower connection complexity, Alice and Bob split their communication into blocks of k bits, and treat a k-bit counter as a possible communication history for the next k bits exchanged in P. More precisely, the ith bit of the counter is treated as the ith bit sent in a communication history.

Alice sends a 1 when the current counter value corresponds to a plausible communication history for her (otherwise she sends $\star$), and Bob does similarly. Observe that they both send a 1 in a time step if and only if the counter value is the actual communication history for the next k steps. When they receive 1, they proceed to the next block. The simulation takes $O(D(\Pi)/k)$ rounds, using 2^k steps and only one connection in each round. ■

Therefore, we can always save a $\log n$ factor in the polytime *connection* complexity of a problem. (Recall in the case of polytime bit complexity, it is an open question if this is true.) Letting $k = \log n$ in Theorem 4.1, we have

Corollary 4.2: For all Π, $PC(\Pi) \leq O(D(\Pi)/\log n)$.

This already suggests that the connection model is stronger than the synchronized bit model. However, we can say much more. Define the communication problems $EQ = \{(x,x) | x \in \{0,1\}^n\}$ and $GT = \{(x,y) \mid x \leq y, x, y \in \{0,1\}^n\}$, where $\leq$ is the lexicographical order on strings. It is well known that EQ and GT both have $\Theta(n)$ deterministic communication complexity, and hence they have $\Omega(n/\log n)$ polytime bit complexity. In contrast, both problems have very low polytime connection complexity.

Theorem 4.2: EQ and GT can be solved in $2n$ steps and at most one connection.

Proof: We start with a protocol for EQ. Let A and B be the n-bit strings of Alice and Bob. In even-numbered steps of the form $2i$, Alice sends 1 if $A[i] = 0$, and does not speak otherwise (sends $\star$); Bob sends 1 if $B[i] = 1$ (and $\star$ otherwise). In steps of the form $2i+1$, Alice sends 1 if $A[i] = 1$ (otherwise $\star$) and Bob sends 1 if $B[i] = 0$ (otherwise $\star$). If a connection ever takes place, the two parties can immediately conclude that their strings are unequal. If no connection has occurred after $2n$ steps, the two parties conclude that their strings are equal.

Note this protocol can also be used to determine if the first player's string is less than the second player's, when the two strings are treated as integers. ■

Let $DISJ_n = \{(x,y) \mid \vee_{i=1}^{n} (x_i \wedge y_i) = 0\}$. $DISJ_n$ is well known to be $coNP^{cc}$-complete [BFS86] under *rectangular reductions* (the definition of which will not be necessary for this paper), and $DISJ_n$ requires $\Omega(n)$ communication, even in the randomized public-coin model [KN97]. On the other hand, it is easy to see that $DISJ_n$ has low polytime connection complexity.

Proposition 2: $DISJ_n$ can be solved in n steps and at most one connection.

Proof: A party sends 1 when the counter i is such that the ith bit of their input is 1, and does not speak otherwise (sends $\star$). If both parties send 1 at the same time, then the strings are not disjoint. Otherwise after n steps the players conclude that the strings are disjoint. ■

By extending this simple observation, we can show a tight correspondence between the synchronized connection model and the communication analogue of P^{NP}. Namely, problems solvable efficiently with a few connections are essentially those that can be solved with a few queries to an NP^{cc}-complete set (e.g., the complement of $DISJ_n$).

Let us first set up the query model for communication. Define a Ψ*-oracle protocol tree* as a protocol tree with an additional type of *query node*. At a query node, the outputs of arbitrary functions on their respective inputs by Alice and Bob are posed as queries to an oracle for Ψ, and the binary response from the oracle dictates whether the 0-edge or 1-edge is followed in the tree. In other words, a query q consists of a pair q_A, q_B, where q_A can depend arbitrarily on Alice's input and the previous communication and query answers, and likewise q_B can be an arbitrary function of Bob's input, the previous communication and previous query answers. The two parties then both learn in one step whether $(q_A, q_B) \in \Psi$.

Let Π be a communication problem. Recall $N(\Pi)$ is the nondeterministic communication complexity of Π. Since the proof of Theorem 4.1 performs an exhaustive search on plausible communication histories, it also works for nondeterministic protocols, in the following sense:

Corollary 4.3: For all Π, $\Pi \in TC[2^{N(\Pi)}, 1]$.

