Efficient DHT attack mitigation through peers' ID distribution

Thibault Cholez, Isabelle Chrisment and Olivier Festor
{thibault.cholez, isabelle.chrisment, olivier.festor}@loria.fr

LORIA - Campus Scientifique - BP 239 - 54506 Vandoeuvre-les-Nancy Cedex

April 23rd 2010

Outline

Outline

Background on KAD

KAD is :

- A fully distributed P2P network (Kademlia DHT)
- Used for file sharing
- Implemented by open source clients (eMule and aMule)
- Widely deployed (∼3 millions simultaneous users)

KAD DHT used to index keywords & files :

- KAD ID : place of a peer in the DHT (128 random bits)
- target (content) ID : MD5(keyword) or MD5(file)
- prefix = number of common bits between a peer & a content

Type	ID	prefix
target ID	4772212658290 86C74988C40EFE63DAF	-
peer ID	477229E3D7CFC729F337ABBB69C983C6	20 bits

The KAD DHT

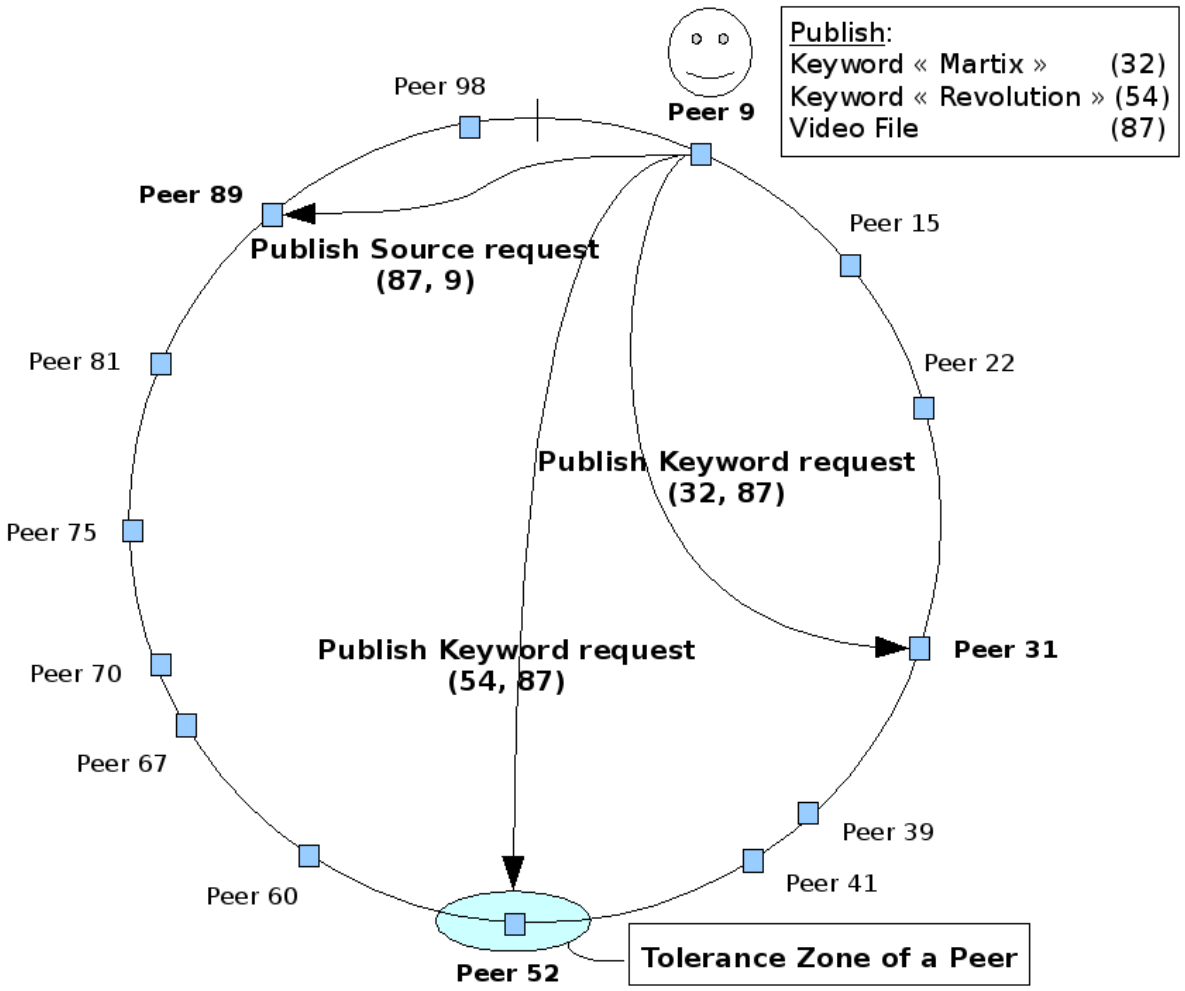


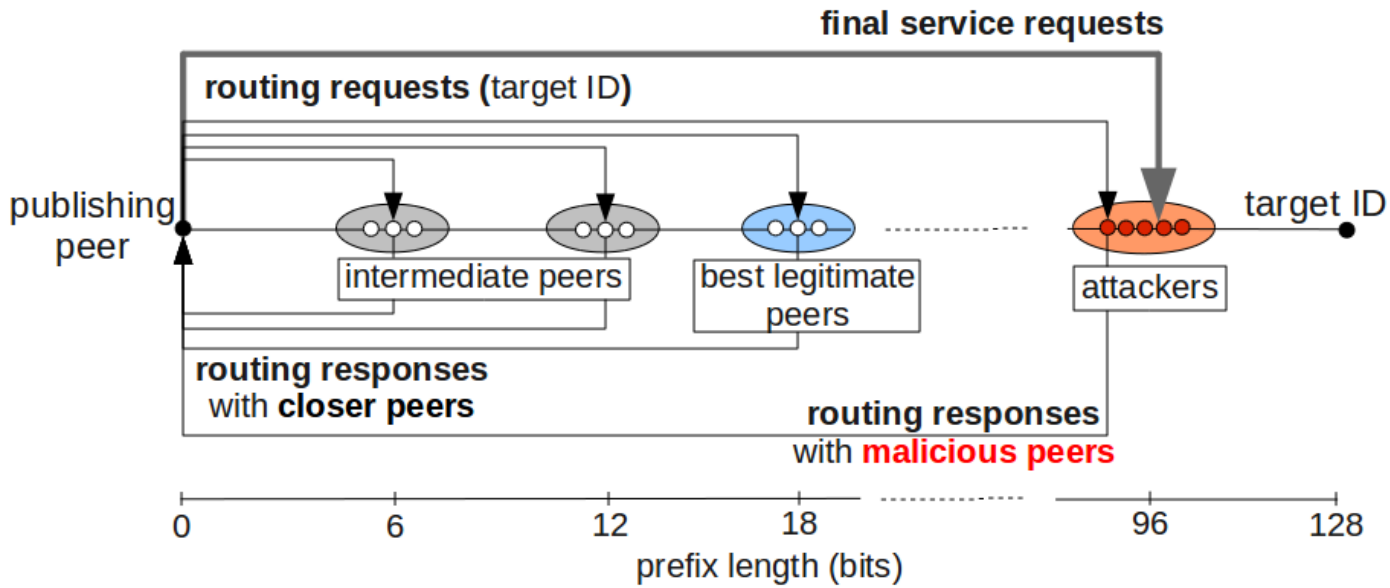
Fig.: Double indexation mechanism used to publish contents

Exploiting KAD Search

Despite recent protective rules, localized attacks are possible :

- Each peer is free to chose its KADID
- Very efficient KAD Search procedure "store to the closest peers possible"

- Place few distributed peers close to the targetID (Sybil attack)
- Honeypeers attract all the 10 replicated "service" requests



Motivation

Such attack raises :

- privacy issues (attackers monitoring shared contents)
- denial of service issues (eclipse attack removing information from the DHT)
- security issues (fake files and sources insertion : pollution, malware diffusion)

Protecting the KAD network is very challenging :

- fully distributed design
- strong need of backward compatibility between clients
- no existing solution is suitable (central authority, crypto-puzzles, social networks, distributed certification)

