

# CloudMan: A fully decentralized middleware

Márk Jelasity  
MTA and University of Szeged, Hungary

# Outline

- **CloudMan**
  - Motivation
  - Basic concepts
  - The architecture
- **Slicing**
  - The problem
  - The protocol
  - Fault tolerance

# Motivation

- P2P applications are successful
  - Successfully exploit decentralized dynamic resources
    - **Content distribution, file sharing, search, telephony, streaming, etc**
  - Majority of Internet traffic
  - Immense effect on culture, entertainment, science and technology

# Motivation

- But
  - Popular applications are hardwired by nature
    - **focus on single functions**
    - **No possibility to stop/load/update applications**
    - **No possibility to share resources (“multitasking”)**
  - In research: small scale, researched focused middleware (PlanetLab)
    - **Mechanisms to manage applications**
    - **Resource assignment is “by hand”**
    - **Small scale and error prone**
    - **Heavy weight: difficult to join**

# Motivation

- The key is the platform. We see
  - ISP-s handing out modems, routers
  - Cable companies handing out set-top boxes
  - Telcos handing out phones and moving to IP networking
  - These are all in the range of million-device, privately owned, but unreliable, IP networks
  - These networks represent enormous potential which is mostly wasted

# Motivation

- In the networks mentioned above it is possible to move P2P to the next level
  - Single administrative domain: strong security possible
  - Application possibilities are endless
    - **Managing the network itself**
    - **Offering services**
    - **Even reselling P2P resources to third parties (GRID)**

# Wish List

- **Middleware that is able to**
  - Support arbitrary existing p2p algorithms
  - start/stop p2p applications dynamically
  - Support many applications in parallel
  - Manage resources
- **Properties**
  - Minimal and high level user intervention (autonomic)
  - Scalable, lightweight

# System model

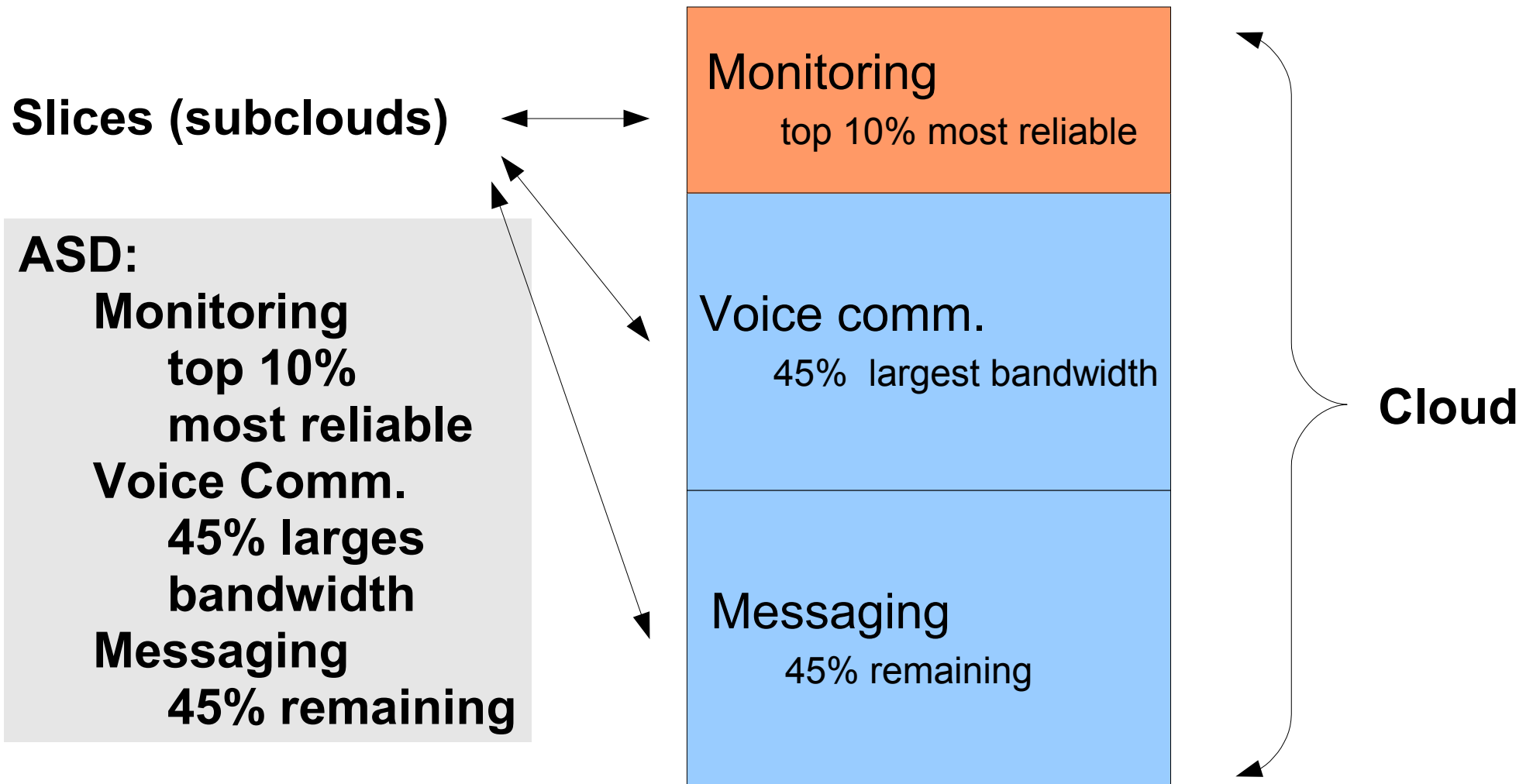
- Cloud
  - Perhaps millions of computing devices, constantly joining and leaving
  - Continuous medium (like the amorphous medium)
  - Single administrative domain
- All benign faults are possible
  - Churn, node and message failures
- Expected deployed services a few dozen
  - Systems services: monitoring, control
  - User services: gaming, messaging, voice, etc



