Improved multi-party contract signing

Aybek Mukhamedov and Mark Ryan

March 2, 2007

Digital Contract Signing

- Use digital signatures to sign a pre-agreed contract over a computer network
- Potentially useful for e-commerce
- Why it is not simple:

$$\begin{array}{lcll} A & \longrightarrow & B & : Sign_A(contract) \\ B & \longrightarrow & A & : Sign_B(contract) \end{array}$$

Someone has to start first.

Contract Signing protocol

- Main property: fairness
  - 2-party: if A gets B‘s signature, then B can get A‘s signature, and vice-versa
  - n-party: if any agent gets a signature from any other agent, then all agents can get signatures from every other agent.
- Must not fail in the presence of an active adversary on the network
- ... controlling a coalition of up to $n - 1$ dishonest agents

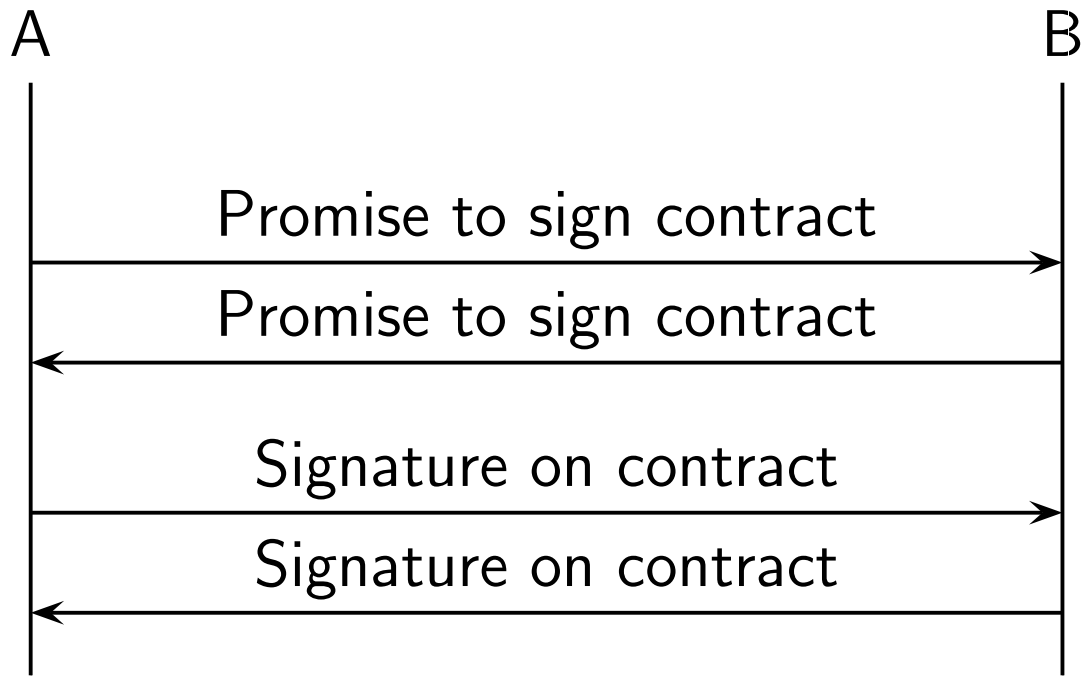
Approaches to obtaining fairness

- Use trusted party T to collect and distribute the signed contracts
  - Problem: T may become a bottleneck.
- Optimistic protocols:
  - The agents can complete the contract signing without T (optimistic case)
  - T will be invoked and will take decisions iff something goes amiss.
  - Channels between parties and T are resilient.

Outline

1. Multi-party contract signing
2. 'Optimistic' protocols
3. A protocol by Garay and MacKenzie
4. New protocol and its properties
5. Conclusions

"Optimistic" protocols: 2-party



- T will enforce the contract if presented with both promises
- More involved for n-party

"Optimistic" protocols: T

- T can enforce the contract by converting promises to signatures
  - it will do so if it has proof that all parties have issued a promise
- T can issue an abort token
  - 2-party: means that it will not enforce contract
  - n-party: means that it will not enforce contract; but it may overturn this abort decision if presented with evidence of cheating by the signer that got the abort
- T acts only when requested by an agent
  - decides whether to abort or resolve based on the evidence in the complaint

"Optimistic" multi-party contract signing protocols

- Baum-Waidner, Waidner, ICALP'00
- Garay, MacKenzie, DISC'99:
  - Attack and fix by Chadha, Kremer, Scedrov and formal analysis for runs with three and four signers (CSFW'04)
  - "Impossible to fix" for runs with five and more signers by Mukhamedov and Ryan (CSFW'06)

