Seminar Routingprotokolle

XL: An Efficient Network Routing Algorithm

Christopher Israel

cisrael@uni-koblenz.de

20. November 2008

Gliederung

Gliederung

Problem beim Link-State Routing

- große Netze erzeugen häufiger Routing-Updates
- es kann schwierig werden, auf jede Änderung zu reagieren

Lösungsmöglichkeiten

- Netz in Routing-Domänen (Areas) unterteilen

Lösungsmöglichkeiten

- Netz in Routing-Domänen (Areas) unterteilen
  Nachteil: es ist Handarbeit notwendig und schwierig, eine optimale Konfiguration zu finden

Lösungsmöglichkeiten

- Netz in Routing-Domänen (Areas) unterteilen
  Nachteil: es ist Handarbeit notwendig und schwierig, eine optimale Konfiguration zu finden
- Routing-Updates (LSA-Nachrichten) selektiv weiterleiten

Lösungsmöglichkeiten

- Netz in Routing-Domänen (Areas) unterteilen
  Nachteil: es ist Handarbeit notwendig und schwierig, eine optimale Konfiguration zu finden
- Routing-Updates (LSA-Nachrichten) selektiv weiterleiten
  Nachteil: nicht immer kürzeste Wege

Was ist XL?

- XL steht für „Approximate Link state“
- ein Link-State Routing Algorithmus, der Netzwerk-Kommunikation minimieren soll
- leitet Updates nicht an alle anderen Knoten im Netz weiter
- effizienter als gewöhnliche Link-State und Distance-Vektor Routing Algorithmen
- benutzt nicht immer die kürzesten Wege

Gliederung

Regeln für das Weiterleiten von Updates

Ein Routing-Update sollte dann weitergeleitet werden:

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

Korrektheit und Vollständigkeit und „Stretch“

- Eine Konfiguration ist korrekt, wenn jeder Eintrag in einer Forwarding-Tabelle auch tatsächlich zum Ziel führt

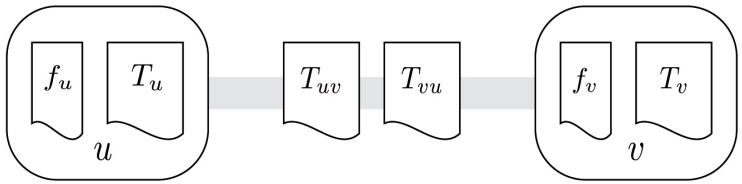
Korrektheit und Vollständigkeit und „Stretch“

- Eine Konfiguration ist korrekt, wenn jeder Eintrag in einer Forwarding-Tabelle auch tatsächlich zum Ziel führt
- Eine Konfiguration ist vollständig, wenn sofern es einen endlichen Pfad gibt, auch Einträge in der Forwarding-Tabelle existieren.

Korrektheit und Vollständigkeit und „Stretch“

- Eine Konfiguration ist korrekt, wenn jeder Eintrag in einer Forwarding-Tabelle auch tatsächlich zum Ziel führt
- Eine Konfiguration ist vollständig, wenn sofern es einen endlichen Pfad gibt, auch Einträge in der Forwarding-Tabelle existieren.
- Der Stretch einer Konfiguration ist das Verhältnis von tatsächlichen Pfadkosten zu optimalen Pfadkosten

Views



- Jeder Knoten hat eine „View“, in der die Kantengewichte des Netzes zu einem Zeitpunkt festgehalten werden.
- Ein Knoten hat dabei eine interne View und für jeden benachbarten Knoten je eine externe View
- in der internen View sind die aktuellen Informationen gespeichert
- in den externen Views werden die Informationen für die jeweiligen Nachbarknoten gespeichert
- das Aktualisieren einer externen View kostet, da dabei ein Update an den jeweiligen Nachbar geschickt wird

Gliederung

Algorithmus

Am Anfang sind in den Views alle Kosten als unendlich vermerkt.
Der Update-Algorithmus berechnet dann die interne und die externen Views, sowie die Forwarding-Tabelle.
Der Update-Algorithmus läuft in drei Phasen ab:

