Adaptive Data Propagation in Peer-to-Peer Systems

Thomas Repantis

trep@cs.ucr.edu

Overview

1. Problem
2. Solution
3. Simulation Results
4. Conclusion

Problem Definition

- How can we efficiently locate an object in an unstructured peer-to-peer system, when a reference to that object is given?
- Traditionally flooding, propagating query hop-by-hop, with many disadvantages:
  - Messages travel a large number of hops.
  - Waste processing power of many nodes.
  - Produce large amounts of network traffic.
  - Delay the answer.

Suggested Solutions

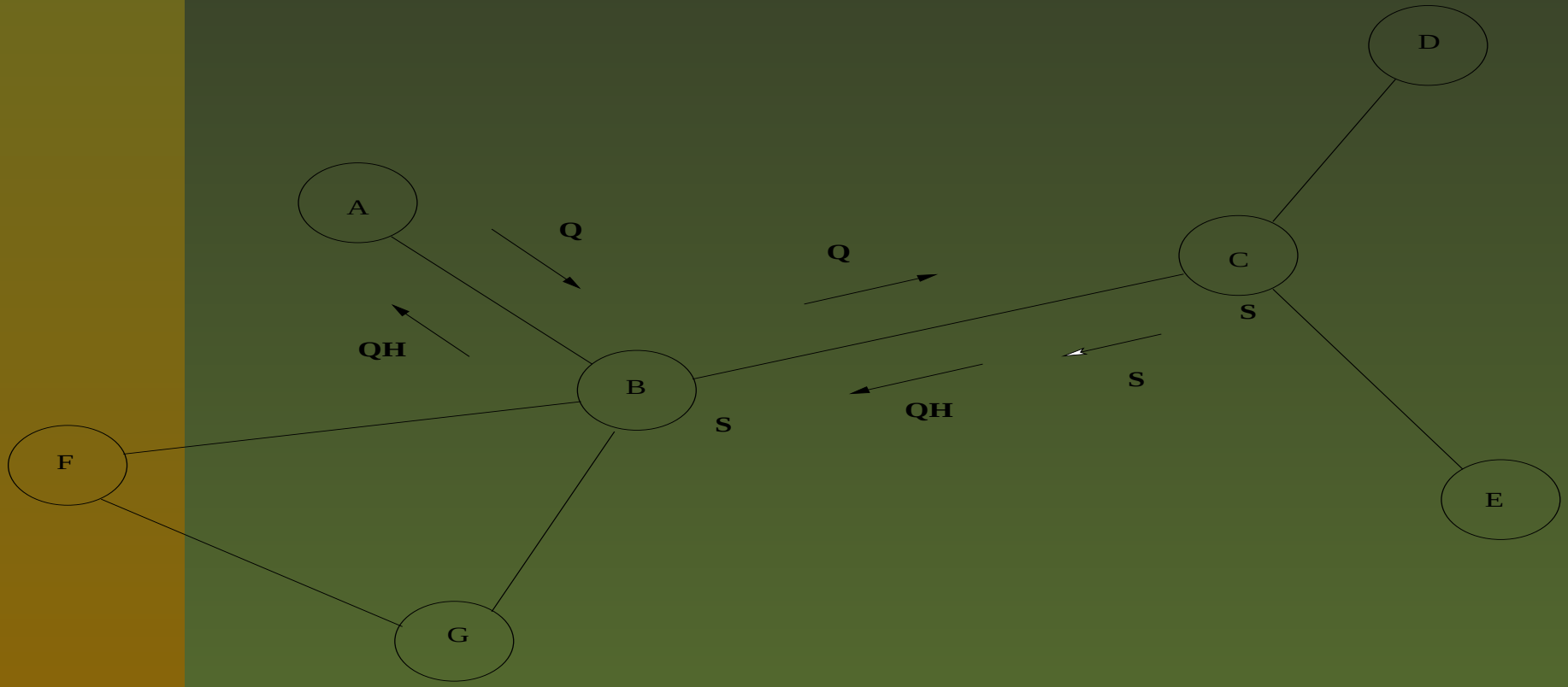
- Organize nodes according to their interests.
- Use Bloom filters to summarize data stored in nodes.
  - Each node examines the content synopses of its neighbors to decide were to propagate a query.

We suggest going even further

- Let us propagate the Content Synopses adaptively, according to parameters like:
  - Number of queries we have received from a peer.
  - Number of local hits the queries of a peer have produced.