GM: main protocol

$P_i \quad P_{i-1} \quad \ldots \qquad\qquad P_1$

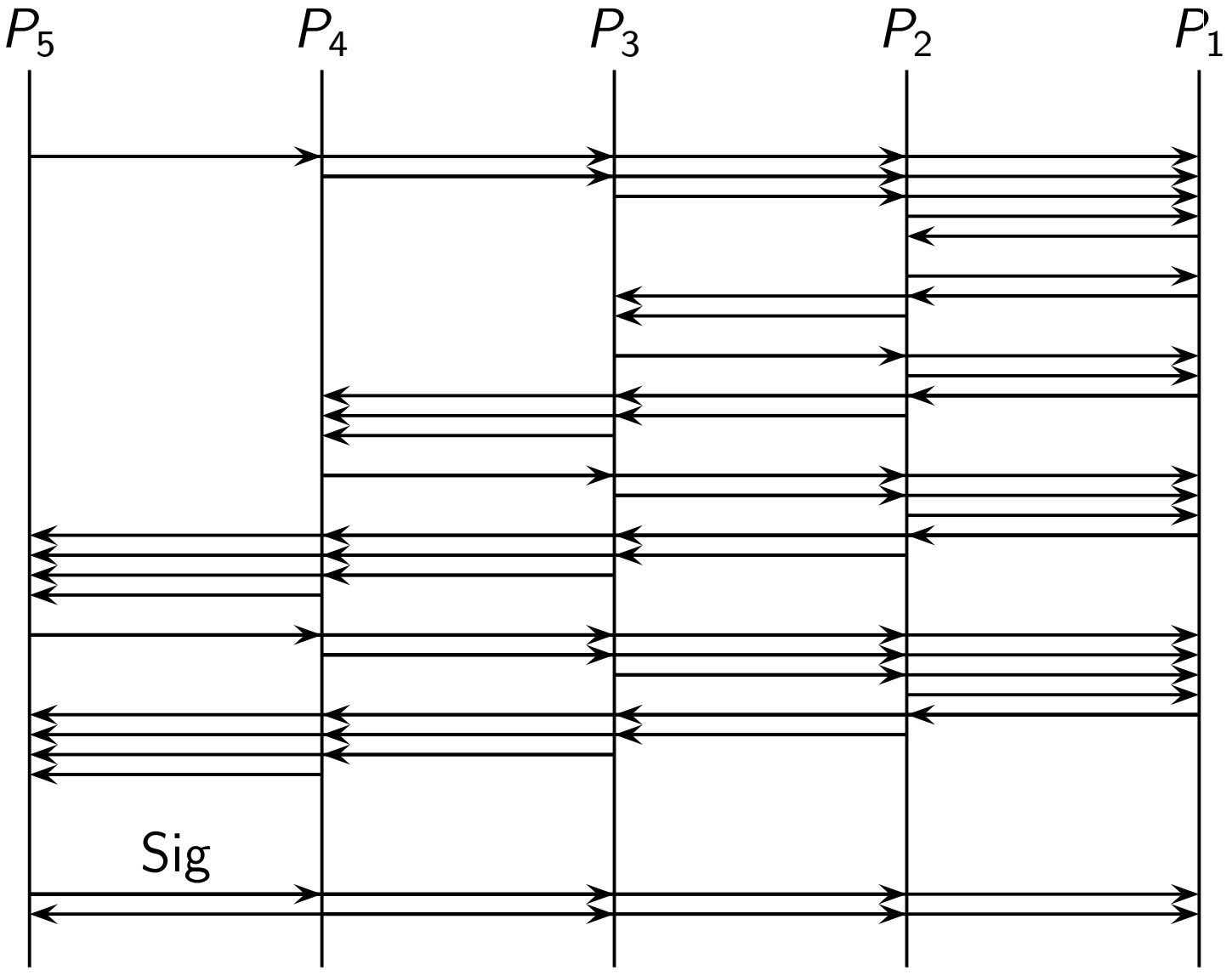
Distribute 1-level promises to $P_{<i}$

$i-1$-level protocol

Collect $i-1$-level promises

Exchange i-level promises

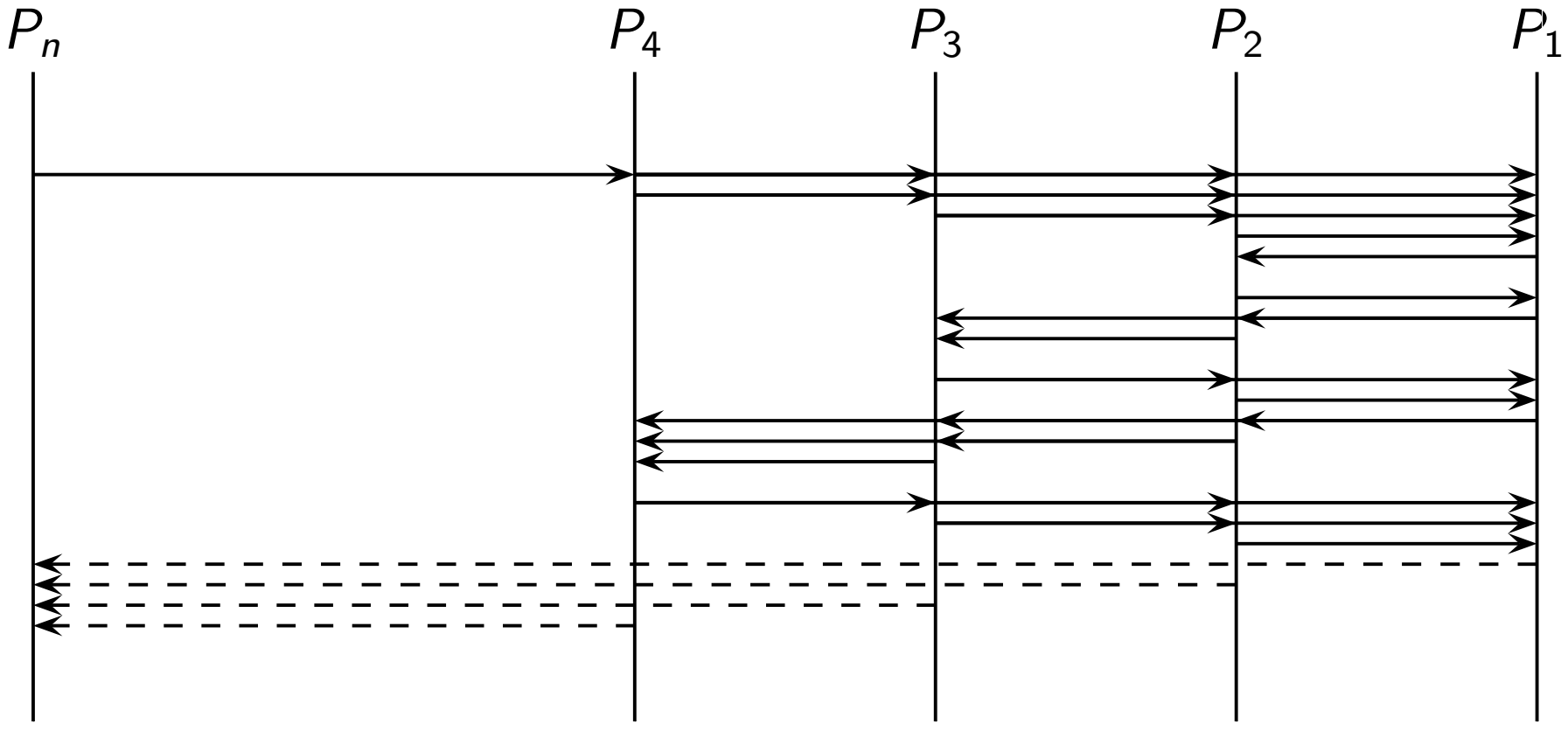
