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

T-79.4001 Seminar on Theoretical Computer Science

Aleksi Hänninen

21.03.2007

Outline

Outline

Notation

- n is the number of nodes, m is the number of edges
- Standard set of restrictions
  $\mathbf{R} = \{\textit{Bidirectional Links}, \textit{Connectivity}, \textit{Total Reliability}\}$
- $\mathbf{M}[P]$ is the number of messages needed in protocol P
- $\mathbf{T}[P]$ is the time required in protocol P
- $\mathbf{B}[P]$ is the number of bits needed in protocol P
- $\langle \mathbf{T}, \mathbf{B} \rangle$ 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 δ simultaneosly. 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 Δ

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 δ (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 Minimum Finding

AsFar $+$ delay in send $=$ **Speed**

- **IR** $\cup$ **Synch** $\cup$ *Ring* $\cup$ *InputSize*(2^c)
- **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 **AsFar** is (message with smallest id is newer trashed)

Complexity

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

- Message complexity smaller than the $\mathcal{O}(n\log(n))$ lower bound of asynchronous rings
- However, $\mathbf{T}[Speed] = \mathcal{O}(n2^{\mathbf{i}})$
- Exponential to the input values, worse than to size n.

TwoBits

- Situation: x wants to send bits α to his neighbor y with only two bits.
- $Int(1\alpha)$ is some known bijection between bit strings and integers.
- Silence is information also!

1. Entity
   1. Send "start counting"
   2. Wait $\mathit{Int}(1\alpha)$ 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 $c_2 - c_1 = \mathit{Int}(1\alpha)$, from which the α can be deduced.

Total Cost

- 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.

$$
\begin{aligned}
&Cost\,[Protocol\ P] = \langle \mathbf{B}\,[P]\,, \mathbf{T}\,[P] \rangle \\
&Cost\,[Speed(\mathbf{i})] = \langle \mathcal{O}(nlog(\mathbf{i})), \mathcal{O}(n2^{\mathbf{i}}) \rangle \\
&Cost\,[TwoBits(\alpha)] = \langle 2, \mathcal{O}(2^{|\alpha|}) \rangle
\end{aligned}
$$

Outline

Two-Party Communication Problem **TPC**

- 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 $q_1, \dots, q_{k-1}$.
- The ordered sequence $\langle p_0 : \mathbf{q}_1 : \cdots : \mathbf{q}_{k-1} : p_{k-1} \rangle$ 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 $\mathcal{C}_2$

- $\mathit{encode}(\mathbf{i}) = \langle b_0 : \mathbf{i} : b_1 \rangle$
- $\mathit{decode}(b_0 : \mathbf{q}_1 : b_1) = q_1$
- $\mathit{Cost}\,[\mathcal{C}_2(\mathbf{i})] = \langle 2, \mathbf{i} + 2 \rangle$

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 $\mathcal{R}_2(\mathbf{i})$ and the time reduces to $2 + \frac{\mathbf{i}}{4}$.
- From now on, we restrict ourselves to corruption tolerant TPC, and denote the communication sequence with only the quantas $\mathbf{q}_i$ as a tuple $\langle \mathbf{q}_1 : \cdots : \mathbf{q}_k \rangle$

3-bit Communicators

- *encode*$(\mathbf{i}) = \langle \lfloor \sqrt{\mathbf{i}} \rfloor : \mathbf{i} - (\lfloor \sqrt{\mathbf{i}} \rfloor)^2 \rangle$.
- *decode*$(\mathbf{q}_1 : \mathbf{q}_2) = \mathbf{q}_1^2 + \mathbf{q}_2$.
- *Cost* $[\mathcal{C}_2(\mathbf{i})] = \langle 3, 3 + \lfloor \sqrt{\mathbf{i}} \rfloor \rangle$, 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 $\mathcal{O}(\mathbf{i}^{\frac{1}{k}})$ 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 $(p-1)Time(C, I)$.

Pipleline - 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 $(p-1) + Time(C, I)$
- The bit cost is $(p-1)Bit(C, I)$ 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 $(p-1)Bits(C, I_{max})$, same as without pipeline.
- Time is reduced to $(p-1) + Time(C, I_{max})$.
- 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 of 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**
  $Cost\left[\tau\left[C\right](A)\right] \leq$
  $\langle M(A) Packets(C, m(A)), T_{causal}(A) Time(C, 2^{m(A)})\rangle$

An Example

- Election in a Synchronous Ring using Stages and a transformer $\longrightarrow$ *SynchStages*.
- *SynchStages* uses $\mathcal{O}(n \log n)$ bits and $\mathcal{O}(\mathbf{i}n \log n)$ time
- Better than the cost of time $\mathcal{O}(n \log \mathbf{i})$ and packets $\mathcal{O}(\mathbf{i}n2^{\mathbf{i}})$ of the **Speed**.

Outline