Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT

Liang Wang Jussi Kangasharju

Department of Computer Science, University of Helsinki, Finland

In Proceedings of IEEE International Conference on Peer-to-Peer Computing,
Trento, Italy, September 2013

Why do people measure?

System measurement is crucial in computer science in terms of following perspectives:

- Study system performance in real world.
- Fine-tune system parameters.
- Detect anomalies.
- Locate bottlenecks.
- Study user behaviors.

Why do people continuously measure?

Tons of measurement papers on P2P system already exist, why do people still continue measuring? Does it make sense?

- Different methods may lead to different “picture”.
- System evolves, more implementations.
- User behavior changes.
- Environment changes (e.g. government policy).
- Measurement methodology improves!

What are we doing here?

After BitTorrent has been actively studied for over a decade, what we are doing here?

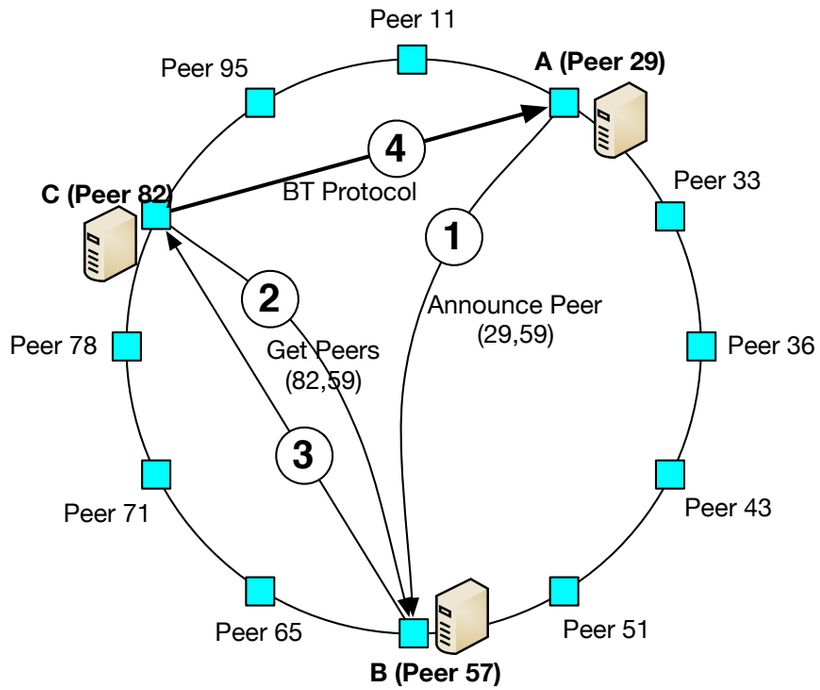
Yet another BitTorrent measurement paper?

- Identify a systematic error in previous measurement work which leads to over 40% estimate error!
- Propose a new methodology which generates more accurate result with less overheads!

BitTorrent Mainline DHT basics

Our measuring target is BitTorrent Mainline DHT (MLDHT).

MLDHT implements the minimum functionality of Kademlia, only 4 control messages.



Node has 160-bit ID. Nodes in n-bit zone share the same prefix of n bits in their IDs.

Example: two nodes from 6-bit zone →

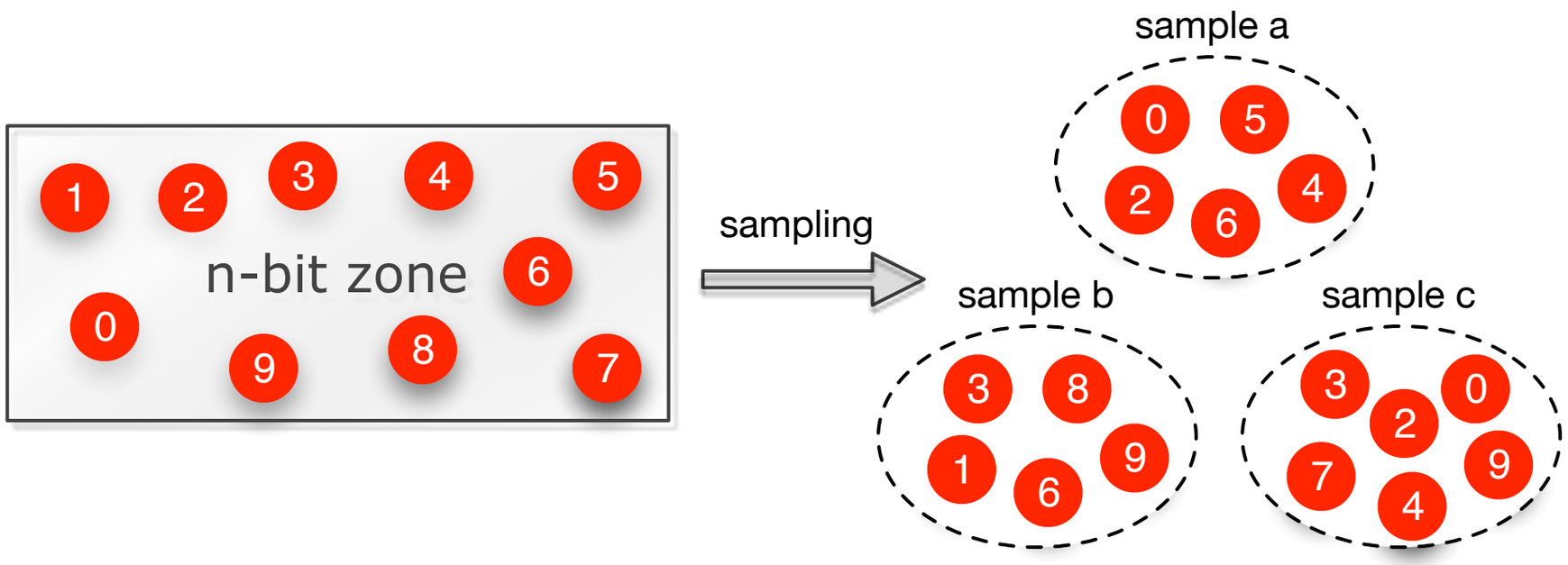
ID1: 001011110110101……010
ID2: 001011001110101……000

