(Tourist 続き)

level l のノードについて
  nodeId の l-bit prefix
    固有prefix
    (prefix eigenstring)
  nodeId の l-bit suffix
    固有suffix
    (suffix eigenstring)

message routing (キーは k)
1) 自ノードが root node であるか？
   (is the local node the root node?)

  自分と k が同じ prefix なら (自分の固有prefix)
  (if the localnode has common prefix with k,)
    そういうノードを全員知っているから (prefix table)
    (because the local node knows ALL nodes with the prefix,)
    自分が root か判断できる
    (the local node can determine whether it is the root node)
      自分だったら終了
      (if so, done.)
      自分でなければ(誰であるかはわかっている) 2) へ
      (else proceed to step 2)
  自分と k の prefix が違うなら
  (otherwise, proceed to step 3)
    root かどうかはわからない 3) へ

2) root node が prefix routing table に入っているか？
  (is the root node contained in the prefix table?)
  1) で自分とkの prefix が一致して
  (it is the case the local node has common prefix with k)
    なおかつ自分は root でなかった場合
  そうならそのノードに転送
  (forward the message to the root node)
  one-hop

3) suffix routing table から転送先を探して転送
  (select a node from the suffix table,)
  条件: そのノードの prefix routing table に
    root node が入っている
  (whose prefix routing table must contain the root node)
  two-hop

  Tourist の key advantage
  X が Y へのポインタを持っていれば
  (if X holds a ponter to Y,)
    Y の prefix table に root があるかを
    X は自分で判断できる
    (X can judge whether Y’s prefix table contains the root)

  key: k とする
  Y のレベルを ly として
  k の ly-bit prefix が Y と一致すれば
    Y の prefix table に root がある
    (上の 1, 2に該当)

  XOR-distance を採用したことによる利点
  (引き算の距離だと, prefix が一致しなくても近いことがある
   (numerical difference would lack completeness)
    k:00111 と node:01000 とか)

  ？そのような Y が必ず X の suffix table にあるのか？
  その保証はない -> backup structure へ

1,000,000 ノードでは
  two-hop のためには, それぞれ 2170エントリの table が必要
    sqrt(4.7 N) for 99.9%

表の管理には multicast を使う

multicast in Tourist
  more sophisticated
  join/leave の multicast
    routing table にそのノードを含むものだけに送るべき
  section III で

level の自動調整 (auto adjustment)
  PC の性能に応じて level が違う
  (depend on the capability of nodes)
  環境変化で自分のレベルを調整

  ノードは帯域の上限 W(bps) を設定
  (upper bandwidth threshold)
  必要帯域が W/2 - W になるように level を自動調整
  (adjust the level to keep actual bandwidth lying between W/2 and W)

  システムの規模の拡大 -> table も拡大
  maintenance cost Wx も増加
  level <- level + 1 (lower the level)
  table は半分になる

  table entry の数はほぼ一定になる
  必要帯域もほぼ一定

  routing efficiency に影響
  one-hop で届く確率は (Px + Sx)/N
    Px,Sx - prefix,suffix table の大きさ

churn rate にも self-adaptive
  lifetime 減少
  multicast の頻度上昇
  帯域の上限に達すると level = level + 1 (lower the level)

  tablesize は average lifetime にも比例

帯域以外の条件でもレベル調整をする
  memory, CPU, storage?
(a node can adjust its level for any reason)

B. XOR-based Distance Metric
there would be a "routing puzzle"
leaf set と prefix table の違い
  leaf set - symmetric
    ？+方向も -方向も含むということか？
Figure 2
  X: level 4 (4-bit prefix table)
  prefix table:
    key A000 が来たとき, X(A013)は自分がrootであることがわかる
  numerical distance:
    9Fxx の存在がわからないから判定できない

同様に, 他の Y が root へのポインタを持っているかが X にわかる
  Y の固有prefix と k がわかれば

C. Backup Routing Table
  core structure may fail in
    extremely large
    dramatically dynamic
  situations
  においては, routing に失敗する可能性
  そのための backup routing table

level-l node
  backup routing table - l items
  i 番目に入るノードは (the i-th node)
    先頭 (i-1) bits が自ノードと一致

(first (i-1) bits are the same with the local node)
i-th bit目は異なる

Figure 3
level-4 のノードの backup table
たいていは suffix rouing table のエントリへの(仮想)ポインタ
(in most cases, the pointers can be found in the suffix table)
suffix table に該当ノードがいない場合のみ本当のポインタ
実験によると very large and dynamic な場合だけ出現