- Phase 1: Berechnen der internen View und der vorläufigen externen Views
- Phase 2: Berechnen des shortest-path trees und der Forwarding-Tabelle
- Phase 3: Berechnen der externen Views

Phase 1: Berechnen der internen View und der vorläufigen externen Views

- eigene externe Views an die der Nachbarn angleichen
- jeweils aktuellere Information aus beiden externen Views in die eigene eintragen
- aktuelle Informationen aus den externen Views und den lokalen Kantengewichten in die eigene interne View einfließen lassen

Phase 2: Berechnen des shortest-path trees und der Forwarding-Tabelle

- shortest-path tree anhand der Informationen aus der internen View berechnen
- identisch zu der Vorgehensweise im gewöhnlichen Link-State Algorithmus
- Forwarding-Tabelle anpassen

Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:

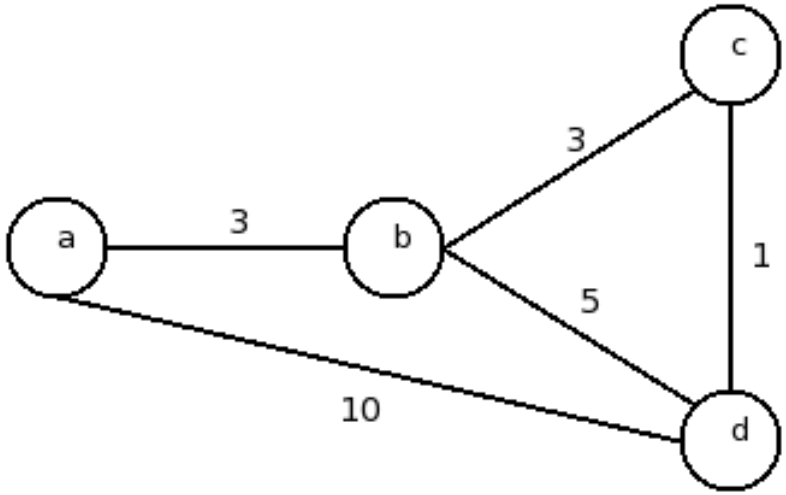
Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:

- eine Verschlechterung eines Links wird in alle externen Views eingetragen
  - Wenn das Update eine Erhöhung der Kosten bedeutet

Phase 3: Berechnen der externen Views

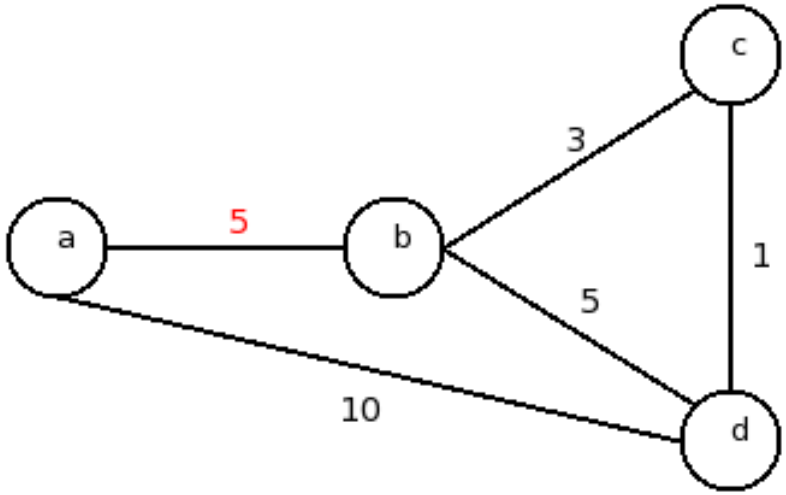
Die externen Views erhalten anhand der drei Regeln neue Informationen:



- eine Verschlechterung eines Links wird in alle externen Views eingetragen
  - Wenn das Update eine Erhöhung der Kosten bedeutet