Efficient Pollution

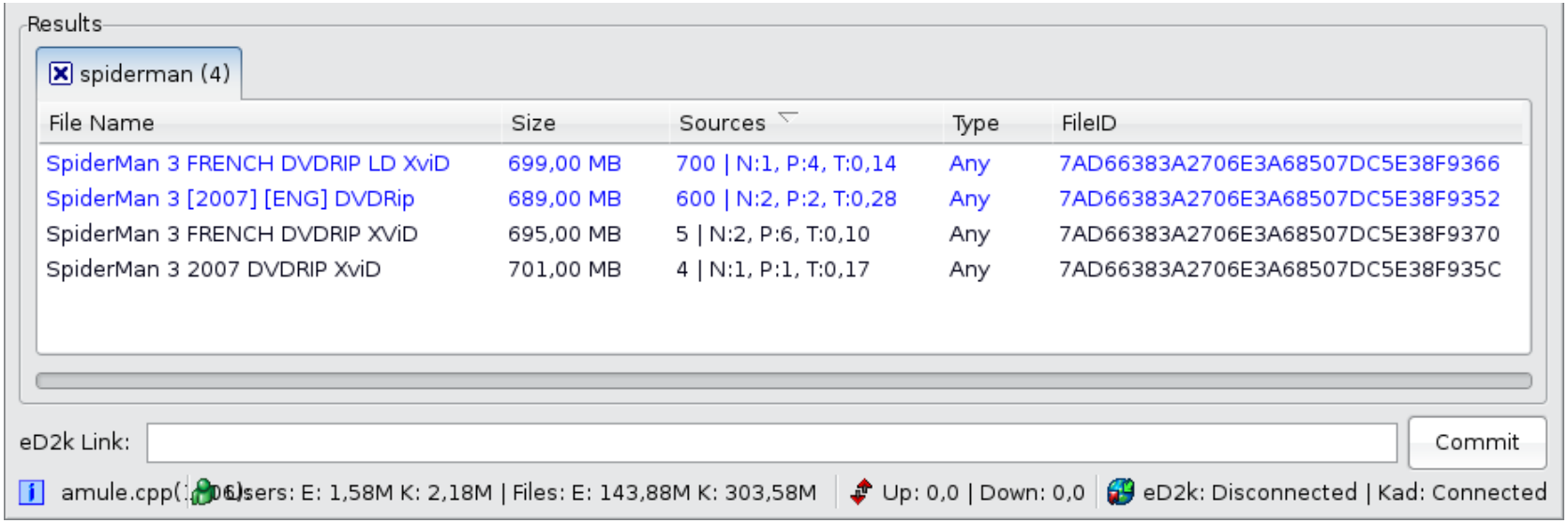


Fig.: Result of a search for "spiderman" under eclipse and poison (4 fake files)

Outline

Key Idea

Instead of controlling peer IDs :

- let them randomly choose their ID...
- but check if IDs distributions are really random !

To target an ID, DHT attacks introduce :

- proximity abnormalities in IDs distribution
- density abnormalities in IDs distribution

Type	KADID	prefix
content	477221265829086C74988C40EFE63DAF	-
attacker	477221265829086C74988C4070D6E0F1	96 bits
normal	477229E3D7CFC729F337ABBB69C983C6	20 bits

Tab.: Example of IDs

Theoretical IDs distribution

Mean number of peers sharing at least x bits with a target ID with N peers in the network :

$$F(x) = \frac{N}{2^x} \tag{1}$$

with $N = 4 \times 10^6$ and $x \in [1; 128]$.

