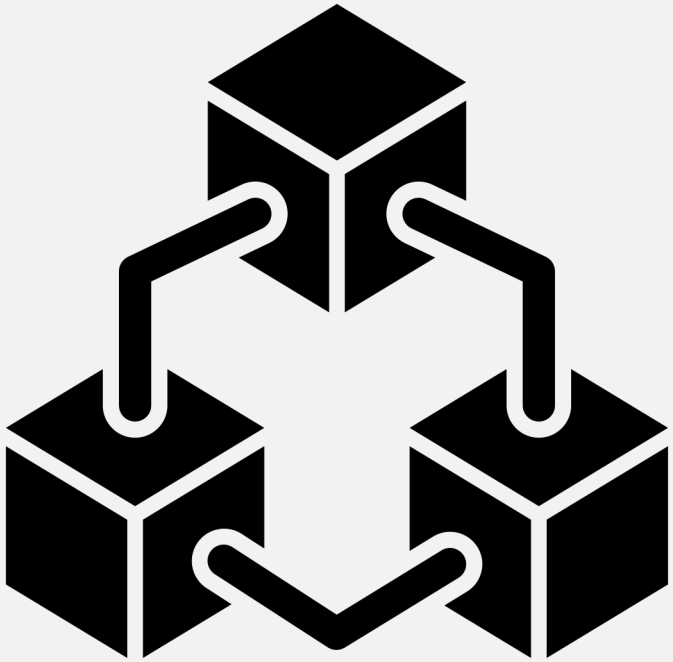
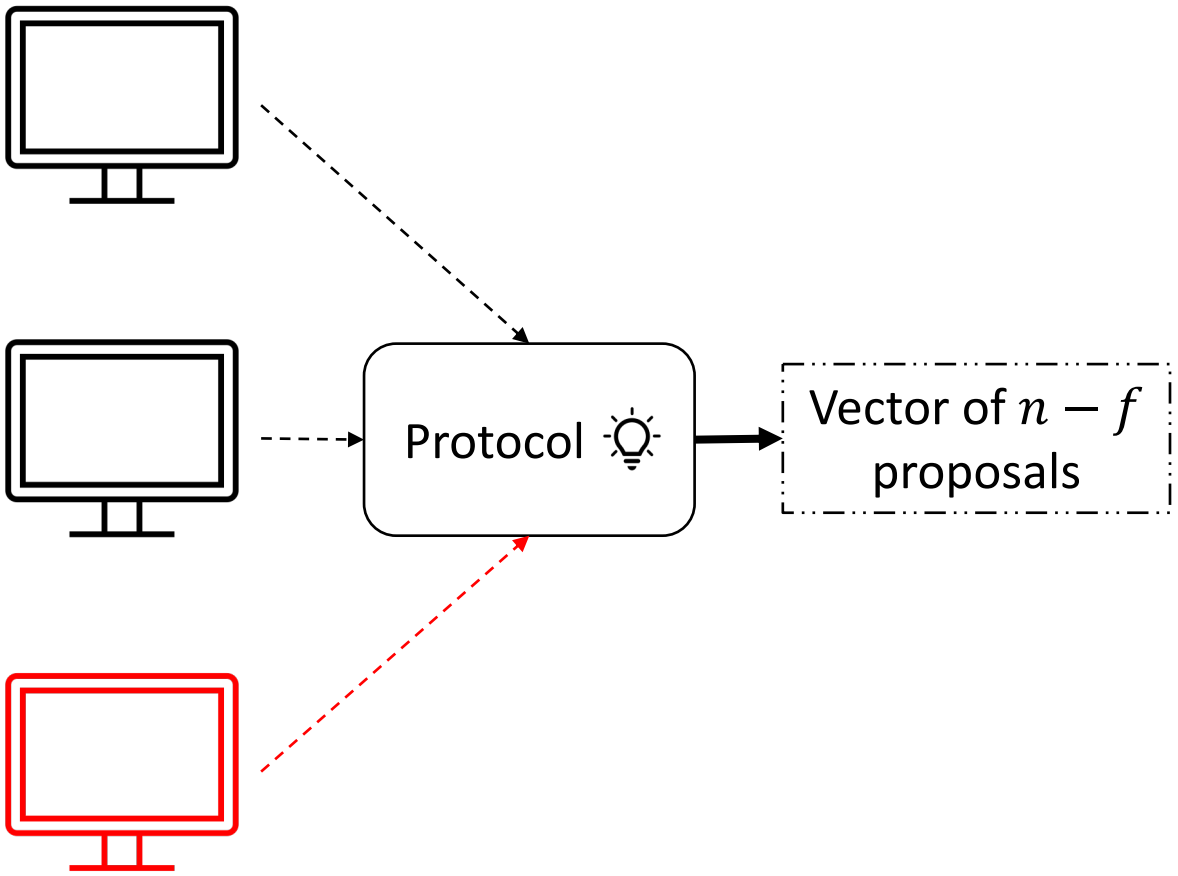


Improving the Complexity of Byzantine Vector Consensus

Manos Chatzakis
emmanouil.chatzakis@epfl.ch

EPFL

What is Byzantine Vector Consensus?

