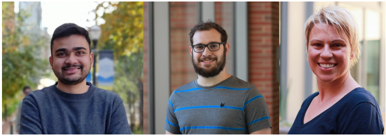
Communication-Efficient LDPC Code Design for Data Availability Oracle in Side Blockchains

Debarnab Mitra, Lev Tauz, and Lara Dolecek

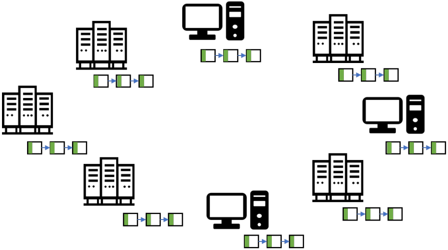
Electrical and Computer Engineering
University of California, Los Angeles

ITW 2021



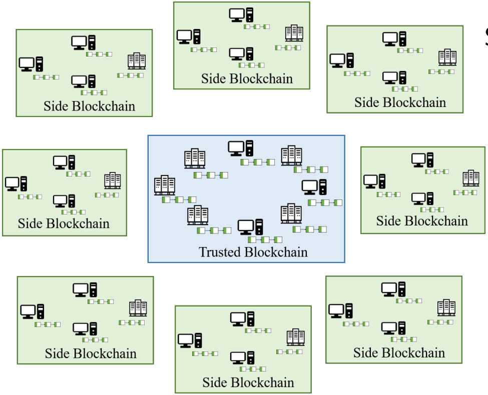
Side Blockchains

- Blockchain systems suffer from low transaction throughput



Side Blockchains

- Blockchain systems suffer from low transaction throughput
- To improve the transaction throughput, they run Side Blockchains

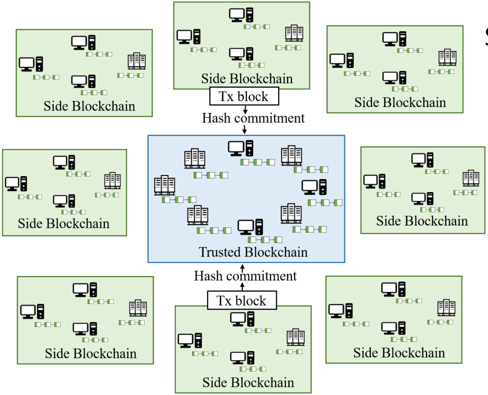


Side Blockchain:-

- Smaller blockchain systems

Side Blockchains

- Blockchain systems suffer from low transaction throughput
- To improve the transaction throughput, they run Side Blockchains



Side Blockchain:-

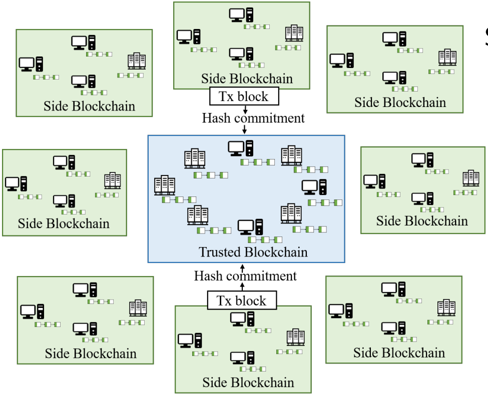
- Smaller blockchain systems

Side Blockchain nodes:

- Push hash commitment of their block to the trusted blockchain

Side Blockchains

- Blockchain systems suffer from low transaction throughput
- To improve the transaction throughput, they run Side Blockchains



Side Blockchain:-

- Smaller blockchain systems

Side Blockchain nodes:

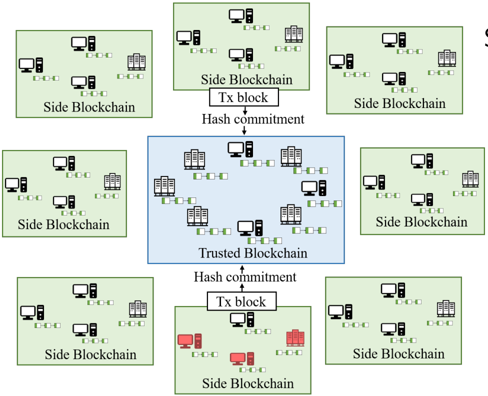
- Push hash commitment of their block to the trusted blockchain

