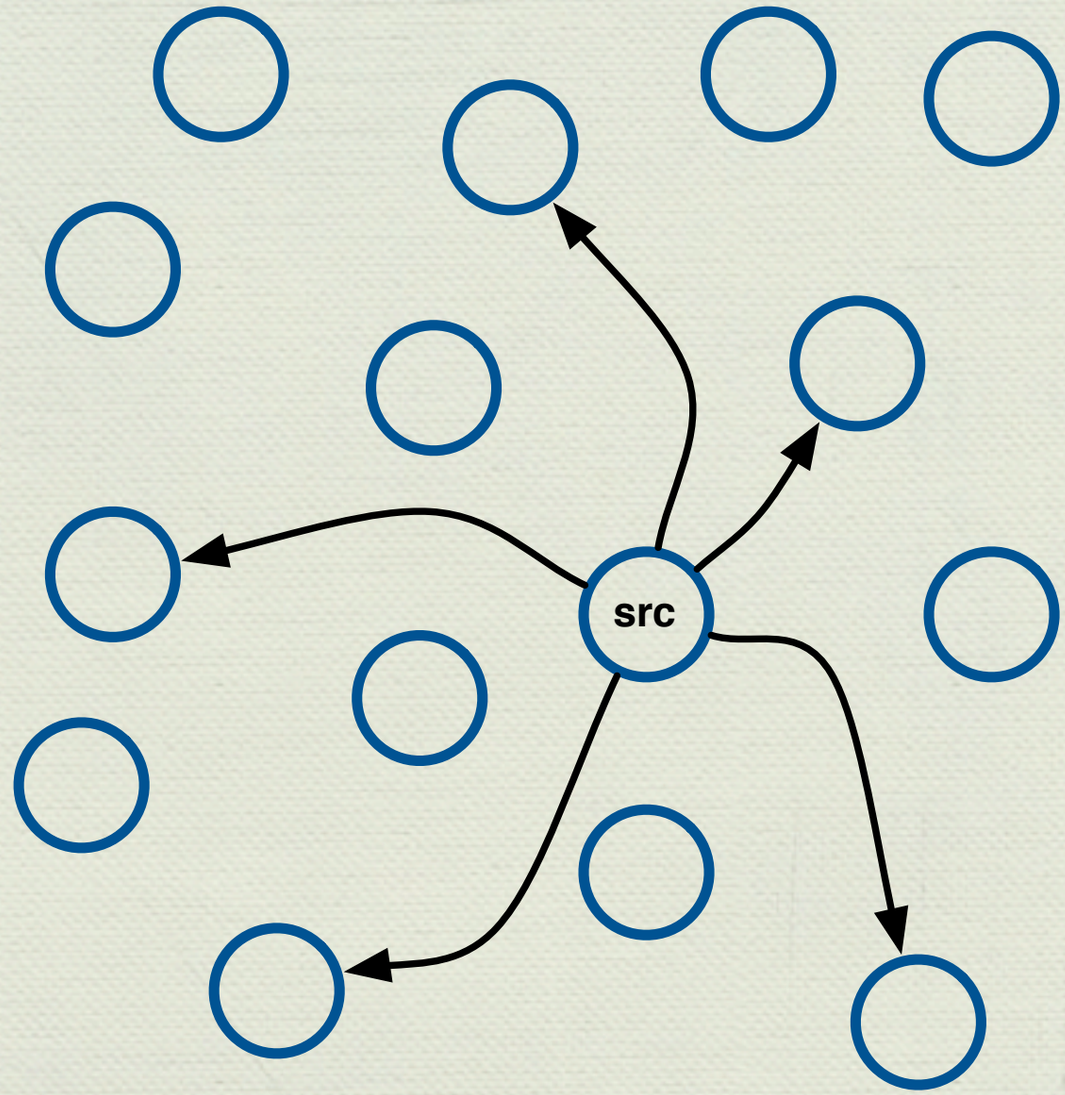
Pipelined Regeneration with Regenerating Codes for Distributed Storage Systems

Jun Li[1], Xin Wang[1], Baochun Li[2]
[1]Fudan University, China
[2]University of Toronto, Canada

Distributed Storage

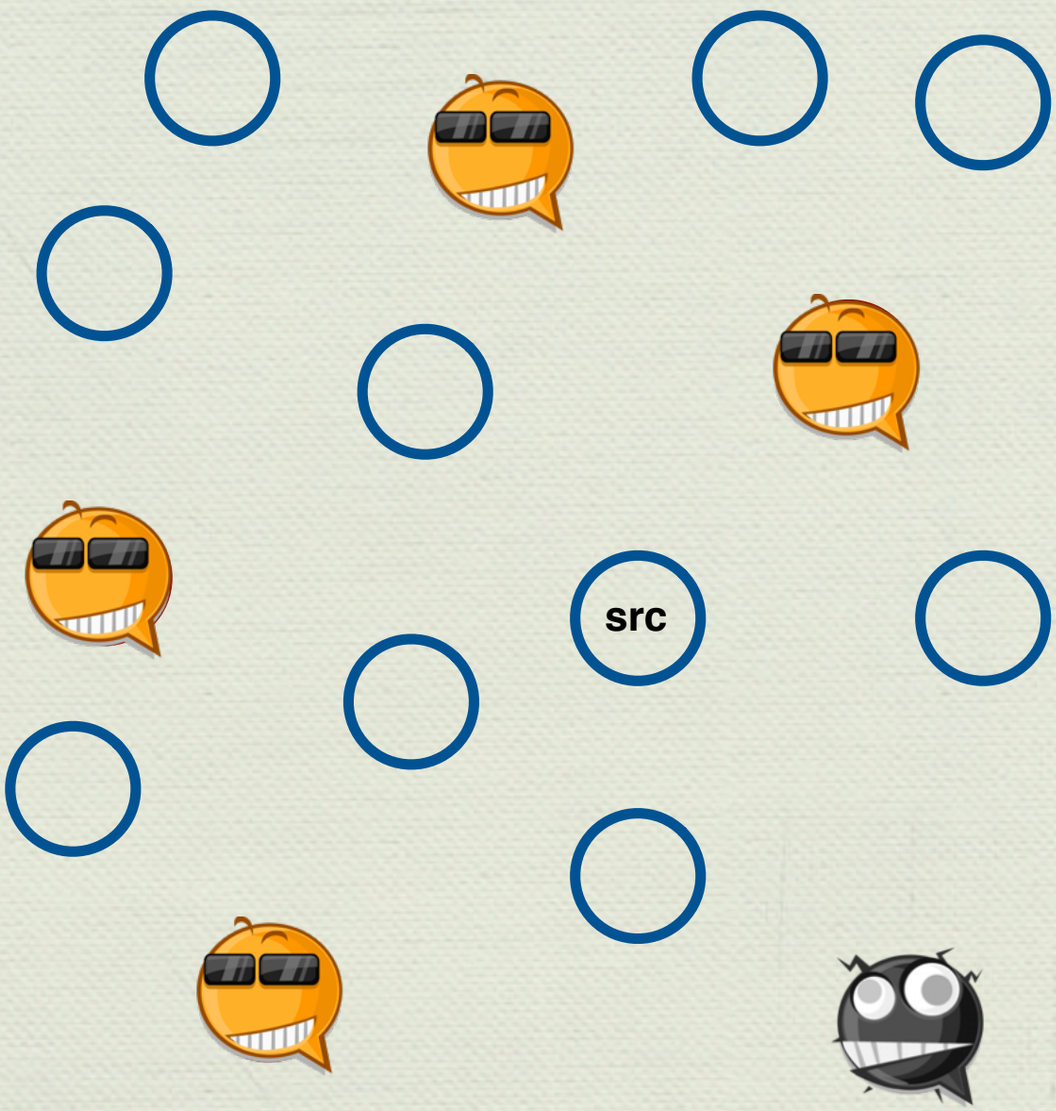
Distributed Storage

- Store data in a large number of storage nodes



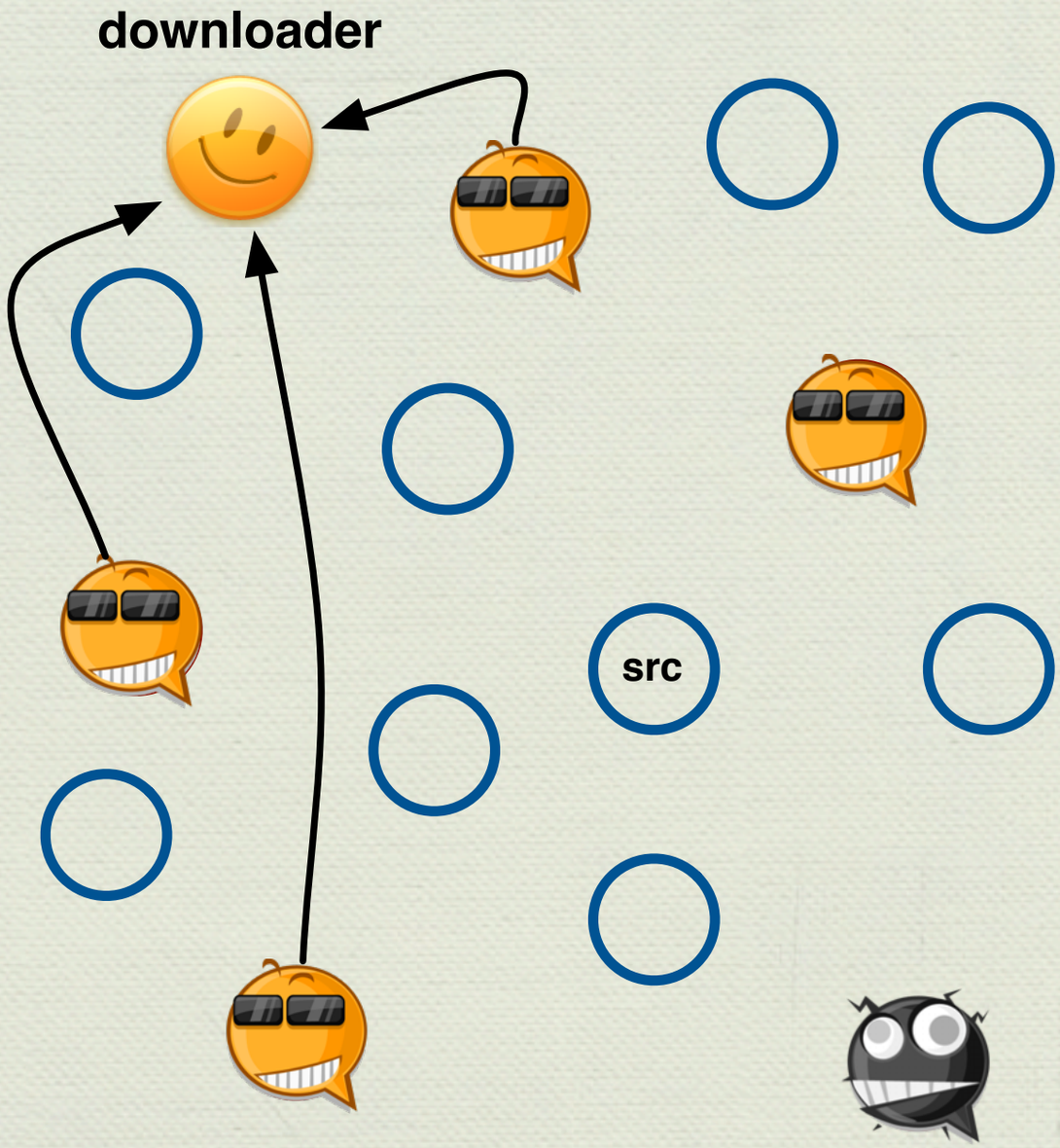
Distributed Storage

- Store data in a large number of storage nodes
- Handle node failures as the rule, rather than exceptions



