Synchronous Computations, Basic techniques (Secs. 6.1-6.2)

T-79.4001 Seminar on Theoretical Computer Science

Aleksi Hänninen

21.03.2007
Outline

**Synchronous Systems**
- Definitions
- New Restrictions

**Synchronous Algorithms**
- Speed
- TwoBits
- The Cost of Synchronous Protocols

**Communicators, pipeline, and transformers**
- Two-Party Communication Problem
- Pipeline
- Asynchronous-to-Synchronous Transformation
Outline

**Synchronous Systems**
Definitions
New Restrictions

**Synchronous Algorithms**
Speed
TwoBits
The Cost of Synchronous Protocols

**Communicators, pipeline, and transformers**
Two-Party Communication Problem
Pipeline
Asynchronous-to-Synchronous Transformation
Notation

- $n$ is the number of nodes, $m$ is the number of edges
- Standard set of restrictions
  - $R = \{\text{Bidirectional Links, Connectivity, Total Reliability}\}$
- $M[P]$ is the number of messages needed in protocol $P$
- $T[P]$ is the time required in protocol $P$
- $B[P]$ is the number of bits needed in protocol $P$
- $\langle T, B \rangle$ is the total cost of protocol
Restrictions for fully synchronous system

6.1.1. Synchronized Clocks

► All entities have a clock, which are incremented by one unit $\delta$ simultaneously. Clocks are not necessarily at the same time.
► All messages are transmitted to their neighbors only at the strike of a clock tick
► At each clock tick, an entity will send at most one message to the same neighbor

6.1.2 Bounded Communication Delays

► There exists a known upper bound on the communication delays $\Delta$

System which satisfies these is a fully synchronous system. The messages are called packets, and have a bounded size $c$. 
Synchronous restriction: **Synch**

Every synchronous system with 6.1.1. and 6.1.2 can be “normalized” so that a communication delay is one new tick $\delta$ (assuming that entities know the time when to begin ticking).

6.1.3 Unitary Communication Delays

- In absence of failures, a transmitted message will arrive and be processed after at most one clock tick

6.1.1 + 6.1.3. = **Synch**

- Messages sent in time $t$ is received at time $t + 1$
- Simplifies situation
Outline

Synchronous Systems
  Definitions
  New Restrictions

Synchronous Algorithms
  Speed
  TwoBits
  The Cost of Synchronous Protocols

Communicators, pipeline, and transformers
  Two-Party Communication Problem
  Pipeline
  Asynchronous-to-Synchronous Transformation
Synchronous Minimum Finding

\[ \text{AsFar} + \text{delay in send} = \text{Speed} \]

- \( \text{IR} \cup \text{Synch} \cup \text{Ring} \cup \text{InputSize}(2^c) \)
- \( \text{AsFar} \): Optimal message complexity on average but worst-case is \( \mathcal{O}(n^2) \).
- Idea: Delay forwarding messages with large id to allow smaller id messages catch it up.
- Delay is \( f(i) \) which is monotonical
- Correctness: Basically because \( \text{AsFar} \) is (message with smallest id is newer trashed)
Complexity

If \( f(i) := 2^i \)
\[
M[\text{Speed}] = \sum_{j=1}^{n} \frac{n}{2^{j-1}} < 2n
\]
In the time when first has went through, second largest has went at most half the way, third largest \( n/4 \ldots \)
\[
M[\text{Speed}] = \mathcal{O}(n)!
\]

- Message complexity smaller than the \( \mathcal{O}(n \log(n)) \) lower bound of asynchronous rings
- However, \( T[\text{Speed}] = \mathcal{O}(n2^i) \)
- Exponential to the input values, worse than to size \( n \)
Situation: \( x \) wants to send bits \( \alpha \) to his neighbor \( y \) with only two bits.

\( \text{Int}(1\alpha) \) is some known bijection between bit strings and integers.

Silence is information also!
1. Entity
   1. Send “start counting”
   2. Wait $\text{Int}(1\alpha)$ ticks
   3. Send “stop counting”

2. Entity
   1. Upon receiving “start counting”, record current time $c_1$
   2. When “stop counting” is received, calculate the difference of current time $c_2$ and $c_1$ and use the fact that $c_2 - c_1 = \text{Int}(1\alpha)$, from which the $\alpha$ can be deduced.
The cost of a fully synchronous protocol is both time and transmissions.

Time can be saved by using more bits, and bits can be saved by using more time.

\[
\text{Cost}[\text{Protocol } P] = \langle B[P], T[P] \rangle
\]
\[
\text{Cost}[\text{Speed}(i)] = \langle O(n\log(i)), O(n2^i) \rangle
\]
\[
\text{Cost}[\text{TwoBits}(\alpha)] = \langle 2, O(2^{|\alpha|}) \rangle
\]
Outline

Synchronous Systems
  Definitions
  New Restrictions

Synchronous Algorithms
  Speed
  TwoBits
  The Cost of Synchronous Protocols

Communicators, pipeline, and transformers
  Two-Party Communication Problem
  Pipeline
  Asynchronous-to-Synchronous Transformation
In fully synchronous systems, also the absence of transmission can be used to convey information between the nodes.