By a straightforward reduction, we can generalize this corollary to protocols that make multiple queries to a problem with low nondeterministic complexity.

Theorem 4.3: If Π has a Ψ-oracle protocol with communication complexity c making at most q queries, then $\Pi \in TC[c + q \cdot 2^{N(\Psi)}, q + c]$.

Proof: Let $t = N(\Psi)$. Consider a Ψ-oracle protocol tree for Π where each path from root to leaf has at most q query nodes. All non-query nodes can be directly simulated. We replace each query with a protocol described as follows.

Note the nondeterministic protocol tree P for Ψ on query v has at most 2^t leaves. Both parties can compute their part of the query, and thus determine the subset of such leaves consistent with their part of the input to Ψ and corresponding to accepting runs of the protocol. The goal is then to determine whether these sets intersect, since such an intersecting leaf would be an accepting run of the nondeterministic protocol on the pair of inputs (and vice versa).

To do this, they simulate the above set disjointness protocol, with linear time and $O(1)$ connections. Since the sets are from a universe of size 2^t, this takes 2^t time and constant connections. This is done q times to simulate the entire protocol, so the new protocol has total time $O(c + q \cdot 2^t)$ and connection complexity $O(c + q)$. ■

Next we show a converse simulation: any problem solvable with q connections in T time can be solved with approximately q queries to the complement of set disjointness (or any other NP^{cc}-complete problem), where each query has non-deterministic communication complexity logarithmic in T (e.g., is a set disjointness problem of size about T).

Theorem 4.4: If $\Pi \in TC[T, q]$ then Π has a $DISJ_n$-oracle protocol that makes $O(q \log(T/q))$ queries of length $n = O(T/q)$ (and hence each query has non-deterministic complexity $t \leq \log(T/q) + O(1)$).

Proof: Let P be a synchronized protocol for Π. In our $DISJ_n$-oracle protocol, there are $O(q)$ phases. In each phase, Alice and Bob either simulate T/q steps of P, simulate P until the next time a connection occurs, or simulate one bit of communication in P. Note that each of these events (a connection occurring, or T/q steps passing) can happen at most q times, so the number of phases of the simulation is at most $3q$.

In each phase, the parties have already simulated P up to a certain time step i. If a connection is active at time i, the parties simulate one bit of communication and go to time $i + 1$, and the phase ends. (They can simulate one bit of communication with a trivial set intersection query on a set of size 1.)

Otherwise, the two parties individually simulate P up to time $i + T/q$, assuming that there are no connections made and so no messages received. The parties each individually compute the set of times j in this interval that they would attempt to start a connection. The next connection to occur, if there is one, is the first time that both attempt to connect, i.e., the least element in the intersection of these two sets (since, until a connection is made, the assumption that no messages are received is correct.) The parties then ask one set disjointness query about these two sets. If the answer is that the sets are disjoint , they both individually emulate the protocol up to time $i + T/q$. If not, they use binary search to find the first such time j that a connection is formed. This binary search involves making up to $\log(T/q)$ set disjointness queries of size at most T/q.

If they reach the last time step, they return the answer in the simulated protocol. The simulation involves $O(q)$ phases, and each phase involves at most $\log(T/q)+1$ set disjointness queries of size at most T/q, which is as claimed. ■

We can now show how polytime connection complexity relates to P^{NP} in classical communication. First we define the class of problems with low polytime connection complexity.

Definition 4.4:

$$P^{conn} := \bigcup_{k>0} TC[n^k, \log^k n].$$

That is, P^{conn} contains those problems that can be solved in poly(n) time and poly$(\log n)$ connections.

Define $P^{NP^{cc}}$ to be the class of problems solvable with poly$(\log n)$ communication complexity with poly$(\log n)$ queries to $DISJ$ instances of size at most $2^{\text{poly}(\log n)}$. ($DISJ$ could be equivalently replaced with any other problem complete for $coNP^{cc}$.) $P^{NP^{cc}}$ is exactly the problems solvable in quasi-polynomial time and poly logarithmic connections, and hence is slightly larger to but related to the class P^{conn}. The following characterization of $P^{NP^{cc}}$ is immediate from Theorems 4.3 and 4.4.

