Proof of concept のその先に
～ オーバレイネットワークの実際 ～

首藤 一幸

内容

- 自己紹介
  - 自身, 会社
- オーバレイネットワーク
  - unstructured / structured オーバレイ
- Proof of concept のその先に
  - Overlay Weaver (structured)
  - オーバレイマルチキャスト (unstructured)

首藤一幸 (34)

◆略暦
- ▪1998年 早稲田の助手
- ▪2001年 博士課程修了
- ▪2001年 産業技術総合研究所
- ▪2006年 ウタゴエ

- ソフトウェアエンジニア, 技術フェチ
- 研究職から (いわゆる) ベンチャーへ。
- これまでの仕事 のうち、わかりやすいもの
  - Java JITコンパイラ shuJIT (1998〜)
  - P2P ミドルウェア Overlay Weaver (2005〜)
  - 書籍 Binary Hacks (2006.11) 執筆
    - with 高林 (グーグル(株)) 鵜飼 (グーグル(株)) , 佐藤, 浜地 (グーグル(株)) , …
  - カラオケグリッド: 全世界カラオケセッション (2003.11)
    - Access Gridを使った、5カ国 2X拠点のセッション。
  - AES候補 (暗号) の高速な実装 with NTT (1998)

ウタゴエ

- 技術中心のソフトウェア開発企業
- メンバ 20名前後
- 2001年 1月 創業
  - 園田 (社長) 博士課程 1 年目
- 2007年 11月 San Jose オフィス
- 歌声：コミュニケーションのシンボル

プロダクト, サービス

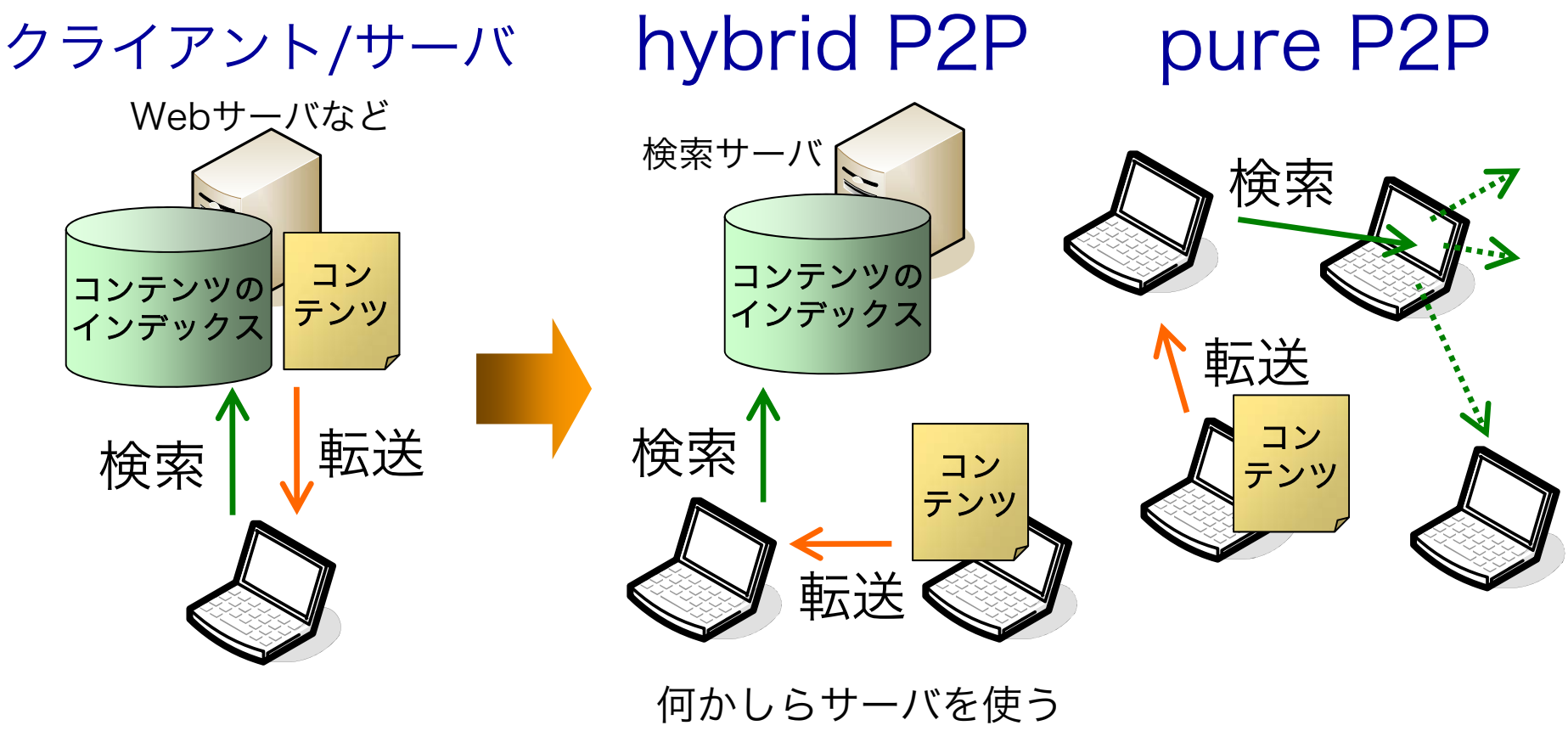
- うたごえ検索 (1997〜)
  - タタタ〜と歌って楽曲検索。携帯に内蔵。
- Ocean Grid (Casting Grid) (2005〜)
  - Peer-to-peer ライブ配信ソフト。
  - IPA未踏ソフト 2006年上期 池長・首藤プロジェクトの成果を採り入れた。
-
-
- …

オーバレイネットワーク

unstructured / structured overlay

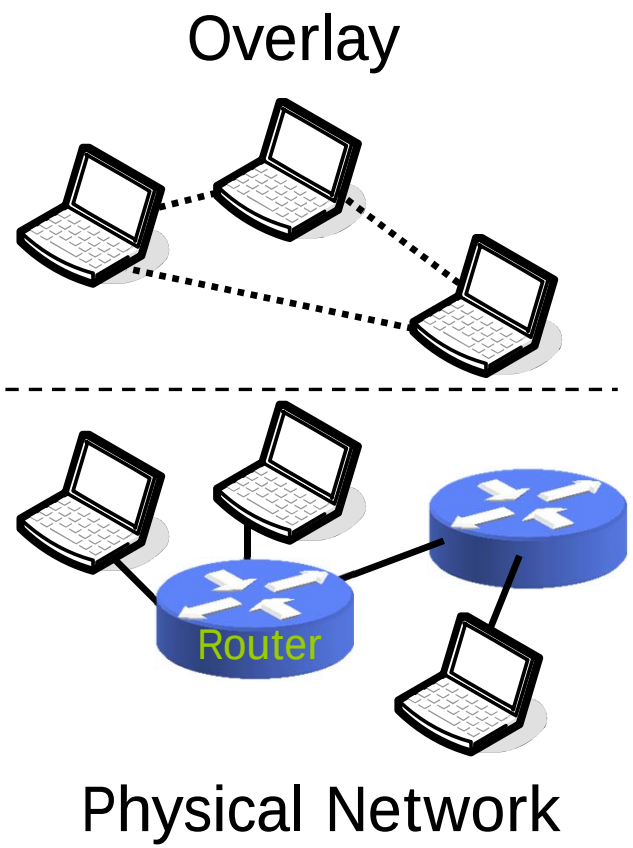
前置き: hybrid / pure P2P

- P2P ファイル共有ソフトの例



オーバレイネットワーク (overlay network)

- 下位ネットワークとは独立した
  トポロジを持つ上位ネットワーク
  - UUCP, NetNewsサーバのトポロジ
  - 電話網上のインターネット
  - インターネット上のVPN, CDN
  - MBone, 6bone, …
  - GENI
    - 米国 NSF の次世代ネットワーク研究プロジェクト
    - 研究者は、自身の “net” を構築
    - PlanetLab
  - (pure) P2P
    - 特徴: 多ノード/低管理コスト
      → 自律的にオーバレイを構成



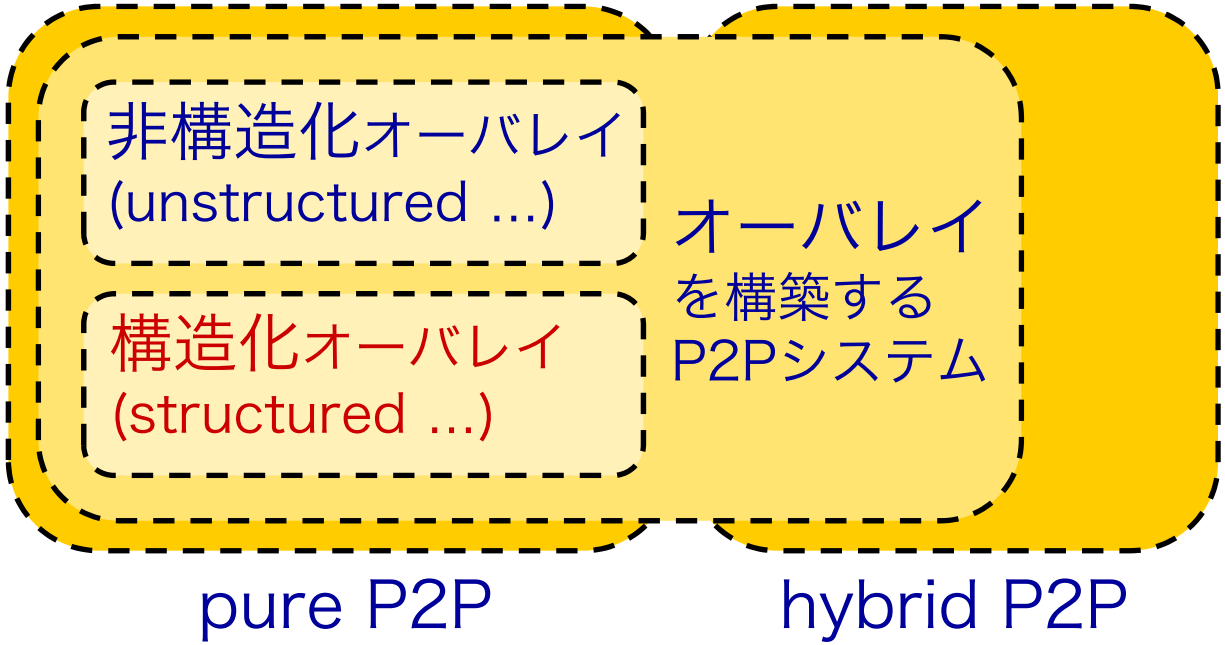
オーバレイとP2P

- (pure) P2P → 非集中 (decentralized)
  - 集中サーバなしに何らかの機能を果たすため、アプリケーションレベルのネットワークを構成する。
    - 例: ファイル共有ソフトのネットワーク
      FastTrack, Gnutella, … ← ノード数 100万以上
    - 機能: 発見, マルチキャスト, メッセージ配信, …
  - そのトポロジはインターネットの論理的 / 物理的トポロジとは独立
    → オーバレイネットワーク
  - 特徴: ノード数が多い (〜数百万) and/or 管理の手間をかけられない。
    → 自律的に構成する。

オーバレイとP2P

- pure P2P → 要オーバレイ
  hybrid P2P → オーバレイを構築したりしなかったり
  - アプリケーション層マルチキャスト (ALM) は、
    hybrid P2Pであってもオーバレイを構築する。

- pure P2Pシステムが構築するオーバレイには、２種類、
  非構造化 (unstructured) と構造化 (structured) がある。



Unstructured と Structured

- Unstructured オーバレイ
  - 例： Gnutella ネットワーク, Winnyネットワーク
  - 誰を隣接ノードとするか、トポロジに制約がない。
  - 存在するオブジェクトは、発見できる可能性がある。
  - 一般に、効率は良くないが、柔軟な検索が可能。

- Structured オーバレイ
  - 例： DHT（分散ハッシュ表）のネットワーク
  - 誰を隣接ノードとするか、トポロジに制約がある。
  - 存在するオブジェクトは、（たいてい）発見できる。
  - 一般に、効率は良いが、柔軟な検索が苦手。

Unstructured と Structured

- 当初、pure P2P のファイル共有ネットワークは unstructured オーバレイだった。
  - Gnutellaネットワーク, Winnyネットワーク
- 最近は、structured オーバレイの応用も始まっている。
  - eDonkey2k (hybrid P2Pプロトコル) から派生した Kad Network。eMule 等が実装。
  - BitTorrent、Azureus のトラッカーなし動作
    - Azureus: 100万以上のノードでオーバレイを構成?