System operation example

- According to its criteria, C propagates S only to B
- B based on S routes Q only to C
- QH is routed back to A

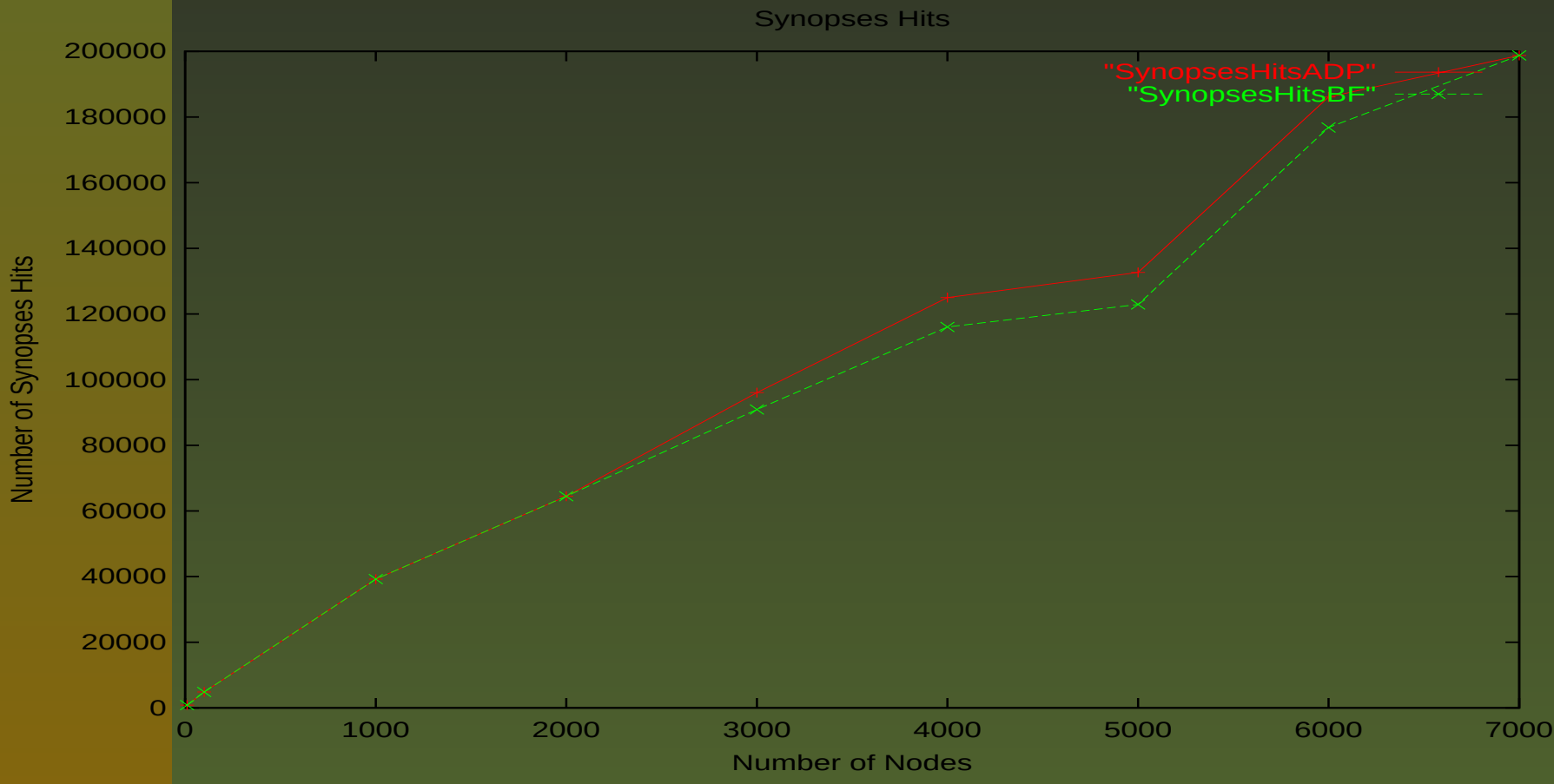


Simulation Parameters

- We implemented our protocols on top of the Neurogrid simulator
- Counting Bloom Filters, 4-bit counter, 10 bits length, 4 hash functions
- 300 possible Documents
- 400 possible Keywords
- 30 Documents per Node
- 1 Keyword per Document
- 50 Maximum Connections per Node
- TTL 7

