Zusammenfassung Vertiefung Rechnernetze SS 2017

Jochen Peters

14. Oktober 2017

Inhaltsverzeichnis

Das ist ein kleiner Überblick über die Paper und deren Inhalte, um effizient für die Prüfung lernen zu können.
unvollständig und möglicherweise nicht ganz (inhaltlich) fehlerfrei

1 Design Phil von DARPA

Link zum Paper

1.1 Autoren, Jahr, Motivation

- David D. Clark
- Aug 1988
- Massachusetts Inst of Tech (Cambridge)
- Überblick, Konsequenzen und Erfolg der Designentscheidungen von vor 15 Jahren
- SIGCOMM

1.2 Inhaltlicher Aufbau / Argumentationen

- Paper fasst alte Quellen und Designüberlegungen zusammen
- Ziele

  - ARPANET und ARPA packet radio Network sollten zusammengeführt werden
  - nicht effizient alles neu, sondern Basisnetz auch für zukünftige Netze/Funktionen
  - dezentrale Administration mit Gateways sollte bestehen bleiben
  - Basis war packetbasiert, da gute Erfahrungen damit gemacht wurden
- TCP/IP soll bestehende Netze über Gateway einbinden
- Store-and-Forward war die erste Design-Idee
- Prio
  - Ausfall von Teilnetzen soll nicht zu komplettausfall führen
  - Anbieten versch. Kommunikationsservies
  - Verbinden untersch. Netze
  - verteiltes Management der nötigen Resourcen
  - kostengünstig
  - je Host einfach anzuschließen
  - Erfassung des Resourcenverbrauchs (Wer, Wo, Wie viel)
- Militär: unkaputtbar wichtiger als Resourcen kosten oder Verbrauchserfassung (heute anders)

1.2.1 Grundidee Ausfallsicherheit

- Schichtenmodell mit Maskierung+Behandlung von Fehlern je Schicht
- Statusinfos dürfen nur auf (den sicheren) Endpunkten gehalten werden

1.2.2 Types of Service

- TCP macht IP Verbindungsorientiert
- UDP ist nur dafür da, um eigene Protokolle entwickeln zu können auf Basis von IP

1.3 Kernaussagen

- damalige Ziele führten zu guten Design
- Integration bestehender Netze war klarer Vorteil statt Neuentwicklung
- Abrechenbarkeit könnte mehr in Vordergrund gestellt werden

1.4 Begriffe

DARPA Defense Adv Research Proj Agency

Packet switching im Gegensatz zum *circuit switching* (Standleitung), wird die Nachricht in Packete zerteilt und kann untersch. Wege im Netz nehmen.

Store-Forward es kommt zu delays, dafür können aber die Daten an den Netzknoten bereits auf Richtigkeit (z.B. in der IP und TCP Schicht) geprüft werden.

XNET Debugger, der mit Datagram arbeitet und dies auch braucht. Die Natur von Bugs ist derart, dass ein komplexer Handshake wie bei TCP im Ernstfall nicht mehr möglich ist.

2 Selbstähnlichkeit von Ethernet Traffik

Link zum Paper

2.1 Autoren, Jahr, Motivation

- Herren Leland, Willinger, Taqqu, Wilson
- Bellcore NY, Uni Boston
- ca 1993
- Traffic ist statistisch nicht glatt, wie bislang angenommen. Nachweis und Konsequnezen.
- SIGCOMM

2.2 Kernaussagen

- "Entropie" (oder H) bzw. "Burstiness" als Maß der Auslastung !!
- Kapitel 5.1: ein User hat kein Poison Verteiltes Netzwerk-Verhalten. Aggregierung dieser führt zu Selbstähnlichkeit.
- Kapitel 5.3: TCP skaliert auch bei Überlast gut (Burst-Phasen sind möglich)
- Überlegungen Buffer immer weiter zu vergrößern machen keinen Sinn: Burst-Phasen machen sowas kaputt
- Buffer vergrößern bringt nichts: Bursts machen das immer "kaputt"
- Theoretische Überlegungen zur Verbesserung der Performance führen zwangsläufig zu "nicht optimalen" Lösungen, da Traffic eben KEIN weißes Rauschen ist.
- Es wird versucht mit untersch. Methoden die selbstähnlichkeit zu erfassen:
  - Hurst-Parameter am einfachsten.
  - log-Auftragen von Varianz in Blöcken von untersch. großen Messreihen und entsprechend großen Zeitfenstern
  - leider gibt es keine Grenzen bzgl. H und somit ist es schwer einen statistische Vergleich mit Konfidenzintervall und Parametern bzgl. H und den Messwerten an zu geben.
  - Einen Schätzer für H via maximum likelihood ist aber möglich
  - in Kapitel 4.2 wird auch ein "Periodogram" benutzt (Grafiken mit Aggregation Level auf der rechtswert-Achse)