# CloudMan approach

- Application suite description
  - Declarative description of all applications with attributes (requested resources, etc)
  - ALL nodes in the cloud have a copy of this
  - It describes how the system should look like referring only to the cloud as a continuous medium
    - **Independent of cloud size: easy interface for merge and split**
    - **Easy to understand by humans**
- Self-organizing middleware to enforce the ASD in the face of the dynamic cloud

# ASD example



# Loose biological analogy

- ASD is roughly DNA
  - Common to all participants (nodes/cells)
  - Defines how the system/organism should look like and behave
  - Self-organization based on ASD/DNA instead of central control
- differences
  - We allow changing the ASD at run time and the system must adapt quickly
  - Obviously the DNA is not directly declarative

# ASD as user interface

- Stopping an application
  - Remove it from the ASD and adjust the resource assignments in the ASD if needed
- Reassign resources
  - Reassign resources in the ASD
- Start an application
  - Add an entry to the ASD
- Users/admins only interact with the ASD

# Implementation

- **Cloud**
  - Peer sampling service: gossip based implementation (newsast, cyclon, etc)
- **ASD**
  - (small) distributed database with epidemic updates
- **Middleware services**
  - Slicing
  - Bootstrapping (T-Man, T-Chord, etc)
  - Churn handling (leave and join)

# Usage scenario (short running)

- User writes app against CloudMan API
  - Read ASD, access to services
- Launch application
  - Modify ASD, as a result a subcloud will be assigned
  - Application code is spread, supporting overlay networks are built by the bootstrapping service
- Terminate application
  - Remove its entry from the ASD

# Example applications

- Simple services for users of the network
  - Messaging, content distribution, etc
- Delegating subclouds to external authorities
  - Recursive structure (also delegation)
  - A GRID, for example, with special services specific inside the subcloud

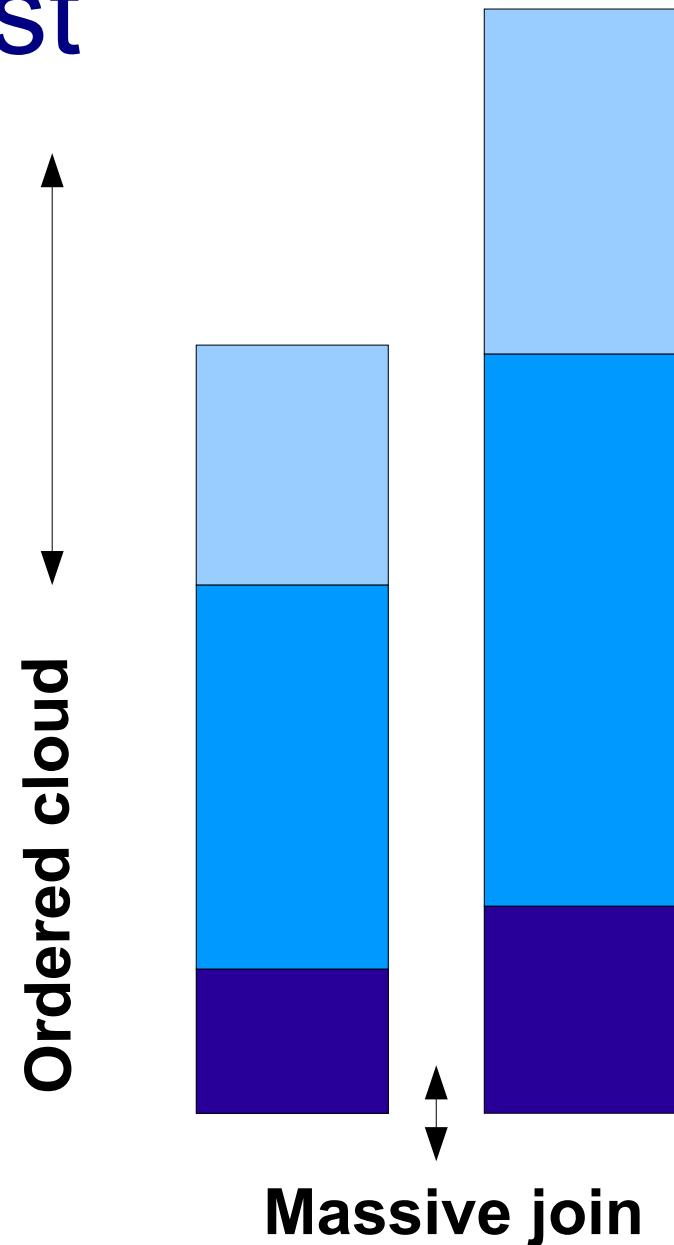
# Ordered Slicing

- Core service of CloudMan
  - ASD defines non-overlapping slices (subclouds)
  - This is the main resource allocation mechanism
- Other similar systems with resource management
  - PlanetLab: discovery, followed by static assignment
  - GRID mechanisms
- We want to automatically maintain the resource assignment!