A seemingly trivial common technique

A common “technique”:

Take a sample from system, use it as zone density, scale up!

So what’s wrong here?



Missing node issue refers to the problem that a crawler systematically misses some nodes when it is crawling a target zone.

Should we recheck previous work?

Lot of previous work based their claims on network size.

- Qualitatively correct, no big problem.
- Quantitatively, suspicious!

What if this fundamental number – network size, is wrong?

Furthermore, how to answer the following questions:

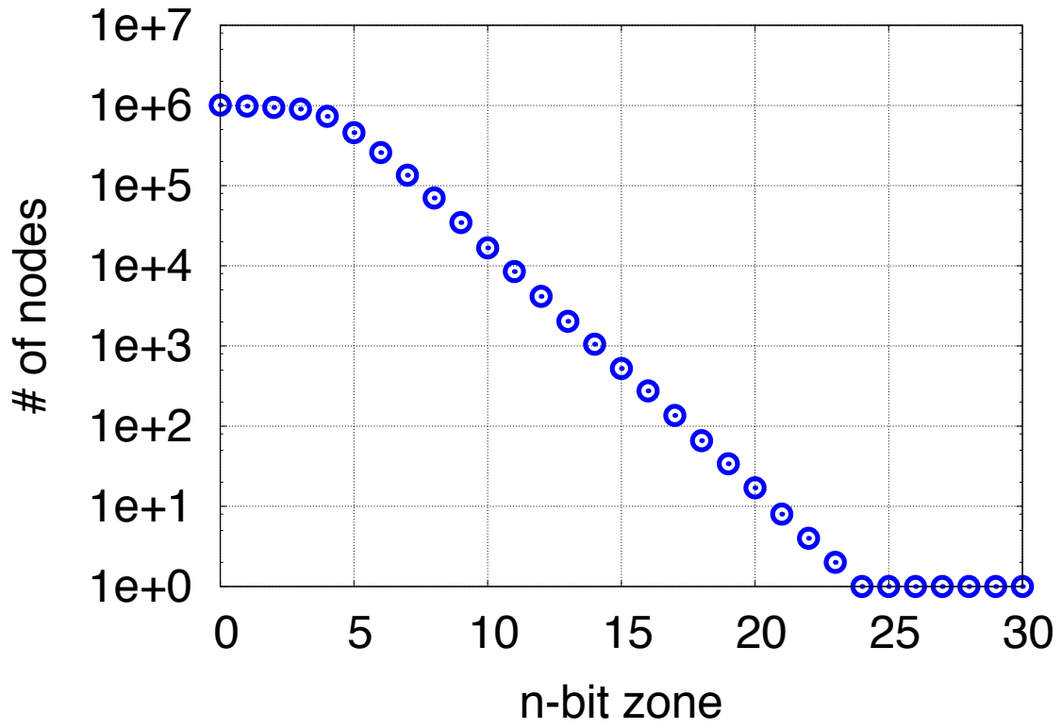
- What is the estimate error?
- How to reduce the estimate error?

Why missing node was overlooked?

All measurements have errors.

In MLDHT, they come from missing node issue.

Why it was overlooked by previous work?