Corollary 4.4: $P^{NP^{cc}} = \bigcup_{c,d>0} TC[2^{\log^c n}, \log^d n]$.

Therefore, we can use the synchronized connection model to prove lower bounds on $P^{NP^{cc}}$. (In retrospect, it is also easy to rephrase the proofs directly in terms of queries to $DISJ$, but thinking in terms of connections helped us discover it.)

B. Lower Bounds for Connection Complexity

We now turn to proving lower bounds in the synchronized connection model. To start, we present a simplified form for protocols in the model, which facilitates analysis. Essentially, the connection model can be seen as a classical model where an intermediate channel between the parties performs an AND of the two bits being communicated, and only charges for communication when this AND is 1. A related notion is that of *energy complexity* [UDM06], [UT07] for circuits: the energy complexity is the maximum number of gates that output 1 over all inputs to the circuit.

Definition 4.5: An *AND-protocol* is a (synchronized) communication protocol that works as follows. In every step, two players each send a bit to a third party. The third party computes the AND of the bits, and sends the result to both players. The *AND-complexity* of a protocol is the maximum number of times the AND evaluates to 1, over all possible conversations in the protocol.

So in an AND-protocol, the parties are only charged for 1's output by the third party. We can think of protocol trees in this model in the following way: each inner node v is labeled with the AND of two functions $f_v(x)$ and $g_v(y)$. Each node has two children, a 1-child and 0-child. A protocol that has connection complexity k is such that every path from root to leaf has at most k 1-edges. A protocol taking time t has depth at most t. Note in a protocol with connection complexity k and time t, there are at most $\binom{t}{k}+1$ leaves. We say that a node *evaluates* to a 0-1 value on input (x, y) if $f_v(x) \wedge g_v(y)$ equals that value.

Theorem 4.5: For every protocol with connection complexity c and running time t, there is an AND-protocol computing the same function with AND-complexity c and running time at most $5t$.

Proof: Start with a protocol P with c connections, and t time. Our AND-protocol will simulate a single step of P

within at most five steps. In each phase of the AND-protocol, Alice and Bob increment a counter ranging from $1, \ldots, 4$, and interpret the counter value as a pair $(\sigma_A, \sigma_B) \in \{0,1\}^2$.

Suppose we are at node v in P at the start of a phase. Alice (resp. Bob) sends 1 if the party would send σ_A (resp. σ_B) when in node v of P, otherwise the party sends 0. If both send a 1 simultaneously, the simulation of P continues along the (σ_A, σ_B)-edge out of v, the phase ends, and the counter resets. If the counter reaches 5, the simulation continues along the $\star$-edge out of v. It is clear that this AND-protocol simulates P faithfully and increases the running time by a factor of 5. ■

The upshot of Theorem 4.5 is that we may assume without loss of generality that Alice and Bob only send *bits* (and never $\star$). They only see the AND of what both parties sent in a timestep, and they are only charged for communication when both parties send 1 at the same time.

A complication in analyzing AND protocols is that the set $\{(x,y)\}$ corresponding to a leaf is no longer a combinatorial rectangle (as is the case with typical communication complexity). For example, consider a protocol tree with one inner node labeled with functions f and g such that $f(x_1) = 0$, $f(x_2) = 1$, $g(y_1) = 1$, $g(y_2) = 0$. Then (x_1, y_1) and (x_2, y_2) both lead to the 0-leaf, but (x_2, y_1) leads to the 1-leaf. Nevertheless, we can still prove lower bounds by arguing that protocols with low time and connection complexity only work for problems with large monochromatic rectangles. This is interesting, since it is clear that the connection model *can* efficiently solve some problems having *many* monochromatic rectangles, such as EQ. (However, observe that EQ does have a very large monochromatic rectangle.)

We say that the *area* of a combinatorial rectangle is the product of its two dimensions.

Theorem 4.6: For any problem Π with connection complexity c in time t, the matrix for Π_n contains a monochromatic combinatorial rectangle with area at least $\left(\frac{4c^2}{(c+t)^2}\right)^c \left(1 - \frac{2c}{t+c}\right)^{t-c} \cdot 2^{2n} \geq \left(\frac{c}{2e^{1/2}t}\right)^{2c} \cdot 2^{2n}$.