Trusted Blockchain:

- Only store the hash of the side blockchain
- Side blockchains make commitments in parallel

Side Blockchains

- Blockchain systems suffer from low transaction throughput
- To improve the transaction throughput, they run Side Blockchains



Side Blockchain:-

- Smaller blockchain systems

Side Blockchain nodes:

- Push hash commitment of their block to the trusted blockchain

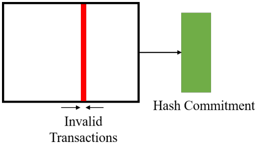
Trusted Blockchain:

- Only store the hash of the side blockchain
- Side blockchains make commitments in parallel

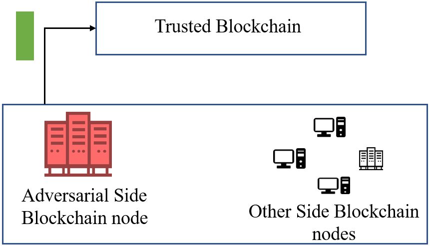
Issue: Side Blockchains with a majority of dishonest nodes are vulnerable to data availability attacks [Sheng '20]

Data Availability (DA) Attack in Side Blockchains

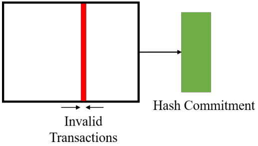
Adversary creates an invalid block



Data Availability (DA) Attack in Side Blockchains



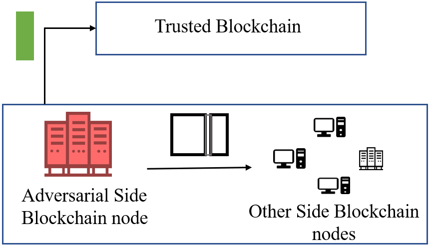
Adversary creates an invalid block



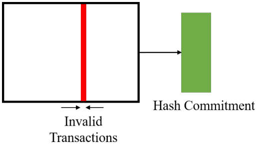
Adversarial Side Blockchain node:

- Pushes hash commitment to the trusted blockchain

Data Availability (DA) Attack in Side Blockchains



Adversary creates an invalid block

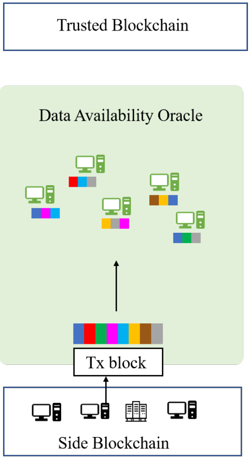


Adversarial Side Blockchain node:

- Pushes hash commitment to the trusted blockchain
- Full block not available to other side blockchain nodes

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

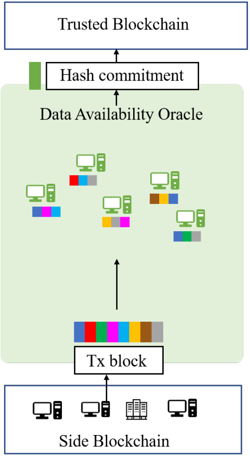


Oracle layer goal

- Collectively and efficiently store chunks of the Tx block (to guarantee availability)

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

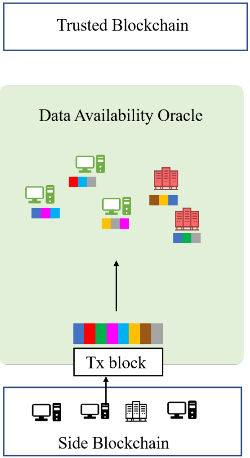


Oracle layer goal

- Collectively and efficiently store chunks of the Tx block (to guarantee availability)
- Push the Tx block's hash commitment iff the block is available

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

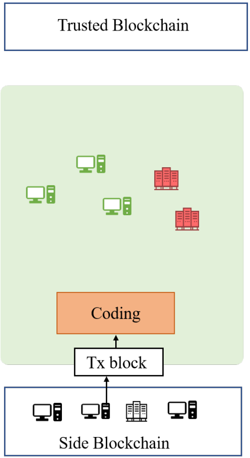


Oracle layer goal

- Collectively and efficiently store chunks of the Tx block (to guarantee availability)
- Push the Tx block's hash commitment iff the block is available
- Oracle nodes can be malicious (honest majority)