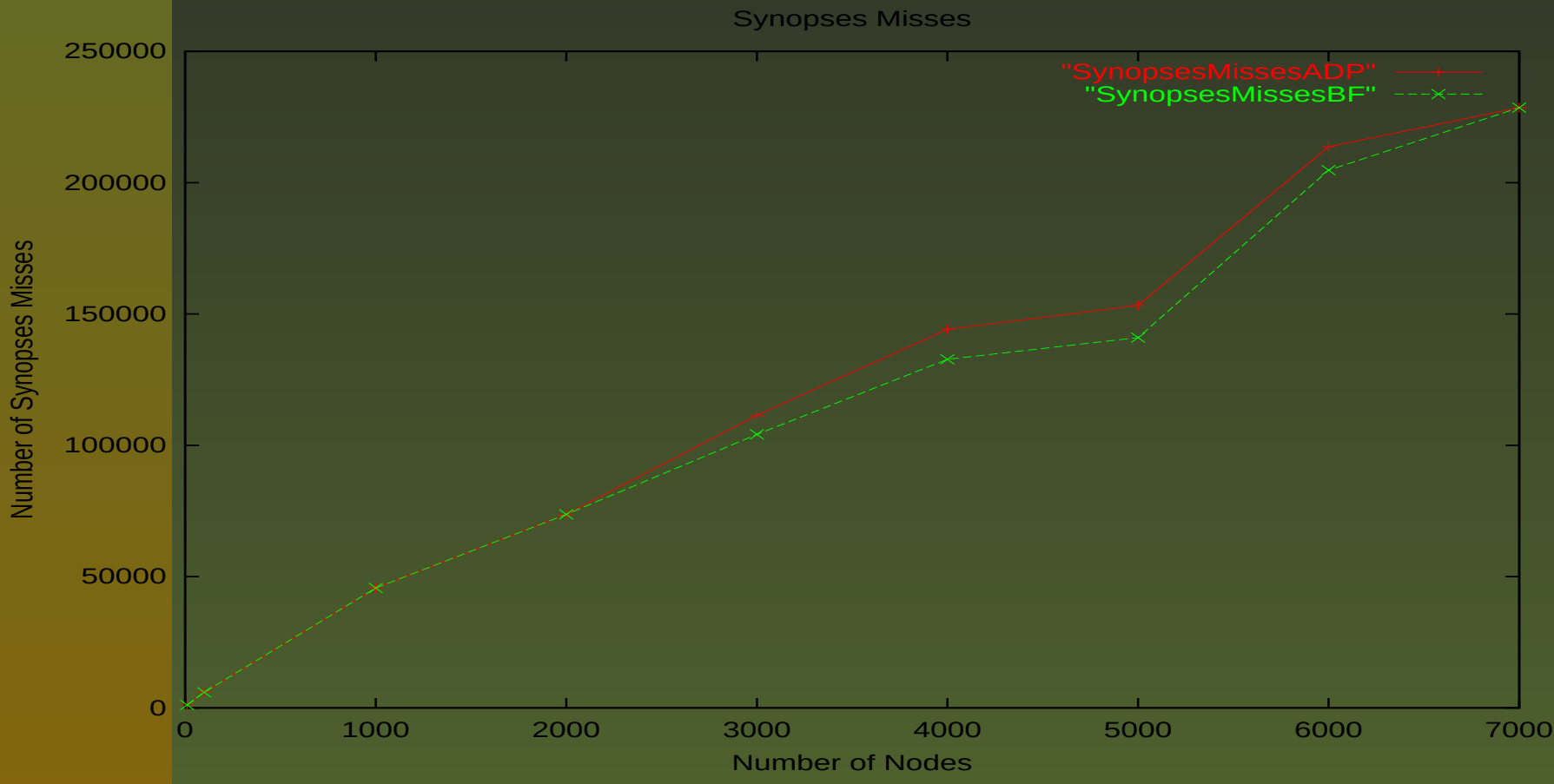
Synopses Hits

Queries found in neighbors' content synopses.