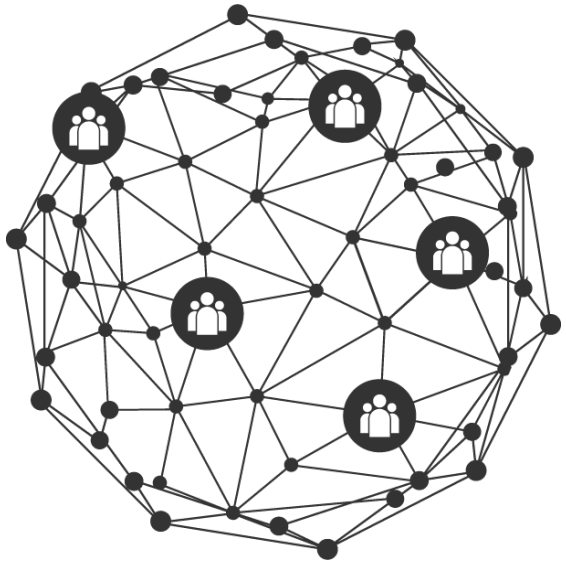


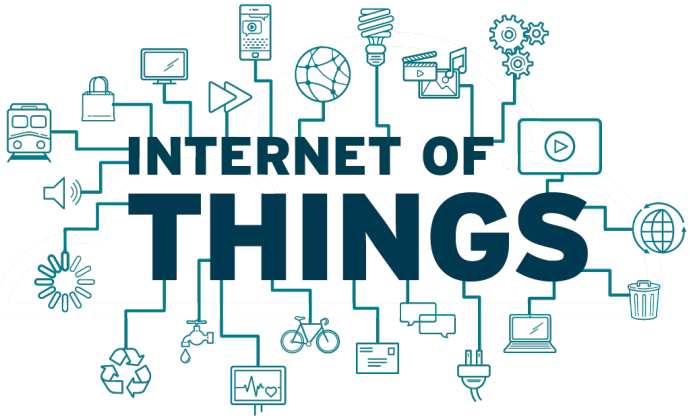
- System: n processes
  - f Byzantine
  - $n = 3f + 1$
- Decide a **vector of $(n - f)$** distinct **proposals**
- Partial Synchrony: Delays are initially unbounded, until $\boldsymbol{GST}$
  - Delays bounded by δ
  - Clocks do not drift

Why do we care?

- Interesting problem for distributed computing

- Real-world applications
  - Blockchain
  - Decentralized Computing
  - IoT

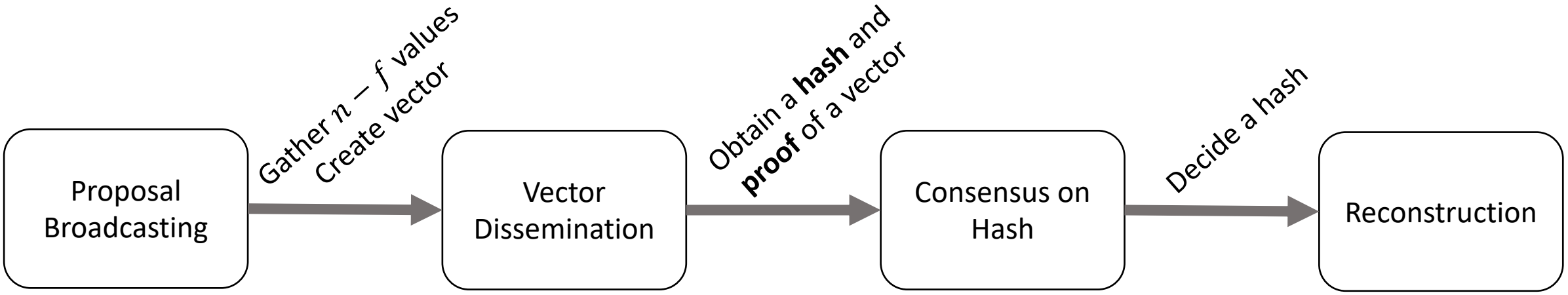




Existing solutions cannot work

- Byzantine Consensus is $O(n^2)$: QUAD
  - For vector consensus: $O(n^3)$ communication complexity, $O(f)$ latency
  - Message size is $O(n)$
- Alternative: Work with hashes instead of vectors
  - $O(n^2 \log(n))$ communication complexity, $O(n^f)$ latency
- Can we do better?
  - We achieved $\boldsymbol{O(n^2\sqrt{n})}$ **communication complexity** and $\boldsymbol{O(n\sqrt{n})}$ **latency**!