# Wish list

- Self-organizing;  
maintain ASD and
  - Handle churn
  - Massive join/leaves
- Possibility to order
  - Top 10% most reliable
  - 50% largest storage
  - Etc
- Purely local decision



# Approach

- Possible approaches
  - Determine ranking and network size
  - Approximate distribution
- Our approach is gossip based
  - Assign a random ID to all nodes
  - Gossip based ordering of nodes according to a selected criterion
  - Determine slices

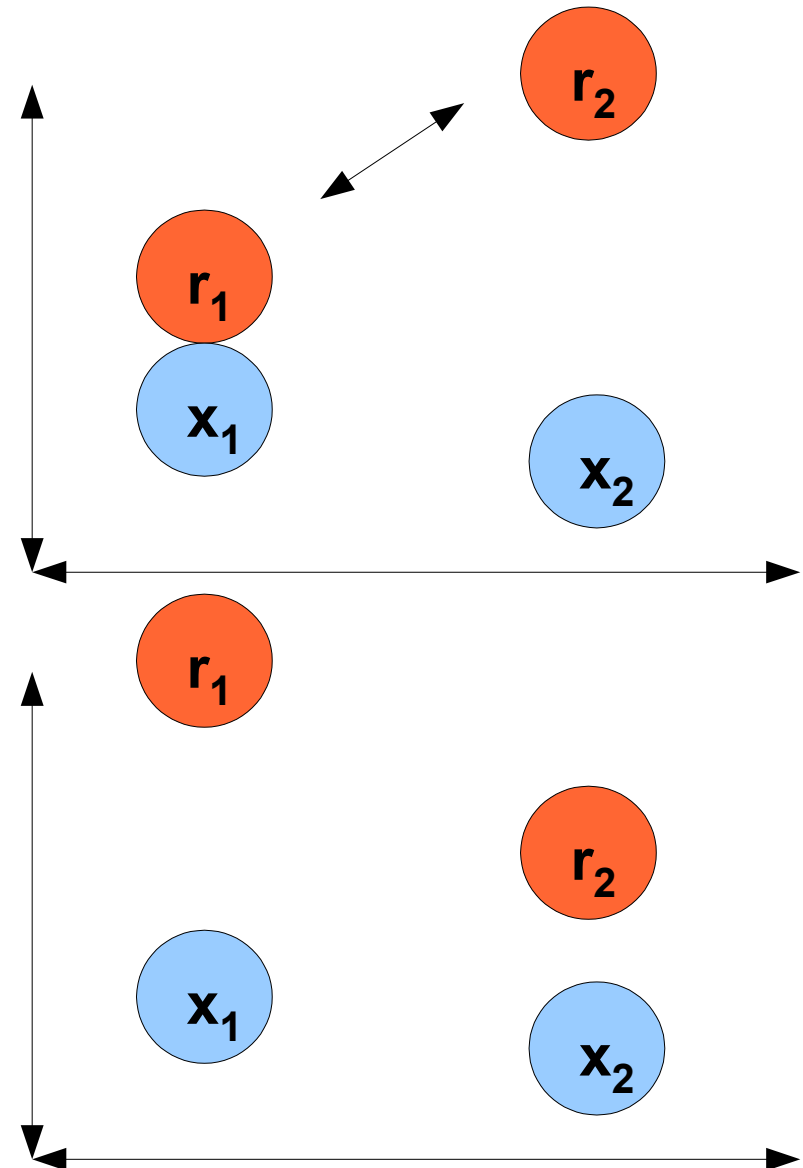
75	1.0
49	1.1
43	1.2
62	3.0
92	5.0
12	9.0



12	1.0
43	1.1
49	1.2
62	3.0
75	5.0
92	9.0

# Main idea

- Start with peer sampling service
  - Periodic gossip with random peers
- When possible, swap random value
  - Possible when resource value is smaller but random value is larger or vice versa



