Survey of Research towards Robust Peer-to-Peer Networks: Search Methods

John Risson and Tim Moors

2004/11/04 IDEON camp memo

1 概要

1.1 位置付け

P2P の定義: Self-organization, symmetric communication and distributed control (Rousopoulos, Baker, et al. 2004)

P2P と言っても 5,800 publications ぐらい存在する (Citeseer)。Search, Storage, Security, Applications に分けた上で、この論文では Search に集中している。そもそも、この論文の question が、“How robust are P2P search networks?” である。

P2P routing は structured と unstructured にわかれる (yd: 昔は pure/hybrid/server-client と言ってた)。Structured network は routing overhead(traffic) において有利で、かつ存在するデータに関しては高い確率でデータに到達可能。他方 structured network はノードが頻繁に参加・離脱を行う場合や、ネットワークの条件が良好な場合しか動作しない、という批判もある。もちろん、complex query の問題もある。

Index は全ての鍵。Local Index (各ノードの中だけで保持) は衰退しつつある。centralized は Napster が負けたので衰退したように見えるけど、Google やら Yahoo やらの存在は、Centralized Index の可能性を暗示している。distributed index はキーワード検索できない所が難しい。

そんなわけで、section3 以降は Index の中でも Semantic-Free Index と Semantic Index、そしてその上に構築される Search について書いてある。

1.2 Semantic-Free Index

DHT の性質: a) low degree b) low diameter c) greedy routing d) robustness。基本は Plaxton Trees (axton, Rajaraman et al. 1997) か、Consistent Hashing (Karger, Lehman et al. 1997) にある。SDDS の論文 (Litwin, Niemat et al. 1993) は見落されていた。

Plaxton Tree の原理は、Pastry / Tapestry / Chord / Ocenastore に影響を与えている (yd: Chord は正直疑問?)。全ての node が spanning tree の根になり、検索は根に向って経路付ける (固定長の) 経路表に基いて行われる。最初の Plaxton Tree 自体は、global knowledge が必要とか、根が一点障害になる、node の追加・削除ができない、hot spot における混雑を避ける手段が無い、といった批判 (Zhao, Kubiatowicz et al. 2001) もある。

Consistent Hashing は、client と server 間で同じ hash 関数を用いる方式 (Chord は違うの?) で、LH*という方式が代表例として紹介されている。LH*は Chord とよく似ているが、Chord は insert 時にすぐ node を分割するのに対して、LH*は insert 時に node の分割を予約するだけで実行しない、といった差がある。また、Chord

はO(log N) hops必要なのに対して、LH*はtwo hopのみ必要。LH* serverは N/2 の経路表を持つ (clientは1から N/2 の間)。

比較の問題: 比較は *fault-free performance* / *static dependability* (リカバリアルゴリズムが走る前) / *dynamic dependability* (主に規模障害時) の三つに分類できる。

整理 (A: 比較項目, B: 対象, C: 結論, D:手段)

- Fauit-free performance
  - (Christin and Chuang 2003; Christin and Chuang 2004)
    - A: latency, service, routing, and mantainance costs
    - B: De Bruijn, D-dimentional tori, Plaxton tree and Chord ring
    - C: 明確な勝者無し
  - (Rhea, Roscoe et al. 2003)
    - A: performance (not robustness)
    - B: Tapestry and Chord
    - C: ファイルが小さい場合には、Tapestry が有利 (recursive vs. iterative?)
- Static dependability
  - (Gummadi, Gummadi et al. 2003)
    - A: ?
    - B: tree, hypercube, butterfly, ring, XOR, hybrid geometries
    - C: 不明
  - (Loguinov, Kumar et al. 2003)
    - A: diameter and connectivity
    - B: Chord, CAN, de Bruijn graph
    - C: de Bruijn が最も良い
  - (Hsiao and King 2003)
    - A: adaptability (number of alternatives for next hop)
    - B: Chord, Tapestry, Pastry, Tornado
    - C: 不明
- Dynamic dependability
  (Static dependability はそれぞれの node に state がたくさんある方式に有利)
  - (Li, Stribling et al. 2004)
    - A: lookup latency against avarage B/W (bytes/nodes/sec)
    - B: Chord, Kelips, Tapestry, Kademlia (文中Kadlemiaとあるがtypo？)
    - C: 低帯域において Chord 有利 (on latency)
    - D: p2psim を利用

formal analysisについては中身を読まないと結論も何もわからないので省略。Chordについての研究が2つもあるのが印象的。また、API の話は 3 つ (Awerbuch and Scheideler 2003; Dabek Zhao et al. 2003; Huebsch, Hellerstein et al. 2003) ある。