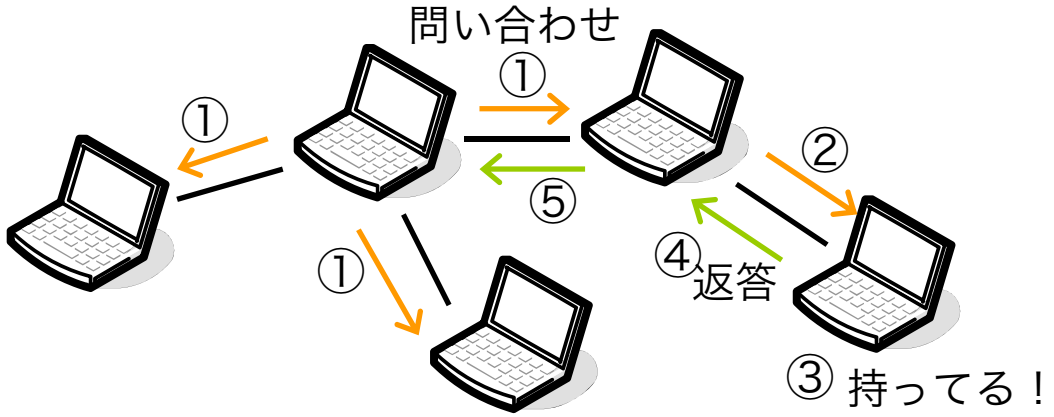
Unstructured と Structured

- Unstructured オーバレイ
  - ファイル共有の Gnutella プロトコル (0.4)

- Structured オーバレイ（アルゴリズム）
  - ルーティングアルゴリズム
    - Chord, Pastry, Kademlia, …
  - の上に、より高レベルなサービス
    - 分散ハッシュ表 (DHT), マルチキャスト, …

Unstructuredオーバレイの例:
Gnutella プロトコル (0.4)

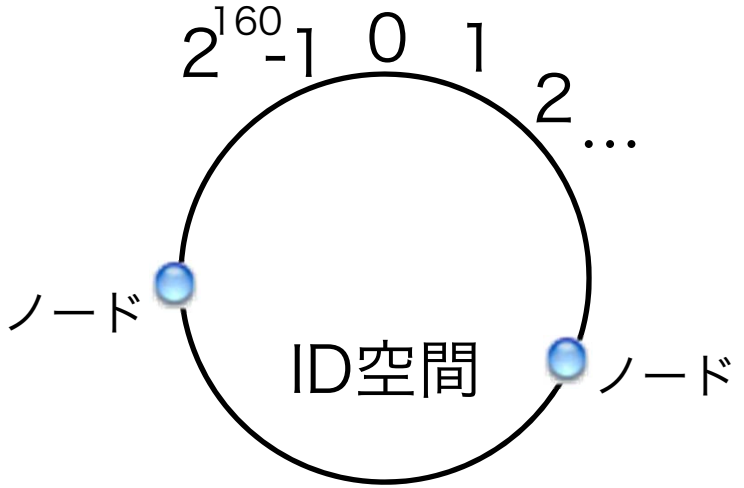
- 各ノードは、一定数の隣接ノード (neighbor) を持つ。
  - どのノードを隣接ノードとするか、および、隣接ノードの数はアプリ依存。例えば 4。
- 検索時、各ノードは問い合わせを全隣接ノードに転送する → flooding
  - TTL（time-to-live, 最大ホップ数）は 7。
  - 返答は、転送経路に沿って返される。

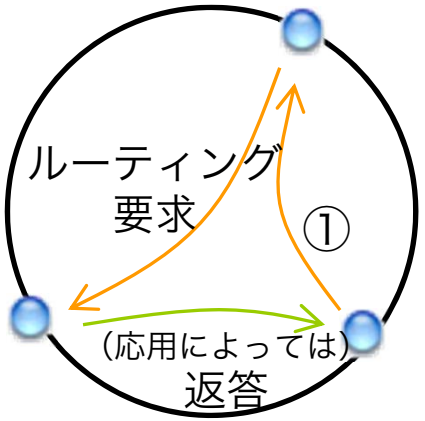


- Gnutella プロトコル 0.6 は、super-peers の概念を採り入れた。

Structured オーバレイの基本

- ノード（計算機）とオブジェクトの両方にIDが振られる。
  - IDはたいてい整数値。160ビットだったり 128ビットだったり。
  - オブジェクト：任意の文字列だったり、ファイルだったり、プロセスだったり。
- ノードは、ID空間中のある範囲を受け持つ。
  - だいたい、ノードのIDと数値的に近い範囲を受け持つ。
- IDを宛先としてルーティングが行われ、その行き着く先は、そのIDを受け持つ担当ノードとなる。

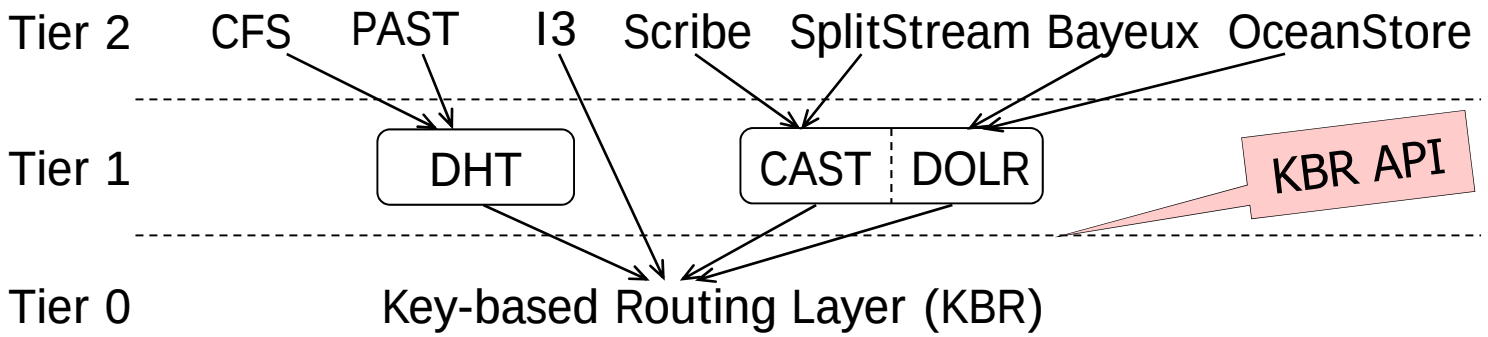




Structured オーバレイの基本

- 肝はルーティング（アルゴリズム）
- 資源にかかる負担が、ノード数 n として O(log n)。
  - ルーティング時のメッセージ数など。
  - cf. unstructured オーバレイ上の flooding
- 様々なアルゴリズムが提案されてきた。
  - CAN, Chord, Tapestry, Pastry, Kademlia, Koorde, Broose, Accordion, ORDI, DKS, D2B, Symphony, Viceroy, …

- Structured オーバレイ上に、いろいろなサービスが載る。
  - 分散ハッシュ表 (DHT), マルチキャスト, メッセージ配送, …



Cited from [Dabek03]

分散ハッシュ表 (DHT)

- Structured オーバレイ上に載るひとつのサービス。pure P2P な DB。ハッシュ表。
  - put (key, value)
  - get (key)
  - キー・値ペアの削除

- 処理
  - put: keyをキーとしてルーティングを行い、担当ノードにキー・値ペアを保持させる。
  - get: keyをキーとしてルーティングを行い、担当ノードが保持している key に対応する値をもらう。

  - 注) key が ID になっていない場合、ハッシュ値を求めて ID にしておく。例えば（暗号学的ハッシュ関数） SHA1 を通すと、160ビットの ID を得られる。

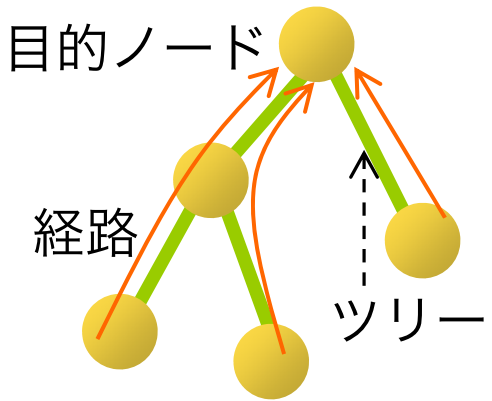
分散ハッシュ表 (DHT)

- 応用: もろもろの名前解決や位置解決
  - ホスト名 → IPアドレス, 名前 → 電話番号, 曲名 → 楽曲ファイルやそのURL などなど。
  - DNS を作ってみました、という研究は多い。
    - “A Comparative Study of the DNS Design with DHT-Based Alternatives”, Proc. INFOCOM 2006.

- 弱点: (そのままでは) 範囲検索ができない。
  - 例: 9月1日〜3日の…, 経度緯度が…
  - 範囲検索が可能な構造化オーバレイ: Skip Graph
  - DHT上で範囲検索、という研究も多い。
    - 中台さん (NEC) は Overlay Weaver上に範囲検索を実装。

Structured オーバレイ上のアプリケーション層マルチキャスト(ALM)

- ある ID に対して複数のノードからルーティングすると、それら経路の集合はツリーを成す。
  これを配送木として利用する。
- チャネルが ID で表されるので、多チャネル。
  ≠ ブロードキャスト
- 処理
  - あるチャネルにsubscribeするノードは、チャネルの識別子をキーとしてルーティングをする。
  - 経路上のノードは、1ホップ手前と1ホップ先のノードを、それぞれツリーの子と親として記憶する。
  - 各ノードが、ツリー上の親子関係に沿ってトラフィックを転送する。



マルチキャストの応用

- 同報通信を行うもろもろの応用
  - グループチャット・音声 / ビデオ会議
  - 放送
  - VPN
    - L2 のブロードキャストに使える?
    - 例: x-kad: Kademliaというアルゴリズムを応用した VPN
  - 分散処理
    - コードの配布やプロセッサ間同期、ブロードキャスト

- 配送木を使って、エニーキャストも可能。
  - グループ中のノードどれかにメッセージ配信
  - 応用：サーバ群の負荷分散

Structured オーバレイの（ルーティング）アルゴリズム

- 宛先 ID に（数値的に）より近い ID を持つノードに、要求をまわしていく。
  - いつかは最も近い ID を持つノードに辿り着く。

- 近づき方の、アルゴリズムごとの違い
  - Chord
    - ID 空間を時計まわりに近づいていく。
  - Pastry, Tapestry
    - ID を、上位桁から順に、b (例 4) ビット単位で揃えていく。
    - Pastry は、最後の 1ホップは違う方法で近づく。
  - Kademlia
    - ID を、上位桁から順に、1 ビット単位で揃えていく。

内容

- 自己紹介
  - 自身, 会社
- オーバレイネットワーク
  - unstructured / structured オーバレイ
- Proof of concept のその先に
  - Overlay Weaver (structured)
  - オーバレイマルチキャスト (unstructured)
- 産と学の間で

Proof of concept のその先に

Overlay Weaver

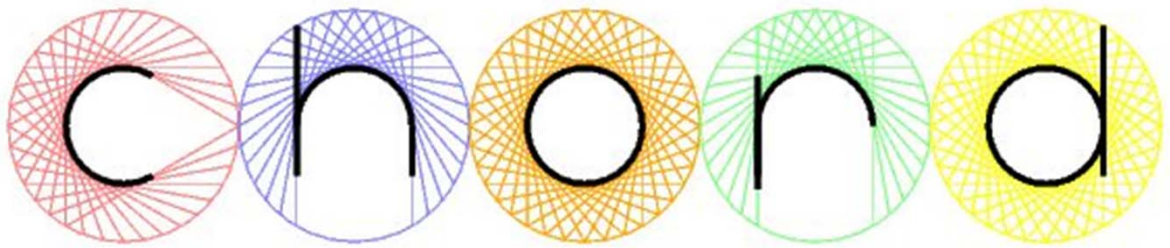
(structured overlay)

オーバレイ織り器

Overlay Weaver

- structured オーバレイ関係の名前には、糸を織るというアナロジが見られる。
  - Chord -- 弦
  - Tapestry -- 壁掛けの織物
  - Pastry -- 練り粉, パイの皮 ??
- Overlay Weaver
  -- (structured) オーバレイの織り器
  - Weave -- 織る, 編む
  - Weaver -- 織り手, 織工, 編む人

Weaver

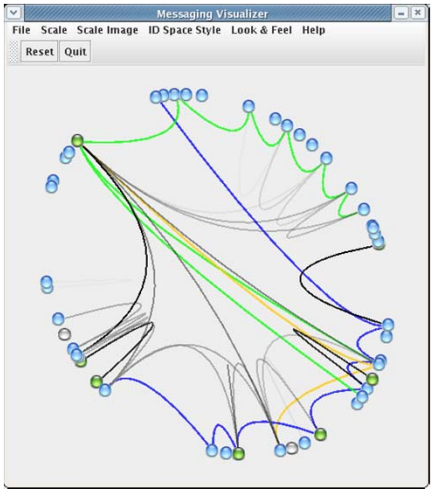


Tapestryのロゴ

Overlay Weaver

- 構造化オーバレイライブラリ
  - 記述言語は Java。
  - 約 ２万８千ステップ。
    - ４万５千行。
  - Apache License 2.0。
    - いろいろな目的に使いやすい。