Solution using a Data Availability Oracle

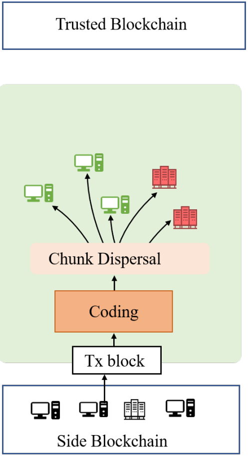
An oracle layer was introduced to ensure data availability [Sheng '20]



- Transaction block: chunked and coded

Solution using a Data Availability Oracle

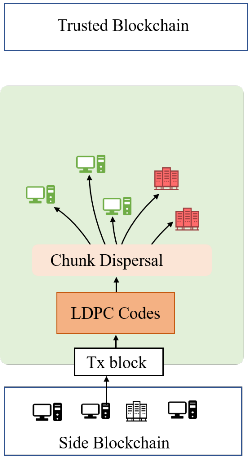
An oracle layer was introduced to ensure data availability [Sheng '20]



- Transaction block: chunked and coded
- Coded chunks dispersed among N oracle nodes

Solution using a Data Availability Oracle

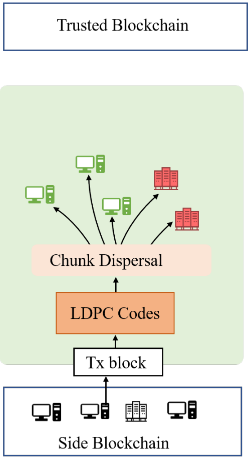
An oracle layer was introduced to ensure data availability [Sheng '20]



- Transaction block: chunked and coded
- Coded chunks dispersed among N oracle nodes
- To improve storage efficiency, LDPC codes are used

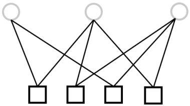
Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



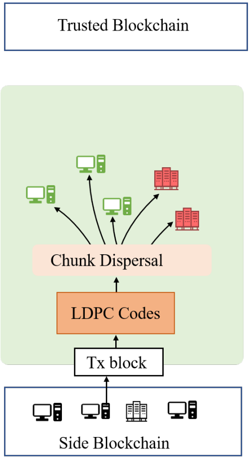
- Transaction block: chunked and coded
- Coded chunks dispersed among N oracle nodes
- To improve storage efficiency, LDPC codes are used

Issues with LDPC codes: small stopping sets (SS)



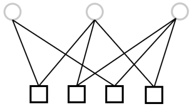
Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



- Transaction block: chunked and coded
- Coded chunks dispersed among N oracle nodes
- To improve storage efficiency, LDPC codes are used

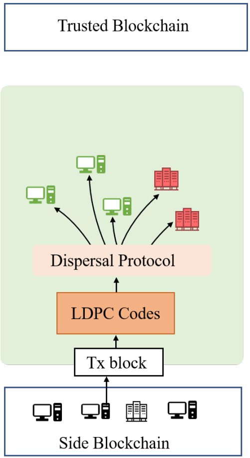
  Issues with LDPC codes: small stopping sets (SS)



- If VNs corresponding to a small stopping set are hidden from the oracle nodes, original block cannot be decoded back by a peeling decoder

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

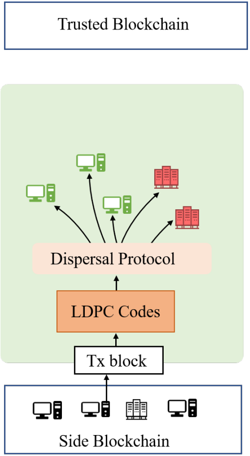


Dispersal Protocol

- Rule about which oracle node stores which coded chunks

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

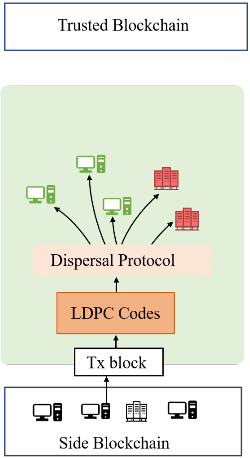


Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



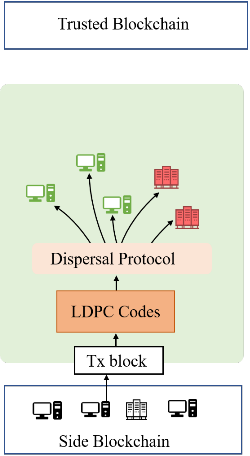
Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur

- $M_{\min} \downarrow$

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]

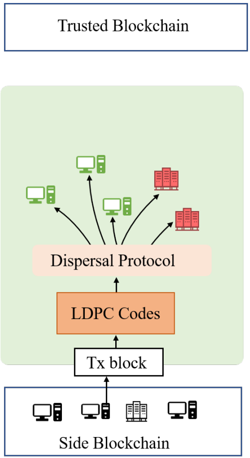


Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur
- $M_{\min} \downarrow \implies$ send more chunks to oracle nodes $\implies$ communication cost $\uparrow$

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



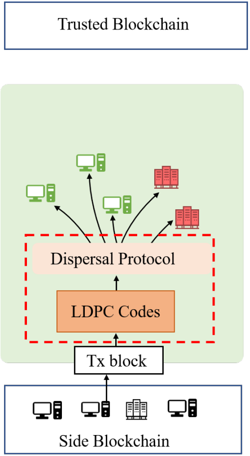
Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur

- $M_{\min} \downarrow \implies$ send more chunks to oracle nodes $\implies$ communication cost $\uparrow$
- Known hard problem to design LDPC codes with large $M_{\min}$ [Jiao '09], [He '11].

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



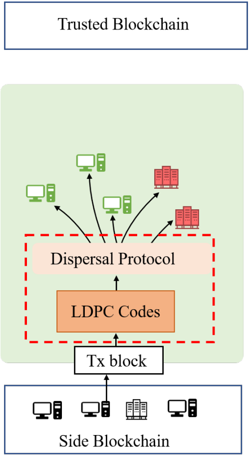
Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur

- $M_{\min} \downarrow \implies$ send more chunks to oracle nodes $\implies$ communication cost $\uparrow$
- Known hard problem to design LDPC codes with large $M_{\min}$ [Jiao '09], [He '11].

Solution using a Data Availability Oracle

An oracle layer was introduced to ensure data availability [Sheng '20]



Dispersal Protocol

- Rule about which oracle node stores which coded chunks
- Sufficient coded chunks sent to each oracle node such that SS failure cannot occur
  - $M_{\min} \downarrow \implies$ send more chunks to oracle nodes $\implies$ communication cost $\uparrow$
  - Known hard problem to design LDPC codes with large $M_{\min}$ [Jiao '09], [He '11].

Our work: Co-design of LDPC codes and a tailored dispersal protocol to significantly lower the communication cost.

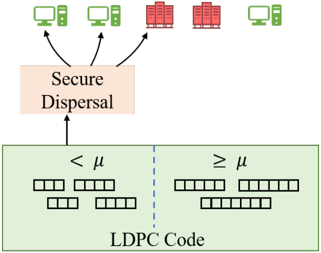
Dispersal Protocol Design

Our dispersal strategy is a two step protocol

Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase
   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)
   - Coded chunks are dispersed such that the small SS failures cannot occur



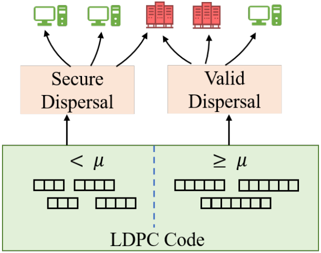
Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase

   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)

   - Coded chunks are dispersed such that the small SS failures cannot occur



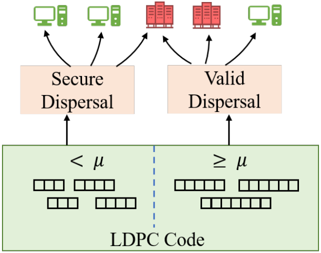
2. Valid Phase

   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase
   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)
   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$

