

Provably Competitive Adaptive Routing

Stefan Nessel

7. Januar 2008

1 Einleitung

Wireless Ad-hoc-Netze setzen sich aus vielen mobilen Knoten zusammen, die über Funk miteinander kommunizieren. Da die Reichweite einer Funkverbindung nur begrenzt ist, werden Daten, bei denen keine direkte Verbindung zwischen Sender und Empfänger hergestellt werden kann, über mehrere Netzknotten weitergeleitet, bis der Zielknoten erreicht wird. Hierbei wird ein Routingverfahren eingesetzt, damit die Datenpakete vom Sendeknoten auch den richtigen Zielknoten erreichen.

Routingverfahren sind jedoch anfällig für verschiedenste Angriffe. Fehlerhafte Knoten können zum Beispiel durch „Denial of Service“-Angriffe Teile des Netzwerks stören. Diese lassen sich aber durch geeignete Routingverfahren einfach umgehen. Das Ziel von [BAR] ist es ein Routingverfahren zu entwickeln, welches sich auf einen Angreifer einstellen kann, der durch Kenntnisse über Netzwerk und Netzwerkverkehr das Netzwerk gezielt und dynamisch stören kann. Schon vorhandene Lösungen bauen z.B. auf rechnergestütztes Lernen auf. Sie finden fast optimale Pfade anhand der existierenden Daten über ein Netzwerk. Dabei werden auch dynamische Kosten für die Verbindung zwischen zwei Netzwerkknoten berücksichtigt. Eine weitere Methode ist das „Reinforcement Learning“. Hierbei werden nach dem Vorbild der Ameisenstraße die optimalen Pfade eines Netzwerks stärker gewichtet. Mit [BAR] soll nun ein Routingverfahren entwickelt werden, welches diese Charaktereigenschaften auch beinhaltet.

2 Das Routingverfahren

Das neue Routingverfahren beruht auf dem ständigen Testen von neuen suboptimalen aber fehlerfreien Pfaden. Diese weichen nur wenig von dem optimalen Pfad ab. Das Routingverfahren beginnt mit dem Vorhaben Daten von einem Sendeknoten s zu einem Empfängerknoten r zu senden. Über die Netzwerktopologie sind zuerst nur die m möglichen Verbindungen zwischen allen Knoten bekannt. Wie gut die jeweilige Verbindung zwischen den

einzelnen Knoten ist, wird erst im Laufe des Verfahrens bestimmt.

Im nächsten Schritt wird das Netzwerk in einen geschichteten und gerichteten Graphen T_R umgewandelt, so dass in Schicht 0 der Empfängerknoten ist. Der neue Graph hat H Schichten. Somit erreicht jedes Paket den Empfänger in maximal H Sprüngen.

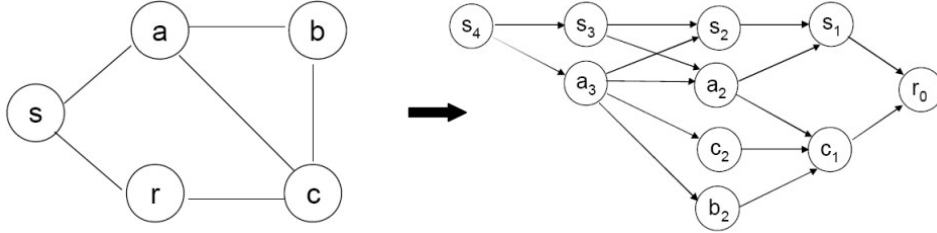


Abbildung 1: Netzwerk und geschichteter Graph

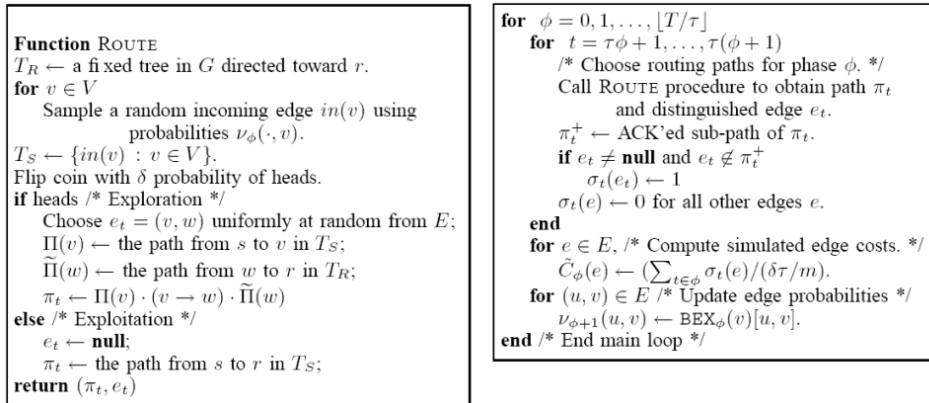


Abbildung 2: Der Routingalgorithmus

Der Algorithmus läuft in Zeitschritten von 1 bis T , wobei jeweils τ Zeitschritte zu einer Phase gehören. Der aktuelle Zeitabschnitt wird mit t und die Phase mit ϕ gekennzeichnet.