Proof: Consider an AND-protocol for Π. At each time step, we will maintain a large combinatorial rectangle of inputs, all of which reach the current node we are visiting in the protocol tree. When we reach a leaf, this rectangle must be monochromatic. Initially, we begin at the root and the current rectangle is the complete $2^n \times 2^n$ matrix for Π.

Let K be a parameter. Suppose we have reached a node v in the tree and the current rectangle is R. Let p (respectively, q) be the fraction of x-inputs (respectively, y-inputs) in R where a 1 is sent in node v. If $pq \geq 1/K^2$, then set

$$R \leftarrow \{(x,y) \in R \mid v \text{ evaluates to 1 on } (x,y)\},$$

and set the current node to be the 1-child of v. Note R is still a combinatorial rectangle, and R is at least a $1/K^2$ fraction of its original area. If $pq < 1/K^2$, let $r = \max\{1-p, 1-q\} \geq 1 - 1/K$. Without loss of generality, suppose $p \geq q$. Then set

$$R \leftarrow \{(x,y) \in R \mid v \text{ evaluates to 0 on } x\}$$

and set the current node to be the 0-child of v. (If $q < p$, the same operation is done with y in place of x.) Now the new rectangle R is at least a $1 - 1/K$ fraction in area. Follow a path down the tree in this manner, until a leaf is reached. Observe that for a path where c connections were made and t time was taken, the size of R at the leaf is at least

$$\left(\frac{1}{K^2}\right)^c \left(1 - \frac{1}{K}\right)^{t-c} \cdot 2^{2n},$$

and R is monochromatic. To determine K to maximize this value, we differentiate $f(K) = \left(\frac{1}{K^2}\right)^c \left(1 - \frac{1}{K}\right)^{t-c}$ and find that $f(K)$ achieves its maximum when $K = \frac{c+t}{2c}$.

For our applications, it will suffice (and be very convenient) to give a simplified form. Recall that when $x \leq 1/2$, we have $1 - x \geq e^{-2x}$. Let $K = \frac{2t}{c}$. Since $t \geq c$, we have $\frac{1}{K} \leq \frac{1}{2}$ and therefore $(1 - 1/K) \geq e^{-2/K}$, so the lower bound on area simplifies to $\left(\frac{c^2}{4t^2}\right)^c e^{-c/t} \cdot 2^{2n} \geq \left(\frac{c}{2e^{1/2}t}\right)^{2c} \cdot 2^{2n}$. ■

Corollary 4.5: Any communication problem in $P^{NP^{cc}}$ must have a monochromatic rectangle with area at least $\frac{2^{2n}}{2^{(\log n)^c}}$ for some constant $c \geq 1$.

The following is immediate from the proof of Theorem 4.6 and the reduction from $P^{NP^{cc}}$ protocols to synchronized connection protocols (Theorem 4.3).

Theorem 4.7: Any function computable with q queries to a language with non-deterministic communication complexity m has a monochromatic combinatorial rectangle $S_1 \times S_2$ where $|S_i| \geq 2^{n - O(qm)}$ for $i = 1, 2$.

The Inner Product function has been studied thoroughly in Communication Complexity. Define the problem $IP = \{(x,y) \mid \sum_{i=1}^n x_i y_i = 0 \bmod 2\}$.

Corollary 4.6: IP requires $\Omega(c \cdot 2^{n/(2c)})$ time in any protocol with at most c connections.

Proof: The problem does not have a monochromatic rectangle of area greater than 2^n ([KN97],p.12). By Theorem 4.6, we have $2^n > \left(\frac{c}{2e^{1/2}t}\right)^{2c} \cdot 2^{2n}$, or

$$\left(\frac{2e^{1/2}t}{c}\right)^{2c} > 2^n.$$

Taking the $2c$-th root and multiplying by c, the condition becomes $t \geq \Omega(c \cdot 2^{n/(2c)})$. ■

Corollary 4.7: $PC(IP) \geq \Omega(n/\log n)$ and $IP \notin P^{NP^{cc}}$.

The above lower bound is nearly tight: we can send an entire n-bit string in $O(c2^{n/c})$ time with only c connections, hence IP requires $\Theta(n/\log n)$ connections in polytime. It is interesting that in the classical deterministic and randomized models, there is little difference in the complexities of Inner Product and Set Disjointness, as both require $\Omega(n)$ bits, while in our (deterministic) model there is a major difference. This reflects the fact that Inner Product is $\oplus P^{cc}$-complete while Set Disjointness is only $coNP^{cc}$-complete.