Phase 3: Berechnen der externen Views

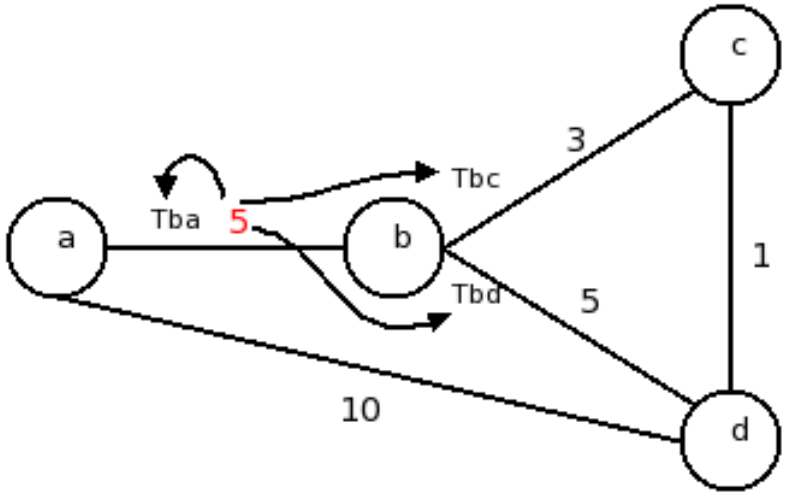
Die externen Views erhalten anhand der drei Regeln neue Informationen:



- eine Verschlechterung eines Links wird in alle externen Views eingetragen
  - Wenn das Update eine Erhöhung der Kosten bedeutet

Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:



- eine Verschlechterung eines Links wird in alle externen Views eingetragen
  - Wenn das Update eine Erhöhung der Kosten bedeutet

Phase 3: Berechnen der externen Views

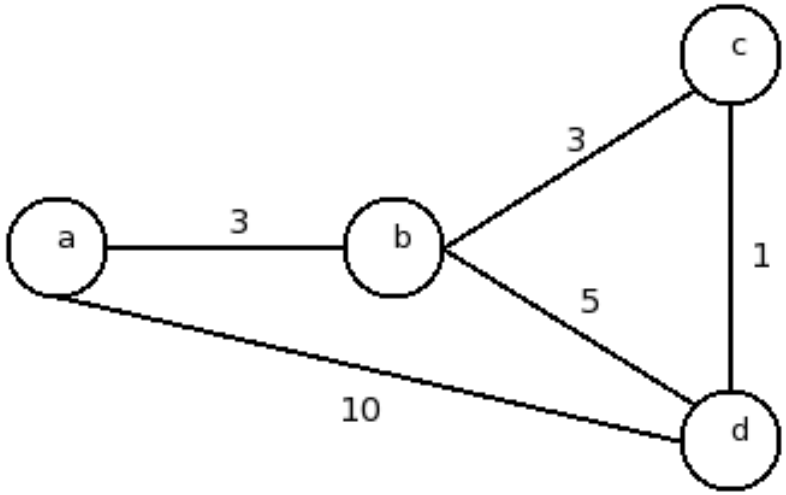
Die externen Views erhalten anhand der drei Regeln neue Informationen:

- eine Verschlechterung eines Links wird in alle externen Views eingetragen
- wenn der kürzeste Weg zu einer Kante durch einen Nachbarknoten führt, erhält der Nachbarknoten die neuen Informationen über alle Kanten des Weges

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)

Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:

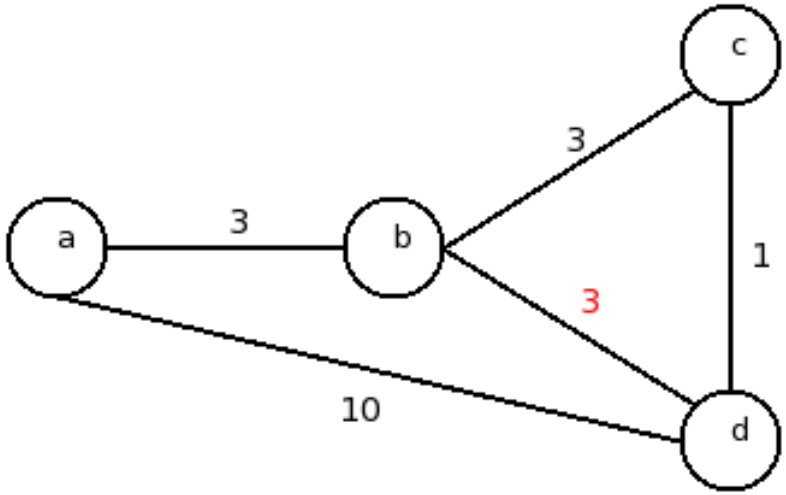


- eine Verschlechterung eines Links wird in alle externen Views eingetragen
- wenn der kürzeste Weg zu einer Kante durch einen Nachbarknoten führt, erhält der Nachbarknoten die neuen Informationen über alle Kanten des Weges

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)

Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:



- eine Verschlechterung eines Links wird in alle externen Views eingetragen
- wenn der kürzeste Weg zu einer Kante durch einen Nachbarknoten führt, erhält der Nachbarknoten die neuen Informationen über alle Kanten des Weges

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)

Phase 3: Berechnen der externen Views

Die externen Views erhalten anhand der drei Regeln neue Informationen:

- eine Verschlechterung eines Links wird in alle externen Views eingetragen
- wenn der kürzeste Weg zu einer Kante durch einen Nachbarknoten führt, erhält der Nachbarknoten die neuen Informationen über alle Kanten des Weges
- ...

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

Phase 3: Berechnen der externen Views

- Wenn in einer externen View eine Kante auf einem Pfad liegt, dessen Gewicht um einen Faktor $1 + \epsilon$ größer ist als das minimale Gewicht zum gleichen Ziel, so ist diese Kante „heiß“

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

Phase 3: Berechnen der externen Views

- Wenn in einer externen View eine Kante auf einem Pfad liegt, dessen Gewicht um einen Faktor $1 + \epsilon$ größer ist als das minimale Gewicht zum gleichen Ziel, so ist diese Kante „heiß“
- Es werden alle heißen Kanten aktualisiert

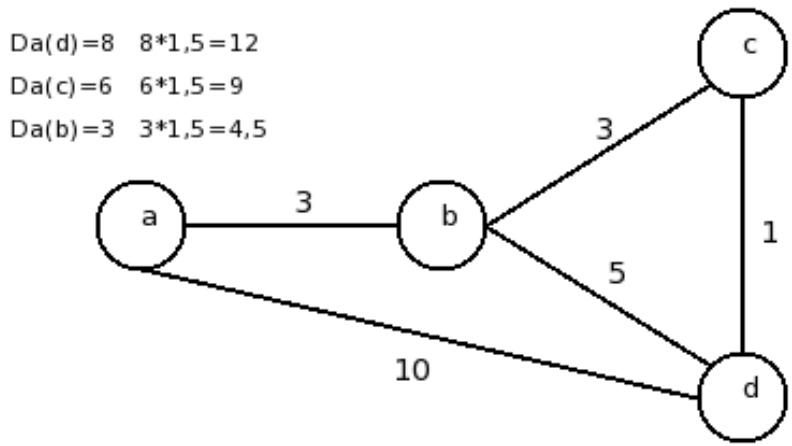
- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

Phase 3: Berechnen der externen Views

- Wenn in einer externen View eine Kante auf einem Pfad liegt, dessen Gewicht um einen Faktor $1 + \epsilon$ größer ist als das minimale Gewicht zum gleichen Ziel, so ist diese Kante „heiß“
- Es werden alle heißen Kanten aktualisiert
- Man benötigt eine „Minimum Distance Proxy Function“