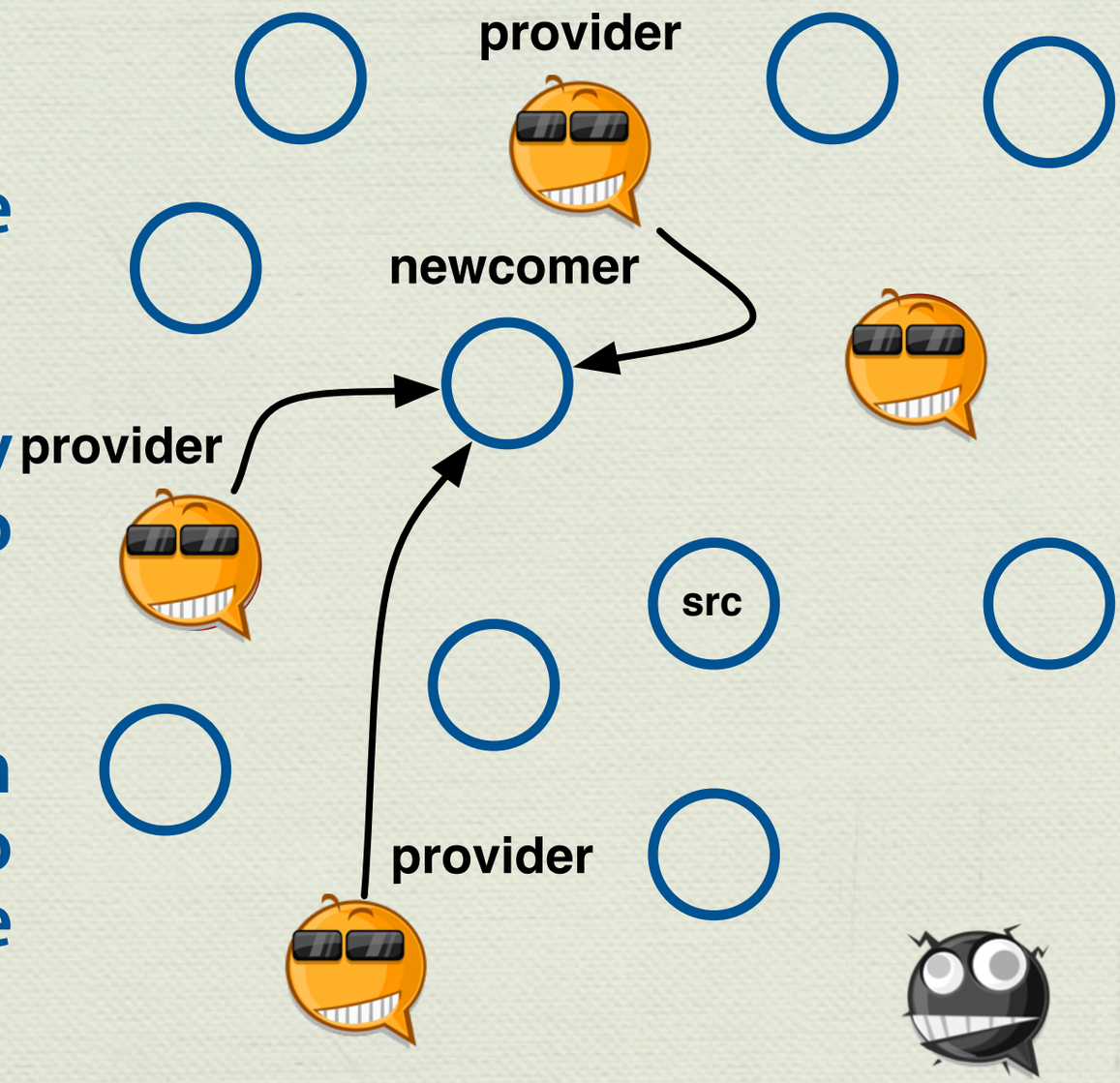
Distributed Storage

- Store data in a large number of storage nodes
- Handle node failures as the rule, rather than exceptions
- MDS codes guarantee that any k storage nodes suffice to recover the original data



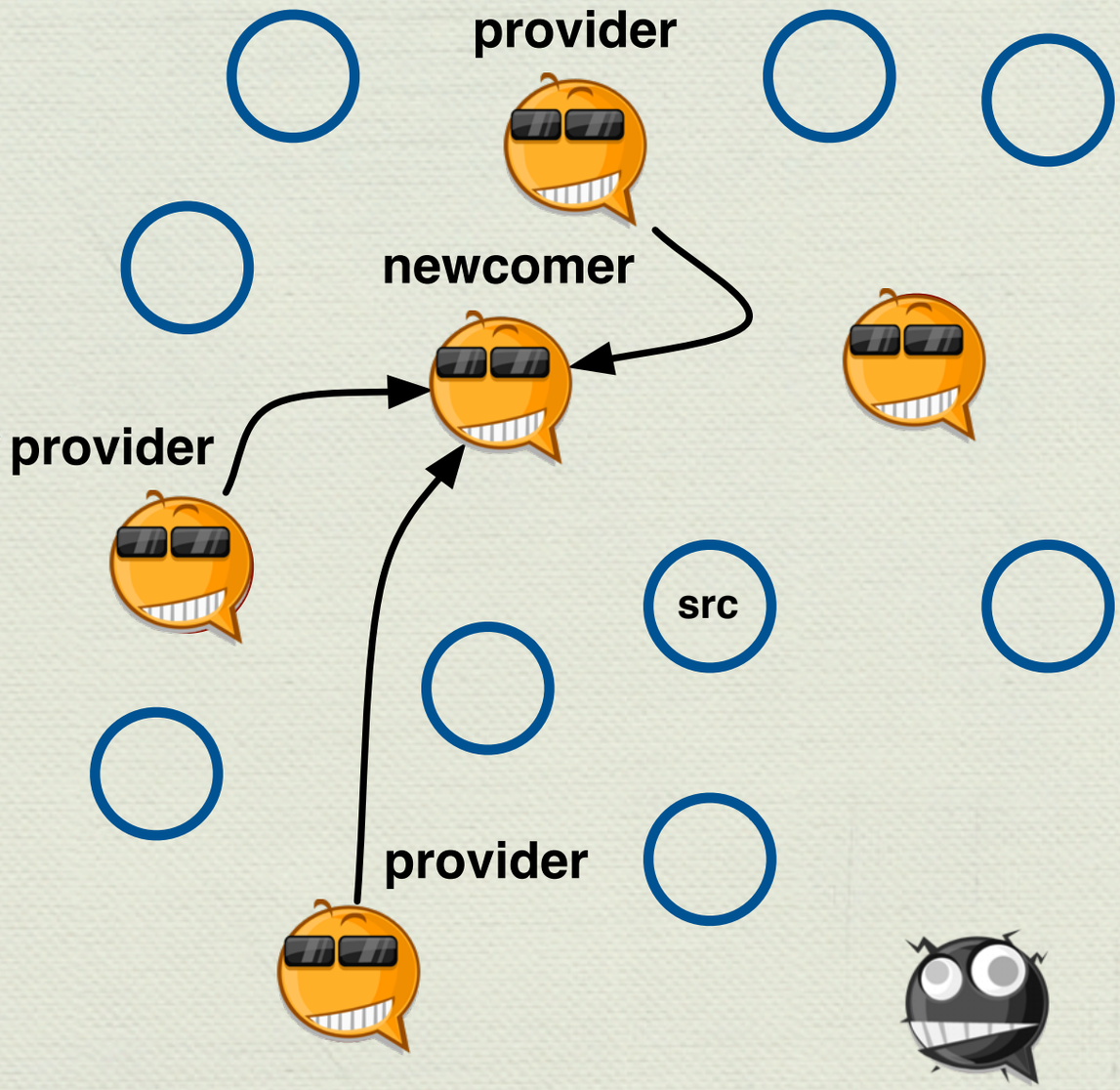
Distributed Storage

- Store data in a large number of storage nodes
- Handle node failures as the rule, rather than exceptions
- MDS codes guarantee that any k storage nodes suffice to recover the original data
- MDS codes require for data in at least k nodes (providers) to regenerate data in one node (newcomer)



Distributed Storage

- Store data in a large number of storage nodes
- Handle node failures as the rule, rather than exceptions
- MDS codes guarantee that any k storage nodes suffice to recover the original data
- MDS codes require for data in at least k nodes (providers) to regenerate data in one node (newcomer)



Regenerating Codes

Regenerating Codes

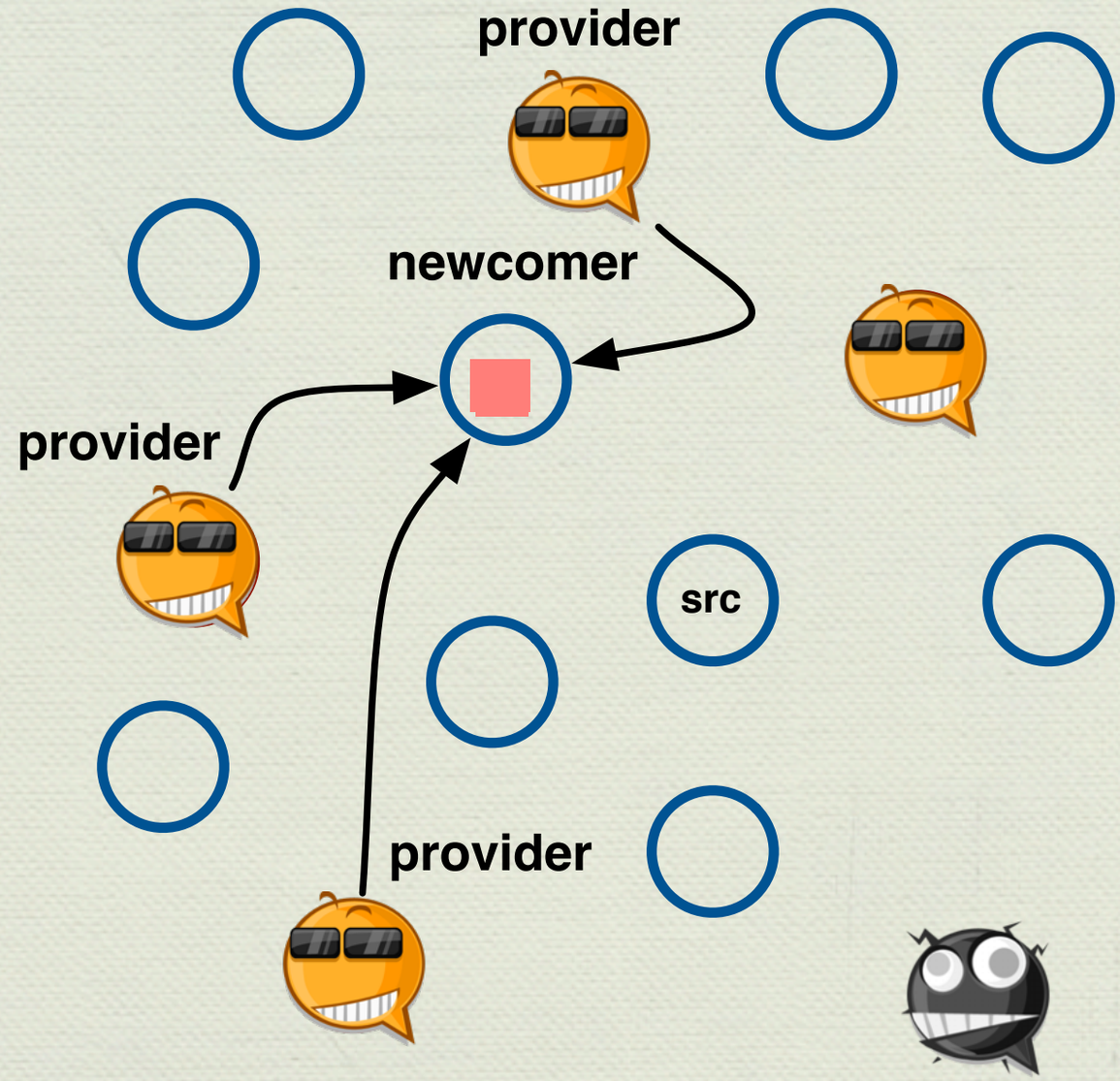
- A storage node stores a coded block with a size of M/k bits