GM main protocol: five signers



Resolve-impossibility for GM protocol

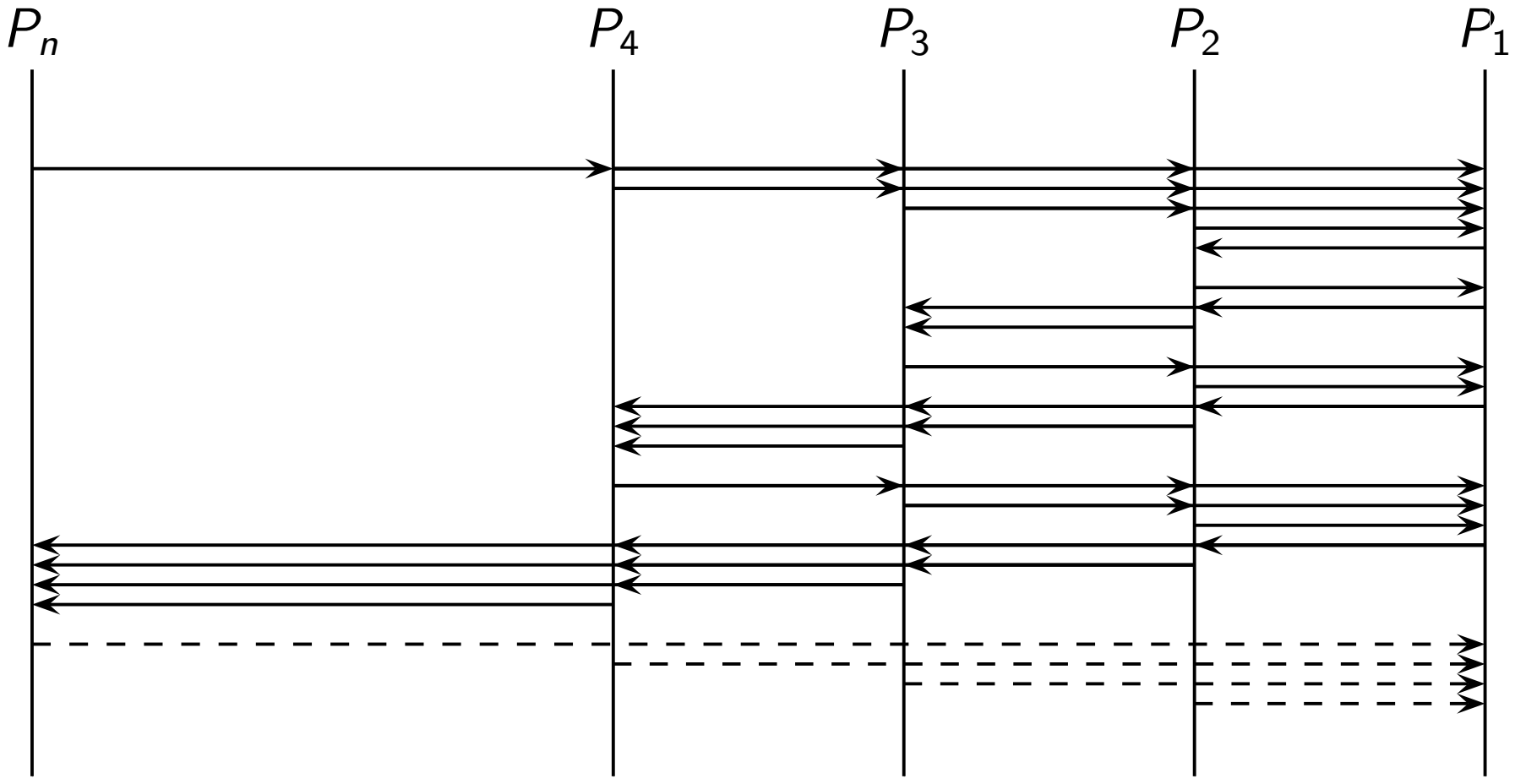
- Attacks do not depend on the resolve protocol:
  - for any resolve protocol, the main protocol is subject to attacks on fairness
- Resolve impossibility follows from case-by-case analisys