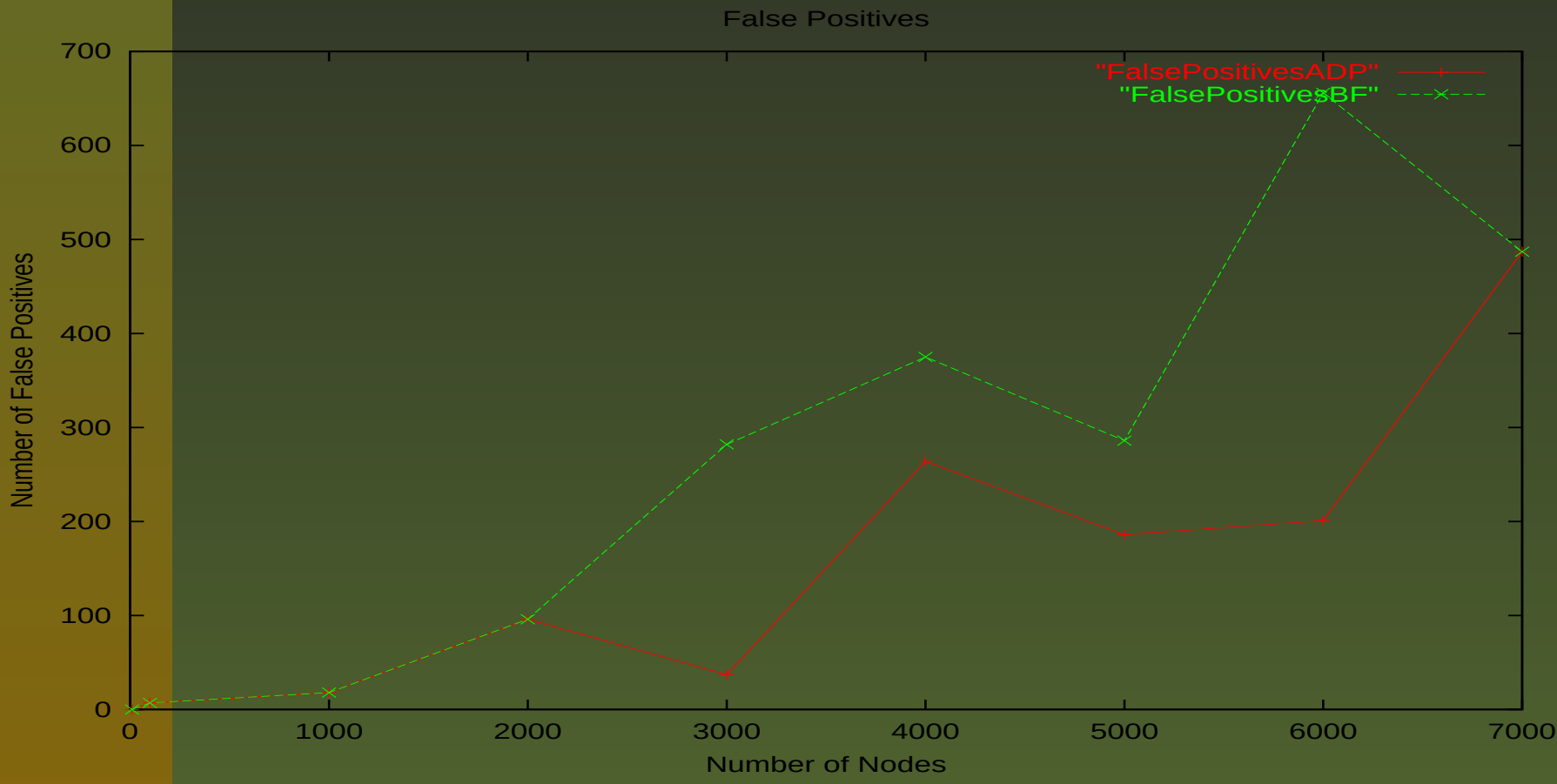
Synopses Misses

Queries not found in neighbors' content synopses.



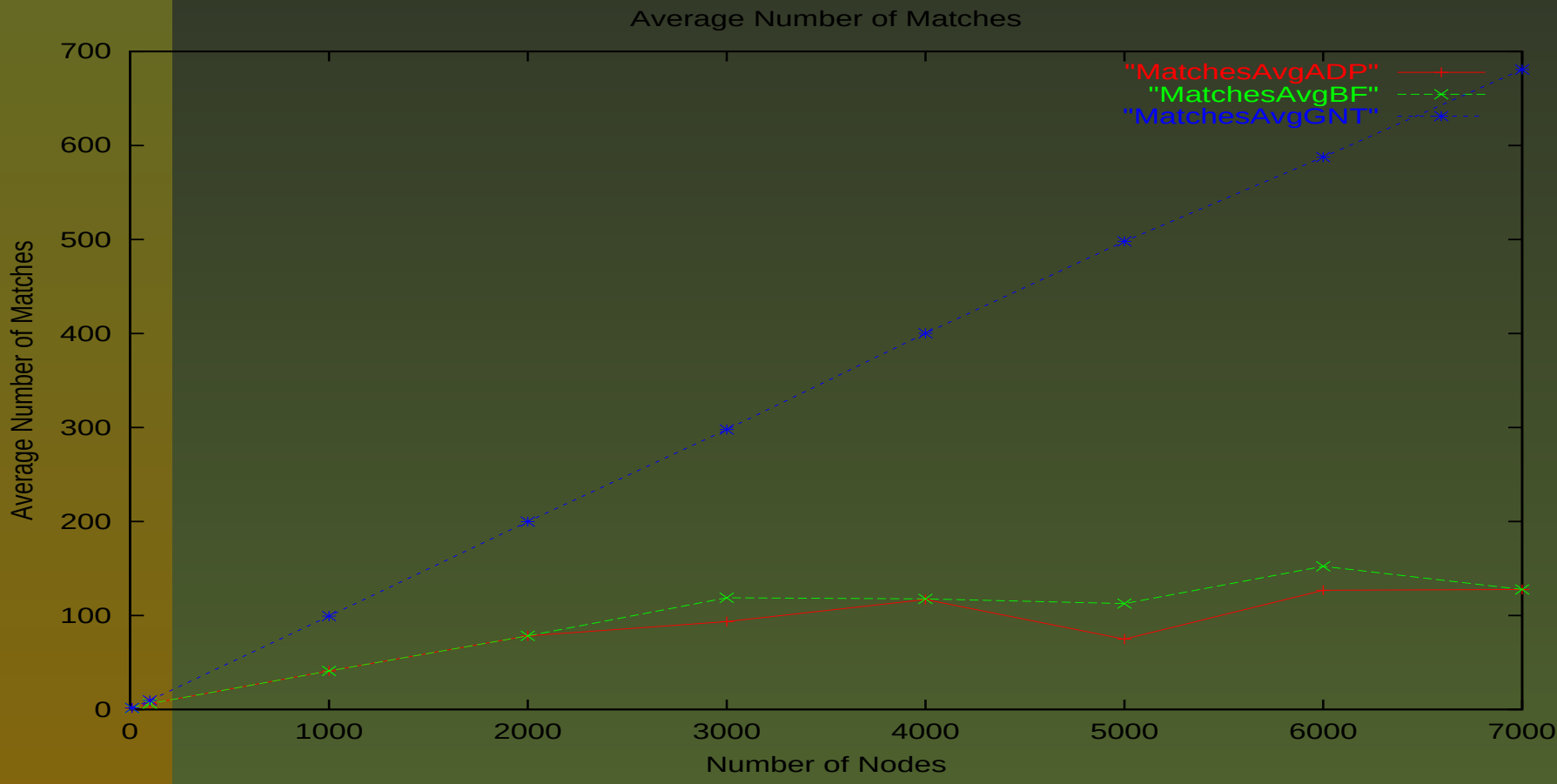
False Positives

Queries falsely propagated.