# The sorting protocol

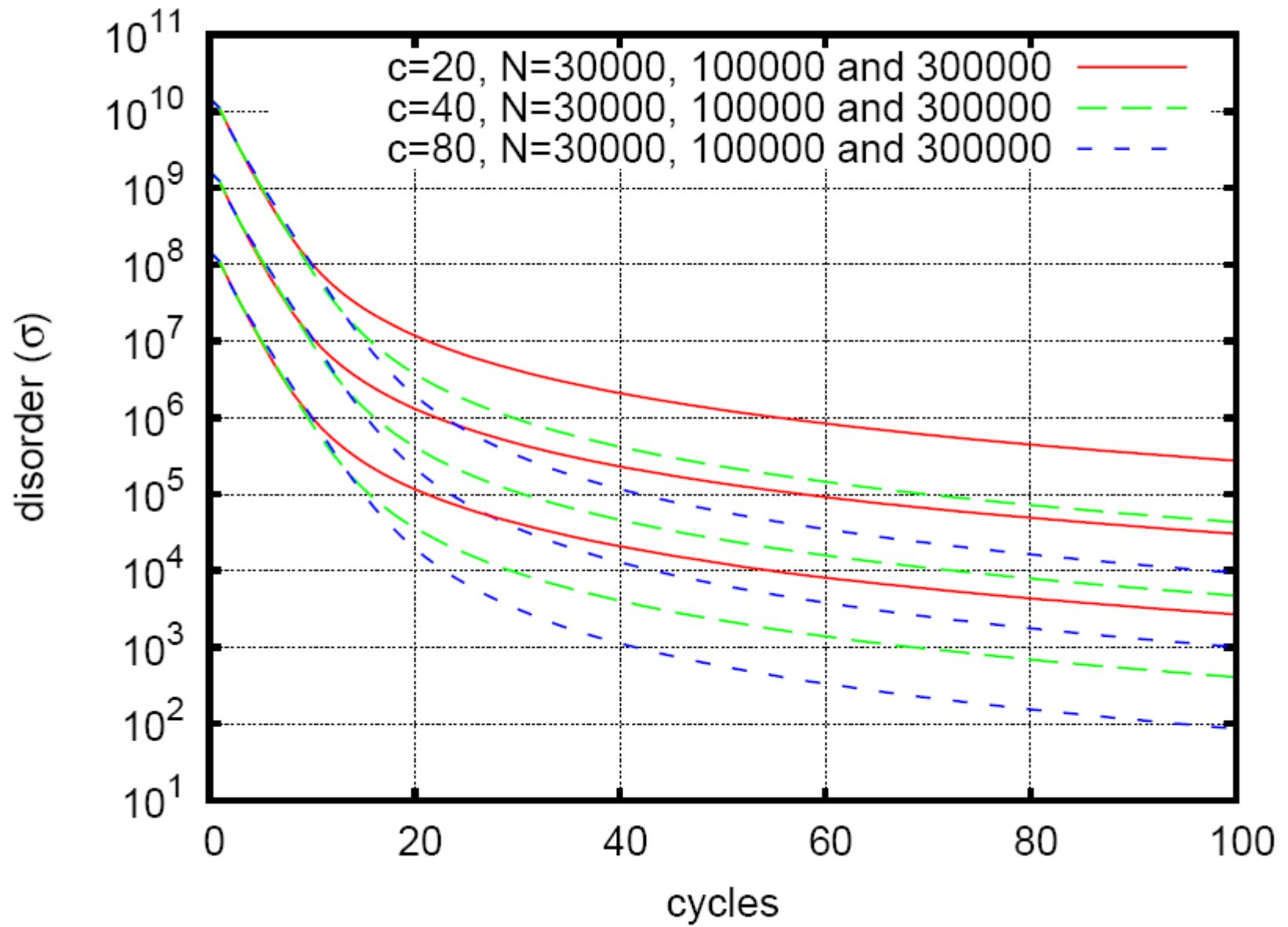
```
1: loop
2:   wait( $\Delta_r$ )
3:    $p \leftarrow$  random element from view
4:   buffer  $\leftarrow$  view  $\cup$   $\{(myAddress, timestamp, x_q, r_q)\}$ 
5:   send buffer to  $p$ 
6:   receive buffer $_p$  from  $p$ 
7:   view  $\leftarrow$  youngest  $c$  entries of buffer $_p \cup$  view
8:    $i \leftarrow$  peer from view such that  $(x_i - x_q)(r_i - r_q) < 0$ 
9:   send  $(x_q, r_q)$  to  $i$ 
10:   $r_q \leftarrow$  get  $r_i$  from  $i$ 
11: end loop
```