Finally, we give a lower bound for a communication problem known to be in $\Sigma_2^{cc} \cap \Pi_2^{cc}$. Define Block-Equality to be the set

$$\{(x,y) \in \{0,1\}^{\ell^2} \times \{0,1\}^{\ell^2} \mid \exists\, i \in [\ell], \forall\, j \in [\ell], x_{ij} = y_{ij}\}$$

where z_{ij} is the $(i \cdot \ell + j)$th bit of z. Block-Equality is obviously in Σ_2^{cc}; Lam and Ruzzo [LR89] proved that Block-Equality is also in Π_2^{cc}. In fact, their protocol uses random hash functions in the universal quantifier, and so the containment can be improved to AM^{cc}.

Theorem 4.8: The Block-Equality problem requires $\Omega\left(\frac{c}{n^{1/(4c)}} \cdot 2^{\sqrt{n}/(2c)}\right)$ time for any protocol with at most c connection complexity.

Proof: We argue that every 1-rectangle and 0-rectangle in the matrix for Block-Equality is small. Suppose we pick a uniform random pair of strings from $\{0,1\}^{\ell^2}$. The probability that two ℓ-bit strings are equal is $1/2^\ell$, and so the probability that one of the ℓ blocks has two equal strings is at most $\ell/2^\ell$. So there are at most $\frac{\ell 2^{\ell^2}}{2^\ell}$ ones in the entire matrix, and hence any 1-rectangle has area at most this.

Now consider any 0-rectangle, $A \times B$. For $i = 1, \ldots, \ell$, let A_i be the set of all ℓ-bit strings appearing in the ith block of the ℓ^2-bit strings of A. Let B_i be defined similarly. Note $A_i \cap B_i = \varnothing$, else there is a 1-entry in $A \times B$. Therefore $|A_i| + |B_i| \leq 2^\ell$, hence $|A_i||B_i| \leq \frac{1}{4} \cdot 2^{2\ell}$. The measure of $A \times B$ can be bounded by

$$\mu(A \times B) \leq \prod_{i=1}^{\ell} \mu(A_i) \cdot \mu(B_i) \leq \left(\frac{1}{4}\right)^{\ell},$$

so the largest 0-rectangle has area at most $\frac{2^{\ell^2}}{4^\ell}$. Therefore by Theorem 4.6, any t time and c connection protocol satisfies

$$\frac{\ell 2^{2\ell^2}}{2^\ell} \geq \left(\frac{c}{2e^{1/2}t}\right)^{2c} \cdot 2^{2\ell^2}.$$

That is, $\left(\frac{2e^{1/2}t}{c}\right)^{2c} \geq \frac{2^\ell}{\ell}$. By an identical argument as Corollary 4.6, we have $t \geq \Omega(c \cdot 2^{\ell/(2c)}/\ell^{1/(2c)})$, and Block-Equality requires $\Omega(c \cdot 2^{\sqrt{n}/(2c)}/n^{1/(4c)})$ time with c connections. In the polynomial time case, $\Omega(\sqrt{n}/\log n)$ connections are required. ■

The below theorem summarizes the known and new relationships between $P^{NP^{cc}}$ and other communication classes.

Theorem 4.9: (i) $P^{NP^{cc}} \subsetneq \Sigma_2^{cc} \cap \Pi_2^{cc}$
(ii) $P^{NP^{cc}} \nsubseteq MA^{cc}$ and $AM^{cc} \nsubseteq P^{NP^{cc}}$
(iii) $\oplus P^{cc} \nsubseteq P^{NP^{cc}}$ and $P^{NP^{cc}} \nsubseteq \oplus P^{cc}$

Proof: Theorem 4.8 implies (i). The first part of (ii) follows from an $\Omega(\sqrt{n})$ lower bound in MA^{cc} for $DISJ_n$ [Kla03], and the second part follows from Theorem 4.8 and the fact that Block Equality is in AM^{cc}. Finally, (iii) follows from Theorem 4.6 and an $\Omega(n)$ lower bound for $DISJ_n$ in $\oplus P^{cc}$ [DKMW04]. ■

V. Discussion

To our knowledge, all extensions to the original model of communication complexity have arisen from granting various extended modes of computation from complexity theory to the parties in a communication protocol. This has led to a very productive and quite comprehensive view of how these different modes of computation affect what can be computed with efficient communication. In this paper, we took a different approach towards extensions of communication complexity. We added new types of *actions* to the parties in a communication protocol: the ability to be silent, and the ability to opt out of communication, for some duration of time. Since these abilities are often available in practice, it is of great interest to measure and understand their power. Our approach can be recast in the traditional language of this subject, and in fact it tells us something new about the communication version of P^{NP}.