Regenerating Codes

- A storage node stores a coded block with a size of M/k bits
- Bandwidth consumption to regenerate one coded block:

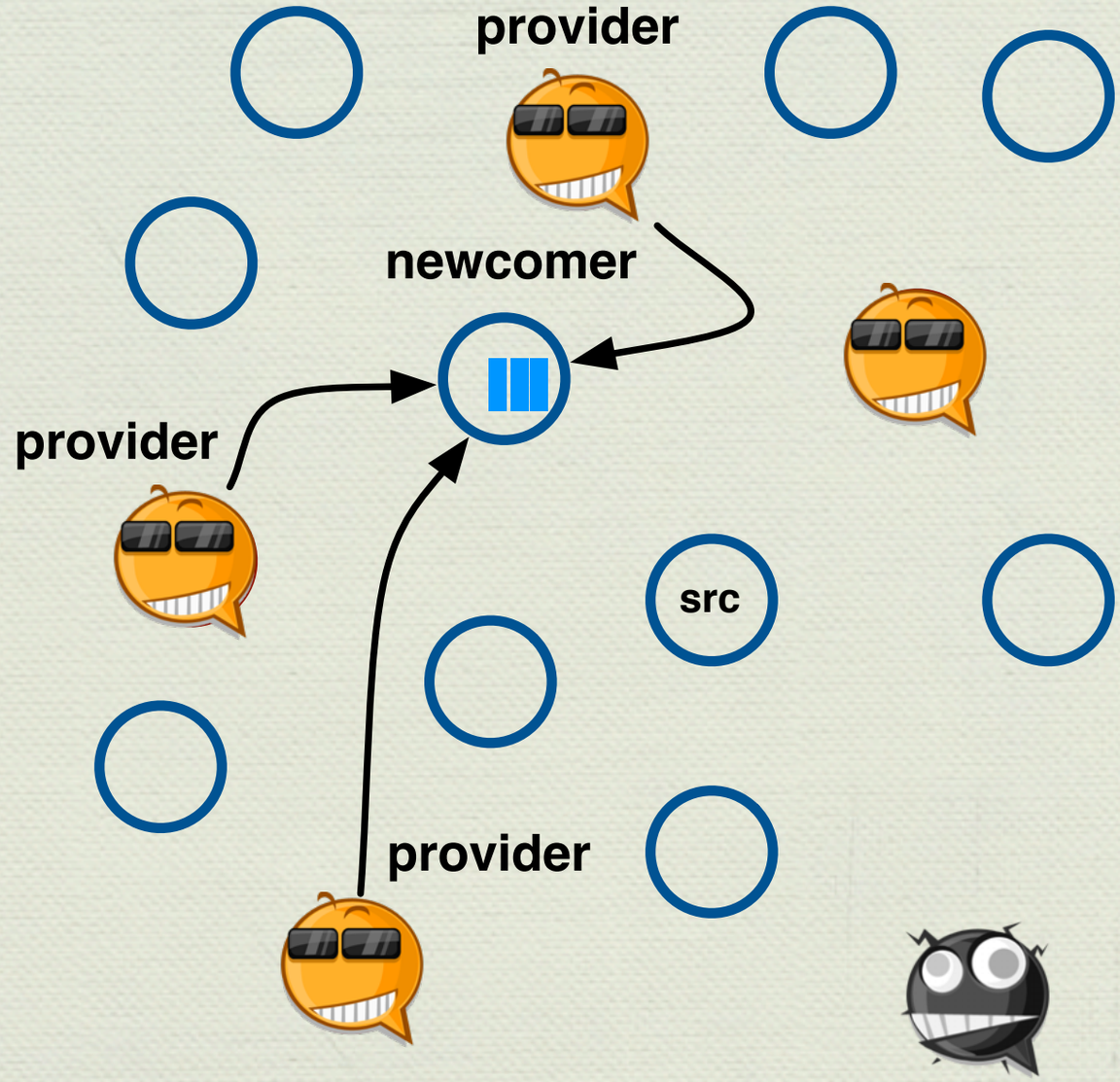
Regenerating Codes

- A storage node stores a coded block with a size of M/k bits
- Bandwidth consumption to regenerate one coded block:
  - MDS codes: M bits when accessing data from k providers



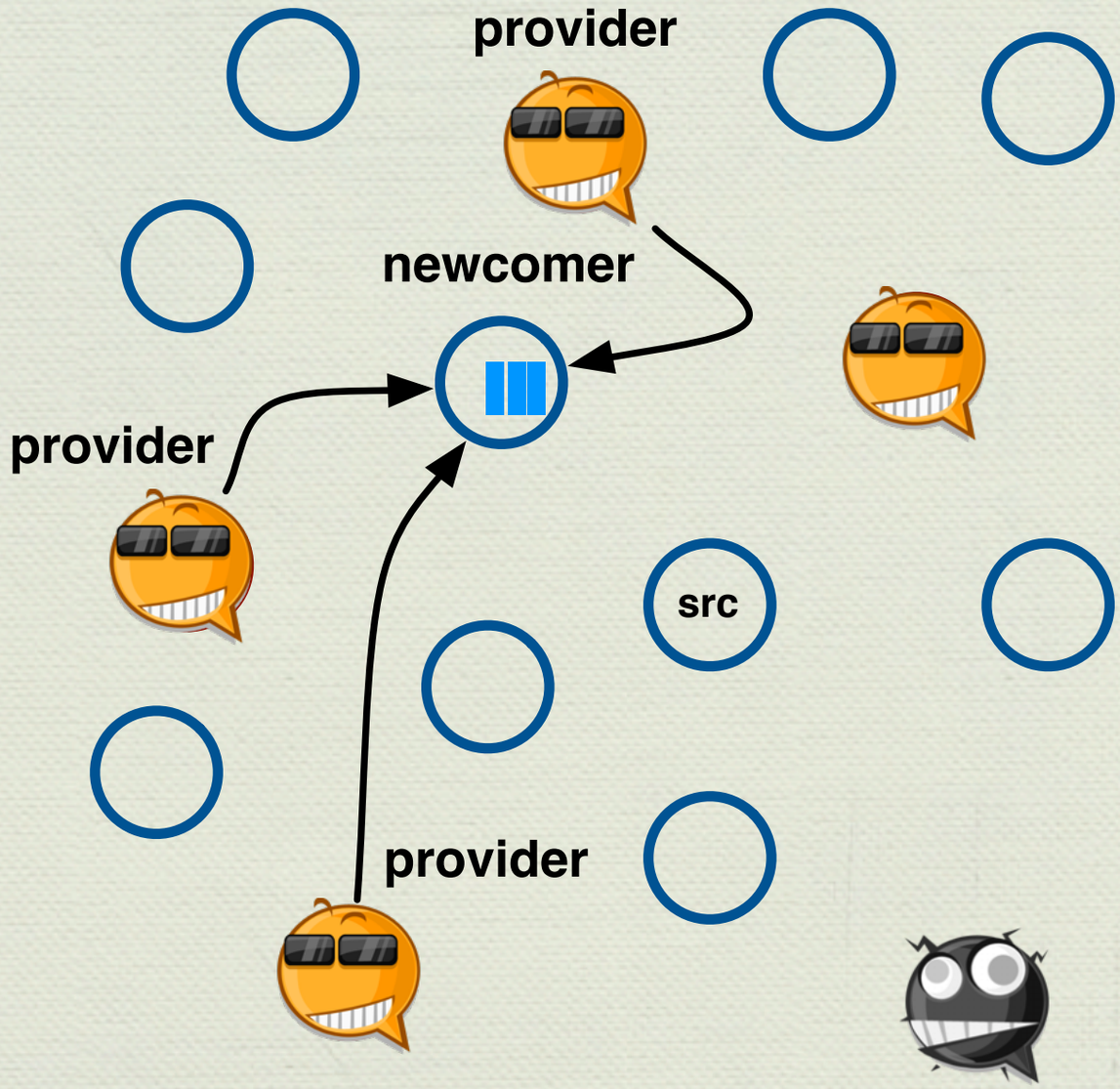
Regenerating Codes

- A storage node stores a coded block with a size of M/k bits
- Bandwidth consumption to regenerate one coded block:
  - MDS codes: M bits when accessing data from k providers
  - Minimum-Storage Regenerating (MSR) Codes: (M(r-1))/(k(r-k)) bits when accessing data from r-1 providers



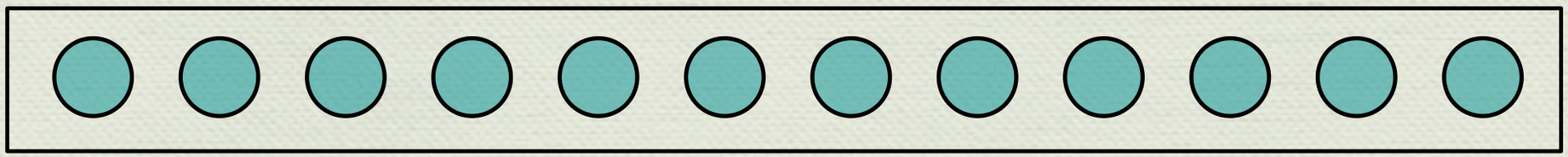
Regenerating Codes

- A storage node stores a coded block with a size of M/k bits
- Bandwidth consumption to regenerate one coded block:
  - MDS codes: M bits when accessing data from k providers
  - Minimum-Storage Regenerating (MSR) Codes: (M(r-1))/(k(r-k)) bits when accessing data from r-1 providers
- **Is it possible to achieve the same performance with much fewer participating nodes?**