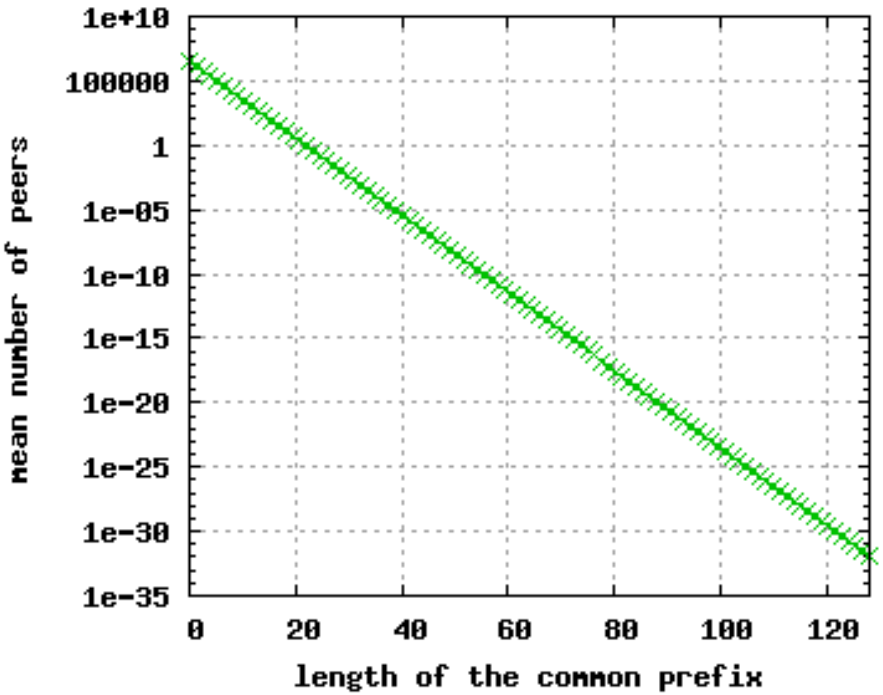


Fig.: Mean number of peers sharing a given prefix with a target

Real IDs distribution

Real network measurement :

- 1800 lookups on safe (random) DHT entries
- for each lookup : what are the prefixes of the 10 best peers found ?

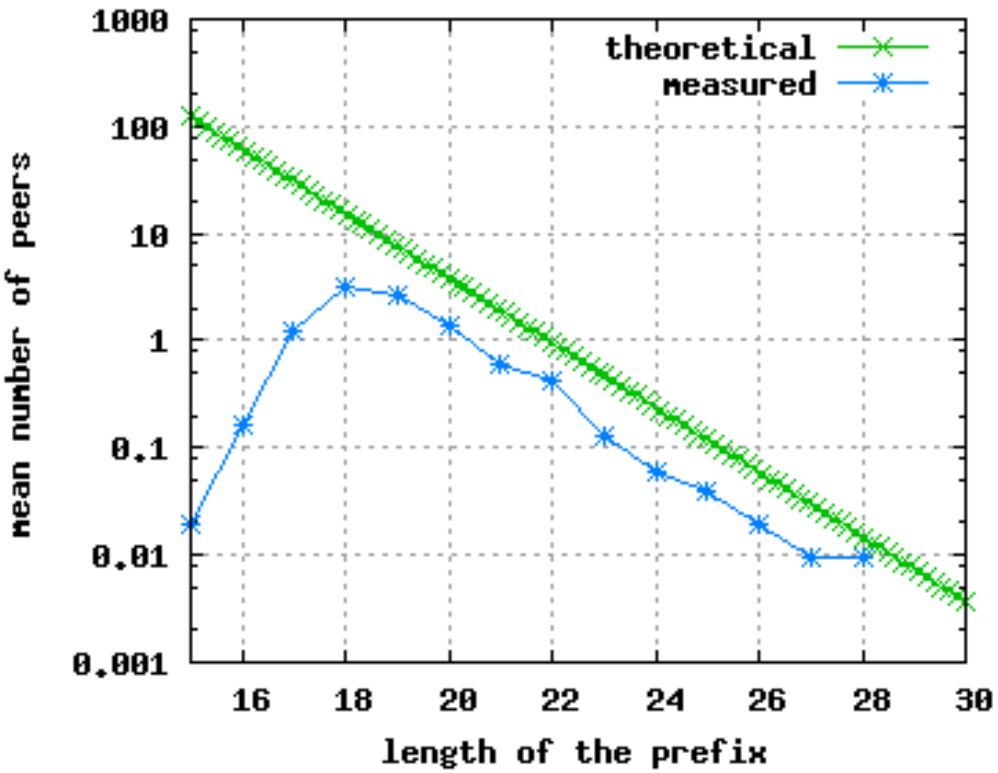


Fig.: Average Prefix distribution of the 10 best found contacts

Real IDs distribution

Results show :

- KAD lookup procedure is efficient enough to give a representative view of the closest peers possible.
- The theoretical random ID distribution (geometric distribution with parameter 1/2) is sufficient to characterize the results obtained in a real lookup process.