2.3 Begriffe

Rescaled Adjusted Range Eine Art Verhältnis zwischen den größtmöglichen tatsächlichen Auftreten einer Abweichung der Messwerte vom Mittelwert gegenüber der Streuung. Im stochastischen Modell würde bei der Zunahme an Messwerten eine Annäherung an H = 0,5 bedeuten. Werte darüber deuten auf Selbstähnlichkeit hin

$$\frac{R(n)}{S(n)} \sim a * n^H$$

Hurst Effekt Hurst Parameter ist in Natur 0,73 und deutet auf Selbstähnlichkeit hin. 1968 wird mit $1/2 < H < 1$ der Begriff "fractional Gaussian noise" geschaffen (Mandelbrot, van Ness).

2.4 Kritik am Paper

- sehr kleine, überschaubare beispielhafte Messung

3 Synchronisation periodisch Routing Nachrichten/Traffik

Link zum Paper

3.1 Autoren, Jahr, Motivation

- Frau Floyd, Herr Jacobson
- Lawrence Berkeley Lab
- April 1994
- keine Konferenz (IEEE/ACM)
- praktische Analyse und theor. Erklärung für zykl. Netzwerkprobleme

3.2 Inhaltlicher Aufbau / Argumentationen

- fällt Strom aus sind alle Router gleichzeitig betroffen
- periodische Engpässe, Latenzen und Ausfälle treten auf
- die "leichte Kopplung": direktes Update bei wegfall einer Route
- wegfall einer Route kann zu Engpass und Ausfall anderer bedeuten
- Route durch leichte Kopplung trotzdem synchron

3.3 Kernaussagen

- Routerausfälle, während Update ist Mist
- Umgehendes Update nach Routerausfall ist auch Mist
- mehr zufälliger Puffer muss sein, obwohl Latenz dadurch beeinträchtigt wird

3.4 Kritik am Paper

Das ist heute nicht mehr so ein Problem:

- heute Software und Protokollimpl. stabieler
- Router kann Routen und Update laufen lassen

4 Verstopfungsvermeidung und Kontrolle

Link zum Paper

4.1 Autoren, Jahr, Motivation

- wieder Van Jacobson, Herr M. J. Karels
- Lawrence Berkeley Lab, Uni of California
- Untersuchung und Begründung zu Verstopfungen im Netzwerk und Verfahren zur Erkennung und Behebung
- Nov 1988
- ACM SIGCOMM (bekommste da was eingereicht, hasste Dr. quasi inner Tasche)

4.2 Inhaltlicher Aufbau / Argumentationen

- Autoren beobachteten, dass 10% der Pakets am Gateway wegen überfüllter Buffer gedroppt wird
- es gab 1986 einen Crash: 32000bps auf 40bps
- Problem liegt in TCP Protokollimplementierung
- Verbesserung 4BSD TCP in diesem Paper beschrieben:
  - bessere Roundtrip Time Abschätzung
  - exponentieller Timer bei erneutem Senden

  - slow-start
  - dynamisch Fensteranpassung bei Verstopfung

- Gründe dafür, dass die Sender-Empfänger-Sender Schleife und das "Konservieren der Pakete in der Leitung" gestört ist:
  - die Verbindung ist von Beginn an bereits gestört
  - der Sender fügt bereits Pakete hinzu, obwohl alte Pakete noch nicht angekommen sind
  - durch Resourcenlimit entlang des Pfads kann nie eine gleichmäßige Verbindung erreicht werden

Das Paper arbeitet sich an diesen 3 letzten Punkten ab

4.2.1 Slow-Start

- um ein "equilibrium" zu Anfang erreichen zu können, darf man nicht bereits zu Anfang mit Kanonen auf Spatzen schießen
- der erste Schwall an Paketen einer 10Mbps Leitung kann eine 56kbps Verbindung bereits dauerhaft einen dauerhaften Schaden anrichten
- Fenstergröße öffnet sich nach jedem ACK linear
- bei Verlust oder nach langer Pause startet es wieder mit kleiner Fenstergröße
- Es wird das minimum aus der Berechneten und der "veröffentlichten" Fenstergröße genommen