Structure of Byzantine Vector Consensus



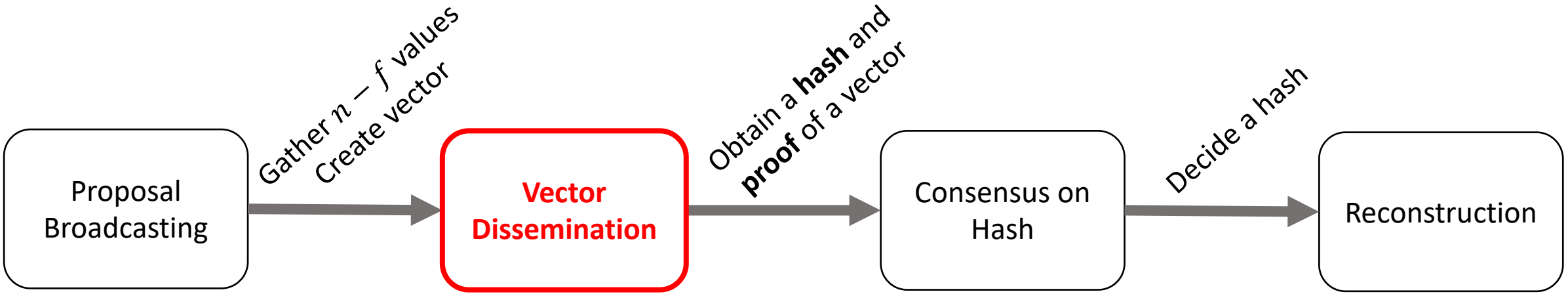
Communication Complexity

$O(n^2)$	???	$O(n^2)$	$O(n^2 \log(n))$

Latency

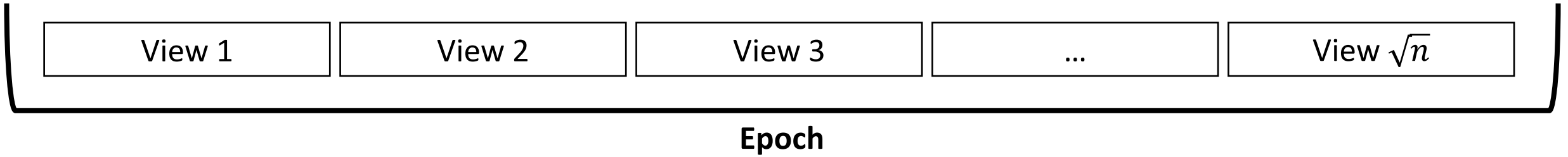
$O(1)$	???	$O(f)$	$O(1)$

Structure of Byzantine Vector Consensus



Communication Complexity	$O(n^2)$	$\boldsymbol{O(n^2\sqrt{n})}$	$O(n^2)$	$O(n^2 \log(n))$
Latency	$O(1)$	$\boldsymbol{O(n\sqrt{n})}$	$O(f)$	$O(1)$