There is much opportunity for future work in both synchronized bit complexity and synchronized connection complexity. We do not know if the upper bound on $PB(\Pi)$ can be improved to $O(D(\Pi)/\log n + 1)$ in general. Also, it would be interesting to determine the polytime bit complexity of relations. For instance, if Alice and Bob each have a subset of $\{1, \ldots, n\}$, can the median of all their numbers be computed in polytime and $O(1)$ bits? (Note this can be solved with $O(\log n)$ deterministic communication, cf. [KN97].)

As another potential direction, we believe that by supplementing the traditional communication model with other intuitive axioms, it should be possible to recover other communication classes and in turn, understand them better. For example, it is still an open problem if $\Sigma_2^{cc} = \Pi_2^{cc}$. Our paper suggests an approach to this problem: define natural communication models that capture the two classes in some way, and use the intuition for that model to attack the problem.

Finally, it would be also interesting to develop a theory of randomized synchronous communication. For example, do problems with low randomized connection complexity (and time efficiency) correspond to BPP^{NP}? Preliminary results suggest that this is the case. Does randomness help significantly in the polytime bit complexity of problems?

Acknowledgements: We thank the program committee, Brendan Juba, and Oded Goldreich for helpful comments.

References

[AW08] S. Aaronson and A. Wigderson. Algebrization: A New Barrier in Complexity Theory. In *Proc. STOC*, 731–740, 2008.

[AB09] S. Arora and B. Barak. Communication complexity: a modern approach. Cambridge, 2009.

[Awe85] B. Awerbuch. Complexity of network synchronization. *JACM* 32(4):804–823, 1985.

[BFS86] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). *Proc. FOCS*, 337–347, 1986.

[DKMW04] C. Damm, M. Krause, C. Meinel, and S. Waack. On relations between counting communication complexity classes. *JCSS* 69:259–280, 2004.

[DFO06] A.K. Dhulipala, C. Fragouli, and A. Orlitsky. Silence Based Communication for Sensor Networks. *Proc. IEEE International Symposium on Information Theory (ISIT)*, 212–216, 2006.

[FO05] C. Fragouli and A. Orlitsky. Silence is golden and time is money. *Allerton Annual Conference on Communications, Control, and Computing*, 2005.

[FL87] G. N. Frederickson and N. A. Lynch. Electing a Leader in a Synchronous Ring. *JACM* 34(1):98–115, 1987.

[GHS83] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM TOPLAS* 5(1), 66–77, 1983.

[HIKNV04] P. Harsha, Y. Ishai, J. Kilian, K. Nissim, and S. Venkatesh. Communication Versus Computation. *Proceedings of ICALP*, 745–756, 2004.

[IKK09] R. Impagliazzo, V. Kabanets, and A. Kolokolova. An Axiomatic Approach to Algebrization. To appear in *STOC*, 2009.

[Kla03] H. Klauck. Rectangle Size Bounds and Threshold Covers in Communication Complexity. *Prc. IEEE Conf. on Computational Complexity*, 118–134, 2003.

[KN97] E. Kushilevitz and N. Nisan. Communication Complexity. Cambridge University Press, 1997.

[LR89] T. W. Lam and W. L. Ruzzo. Results on Communication Complexity Classes. *Proc. Structure in Complexity Theory Conference*, 1989.

[Lam84] L. Lamport, Using time instead of timeout for fault-tolerant distributed systems. *ACM Trans. Prog. Lang. and Syst.* 6(2):254–280, 1984.

[Lis93] B. Liskov. Practical uses of synchronized clocks in distributed systems. *Distributed Computing* 6(4):211–219, 1993.

[Lyn96] N. Lynch. Distributed Algorithms. Morgan Kauffman, 1996.

