

# Pairwise Interaction Model for Gossip Protocols

**Rena Bakhshi**, Daniela Gavidia,  
Wan Fokkink and Maarten van Steen

Department of Computer Science  
Vrije Universiteit Amsterdam

ROCKS workshop  
March 27, 2011



# The Big Picture

- Applications for large-scale networks
- Class of gossip protocols
- Modelling pairwise interactions
- Emergent behaviour of gossip systems



# The Big Picture

- Applications for large-scale networks
- Class of gossip protocols
- Modelling pairwise interactions
- Emergent behaviour of gossip systems
- Combination with other frameworks



# The Framework Scope

- performance of large-scale system
- different data items (e.g., news headlines)
- the data does not fit into the local cache (non-trivial case)
- observe the flow of the data

# The Framework Scope

Each node

- holds list of data items (*cache*)
- periodically communicates with random peer
- exchanges items

## Generic Algorithm

```
wait ( $\Delta t$  time units)  
 $p \leftarrow \text{RandomPeer}();$   
 $\sigma \leftarrow \text{PrepareMsg}();$   
send  $\sigma$  to  $p$ ;  
wait until receive( $\sigma_p$ )  
 $\sigma \leftarrow \text{Update}(\sigma, \sigma_p);$ 
```

(a) active thread

```
wait until receive( $\sigma_p$ )  
 $\sigma \leftarrow \text{PrepareMsg}();$   
send  $\sigma$  to sender( $\sigma_p$ );  
 $\sigma \leftarrow \text{Update}(\sigma, \sigma_p);$ 
```

(b) passive thread

# The Framework Scope

Each node

- holds list of data items (*cache*)
- periodically communicates with random peer
- exchanges items

## Generic Algorithm

```
wait ( $\Delta t$  time units)  
 $p \leftarrow \text{RandomPeer}();$   
 $\sigma \leftarrow \text{PrepareMsg}();$   
send  $\sigma$  to  $p$ ;  
wait until receive( $\sigma_p$ )  
 $\sigma \leftarrow \text{Update}(\sigma, \sigma_p);$ 
```

(a) active thread

```
wait until receive( $\sigma_p$ )  
 $\sigma \leftarrow \text{PrepareMsg}();$   
send  $\sigma$  to sender( $\sigma_p$ );  
 $\sigma \leftarrow \text{Update}(\sigma, \sigma_p);$ 
```

(b) passive thread

Protocol-dependent operations:

- RandomPeer()
- PrepareMsg()
- Update( $\sigma, \sigma_p$ )

# Gossip Protocols

**Newscast:** simple data dissemination

**Shuffle:** data preservation

**Cyclon:** peer sampling service ...

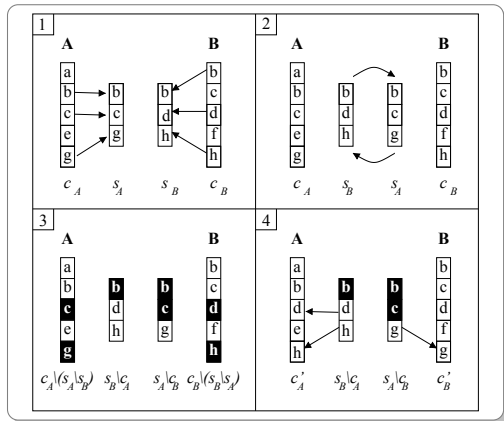


Figure: Shuffle

# The Framework

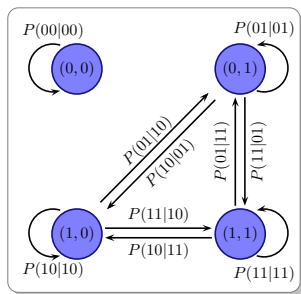
- Modelling pairwise interactions

## Assumptions

- caches are full
- atomic procedure of the data exchange
- $0 < \text{message size } s \leq \text{cache size } c \ll \text{item types } n$

## State Transitions

- The spread of a generic item
- States
  - item is *not* in cache (**0**)
  - item *is* in cache (**1**)