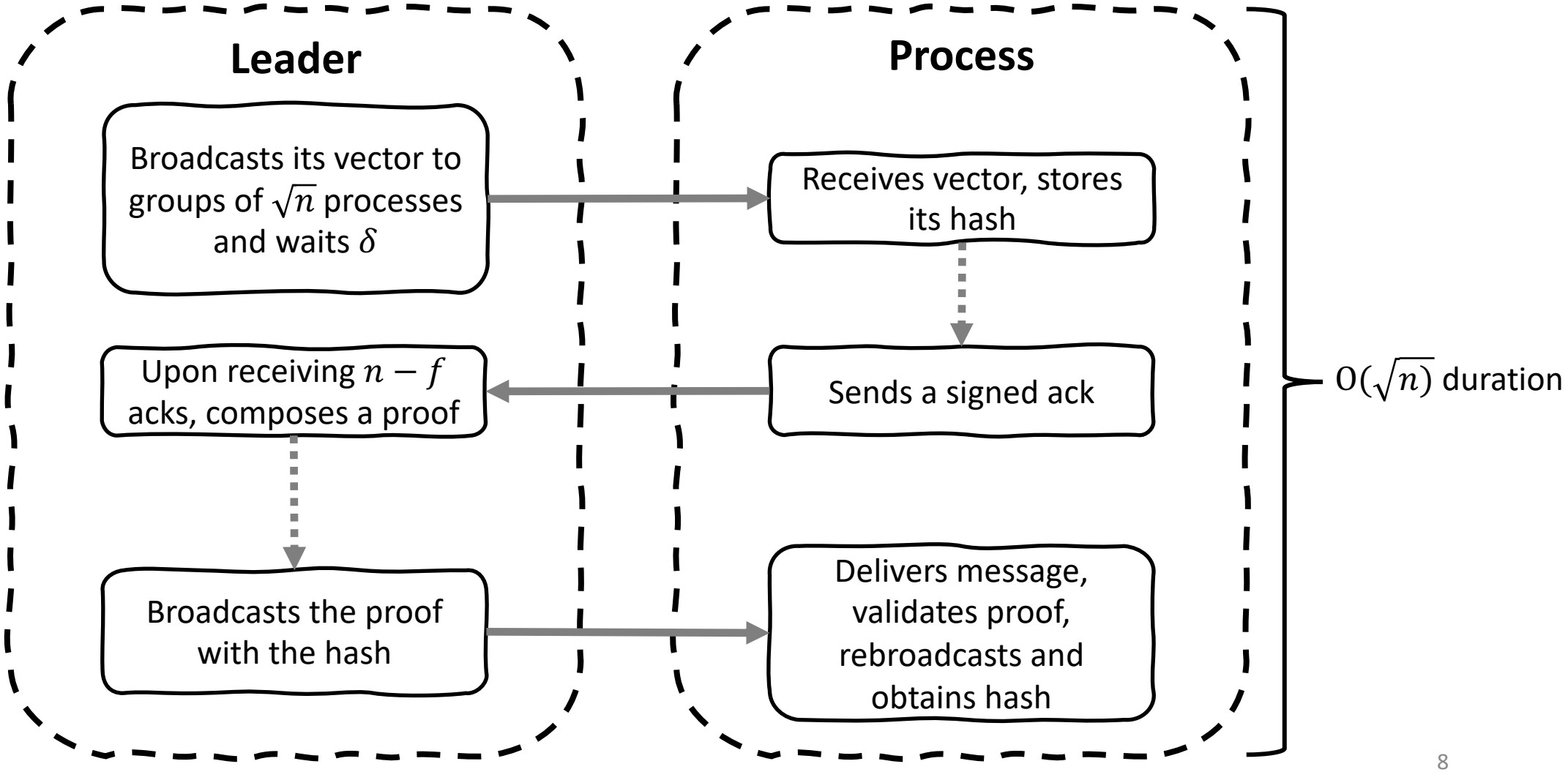
Leader-Based Vector Dissemination



- Each view has a leader process that tries to disseminate their vector (**View Core**)

- Hash obtained when all processes overlap in a view with correct leader
  - Processes advance over views until they synchronize (**View Synchronizer**)

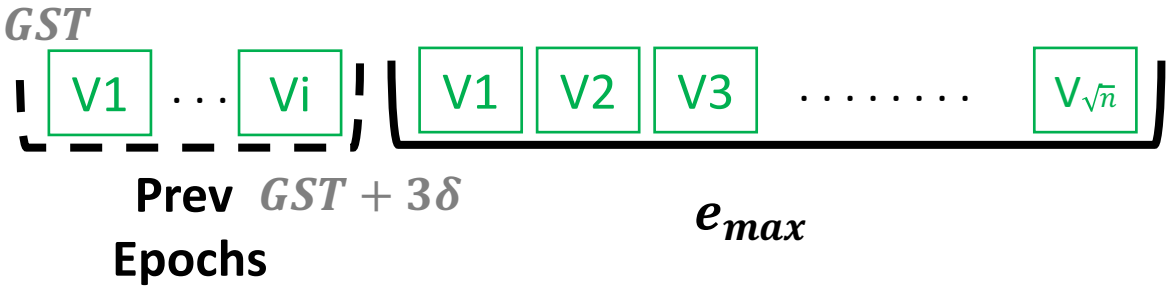
View Core


Leader
Process
Broadcasts its vector to groups of $\sqrt{n}$ processes and waits δ
Receives vector, stores its hash
Upon receiving $n - f$ acks, composes a proof
Sends a signed ack
Broadcasts the proof with the hash
Delivers message, validates proof, rebroadcasts and obtains hash
$O(\sqrt{n})$ duration

View Synchronizer

- RareSync: Processes overlap for every view of each epoch entered after GST
  - $\sqrt{n}$ views
  - $f + 1$ views (worst case) to reach a correct leader
- Processes communicate only at start/end of epoch
  - Local clocks to advance inside epochs
  - Wait δ before entering new epoch
- **Synchronization time**: Processes synchronized, correct leader: Obtain hash!