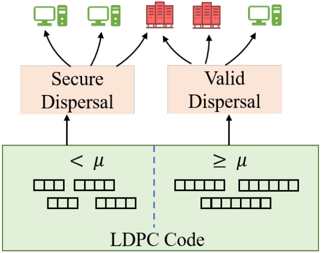


2. Valid Phase
   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase
   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)
   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$ found greedily: $Greedy\text{-}Set(\mathcal{S})$



2. Valid Phase
   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

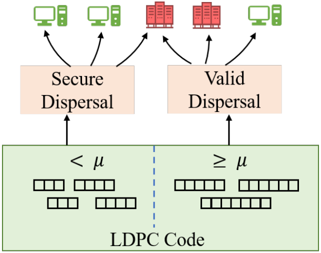
Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase

   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)

   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$ found greedily: $Greedy\text{-}Set(\mathcal{S})$
   - Each VN in $\mathcal{V}$ sent to sufficient nodes such that small SS failures cannot occur



2. Valid Phase

   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

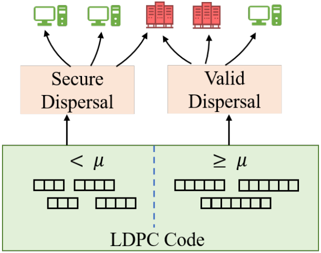
Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase

   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)

   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$ found greedily: $Greedy\text{-}Set(\mathcal{S})$
   - Each VN in $\mathcal{V}$ sent to sufficient nodes such that small SS failures cannot occur
   - Communication cost $\propto |Greedy\text{-}Set(\mathcal{S})|$



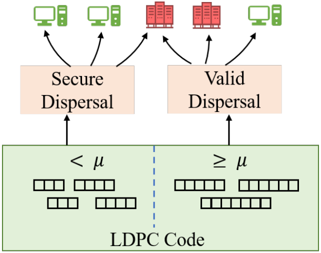
2. Valid Phase

   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase
   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)
   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$ found greedily: $Greedy\text{-}Set(\mathcal{S})$
   - Each VN in $\mathcal{V}$ sent to sufficient nodes such that small SS failures cannot occur
   - Communication cost $\propto |Greedy\text{-}Set(\mathcal{S})|$



2. Valid Phase
   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

> Code Design Strategy:
> Design LDPC codes that have low $|Greedy\text{-}Set(\mathcal{S})|$

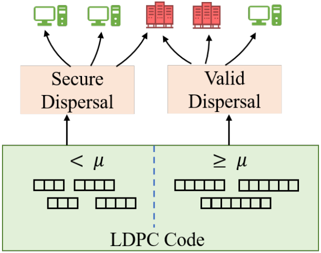
Dispersal Protocol Design

Our dispersal strategy is a two step protocol

1. Secure Phase

   $\mathcal{S}$ = all SSs of size $< \mu$ (small stopping sets)

   - Coded chunks are dispersed such that the small SS failures cannot occur
   - $\mathcal{V}$: set of VNs that *cover* all SSs in $\mathcal{S}$ found greedily: $Greedy\text{-}Set(\mathcal{S})$
   - Each VN in $\mathcal{V}$ sent to sufficient nodes such that small SS failures cannot occur
   - Communication cost $\propto |Greedy\text{-}Set(\mathcal{S})|$



2. Valid Phase

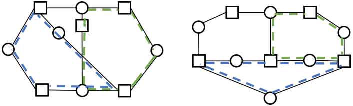
   - Coded chunks are dispersed such that large (size $\geq \mu$) SS failures cannot occur

Code Design Strategy:
Design LDPC codes that have low $|Greedy\text{-}Set(\mathcal{S})|$

-Modify the PEG algorithm [Xiao '05]

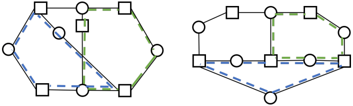
Dispersal-Efficient (DE)-PEG Algorithm

- SSs are made up of cycles [Tian '03]