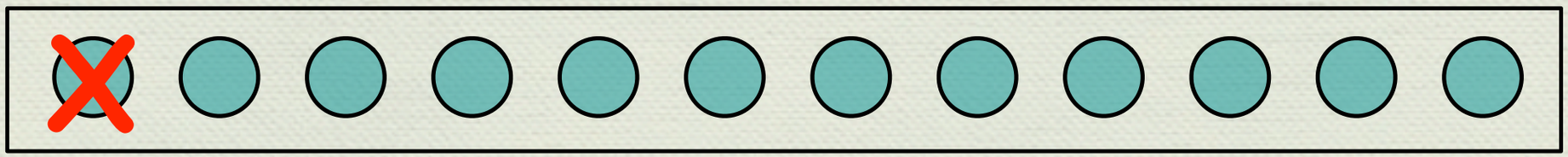
Our Solution:
Pipelined Regeneration

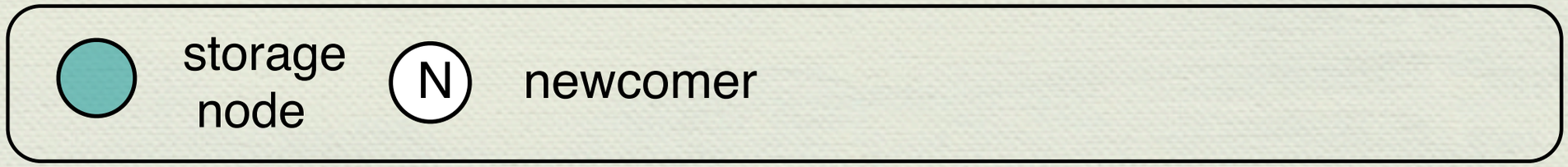

storage
node



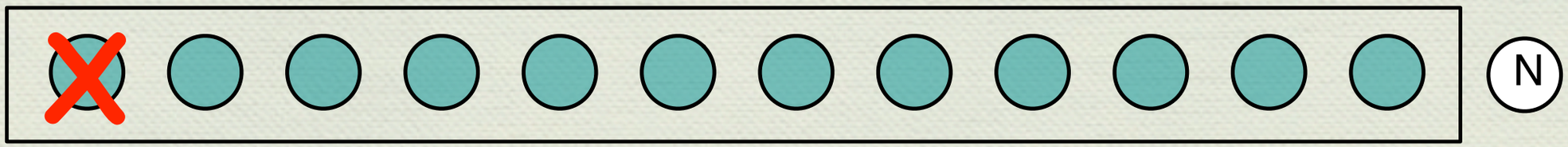


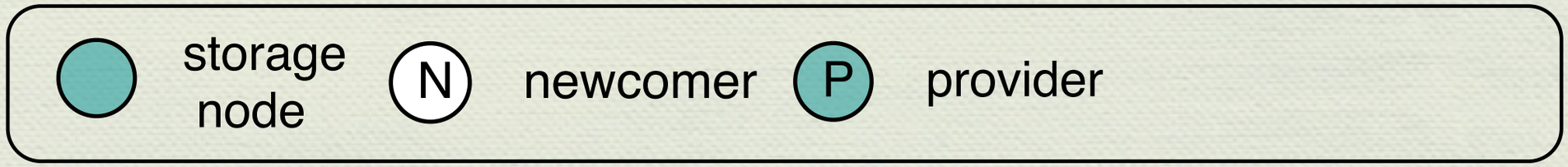
Conventional Regeneration



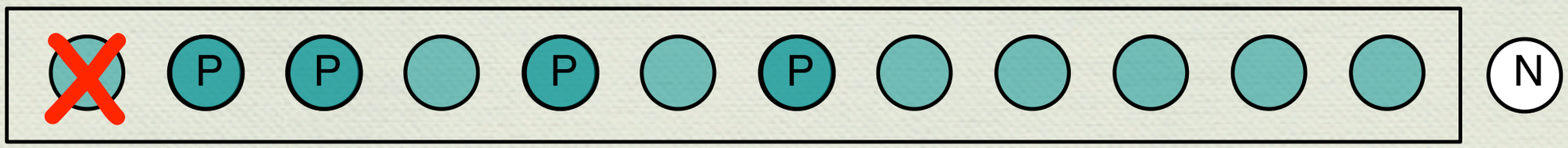


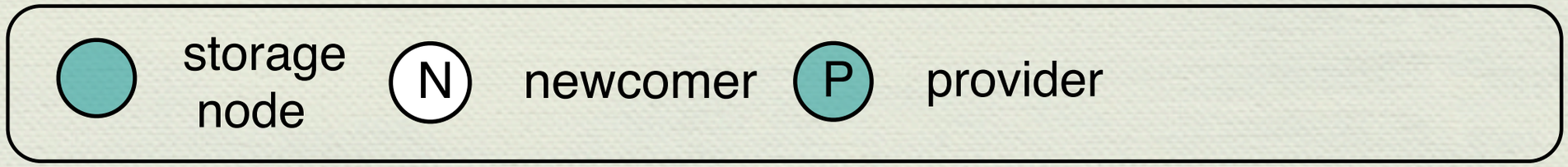
Conventional Regeneration



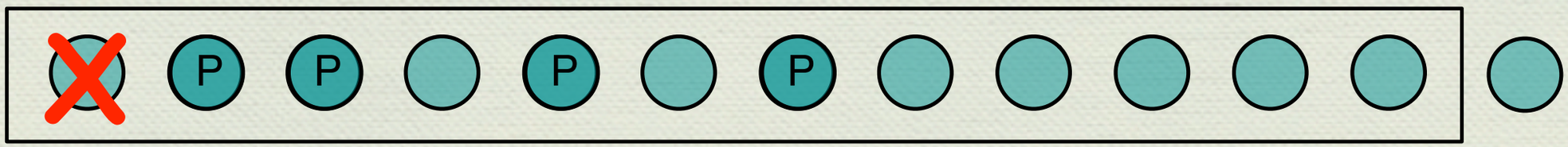


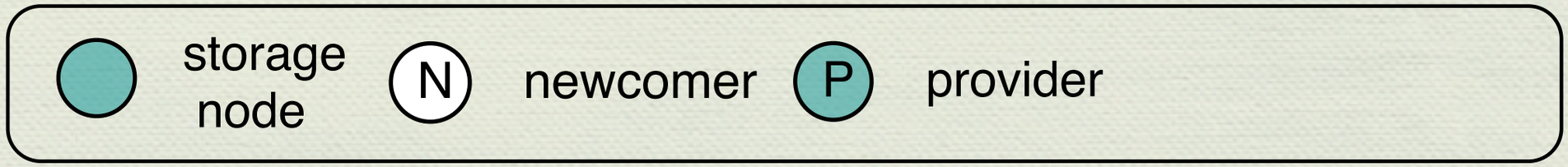
Conventional Regeneration



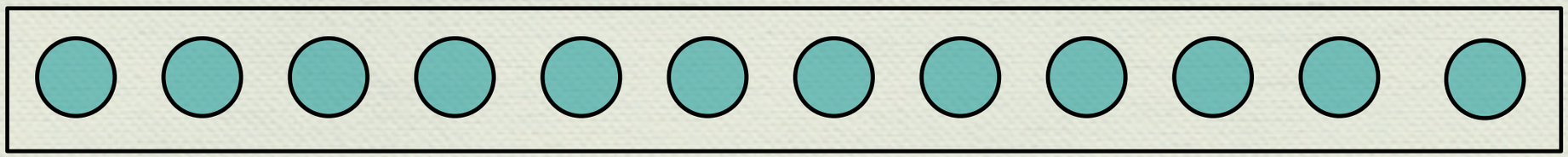


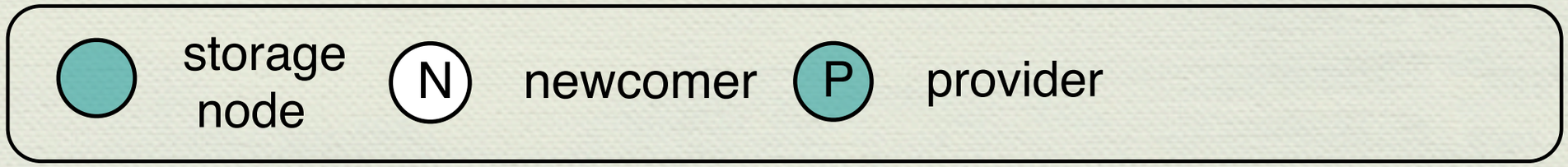
Conventional Regeneration



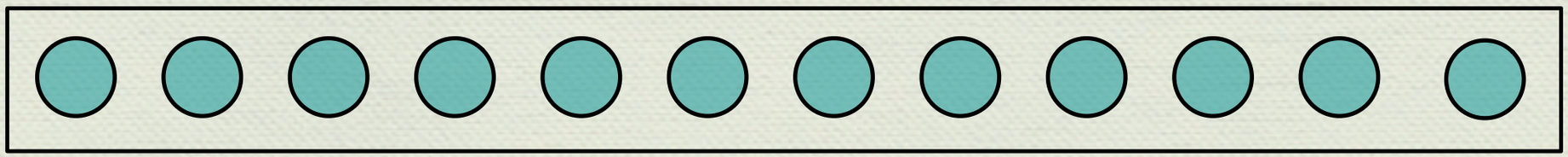


Conventional Regeneration

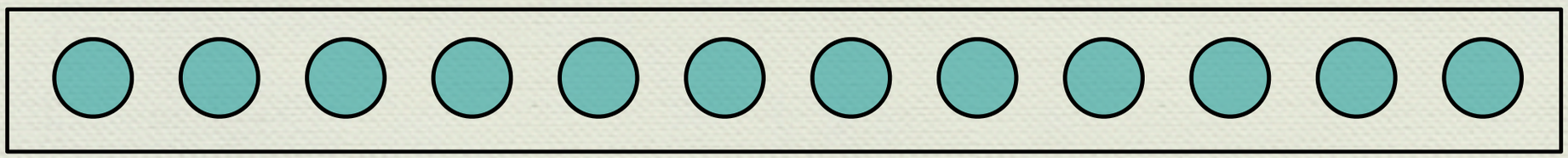


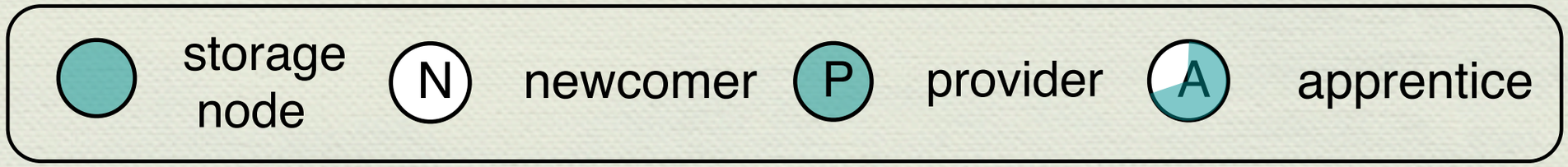


Conventional Regeneration

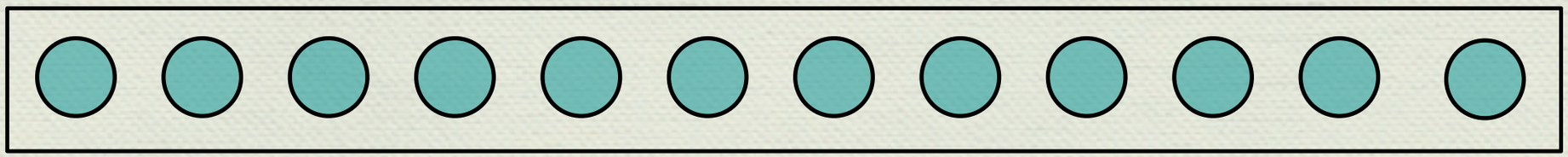


Pipelined Regeneration

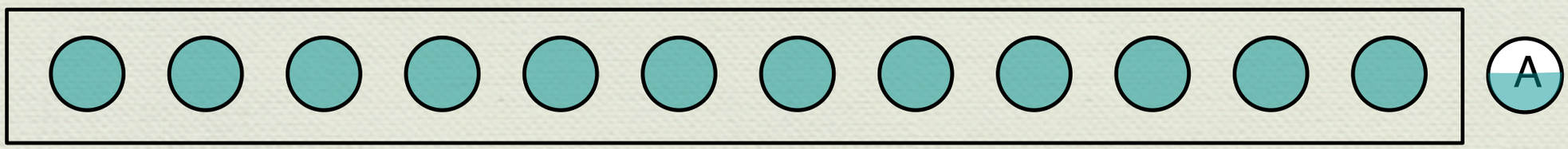


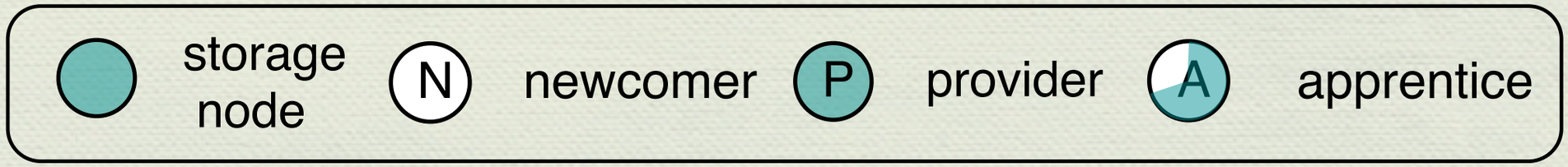


Conventional Regeneration

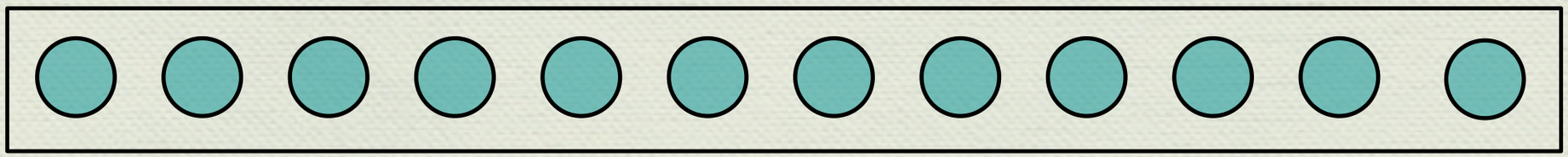


Pipelined Regeneration

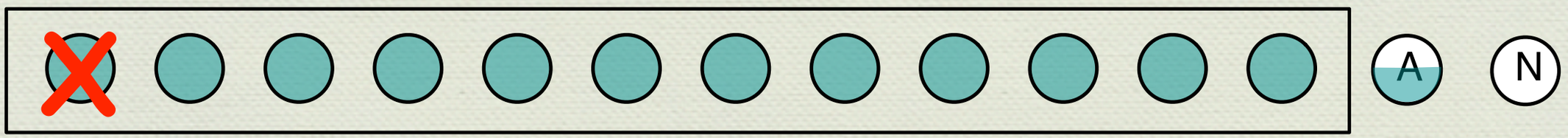


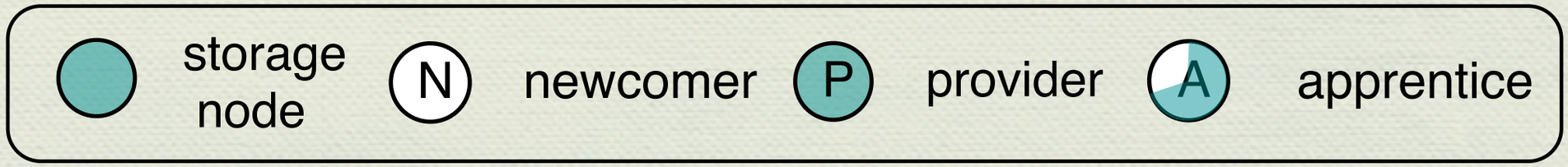