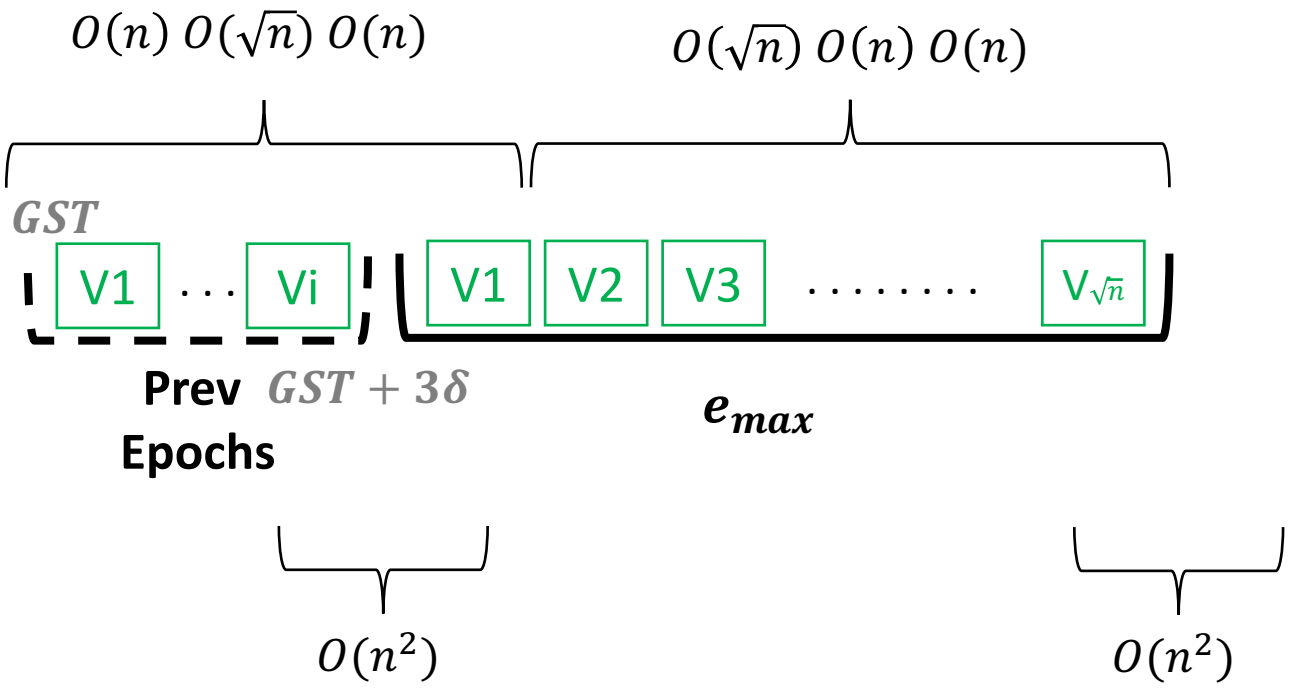
Communication Complexity



- e_{max}: Epoch of most advanced process at GST.

- Worst case:
  - At GST, all correct processes are leaders
  - Most advanced process in e_{max}: All correct processes will enter e_{max} by $GST + 3\delta$
    - All leaders correct but processes misaligned