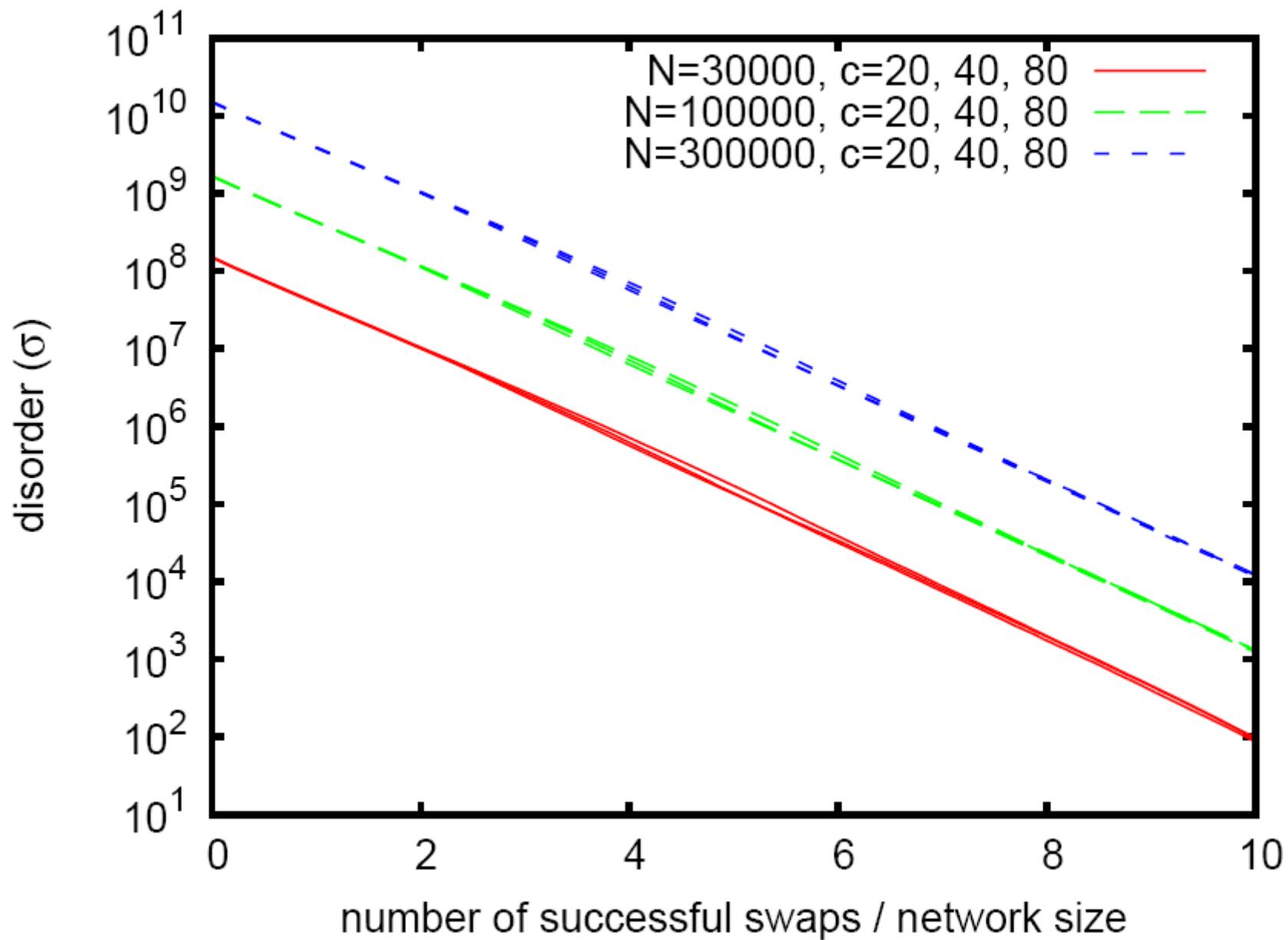
(a) active thread at node  $q$

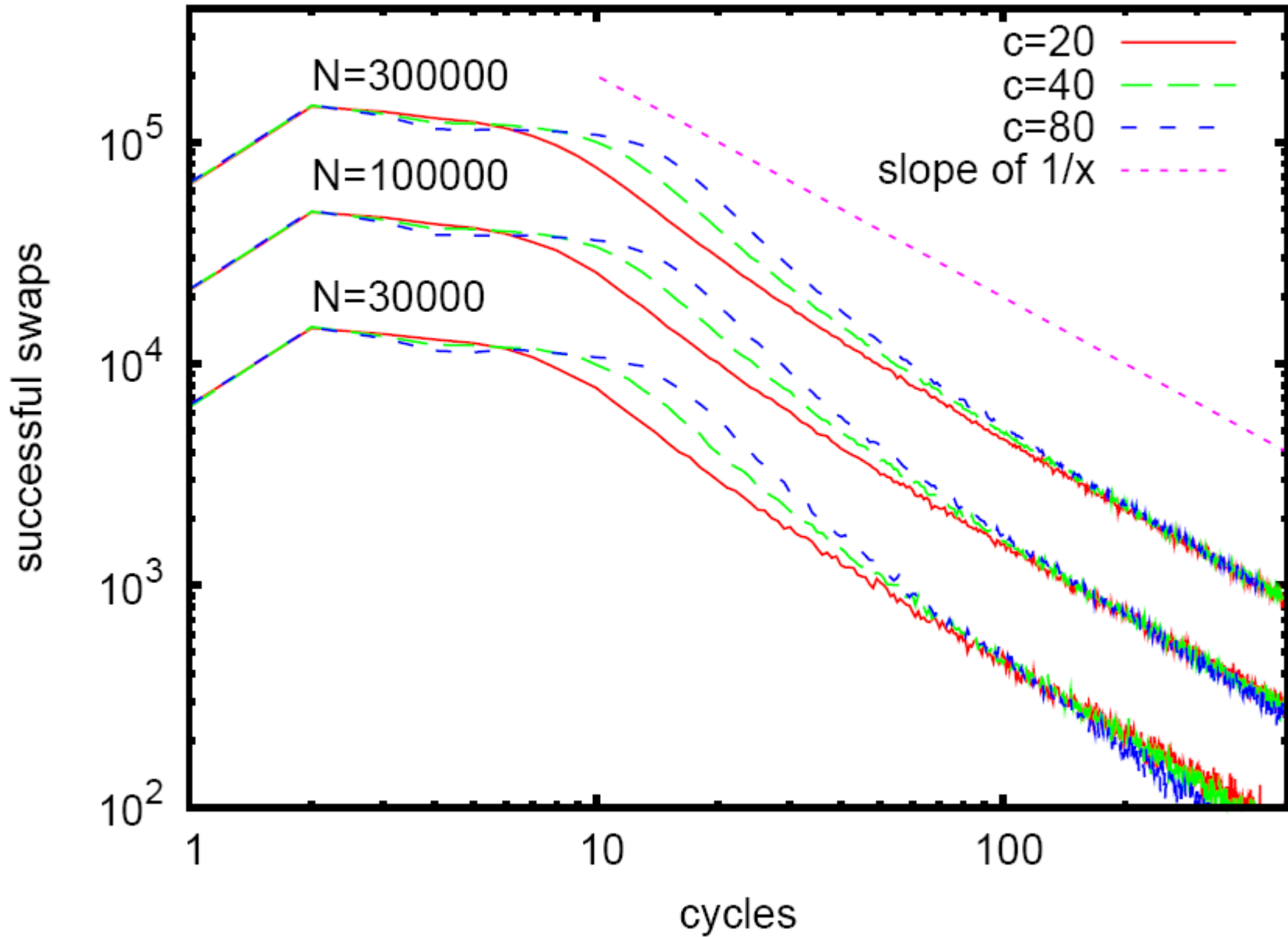
# Measure of performance

- A variance