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

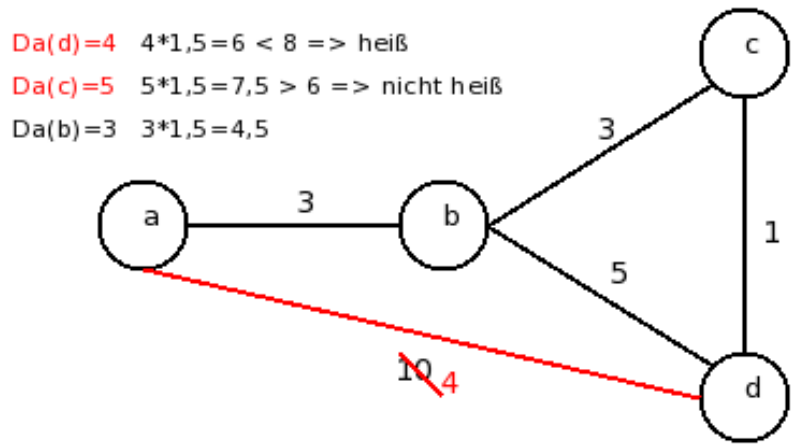
Phase 3: Berechnen der externen Views



- Wenn in einer externen View eine Kante auf einem Pfad liegt, dessen Gewicht um einen Faktor $1 + \epsilon$ größer ist als das minimale Gewicht zum gleichen Ziel, so ist diese Kante „heiß“
- Es werden alle heißen Kanten aktualisiert
- Man benötigt eine „Minimum Distance Proxy Function“

- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

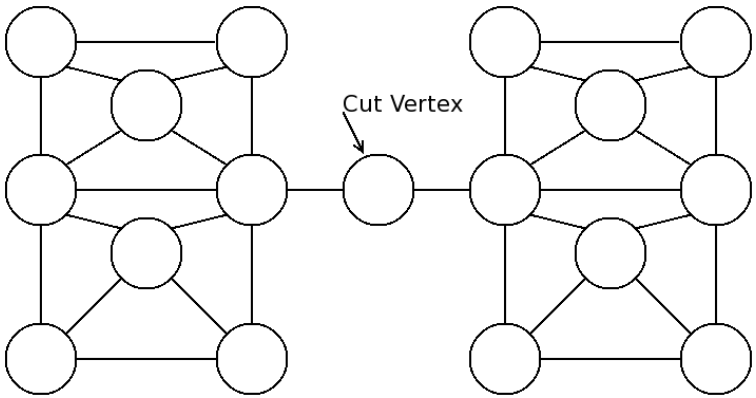
Phase 3: Berechnen der externen Views



- Wenn in einer externen View eine Kante auf einem Pfad liegt, dessen Gewicht um einen Faktor $1 + \epsilon$ größer ist als das minimale Gewicht zum gleichen Ziel, so ist diese Kante „heiß“
- Es werden alle heißen Kanten aktualisiert
- Man benötigt eine „Minimum Distance Proxy Function“

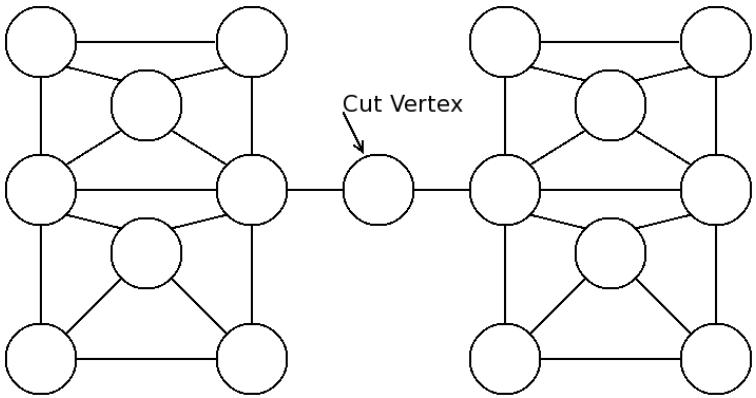
- Wenn das Update eine Erhöhung der Kosten bedeutet
- Wenn der Link im shortest-path tree des Knotens verwendet wird (nur an den nächsten Knoten weiterleiten)
- Wenn es die Kosten zu einem beliebigen Ziel um einen Faktor größer als $1 + \epsilon$ verbessert

