Ch 10 - Goals

- Introducing the synchronous abstraction
- Simulating the synchrnous abstraction
- Approximate agreement
- Pulse synchronization with better skew

Synchronous Abstraction

- At initialization each node may receive an input.
- Execution proceeds in rounds:
  - nodes perform local computations
  - send messages to their neighbors
  - receive the messages form their neighbors
  - optionally: report output value and terminate

- For a network G=(V,E) and a set F of faulty nodes, we denote by V_g the set of correct nodes.
- Synchronous execution of a deterministic algorithm at the correct nodes is totally determined by the input values and messages sent by the faulty nodes to correct nodes.
- A very clean abstraction.

Our Model

- We have logical clock algorithm that satisfies:
  - Upper bound on Logical Clock skew $\mathcal{G}$.
    - $\max_{v,w \in V_g}\{L_v(t)-L_w(t)\} \leq \mathcal{G}$
  - Bound on logical clock drift $\boldsymbol{\beta}$.
    - $t - t' \leq H_v(t) - H_v(t') \leq L_v(t) - L_v(t') \leq \boldsymbol{\beta}(t - t')$
  - The end-to-end message delay is d.
  - Assume that initial logical clock skew is also $\mathcal{G}$.

- In our model messages are being exchanged among the correct nodes and the faulty nodes inject their messages.
- The objective is to separate the message exchange into clean rounds, so each correct node will associate messages from correct nodes to separate rounds in a consistent way.

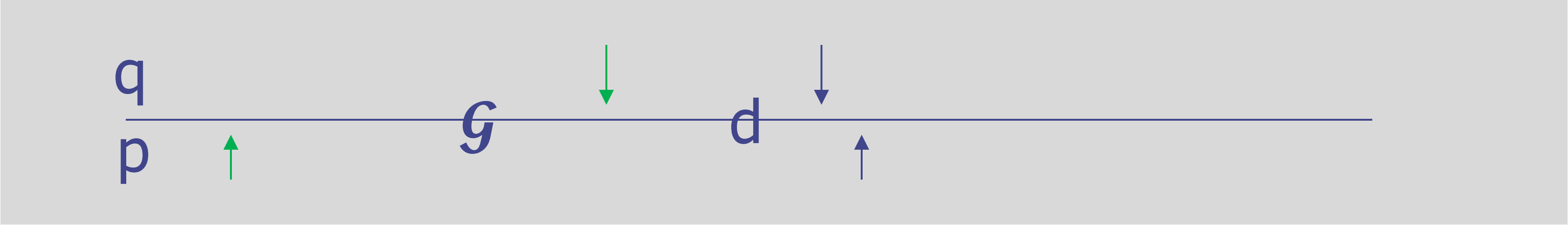
Breakout Room

- How one can obtain a synchronous abstraction using logical clocks?

Simulating Synchronous Abstraction

- Logical clocks satisfy:
  - Upper bound on Logical Clock skew $\mathcal{G}$.
  - Bound on logical clock drift $\boldsymbol{\beta}$.
  - end-to-end message delay d.
  - Assume that initial logical clock skew is also $\mathcal{G}$.

- Send round r messages at time t satisfying
  - $L_v(t) = \boldsymbol{\beta}\,\mathcal{G} + (r-1)\,\boldsymbol{\beta}\,(d + \mathcal{G})$

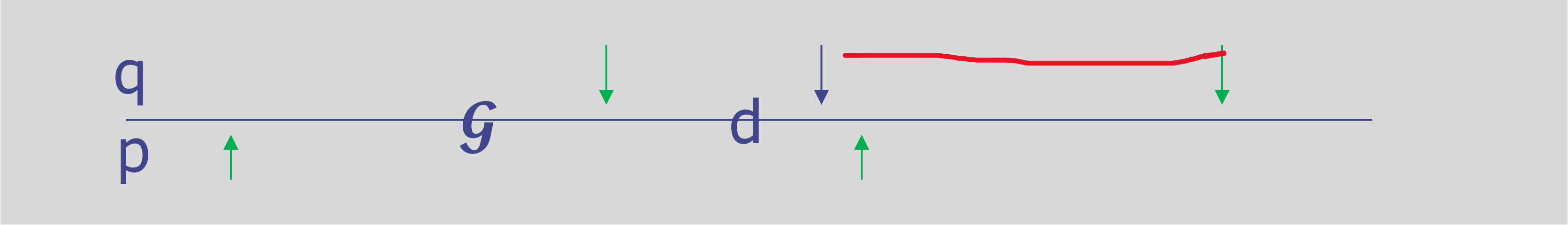


- Observe: all messages from correct nodes of round r arrive before any correct node needs to send messages for the next round, round r+1.