可視化ツール



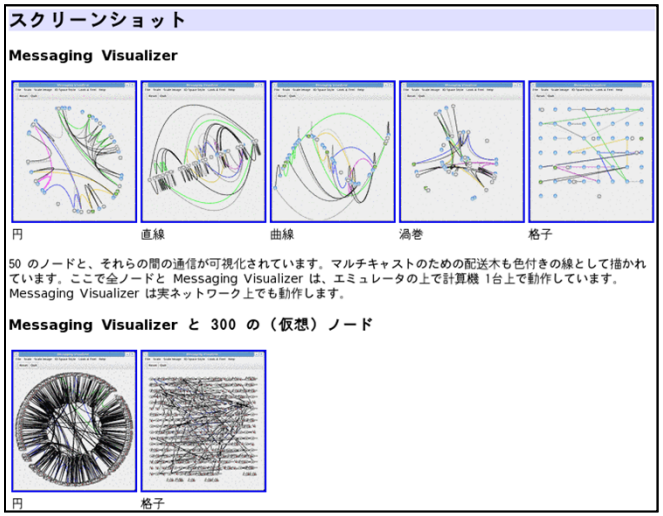
- 特徴
  - 分散ハッシュ表 (DHT) だけじゃない。アプリケーション層マルチキャストも。
  - ルーティングアルゴリズムを差し替え可能：
    Chord, Kademlia, Koorde, Pastry, Tapestry
  - コーディングなしで運用や実験が可能。
    - サンプルツール (DHTシェル等) を使った運用。
    - エミュレータを使った実験: メッセージ数やホップ数の計測。PC 1台で 40,000ノード動作。
  - 動作状態を可視化して楽しめる。デモできる。
  - XML-RPC 経由で DHT を使える。
    - Bamboo, OpenDHT と同じプロトコル → 同じクライアントを使える。
  - 地球規模テストベッド PlanetLab 上で運用している。
    - 35ヶ国 418台

オープンソースソフトウェア, Overlay Weaver 研究プラットフォームとして

- http://overlayweaver.sf.net/ (SourceForge)
  - 2006年 1月 17日、リリース。
  - Apache License 2.0

- 状況 (2008/2/28 19時)
  - ダウンロード 8,867件。
  - メーリングリスト登録数
    - 英語 70名, 日本語 90名
  - Mixi コミュニティのメンバ数 151名

ウェブサイト



- 研究プラットフォームとして
  - ブラジル, ロシア, 英国, ハンガリー, …
  - 第三者が、ルーティング方式を追加実装: Symphony, EpiChord, …
  - 第三者による利用 / 応用多数: RDFのDB(産総研), XMLデータ検索(筑波大), 複製数の影響 (U Federal do Rio Grande do Sul), 複数サーバの横断検索 (お茶大), ウェブアクセス動向集計(東工大), 複製配置手法 (早大), コミュニケーション向けユーザ管理サーバ(日立), 多次元範囲検索システム(NEC), …

Proof of concept

- 論文に書いたセールスポイント (concept)
  - 研究プラットフォームとして有用
    - 数百ステップ程度でアルゴリズムを実装可能。
    - (PC 1台上で) 多ノードのエミュレーションが可能。
    - アルゴリズム間の公正な比較が可能。
  - アルゴリズム研究を応用に直結させる
    - PC 1台上で研究・開発を進めると、そのまま実環境で動作する。

- Proof of concept としてやったこと
  - 複数のアルゴリズムを実装し、容易であることを示した。
  - アルゴリズム間の公正な比較が可能であることを示した。
  - PC 1台での大規模実験が可能であり、そこでアルゴリズム間の公正な比較が可能であることを示した。
    - 数千ノードのエミュレーションと、状況の測定。
  - 実環境で動作することを示した。
    - 実機 約 200台での動作と、動作状況の測定。

…さて、論文は書けた。そのとき 2006年 1月

Proof of concept の先

- 世の中にインパクトを与えたい。
  できれば、自分の手で直接。
- 自慢したい。

- ソフトウェアの公開・配布
- 実用性の向上
- 宣伝, ML 運営など、利用の拡大策
  - これについてはまたの機会に。

Proof of concept の先に

- UDP hole punching と UPnP NAT Traversal
- ルーティングの TTL
- black list
- 公開・配布
- Koorde でのホップ数削減
- ロックに起因する激しい成功率低下
- PlanetLab 上での運用
- タイムアウト時間の動的な調整
- 壊れた UDP データグラムの受信
- JPCERT/CC からの通告
- (Chord での) predecessor の生存確認
- 不正確な時計との格闘

PoC の先に:
UDP hole punching と UPnP NAT Traversal

- 今や、インターネットにつながったマシンのほとんどが NAT 経由
  - ご家庭はほぼ 100% NAT 経由では?
  - インターネット側から内側のマシンに対する送信 / 接続確立ができない。
  - 既存の構造化オーバレイアルゴリズムは、各ノードが双方向に通信可能であることを前提としている。NAT は困る。

- NAT 越え手法 2種を実装した。
  - UDP hole punching
    - 独自プロトコル (≠ STUN)。full cone NAT を越えられる。
    - (port) restricted cone NAT は非対応。symmetric NAT は UDP hole … では越えられない。
  - UPnP NAT Traversal
    - 今野氏の CyberLink for Javaというライブラリを使用。
      ライブラリの使い方は Google Code Search で検索。サクッと。
  - どちらも、実装 (&簡単な試験) は 1,2日程度の労力。

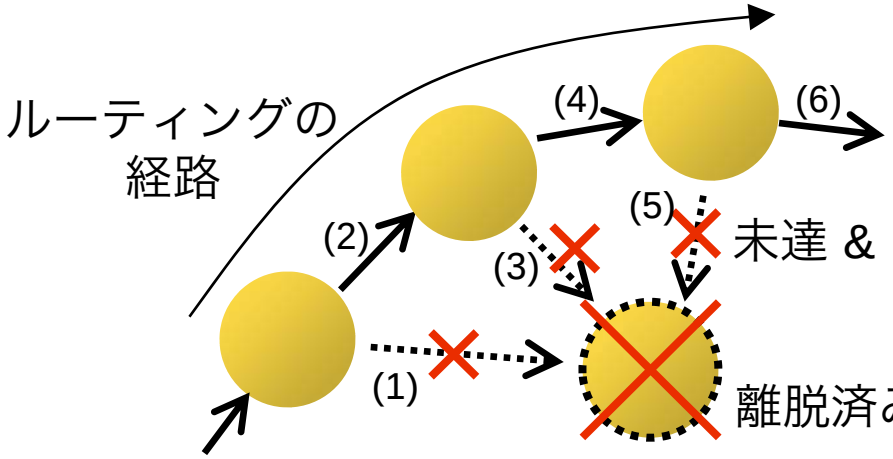
PoC の先に:
ルーティングの TTL

- アルゴリズムの実装にバグがあると、ルーティングがループして無限に…
- 1台上のエミュレーションではともかく、実環境では許されない。

- 最大ホップ数を定め、そこでルーティングを打ち切るようにした。
  - recursiveルーティングでは、ルーティング要求メッセージ中に TTL を含める必要あり。
- 実環境には必須の機能だが、論文中では見たことがない。

PoC の先に:
black list

- ルーティング中、すでに離脱したノードに何度も通信を試みてしまうことがある。
  - recursiveルーティングで起こり、iterativeルーティングでは起きない。
  - TCP であっても、内部で、遅延に応じたタイムアウト時間を動的に決めている。再送を繰り返し、完全に諦めるまで秒単位の時間がかかることもざら。



タイムアウト 5秒だとすると
5秒 × 3 = 合計 15秒 待ち！

- black list を導入
  - 通信失敗の相手ノードを black list に載せ、ルーティング要求メッセージに含める。各ノードは、black list に載っているノードには通信を試みない。
  - 注: black list 上のノードを経路表から落とすか否かは、難しい問題。

PoC の先に:
公開・配布


産総研がP2Pの開発効率を高めるソフト開発キット「Overlay Weaver」を公開

2006/01/18 20:36

Keyword: プロトコル　Java / Linux / オープンソース / コラボレーション / コンピュータ / パソコン / ネットワーク / ソフトウエア / 開発ツール / データベース　印刷用ページ

産業技術総合研究所（産総研）グリッド研究センターの首藤一幸氏らは，P2P（Peer-to-Peer）のアルゴリズム研究やアプリケーション開発のためのソフトウエア開発ツールキット「Overlay Weaver」を開発し，オープンソース・ソフトウエアとして公開した（公開ページ）。P2P技術の実装に必要な開発支援機能を提供し，P2Pアルゴリズムや P2Pアプリケーション（例えばファイル配信やコラボレーション・ツール）を効率よく開発，評価することが可能となる。

「Overlay Weaver」が備える可視化ツール。ルーティング・アルゴリズムの動作を表示している。

- 2006年 1月、ウェブで公開・配布開始。
  - 第一の目的は、自慢・インパクト。加えて…
  - 3月末の退職後も自由に開発・利用を続けられるように
    - 制約の緩いライセンスで
    - 所属組織の外 (SourceForge.net) にて
    - 在職中に公開しておいた。
  - 以来、0.1 ～ 0.7.7 で 45回のリリース。
  - 日本語を優先する理由がなかったので、ウェブ上の文書は英語で用意。それを和訳。
  - マニュアルだけではどう使ったらいいかわからない？ → チュートリアルも用意。
  - 面白さを感じてもらうために、スクリーンショットに加えて、クリックだけで起動するデモを用意。
  - 関連する ML と知人にアナウンスのメールを送りまくり。
    - 日経BP Tech-On! (ウェブ媒体) に記事を書いてくれた。

PoC の先に:
Koordeでのホップ数削減

- 2006年 4月、ルーティングアルゴリズム Koorde を実装。
  - de Bruijn graph に基づく。
  - 隣接ノード (neighbor) の数 (degree) が少ない = 経路表が小さい。
    → 生存確認などの保守コストが低い。
  - その代わり、ルーティング時のホップ数が大きい。
    - 4000ノード時、中央値が Chord: 約 10、Koorde: 約 20ホップ。
    - 論文は、オーダー (O(logノード数)) しか論じてない…

- 実装当初、ホップ数が非常に大きかった。
  - 1000ノード参加時、最大で **300ホップ**！

- ある変数 (imaginary ID) の初期値の決め方をで、ホップ数を大幅に減らせた。
  - 1000ノードで最大 38ホップ。
  - 論文はほとんど何も言ってない。この程度:
    “we can reduce the number of hops to O(log n) with high probability by carefully selecting an appropriate imaginary starting node.”

PoC の先に:

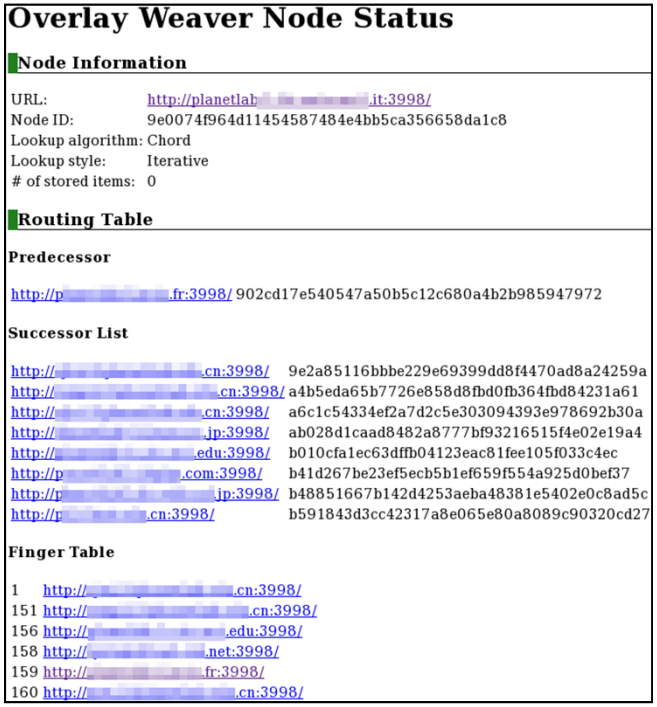
ロックに起因する激しい成功率低下

- ルーティングアルゴリズムKoorde では、DHTでのget成功率が大きく低下していることに気づいた。リリース前試験で発覚。
- プロファイリングの結果には表れない。
- 解決の手順:
  - 正常動作する最後の版を見つける。
  - 異常な版とソースコードの差分 (diff) をとる。
  - 差分を**目視**し、原因を推測。
  - 推測に基づいて修正＆試験。
- 原因は、ロックの粒度 (ロック保持の期間) を大きくしてしまっていたことだった。
- 他にどんな問題が残っているか知れない…

```
synchronized {
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
```