Cut Vertex Partitioning



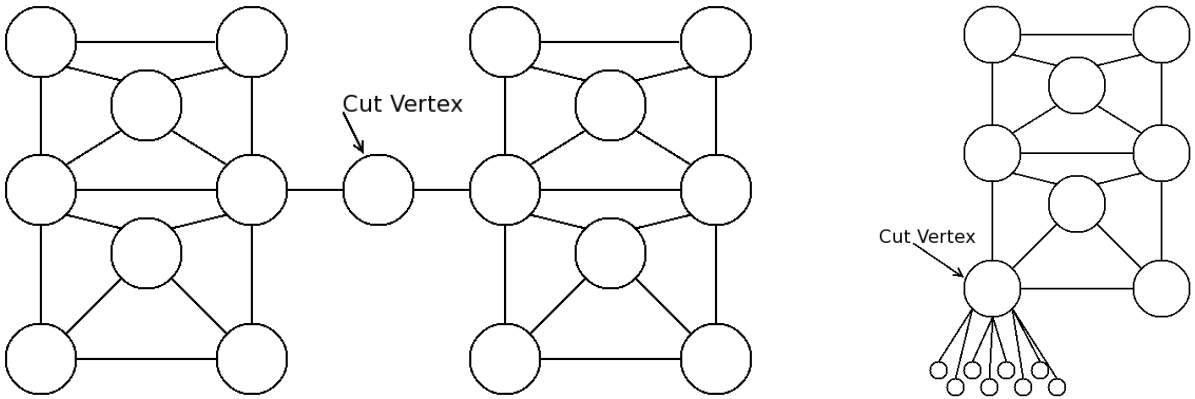
- Knoten, der ein Netz in mehrere Teilnetze unterteilt

Cut Vertex Partitioning



- Knoten, der ein Netz in mehrere Teilnetze unterteilt
- Ein Knoten in einem Teilnetz muss nichts von den anderen Netzen wissen, sondern Pakete nur zum Cut-Vertex weiterleiten

Cut Vertex Partitioning



- Knoten, der ein Netz in mehrere Teilnetze unterteilt
- Ein Knoten in einem Teilnetz muss nichts von den anderen Netzen wissen, sondern Pakete nur zum Cut-Vertex weiterleiten
- Reale Netzwerke haben keine CVs, die Netze in Teilnetze unterteilen, sondern CVs, die Netze von Hosts trennen

Gliederung

Simulationsergebnisse

Der Algorithmus wurde in einer Simulation gegen andere Routing-Algorithmen getestet:

`ls`	The standard link-state algorithm [22] which is the basis for OSPF and IS-IS.
`dv`	A distance vector algorithm very similar to RIP [20] with split horizon. The maximum distance bound is a global parameter of the algorithm.
`dv+p`	A modern distance vector algorithm which uses a parent pointer to detect loops [4, 12, 28].
`lv`	The Link Vector algorithm proposed by Behrens and Garcia-Luna-Aceves [3].
`xl`	The XL algorithm described in this paper, parametrized by error ϵ. When $\epsilon = 0$, all forwarding paths are optimal just as with the above algorithms.

Anzahl der Nachrichten

	Standard model					Flapping model				
	`dv`	`dv+p`	`lv`	`xl`		`dv`	`dv+p`	`lv`	`xl`	
CROWN 64	3.41	1.07	1.06	0.68	0.46	1.09	0.79	0.78	0.49	0.17
H. 16 × 16	1.09	0.73	0.68	0.35	0.23	0.42	0.71	0.64	0.24	0.09
Q. 16 × 16	0.16	0.45	0.43	0.18	0.14	0.12	0.44	0.42	0.10	0.07
ABILENE	0.97	0.77	0.77	0.64	0.55	0.98	0.83	0.83	0.55	0.46
ARPANET	2.28	0.91	0.89	0.51	0.45	1.86	0.89	0.87	0.39	0.28
FUEL1221	7.32	0.46	0.46	0.12	0.09	6.56	0.44	0.43	0.10	0.05
FUEL1239	4.85	0.23	0.23	0.20	0.11	1.16	0.21	0.21	0.16	0.05
F. 1221C	0.74	0.38	0.38	0.37	0.26	0.34	0.35	0.36	0.30	0.16
F. 1239C	0.95	0.22	0.22	0.22	0.11	0.20	0.22	0.21	0.17	0.05

Table 5: Average (over 10 simulations) of the maximum number of messages generated by any one node, relative to `ls`. The `xl` columns shows values for algorithm parameters $\epsilon = 0.0$ (first value) and $\epsilon = 0.5$ (second value).