4.2.2 Roundtrip Time - Wann kann ich ein ACK erwarten?

- bei Last ρ skaliert die RTT mit $(1 - \rho)$
- man hatte im April 88 und einer Auslastung von 75% im Arpanet eine RTT die um das 16fache variierte
- bisheriger Tiefpassfilter für die RTT anhand gemessender ACKs schlecht:
  - bei einem Load von 30% varriert RTT zu stark
  - die falsche vorhersage der RTT ist wie Benzin ins Feuer kippen
  - obwohl Leitung belastet, werden unnötig Re-Transmits hinzugefügt
  - eigentlich kommen ACKs nur verspätet an
  - Leitung verstopft noch mehr

Argumentation bzgl. exponetielles Warten bei (vermutetem) Packetverlust:

- Netzwerk ist in etwa ein lineares System
- Ein lin. System sei stabil, wenn die stabilität exponentiell ist
- Wenn ein zufälliger load Shock die Stabilität zerstört, bekommt man es nur mit expon. Eingriff wieder hin

4.2.3 Verstopfungsvermeidung

- Packetverlust eigentlich immer weil ein Buffer zu voll ist
- unter 1 % der Packete kommen nicht an, weil kaputt
- Puffer ist an Bandbreite angepasst
- Paketverlust erkennen heißt Verstopfung erkennen
- wenn alle, die über die verstopfte Stelle senden, ihre Fenstergröße bei Verstopfung halbieren, dann ist schnell allen geholfen
- Fenstergröße dann immer um einen konstanten Wert nach ACK anwachsen lassen

- konstant: geht schnell genug und ist "feiner" als multiplikativ
- multiplikativ: würde ein oszilierendes Verhalten verursachen
- Vorsicht: Algorithmus muss mit "slow-start" kombiniert werden!

4.3 Kernaussagen

- wollen ein Bit bzgl Verstopfung im Header haben
- sofort erneutes Senden bei Paketverlust unsinnig; Fenstergröße halbieren besser.
- Begründung zur Fensterhalbierung bei Verstopfung
- Konzept des langsamen Anwachsens der Fenstergröße beim Verbindungsaufbau bzw nach langer Pause
- sofort erneutes Senden bei Paketverlust verstopf nur noch mehr
- Paketverlust eigentlich immer nur, weil Verstopfung da ist

4.4 Kritik am Paper

- (zu) viel in Fußnoten, Anhang andere Paper ausgelagert

5 Bitcoin und mehr: tech. Überblick dezentraler digitaler Währungen

Ich habe viele Notizen gemacht aber ich befürchte, weil die Veranstaltung+Prüfung auch zu Ende ist, dass ich nicht mehr dazu komme speziell den Teil der Skripte und das Mining
Link zum Paper

5.1 Autoren, Jahr, Motivation

- Herr Tschorsch/Scheuermann
- ca 2015
- Humboldt Uni Berlin
- Überblick: Technik, Wildwuchs, Probleme, Sicherheit, Aussicht
- Eingereicht: IEEE ?

5.2 Inhaltlicher Aufbau / Argumentationen

- Überlegungen bzgl. alternative zu CPU-lastigen Verfahren (z.B. Würfeln nach Zeitspanne)
- Problem: Coin 2x ausgeben
- Problem: Coin ausgegeben, aber verteiltes System behauptet, es sei nicht passiert (ungültig?)

5.3 Kernaussagen

- durch asym. Schlüssel wird nachgewiesen, dass der Ausgang (und die neue "Einlage" für neue Transaktion) wirklich von einem ist.
- trotz spezieller Dienste: Bitcoin ist nicht anonym (wer mit wem) !
- problematisch sind chains, die versteckt ablaufen und dann plötzlich ins "Spiel" kommen
- es funktioniert und problematische Angriffe bzgl. Zusammenbruch von Keinem gewollt

5.4 Stichworte

Skalierbarkeit aktuell nur 4 Transaktionen/Sec im GANZEN verteilten System Bitcoin. Wolle man dies erhöhen stößt man auf Grenzen, da mit mehr Transaktionen pro Sek. auch Blöcke und somit die Chains länger werden. Das macht Konflikte wahrscheinlicher, es müssen im Netzwerk längere Chains (und somit viel mehr Daten) ausgetauscht werden. Dies ginge nur mit einem fat- und thin-Client Konzept (z.B. dogecoin).