PoC の先に:
PlanetLab 上での運用

- 実機上、できればインターネット上で、実際に大規模な試験・運用をしたい。
- PlanetLab: 地球規模テストベッド
  - PC 2台を供出すれば、使える。
    - プロジェクトごとにVM (Linux-VServer) を用意できる。
  - ただし、企業の fee は $25,000 / 年〜！
  - 大学で客員研究員にして頂き、大学を PlanetLab に参加させた。
  - 最大で、35ヶ国 418台で動作させた。



Overlay Weaver Node Status

Node Information

URL: http://planetlab[redacted].it:3998/
Node ID: 9e0074f964d11454587484e4bb5ca356658da1c8
Lookup algorithm: Chord
Lookup style: Iterative
# of stored items: 0

Routing Table

Predecessor

http://p[redacted].fr:3998/ 902cd17e540547a50b5c12c680a4b2b985947972

Successor List

http://[redacted].cn:3998/	9e2a85116bbbe229e69399dd8f4470ad8a24259a
http://[redacted].cn:3998/	a4b5eda65b7726e858d8fbd0fb364fbd84231a61
http://[redacted].cn:3998/	a6c1c54334ef2a7d2c5e303094393e978692b30a
http://[redacted].jp:3998/	ab028d1caad8482a8777bf93216515f4e02e19a4
http://[redacted].edu:3998/	b010cfa1ec63dffb04123eac81fee105f033c4ec
http://p[redacted].com:3998/	b41d267be23ef5ecb5b1ef659f554a925d0bef37
http://p[redacted].jp:3998/	b48851667b142d4253aeba48381e5402e0c8ad5c
http://p[redacted].cn:3998/	b591843d3cc42317a8e065e80a8089c90320cd27

Finger Table

1	http://[redacted].cn:3998/
151	http://[redacted].cn:3998/
156	http://[redacted].edu:3998/
158	http://[redacted].net:3998/
159	http://[redacted].fr:3998/
160	http://[redacted].cn:3998/

PlanetLab上で運用しているノードのウェブインタフェース

PLANETLAB
An open platform for developing, deploying, and accessing planetary-scale services

PoC の先に:
タイムアウト時間の動的な調整

- 送信が失敗したと判断するまでの通信タイムアウト時間が、ルーティング全体の所要時間に大きく影響する。
  - cf. 「black list」のスライド
  - 長いと所要時間が長くなり、短いと失敗という誤判定が増える。

- 適切であろうタイムアウト時間を、通信遅延に応じて適応的に決める。
  - TCPの再送タイムアウト (RTO) 計算方式を実装。
    - Van Jacobson, Michael J. Karels, “Congestion Avoidance and Control”, Proc. SIGCOMM’88. の Appendix A。
  - DHT 実装 Bamboo の論文 “Handling Churn in a DHT” のマネ。

PoC の先に:
壊れた UDP データグラムの受信

- 2007年 3月、PlanetLab 上でのこと。
  受信したメッセージの中身が異常。
  - プロトコル上、null でないはずの個所が null。

- 受信した UDP データグラムが**なぜか**壊れており、解釈に失敗していた。バグにより、解釈失敗を無視し、解釈できたものとしていた。
  - → バグ修正

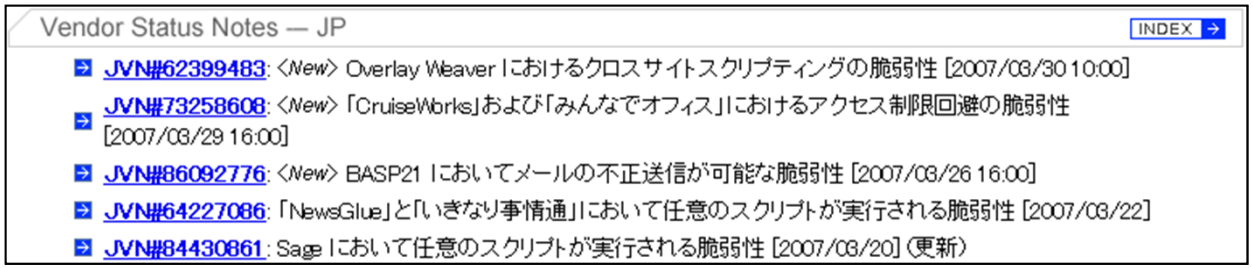
- 壊れた UDP データグラムを受信する機会は稀。
  - IP層で分割され、一部が失われた場合は、データグラム全体が捨てられるし…
  - それまで、PCクラスタ (LAN) 上では遭遇しなかった。
- PlanetLab ならでは (?) の現象。
  - 多様なネットワーク環境

PoC の先に:
JPCERT/CC からの通告

- Date: Thu, 22 Mar 2007 18:08:11 +0900 (JST)
  はじめまして JPCERT/CC 脆弱性関連情報ハンドリングチーム〇〇と申します。
- ノードの状態を表示するウェブサーバ機能に、cross-site scripting 脆弱性があった。
- その後の手順
  - 脆弱性情報を受け取るために、PGP 公開鍵を交換
    - PGPを利用できない場合、脆弱性情報は CD-R で送られてくるとのこと。
  - 開発者情報を連絡・登録
    - 所属, 役職, 氏名, メールアドレス, 電話番号, 住所
  - JPCERT/CC から電話。こちらの PGP 鍵について fingerprint を確認。
  - 脆弱性情報が、暗号化されたメールで送られてきた。
  - 脆弱性を検証。対応法と日程を決定。公表する事柄と方法を決定。以上を JPCERT/CC に連絡。
  - JPCERT/CC が、JPCERT/CC による公表日を決定。
  - 公表する文面を作成。

PoC の先に:
JPCERT/CC からの通告

- やっぱり、対応は大変。
  - まず、その脆弱性をきちんと理解しないと、適切な修正ができない。
    → CSS脆弱性を学習した。
  - まっとうなアナウンスを出すためには、脆弱性の影響 (impact) をきちんと見積もりたい。
    影響見積もりは、ただ修正するよりもかなり難しい
    - 攻撃する立場になって exploit し尽くさねばならない。
    - 頑張って exploit した → 条件が重なると cookie 漏洩の恐れがあることを確認できてしまった。
      - IPA「情報漏えいの可能性: なし」だったが…
- 勝手には、新版を出せなくなる。
  - JPCERT/CC「開発者単独でのパッチや脆弱性情報の公開はしないようお願い致します」
  - 脆弱性とは無関係のバグ修正 → 新版リリースも制限を受ける。
    なぜなら、脆弱性修正版のリリースはパッチ公開に等しいし、脆弱性を残したままの新版リリースは、それはそれでマズいだろうから。


Vendor Status Notes — JP

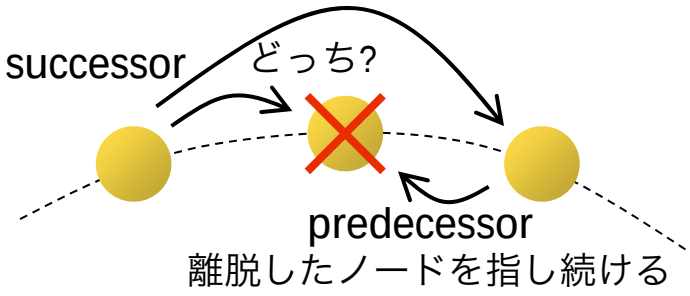
INDEX

- JVN#62399483: 〈New〉 Overlay Weaver におけるクロスサイトスクリプティングの脆弱性 [2007/03/30 10:00]
- JVN#73258608: 〈New〉「CruiseWorks」および「みんなでオフィス」におけるアクセス制限回避の脆弱性 [2007/03/29 16:00]
- JVN#86092776: 〈New〉 BASP21 においてメールの不正送信が可能な脆弱性 [2007/03/26 16:00]
- JVN#64227086: 「NewsGlue」と「いきなり事情通」において任意のスクリプトが実行される脆弱性 [2007/03/22]
- JVN#84430861: Sage において任意のスクリプトが実行される脆弱性 [2007/03/20] 〈更新〉

3月 29日
脆弱性に対処した版をリリース
3月 30日
JPCERT/CC, IPA がアナウンス

PoC の先に:
(Chord での) predecessor の生存確認

- アルゴリズム Chord の欠陥の１つ:
  predecessor の生存確認をしない。
  - 困った現象
    1. (stabilize 処理にて) successor から、すでに離脱したノードを紹介される。そのノードを新たな successor だと思う。
    2. いざ通信してみると、そのノードとは通信できない。
       successor を元のノードに戻す。
    3. 1. に戻る。
  - ルーティング自体の完遂に支障はないが…
    - タイムアウト待ち (〜数秒) が頻発する。
    - すでに離脱したノードの情報が他のノードの経路表にも伝染する。
      → 他のノードでもタイムアウト待ちが起きる。



- PlanetLab での運用で発覚。
  - 「な〜んか時間がかかるなあ」
  - 「離脱したノードがいつまでも経路表に…」
  - 実環境での運用の成果。

PoC の先に:
不正確な時計との格闘

- PC 1台でのエミュレーションで、
  数秒～数十秒分のイベントが一気に発生。
  → PC が過負荷に陥り、実験が破綻。

- 原因究明は難航
  - NTP での時刻合わせで時刻が飛ぶ？ NTP daemon を止めても起こるので、違う。
  - GC でポーズが起きてる？ incremental GC を使っても起こるので、違う?

```
class ow.tool.dhtshell.Main
arg –a Chord –r Iterative

# 3ノードを起動
schedule 0 invoke
schedule 100 invoke emu0
schedule 200 invoke emu0

# DHT に対する put / get
schedule 500 control 0 put aKey aValue 3600
schedule 1000 control 2 get aKey
```

エミュレーション
シナリオ の例

PoC の先に:
不正確な時計との格闘

- OS (Linux 2.6) 自体の時計が一気に進む現象が起きていた。
  特に CPU 負荷が高いとき。
  - 何をやっても解決しない。
    - 1分おきに NTP で時刻合わせ。
    - カーネル内の時刻合わせ手段をいろいろ変更してみた。
      - PIT (Programmable Interval Timer), TSC (Timer Stamp Counter), HPET (High Precision Event Timer), ACPI, jiffies
  - 本当の原因は不明。
    - ハードウェアクロックが不正確であるのは確か。電圧が不足ぎみ (AC 95V 程度) だから?

- ソフトウェアの側で対処
  - エミュレーションシナリオを駆動するソフトウェアの側で、時計の不自然なジャンプを検出・対処するようにした。
  - イベントを発生させる際に、予定された時刻と実際の時刻を比較する。時計が大きく進んでいるようなら、スケジュール済み & その後スケジュールされるイベントをその分遅らせる。

Proof of concept のその先に

オーバレイマルチキャスト

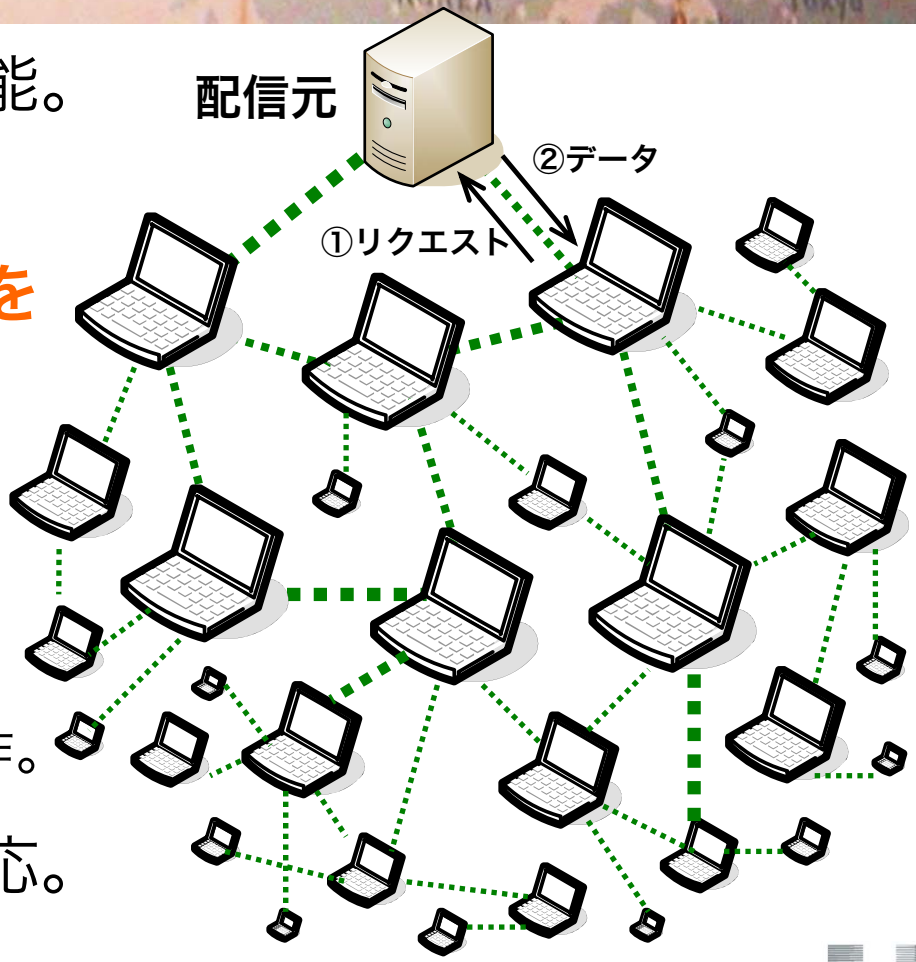
(unstructured overlay)