Moreover, IDs distribution is stable : all tested parameters do not affect it

- time spent in the P2P network
- distance between the publishing peer and the published data
- type of published information (keyword or file)
- type of requested services (publish or search)

Preventive rules

IP address limitation

- service requests must be sent to peers from different subnetwork
- already applied to filter peers inserted in routing table
- distribute a DHT entry on the IP network scale

Discarding close nodes

- currently prefixes $\geq$ 28 bits very unlikely
- change the tolerance zone from [8 ;128] to [8 ;28]

Outline

DHT attack detection

Major difficulty :

- few (10) best peers constitute a very small sample size
- common statistic tools comparing distributions (chi-square, Kolmogorov-Smirnov) inefficient
- KL-divergence efficient but must be interpreted

Kullback-Leibler divergence (G-test) to detect attacks :

$$D_{KL}(M \mid T) = \sum_i M(i) log \frac{M(i)}{T(i)} \quad (2)$$

Prefix	18	19	20	21	22	23	24	25	26	27	28
M (attack)	0	0	0	0	0	0	0	0	0.5	0.5	0
M (safe)	0.6	0.2	0.1	0.1	0	0	0	0	0	0	0
T	$1/2$	$1/2^2$	$1/2^3$	$1/2^4$	$1/2^5$	$1/2^6$	$1/2^7$	$1/2^8$	$1/2^9$	$1/2^{10}$	$1/2^{11}$

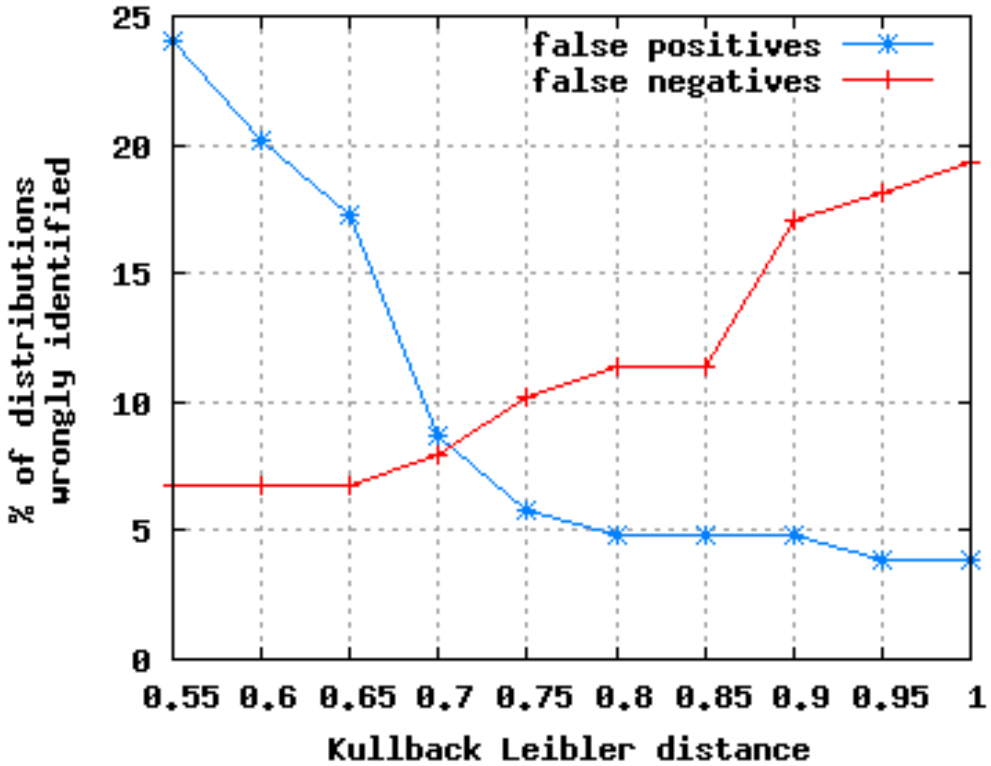
Tab.: Distributions compared with KL-distance to detect attacks

DHT attacks detection

Evaluation of the detection metric & threshold :

- 2 data sets : simulated attack distributions vs real DHT dist.
- the few false negatives are not dangerous attacks : few peers inserted (5 or less) on low prefixes (18-19 bits)

- detection threshold $= 0.7$
- false positives & negatives $< 9\%$



DHT attacks mitigation

When an attack is detected :

- countermeasures progressively filter the attacked prefixes
- while the distribution is not 'safe', remove peers with the most suspicious prefix, update distribution and distance
- peers with lower prefixes (< 18 bits) fill the left places among the 10 best

Prefix	Avg number of contacts
13	0.60
14	1.36
15	2.78
16	3.62
17	3.75

Tab.: Best remaining contacts with prefix under 18bits

DHT attacks mitigation

- countermeasure removes almost all malicious peers
- safe threshold defines the countermeasure tolerance