Konvergenzzeit

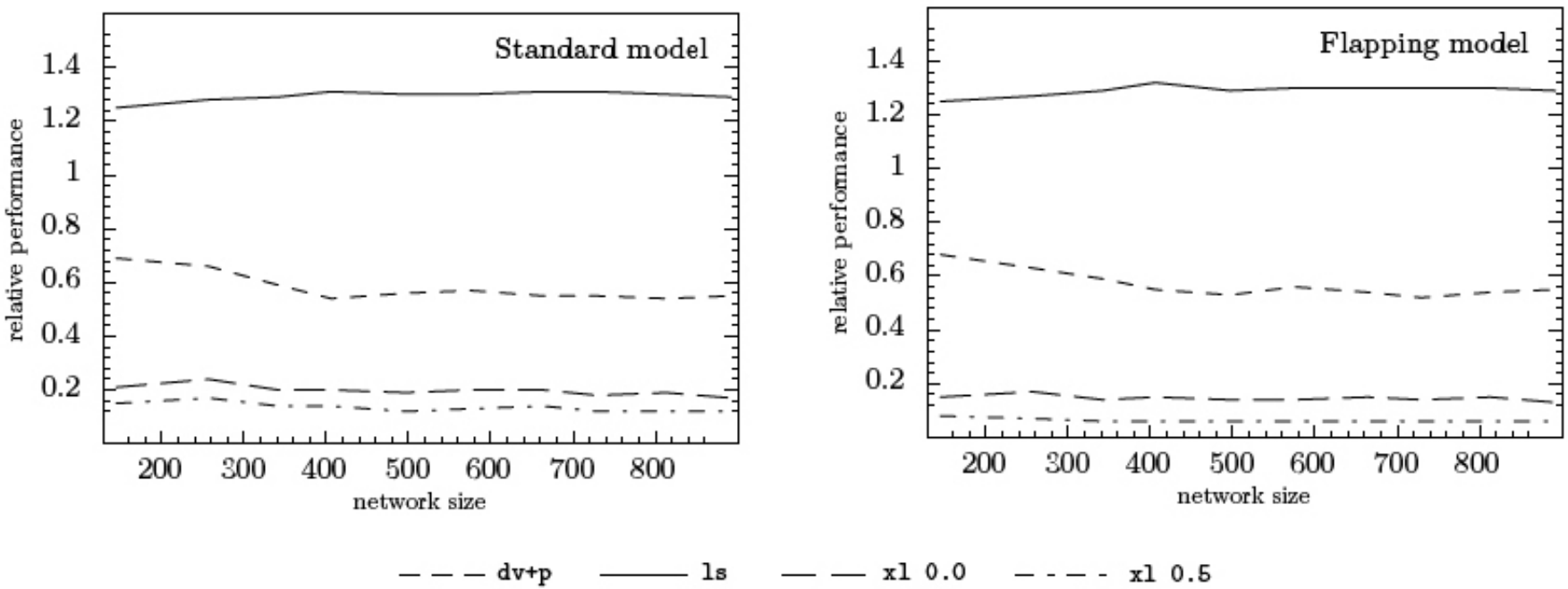
	Standard model					Flapping model				
	dv	dv+p	lv		xl	dv	dv+p	lv		xl
CROWN 64	4.08	0.00	0.00	1.04	0.88	9.28	0.00	0.00	1.17	0.66
H. 16 × 16	17.19	0.00	0.00	0.99	0.88	1.49	0.00	0.00	0.90	0.80
Q. 16 × 16	5.96	0.00	0.00	1.00	0.98	1.24	0.00	0.00	1.16	1.03
ABILENE	2.27	0.00	0.00	0.79	0.87	1.83	0.00	0.00	0.93	0.98
ARPANET	3.12	0.00	0.00	0.91	0.82	2.86	0.00	0.00	0.94	0.82
FUEL1221	74.23	0.00	0.00	0.79	0.79	46.01	0.00	0.00	0.79	0.81
FUEL1239	85.64	0.00	0.00	0.92	0.87	24.87	0.00	0.00	0.95	0.85
F. 1221C	10.80	0.00	0.00	0.87	0.85	2.60	0.00	0.00	0.96	0.95
F. 1239C	25.12	0.00	0.00	0.95	0.86	2.24	0.00	0.00	0.99	0.85

