P2P and Grid Computing
- Exercise 7 -
- C110 -

Michael Scholz (Matr. 1576630)
Ulf Gebhardt (Matr. 1574373)

H 7.1:

a)
Das Paper „Attacking the Kad Network“ zeigt Schwächen und neue Angriffsszenarien von/gegen KAD Netzwerke (eDonkey). Es werden unter anderem Reflection Attack, Sybil Attack und Indexpoisoning Attack vorgestellt, die es wenigen Teilnehmern des KAD Netzwerkes erlaubt, große Teile des Netzwerks außer Kraft zu setzen und dies bei geringem Ressourcenaufwand (nur Endsysteme). Die Angriffe zielen generell darauf ab Suchanfragen fehlschlagen zu lassen. Die neu aufgeführten Angriffe werden mit schon bekannten Angriffen auf KAD Netzwerke verglichen und es werden Lösungsvorschläge vorgestellt, die benannte Schwächen beheben sollen.

b)
Kademlia benutzt Routingtabellen, welche eine eindeutige ID für jeden Knoten nutzen. Dabei wird als Distanz zwischen zwei Knote die XOR-Metrik der beiden Node-IDs verwendet. Desweiteren werden alle Daten mit einer ID indentifiziert, welche in dem selben ID-Bereich liegen wie die Knoten-ID, der sie hält.
Jeder Client hält dabei eine Routingtabelle (max. O(log(N), wobei N die Anzahl der Knoten im Netzwerk ist), welche in k Buckets gegliedert ist. Die Buckets werden anhand von Knoten-ID-Prefixen identifiziert und bevorzugt langlebige Knoten.
Die maximale Anzahl an Queries für eine Suche ist O(log(N)).
Die Suchanfrage nach Key x wird an k Knoten geschickt(k andhand der Routingtabelle). Welche Wiederum jeweils m Knoten zurückgeben, von denen wieder die nächsten gewählt werden(Routingtabelle). Das wird wiederholt bis x gefunden wurde.

KAD benutzt Routingtabellen, welche eine 128-bit ID zum identifizieren der Knoten benutzt. Für die ID gibt es keine Vorgaben. Die KAD-Routingtabelle beginnt erst auf Level 4, was im Vergleich zu Kademlia zu wesentlich kürzeren Suchpfaden führt. Dafür ist die Tabelle breiter als bei Kadmelia.

Kad-Routingtabelle:

1111 0000	1110 0000	1101 0000
1100 0000	1011 0000	1110 0000
1001 0000	1000 0000	0111 0000
0110 0000	0101 0000	0100 1000
0100 0000	0011 1000	0011 0000
0010 1000	0010 0100	0010 0000
0001 1100	0001 1000	0001 0100
0001 0010	0001 0000	0000 1110
0000 1100	0000 1010	0000 1000
0000 0110	0000 0100	0000 0010

Kademlia-Routingtabelle:

Prefixmatch	Nähester Knoten
0	1000 0000
1	0100 0000
2	0010 0000
3	0001 0000
4	0000 1000
5	0000 0100
6	0000 0010
7	0000 0001

c)
HELLO_REQ:
Jeder Knoten schickt regelmäßig (typischerweise alle zwei Stunden) eine „HELLO_REQ“-Nachricht an die Knoten in seiner Routingtabelle, um zu überprüfen, ob die Knoten noch erreichbar sind. Somit können nicht mehr verfügbare Knoten aus der jeweiligen Routingtabelle entfernt werden.

SEARCH_REQ:
Nach Auffinden eines Replicaknotens mittels „KADEMLIA_REQ“-Nachricht sendet der Initiatorknoten eine „SEARCH_REQ“-Nachricht an den Replicaknoten, welche den ursprünglichen Schlüssel beinhaltet. Der Knoten, der die Nachricht empfängt, schickt dem Absender alle Treffer zu dem geforderten Schlüssel zurück. Der Austausch wird beendet, wenn der Sender der „SEARCH_REQ“-Nachricht mehr als 300 Treffer zurückgeschickt bekommen hat.