[PS84] C. H. Papadimitriou and M. Sipser. Communication Complexity. *J. Comput. Syst. Sci.* 28(2):260–269, 1984.

[UDM06] K. Uchizawa, R. J. Douglas, and Wolfgang Maass. Energy Complexity and Entropy of Threshold Circuits. In *Proc. ICALP Vol.1*, 631–642, 2006.

[UT07] K. Uchizawa and E. Takimoto. An Exponential Lower Bound on the Size of Constant-Depth Threshold Circuits with Small Energy Complexity. In *Proc. IEEE Conf. on Computational Complexity*, 169–178, 2007.

Appendix

A. Bit-Optimal Methods for Sending Strings

To give more intuition for the synchronized bit model, we further study the problem of sending one's entire input in this appendix. In the usual communication setting, this problem has no nontrivial solution.

Proposition 3: One party can send its n-bit string to the other in $2 + \log_3(2)n \approx 0.631n$ time and at most $2 + \frac{2}{3} \cdot \log_3(2)n \approx 0.421n$ bits.

Proof: One party encodes their n-bit string in ternary (using $\log_3(2)n$ symbols), and notes the ternary symbol σ that occurs most often in the encoding. The party sends two bits indicating σ, then sends the string over the alphabet $\{0, 1, \star\}$, letting $\star$ denote σ. The number of bits sent is at most $2/3 \cdot \log_3(2)n$. ■

Although it is very simple, the above proposition is essentially optimal with respect to time. There are $E(\delta, t) = \binom{t}{\delta t} 2^{(1-\delta)t}$ ways to send $(1-\delta)t$ bits in t time. Minimizing t subject to $E(\delta, t) \geq 2^n$ can be done by maximizing the exponent of $E(\cdot, t)$ with respect to δ. By calculus we find that $\delta^* = 1/3$ maximizes the exponent and $E(1/3, t) = 2^{t/\log_3(2)}$. Hence we can choose $t = \log_3(2)n \approx 0.631n$ and send $(1-\delta^*) \cdot \log_3(2)n \approx 0.421n$ bits, which are precisely the bounds of part (ii) up to 2 bits.

It is natural to ask how many bits suffice to send an arbitrary string in n steps, since this is the number of steps required in the original communication model. In this case we can save many more bits of communication. Let H be the binary entropy function, i.e., $H(p) = p \log_2(1/p) + (1 - p)\log_2(1/(1-p))$.

Theorem A.1: Let $\delta > 0$ satisfy $\delta + H(\delta) \geq 1$. An n-bit string can be transmitted in n steps and at most δn bits.

Note we can choose $\delta \approx 0.228$, so it suffices to send less than $n/4$ bits.

Proof: Choose $\delta + H(\delta) = 1$. Divide a $(\delta + H(\delta))n$ bit string z in $z = xy$, where $|x| = \delta n$ and $|y| = H(\delta)n$, respectively. A standard combinatorial fact is that $2^{H(\delta)n} \geq \binom{n}{\delta n}$. Hence there is a bijection f from strings y of length $H(\delta)n$ to sets $S \subseteq \{1, \ldots, n\}$ with $|S| = \delta n$. Now run the protocol:

```
Let $j = 1$, $S_y = f(y)$.
For time steps $i = 1, ..., n$,
    If $i \in S_y$, then send the $j$th bit of $x$ and increment $j$.
                else send nothing.
```

Since $|S_y| = \delta n$ the protocol clearly sends all the bits of x. The set S_y can be determined by recording those steps in which a bit was sent. Therefore y can be determined by computing $f^{-1}(S_y)$. ■

The above is the best possible for sending an n-bit string in n time steps. Intuitively, if no more than b bits are sent and t time is taken, then there are at most $K = \sum_{i=1}^{b} \binom{t}{i} 2^i$ possible distinct communications from one party to the other. To send every n-bit string successfully, we must have $K \geq 2^n$. But if $t = n$ then we must have $b \geq 0.227n$ in order for $K \geq 2^n$.

Proposition 4: An n-bit string can be sent in polynomial time with only $O(n/\log n)$ bits of communication.