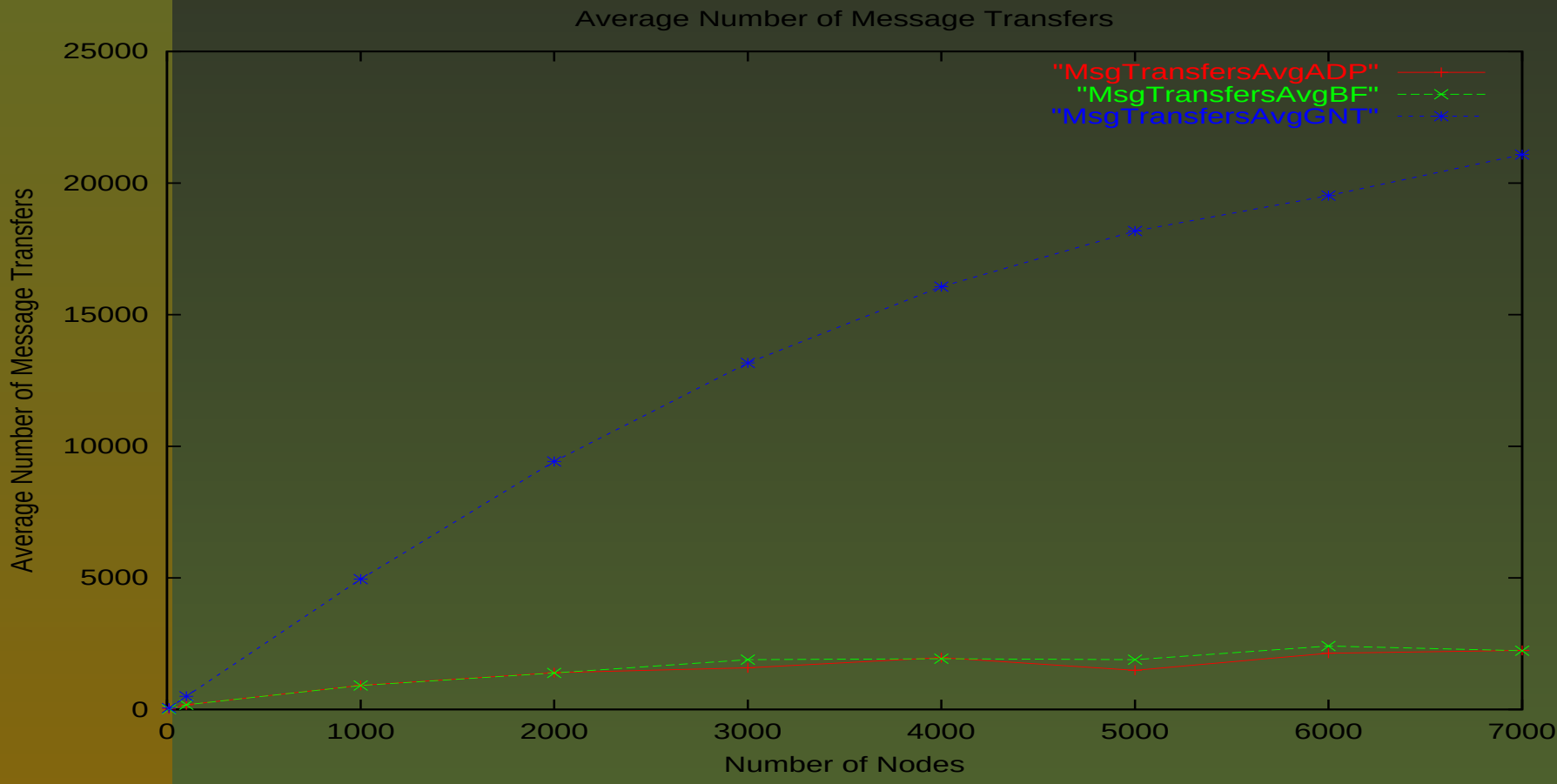
Average Number of Matches

Number of matching documents found for a search.