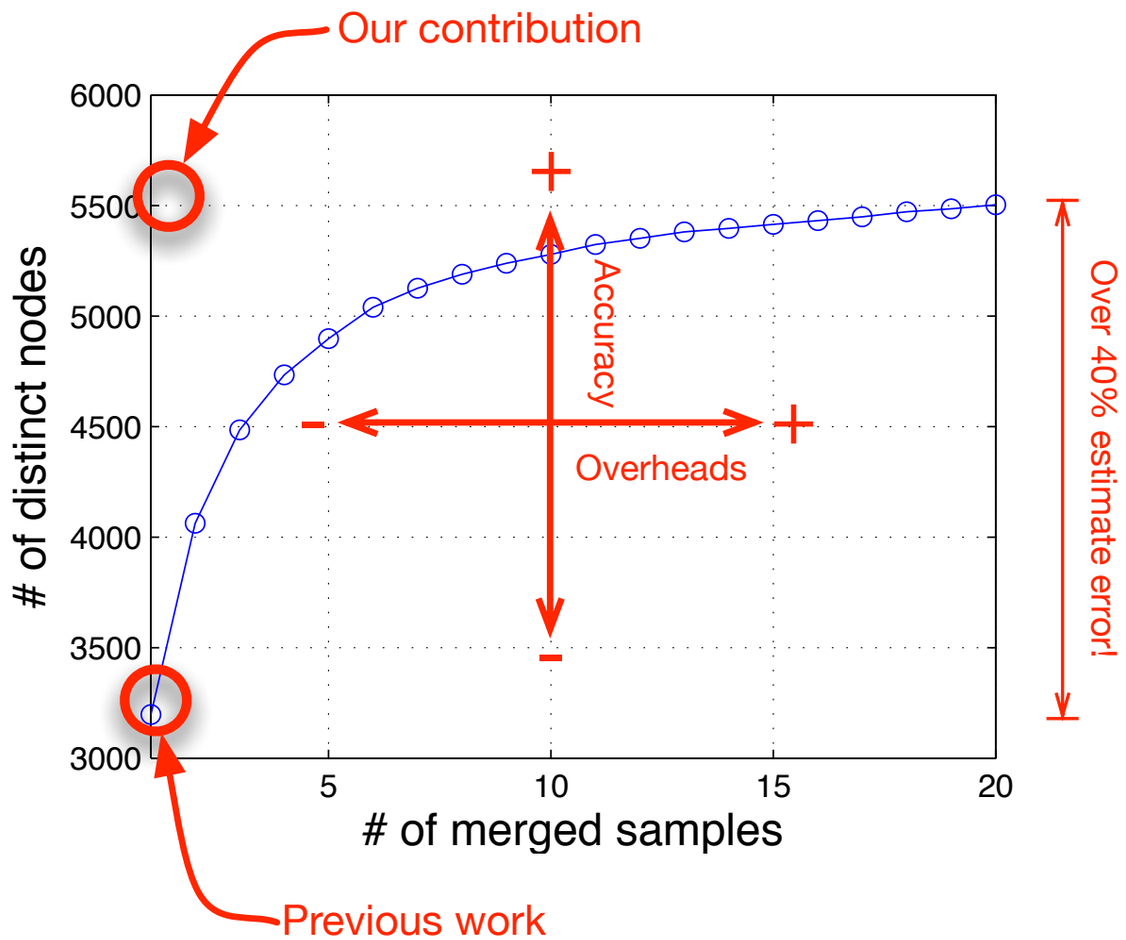


- Overly optimistic and ideal assumptions on MLDHT protocol in a imperfect world.
- Node density in different n-bit zones. A deceivable figure, very stable behavior between 5 ~ 23-bit zone.
- Can we safely scale up zone density?

Estimate error due to the missing node issue will be magnified exponentially after simple scaling up!

A straightforward but strong evidence

An experiment with 20 simultaneous samples were taken from 12-bit zone in MLDHT. Each contains 3000 ~ 3500 nodes.



# of distinct nodes increases as we merge more and more samples. The increase rate decays fast.

Each sample only covers part of the zone.

Estimate error originates from missing node issue!

Can simultaneous samples save us?

Simultaneous samples definitely improve accuracy, and drive our estimate closer to the real value. However:

- Taking simultaneous samples is expensive!
- We still don’t know the estimate error?
- If we don’t know how far we are to the real value, then how many simultaneous sample we should take?

How to fix the problem?

Model the measurement as a Bernoulli process.

- Conceptually, a crawler goes through all the nodes in a target zone one by one.
- Each node has two options: being selected or being ignored.
- What is the probability (p) of being selected?
- Correction factor is defined as 1/p.

An analogy for system measurement

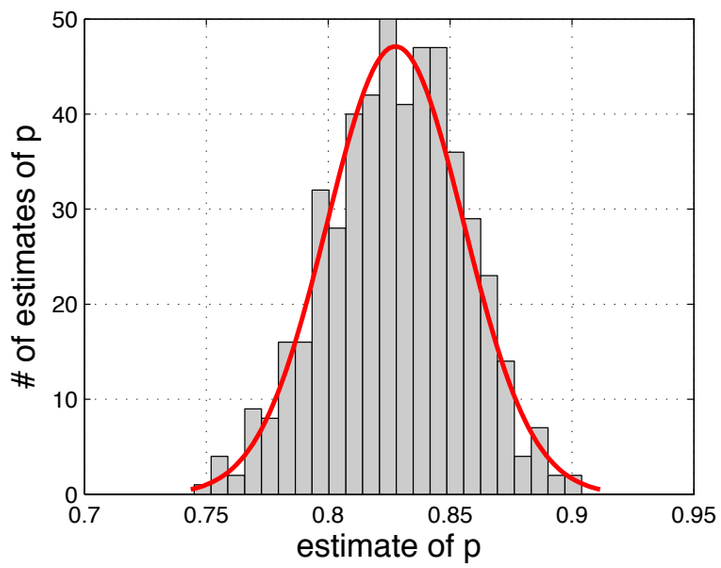
Assume we have a box of black marbles, and we are only allowed to take a “small” sample from it. We don’t know what fraction the sample is of the whole population. How to estimate the number of marbles in the box?

- Mix certain amount of white marbles into the box.
- Make sure they are well mixed, then take a sample.
- Count how many white marbles in the sample.
- Calculate p value.
- Inferring the number of black balls is trivial.

How do we measure in practice?

Aim: fast, accurate, low overhead, minimum impact on system

- Choose a random zone.
- Insert "controlled" nodes into the zone (less than 1%).
- Take a sample from the zone, count the "controlled" nodes and estimate p.
- Repeat the experiments until reaching satisfying variance.



An experiment where 500 estimates of p were calculated. The samples were taken from 12-bit zone, and the estimates passed Jarque-Bera test.