Jeder Zeitschritt t beginnt mit der Auswahl eines Pfades π_t und einer Kante e_t . Dies wird durch die zufällige Auswahl einer eingehenden Kante für jeden Knoten im Graphen T_R erreicht. Die zufällige Auswahl richtet sich nach einer Wahrscheinlichkeitsverteilung $\nu_\phi(\cdot, v)$ für die jeweiligen Knoten und ihren eingehenden Kanten. Die Wahrscheinlichkeit eine bestimmte Kante zu wählen ist in der ersten Phase vom Algorithmus noch für alle Kanten gleich. Mit der zweiten Phase wird sich dies jedoch ändern. Die ausgewählten Kanten bilden nun mit ihren Knoten einen neuen Graphen T_S . In diesem

Graphen ist jeder Knoten vom Sendeknoten s durch exakt jeweils einen Pfad erreichbar.

Als nächstes wird durch einen Münzwurf mit der Wahrscheinlichkeit δ entschieden, ob es nun als „Sampling“- bzw. „Exploration“-Schritt weiter geht oder es einen „Exploitation“-Schritt gibt. Mit einem „Sampling“-Schritt wird eine zufällige Kante $e_t = (v, w)$ aus dem Netzwerk ausgewählt und ein Pfad π_t gebildet, der von s nach v über einen Pfad in T_S läuft, dann weiter über die Kante e_t und schließlich über einen weiteren Pfad in T_R von w nach r . Handelt es sich aber um einen „Exploitation“-Schritt, so ist der Pfad π_t der Pfad von s nach r in dem Graphen T_S . Der Wert von e_t wird leer gelassen.

Anschließend wird nun über den Pfad π_t ein Datenpaket gesendet. Jedemal nachdem ein Knoten das Datenpaket auf dem Pfad weiterleitet, wird ein Timer gestartet. Gibt es nach einer bestimmten Zeit kein ACK, also eine Bestätigung von dem Knoten an den das Paket weitergeleitet wurde, so gilt das Paket als verloren. Daraufhin wird ein negatives ACK (NACK) dem Sendeknoten zugesendet. Kommt das Datenpaket jedoch beim Empfänger-knoten an, so wird auch dies dem Sendeknoten mitgeteilt. Somit weiß der Routingalgorithmus nun bis wohin der gewählte Pfad π_t zuverlässig ist. Dieser Teilpfad wird mit π_t^+ notiert. In einem „Sampling“-Schritt wird nun für den Fall, dass die Kante e_t nicht in π_t^+ liegt, diese Kante mit $\sigma_t(e_t) \leftarrow 1$ als Fehlschlag bewertet. Alle anderen Kanten werden sowohl im „Sampling“- als auch im „Exploitation“-Schritt mit $\sigma_t(e) \leftarrow 0$ bewertet. Mit den Bewertungen der Kanten wird ein Zeitschritt t abgeschlossen.

Diese Zeitschritte werden nun solange wiederholt, bis eine Phase ϕ beendet ist. Zwischen den Phasen werden die Wahrscheinlichkeitsverteilungen der Kanten für die nächste Phase neu berechnet. Dazu werden zuerst für alle Kanten die simulierten Kosten $\tilde{C}_\phi(e)$ in der Phase ϕ berechnet. Diese setzen sich aus der Summe der fehlgeschlagenden „Sampling“-Schritte für die jeweilige Kante geteilt durch erwartete Anzahl der „Sampling“-Schritte $\delta\tau/m$ zusammen.

```

Function BEX $_\phi(v)[u, v]$ 
/* Sum weights of incoming edges to v. */
 $W \leftarrow \sum_{(w,v) \in E} \beta^{\tilde{C}_\phi(w,v)}$ 
/* Return probability of edge (u, v). */
return  $\beta^{\tilde{C}_\phi(u,v)} / W$ 

```

Abbildung 3: Black Box BEX(v)

Der Routingalgorithmus benutzt für die Berechnung der Wahrscheinlichkeiten für jede Kante (u, v) eine Black Box $BEX_\phi(v)[u, v]$. In der Black Box

wird nun $\beta \tilde{C}_\phi(w, v)$ für jede eingehende Kante (w, v) vom Knoten v berechnet. Die Werte werden in W aufsummiert und die Black Box gibt als Rückgabewert $\beta \tilde{C}_\phi(u, v)/W$ zurück. Dieser Rückgabewert wird in $\nu_{\phi+1}(u, v)$ gespeichert, damit er für die nächste Phase zur Verfügung steht. Der Parameter $\beta = 1 - \epsilon$ bestimmt wie schnell sich der Routingalgorithmus Veränderungen im Netzwerk anpasst. Ein kleiner Wert für β bewirkt ein gieriges Verhalten.