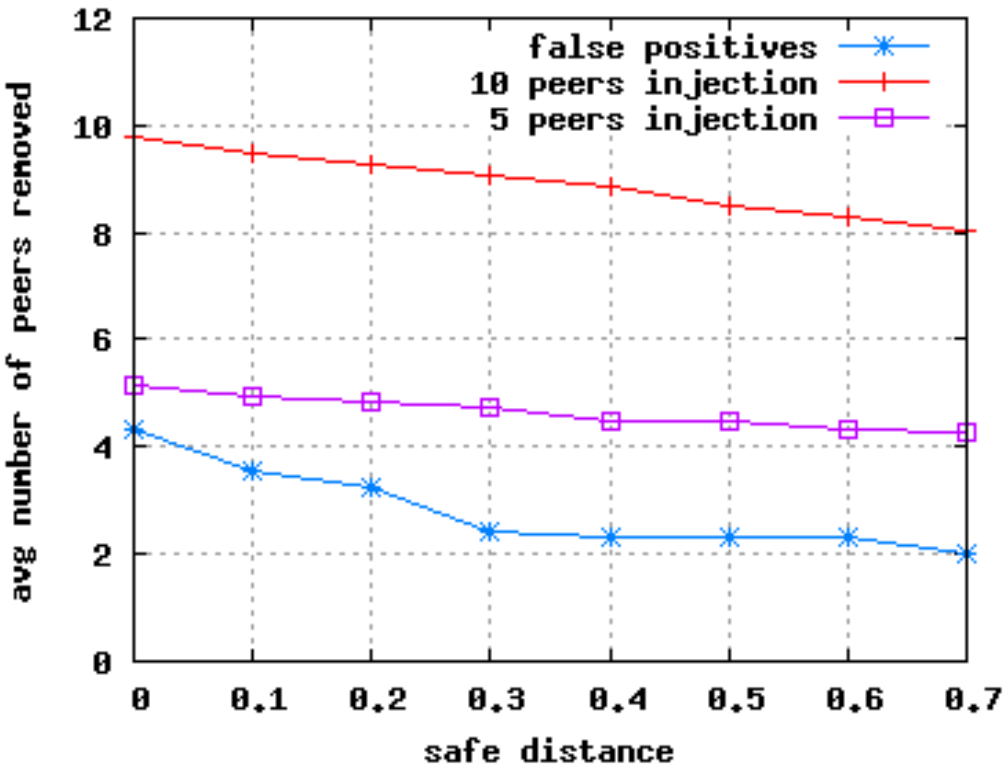


Fig.: Average number of contacts removed among the 10-best by the countermeasure

Full defense scheme

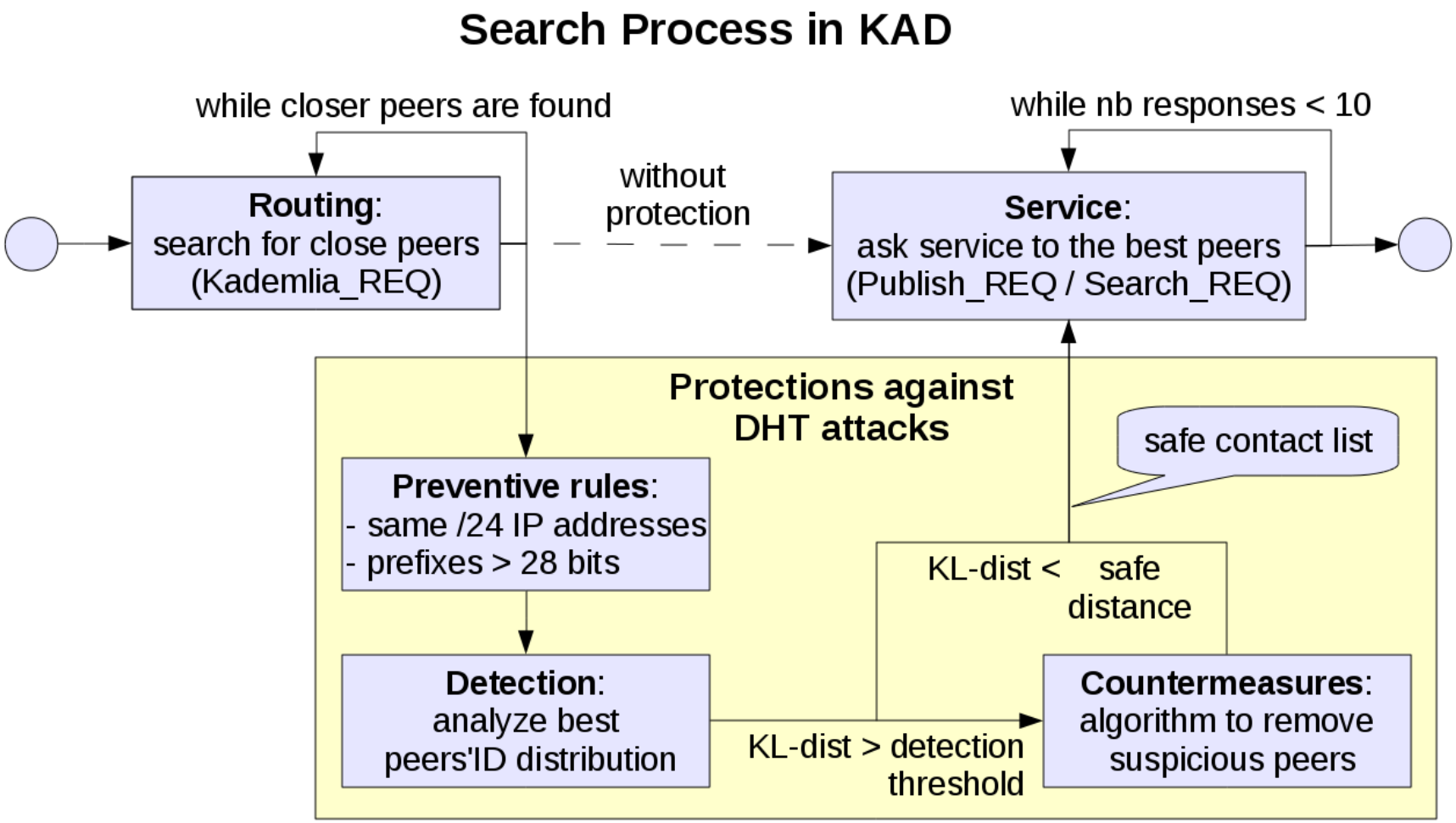


Fig.: Full defense scheme applied to KAD

Is the KAD network really threatened ?

Yes ! local attacks are running

Simple test :

- choose few "well-known" keywords
- launch DHT lookups
- write the prefix of the closest peer found

keyword	best prefix
avatar	126
invictus	123
sherlock	122
princess	122
frog	98
ncis	96
nero	96

keyword	best prefix
nine	122
love	122
american	97
russian	97
black	96
pirate	96
...	...

Outline

Conclusion

Our solution :

- is efficient ; introduces no overhead
- provides full backward compatibility
- can be applied to any DHT with iterative routing and replicated data

Future (current) work :

- crawl the KAD DHT to detect real attacks
- evaluate the implementation
- dynamically set the detection parameters

How to simulate attack distributions