または
ALM: アプリケーション層マルチキャスト
ESM: エンドシステムマルチキャスト

Peer-to-peer ライブ配信

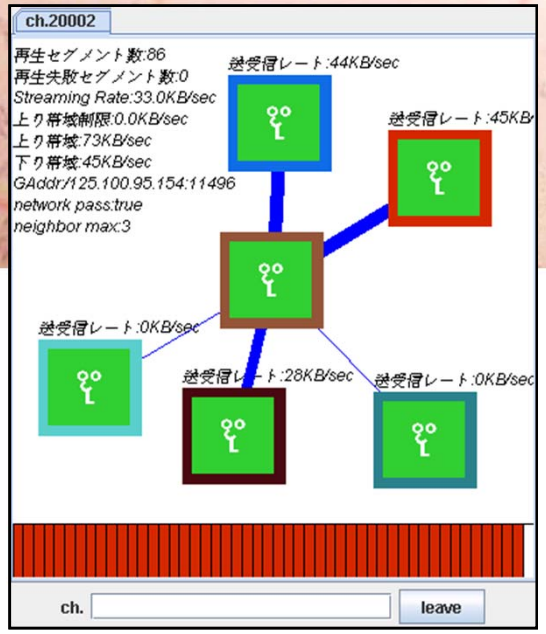
実ネットワークに適応するオーバレイマルチキャスト 放送基盤

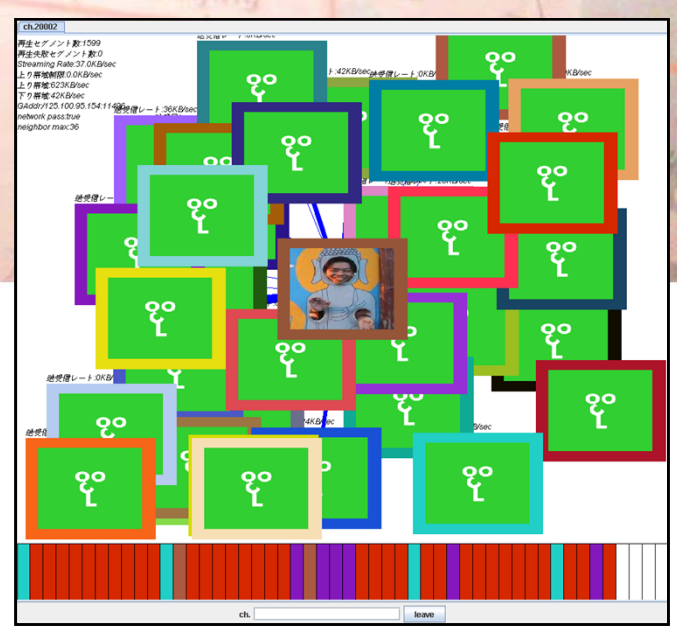
- 映像・音声の**ライブ**配信が可能。
  ここが技術的に面白い。メッシュ型 (⇔ツリー型)
  cf. YouTube, ○○動画
- 配信元の**ネットワーク帯域幅を**
  ほとんど**食わない**。
  ⇒ 極めて**低い配信コスト**
  ⇒ 誰でも発信
  cf. サーバ-クライアント, CDN
- **1万ノード**での動作実績。
  (PC クラスタ上)
  実地 & PlanetLab上でも大規模動作。
- 様々なネットワーク環境に適応。
  xDSL〜光ファイバ, ファイアウォール/NAT



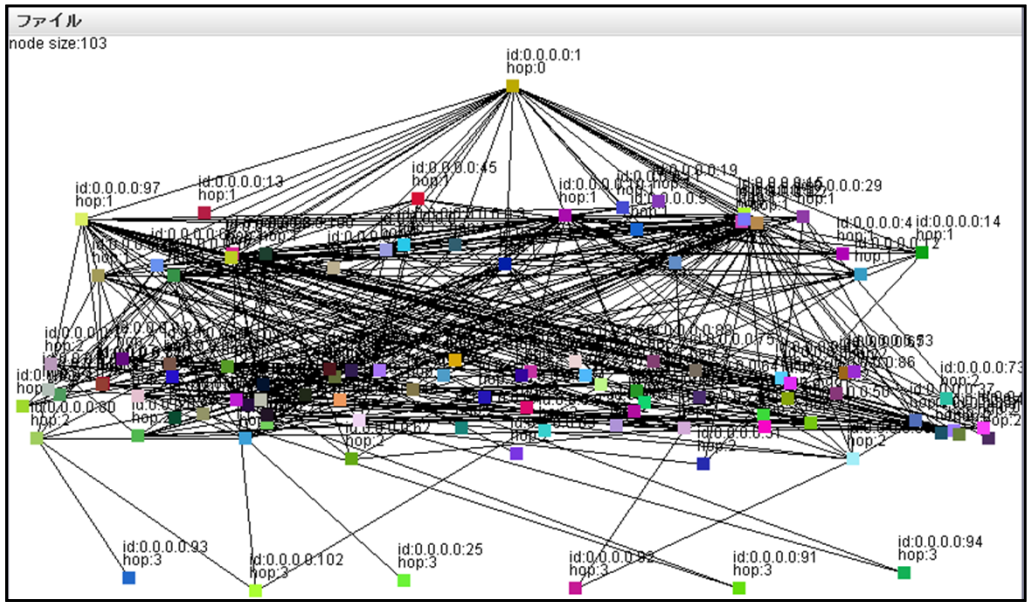
池長慶彦 (早稲田大学 村岡研究室)
首藤一幸 (ウタゴエ (株))

IPA未踏ソフトウェア 2006年度上期 並木PM





各ノードで、周辺状況を可視化



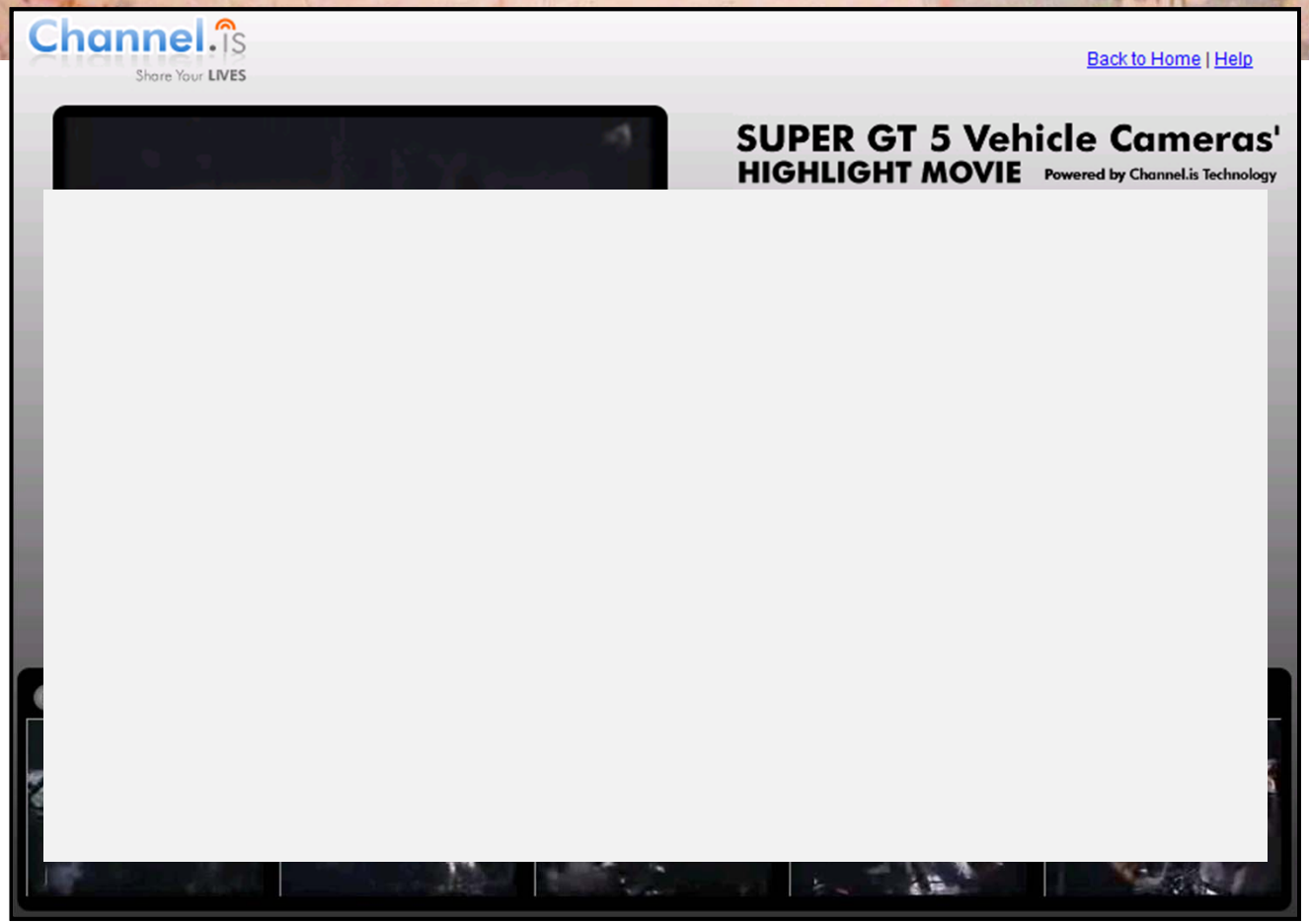
全ノードの接続状況を可視化

ウタゴエ社で商用化

プロダクト名
Ocean Grid (OG)

- 開発者から会社に、著作(財産)権を譲渡。
  - 著作者人格権は不行使、という契約。
  - 開発者が個人的にイジることは妨げたくないので、研究目的での使用や改変を、逆にウタゴエから開発者に対して許諾。
- 対価 (首藤は、なし)
  - 金銭。数ヶ月間のコンサルティング代として。
  - ストックオプション。
- 将来の物語
  - IPA, 未踏事業の御支援を頂いて、ビジネス上も価値がある素晴らしいソフトを開発。
  - 開発の対価だけでなく、S.O.も取得。
  - 会社が経済的にも成功した暁には…