Conventional Regeneration

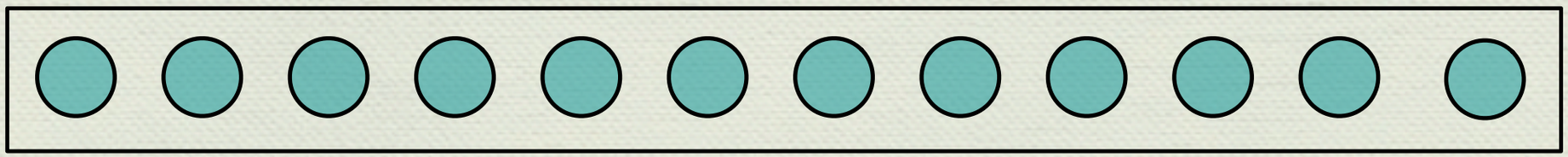


Pipelined Regeneration

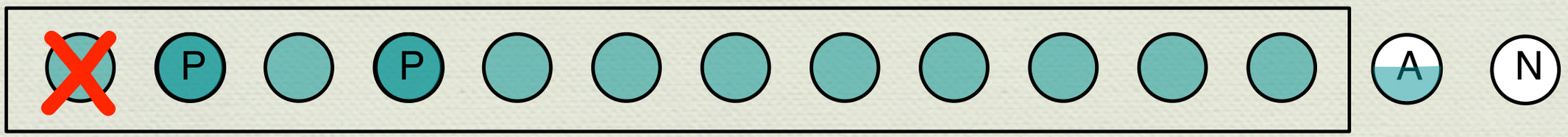


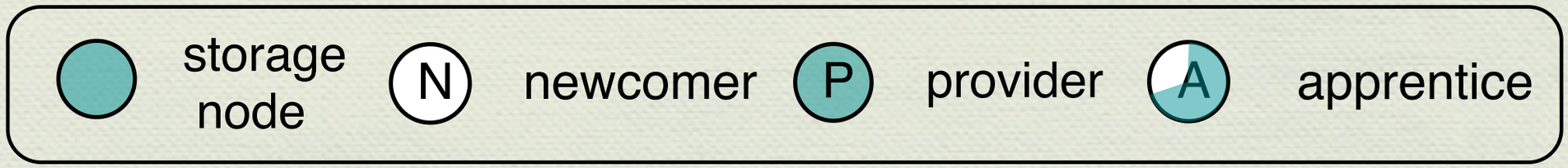


Conventional Regeneration

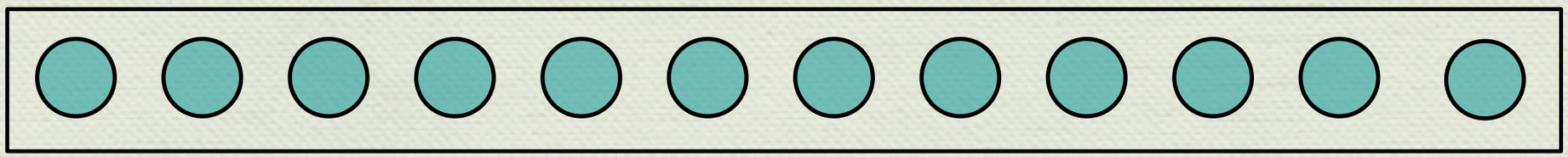


Pipelined Regeneration

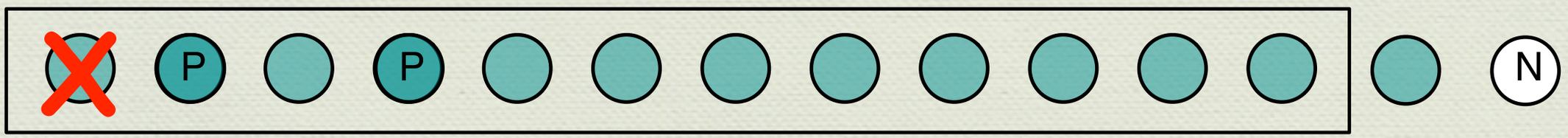


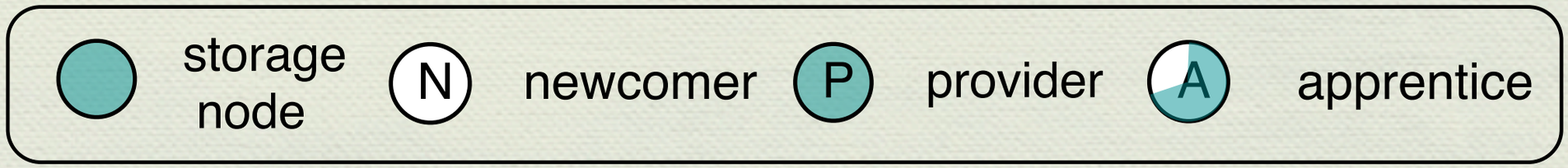


Conventional Regeneration

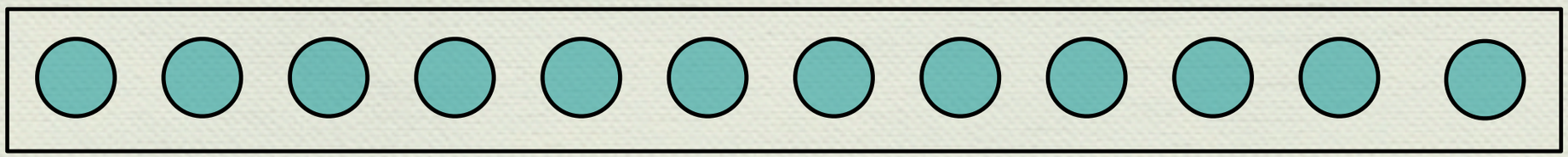


Pipelined Regeneration

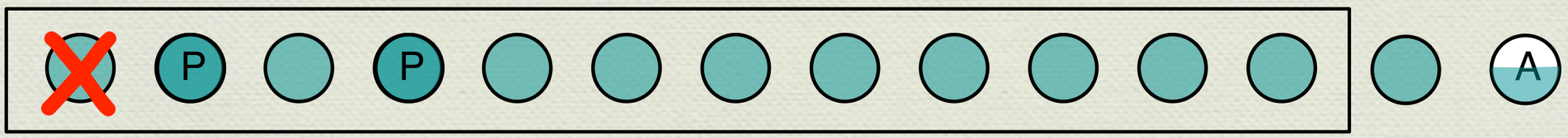


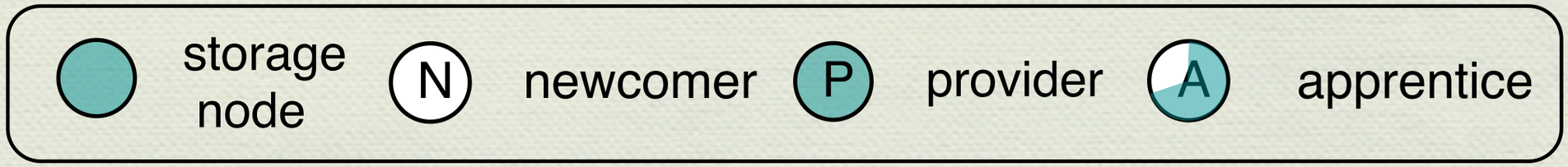


Conventional Regeneration

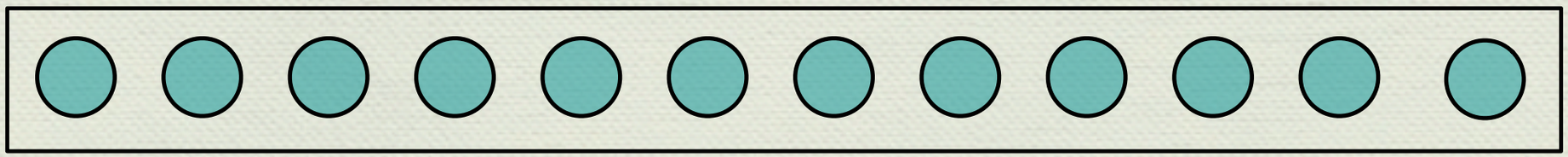


Pipelined Regeneration

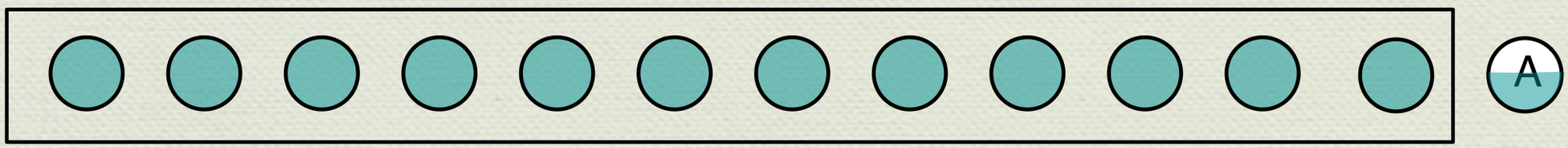


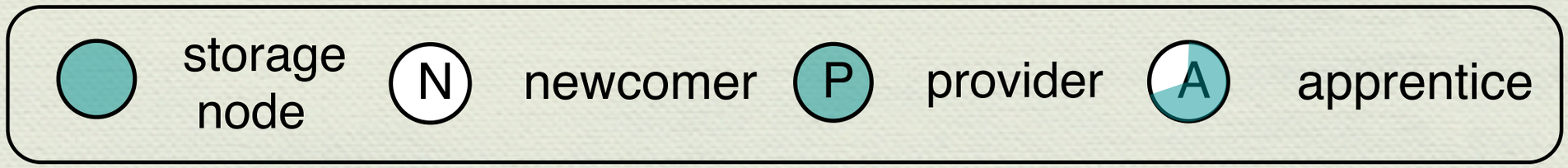


Conventional Regeneration

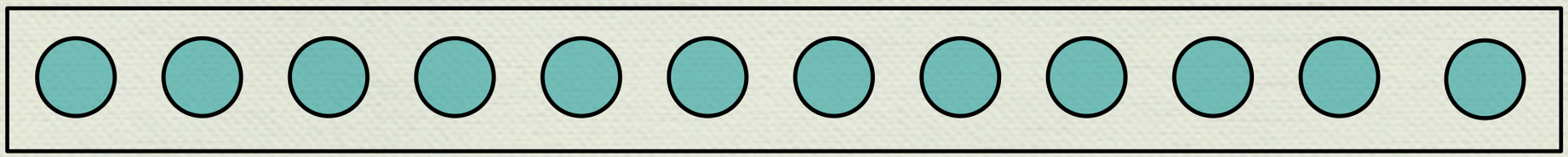


Pipelined Regeneration

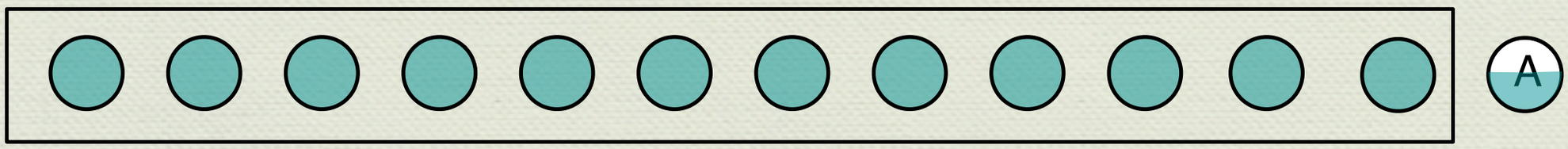




Conventional Regeneration



Pipelined Regeneration



Open Questions:

How to encode data to support pipelined regeneration?
How to deliver data during pipelined regeneration?
What is the relationship between #apprentices and #providers?