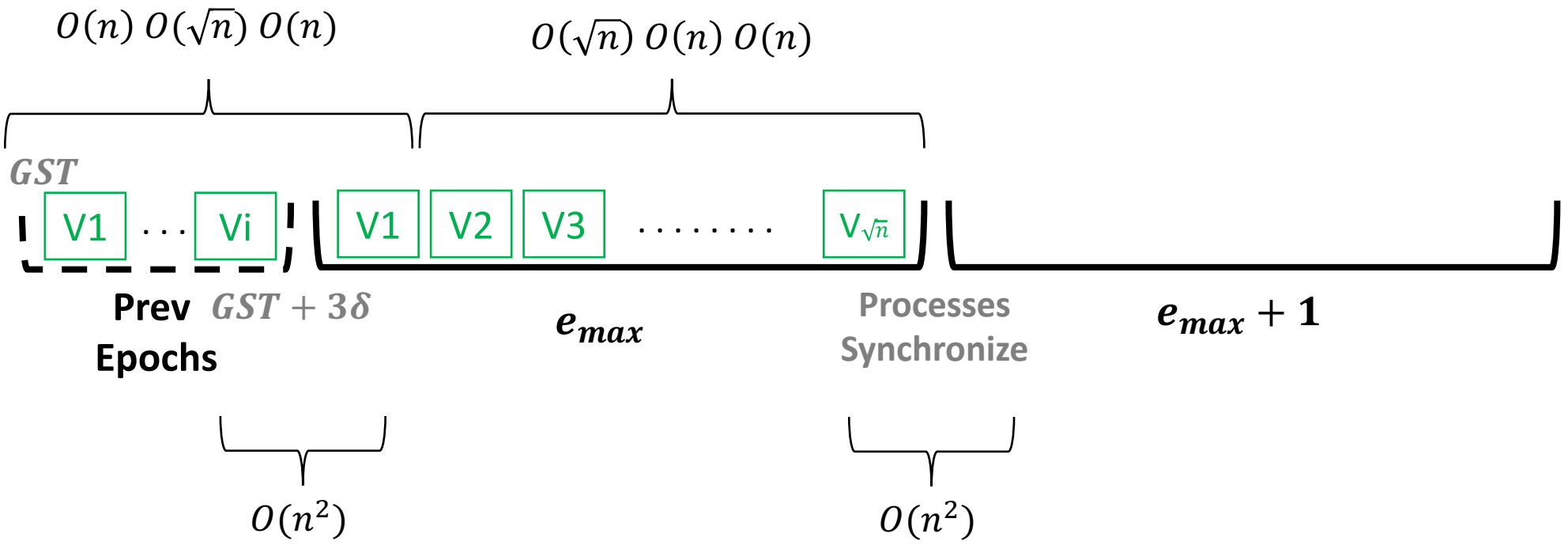
Communication Complexity



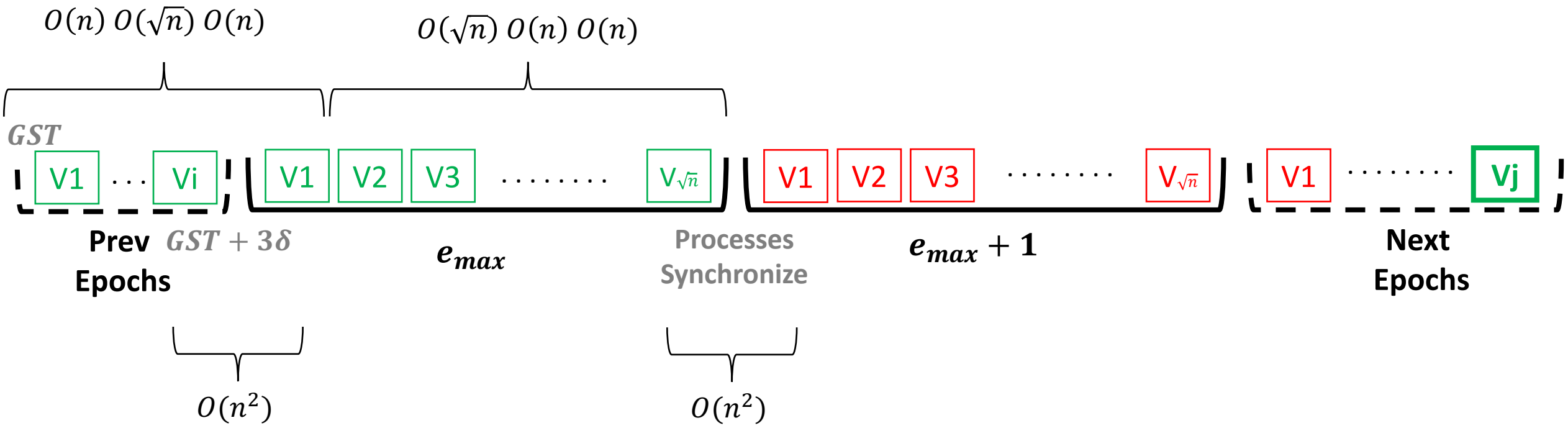
- e_{max}: Epoch of most advanced process at GST.

- Worst case:
  - At GST, all correct processes are leaders
  - Most advanced process in e_{max}: All correct processes will enter e_{max} by $GST + 3\delta$
    - All leaders correct but processes misaligned