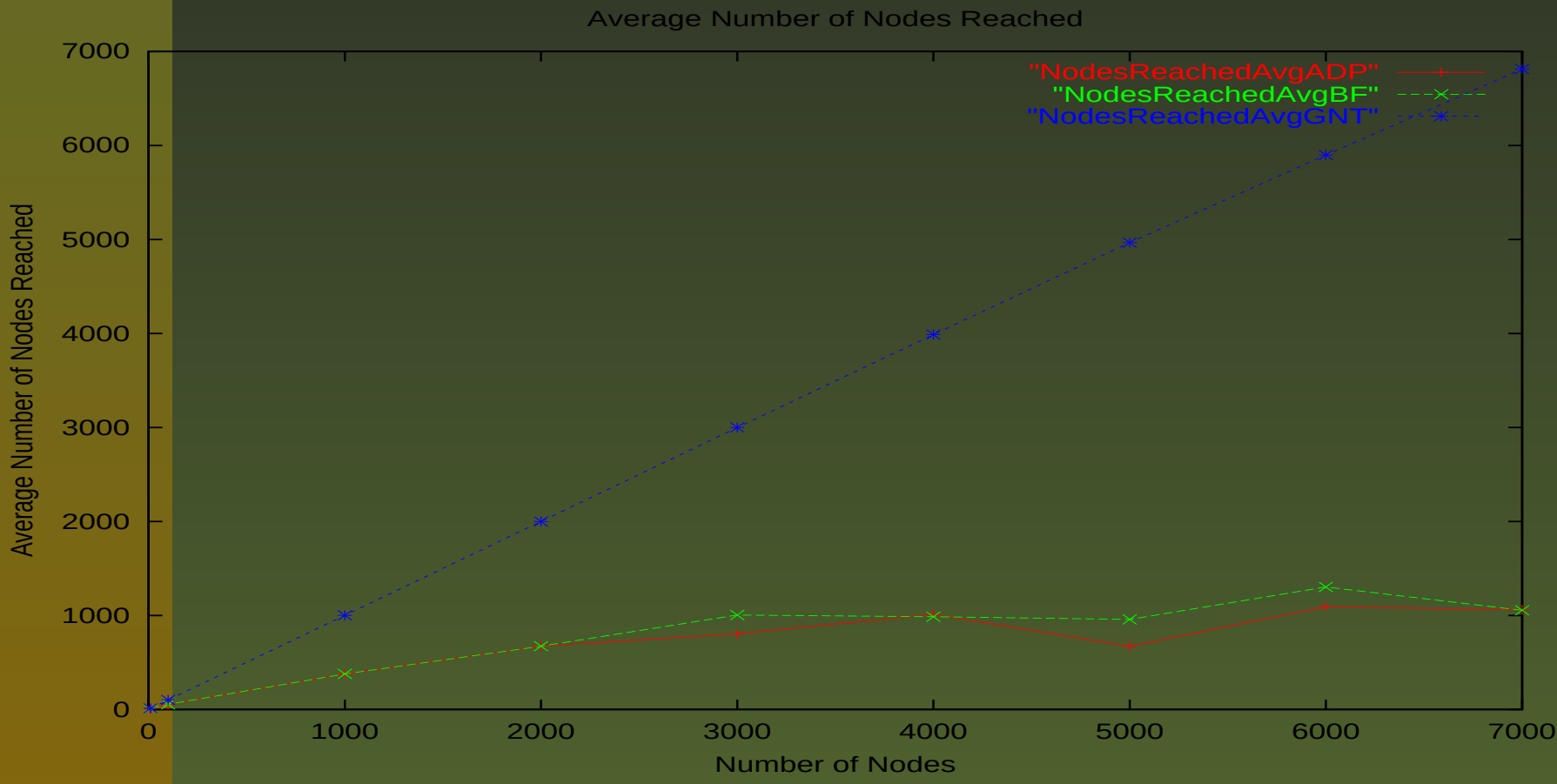
Average Number of Message Transfers

Number of messages sent during a search.