Encoding

Encoding

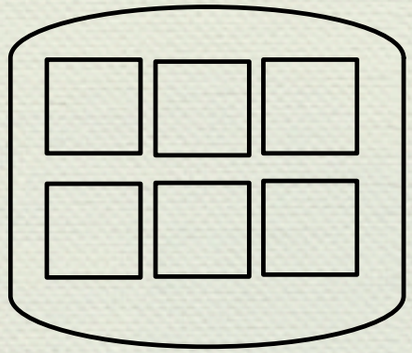
- **Goal:**
  - Block size: M/k bits
  - Bandwidth consumption: (M(N-1))/(k(N-k)) bits, with n participating nodes (n<N, k<N)

Encoding

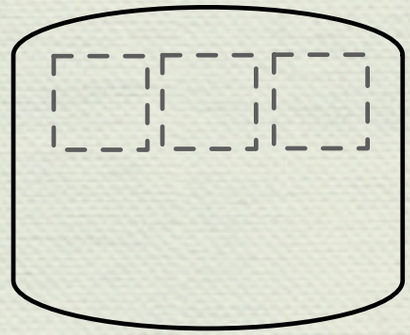
- **Goal:**
  - Block size: M/k bits
  - Bandwidth consumption: (M(N-1))/(k(N-k)) bits, with n participating nodes (n<N, k<N)

- **Randomized Minimum-Storage Regenerating Codes**
  - encode blocks randomly on a finite field GF(2^q)
  - divide the original file into k(N-k) segments
  - each storage node stores one coded block (containing N-k coded segments)
  - when q is large enough (e.g., q=16 or 32), k coded blocks can recover the original file (with a very high probability)
  - to regenerate a coded block, the newcomer should obtain N-1 randomly coded segments from N-1 providers, and then generate N-k randomly coded segments from these N-1 segments

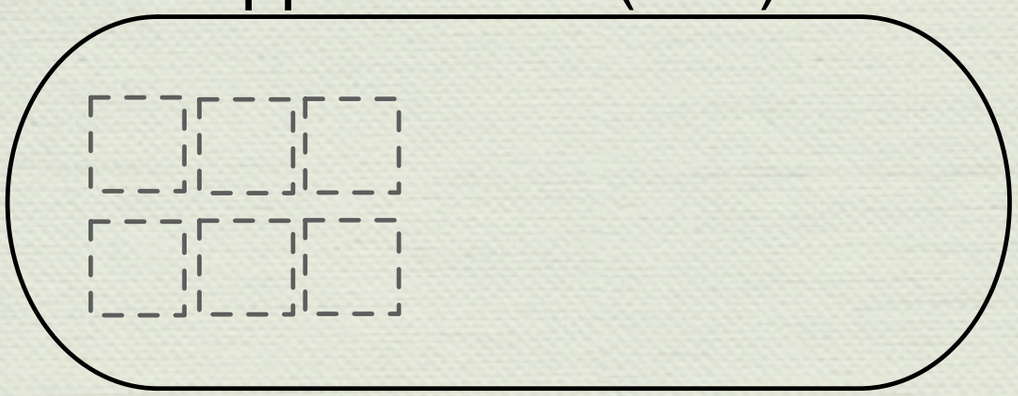
provider 1

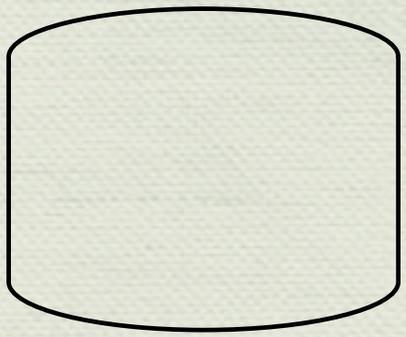


apprentice 1

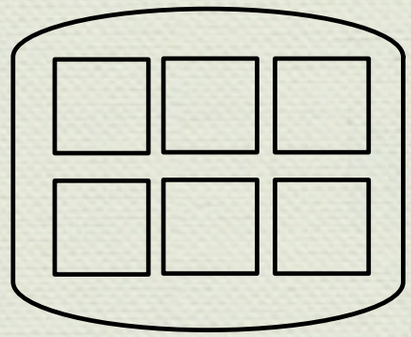


apprentice 2 (root)