Correction factor – summary

- Correction factor is not a universal constant, but a simple methodology.
- Lead to more accurate result with explicit estimate error.
- Lower the requirements on crawler design (performance, etc.).
- Make results from different crawlers consistent.
- In a given environment, correction factor is very stable.
- On the actual measurement platform, moving average is a good option and only needs periodic calibration.
- No need to choose a specific zone, but recommend (5~20)-bit zone.

Validation of Methodology

Step-by-step validation

We validated each step of our methodology from the assumptions to the final outcome.

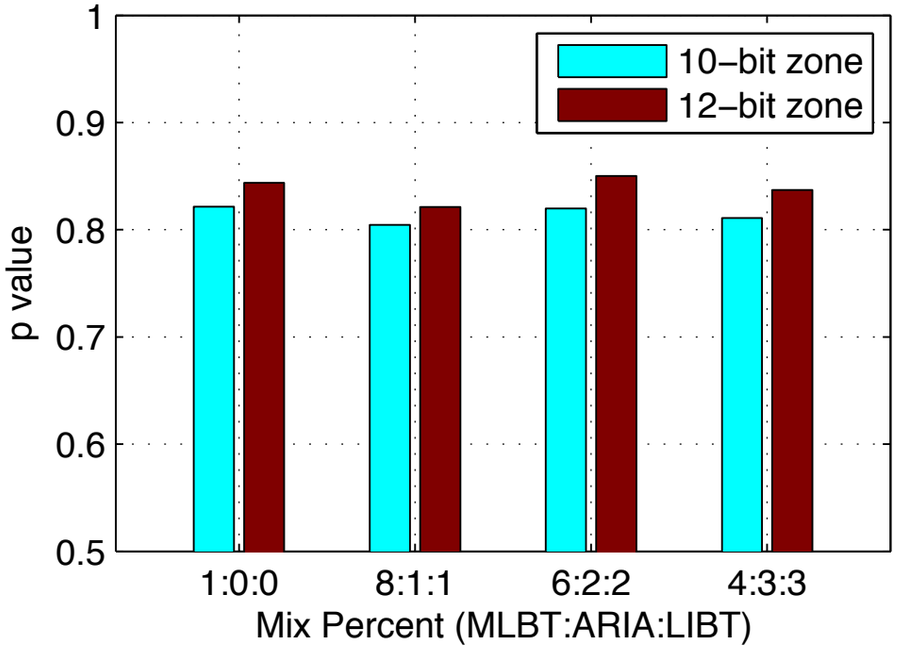
- Uniform node density: We carefully examined a large set of samples (over 32000) from the different parts.

- Missing node: We identified and verified this by “merging samples” and “injecting controlled nodes” experiments.

- Validity of correction factor: We designed large-scale emulation in controlled environment.

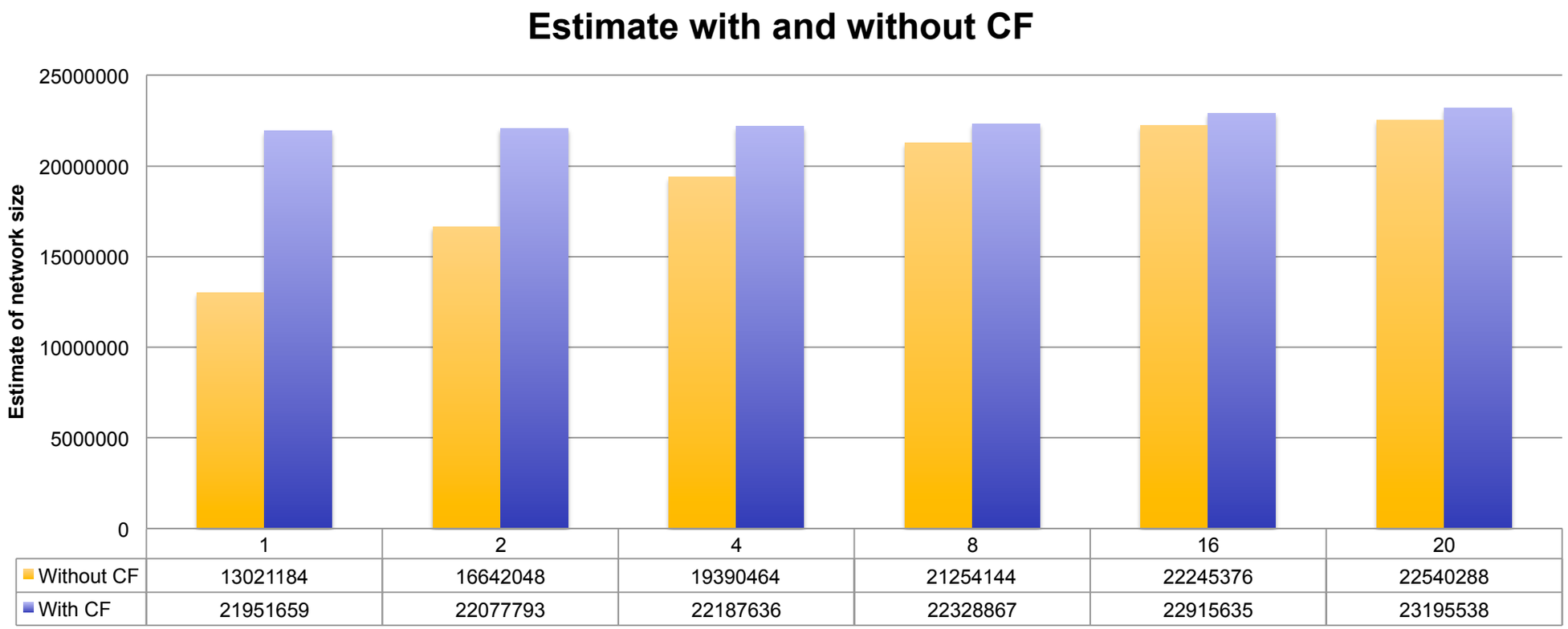
Validation in controlled environment

- A high performance computing cluster of 240 nodes was used to deploy a million-peer emulation.
- Different mixes of three implementations to mimic a realistic ecosystem.
- Small variation, estimates are consistent after applying correction factor.