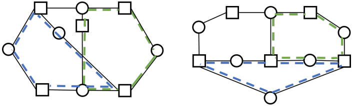
Dispersal-Efficient (DE)-PEG Algorithm

- SSs are made up of cycles [Tian '03]
- Want to design LDPC codes with low $|Greedy\text{-}Set(\mathcal{S})|$, $\mathcal{S} =$ all SSs of size $< \mu$



Dispersal-Efficient (DE)-PEG Algorithm

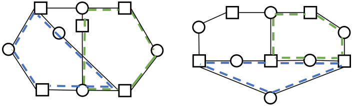
- SSs are made up of cycles [Tian '03]
- Want to design LDPC codes with low $|Greedy\text{-}Set(\mathcal{S})|$, $\mathcal{S} =$ all SSs of size $< \mu$



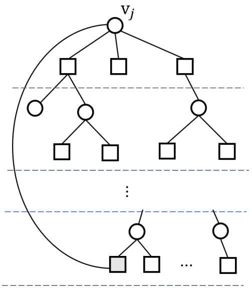
- Design LDPC codes to reduce $|Greedy\text{-}Set(\mathcal{L})|$, $\mathcal{L} =$ List of cycles of length $\leq g$

Dispersal-Efficient (DE)-PEG Algorithm

- SSs are made up of cycles [Tian '03]
- Want to design LDPC codes with low $|\mathit{Greedy\text{-}Set}(\mathcal{S})|$, $\mathcal{S}$ = all SSs of size $< \mu$



- Design LDPC codes to reduce $|\mathit{Greedy\text{-}Set}(\mathcal{L})|$, $\mathcal{L}$ = List of cycles of length $\leq g$



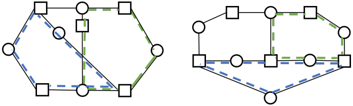
All CNs exhausted

DE-PEG Algorithm
For each VN v_j
Expand Tanner Graph in a BFS fashion
If $\exists$ CNs not connected to v_j
- Select a CN with min degree not connected to v_j

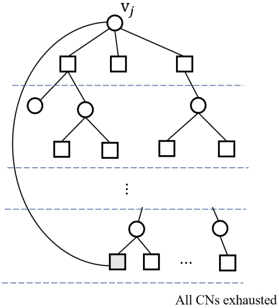
Else *(new cycles created)*
- Find CNs most distant to v_j
- Select CNs with minimum degree

Dispersal-Efficient (DE)-PEG Algorithm

- SSs are made up of cycles [Tian '03]
- Want to design LDPC codes with low $|Greedy\text{-}Set(\mathcal{S})|$, $\mathcal{S}$ = all SSs of size $< \mu$



- Design LDPC codes to reduce $|Greedy\text{-}Set(\mathcal{L})|$, $\mathcal{L}$ = List of cycles of length $\leq g$



DE-PEG Algorithm

For each VN v_j

Expand Tanner Graph in a BFS fashion

If $\exists$ CNs not connected to v_j

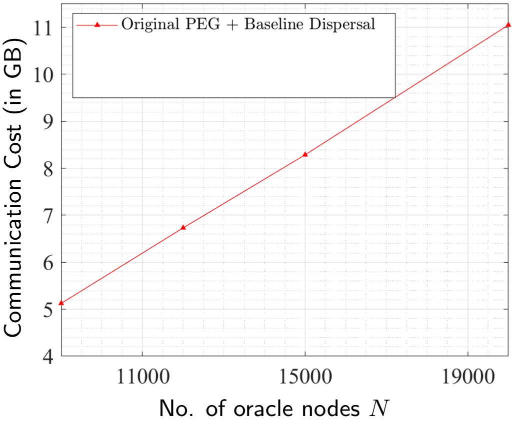
- Select a CN with min degree not connected to v_j

Else *(new cycles created)*

- Find CNs most distant to v_j
- Select CNs with minimum degree
- Select one with minimum $|Greedy\text{-}Set(\mathcal{L})|$

