

Pastry und SCRIBE

Pastry [3] ist ein skalierbares, verteiltes Routing- und Objekt-Lokalisierungsbackbone für weitreichende Peer-to-Peer (P2P) Systeme. Es ist vollständig dezentral, skalierbar und selbstverwaltend, d.h. Knoten werden automatisch eingefügt bzw. entfernt und Störungen erkannt und behoben.

In P2P-Systemen [4] haben alle Knoten die gleichen Aufgaben und Funktionen (eng. peer: Gleichgestellter). Sie sind verteilt und die Kommunikation zwischen ihnen ist symmetrisch. Aufgrund der Gleichberechtigung der Knoten gibt es keine Server oder Clients und auch keine zentralen Kontrolleinheiten. Solche Systeme werden oft als Overlay-Netzwerk realisiert, d.h. auf Anwendungsschicht wird eine P2P-Schicht eingefügt, die z.B. Routing und/oder Lokalisierung übernimmt. Jeder Knoten in einem P2P-Netz stellt üblicherweise Ressourcen zur Verfügung (z.B. Dateien). Der Zugriff auf diese dezentralen Ressourcen erfolgt außerhalb des Domain Name Systems (DNS).

SCRIBE [1] ist eine verteilte Multicast-Infrastruktur, die auf Pastry aufbaut. Es unterstützt eine große Anzahl an Gruppen und kann als Publikations- und Abonentensystem verwendet werden.

1 Pastry

Die Popularität und somit auch die Verbreitung von Peer-to-Peer Anwendungen im Internet ist in den vergangenen Jahren immer weiter gestiegen, das Kernproblem ist jedoch das gleiche geblieben: die Bereitstellung eines geeigneten Backbones (als Overlay-Netzwerk auf dem Internet), welches effiziente Algorithmen für Routing und Objektlokalisierung besitzt.

Das in diesem Beitrag vorgestellte Overlay-Netzwerk namens Pastry bietet solche effizienten Algorithmen. Es ist vollständig dezentral organisiert, zu einem hohen Grad ausfallsicher, skalierbar und zuverlässig.

Jeder Knoten innerhalb des Netzes bekommt eine eindeutige ID zugewiesen, welche z.B. aus dem SHA-1 Hash der IP-Adresse oder des öffentlichen Schlüssels berechnet werden kann. Dadurch wird gewährleistet, dass mit einer hohen Wahrscheinlichkeit benachbarte Knoten im Pastry-Raum gleichmäßig über das darunterliegende IP-Netzwerk verteilt sind. Pastry routet Nachrichten mit einem gegebenen Schlüssel immer zu dem Knoten, der diesem Schlüssel numerisch am nächsten ist. Dabei werden die Lokalitätseigenschaften des darunterliegenden Netzes berücksichtigt.

Es wurden schon einige Anwendungen für Pastry entwickelt, darunter PAST [2], ein persistenter Dienst für eine verteilte Datenspeicherung von Dateien, und das hier vorgestellte SCRIBE [1].

1.1 Design

Das gesamte Pastry-System ist ein selbstverwaltendes Overlay-Netzwerk, aufgesetzt auf dem Internet (oder einem entsprechenden IP-Netz). Jeder Knoten mit der entsprechenden Software kann uneingeschränkt als Pastry-Knoten fungieren.

Dazu wird jedem Knoten eine zufällige, eindeutige 128-Bit ID zugewiesen. Demzufolge sind Knoten mit „benachbarten“ IDs mit einer hohen Wahrscheinlichkeit weder geographisch, noch topologisch, noch in irgendeiner Weise miteinander korreliert. Die Knoten-ID beschreibt die Position des Knotens im zirkulären Namensraum von Pastry von 0 bis 2^{128} .

Das Ziel von Pastry ist eine effiziente Suche eines Schlüssels K innerhalb des Namensraums. Der Schlüssel kann z.B. eine Datei (PAST) oder ein Nachrichtentopic (SCRIBE [1]) sein. Sei N die Anzahl der Knoten in einem Netzwerk. Pastry kann zu jedem Knoten in höchstens $\lceil \log_{2^b} N \rceil$ Schritten routen. Die Grundlage hierfür ist ein Baum mit jeweils 2^b Blättern pro Knoten (vgl. B*-Bäume). Die Variable b ist ein Konfigurationsparameter mit einem typischen Wert von 4. Selbst bei einem Ausfall von $\lfloor |L|/2 \rfloor$ (L Blattmenge vgl. Kapitel 1.1.1) Knoten mit benachbarten IDs kann noch eine Zustellung der Pakete garantiert werden. Typische Werte für $|L|$ sind 16 oder 32.