Inflation das Fee, was man bekommt, erhöht im System die Anzahl der Coins. Gegenmaßnahmen: initial 50, ca alle 4 Jahre (alle 210000 Blocks) halbierung. 2140 ende und 10exp-8 BTC = 1 satoshi

Transaktionsgebühr Der die längste Block-Chain und somit den meisten Transaktionen publiziert bekommt, Gewinnt das Fee und die Transaktionsgebühren aller Transaktionen.

6 Zeit, Ordnung und die synchr. in Verteilten Systemen

Link zum Paper

6.1 Autoren, Jahr, Motivation

- Leslie Lamport
- Juli 1978
- Beweis und Algorithmus zur Möglichkeit der Zustandsermittlung eines Verteilten Systems
- Eingereicht: ACM ?

6.2 Inhaltlicher Aufbau / Argumentationen

- Ein System besteht aus Prozessen
- Ein Prozess besteht aus Events
- Nachrichtenaustausch zwischen Prozessen sind ebenfalls Events
- System ist verteilt, wenn man die Zeit zum Nachrichtenaustausch nicht vernachlässigen darf

6.3 Kernaussagen

- aus Hierarchie der Prozesse und deren Zeit kann man beim Mitführen des "lokalen Zeit" Zeitstemples das Ereignis "total" einordnen
- die schnellste Uhr dominiert
- mit echten Uhren kann man auch Bezüge zu Events außerhalb des Systems herstellen
- besser, wenn man keine echte Uhrzeit nimmt
- durch eine totale Ordnung kann das System als "State Maschine" betrachtet werden

6.4 State Maschine bei verteiltem Resourcenzugriff

Weil ich in der Prüfung an der Stelle etwas ins straucheln kam, hab ich es nochmal nachgelesen.

6.4.1 Daten

- Jeder Prozess kennt die Anzahl aller beteilgten Prozesse
- Jeder Prozess hat seine eigene Uhr (in dem Fall kein Vektoruhr - also nur schwache Clock Condition/Lamport Uhr)
- Jeder Prozess hat EINE Resource
- Jeder Prozess hat eine Liste mit Resourcenzugriffen (Zeistempel+Prozessnummer)

6.4.2 Algorithmus

- Ein Prozess i sendet Resourcenanfrage (Req) (Zeistempel + Prozessnummer i) an alle und fügt diesen in seine Queue
- Ein Prozess j bekommt diese Res.Anfrage und antwortet mit seinem aktuellem Zeitstempel (falls j nich bereits eine Nachricht mit älterem Zeitstempel als den Empfangenen an i gesendet hat.
- Zugriff auf Resource aufheben: i entfernt Req. aus Liste und sendet an alle (Zeistempel+Prozessnummer) zum Release der Resource
- Prozess j bekommt Release-Nachricht: j entfernt den Request aus Liste
- Prozess i darf auf Resource zugreifen, wenn (I) es der älteste Request auf seine Resource in seiner Liste ist und (II) Prozess i von allen anderen Prozessen (z.B. durch die ACKs) sicher sein kann, dass bei denen die Zeit bereits weiter geschritten ist (Zeitstempel höher)

Im Grunde wird nur durch Fluten eine Request-Liste an alle Prozesse verteilt und durch die Zeitstempel wird dafür gesorgt, dass der Prozess, der auf die Resource zugreift, auch wirklich den kleinsten (frühsten) Zeitstempel hat.
Kritik: Fluten. Was bei Fehlern? Wie mit Churn umgehen?

6.5 Begriffe

anormales Verhalten A happend before B aber die Uhr von A ist nicht kleiner als die von B. Es wird eine Uhr (z.B. Vektoruhr) gebraucht, die die Strong Clock Condition erfüllt

Happend Before Relation Die Relation (fetter Pfeil) $a \rightarrow b$ heißt: a passiert vor b, a sendet, b empfängt, transitiv (Dreiecksungleichung)

Gleichzeitig a und b ist gleichzeitig, wenn kein Nachrichtenaustausch zwischen a und b passiert und daher keine Aussage getroffen werden kann, ob a vor b, oder b vor a passiert. Bei Vektoruhren liefert before(a,b) und before(b,a) jeweils NICHT.

schwaches Konsistenzkriterium (oder nur "Clock Condition") gilt $a \rightarrow b$ dann folgt daraus, dass die Uhr von a einen kleineren Wert hatte, als die von b. VORSICHT ! Das bedeutet nicht, dass a "Happend Before" b gilt!