of T's actions in the previous attack:
  - no matter what T does, it is unfair to someone, who could be honest.

Resolve impossibility for GM protocol



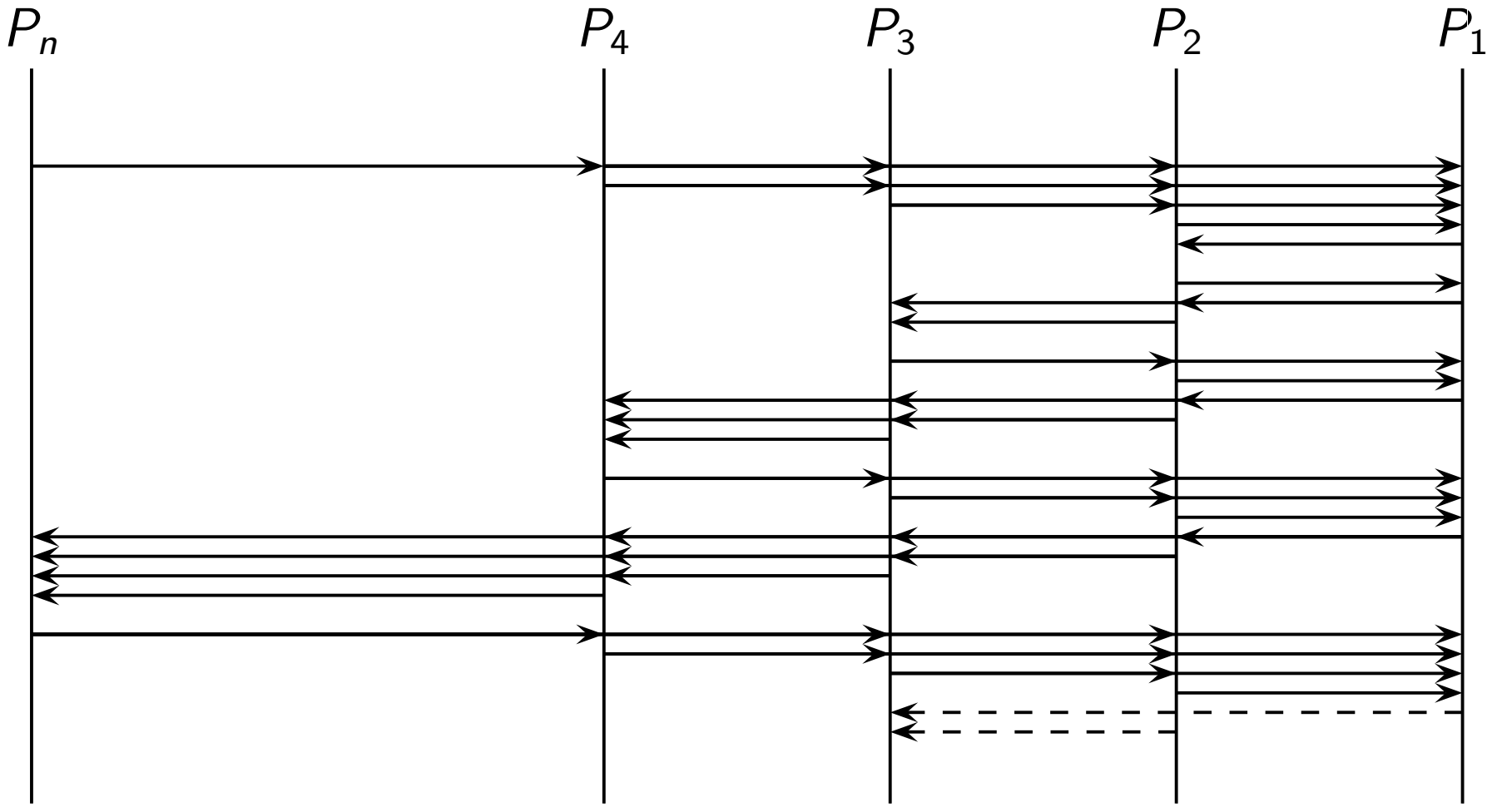
If P_n requests abort claiming not to have received dotted messages, T must grant it.