App Percent (%)			10-bit			12-bit		
MLBT	ARIA	LIBT	p	stdev	Corr. Factor	p	stdev	Corr. Factor
100	0	0	0.8215	0.0173	(1.1681,1.2708)	0.8437	0.0263	(1.1157,1.2641)
80	10	10	0.8045	0.0164	(1.1943,1.2958)	0.8213	0.0291	(1.1370,1.3104)
60	20	20	0.8198	0.0211	(1.1601,1.2860)	0.8501	0.0243	(1.1127,1.2477)
40	30	30	0.8109	0.0190	(1.1780,1.2938)	0.8372	0.0257	(1.1254,1.2726)

Validation in open environment



	1	2	4	8	16	20
Without CF	13021184	16642048	19390464	21254144	22245376	22540288
With CF	21951659	22077793	22187636	22328867	22915635	23195538

Simultaneous samples	**1**	**2**	**4**	**8**	**16**	**20**
Error	40.68%	24.62%	12.61%	4.81%	2.92%	2.82%

- In open environment, we don't know the true network size, but we can validate effectiveness of CF by comparing how two methods converge.
- Small number of simultaneous (without CF) samples leads to big error.
- CF effectively reduces the error even with small amount of samples.

Crawler impacts the result

We chose libtorrent as an example because it has been used in several projects as a crawler.

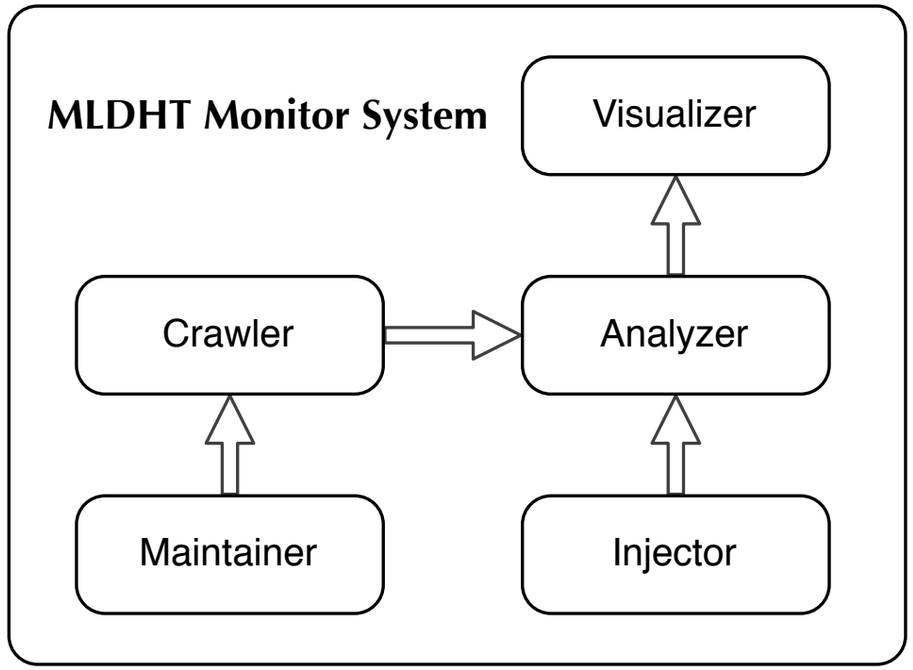
- Without tweaking, libtorrent only reports 1/6 of the actual network size.

- After tweaking, results are improved. But still, only 1/4 to 1/3 of the actual nodes are reported.

- Lots of things can impact the result: queue size, bucket size, mechanism like banning suspicious nodes, abandon malformed messages, routing table operations, etc.