like measure to characterize goodness of sorting

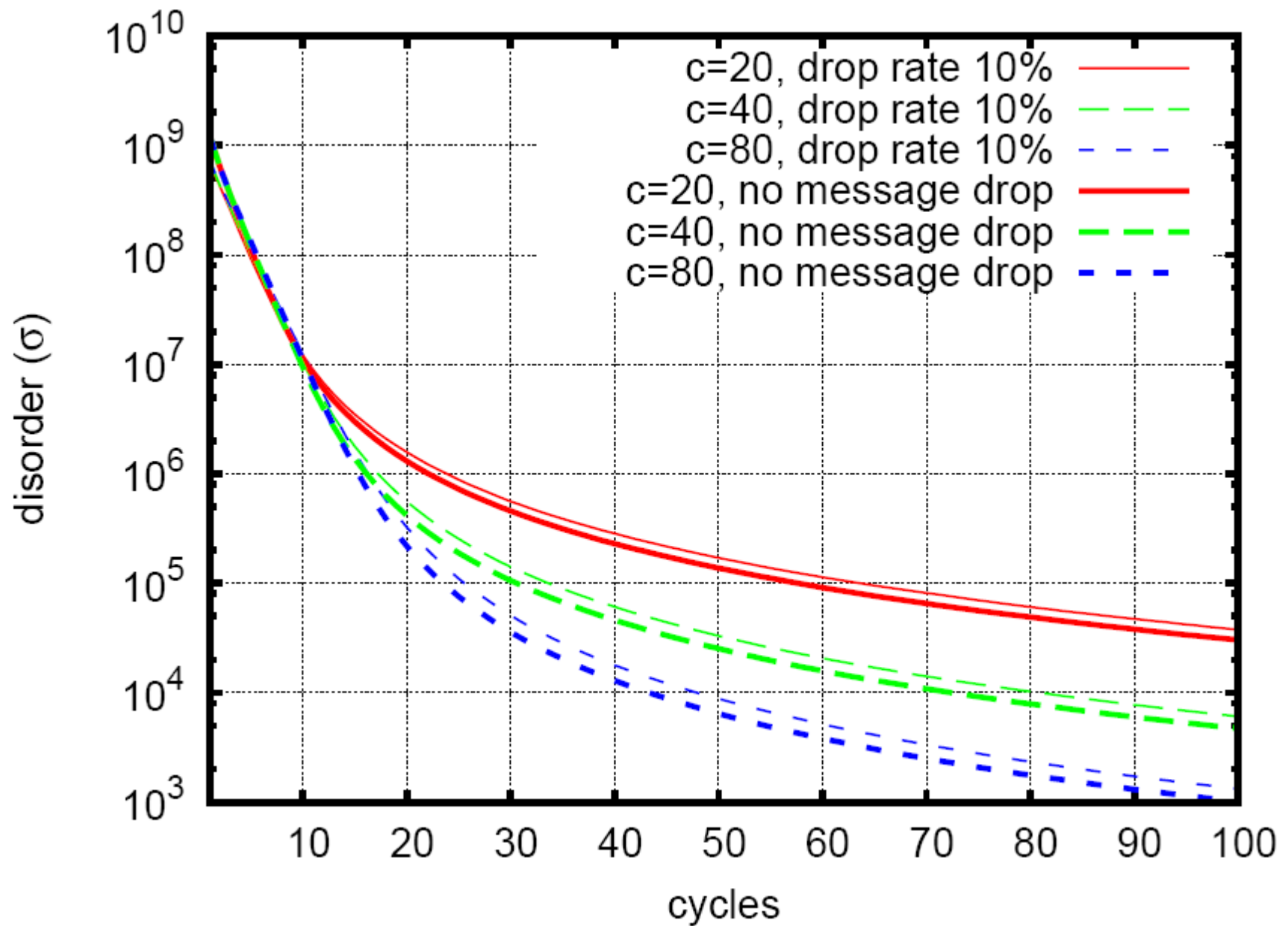
$$\sigma(t) = \frac{1}{N} \sum_{j=1}^N (j - i_j(t))^2$$