Resolve impossibility for GM protocol



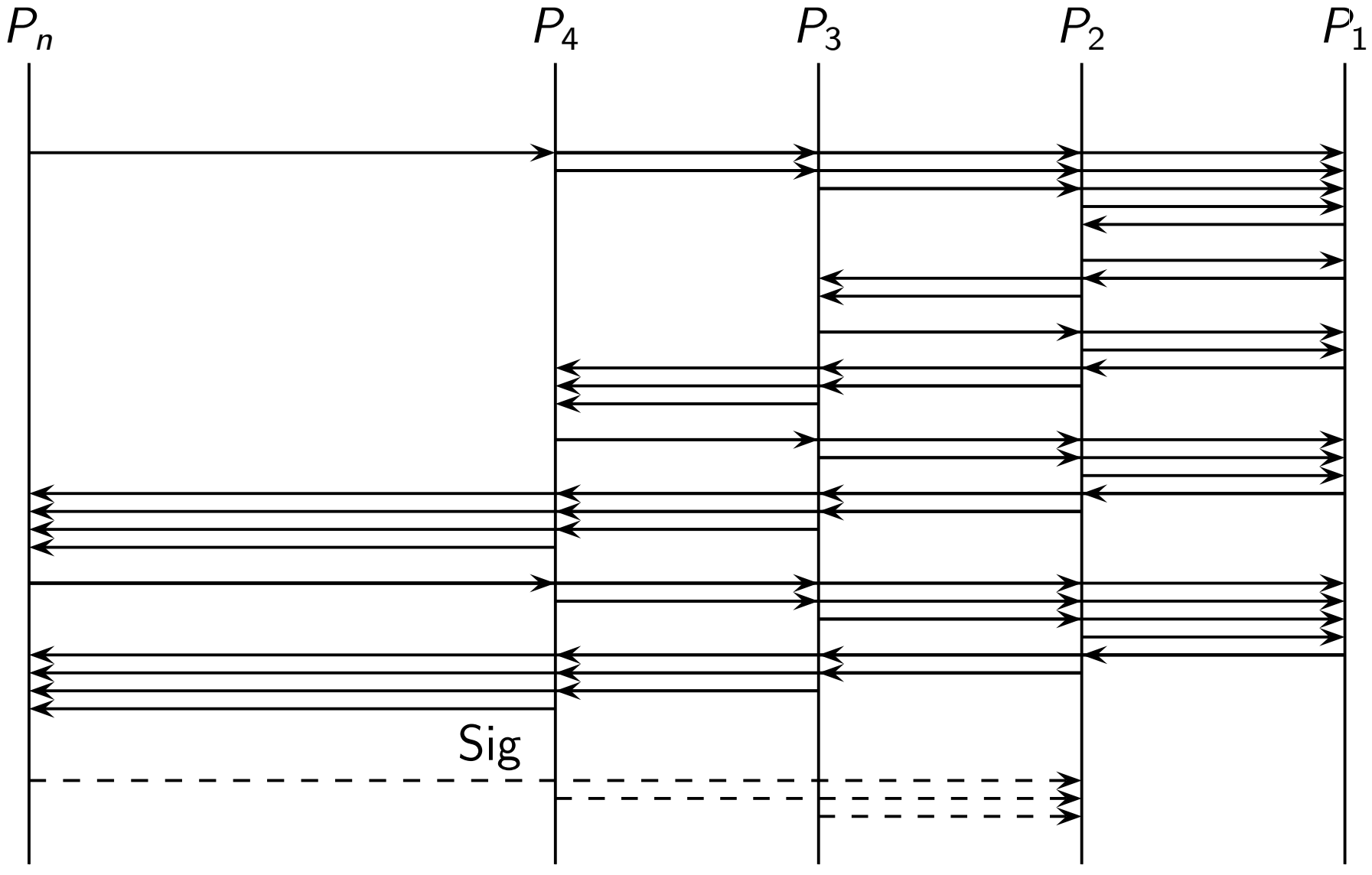
If P_1 requests resolve, T must confirm previous abort.