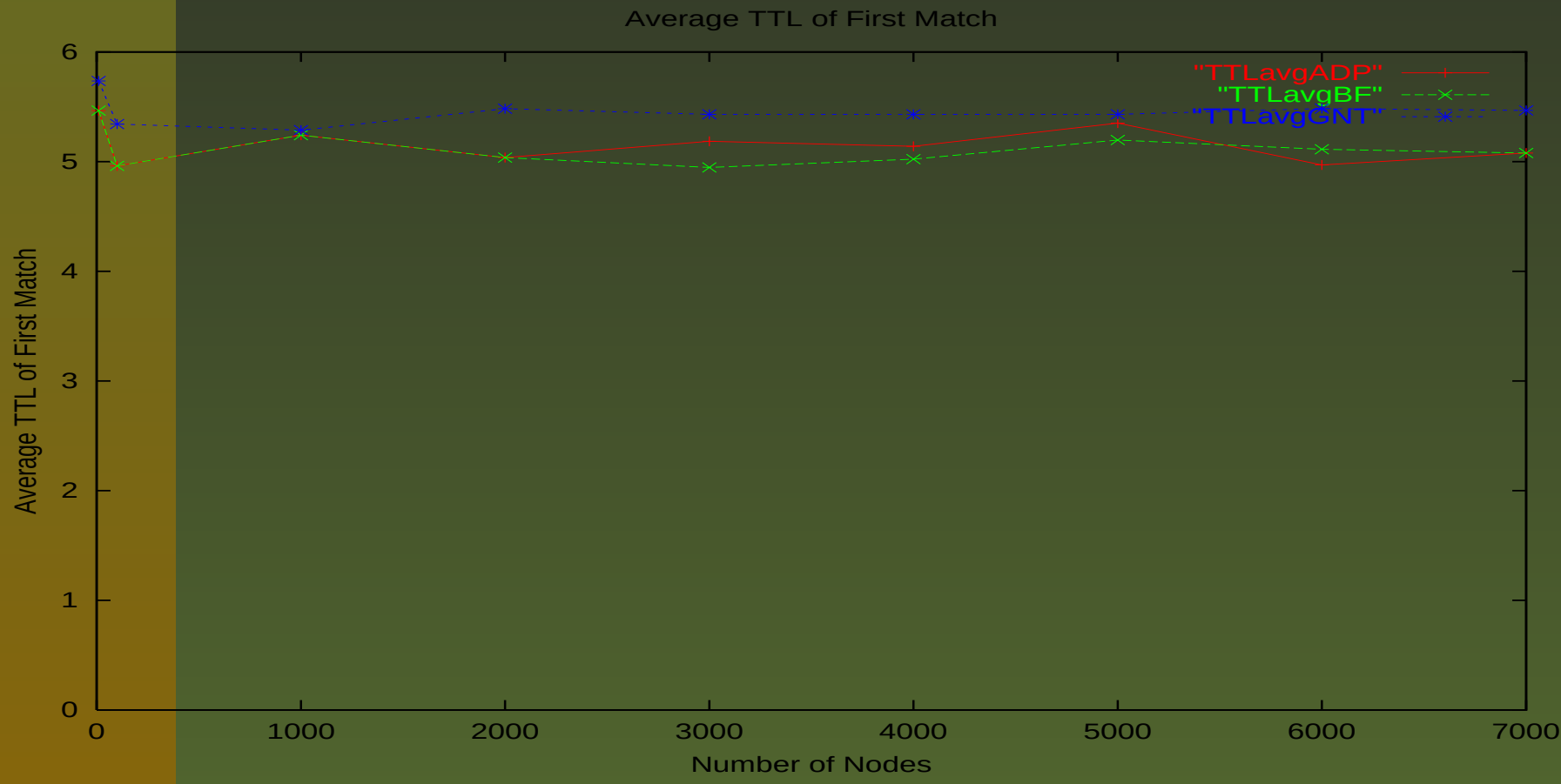
Average Number of Nodes Reached

Number of nodes reached during a search.



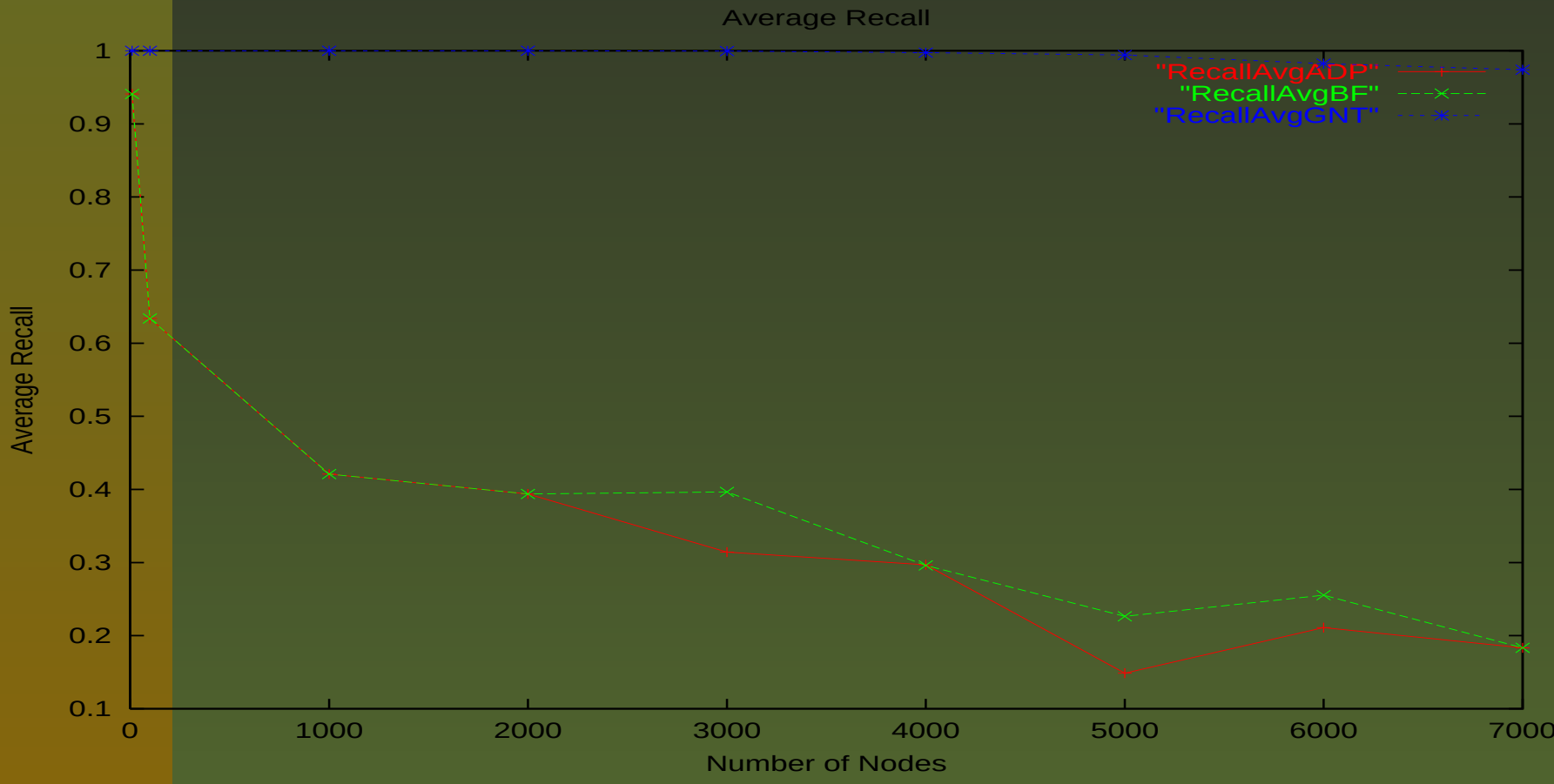
Average TTL of First Match

The TTL of the first message that found the first match (i.e. how many hops before the first hit).