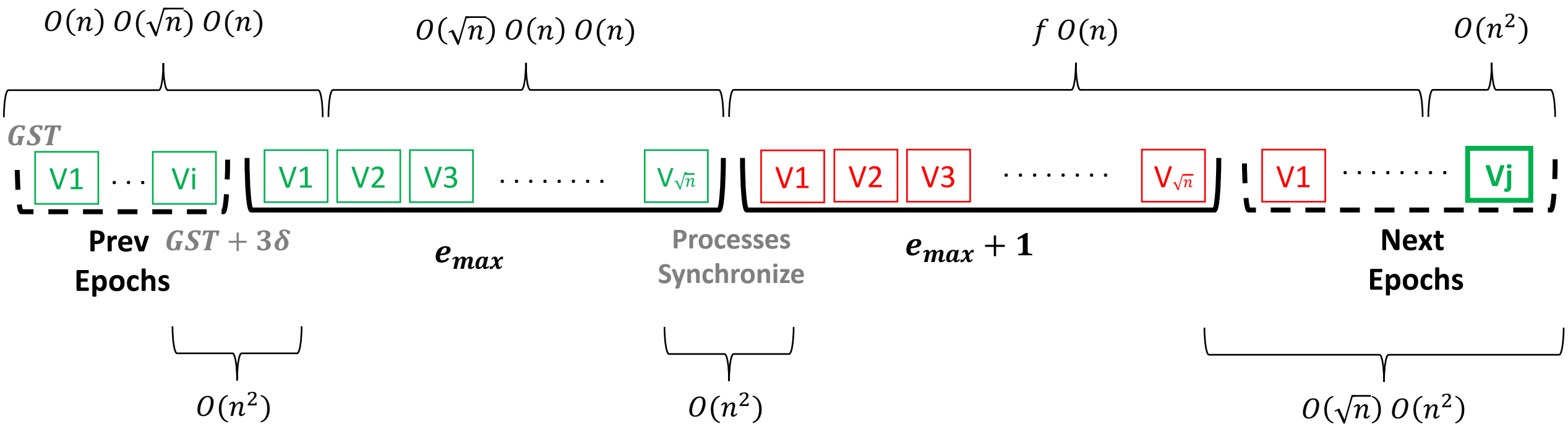
Communication Complexity



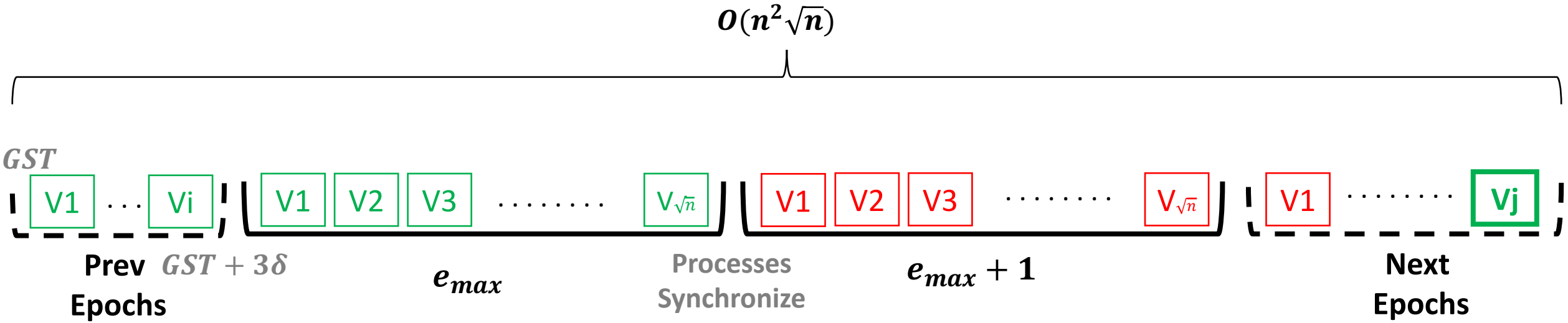
Communication Complexity



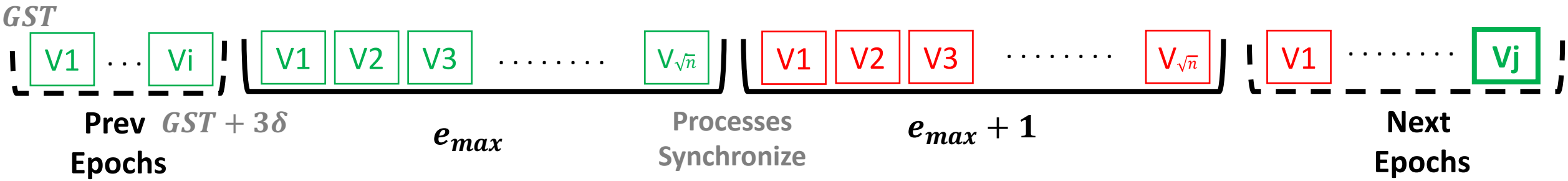
Communication Complexity



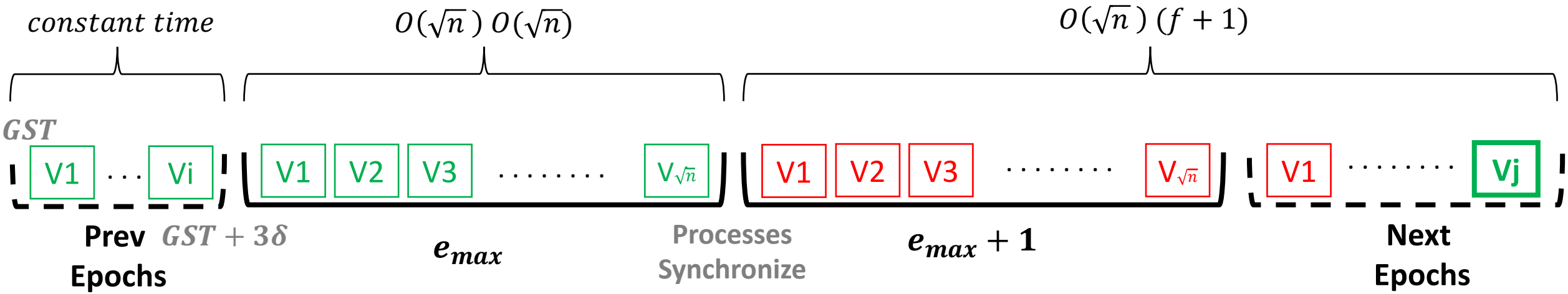
Communication Complexity



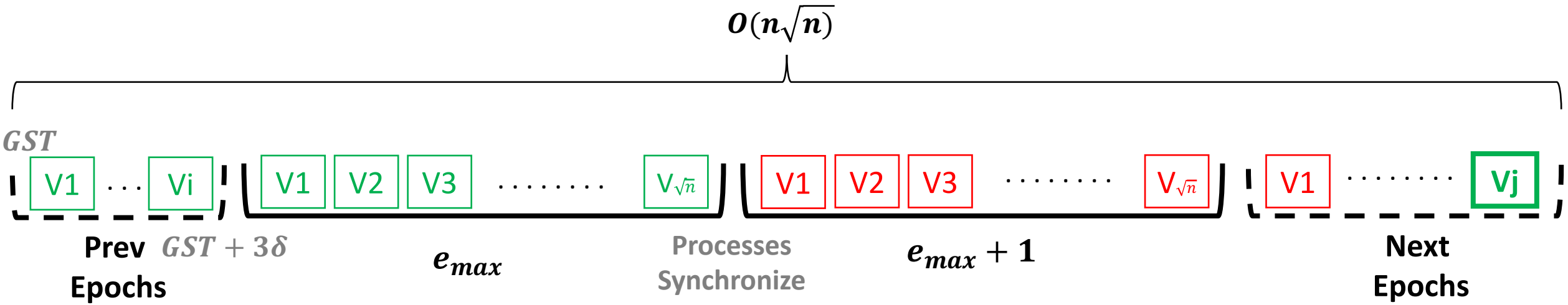
Latency


GST
V1
Vi
V1
V2
V3
V√n
V1
V2
V3
V√n
V1
Vj
Prev Epochs
GST + 3δ
e_max
Processes Synchronize
e_max + 1
Next Epochs

Latency

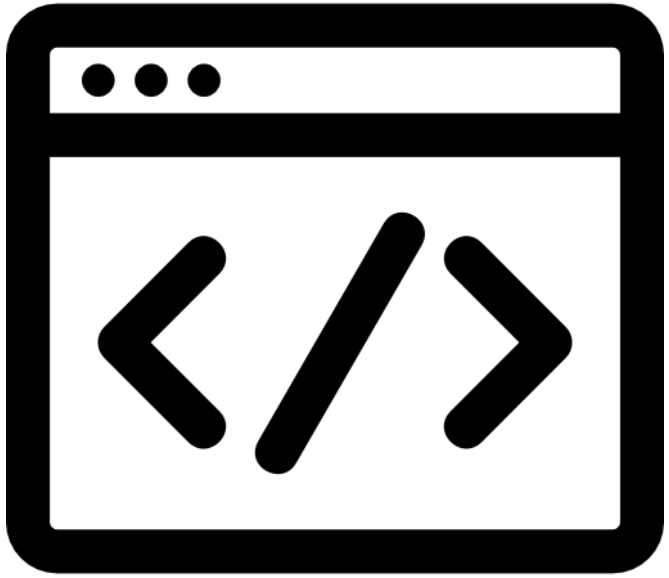

$constant\ time$
$O(\sqrt{n})$ $O(\sqrt{n})$
$O(\sqrt{n})$ $(f+1)$
GST
V1 ... Vi
V1 V2 V3 ..... $V_{\sqrt{n}}$
V1 V2 V3 ..... $V_{\sqrt{n}}$
V1 ..... Vj
Prev Epochs