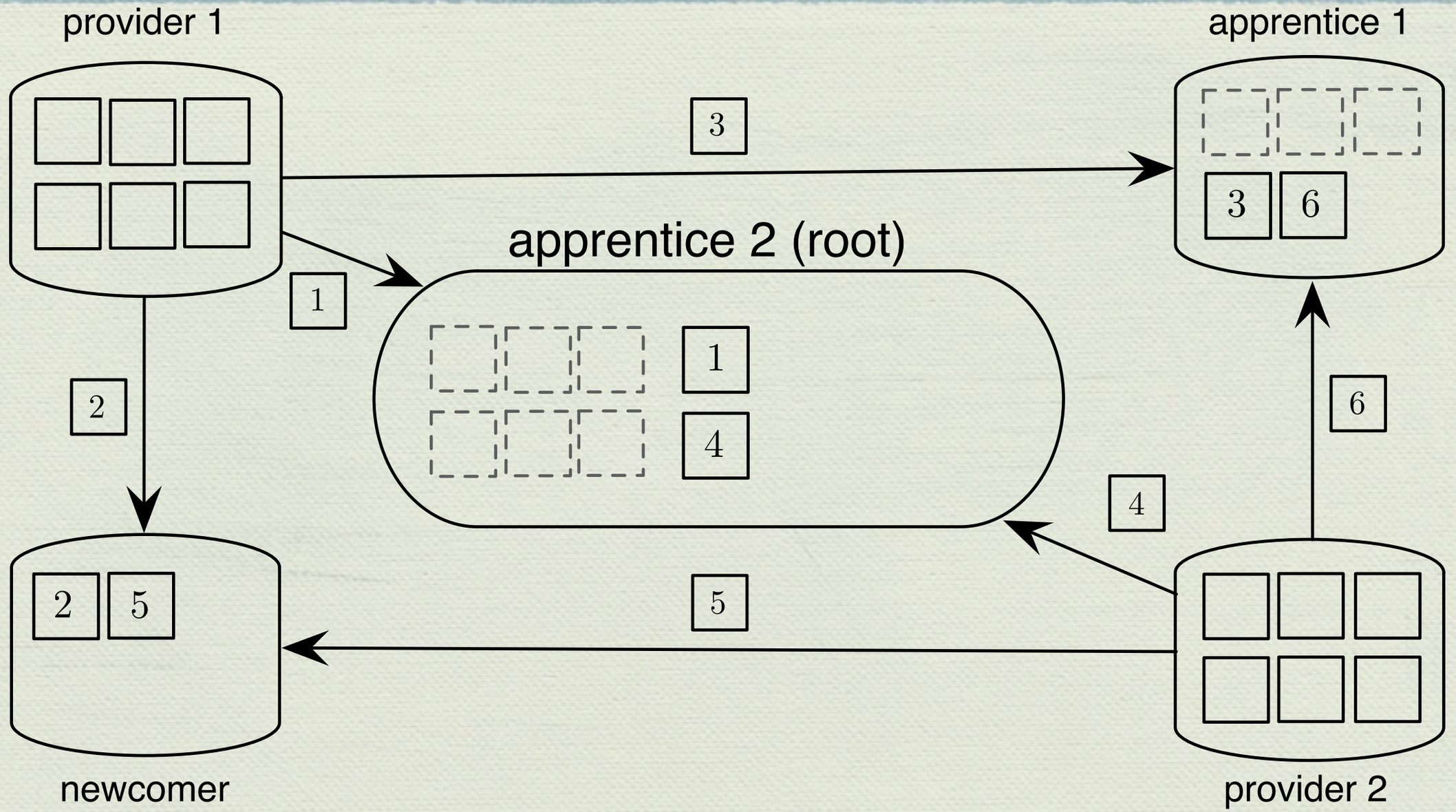




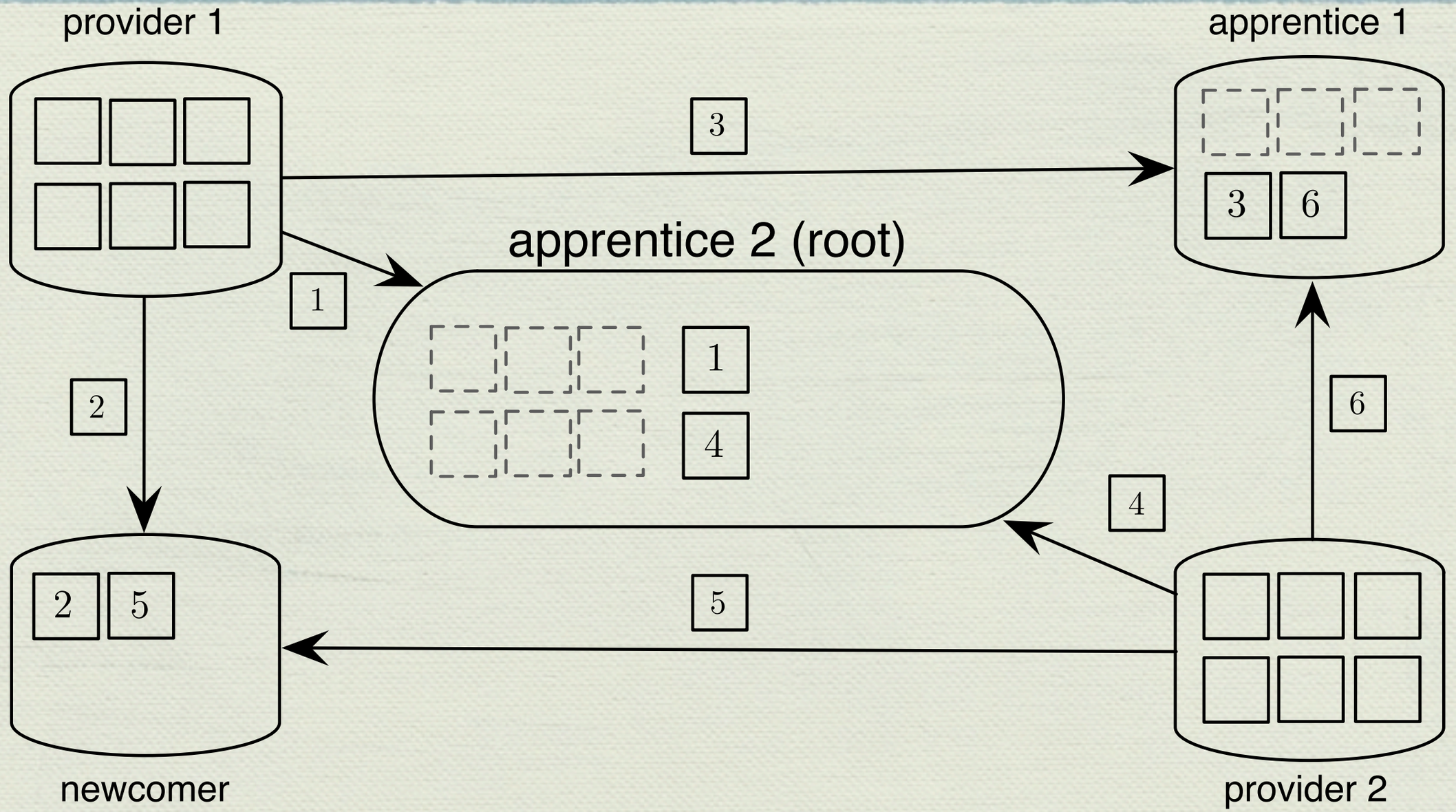
newcomer



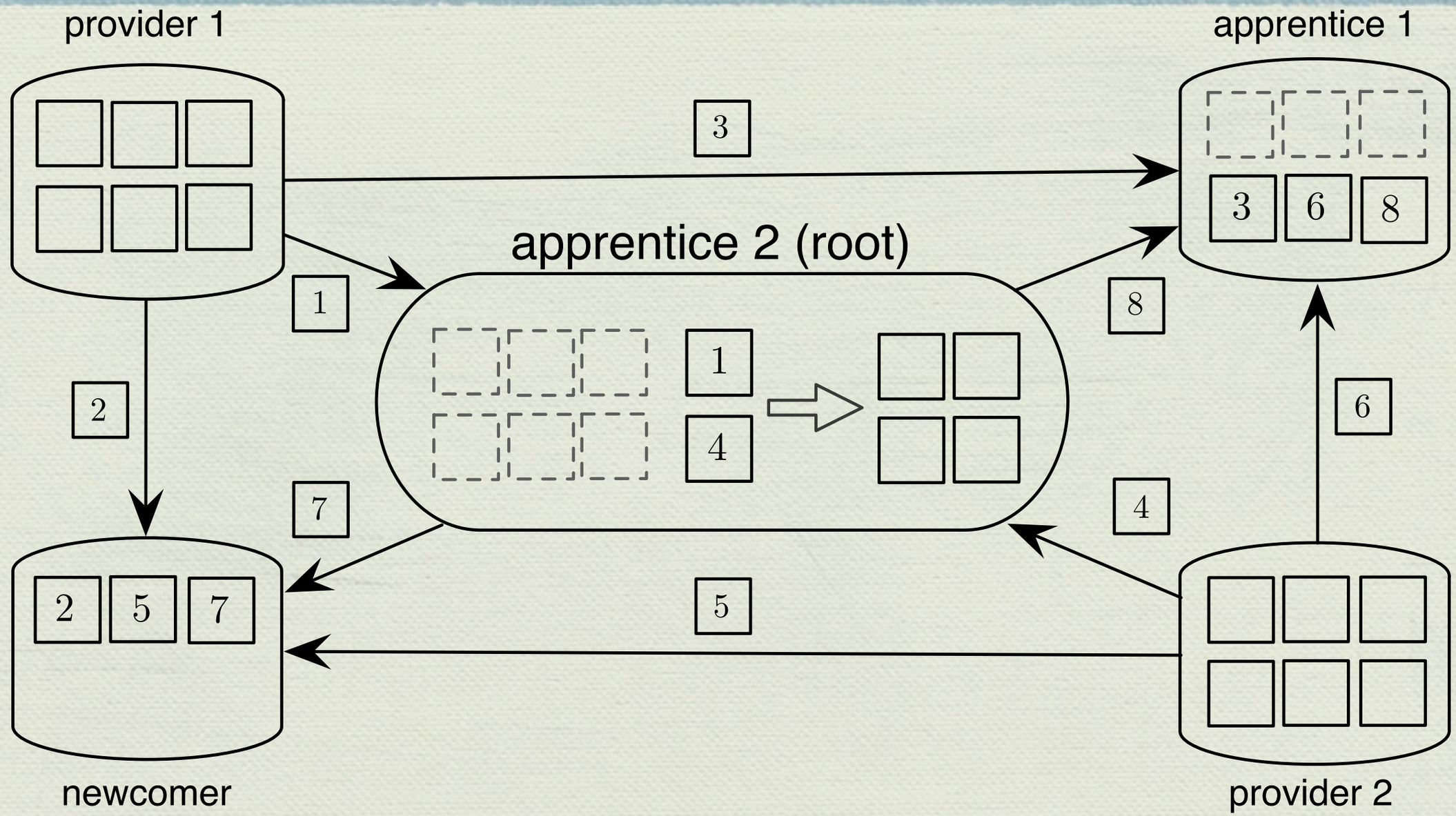
provider 2



- **Each provider sends one randomly coded segments to each of apprentices and the newcomer, respectively**



- **Each provider sends one randomly coded segments to each of apprentices and the newcomer, respectively**
- **The root becomes fully regenerated**



- **Each provider sends one randomly coded segments to each of apprentices and the newcomer, respectively**
- **The root becomes fully regenerated**
- **The root sends one randomly coded segments to each of other apprentices and the newcomer, respectively**

Numbers of Apprentices and Providers

Numbers of Apprentices and Providers

- Assume that there are α apprentices and υ providers in one round of pipelined regeneration

Numbers of Apprentices and Providers

- Assume that there are α apprentices and υ providers in one round of pipelined regeneration
- We can map (1+α) consecutive pipelined regeneration processes to one cooperative regeneration process with (1+α) newcomers and (1+α)υ providers
  - A cooperative regeneration process with N participating nodes and at least k providers consumes M(N-1)/(k(N-k)) bits to regenerate one newcomer

Numbers of Apprentices and Providers

- **Assume that there are α apprentices and υ providers in one round of pipelined regeneration**

- **We can map (1+α) consecutive pipelined regeneration processes to one cooperative regeneration process with (1+α) newcomers and (1+α)υ providers**

  - A cooperative regeneration process with N participating nodes and at least k providers consumes M(N-1)/(k(N-k)) bits to regenerate one newcomer

- **Theorem: If (α+1)(v+1) ≥ N and (α+1)v ≥ k, all roots can be fully regenerated after pipelined regeneration processes, and coded blocks they store can maintain the property that any k coded blocks can recover the original file.**

Minimize participating nodes

Minimize participating nodes

- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$

Minimize participating nodes

- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$
- **Min. $n = \alpha + \upsilon + 1$**
  - s.t. $(\alpha+1)(\upsilon+1)>=N$
  - $(\alpha+1)\upsilon>=k$

Minimize participating nodes

- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$
- **Min. $n = \alpha + \upsilon + 1$**
  - s.t. $(\alpha+1)(\upsilon+1)>=N$
  - $(\alpha+1)\upsilon>=k$
- **n is no less than $2\sqrt{N}-1$**
  - n is minimized when $\alpha=\nu=\sqrt{N}-1$

Minimize participating nodes

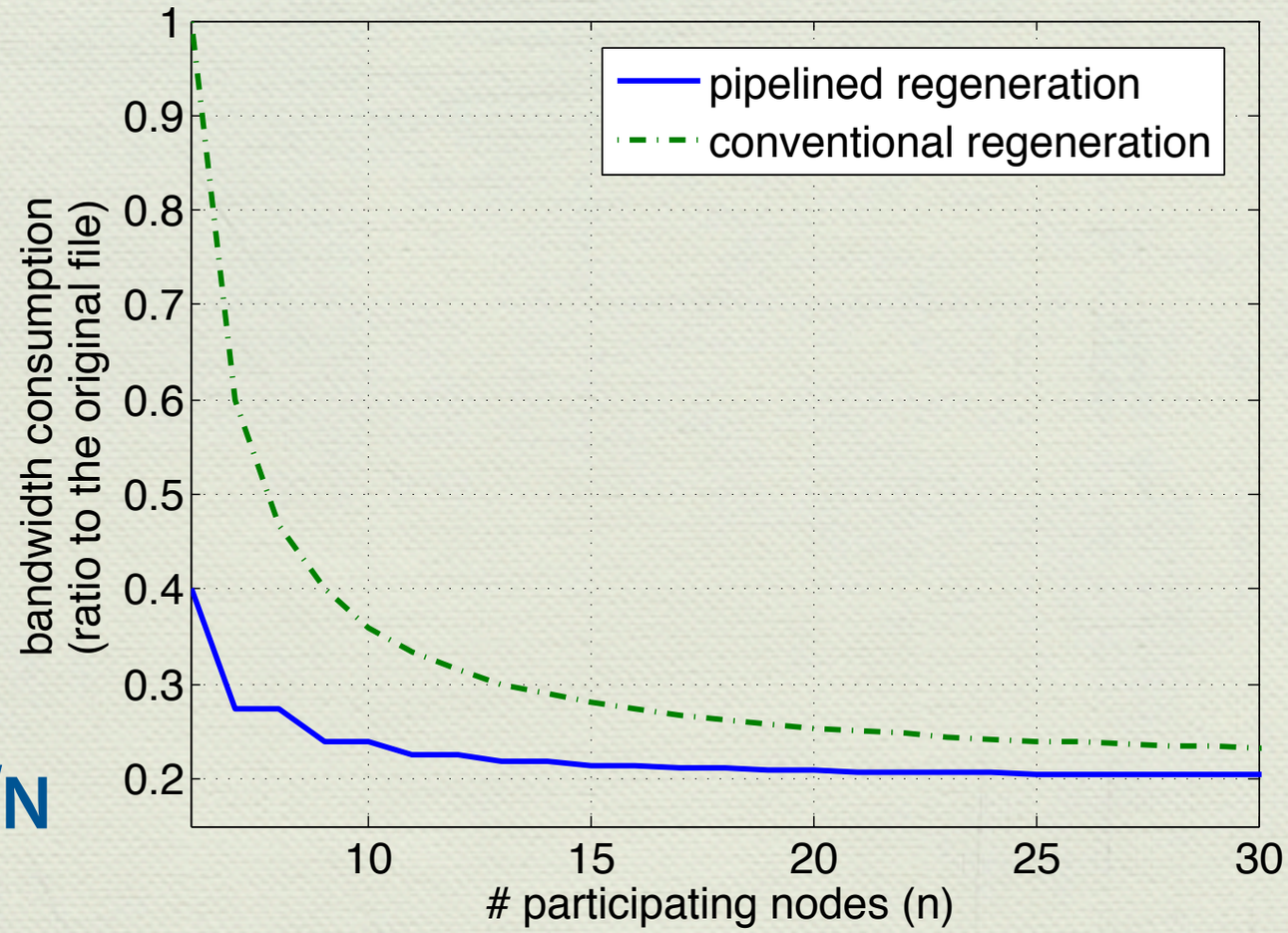
- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$
- **Min. $n = \alpha + \upsilon + 1$**
  - s.t. $(\alpha+1)(\upsilon+1)>=N$
  - $(\alpha+1)\upsilon>=k$
- **n is no less than $2\sqrt{N}-1$**
  - n is minimized when $\alpha=\nu=\sqrt{N}-1$
- **k should be no more than $N-\sqrt{N}$**