- initialize with nodes following the observed average distribution of prefixes
- add different configurations of malicious nodes
- recompute final distribution of the '10 best contacts'

# of malicious peers inserted	# of prefixes targeted	Repartition of the peers	# of generated distributions
5	1	5	11
5	3	2-2-1	8
5	5	1-1-1-1-1	6
10	1	10	11
10	2	7-3	9
10	2	5-5	9
10	3	5-3-2	8
10	4	4-3-2-1	7
10	5	4-2-2-1-1	6
10	6	2-2-2-2-1-1	5
10	7	2-2-2-1-1-1-1	4
10	10	1-1-1-1-1-...-1	2

Countermeasure Algorithm

Input: contact_list [ ] ; prefixes_distribution [ ] ; KL_increments [ ] ;
KL_div ; max_div ;
Output: updated contact_list [ ]
foreach *prefix in prefixes_distribution* **do**
| KL_increments.add(partial_KL_div(*prefix*));
end
KL_div = SUM(KL_increments);
while *KL_div* > *max_div AND MAX*(*KL_increments*) > 0 **do**
| prefix=KL_increments.index(MAX(KL_increments));
| remove_contacts(contact_list, prefix);
| remove_distance(KL_increments, prefix);
| KL_div=SUM(KL_increments);
end

Algorithm 1: Countermeasure to mitigate DHT attacks