$GST + 3\delta$
e_{max}
Processes Synchronize
$e_{max} + 1$
Next Epochs

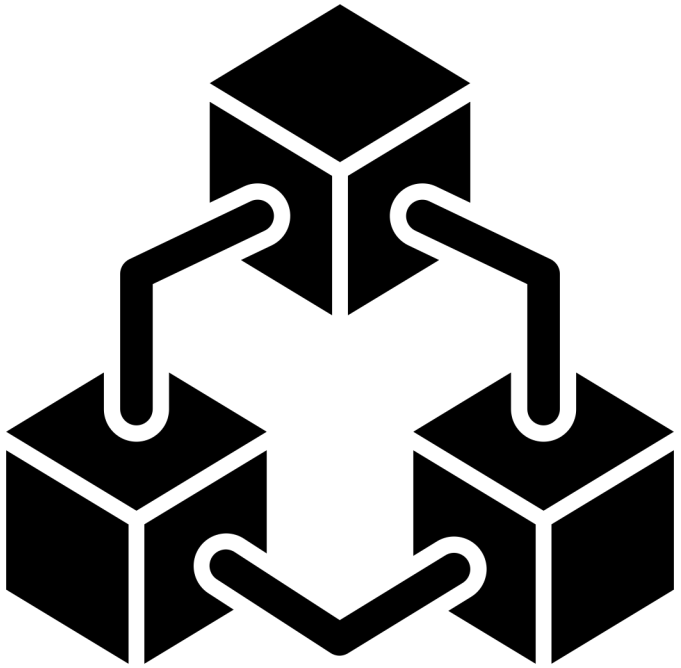
Latency


$O(n\sqrt{n})$
GST
V1
Vi
V1
V2
V3
$V_{\sqrt{n}}$
V1
V2
V3
$V_{\sqrt{n}}$
V1
Vj
Prev Epochs
$GST + 3\delta$
e_{max}
Processes Synchronize
$e_{max} + 1$
Next Epochs

Summary

- Byzantine Vector Consensus is a difficult problem
  - With many real-world applications!
- Previous solutions could not work
  - Either cubic communication complexity or exponential latency
- **Leader-Based Vector Dissemination**
  - Achieves $\boldsymbol{O(n^2\sqrt{n})}$ communication complexity and $\boldsymbol{O(n\sqrt{n})}$ latency!





Thank you!

Questions?