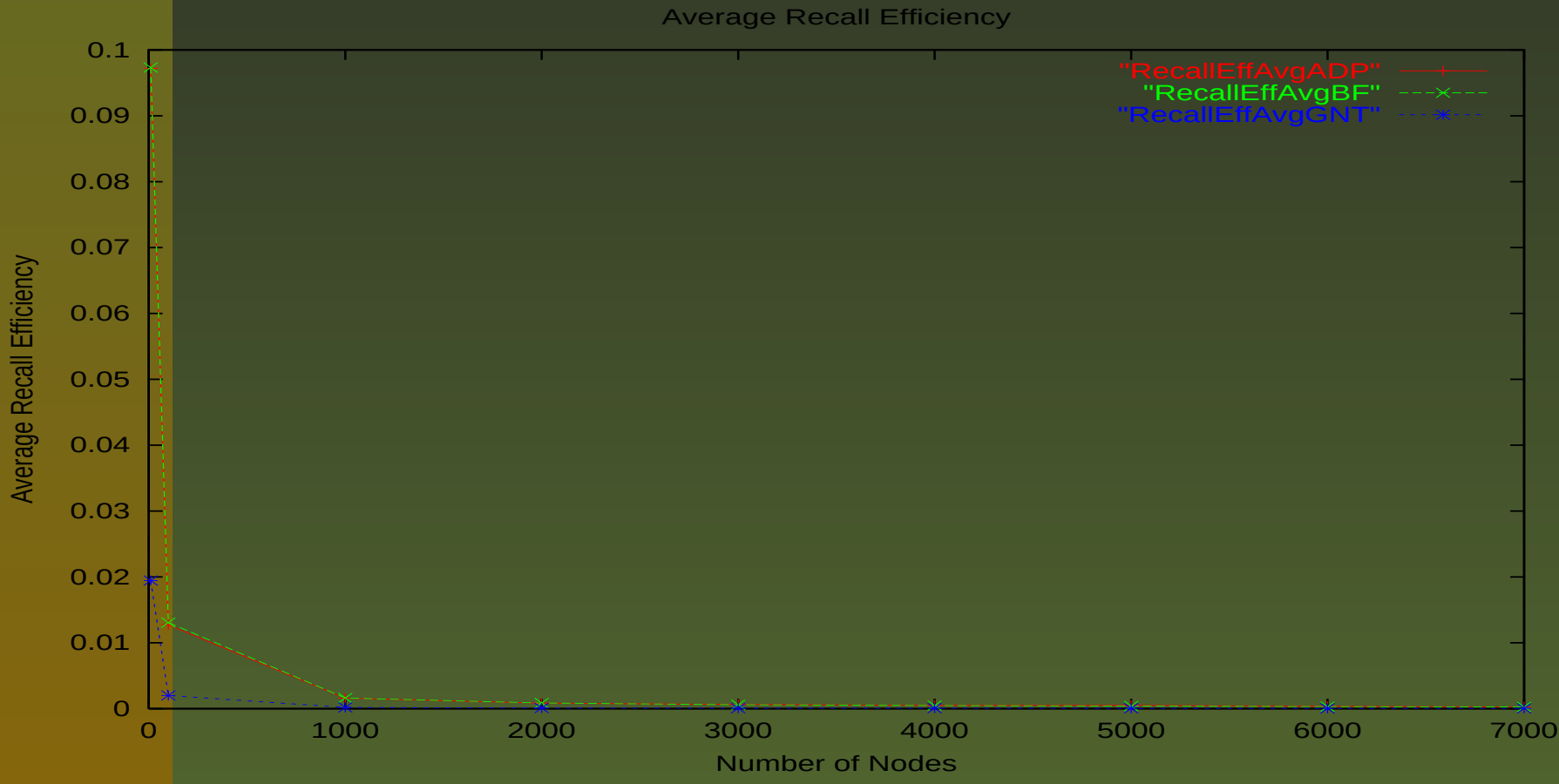
Average Recall

Proportion of all possible matches that was actually discovered.



Average Recall Efficiency

Average Recall divided by the number of messages transferred.



Conclusions

By propagating content synopses to peers that are selected adaptively we get:

- Faster search and retrieval.
- Less bandwidth wasted.
- Less processing power wasted.
- However, the recall of flooding is not reached.
- Room for future work!
  - Other parameters
  - Further propagation
  - Pulling instead of pushing

Thank you!

Questions/comments?