Sind die Wahrscheinlichkeiten für alle Kanten berechnet, endet diese Phase und eine neue beginnt.

3 Performance des Routingverfahrens

Eine Schranke für die Performance des Routingalgorithmus lässt sich mit folgenden Begriffen darstellen. $BCOST_t(\pi)$ ist 1, wenn der Pfad π nicht fehlerfrei ist, ansonsten ist es 0. $ACOST_t(\pi)$ beschreibt die gesamte Anzahl an Kantenfehlern auf dem Pfad π . $BCOST(\pi)$ und $ACOST(\pi)$ liefern dann entsprechend die jeweiligen Durchschnittswerte. Der Algorithmus erfüllt dann nach [BAR] folgende Eigenschaft:

$$\mathbf{E}(BCOST(\text{algorithm})) \leq \mathbf{E}(ACOST(\pi)) + O\left(\left(\frac{H^2 m \log(mT) \log(\Delta)}{T}\right)^{1/3}\right).$$

Für ein entsprechend großes $T = \omega(H^2 m \log(mT) \log(\Delta))$ sind die eigentlichen Kosten des Routingverfahrens $\mathbf{E}(BCOST(\text{algorithm}))$ nur unbedeutend größer als die Kosten des Benchmarks $\mathbf{E}(ACOST(\pi))$.

Der Beweis dazu wird in [BAR] durch Induktion geführt. In [BAR] wird zuerst gezeigt, dass durch die geeignete Wahl $\tau \geq m \log(mT)/\delta$ mit einer Wahrscheinlichkeit von $1 - 1/mT$ die Kosten von $\tilde{C}_\phi(e) \leq 5$ sind und somit vernachlässigbar. Um mit der Induktion beginnen zu können, muss folgende Eigenschaft der Black Box $BEX_\phi(v)$ herangezogen werden, wobei Δ die Anzahl der einkommenden Kanten von v angibt:

$$\sum_\phi \sum_e p_\phi \tilde{C}_\phi(e) \leq \sum_\phi \tilde{C}_\phi(e_0) + O\left(\frac{\epsilon T}{\tau} + \frac{\log(\Delta)}{\epsilon}\right).$$

Die zufällige Auswahl einer Kante durch $BEX_\phi(v)$ ist also fast genau so gut wie die Auswahl irgendeiner Kante e_0 die zu v führt. Von dieser Performancegarantie für $BEX_\phi(v)$ wird in [BAR] eine lokale Performancegarantie hergeleitet. Diese besagt, dass die durchschnittlichen Kosten von $\pi_t(w)$ zu den durchschnittlichen Kosten von $\pi_t(v)$ im Zusammenhang stehen, wenn die Knoten v und w durch eine Kante $e = (v, w)$ verbunden sind:

$$\frac{1}{T} \sum_{t=1}^T \mathbf{E}(C_t(\pi_t(w))) \leq \frac{1}{T} \sum_{t=1}^T \mathbf{E}(C_t(\pi_t(v))) + \frac{1}{T} \sum_{t=1}^T \mathbf{E}(C_t(e)) + O\left(\epsilon + \frac{\tau \log(\Delta)}{\epsilon T}\right).$$

Von der lokalen Performancegarantie wird dann eine globale Performancegarantie hergeleitet. π ist dabei ein Pfad von s nach r über die Knoten $s = v_0, v_1, \dots, v_H = r$. π_j gibt den Teilpfad von s nach v_j an. Mittels Induktion über j , gilt dann folgende Performancegarantie:

$$\frac{1}{T} \sum_{t=1}^T \mathbf{E}(C_t(\pi_t(v_j))) \leq \frac{1}{T} \sum_{t=1}^T \sum_{e \in \pi_j} \mathbf{E}(C_t(e)) + O\left(\epsilon j + \frac{\tau \log(\Delta j)}{\epsilon T}\right).$$

In [BAR] wird nun $j = H$ gewählt und die Kosten der „Sampling“-Schritte addiert, die durch δT nach oben beschränkt sind:

$$\mathbf{E}(BCOST(\text{algorithm})) \leq \mathbf{E}(ACOST(\pi)) + O\left(\epsilon H + \frac{\tau \log(\Delta) H}{\epsilon T} + \delta\right).$$

Da τ durch $m \log(mT)/\delta$ nach unten begrenzt ist, werden in [BAR], um die rechte Seite der Gleichung zu optimieren, folgende Werte angegeben: $\epsilon = \left(\frac{m \log(mT) \log(\Delta)}{HT}\right)^{1/3}$, $\delta = \epsilon H$ und $\tau = m \log(mT)/\delta$.