# Transition Probabilities

## Probabilities

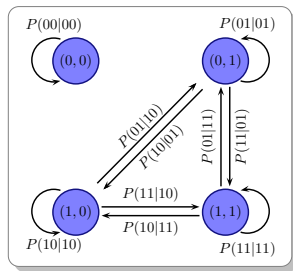
- $P(\mathbf{A}_1, \mathbf{B}_1 | \mathbf{A}_0, \mathbf{B}_0)$
- after state  $(\mathbf{A}_1, \mathbf{B}_1)$
- before state  $(\mathbf{A}_0, \mathbf{B}_0)$
- depend on  $P_{\text{select}}$  and  $P_{\text{drop}}$

## Components

$P_{\text{select}}$  : a node selects an item during exchange

$P_{\text{drop}}$  : a node drops (selected) item after exchange

- $P_{\text{select}}$  depends on PrepareMsg() routine
- $P_{\text{drop}}$  depends on Update( $\sigma, \sigma_p$ ) routine



# Transition Probabilities

## Probabilities

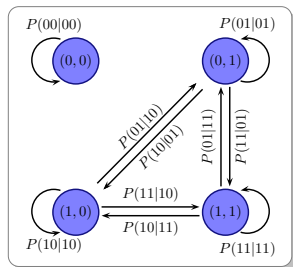
- $P(\mathbf{A}_1, \mathbf{B}_1 | \mathbf{A}_0, \mathbf{B}_0)$
- after state  $(\mathbf{A}_1, \mathbf{B}_1)$
- before state  $(\mathbf{A}_0, \mathbf{B}_0)$
- depend on  $P_{\text{select}}$  and  $P_{\text{drop}}$

## Components

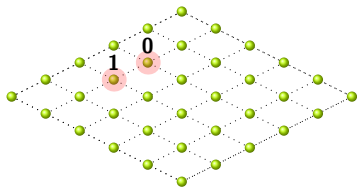
$P_{\text{select}}$  : a node selects an item during exchange

$P_{\text{drop}}$  : a node drops (selected) item after exchange

- PrepareMsg(): uniformly at random
- $P_{\text{select}} = \frac{s}{c}$



## Example (Shuffle)



(0, 0)

(0, 1)

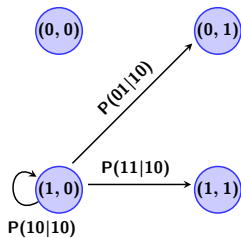
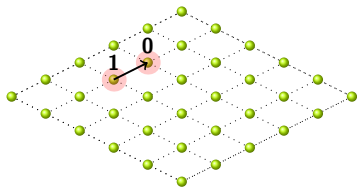
(1, 0)

(1, 1)

## Example

- (1, 0)

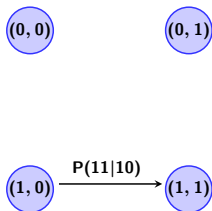
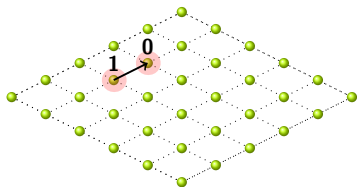
## Example (Shuffle)



Example

•  $(1,0) \rightarrow$

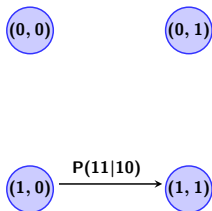
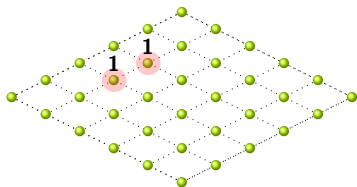
# Example (Shuffle)



## Example

- $(1, 0) \xrightarrow{P(11|10)}$

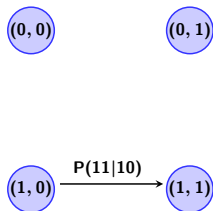
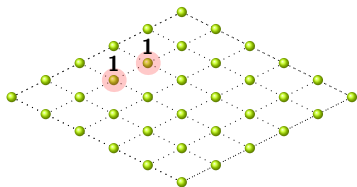
## Example (Shuffle)