Strong Clock Condition (starkes K.k.) auch die Umkehrung (genau dann, wenn) gilt. Nebenläufige/Unabh. Ereignisse sind dann erkennbar. Implementierung als Vektoruhren und Erkennung durch partielle Ordnung.

Vektoruhr a passierte nicht vor b, wenn min. eine Prozess-Zeit-Komponente des Vektors von a größer ist als die jeweilige von b

6.5.1 PC1

Es existiert ein oberes Maß, wie falsch eine Uhr tickt. κ

6.5.2 PC2

Es existiert ein oberes Maß, wie stark Uhren zur selben Zeit abweichen. ϵ

6.5.3 anormales Verhalten

Auf der Basis von PC1 und PC2 und dem sync Algorithmus beim Datenaustausch, der die Uhren nie zurück stellt, folgt:

- anormales Verhalten ist nicht möglich, wenn ungefähr die Zeit μ zum Transver der Nachricht größer ist, als die obigen Werte "zusammen". Konkret:

$$\frac{\epsilon}{1-\kappa} \leq \mu$$

Oh Wunder: Wie oben für ein Verteiltes System definiert wird an dieser Formel klar, warum die Zeit zum Nachrichtenaustausch nicht vernachlässigbar ist :-D

6.6 Kritik am Paper

- Anzahl der beteiligten Prozesse muss bekannt sein

7 Datenreduktion durch XOR Kodierung in kabellosen Netzwerken

Link zum Paper

7.1 Autoren, Jahr, Motivation

- div. Autoren & Autorinnen (inkl. Jon Crowcroft der Uni Cambridge)
- Sep 2006
- SIGCOMM
- Nutzung der Routingprotokolle und des Broadcast Charakters von Wifi, um optional Pakete zusammen zu mischen.

7.2 Kernaussagen

- Wifi ist im Grunde immer Broadcast, daher sind an Knoten Pakete vorhanden/kommen an, die für sie nicht bestimmt sind
- COPE: Zeitspanne zum Verfall mitgehorchter Pakete: 5 sec
- Erweiterung und daher nie schlechter
- TCP muss wegen Packetreihenfolge gepuffert werden
- mitgehörte Packete werden in einem Buffer für große und kleine Packete hinzugemischt
- Problem der Hidden Terminals (das sind die Knoten, die im Grunde nur den Hotspot und nicht die anderen Teilnehmer hören)
- Wegen Hidden Terminals kommt es in Wifi oft zu Paket/Funk Kollisionen. Da TCP beim nicht-ankommen eines Packets alles von neuem senden muss, ist TCP mit Wifi schlecht und COPE entsprechend unperformant.

7.3 Stichworte

Coding Gain In der Theorie kann man mit einer unendlich langen Kette max den Faktor 2 bzgl. des Einsparens an Packeten erreichen. Unendlich, weil am Kettenanfang und Ende kein Coding passieren kann.

Coding+Mac Gain Betrachtet man die übertragene Datenmenge und die durch das MAC-Protokoll aufgeteilte Bandbreite an alle Teilnehmer, so führt eine Rad-(Speichen)-Anordnung (mit unendlich vielen Teilnehmern) zu einem unendlich großen Gain. Auch wenn der Router nur eine minimale Bandbreite nutzen darf, so kann er alle Packete in nur einem Packet übertragen.

7.4 Kritik am Paper

- bei Implementierung ein wenig die Schichten vermischt
- Coding ist in der Paxis bei normalen Wifi mit enormen gewinn anwendbar!

8 Accountweise Transfermessung durch wkeitbassiertes Zählen

Link zum Paper

8.1 Autoren, Jahr, Motivation

- Herr Lieven, Herr Scheuermann
- 2010 (IEEE)
- Uni Düsseldorf
- Weiterentwicklung von FM Sketches zur CPU- und speichereff. Zählung auf Routern (PMC)

8.2 Inhaltlicher Aufbau / Argumentationen

- Auflistung, was es schon gibt
- ist in den 50/50 (geoverteilt) Feldern >30% noch Null, dann Hitcounting
- Hitcounting nutzt Füllrate des Bitarrays zur Ermittling der "false positive" Bits
- 1. Trick: Zufall statt Hash
- 2. Trick: Hash für Matrixpos und flowid
- 3. Trick: Hitcounting (nur geom. Verteilung betrachten) bei geringer Füllrate