Simulating Synchronous Abstraction

- Logical clocks satisfy:
  - Upper bound on Logical Clock skew $\mathcal{G}$.
  - Bound on logical clock drift $\boldsymbol{\beta}$.
  - end-to-end message delay d.
  - Assume that initial logical clock skew is also $\mathcal{G}$.

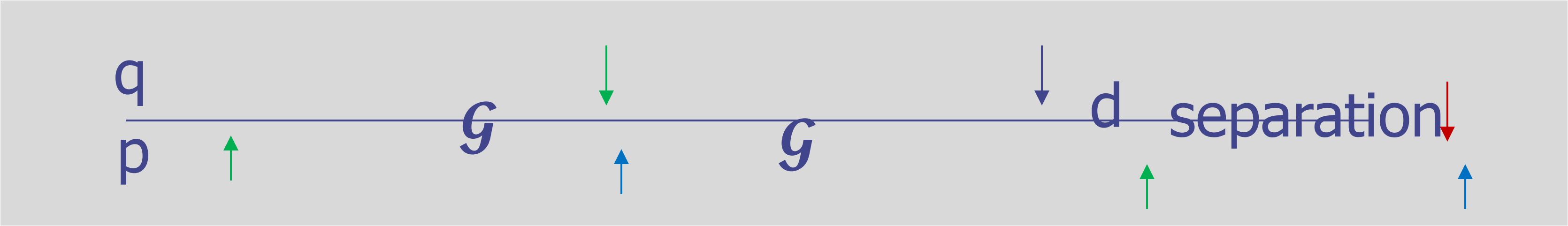
- Send round r messages at time t satisfying
  - $L_v(t) = \boldsymbol{\beta}\,\mathcal{G} + (r-1)\,\boldsymbol{\beta}\,(d + \mathcal{G})$



- The **<u>problem</u>** is that q still collects round r messages when p sends its round r+1 message.

Simulating Synchronous Abstraction

- Begin round r messages at time t satisfying
  - $L_v(t) = \beta \mathcal{G} + (r-1)\, \beta\, (d + 2\mathcal{G})$
- send round r messages at time
  - $L_v(t) = \beta \mathcal{G} + (r-1)\, \beta\, (d + 2\mathcal{G}) + \beta \mathcal{G}$



- p sends after q starts listening for the right round r

Approximate Agreement

- Each node $v \in V_g$ is given and input $r_v \in \mathcal{R}$. Given $\mathcal{E} > 0$. Generate output value $o_v \in \mathcal{R}$, such that
  - <u>agreement</u>: $\max_{v,w \in Vg}\{|o_v - o_w|\} \le \mathcal{E}$
  - <u>validity</u>: for each $v \in V_g$ is $\min_{w \in Vg}\{r_w\} \le o_v \le \max_{w \in Vg}\{r_w\}$
  - <u>termination</u>: each node $v \in V_g$ outputs its value, o_v, and terminate within finite number of rounds.

- Thus, nodes need to exchange their input values and try (iteratively) to compute output values that satisfy the above requirement.
- Faulty nodes may send arbitrary values to try to prevent the convergence of output values.

Approximate Agreement Algorithm

The algorithm proceeds in rounds.
The input to each round is the output of the previous round.
The initial input is the input of the first round.

<u>The Basic Iteration</u>:

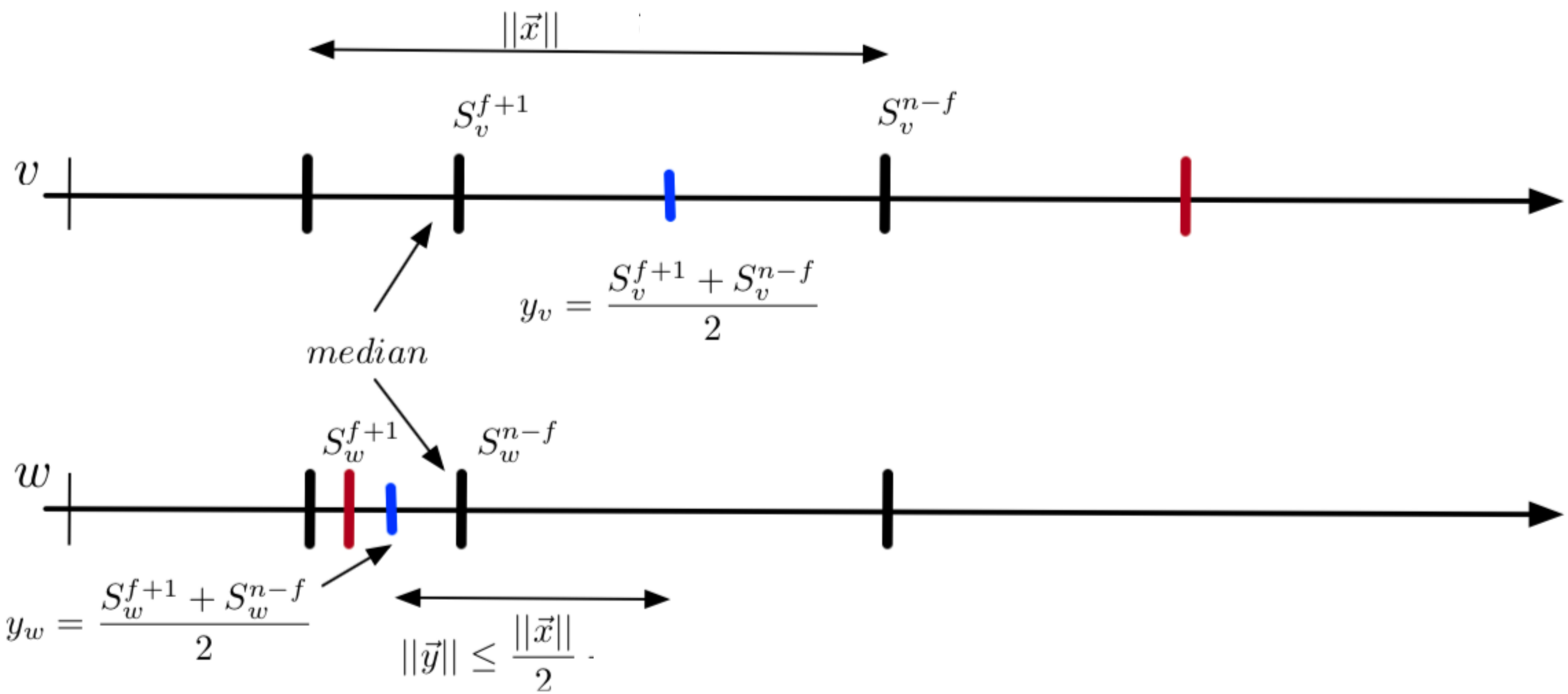
1. send r_v to all.
2. receive $r_{w,v}$, the value sent by w in this round.
   // replace any "missing" value by r_v
3. $S_v := \{r_{w,v}\}$; //ordered set
4. $o_v := (S_v^{(f+1)} + S_v^{(n-f)})/2$; // the (f+1)st and (n-f)-th values in S
5. Return o_v

Notations

- We denote by r_g the ascending vector of the inputs of all correct nodes. For simplicity let's assume that $|V_g|=n-f$. We assume that $n=3f+1$.
- For any ascending vector, say x, $x^{(i)}$ denotes the i-th entry in the vector.
- $[\![x]\!]$, the diameter of x, is the difference between the maximal and minimal values in x.

We will show some basic properties of the algorithm and that each iteration of the algorithm reduces the diameter by half.

4 nodes – 1 faulty



Lemma 10.4

For each $v \in V_g$ is $r_g^{(1)} \leq o_v \leq r_g^{(n-f)}$

Proof: notice that in every iteration we remove the bottom f values and there are at most f faults.

Therefore, for each $v \in V_g$

$$S_v^{(f+1)} \geq \min_{w \in Vg}\{r_{wv}\} \geq r_g^{(1)}$$

$$S_v^{(n-f)} \leq \max_{w \in Vg}\{r_{wv}\} \leq r_g^{(n-f)}$$

The value computed in the iterations satisfies

$$r_g^{(1)} \leq S_v^{(f+1)} \leq o_v := (S_v^{(f+1)} + S_v^{(n-f)})/2 \leq S_v^{(n-f)} \leq r_g^{(n-f)}$$

Lemma 10.5

In each iteration, $[\![o]\!] \leq [\![r_g]\!]/2$

Proof: since 3f+1=n, n-f=2f+1, for every $v \in V_g$

$$r_g^{(1)} \leq S_v^{(f+1)} \leq r_g^{(f+1)} \leq S_v^{(2f+1)} \leq S_v^{(n-f)} \leq r_g^{(n-f)}$$

Therefore, by Lemma 10.4,

$$o_v - o_w = (S_v^{(2f+1)} - S_w^{(2f+1)} + S_v^{(n-f)} - S_w^{(n-f)})/2$$

By the above

$$o_v - o_w \leq (r_g^{(f+1)} - r_g^{(1)} + r_g^{(n-f)} - r_g^{(f+1)})/2$$

and we get

$$o_v - o_w \leq (r_g^{(n-f)} - r_g^{(1)}) /2 = [\![r_g]\!]/2$$

Theorem 10.6

Let $R \geq r_g^{(n-f)} - r_g^{(1)}$. After log (R/ Ɛ) iterations the algorithm obtains the approximate agreement properties.

Proof: In each iteration, $[\![o]\!] \leq [\![r_g]\!]/2$, and for every two correct nodes, $o_v - o_w \leq (r_g^{(n-f)} - r_g^{(1)})/2$.

The rest is immediate.