Im folgenden wird auf das Netzwerk genauer eingegangen.

1.1.1 Knoten

Jeder Knoten von Pastry besitzt Zustandsinformation in Form einer Nachbarschaftsmenge M (*neighbourhood set*), einer Blattmenge (*leaf set*) L und einer Routingtabelle (*routing table*) R .

Die Nachbarschaftsmenge M beinhaltet die Node-IDs und IP-Adressen der $|M|$ topologisch nächsten bekannten Pastry-Knoten gemäß einer gegebenen Metrik (meist Laufzeit oder Anzahl IP-Hops). Sie wird im Normalfall nicht für das Routing benötigt. Die Blattmenge L besteht aus $|L|/2$ numerisch größeren und aus $|L|/2$ numerisch kleineren Knoten. L wird für das Routing verwendet. Typische Werte für $|L|$ und $|M|$ sind 2^b oder $2 \cdot 2^b$.

Die Routingtabelle besteht aus $\lceil \log_{2^b} N \rceil$ Zeilen mit jeweils $2^b - 1$ Einträgen. In jeder Zeile n tauchen nur IDs von Knoten auf, deren führende n Stellen der Knoten-ID der ID des lokalen Knotens entsprechen, beginnend bei Zeile 0 (keine gemeinsamen führenden Stellen). Die Spalten geben die $n + 1$ te Stelle an. Jeder Eintrag der Tabelle enthält somit die IP-Adresse von einem möglichen Knoten mit entsprechendem Präfix.

Folgende Tabelle zeigt einen Pastry-Knoten

Knoten-ID 12212					
leaf set	<	12210	12112	12113	12103
	>	12221	12222	12223	12312
neighborhood set		32013	23001	01320	33233
		21320	01003	30021	21321
routing table		0	1	2	3
	0	01231	-	21213	32132
	1	10221	11213	-	1-3301
	2	12033	12112	-	12312
	3	12203	-	12222	1223-
	4	12210	12211	-	12213

mit der ID 12212 (mit $b = 2$ und $|L| = 8$, Routingtabelle ohne IP-Adressen)

1.1.2 Routing

Gegeben sei eine Nachricht mit dem Schlüssel

D . Mit R_l^i Wert der Routingtabelle an der Stelle l, i (Zeile, Spalte), L_i i -ter numerisch nächste Knoten der Blattmenge (negatives i für numerisch kleiner), D_l l -te Stelle des Schlüssels D (beginnend mit 0) und $shl(A, B)$ Länge des gemeinsamen Präfix von A und B ergibt sich folgender Routing-Algorithmus:

```

(1) if ( $L_{\lfloor |L|/2 \rfloor} \leq D \leq L_{\lceil |L|/2 \rceil}$ ) {
(2)   // D im leaf set
(3)   sende an  $L_i$  mit  $|D - L_i|$  minimal;
(4) } else {
(5)   // Routingtabelle benutzen
(6)   Let  $l = shl(D, A)$ ;
(7)   if ( $R_l^{D_l} \neq null$ ) {
(8)     sende an  $R_l^{D_l}$ ;
(9)   } else
(10)    sende an  $T \in L \cup R \cup M$ , so dass
(11)       $shl(T, D) \geq l$ ,  $|T - D| < |A - D|$ ;
(12)  }
```

Falls der Schlüssel D der Nachricht größer als der kleinste Eintrag und kleiner als der größte Eintrag der Blattmenge ist, wird die Nachricht direkt an den Knoten weitergeleitet, der D am nächsten ist (Zeile 1-3). Andernfalls wird das gemeinsame Präfix von D mit der lokalen Knoten-ID A gebildet (Zeile 6) und die Nachricht wird an einen Knoten weitergeleitet, der mindestens ein Präfix der Länge $l + 1$ mit D besitzt (Zeile 7-8). Im seltenen Fall, dass ein solcher Knoten nicht bekannt ist ($R_l^{D_l}$ leer), route zu einem Knoten T , der numerisch näher an D liegt. Das Verfahren bricht ab, sobald die Nachricht an den Knoten geroutet wurde, der numerisch am nächsten zu D liegt.

Dieses Routingverfahren konvergiert, da die Nachricht in jedem Schritt ihrem Empfänger D näher kommt.

1.1.3 API

Pastry bietet Anwendungen eine einheitlich Schnittstelle, bestehend aus den folgenden Methoden:

nodeId = pastryInit(Credentials, Application): Fügt einen Knoten in das Pastry-Netzwerk ein. *Credentials* dient zur Authentifizierung, *Application* ist ein Handle auf die Anwendung für Callbacks.

route(msg, key): Routet eine Nachricht *msg* zu dem Knoten *key*.