KADEMLIA_REQ:
Sucht ein Knoten nach einem bestimmten Schlüssel, so berechnet er zuerst dessen MD5-Hashwert. Mittels dem berechneten Wert und den Knoten aus seiner Routingtabelle kann der Knoten nun die einzelnen Abstände mit Hilfe der XOR-Metrik berechnen. An die drei Knoten mit der geringsten Entfernung wird nun jeweils eine „KADEMLIA_REQ“-Nachricht gesendet. Die angesprochenen Knoten antworten mit einer „KADEMLIA_RES“-Nachricht, welche

zusätzliche Knoten beinhaltet. Dieses Verfahren wird solange wiederholt bis Replicaknoten gefunden werden.

d)

Ist ein Bucket voll und es wird ein potentieller neuer Knoten für diesen Bucket gefunden, so wird zunächst geprüft, ob die übrigen Knoten im Bucket noch antworten. Ist dies der Fall, so wird der neue Knoten in einer Art Cache gespeichert, um zu einem späteren Zeitpunkt einen „gestorbenen“ Knoten zu ersetzen. Diese Strategie wird gewählt, da davon ausgegangen werden kann, dass Knoten, welche in der Vergangenheit lange verfügbar waren, auch in Zukunft verfügbar sein werden.

e)

Allen Attacken geht das Sammeln von Daten voraus. In dem Paper als *Preparation Phase* bezeichnet. Es werden verschiedene Strategien vorgestellt um dies zu tun.
Crawling: Spawne Nodes welche einen netzwerkspezifischen ID Abstand haben und suche nach Nachbarn.
Backpointer hijacking: Es wird ein Teil der Routingtabelle des Opfers gestohlen. (macht sich unter anderem zunutze, dass die Knoten-ID über den Lebenszeitraum der Knoten-IP hinaus erhalten bleibt.)

Darauf folgt die *Execution Phase*, der eigentliche Angriff. Auch hier gibt es verschiedene Strategien:
Fake Matches: Der Gefragte antwortet mit 300 falschen Suchmatches für die Suchanfrage. 300, da nach 300 Suchergebnissen die Suche abgebrochen wird.
Stale Contacts: Suchanfragen werden mit falschen Hosts(kein KAD/Kadmelia am laufen) beantwortet. Der Timeout von Kademlia führt dazu, das der Client, der das falsche Packet erhalten hat für 30 Sekunden warten muss.

Index poisoning attack:
Bei diesem Angriff wird versucht viele fehlerhafte Verlinkungen zwischen einzelnen Schlüsseln und deren Werten zu erstellen, so dass Suchanfragen nach diesen Schlüsseln nach dem Angriff nicht erfolgreich sind.

- preparation phase: Sammeln aller Key-Value-Paare.
- execution phase: generiere für jedes Key-Value-Paar drei ungültige Verlinkungen.

Sybil attack:
Bei diesem Angriff werden „Sybil Nodes“ in das Kad Netzwerk eingeschleust. Diese sollten Anfragen an sich ziehen, die eigentlich an andere Knoten gerichtet sind. Im Paper wird nun der weitere Verlauf nicht beschrieben. Es ist aber beispielsweise denkbar durch diese Vorgehensweise ein „Blackhole“ im Netzwerk zu schaffen.

Reflection attack:
Bei diesem Angriff wird die gesamte Routingtabelle des Opfers so manipuliert, dass alle Einträge auf das Opfer selber zeigen. Somit kann der angegriffene Knoten nur noch mit sich selbst kommunizieren und wurde somit aus dem Netzwerk „entfernt“. Ziel ist das Kad-Netzwerk funktionsunfähig zu machen. Die Autoren gehen jedoch davon aus, dass sich das Kad-Netzwerk aus mehreren Gründen langsam wieder regenerieren wird.