## Example

- $(1, 0) \xrightarrow{P(11|10)} (1, 1)$

## Example (Shuffle)



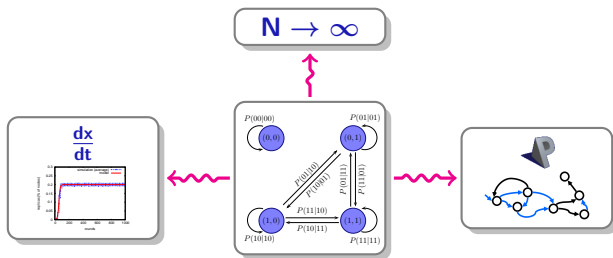
### Example

- $(1, 0) \xrightarrow{P(11|10)} (1, 1)$
- $P(11|10) = P_{\text{select}} \cdot P_{\text{-drop}}$

# Applications

To analyse the emergent behaviour, can be combined with

- mean-field framework or ODEs
- model-checking tools (e.g., PRISM)
- numerical simulations



# Perfect Communication Channels

## Replication

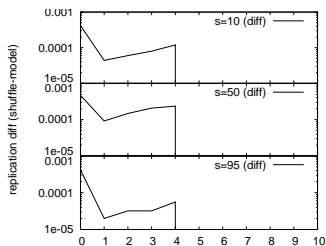
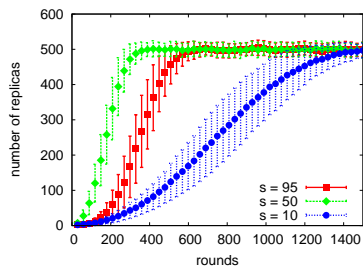


Figure: Shuffle, Grid range 1 (4 neighbours),  $c=100$ ,  $n=500$

# Perfect Communication Channels

## Coverage

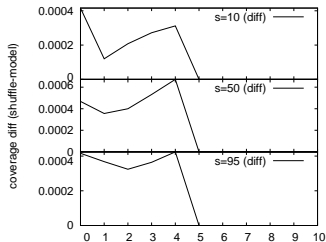
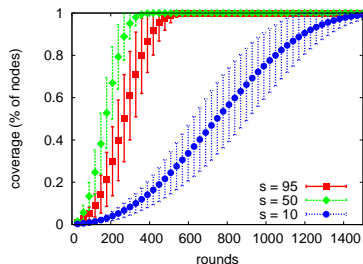


Figure: Shuffle, Grid range 1 (4 neighbours),  $c=100$ ,  $n=500$

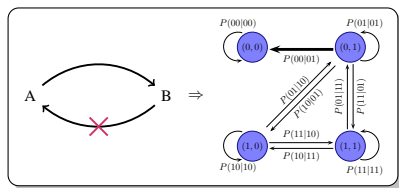
# Gossiping with Losses

? What if channels are lossy

## Case study: Shuffle

Message loss change the emergent behaviour of system

- data preservation



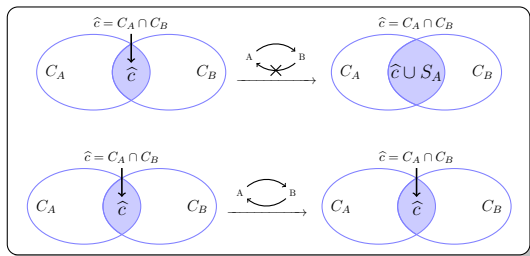
# Gossiping with Losses

? What if channels are lossy

## Case study: Shuffle

Message loss change the emergent behaviour of system

- small-world phenomenon



# Generalized Framework

- ! Relax the assumptions

## Assumptions Affected

- Gossip *is* atomic procedure
- Uniform distribution of items in *whole* network

## Assumptions Revisited

- Gossip is *not necessary* atomic procedure
- Uniform distribution of items in *neighborhoods*



# Topology and Lossy Networks

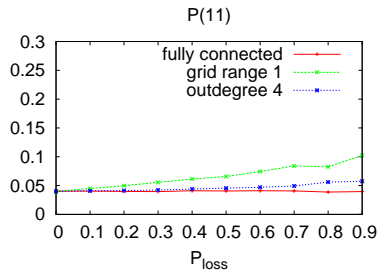
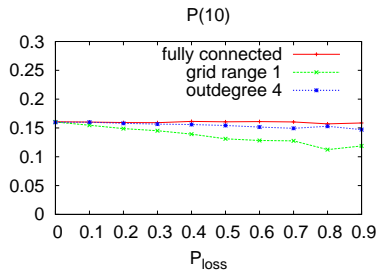


Figure: Probabilities of exchange occurring between nodes  $(1, 0)$  and  $(1, 1)$ .