Die Anwendung selbst muss folgende Methoden bereitstellen:

deliver(msg,key): Wird aufgerufen, wenn eine Nachricht empfangen wird und die lokale Knoten-ID ist numerisch am nächsten zu *key*.

forward(msg,key,nextId): Jede Anwendung hat vor dem Weiterleiten einer Nachricht die Möglichkeit, diese zu lesen oder zu verändern.

newLeafs(leafSet): Über diese Methode wird der Anwendung jede Änderung an dem leaf set mitgeteilt.

1.1.4 Verwaltung und Anpassung

Das Einfügen eines neuen Knotens in das Netzwerk erfordert eine Initialisierung seiner Knotentabellen. Dazu wird eine entsprechende *join*-Nachricht an einen beliebigen, vorher bekannten Pastry-Knoten gesendet. Dieser leitet die Nachricht an einen weiteren Knoten mit nächster Knoten-ID zu dem neuen Knoten. Auf diesem Weg läuft das Paket auf einer Route mit immer ähnlicher werdenden Knoten-IDs. Alle diese Knoten senden eine Nachricht an den Neuankömmling, bestehend aus den Knotentabellen. Bei Übereinstimmung der ersten *i*-stellen der Knoten-ID wird die entsprechende Zeile der Routingtabelle gesetzt. Die Blattmenge wird vom letzten Knoten auf dem Weg übernommen, die Nachbarschaftsmenge vom ersten.

Ein Ausfall eines Pastry-Knotens bedeutet, dass der betreffende Knoten nicht mehr mit seinen unmittelbaren Nachbarn kommunizieren kann. In diesem Fall müssen die Nachbarn ihre Blattmenge aktualisieren, um das Verschwinden des Knotens schnellstmöglich zu kompensieren. Auch für diesen Fall besitzt Pastry geeignete Prozeduren.

1.1.5 Lokalität

Pastry routet Nachrichten gemäß der jeweiligen Knotentabellen. Die Perfomanz hierfür kann in Pastry-Hops angegeben werden. Es gibt jedoch keine Korrelation zwischen dieser Metrik und der tatsächlichen Dauer der Zustellung. So kann es passieren, dass sich zwei (pastry-)benachbarte Knoten im darunterliegenden Internet in vollkommen unterschiedlichen physikalischen Netzen befinden und

daher auch nur sehr langsam miteinander verbunden sind. Somit gibt es gute und schlechte Routen zu einem Ziel, da es immer mehrere potentielle Zwischenknoten gibt. Zu diesem Zweck wird angenommen, dass eine Anwendung eine Entfernungsfunktion implementiert, welche es ermöglicht, eine Distanz von einem beliebigen Pastry-Knoten zu einer IP-Adresse zu berechnen. Bei der Weiterleitung von Nachrichten werden Knoten mit geringerer Distanz als besser angesehen und bevorzugt. Diese Entfernungsfunktion kann z.B. aus der Berechnung der Anzahl der IP-Hops bestehen oder aus der geographischen Entfernung zweier Knoten.

1.2 Perfomanz

Die Simulationen wurden mit bis zu 100.000 Knoten in Java durchgeführt. Die einzelnen Knoten waren hierbei gleichverteilt auf einer Ebene, was natürlich in keinster Weise einem realen Netz wie dem Internet entspricht. Bei der Simulation ist zu sehen, dass das Pastry-Netzwerk gut für eine große Anzahl an Knoten skaliert. Einzelne Ergebnisse, die in dem Paper angeführt werden, sind dabei jedoch nicht sonderlich überraschend. Es ist z.B. nicht verwunderlich, dass das Routing sehr gut skaliert, da es hierarchisch ist. Die unwesentliche Verbesserung wird dadurch erzielt, dass zuerst in die Blattmenge „geschaut“ wird (vgl. Abb. 1).

Ein interessanter Punkt bei der Simulation ist auch der Vergleich zwischen der durch Pastry gefundenen Route und der optimalen Route zwischen zwei Knoten (Abb. 2). Hier ist zu sehen, dass aufgrund der Lokalitätseigenschaf-