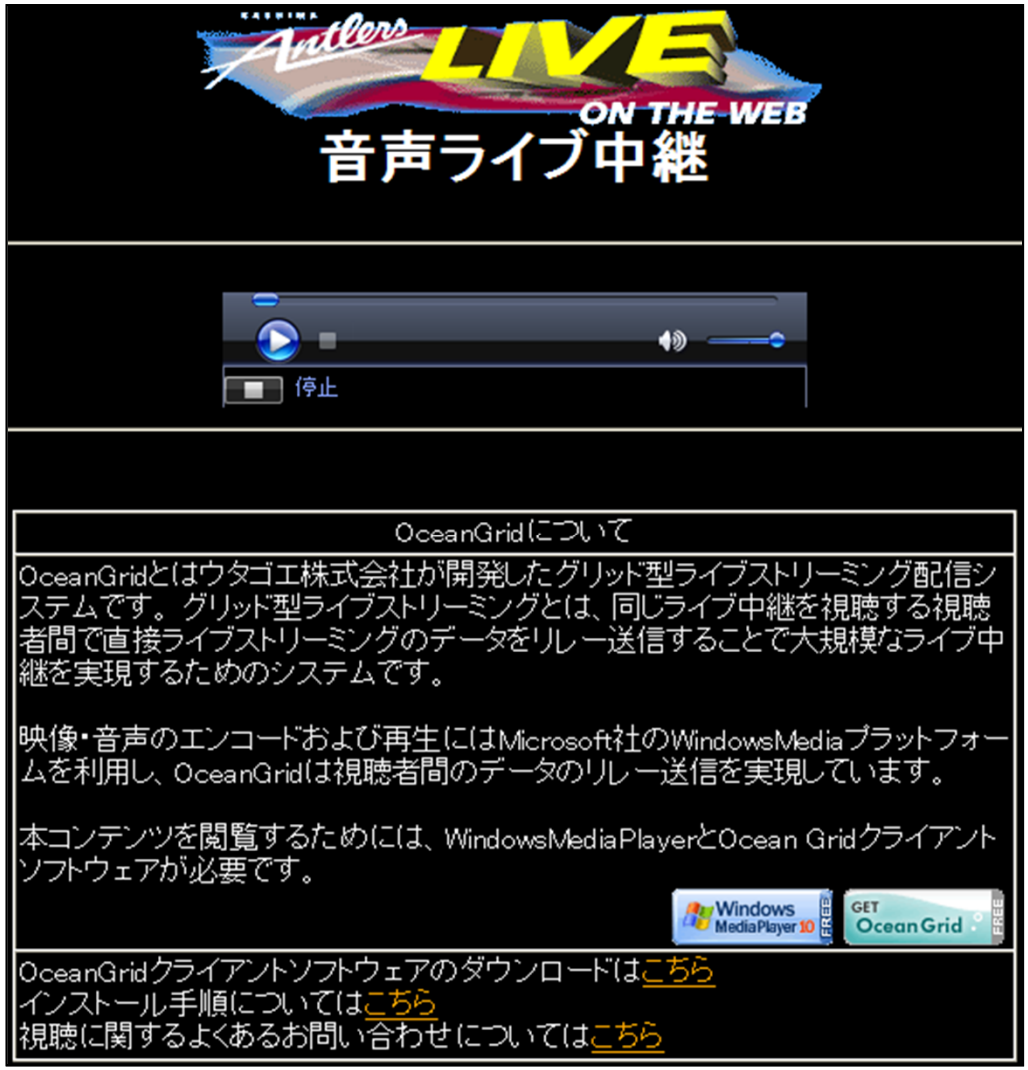
応用事例



SUPER GT

映像提供：GTアソシエイション様, SUPER GT様
協力：早稲田情報技術研究所 様

応用事例

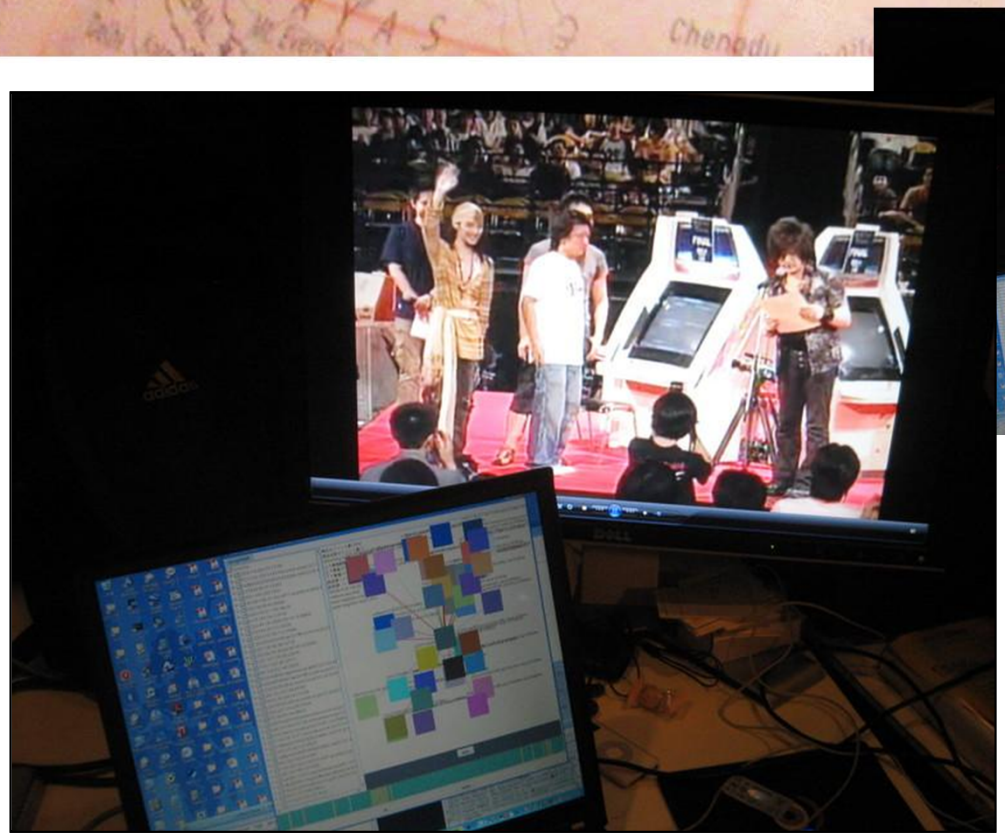


鹿島アントラーズ戦
音声 ライブ配信

協力： Ｊストリーム様

- ２回 / 月 程度
- 2007年は 4～12月
- 2008年も継続

応用事例

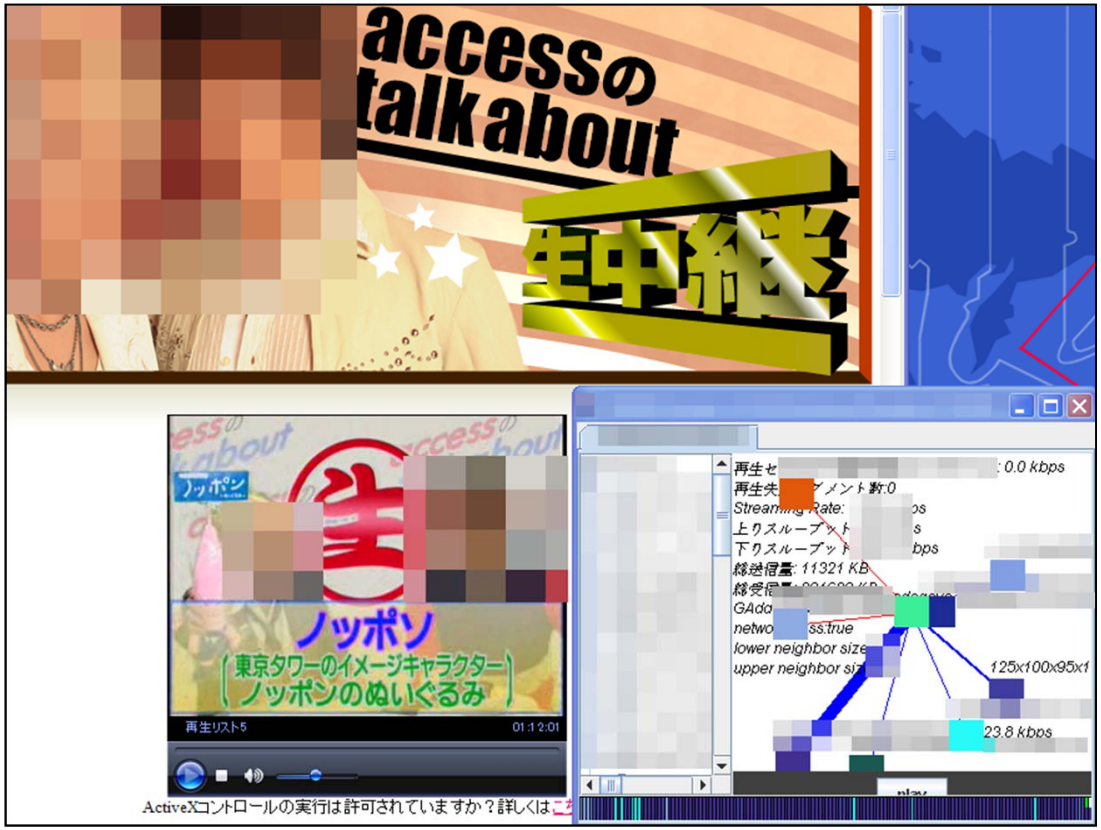


闘劇 '07 (2007/8/13)

協力：プロデュース・オン・デマンド様

1 Mbps の高ビットレート

応用事例



access の talk about: 生中継 (2007/10/29, 2007/11/26)

協力： Jストリーム様, キャステラ様

500 Kbps x けっこうな人数

応用事例

ステージLIVE映像

技術協力：基幹理工学部 後藤滋樹研究室

Standard 標準画質
300 Kbps
（IPv6配信はこちらをクリック）

High 高画質
1 Mbps
（IPv6配信はこちらをクリック）

Excellent 超高画質 P2P
P2Pによる超高画質配信
2 Mbps

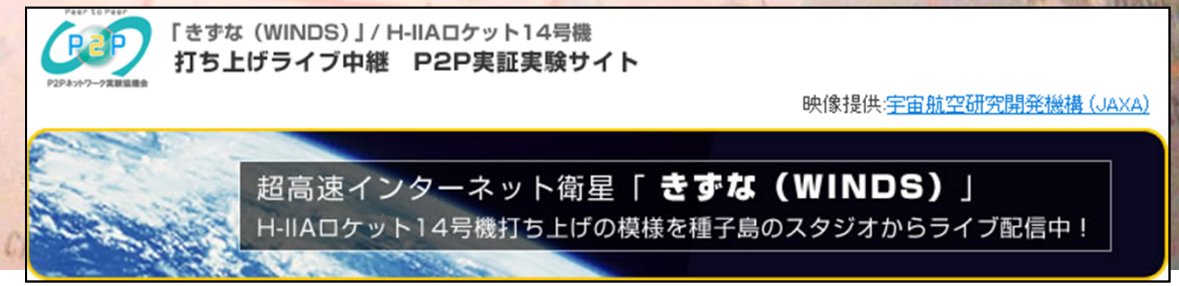
==== P2P配信について ====
専用ソフトダウンロード
ソフトのインストール方法
開発元：ウタゴエ株式会社

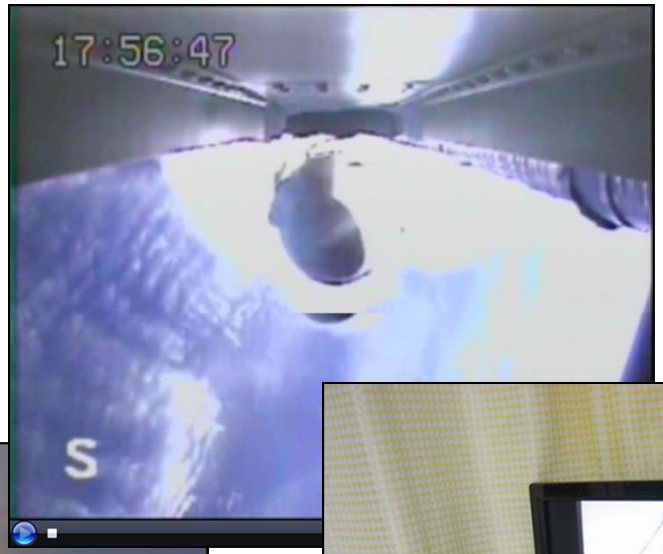
早稲田祭 2007 のステージ中継 (2007/11/3~4)

協力： 早稲田大学 後藤研究室 様

非常に高いビットレート 2 Mbps

応用事例



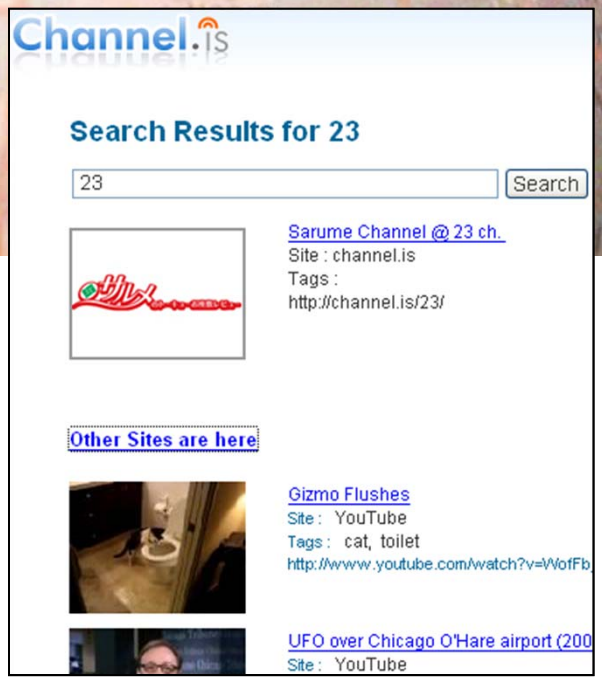


人工衛星「きずな (WINDS)
/ H-IIAロケット14号機 打ち上げ (2008/2/23)