# Topology and Lossy Networks

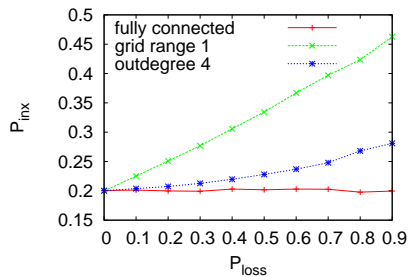
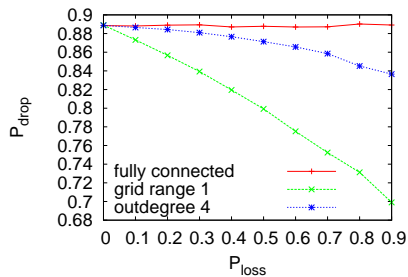


Figure: Probabilities  $P_{\text{drop}}$  and  $P_{\text{inx}}$ .

# Generalized Framework

## The idea

- ! Isolate topology varying *component*

## We can

- model it analytically (given network topology)
- measure it experimentally (from single run)

# Results

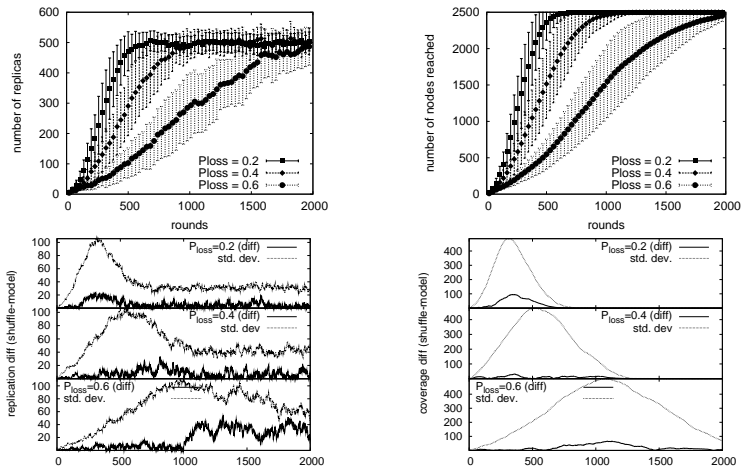


Figure: Replication and coverage, Grid range 1

# Results

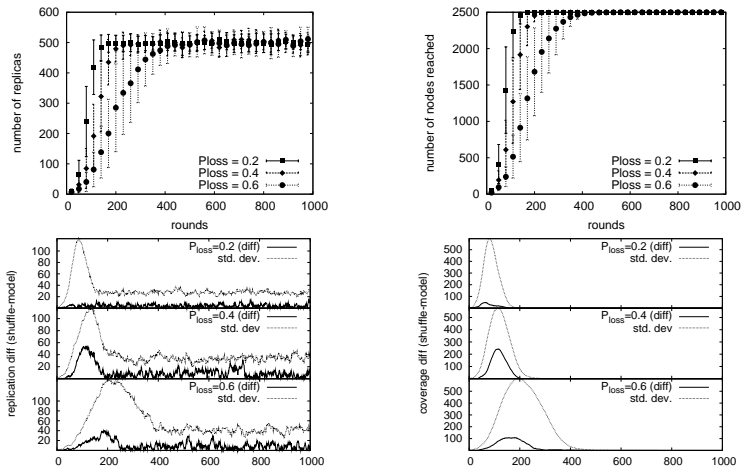


Figure: Replication and coverage, Topology with outdegree 4