Actual measurement platform

Five principal components



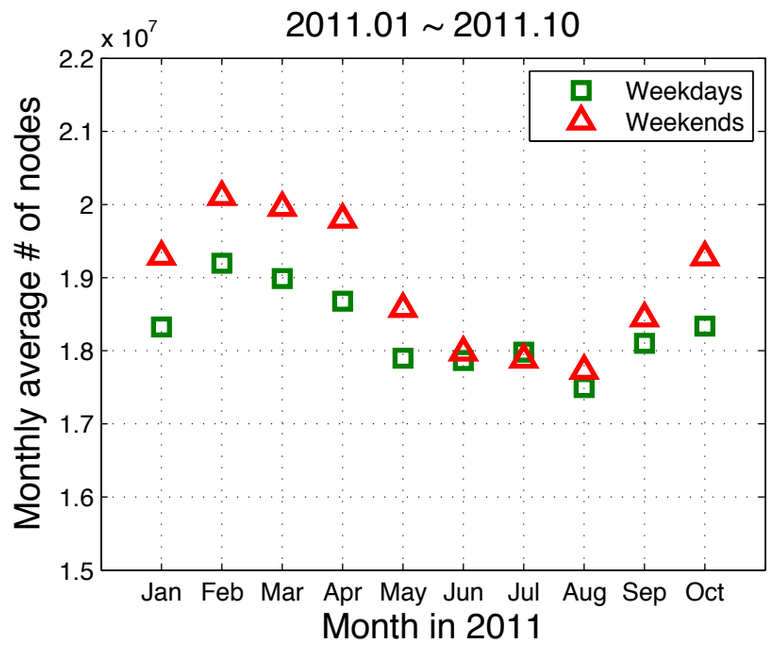
- Visualizer: visualizes data.
- Maintainer: maintains several hundreds active nodes for bootstrapping a crawl.
- Injector: injects “colored” nodes into the target zone, and replaces them periodically.
- Crawler: takes a sample from a target zone.
- Analyzer: counts the “controlled” nodes and calculates p.

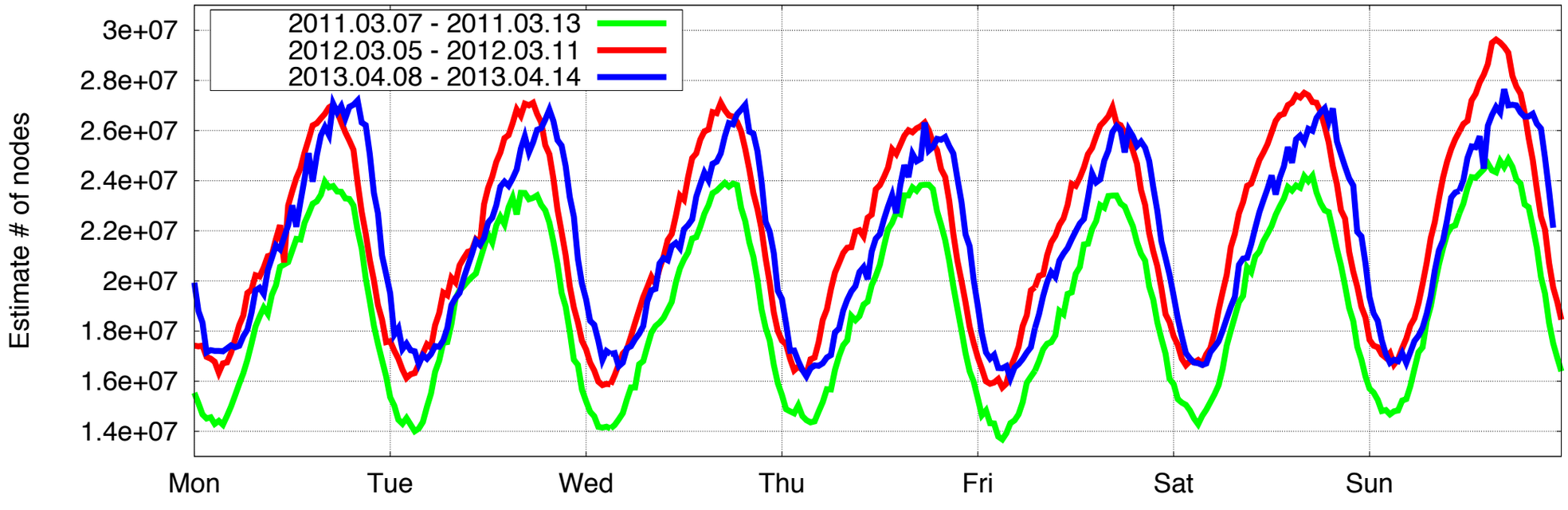
Actual measurement deployment

- Measurement platform started on December 17th, 2010.
- Two physical nodes with one crawler on each, to prevent sample gap due to software/hardware failures.
- Two crawlers use different sampling policies, to cross-correlate with each other.
- Sampling frequency was every 30 minutes, and increased to every 10 minutes since 2013.
- Over 32000 samples were collected.