Proposition 4 is optimal for similar reasons. Consider a one-way communication that takes n^k time and at most L bits are passed from one party to the other. There are $K = \sum_{i=0}^{L} \binom{n^k}{i} 2^L$ distinct communications, and we need $K \geq 2^n$ so that each n-bit string to correspond to a different communication. By standard estimates,

$$2^n \leq K \leq (en^k/L)^L 2^L = (2e)^L 2^{L \log(n^k/L)}.$$

We need $L \geq \Omega(n/\log n)$.

B. Applications to Algebrization

Here we give a summary of implications of our communication lower bounds to the impossibility of collapsing the second level of the polynomial time hierarchy by "algebrizing means".

There are two ways to model what it means to say that a complexity collapse algebrizes. In [AW08], the authors say that $C_1 \subseteq C_2$ algebrizes if for any oracle A and any low-degree extension $\tilde{A}$, $C_1^A \subseteq C_2^{\tilde{A}}$. In [IKK09], the authors give an axiom called ACT (algebraic checkability theorem) and define a complexity statement as algebrizing if any oracle that satisfies ACT also satisfies the statement. Both approaches prove non-algebrization of statements using communication complexity.

For oracles A_0 and A_1, let $A_0 + A_1 = \{(b, x) \mid b \in \{0,1\}, x \in \{0,1\}^*, x \in A_b\}$ be the oracle where questions can be asked to either A_0 or A_1. It is shown in [AW08] that there is a construction of a low degree extension $\hat{A}$ of $A_0 + A_1$ for any A_0 and A_1 where, if one party in a communication protocol knows the subset of of A_0 restricted to strings of length up to n, and the other knows a similar subset of A_1, they can compute whether a string y of length n is in $\hat{A}$ with polynomial in n (which is polylogarithmic in their input sizes) communication. They use this to convert communication complexity separations into oracle constructions showing the containments do not algebrize.

Similary, [IKK09] show that there is a function $B(A)$ that converts an oracle into a stronger oracle so that, for every A_0 and A_1, $B(A_0) + B(A_1)$ satisfies ACT. Since $B(A_0)$ can be computed from just A_0 and $B(A_1)$ from just A_1, a query to $B(A_0) + B(A_1)$ can be simulated by a bit of communication between the two parties described above.

Let BE^{A_0,A_1} be the following language equivalent to the Block Equality problem for oracles A_0 and A_1:

$$\{1^n \mid \exists z \in \{0,1\}^n, \forall w \in \{0,1\}^n \; [(z,w) \in A_0 \text{ if and only if } (z,w) \in A_1]\}.$$

For any A_1, A_2, $BE^{A_1,A_2} \in \Sigma_2^{A_1+A_2}$, and so is also in $\Sigma_2^{\hat{A}}$ and $\Sigma_2^{B(A_1)+B(A_2)}$. Let $l = 2^n$.

Fix any oracle machine M^O running in polynomial time using NP^O queries, and let n be large enough that the running time of M on input 1^n, and the nondeterministic time taken by any query, are at most $l^{1/8} = 2^{n/8}$. If we cannot find $A_0, A_1 \subseteq \{0,1\}^{2n}$ so that $M^{\hat{A}}$ fails to compute whether $1^n \in BE^{A_0,A_1}$ we can use M as an efficient communication protocol to decide the Block-Equality problem on inputs of length $l^2 = 2^{2n}$ as follows. Every time M makes a query to $L \in NP^{\hat{A}}$, the two parties make the following query which can be expressed as a set intersection query: *is there a nondeterministic path of the machine for L and communication patterns for each query along that path to $\hat{A}$ so that the path accepts and is consistent with both A_0 and A_1?*

This contradicts the lower bound of Theorem 4.8. So we can diagonalize against all such machines, for larger input sizes. Thus, the statement $\Sigma_2 \subseteq P^{NP}$ does not algebrize in the [AW08] sense.

Similarly, we can find n and A_0 and A_1 so that the oracle machine $M^{B(A_0)+B(A_1)}$ fails on 1^n. By diagonalizing against all such machines with different values of n, we construct an oracle of the form $B(A_0) + B(A_1)$ so that $P^{NP^{B(A_0)+B(A_1)}} \subsetneq \Sigma_2^{B(A_0)+B(A_1)}$. Since $B(A_0) + B(A_1)$ satisfies ACT, this shows that the same containment does not algebrize in the [IKK09] sense.