8.3 Kernaussagen

- zählen muss in wenigen Nanosekunden erfolgen
- Mischung geometrisch (Größenordnung) und normalverteilter (reduzierung falsch positiver Bitsetzung) Zufallsgenerator: wird mit flow id gehasht und in Bitmaske ein Bit gesetzt
- unwichtig, wie aufwändig man hinterher an die Zahlen kommt
- kleine Zahlen werden anders gezählt
- genauer (kleine Zahlen) und weniger aufwändig, als bisherige Verfahren (MRSCBF)

8.4 Stichworte

virtuelle Matrizen eine Art fake FM-Sketch

Bloom-Filter erkennen, ob etwas nicht in einer Teilmenge enthalten ist (Element wird durch k-viele Hashes auf ein Index eines Bitarray gemappt). Steuung der Hashes soll dafür sorgen, dass man auf jeden Fall mit hoher Wkeit sagen kann, dass ein Element enthalten ist. Auf JEDEN FALL kann man aber sagen, dass ein Element nicht in der Teilmenge ist (es müssen alle Bits der Hashes des Elements nicht 0 sein).

FM-Sketch wie viel bzw. Größenordnung einer Teilmenge. Eine Art Schrotflinte, die auf eine Stelle einer Wand ziehlt. Das Weiter gestreute Punkte auftauchen ist unwahrsch. aber mit Anzahl der Schüsse wahrscheinlicher. Es wird eine geometrische Verteilung beim Setzen eines Bits in Bitarray genutzt (z.B. Anzahl der Nullen am Anfang oder Ende einer Binärdarstellung der Zahl, die Beobachtet wurde). PMC streut dann noch mit normaler Zufallsfunktion auf mehrere Bitreihen, um false-positive Rate zu reduzieren.

9 Tor: die 2. Generation des Onion Routings

Link zum Paper

9.1 Autoren, Jahr, Motivation

- Herren Dingledine, Mathewson, Syverson
- Free Haven Project, Naval Research Lab (US Marine)
- August 2004
- Eingereicht: USENIX Security Symposium
- Abgrenzung zur 1. Generation, Designziele, Angriffsszenarien und Gegenstrategien

9.2 Inhaltlicher Aufbau / Argumentationen

- Kapitel bzgl. Überlastkontrolle: Was da steht ist falsch. Es gibt Szenarien wo Tor2 zusammenbricht. Tor3 hat da nachgebessert!

9.3 Kernaussagen

- Wer/ob man es nutzt, ist erkennbar!
- Wer mit wem kommuniziert ist nur erschwert erkennbar (gewisser Hardwareaufwand und tricks nötig)
- Es hilft nicht bei Protokollen, die von sich aus unverschlüsselt Userdaten übertragen
- Es hilft nicht gegen Angriffe auf mehrere Knoten einer Verbindung
- Es hilft gegen Angriffe auf einen speziellen Konten durch Umleitungen und Verschlüsselung der Verbindungsinfos

9.4 Begriffe

Ablauf DNS DNS ist normalerweise UDP und passiert vor Verbindungsaufbau. Privoxy als HTTP Proxy übernimmt das, so dass DNS nicht vorher passiert.

Ablauf Hidden Service

- Bob hinterlegt signiert infos senes service auf lookup service ab
- Bob baut zu ORs Verbindungen auf, die als Introduction Points (IP) dienen
- Alice bekommt von Bob info über, dass er einen Service hat
- Alice holt infos bei lookup service ab
- Alice baut zu ORs Verbindungen auf, die als Rendevous Points (RP) dienen
- Alice teilt einem IP ihre RPs mit + rendevous Cookie
- Bob teilt einem RP von Alice ihren rendevous Cookie mit und einen Session Key
- Ab jetzt läuft alles nur zwischen Bob-service-OP-OR-...IP-RP-...OR-OP-client-Alice
- Service und Client müssen nur mit Domains der Art "authCookie.ServicepupKeyHash.onion" umgehen können und müssen nicht angepasst werden

Onion Routing Die Hops sind in einer *onion* verschlüsselt (also mehrfach verschlüsselt).

teleskope path building im Gegensatz zur Onion wird hierbei zu jedem Hop eine eigene verschlüsselte Session aufgebaut.

Zur Verbindung zwischen Alice und Bob sind mind. 2 Onion Router (OR) nötig, um ein Ziel zu erreichen: Der erste "Hop" kennt Alice aber Bob nicht. Der 2. Hop (OR) kennt nur Bob (das Ziel) und den OR vor Alice.