Pastry: Tradeoff: reliablity vs. maintainance cost. Churn を生き残れる実装は皆無？MSPastry → less than 0.5 msg/s/node を達成。

Chord: peer の行動を観測しないと optimum stabilization rate はわからない (Liben-Nowell, Balakrishnan et al. 2002) かも。DKS (Alima, El-Ansary et al.2003) は、lookup等のメッセージに stabilization を piggyback させる事で、lookup が correct に行われるよう保証する。ただし、node arrival/departure rate に制約あり。

Butterflies: node failure時の考察が不十分。

de Bruijn: better than greedy routing. N=1 million で degree=20 の時、Chord の直

径は 20 程度だが、de Bruijn の直径は 5 程度。D2B, Koorde, Distance Halving, ORDI が 2003 年に出てきた。

Skip Graph: データ配置に Locality の概念を導入可能な、確率的なアプローチ。ただし、ノード内に保持するコストが大きく、メンテナンストラフィックが多いと考えられる点と、システム内の負荷分散が正しく動作するか不明な点が問題として挙げられている (Aspnes, Kirsch et al. 2004)。

1.3 Semantic Index

名前と content の内容に関連がある (そういう意味では Skip Graph は半分 Semantic だが)。heuristics に頼る傾向があり、存在する情報が発見できるとは限らない。

関連する研究分野は (1)RDBMS (2)Distributed File Systems/CDN。Semantic Index にとって、**(c) 非常に多くの、動的で地理的に分散した、低コストノード間での (a) そこそこの水準のデータ独立性、一貫性、問い合わせの柔軟性と (b) 確率的な問い合わせへの答え**を実現する事がチャンス。

種類:

- Gnutella 系な Keyword Lookup とその子孫 (QRT: Singla and Rohrs 2002) 他
  - 動的環境における堅牢性評価: (Yang, Vinograd et al. 2004)
  - 静的環境における評価: (Cholvi, Felber et al. 2004)
  - inverted list によるキーワード検索: (Gnawali 2002; Reynolds and Vahdat 2003) → MLP(Shi, Guangwen et al. 2004) は SkipNet 上に実装されている。
- Information Retrieval → 文書の表現、問い合わせの表現、問い合わせと文書の関連付けの枠組、問い合わせに対する文書の評価 (ranking) の 4 つで構成される
  - 大規模環境下における Ranking が鍵 (Daswani, Garcia-Molina et al. 2003)
  - P2P の文脈では、Vector Model と Latent Semantic Indexing が主流。Bayesian model はあまり多くない模様。
  - reliability や robustness の話題は少ない一方、再現率や正確性の話題は多い。
  - FIXME: 具体的な技術
- Peer Data Management Systems(PDMS) (Tatarinov and Halevy 2004)
  - Peer 毎に Schema 等のモデルを用意して、Semantic Path を辿る (?)
  - Figure 2: 最初の 6 つが、Information Integration という問題に挑戦している。
  - Global Schema 問題を避けつつ、いろいろ工夫している模様。
  - FIXME: 具体的な技術

1.4 Search

FIXME

1.5 結論

乱暴に言うと:

1. Storage, Security, application の調査が必要
2. Indexing はほぼ成熟している。query はまだまだ

3. 個別の機構の信頼性を評価しないと、意味ないよね
4. 関連研究分野をちゃんと統合して行こう
5. 信頼性の尺度と許容範囲について合意が必要

2　コメント

結局今回は時間の関係から、Semantic-Free Index の話に限定してしまった。世の中の流れに追いつくのがこれほど大変とは思わなかった。

スケーラブルなアーキテクチャは、綺麗にレイヤリングした上で、下層 (Semantic-Free Index) を共有化しつつ、いかに「薄く広く」作っていくかという勝負だと考えている。

ところで、fixed-degree DHTって本当に良いものなんでしょうか……。