Resolve impossibility for GM protocol



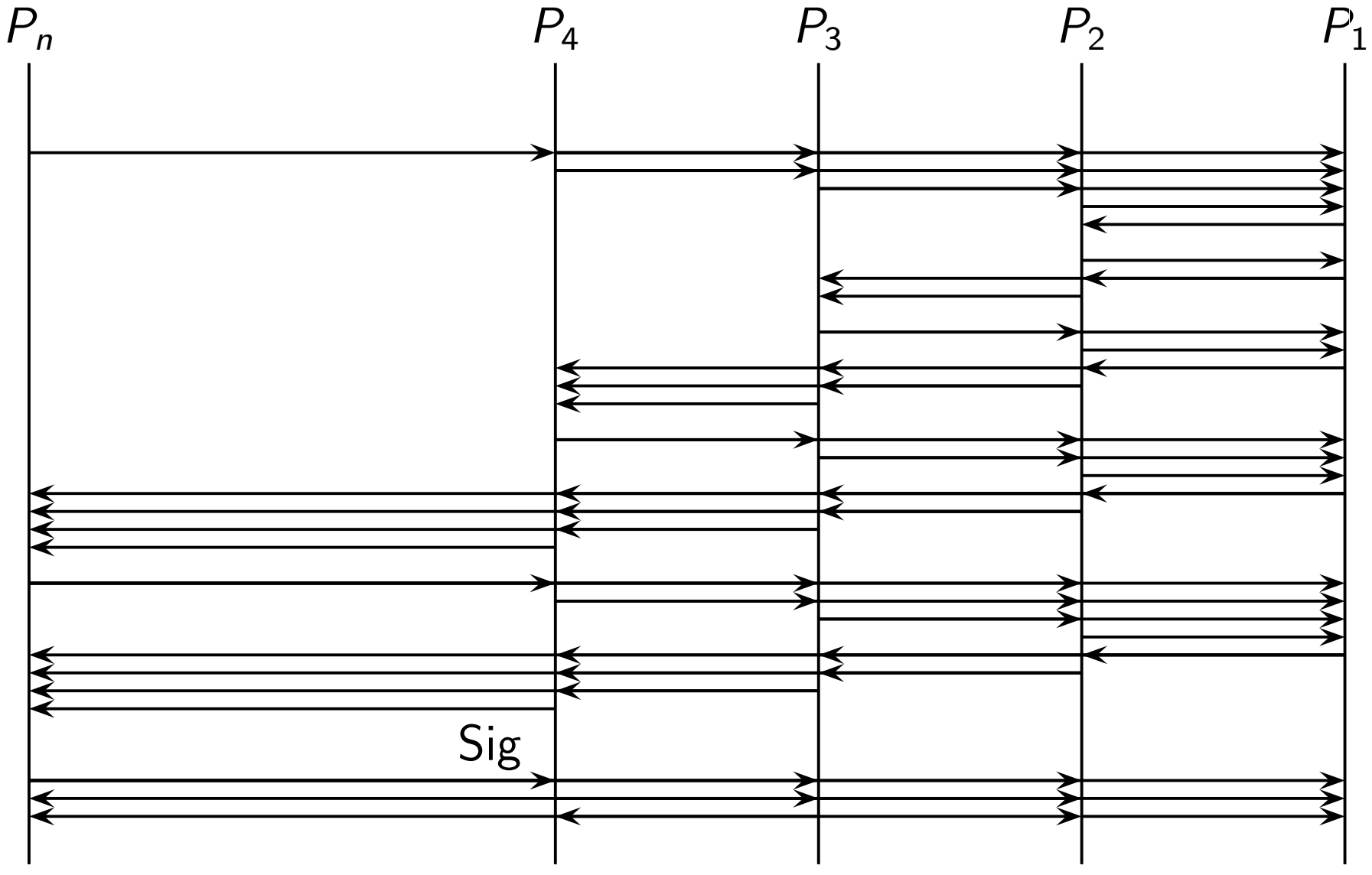
If P_3 requests resolve, T must still confirm previous abort