配信： Jストリーム様, ウタゴエ

応用事例

- Channel.is
  - 誰でも大規模配信サービス
  - 英語圏向けに launch 予定



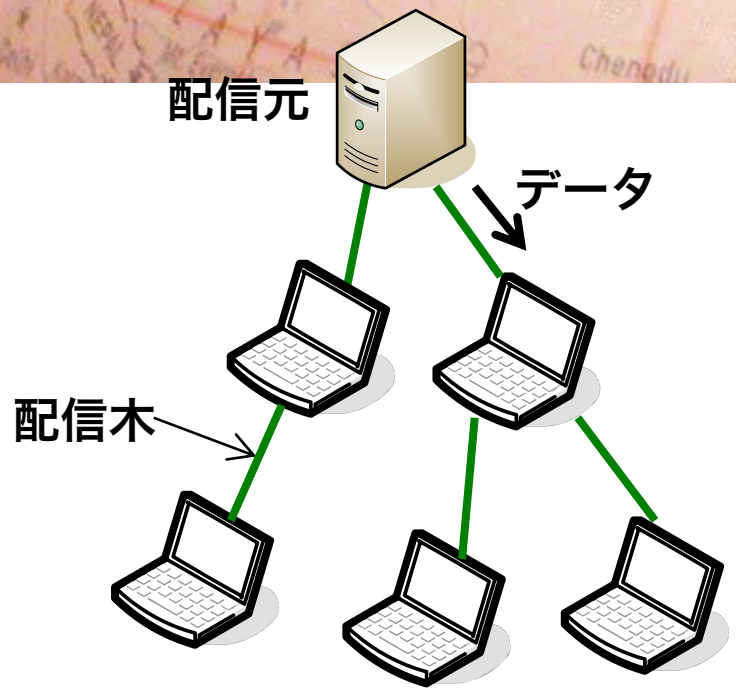
検索

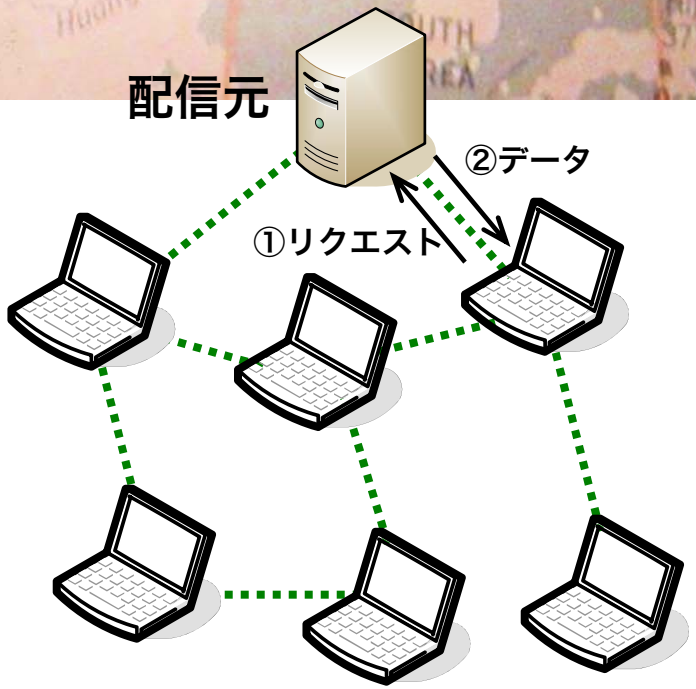
ビューア

IPA 未踏ソフトウェアで

- ワールドビジネスサテライト
  - 2007年1月15日「スーパー技術者争奪戦」　スークリがテーマ
  - サイボウズ・ラボ (奥さん), ウタゴエ, ルナスケープ (近藤さん), グーグル
  - グーグル 43秒, ウタゴエ 3分 6秒

アプリケーション層マルチキャスト (ALM) の手法:
ツリー型 vs. メッシュ型





- ツリー型
  - 明示的に作ったトポロジがデータ流路。
  - 根から葉に向かってpush。
  - △ ノード故障時は、要 迅速な復旧。
  - ○ 末端まで確実に届く。配信遅延 小。
  - 広い上り帯域幅を使い尽くし得る。
- メッシュ型
  - 隣接ノードと緩やかな関係を保つ。
  - 隣接ノードからpull。
  - ○ 生来、ノード故障に強い。
  - △ 末端まで届く保証なし。要 補償策。
  - 狭い上り帯域幅を使い尽くし得る。

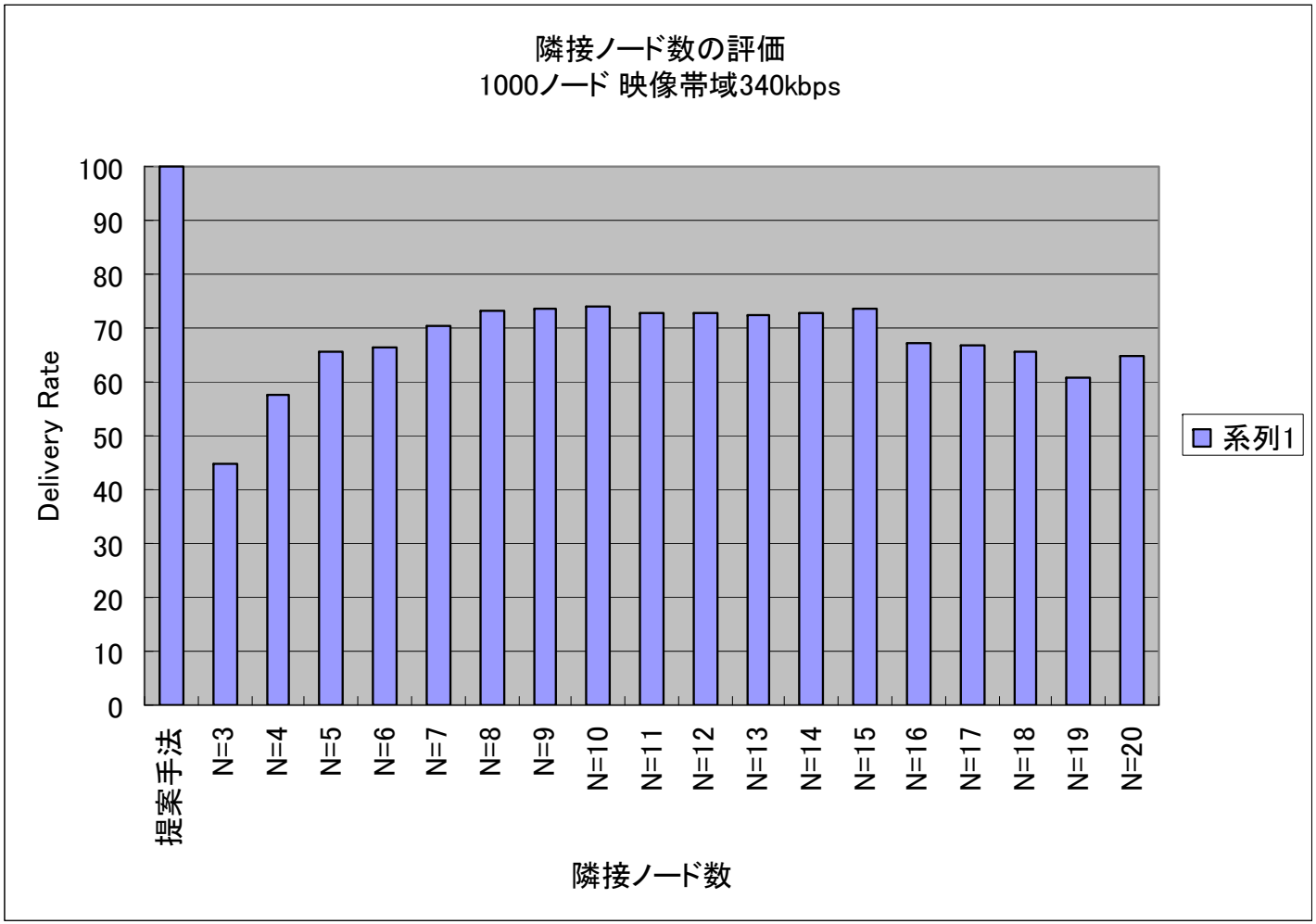
技術の詳細

- メッシュ型の最初の提案は、香港の大学 & カナダの大学から。
  - X. Zhang et al.: CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming, Proc. INFOCOM 2005.

- 未踏プロジェクト: 現実のインターネットでうまく動かすための研究, 開発
  - 隣接ノード数の調整アルゴリズム
  - 上り/下り帯域幅の差への配慮 (秘密の技術アリ)
  - 配信元ノードからデータがスムースに出て行く工夫
  - バックアップの仕掛け
  - NAT越え, ウェブとの連携, …

隣接ノード数の調整

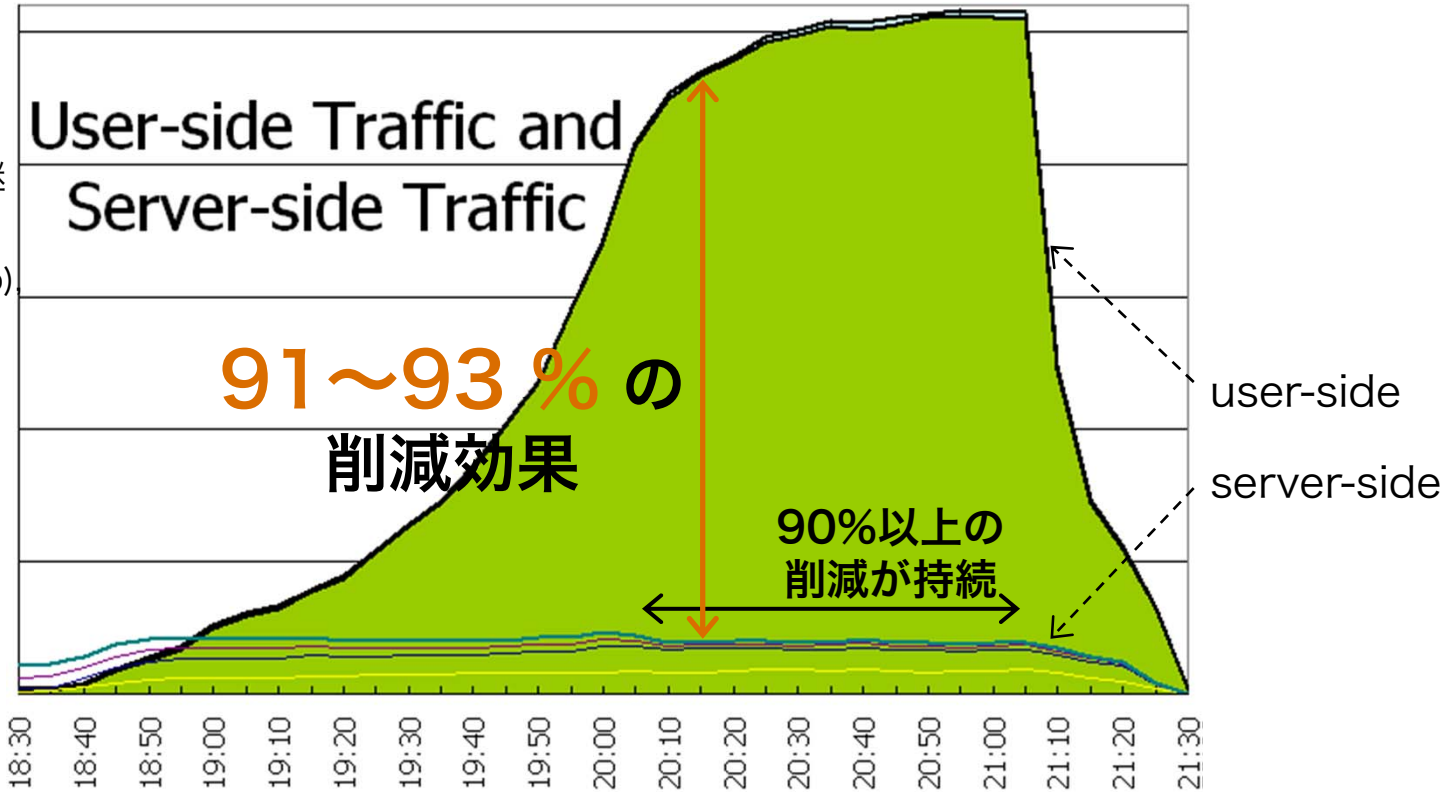
- 帯域幅等に応じて、動的に調整。



(ライブ) ストリーミングでの例

- トラフィック削減効果 90〜95 %
  - ウタゴエ社 Ocean Grid 使用

2007年 11月 26日
accessのtalk about生中継
配信: Jストリーム,
Castella (www.castella.jp)
ウタゴエ



Proof of concept の先に

Overlay Weaver　　P2P ライブ配信

- Proof of concept
  - アイディアの有用性が示された。
- 実用
  - 使いものになる。
- 普及 や 商用
  - 世の中で使われる。
  - 買ってくれるお客様がいる。