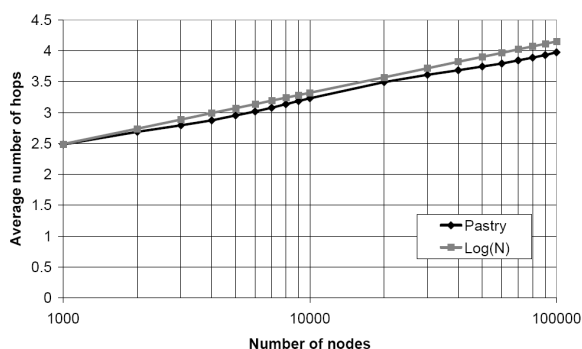


Abb. 1. Verhältnis zwischen Anzahl der Knoten und der Anzahl der Routing-Hops ($b = 4$, $|L| = 16$, $|M| = 32$, 200000 lookups)

ten von Pastry die gewählte Route nur zwischen 30 und 40% schlechter als die optimale ist, gemessen an der Distanz zwischen den simulierten Knoten in der Ebene. Ob sich dieses Ergebnis auch bei einer realen Messung z.B. im Internet zeigt, ist fraglich.

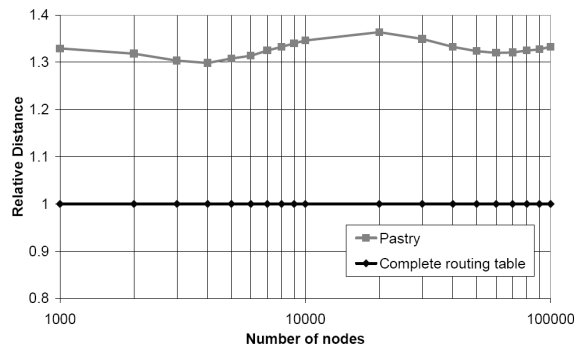


Abb. 2. Optimale Route vs. Pastry-Route ($b = 4$, $|L| = 16$, $|M| = 32$, 200000 lookups)

Es wurden noch weitere Simulationen durchgeführt, auf die an dieser Stelle jedoch nur verwiesen wird.

1.3 Annahmen

Damit die in dem Beitrag vorgestellten Mechanismen und Algorithmen funktionieren, müssen entsprechende Annahmen getroffen werden. Diese sind:

- systemweit eindeutige Knoten- bzw. Objekt-IDs, gleichverteilt im Pastry-Netz
- alle Knoten haben die gleichen Ressourcen und Aufgaben
- symmetrische Verbindungen
- Kenntnis von mindestens einem lebendigen Pastry-Knoten
- Vorhandensein einer Funktion zur Bestimmung einer Entfernung des lokalen pastry-Knotens zu einer gegebene IP-Adresse

2 SCRIBE

SCRIBE ist eine große, dezentrale Multicast-Infrastruktur, die auf dem in Kapitel 1 vorgestellten Overlay-Backbone Pastry aufsetzt. Sie bietet eine effiziente Schnittstelle, um eine große Anzahl an Multicast-Gruppen, Sendern und Empfängern zu ermöglichen und kann als ein Publisher/Subscriber-System verstanden werden, wobei die einzelnen Topics die

Grundlage für die ObjektID gemäß Pastry bilden. Hierfür wird ein sogenannter Multicast-Baum aufgebaut, der es ermöglicht, alle Teilnehmer einer Gruppe zu erreichen. Die einzelnen Routen der Teilnehmer zu der Wurzel bilden den Baum, die Wurzel wird als *rendez-vous point* bezeichnet. Die einzelnen Nachrichten werden nach *best-effort* ausgeliefert.

2.1 Design

Die einzelnen SCRIBE-Knoten können neue Gruppen erstellen, Nachrichten an Gruppen senden und vorhandenen Gruppen beitreten. Die Gruppen wiederum können verschiedene Multicast-Initiatoren und sehr viele Mitglieder besitzen. Die dazugehörige Schnittstelle zur Anwendung sieht folgendermaßen aus:

create(credentials, groupId): neue Gruppe mit der ID *groupId* anlegen

join(credentials, groupId, msgHandler): Gruppe mit der ID *groupId* beitreten

leave(credentials, groupId): Gruppe mit der ID *groupId* verlassen, und

multicast(credentials, groupId, msg): Multicast-Nachricht *msg* ans Gruppe *groupId*.

Die eigentliche Gruppenverwaltung wird von Pastry erledigt.

Das Scribe-System besteht somit aus einem Netzwerk von Pastry-Knoten, auf denen jeweils die SCRIBE-Software zu finden ist. Es gibt insgesamt vier verschiedene Nachrichtentypen, *JOIN*, *CREATE*, *LEAVE* und *MULTICAST* auf die in den nachfolgenden Kapiteln genauer eingegangen wird.