There are many solutions to the Two-Party Communication Problem, called *communicators*.

Time and bit cost of sending information $I$ between neighbours using $C$ are denoted $Time(C, I)$ and $Bit(C, I)$, respectively.
TPC settings

- **Sender** will send $k$ packets which defines $k - 1$ quanta of time $q_1, \ldots, q_{k-1}$.
- The ordered sequence $\langle p_0 : q_1 : \cdots : q_{k-1} : p_{k-1} \rangle$ is called a communication sequence.
- A protocol *communicator* $C$ must specify an encoding function which encodes information to a communication sequence and a decoding function to decode it.
2-bit Communicator Protocol $C_2$

- $encode(i) = \langle b_0 : i : b_1 \rangle$
- $decode(b_0 : q_1 : b_1) = q_1$
- $Cost[C_2(i)] = \langle 2, i + 2 \rangle$
Hacking

- The values of bits are not used, so they can be changed. The protocol is *corruption tolerant*.
- We can also encode 2 bits to the values of $b_0$ and $b_1$. The new protocol is called $\mathcal{R}_2(i)$ and the time reduces to $2 + \frac{i}{4}$.
- From now on, we restrict ourselves to corruption tolerant TPC, and denote the communication sequence with only the quantas $q_i$ as a tuple $\langle q_1 : \cdots : q_k \rangle$. 
3-bit Communicators

- \( encode(i) = \langle \lfloor \sqrt{i} \rfloor : i - (\lfloor \sqrt{i} \rfloor)^2 \rangle \).
- \( decode(q_1 : q_2) = q_1^2 + q_2 \).
- \( Cost [C_2(i)] = \langle 3, 3 + \lfloor \sqrt{i} \rfloor \rangle \), sublinear!
(2^d + 1)-bit Communicators

- The idea can be quite easily extended using this idea with divide and conquer.
- A solution protocol using \( k = 2^d + 1 \) bits will use \( O\left(\frac{i^1}{k}\right) \) time and \( k \) bits.
Pipeline - a setting

- Consider now the case where the two communicators $x$ and $y$ are not neighbours.
- However, a chain $x_1, x_2, \ldots, x_p$, where $x_1 = x$ and $x_p = y$, is known to everybody.
- If the information $I$ is sent from $x_i$ to $x_{i+1}$ using always a Communicator $C$ between them, the time cost is $(p - 1) \text{Time}(C, I)$. 
Pipeline - a solution

- To reduce the amount of time, we can form a pipeline.
- Once a relayer $x_i$, $i \in \{2, \ldots, p - 1\}$, gets a message, it will forward it in the next tick.
- The time cost is reduced to $(p - 1) + \text{Time}(C, I)$
- The bit cost is $(p - 1)\text{Bit}(C, I)$ for both.
Computing in Pipeline

- Also computations can be performed in a pipeline.
- Consider a maximum finding in pipeline, using **TwoBits** as our communicator. All entities $x_i$ has an integer of information $I_i$.
- The sender $x_1$ will send first “start” message and after $I_1$ tick “stop” message.
- All $x_i$ will forward the “start” message without delay. When the “stop” message is received, it is forwarded unless the time between “start” and “stop” is less than $I_i$. In this case the “stop” is delayed accordingly.
Computing in Pipeline

- Number of bits is $(p - 1) \text{Bits}(C, I_{max})$, same as without pipeline.
- Time is reduced to $(p - 1) + \text{Time}(C, I_{max})$.
- Only restriction for $C$:
  “If $I > J$, then in $C$ the communication sequence for $I$ is lexicographically smaller than for $J$.” (from the book)
Asynchronous-to-Synchronous Transformation

- If $A$ is a known solution to some problem in asynchronous setting, it does not depend on the timing conditions.
- The maximum size $m(A)$ of the messages used by $A$ must be less than the packet size $c$, otherwise the message complexity is increased.
- $A$ can also be automatically converted to fit in synchronous system just by using a transformer.
Transformers

- **Transformer** $\tau [C]$ Given any asynchronous protocol $A$, replace the asynchronous transmission-reception of each message in $A$ by the communication, using $C$, of the information contained in that message.
Transformers

- $M(A)$ message complexity of $A$
- $m(A)$ size of the largest message
- $T_{causal}$ the length of the longest chain of communication, $\leq M(A)$

**Lemma 6.2.2 Transformation Lemma**

$Cost[\tau[C](A)] \leq \langle M(A)Packets(C, m(A)), T_{causal}(A)Time(C, 2^{m(A)}) \rangle$
An Example

- Election in a Synchronous Ring using Stages and a transformer → *SynchStages*.
- *SynchStages* uses $O(n \log n)$ bits and $O(in \log n)$ time.
- Better than the cost of time $O(n \log i)$ and packets $O(in2^i)$ of the Speed.
Outline

Synchronous Systems
  Definitions
  New Restrictions

Synchronous Algorithms
  Speed
  TwoBits
  The Cost of Synchronous Protocols

Communicators, pipeline, and transformers
  Two-Party Communication Problem
  Pipeline
  Asynchronous-to-Synchronous Transformation