Proof of concept の先に

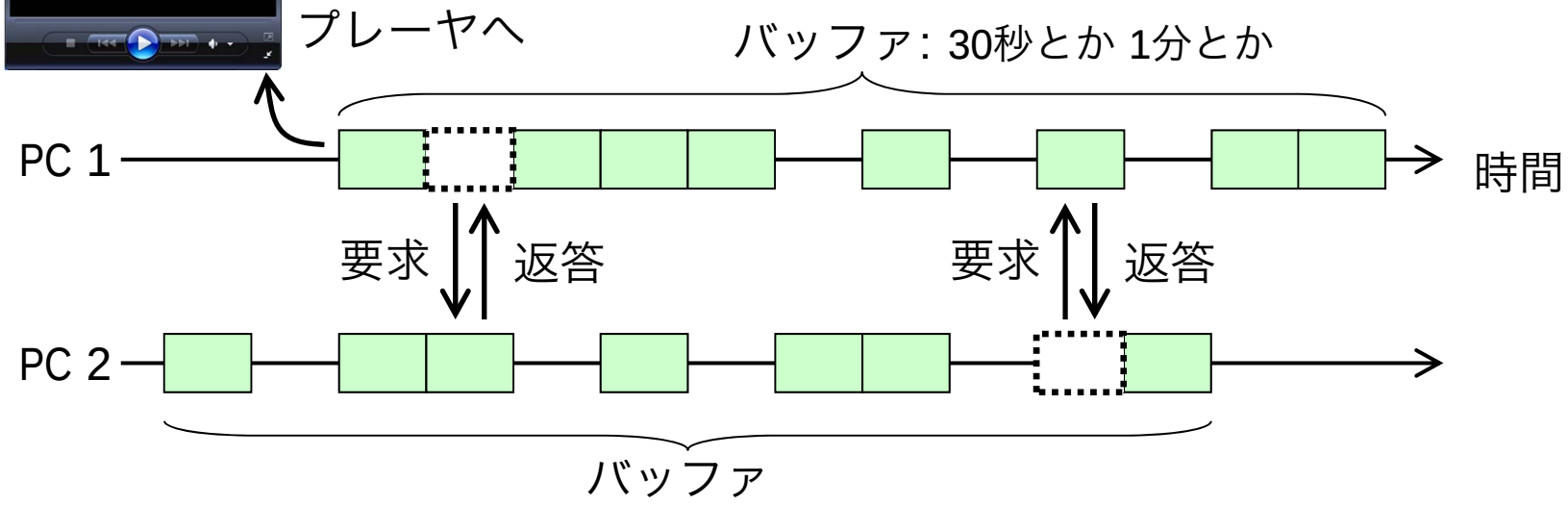
- 実地運用ではじめてわかること
  - PC 内蔵時計の不正確さの影響
  - DNS でのホスト名解決待ち
  - ネットワーク越しのデッドロックの伝染
- 実運用に必要な機能・ソフト
  - ログサーバ, バックアップ, ソフトの (自動) 更新機能, …
- 配布ファイルのスリム化
- 変なものを流されるリスク
- 帯域幅 / 性能の非均質性
- NAT, firewall
- 様々な OS / ブラウザへの対応
- 各種のストリーミングプロトコル

技術の話にフォーカス。
ビジネスの話は、またの機会に。

PoC の先に:
実地運用ではじめてわかること

- PC 内蔵時計の不正確さの影響
  - 時計の進み方がズレていると…
  - ツリー型 (や通常のストリーミング) では、
    大きな問題にはならない。
    - せいぜい、一時的な停止 / 早送り程度?

多種多様な PC が数多く
集まってはじめて発覚する。

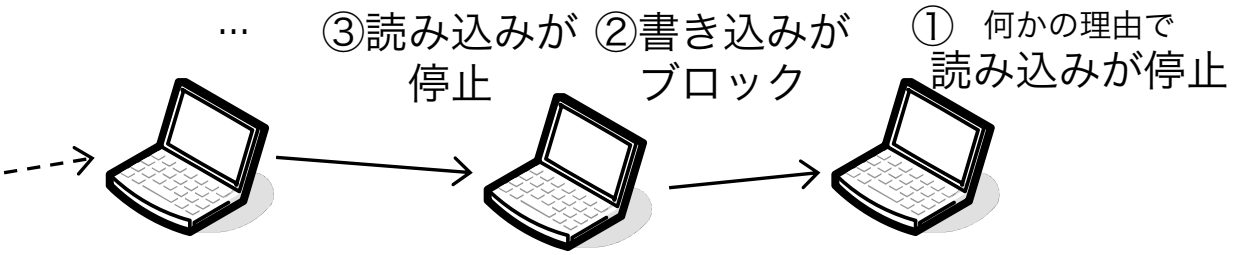


PoC の先に:
実地運用ではじめてわかること

- DNS でのホスト名解決待ち
  - 現象: ソフトが最長数秒間フリーズする。
    - GC か? それにしても長すぎる。
    - フリーズしている瞬間をとらえて、デバッガ (Eclipse) で一時停止させ、各スレッドのスタックトレースを調べた。
  - あるスレッドが、ロックを保持したまま DNS の逆引き待ちをしていた。
    - Java.net.Inet(Socket)Address#getHostName()
    - 別のいくつかのスレッドは、そのロックの解放待ち。
  - 解決
    - DNS 待ちが生じないよう、DNS に問い合わせる恐れのある処理は別スレッドで行うようにした。
    - 一度得た結果をキャッシュするようにした。
- 実地だからこそ発現した問題
  - インターネット (広域ネットワーク) → 逆引き待ちが発生
  - 多くのノードが盛んに参加 / 離脱 → 逆引きの機会が多い

PoC の先に:
実地運用ではじめてわかること

- ネットワーク越しの デッドロックの伝染



  - TCP 接続への書き込みをタイムアウトさせる標準的な方法は、ない。
    - 読み込みをタイムアウトさせる方法:
      setsockopt(…, SO_TIMEOUT, …
  - タイマで監視して、
    脇からソケットを close するようにした。

PoC の先に:
実運用に必要な機能・ソフト

- ログサーバ
  - 商用には、全体の状況把握が不可欠。特に視聴 PC 数。
  - ソフト自体は P2P だが、ログは集約する必要あり。
  - しかも、要二重化。

- バックアップ
  - 理由
    - メッシュ型では、再生時刻までにデータを取得できる保証がない。
      → メッシュ型の仕組みの中でのバックアップ
    - 10年の歴史がある、従来型のストリーミングの方が信頼性が高い。
      → ユニキャストへのフォールバック

- ソフトの (自動) 更新 and/or 更新通知機能
  - 更新版の有無を、利用者はチェックしてはくれない。
  - 要更新: セキュリティ的な問題を修正した場合, プロトコルを変更した場合, …

PoC の先に:
配布ファイルのスリム化

- ダウンロード & インストールに要する時間は短い方がいい。
  - たいてい、ダウンロード時間が律速。
- ソフトの種類に応じて、現実的に許容されるダウンロード量がある。
  - 1 MB まあいいか, 10 MB ちょっと大きい?, 100 MB 無理
  - 我々の場合、Java のランタイム (JRE) がネック。
    - Java Kernel (early 2008) に少しだけ期待。
- 圧縮ツールを手当たり次第試した。
  - 普通の ZIP, DGCA, 7zip, KGB Archiver, …
  - KGB: 圧縮に 25 分！ (2.0 GHz Core Duo)
  - 現在 10 MB くらい。

PoC の先に:
変なものを流されるリスク

- 配信を行う側にとって、何がリスクか？
  - コンテンツを保存されて、再配布 / 放映などされてしまうこと?
- 実は、意図しないコンテンツを流されてしまうこと。ではないか。
  - 犯罪的な映像、等 → 名誉が傷つく。
  - 手段
    - 通常のストリーミング: DNS情報の置き換え等
    - P2P: 中継ノードによるデータのすりかえ (ツリー型で容易)
- この不安を拭ってあげない限り、商用には使ってもらえない。
  - 心配なら DRM かけて、でだいたい大丈夫。
  - でも、DRM 対応していないストリーミングプロトコルもある。近年勢力を伸ばしている Flash とか。
  - 独自に暗号化も実装済み。

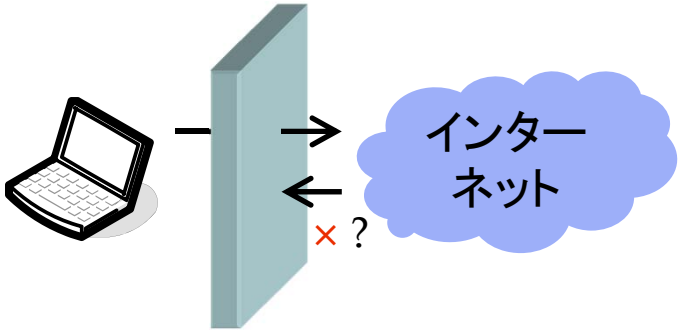
PoC の先に:
帯域幅 / 性能の非均質性

- 視聴者のネットワーク接続は様々。
  - PHS, HSDPA (EMOBILE), xDSL, 光ファイバ, …
  - 数十 kbps ～ 100 Mbps

- P2P コンテンツ配信では、各ノード (PC) の上り帯域幅をフル活用することが重要。
  - それによって、大元の負荷を減らすので。
  - 下り >> 上り帯域幅 → ネックになる上りに注目。
  - 太いところはもちろん、細いところもそれなりに。
    - ツリー型 ALM では、細い上りは活かしにくい。
      多重ツリーでは活かし得るが…
  - 上り帯域幅の活かし方
    - ツリー型: 帯域幅を事前に測定、子ノードの数を計画する。
    - メッシュ型: 隣接ノード数を調整して、間接的に制御。

PoC の先に:
NAT, firewall

- 今や、インターネットにつながったマシンのほとんどが NAT 経由
  - ご家庭はほぼ 100% NAT 経由では?
  - メッシュ型 ALM では、それほど困らない。NAT 内側から確立した接続を使って双方向に通信すればよい。
  - ツリー型 ALM では、NAT 外から接続できないマシンは、子を持てない → 上り帯域幅を活かせない。残念。ただし逆方向に接続を確立するという手も。



NAT, firewall

前提

- HTTP 等の proxy だけ利用可
- 内からの HTTP 接続が可
- 内から 80番/tcp への接続が可
- 内からの TCP 接続が可
- 内からの UDP 送信が可
- UDP hole punching 可能
  - full cone, (port) restricted NAT
- UPnP でのポート番号対応付けが可

- 我々の場合
  - UPnP NAT Traversal を試みる。
    - NAT 内外のポート番号対応付け。
  - 注: TCP ベースなので、UDP hole punching は意味がない。

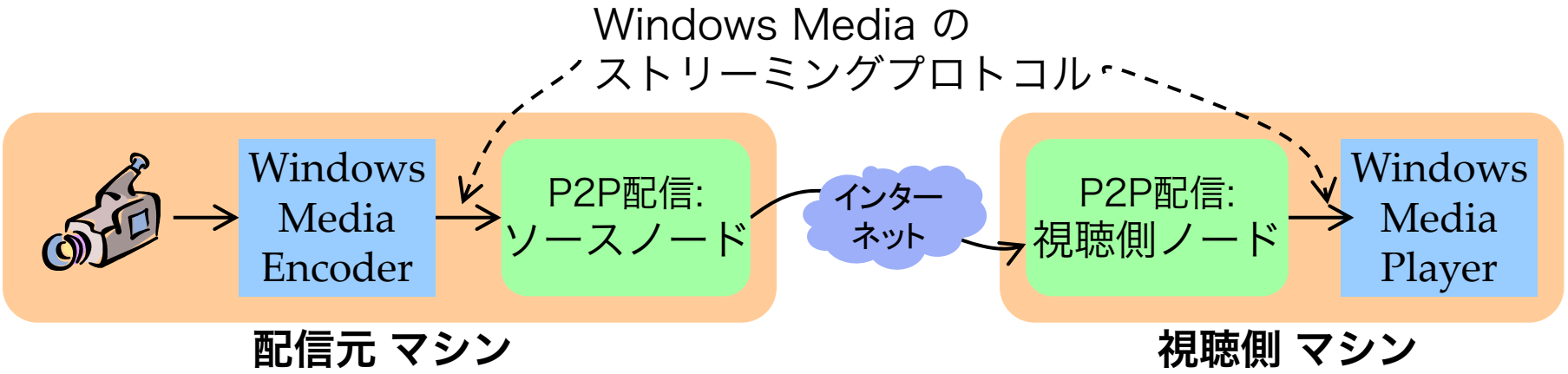
PoC の先に:
様々な OS / ブラウザへの対応

- OS
  - Windows XP, 2000, Vista, Mac OS X, Linux, …
- ブラウザ
  - Internet Explorer, Firefox, Opera, …

- 困難の例
  - Windows Vista 対応がけっこう大変。
    - ソフトの更新が困難。
      - 更新用ミニソフトが、本体のファイルを更新できず。
        などなど
  - Firefox (というかレンダラ Gecko) では、JavaScript のコードから Windows Media Player の内部状態を取得できない。
    - 要 追加のソフトウェア。あまり知られていない?

PoC の先に:
各種のストリーミングプロトコル

- 各種のストリーミングプロトコルを解釈・出力する必要がある。
  - しかも、プロトコルは非公開であることが多い。
    - Windows Media, Flash, …
  - P2P 配信の本質的な部分ではないが、必要。



Proof of concept の先の先の先へ