Table 7: Forwarding loop duration maximum over all source-destination pairs, relative to `ls`. The forwarding loop duration for a pair of nodes u and w is the duration of time $\phi(u, w)$ was infinite.

	Standard model					Flapping model				
	dv	dv+p	lv		xl	dv	dv+p	lv		xl
CROWN 64	2.58	2.74	2.73	1.54	1.74	5.29	5.44	5.37	1.45	1.41
H. 16 × 16	1.19	3.08	2.46	1.10	1.09	1.30	4.85	3.12	1.02	0.93
Q. 16 × 16	1.10	2.54	2.00	1.03	1.03	1.02	2.92	2.12	0.99	0.99
ABILENE	1.25	1.41	1.41	1.05	1.14	1.36	1.55	1.56	1.01	1.02
ARPANET	1.29	1.41	1.34	0.95	0.94	1.20	1.48	1.46	0.96	0.89
FUEL1221	1.04	1.15	1.09	0.60	0.63	1.06	1.16	1.14	0.52	0.52
FUEL1239	1.15	1.44	1.36	0.75	0.76	1.04	1.24	1.22	0.74	0.70
F. 1221C	1.16	1.38	1.36	1.03	1.09	1.33	1.62	1.41	1.00	0.98
F. 1239C	1.54	1.76	1.57	1.05	1.03	1.50	1.70	1.63	1.01	0.93

Table 8: Maximum duration of infinite forwarding-to-optimal distance ratio relative to `ls`. The maximum is taken over all source-destination pairs. The infinite forwarding to optimal distance ratio duration for a pair of nodes u and w is the duration of time when $\|\phi(u, w)\|$ was infinite but $\delta(u, w)$ was not.

Skalierbarkeit



Stretch

	Standard model			Flapping model		
	Med	Avg	Max	Med	Avg	Max
CROWN 64	1.00	1.02	1.43	1.00	1.01	1.39
H. 16×16	1.00	1.05	1.45	1.00	1.02	1.44
Q. 16×16	1.00	1.02	1.43	1.00	1.01	1.40
ABILENE	1.00	1.01	1.22	1.00	1.01	1.18
ARPANET	1.00	1.02	1.45	1.00	1.01	1.41
FUEL1221	1.00	1.01	1.34	1.00	1.01	1.33
FUEL1239	1.00	1.04	1.41	1.00	1.02	1.41
FUEL1221C	1.00	1.02	1.35	1.00	1.01	1.33
FUEL1239C	1.00	1.04	1.42	1.00	1.02	1.41

Table 6: Top centile stretch for `x1` with parameter $\epsilon = 0.5$. The median, average, and maximum of the top centile were taken over all source-destination pairs; a pair's instantaneous stretch is at most its top centile value 99% of the time.

Gliederung

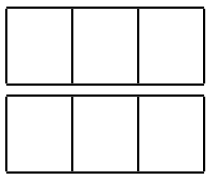
OSPF/XL

- OSPF und XL sind kompatibel
- Interne Views sind bereits vorhanden, Externe Views nicht
- eintreffende Updates müssen in der externen View des Knotens eingetragen werden, von dem sie gekommen sind
- neue Informationen müssen in der internen View eingetragen werden
- eine Proxy Minimum Distance Funktion muss eigeführt werden

Gliederung

Fazit

- durch Approximate Link State wird die Netzwerk-Belastung durch Link-State-Pakete reduziert
- größere Netze können ohne Handarbeit verwaltet werden
- die Wege sind nicht immer optimal
- der Speicherbedarf ist höher als bei anderen Routing-Algorithmen
- eine tatsächlich verwendbare Implementation existiert noch nicht



Fragen?