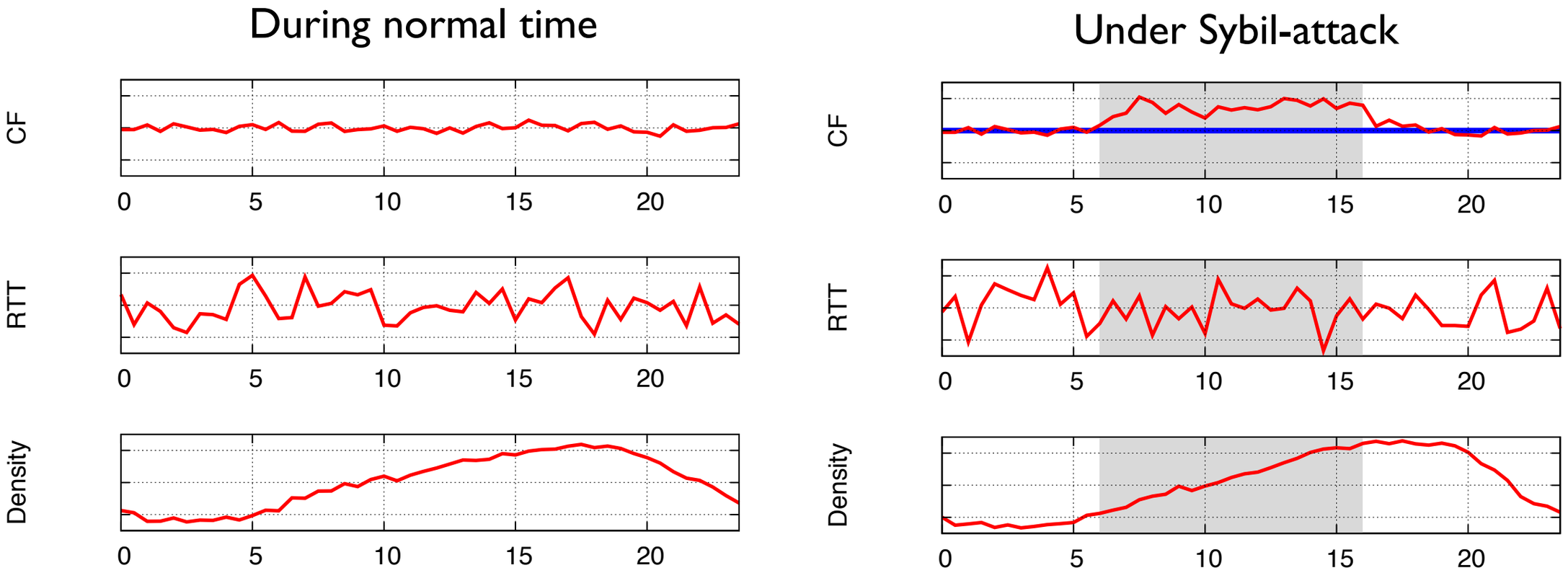
Measurement – system evolution

- P2P is not dying. 10% increase from 2011 to 2012, then remains stable.
- Some countries had an increase, some had a drop.
- Strong diurnal and seasonal pattern still exists.





Measurement - anomaly detection



A real-world Sybil-attack in MLDHT on Jan. 6, 2011. The attack was from two virtual machines of Amazon EC2, and started from 6:00 am.

For details, refer to:

Liang Wang; Kangasharju, J., "Real-world sybil attacks in BitTorrent Mainline DHT," GLOBECOM, 2012

Conclusion

- Correction factor is not only a measurement tool, but also a technique to help current and future measurement.

- Make results from different measurement tools consistent and with explicit measurement error.

- Lower the requirements on crawlers, make crawling more efficient with less overheads.

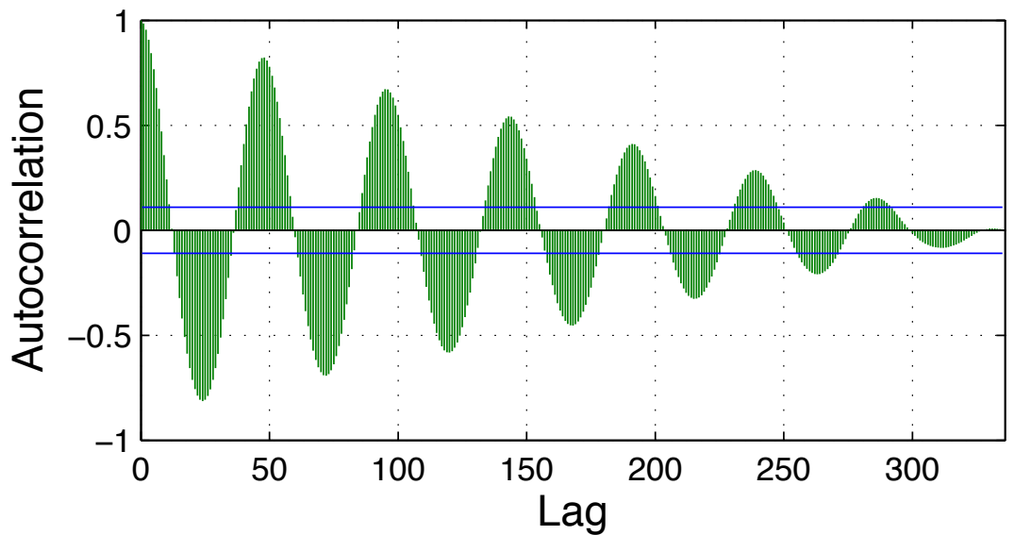
- Some other interesting applications like anomaly detection.

Thank you!

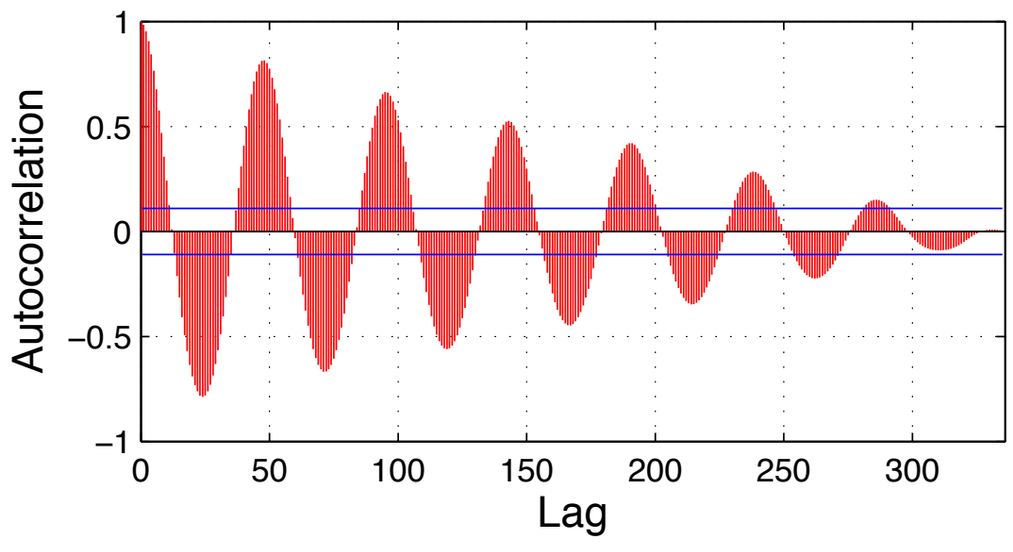
Questions?