Resolve impossibility for GM protocol



If P_2 requests resolve, T must still confirm previous abort

Resolve impossibility for GM protocol



New optimistic contract signing protocol

- Uses private contract signature primitive (Garay et al, Crypto'99):
  - $PCS_A(m, B, T)$ is a *promise* from A to B on m
  - Only B and T can verify its validity
  - T can convert it into a conventional digital signature that binds A on m
- Two parts:
  - Main protocol: defines actions for signers
  - Resolve and Abort protocols: define actions for a T

New contract-signing protocol

- Depending on the level of the protocol execution a signer P_i may:
  - Quit the protocol P_i if did not send any promises
  - Request T to intervene:

    T replies with a resolved contract or an abort token

- Each signer may contact T only once
- T may overturn its abort decision, but never overturns a resolve decision

Main protocol for signer P_i

Round 1

1. For each $j < i$, wait for promise $PCS_{P_j}((m, 1), P_i, T)$ from P_j.
   If not received in timely manner, then quit.
2. For each $j > i$, send promise $PCS_{P_i}((m, 1), P_j, T)$ to P_j.
3. For each $j > i$, wait for promise $PCS_{P_j}((m, 1), P_i, T)$ from P_j.
   If not received in timely manner, then request abort.
4. For each $j < i$, send promise $PCS_{P_i}((m, 1), P_j, T)$ to P_j.

Main protocol for signer P_i

Round r: for $r = 2$ to $\lceil n/2 \rceil$:

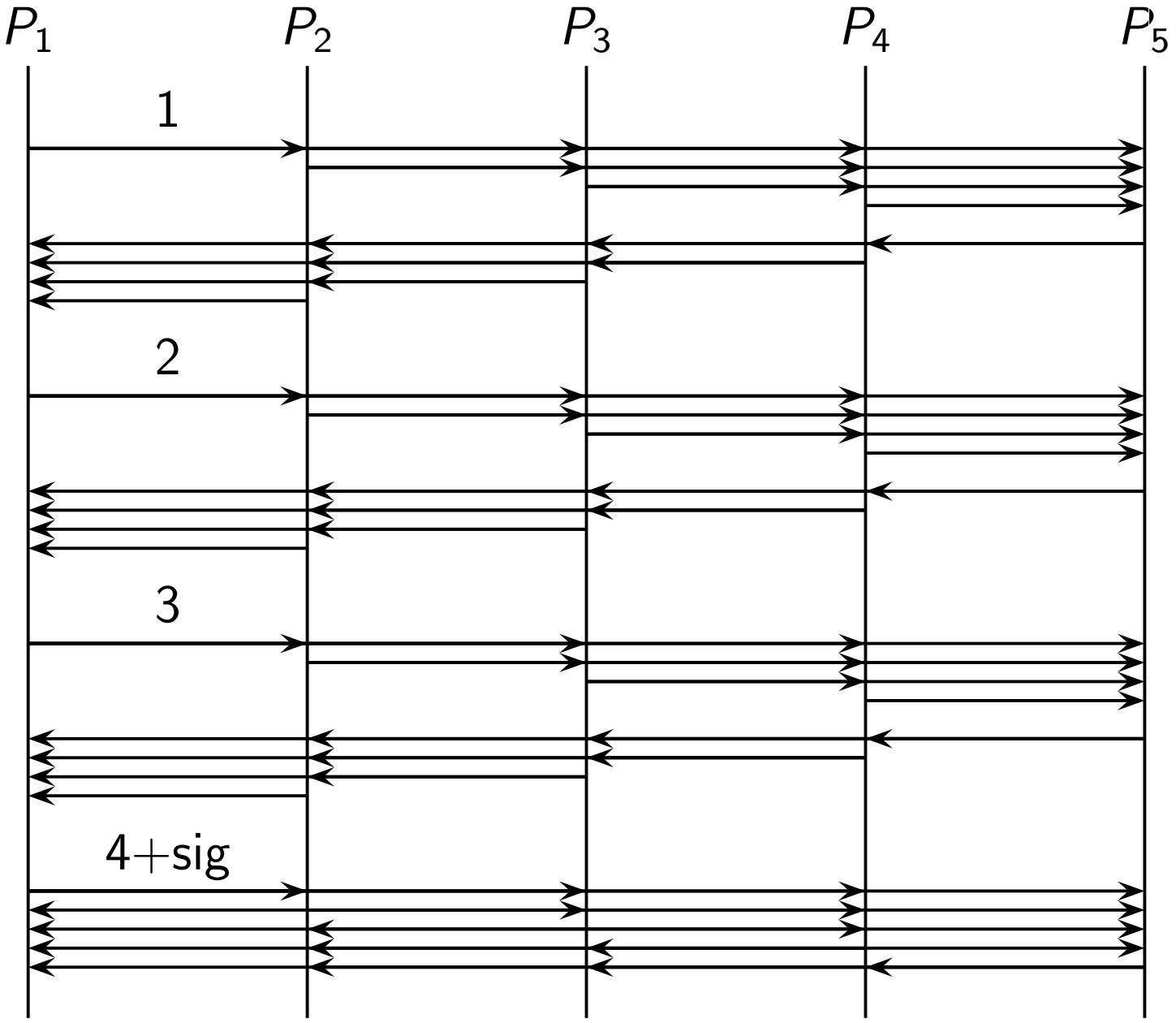
5. For each $j < i$, wait for promise $PCS_{P_j}((m, r), P_i, T)$ from P_j.
   If not received in timely manner, then request resolve.
6. For each $j > i$, send promise $PCS_{P_i}((m, r), P_j, T)$ to P_j.
7. For each $j > i$, wait for promise $PCS_{P_j}((m, r), P_i, T)$ from P_j.
   If received in timely manner, then request resolve.
8. For each $j < i$, send promise $PCS_{P_i}((m, r), P_j, T)$ to P_j.