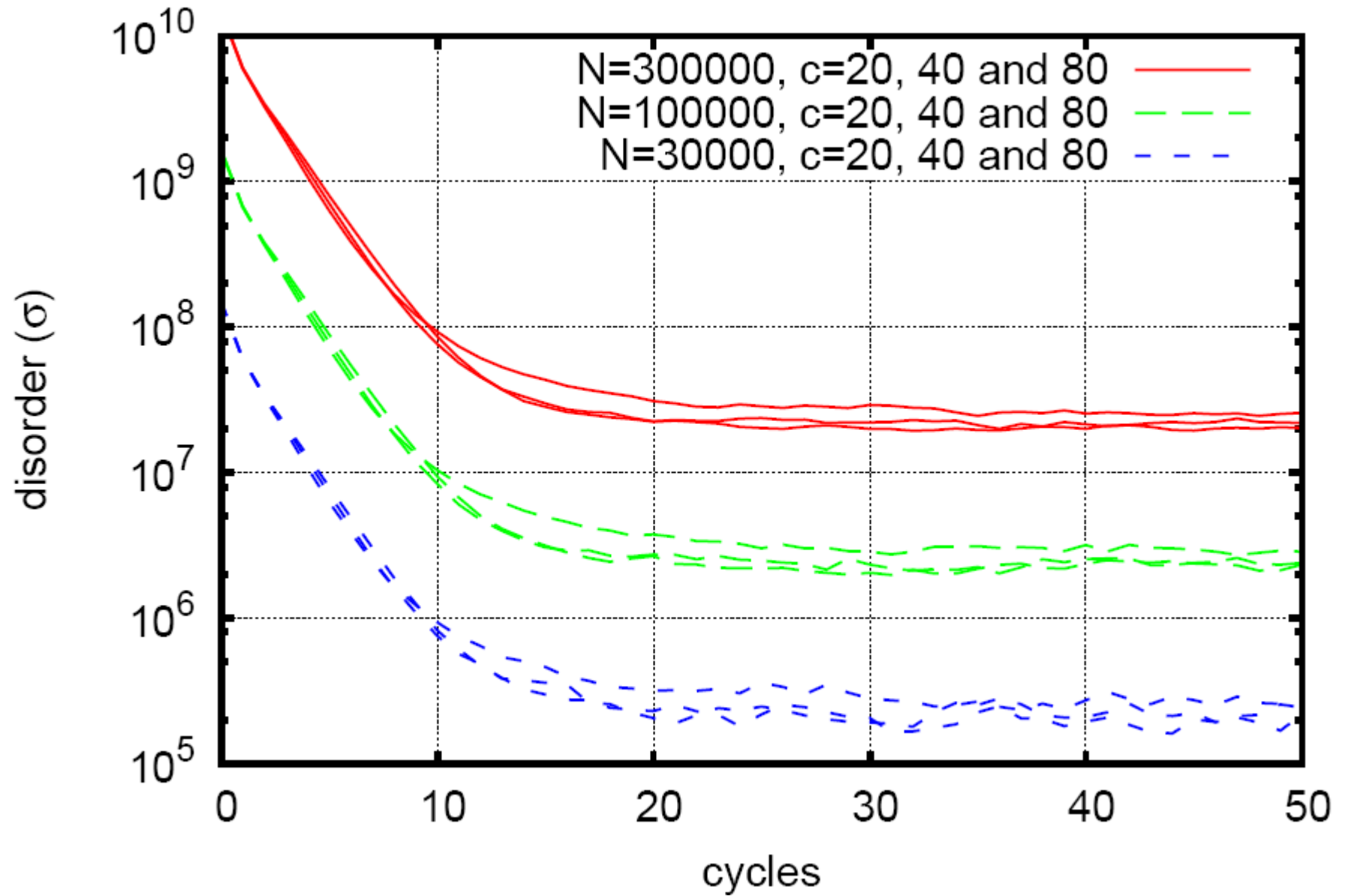






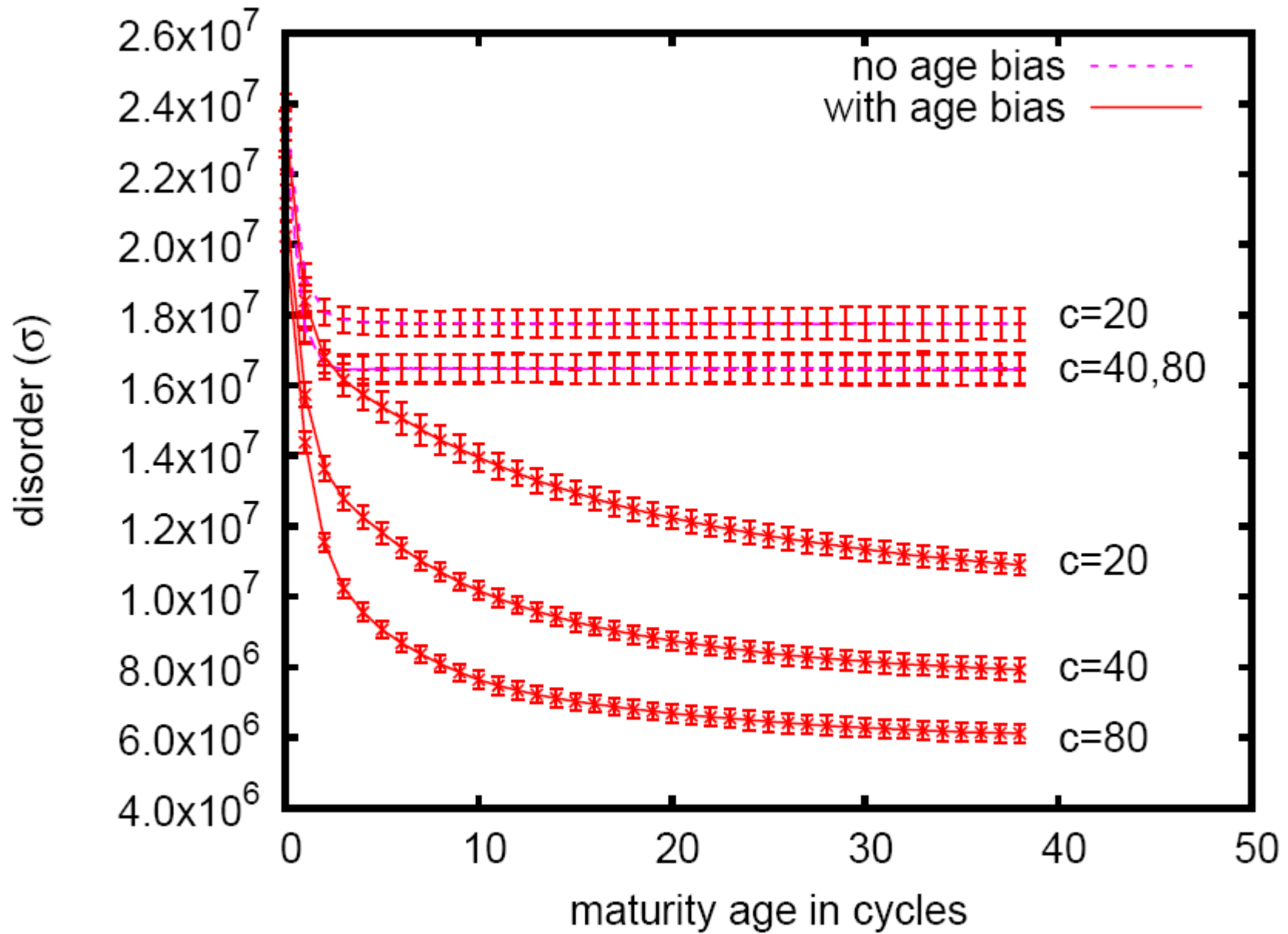


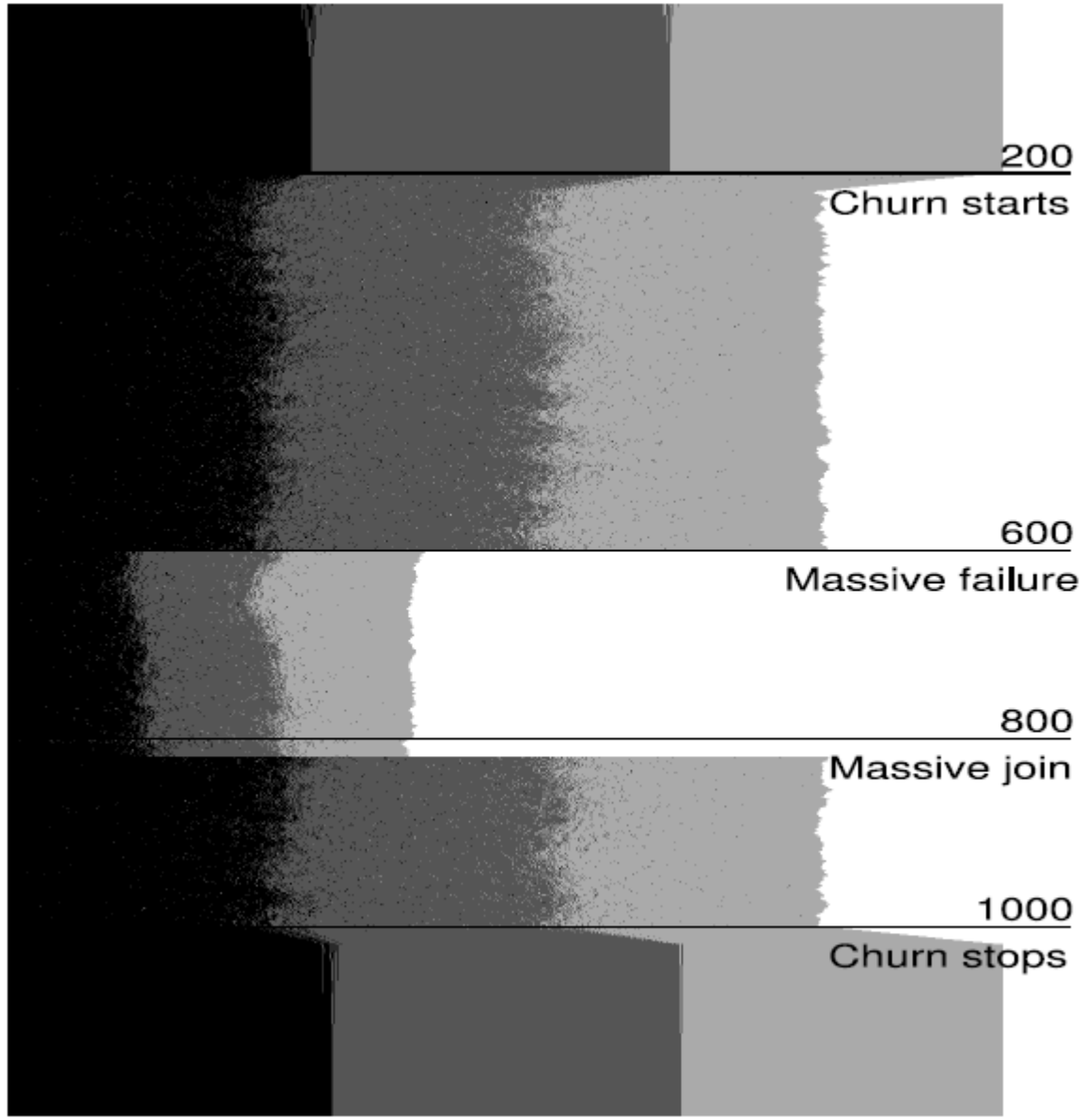
0.1% churn in each cycle



# Age techniques for churn

- We have seen that message drop is tolerated well
- Churn is more problematic
- We use age bias to improve churn performance
  - If there are more peers to swap with, we chose the one that is closest in age
  - Interesting analogies to society and learning





# Summary of slicing

- Simple and cheap
- Tolerates failure and massive joins/leaves
- Problems
  - Too small slices have a very high churn inside
  - If churn is not independent of resource amount, then slice sizes can be skewed
  - Currently works with one single resource dimension
- Published as M Jelasity and A.-M. Kermarrec, Ordered Slicing of Very Large-Scale Overlay Networks, *IEEE P2P 2006*.

# Summary and conclusions

- CloudMan is a promising direction
  - Targeted to better exploit existing and future massively large scale single-owner IP networks
  - We have many components already working (peer sampling, slicing, etc)
  - They have to be improved, glued together, and new protocols are needed as well
- Lots of possibility for future work!