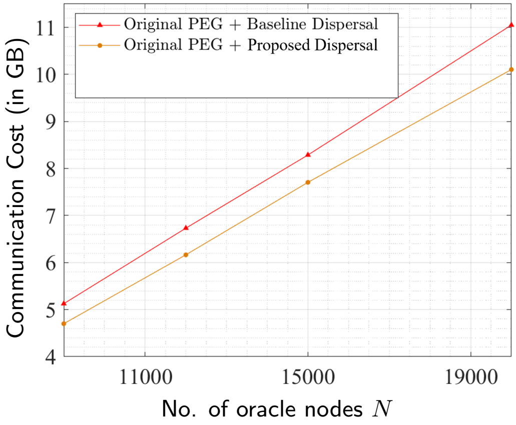
Simulation Results

Communication Cost achieved by different coding schemes and dispersal strategies



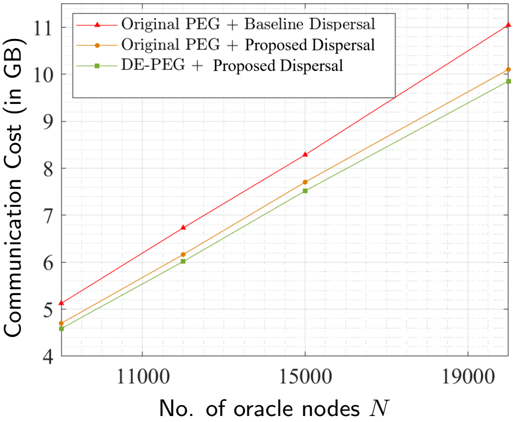
Simulation Results

Communication Cost achieved by different coding schemes and dispersal strategies



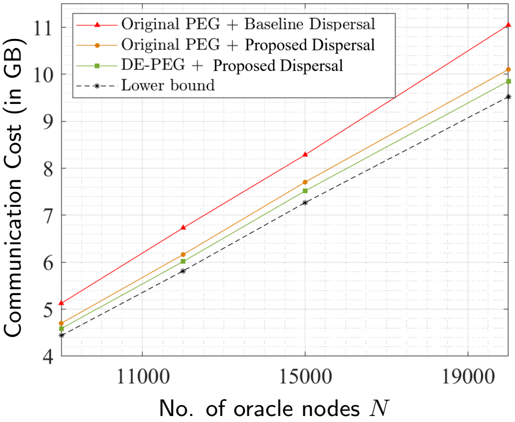
Simulation Results

Communication Cost achieved by different coding schemes and dispersal strategies



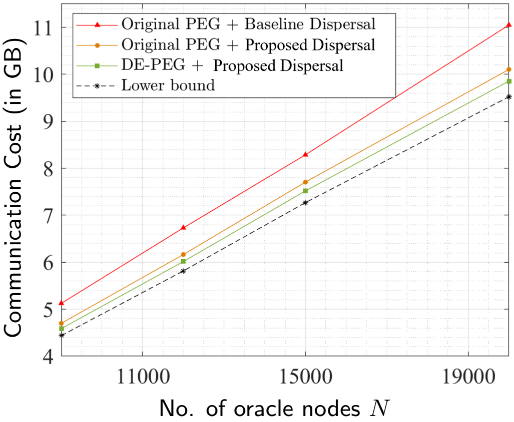
Simulation Results

Communication Cost achieved by different coding schemes and dispersal strategies



Simulation Results

Communication Cost achieved by different coding schemes and dispersal strategies



Co-design of the DE-PEG algorithm and the proposed dispersal protocol reduce the communication cost

References

- (Sheng '20) P. Sheng, et al., *"ACeD: Scalable Data Availability Oracle"* arXiv preprint arXiv:2011.00102, Oct. 2020.
- (Xiao '05) X.Y. Hu, et al., *"Regular and irregular progressive edge-growth tanner graphs,"* IEEE Transactions of Information Theory, vol. 51, no. 1, 2005.
- (Tian '03) T. Tian, et al., *"Construction of irregular LDPC codes with low error floors,"* IEEE International Conference on Communications, May 2003.
- (Jiao '09) X. Jiao, et al. *"Eliminating small stopping sets in irregular low-density parity-check codes,"* IEEE Communications Letters, vol. 13, no. 6, Jun. 2009.
- (He '11) Y. He, et al. *"A survey of error floor of LDPC codes,"* International ICST Conference on Communications and Networking in China (CHINACOM), Aug. 2011.