Main protocol for signer P_i

Round $\lceil n/2 \rceil + 1$

9. For each $j < i$, wait for promise $PCS_{P_j}((m, \lceil n/2 \rceil + 1), P_i, T)$ and signature $S_{P_j}(m)$ from P_j.
   If not received in timely manner, then request resolve.
10. For each $j \neq i$, send promise $PCS_{P_i}((m, \lceil n/2 \rceil + 1), P_j, T)$ and signature $S_{P_i}(m)$ to P_j.
11. For each $j > i$, wait for promise $PCS_{P_j}((m, \lceil n/2 \rceil + 1), P_i, T)$ and signature $S_{P_j}(m)$ from P_j.
    If not received in timely manner, then request resolve.

New contract signing protocol: 5 signers



Main protocol

- P_i requests abort with:

$$S_{P_i}(m, P_i, (P_1, \ldots, P_n), \texttt{abort})$$

- P_i requests recovery with:

$$S_{P_i}(\{PCS_{P_j}((m, \tau_j), P_i, T)\}_{j \in \{1,\ldots,n\} \setminus \{i\}}, S_{P_i}((m, 1)))$$

where τ_j is the (appropriate) level of promise from P_j to P_i.

Protocol for a trusted party T

- Two sub-protocols: resolve and abort
- Uses Chadha, Kremer and Scedrov's (CSFW'04) idea for implementation of the trusted party
- T stores names of agents in a set $S(m)$ to whom it has replied with abort
- For each P_i in $S(m)$, T deduces the highest level promises P_i could have sent to higher and lower indexed agents:
  - T infers P_i's dishonest iff it is later presented with a higher level promise issued by P_i
- Abort is overturned iff T infers that each signer that contacted it in the past has been dishonest

New protocol: properties

- Fairness:
  - Lemma 1: If a resolve request in round $r > 1$ results in an abort decision, then: for all r' s.t. $1 < r' < r$ there are two resolve requests in round r' that resulted in an abort decision, and an abort request in round 1
  - Lemma 2: If some P_i gets abort and then later P_j gets resolve, then P_i was dishonest (continued the protocol).
  - Theorem: Protocol is fair, even if there are $n - 1$ dishonest signers.
- Abuse freeness
  - intuitively follows from properties of private conract signatures

New protocol: properties

- Comparison with the other protocol (Baum-Waidner and Waidner, ICALP'01)
  - efficiency: requires half the number of messages for an optimistic run (for $n = 6$ it requires 120 vs 210 in BW protocol)
  - uses standard notion of a signed contract (in BW protocol $\sigma_A((m, i))$ is a contract only if $i = n + 1$, otherwise mere a promise)

Conclusion and further work

- Currently the only fair multi-party contract signing protocol employing standard notion of signed contract.
  - also satisfies timeliness and abuse freeness
  - half number of messages compared to the other solution

Further work

- Formalise the notion of abuse-freeness (cf. Kuesters et al, ICALP'06)
- Mechanise proof using Isabelle or PVS
  - challenging because it's an "open-ended" protocol (cf. Meadows, FMSE'02)