Diese Werte führen dann zu dem am Anfang angegebenen Theorem.

4 Ergebnisse einer Implementierung

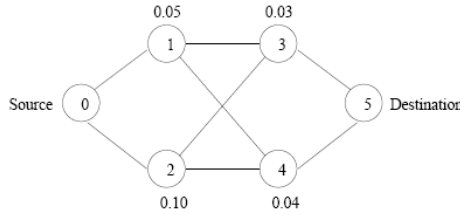


Abbildung 4: Beispiel Netzwerk

In [BAR] wurde ein einfaches Programm getestet, welches das Routingverfahren implementiert. Das simulierte Netzwerk besteht aus 6 Knoten, die von 0 bis 5 bezeichnet werden. Es sollen 10,000 Pakete von Knoten 0 nach Knoten 5 gesendet werden. Ein Gegnermodell entscheidet dabei mit vorgegebenen Wahrscheinlichkeiten, ob ein Knoten im aktuellen Zeitschritt fehlerhaft ist oder nicht. Ein fehlerhafter Knoten würde dann im aktuellen Zeitschritt kein ACK an den Vorgängerknoten schicken und die einkommende Verbindung zum Knoten wird in der nächsten Phase schlechter bewertet. In Abb. 4 sieht man, mit welcher Wahrscheinlichkeit das Gegnermodell einen jeweiligen Knoten stört. Der optimale Pfad führt vom Sendeknoten zu Knoten 1, weiter nach Knoten 3 und dann zum Empfänger. Die Experimente wurden für verschiedene Parameter β durchgeführt. In Abb. 5 sieht man, dass bei kleiner werdendem β sich der Algorithmus auf Veränderungen schneller einstellt. Eine Reduzierung der „Sampling“-Rate auf 0,01 lässt den Algorithmus aber nicht mehr so schnell auf Veränderungen reagieren. Man

sieht deutlich, dass bei den Paketen 6,000 bis 7,000 der optimale Pfad nur noch zu 60% bis 80% genutzt wird.

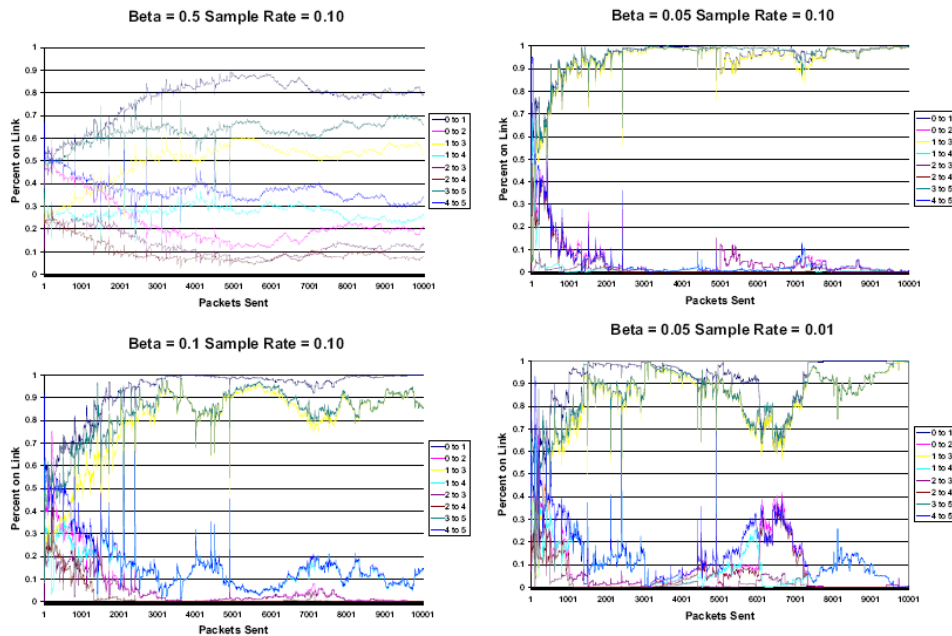


Abbildung 5: Kantenwahlergebnisse

5 Gliederung des Vortrags

- Einleitung/Motivation
- Das Routingverfahren
- Performance des Routingverfahrens
- Ergebnisse einer Implementation
- Fazit

Literatur

[BAR] R. Kleinberg B. Awerbuch, D. Holmer and H. Rubens. Provably competitive adaptive routing. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Miami, USA, March 2005*.