2.1.1 Gruppenmanagement

Jede Gruppe wird durch eine eindeutige GruppenID gekennzeichnet, die aus dem Pastry-Namensraum stammt. Die GruppenID wird üblicherweise aus dem textuellen Namen des Topics und dem Ersteller berechnet. Der Knoten mit der numerisch nächsten Pastry-ID übernimmt die Funktion des *rendez-vous* Punktes, also der Wurzel des Multicast-Baumes der Gruppe. Das Versenden einer *create* Nachricht (`route(CREATE,groupId)` vgl. Pastry-API, Kapitel 1.1.3), um eine neue Gruppe anzulegen, veranlasst Pastry dazu, die Nach-

richt genau zu diesem Knoten zu routen. Es ist auch möglich, den Ersteller einer Nachricht zum *rendez-vous* Knoten zu machen.

2.1.2 Teilnehmerverwaltung

Die einzelnen Pfade der Gruppenteilnehmer zum Wurzelknoten bilden den Multicastbaum. Will nun ein neuer Knoten Mitglied einer Gruppe werden, so sendet er eine spezielle *join* Nachricht (*route(JOIN,groupId)*) an den *rendez-vous* Knoten. Jeder Knoten auf dem Weg zur Wurzel wird Bestandteil des Multicast-Baumes. Knoten, die nur Nachrichten weiterleiten, ohne selbst Mitglied der entsprechenden Gruppe zu sein, nennt man *forwarder*. Erhält ein Knoten des Baumes eine *join* Nachricht, so muss er diese nicht weiterleiten, sondern nur lokal den Sender der Nachricht zu seiner „Kindermenge“ hinzufügen.

Das Verlassen einer Gruppe (*route(LEAVE,groupId)*) erfolgt entsprechend. Sobald ein Knoten gefunden wird, der in seiner Kindermenge noch weitere Knoten als den Sender enthält, wird der Vorgang abgebrochen. Alle anderen Knoten auf der Route dahin sind nur *forwarder* und können den Baum verlassen.

2.1.3 Multicast

Sender einer Multicast-Nachricht benutzen Pastry, um die Quelle des Multicasts zu lokalisieren, den *rendez-vous* Knoten, mit der numerisch nächsten ID zur GruppenID, welche das entsprechende Topic beschreibt (*route(MULTICAST,groupId)*). Es gibt pro Gruppe nur einen einzigen Multicast-Baum, von dessen Wurzel aus sich jeder Multicast ausbreitet. Um die Effizienz des Verfahrens zu erhöhen, wird die IP-Adresse der Wurzel angefragt und bei weiteren Multicasts dazu benutzt, direkt zum *rendez-vous* Knoten zu routen, ohne das Pastry-Routing zu benutzen. Bei einem Ausfall oder einer Änderung des Wurzelknotens muss jedoch erneut Pastry benutzt werden.

2.2 Zuverlässigkeit

SCRIBE bietet nur einen *best-effort* Dienst. Jedoch stellt es ein Framework für Anwendungen zur Verfügung, um stärkere Forderungen/Garantien an Zuverlässigkeit zu ermöglichen.

Falls es einen Ausfall innerhalb des Multicast-Baumes gibt, werden die Mechanismen von Pastry benutzt, um den Baum zu reparieren. Hierzu ist es jedoch notwendig, dass die Knoten innerhalb des Baumes (also keine Blätter) periodisch Heartbeats an ihre Kinder schicken. Falls der Heartbeat ausbleibt, wird eine neue *join* Nachricht erstellt.

Der Ausfall des *rendez-vous* Knotens kann dahingehend kompensiert werden, dass sein Zustand in den *k*-nächsten Knoten zur aktuellen ID repliziert wird, einer von diesen Knoten also die Funktion der Wurzel übernehmen kann.

2.3 Evaluierung

Die Evaluierung wurde durch einfache Simulationen durchgeführt, die eine reale Umgebung stark vereinfacht darstellen (keine Verzögerung durch Puffer, kein Packetverlust, kein Querverkehr). Es zeigt sich, dass SCRIBE gut skaliert, d.h. auch eine große Anzahl an Knoten, Gruppen und Teilnehmern pro Gruppe zufriedenstellend unterstützt wird. Auf die einzelnen Experimente möchte ich an dieser Stelle nur verweisen.

3 Literatur

- [1] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron, *SCRIBE: A large-scale and decentralised application-level multicast infrastructure*, IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications). 2002.
- [2] P. Druschel, A. Rowstron, *PAST: A large-scale, persistent peer-to-peer storage utility*, Proceedings HotOS VIII, Schloss Elmau, Germany, May 2001
- [3] A. Rowstron and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.
- [4] Chonggang Wang and Bo Li, *Peer-to-peer overlay networks: A survey*, Technical Report, Department of Computer Science, HKUST, Feb. 2003.