Minimize participating nodes

- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$
- **Min. $n = \alpha + \upsilon + 1$**
  - s.t. $(\alpha+1)(\upsilon+1)>=N$
  - $(\alpha+1)\upsilon>=k$
- **n is no less than $2\sqrt{N}-1$**
  - n is minimized when $\alpha=\nu=\sqrt{N}-1$
- **k should be no more than $N-\sqrt{N}$**
- **Bandwidth consumption:**
  - $M(N-1)/(k(N-k)) \rightarrow M/k$

Minimize participating nodes

- **n: the number of participating nodes during pipelined regeneration**
  - $n = \alpha + \upsilon + 1$
- **Min. $n = \alpha + \upsilon + 1$**
  - s.t. $(\alpha+1)(\upsilon+1) >= N$
  - $(\alpha+1)\upsilon >= k$
- **n is no less than $2\sqrt{N}-1$**
  - n is minimized when $\alpha=\nu=\sqrt{N}-1$
- **k should be no more than $N-\sqrt{N}$**
- **Bandwidth consumption:**
  - M(N-1)/(k(N-k)) -> M/k



Storage Overhead

Storage Overhead

- **Conventional regeneration with regenerating codes requires N-1 providers**
  - there must be at least N storage nodes existing in the system

Storage Overhead

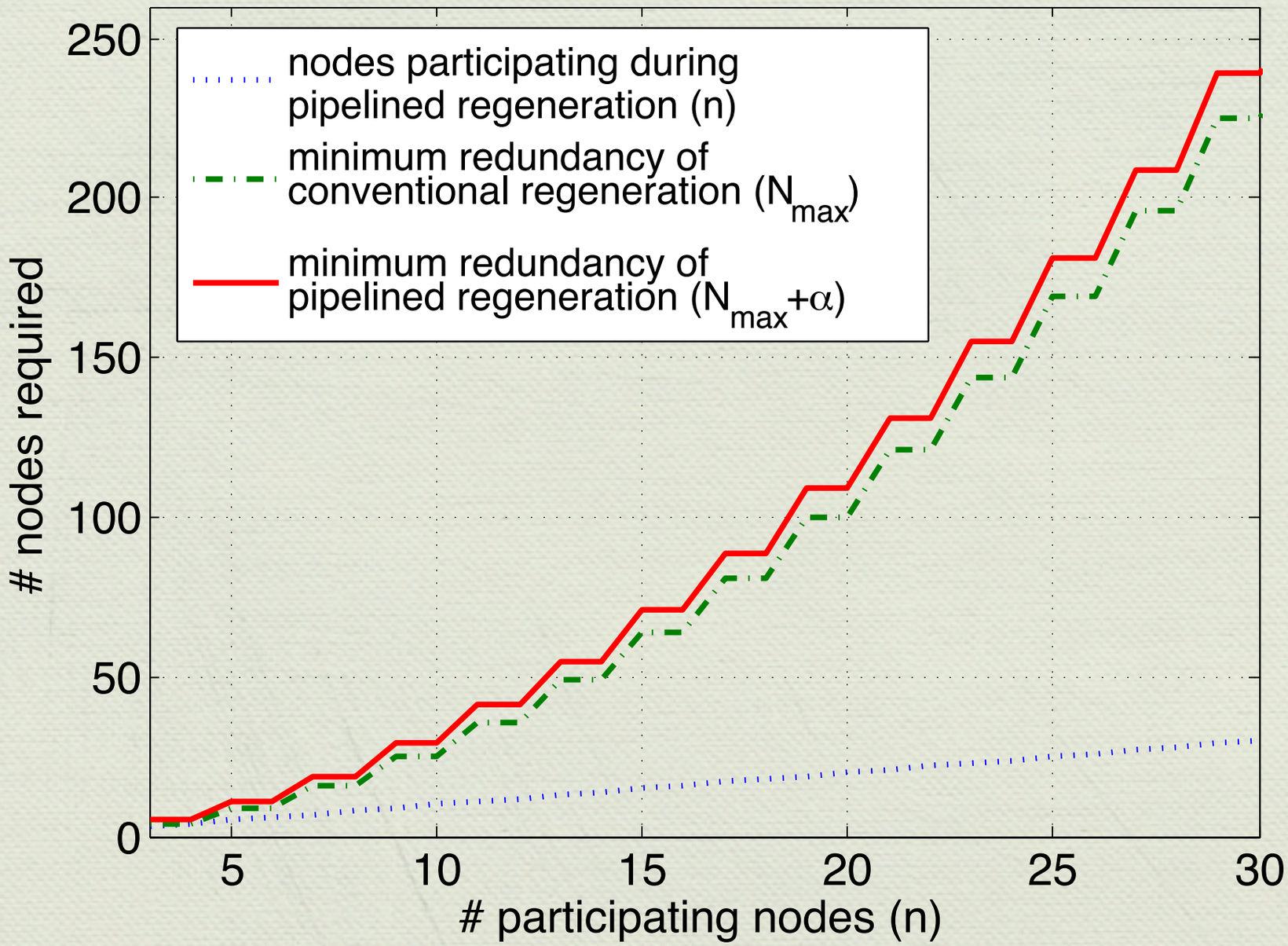
- **Conventional regeneration with regenerating codes requires N-1 providers**
  - there must be at least N storage nodes existing in the system
- **With the same coding scheme, pipelined regeneration requires at least $2\sqrt{N}-1$ participating nodes**
  - apprentice: α
  - how many storage nodes should exist in the system?
  - in α+1 consecutive pipelined regeneration processes, no providers can be utilized more than once
  - pipelined regeneration requires at least N storage nodes in the system

Storage Overhead

- **Conventional regeneration with regenerating codes requires N-1 providers**
  - there must be at least N storage nodes existing in the system
- **With the same coding scheme, pipelined regeneration requires at least $2\sqrt{N}-1$ participating nodes**
  - apprentice: α
  - how many storage nodes should exist in the system?
  - in α+1 consecutive pipelined regeneration processes, no providers can be utilized more than once
  - pipelined regeneration requires at least N storage nodes in the system
- **Due to the overhead of apprentices, pipelined regeneration requires more storage overheads than conventional regeneration**

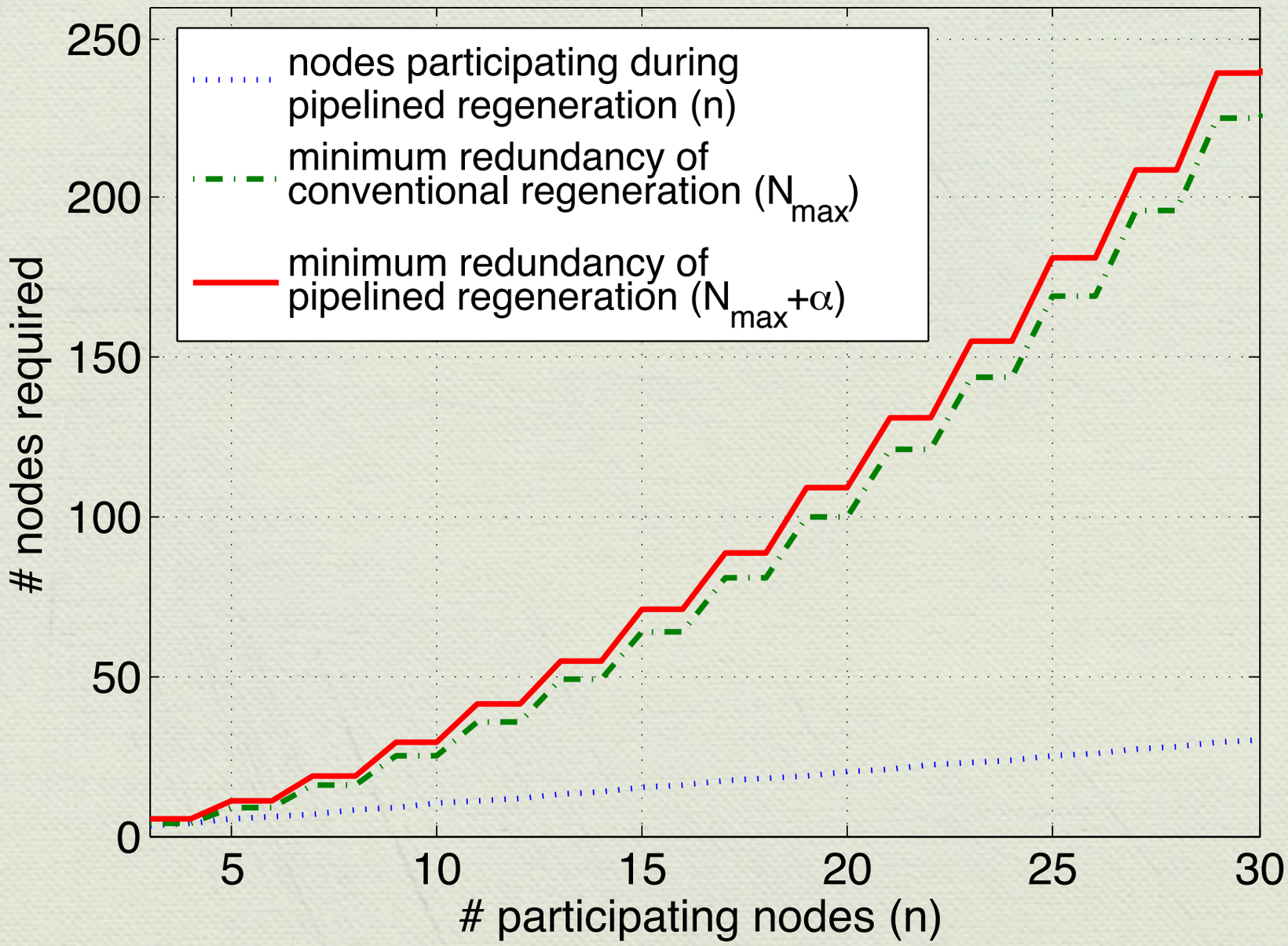
Overhead of Apprentices

- When n is minimized, $\alpha=\sqrt{N}-1$



Overhead of Apprentices

- When n is minimized, $\alpha=\sqrt{N}-1$



The overhead brought by apprentices is marginal

Conclusions

Conclusions

- We propose a novel fashion of regeneration, called pipelined regeneration, that significantly saves the number of participating nodes

Conclusions

- We propose a novel fashion of regeneration, called pipelined regeneration, that significantly saves the number of participating nodes
- Properties of regenerating codes are preserved
  - With the same number of participating nodes, regeneration codes with better performance can be supported

Conclusions

- **We propose a novel fashion of regeneration, called pipelined regeneration, that significantly saves the number of participating nodes**

- **Properties of regenerating codes are preserved**
  - With the same number of participating nodes, regeneration codes with better performance can be supported

- **Storage overhead is marginal**

Thank you!

For more information:
http://homepage.fudan.edu.cn/junli