Measurement – user behavior

10% doesn't necessarily imply user behavior has changed.

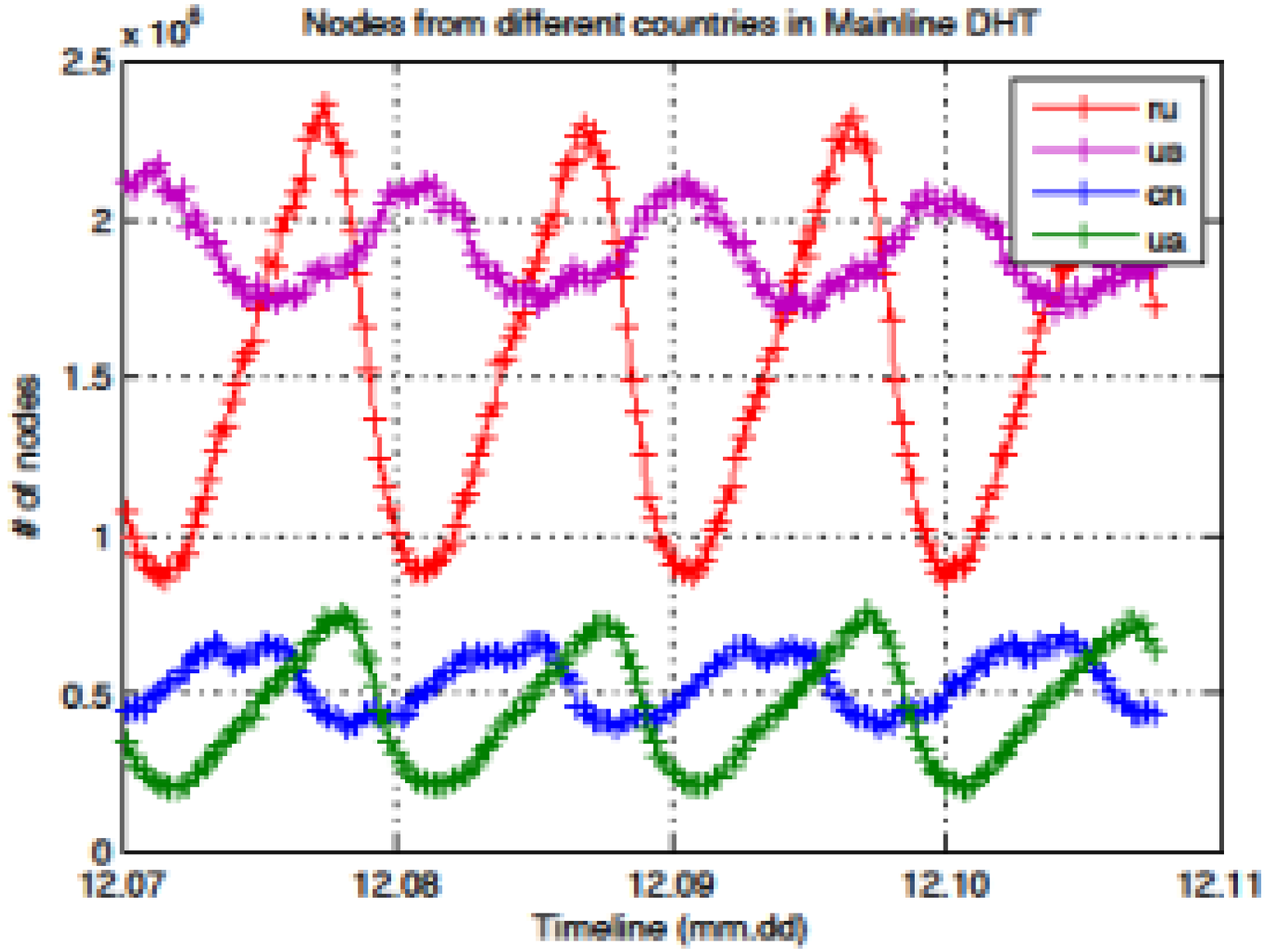


- We calculate autocorrelation function over the samples from 2011 (green) and 2012 (red).



- After the noise was filtered out, system exhibits almost exactly the same behavior over different times.

Measurement – real world event



Causes for missing node issue

The possible causes for missing node issue can be manifold:

- MLDHT protocol's inherent problem.
- Network dynamics: nodes join/leave. This problem becomes severer if crawling time is long.
- Lost messages due to network congestions.
- Some protect mechanisms: e.g. blacklist, banning suspicious node, dropping malformed message, small bucket size, small queue size.
- Stale or false information in the routing table, mainly due to implementation issues.
- Crawler is not efficient enough in terms of greediness and speed.
- Firewall issue.