Pastry の routing table との類似
O(logN) routing

Figure 4
logically complete な routing algorithm
two-hop の候補が見付からなかったときに
backup table を利用する(line(4))

greedy
XOR distance は単調減少

line(3) で複数候補がある場合
実ネットワーク的に近いものを選ぶと有利
これについては section IV

```
rcv_msg(key=k, message=msg) //receive a message.
     if begin_with(k, prefix_eigen) then
                  //if the prefix eigenstring begins with k
     begin
(1)    p := getNearest(prefix_rt, k)
                  //get the nearest pointer to k in the prefix routing
                  //table, using XOR-based distance metric.
       if (p = self) then report(msg, upper_application)
                  //self is the root node, reporting upward.
       else forward(p, k, msg) //send the message to p.
     end else //the root node does not in the prefix routing table.
     begin
(2)    cand := get_reachable(suffix_rt, k)
                  //cand is the set of nodes that can route
                  //the message to the root node via one hop.

       if (cand != null) //two-hop routing from now on.
       begin
(3)       p := get_best(cand, k)
          forward(p, k, msg)
                  //forward the message to the best one in cand.
                  //the choice can be based on various preferences.
                  //basically, a random one is chosen.
       end else //no suitable pointers in the suffix routing table.
       begin
 (4)       p := get_nearest(backup_rt, k)
           forward(p, k, msg)
                  //forward the message to the XOR-based
                  //nearest pointer in the backup routing table.
       end
    end
```

III Routing Table Maintenance
multicast が main issue
prefix と suffix は対称 (symmetric)
ここでは prefix についてだけ記述

A. Tree-based Multicast

X’s (prefix) audience set
prefix table に X を含むノードの集合
(all the nodes whose prefix table contain a pointer to X)
ノードの join/leave/*adjust level* は audience set に伝える必要

audience set の構成法
Figure 5
  audience set の要素:
  それの固有 prefix が X の nodeId と一致
  (whose prefix eigenstring is a prefix of X)
  audience set の要素 Y の prefix table には
    同レベルとそれ以下(値が大きい)のレベルの
    全 audience ノードを含む
    (contains the pointers to all the others at the same or lower level)

  audience set のうちの最高レベルのノードに伝われば十分
  (highest level node in X's audience set)

  Figure 5 では最高レベルは 0 だが
    いつもそうとは限らない

定義
1) X が Y の (prefix) super node とは
  X のレベルが Y より高い
  X の固有prefix が Y の prefix

  X の prefix routing table は
    Y の prefix table を含む

2) T が X の (prefix) top node とは
  T が X の super node のうち, 最高レベル

レベル 0 ノードがあれば, それがつねに top node
(if level-0 nodes exist, they are top nodes)
そうでないとき
  システム全体はいくつかの part に分かれる
    part が違うと prefix table に共通部分無し
  各 part はそれぞれ top nodes をもつ

各ノードは (prefix) top-node list を保持
  t 個のポインタ (現状では t=8)

multicast の実現

X が状態変化
  join / leave / adjust level
event-report message
  to a top node
その top node が tree-based multicast を開始
Figure 6

その top node のレベルが 0 だったとして
(assume the top node is level-0)
  自分の nodeId と最上位bitが異なるノードに送信
  (sends the event to a node whose first bit is different)
  自分の nodeId と第二bitがはじめて異なるノードに送信
  (then to another node whose second bit is different)
  ...
とすすめる
これだけで「すべての」ノードに multicast 可能
(after that, all nodes receive the multicast)

これを audience set に限定すれば OK
(limit the multicast to the audience set)
送信の際, 最もレベルが高いノードに送る必要

complete
  audience set 全員に届く
non-redundant
  メッセージの重複なし
  (each node only receives the event once)

+ レベルが高いノードほど送信量が大きい
  (Higher-level nodes have more out-degrees than lower-level ones)
+ log_2(A) ステップで完了 (A: audience set のサイズ)
  (An event can reach all the nodes in the audience set through about steps)
+ multicast tree は自律的, 動的にきまる
  (The multicast tree is neither unique nor pre-determined)

multicast の際に acknowledgement 必要
  (acknowledgements are required for all the multicast messages)
  返事がない - node failure として処理
  (When a message gets no response after three continuous attempts,)
  prefix table から消す
  (the corresponding pointer will be removed from the prefix routing table)
  別のノードに送信
  (and the message will be redirected to a new target node)

B. Failure Detection
通知なしの離脱をどうするか
(nodes may leave the system without notification)

固有prefixが共通の（レベルも同じ）ノードは
(nodes with the same prefix eigenstring)
  全員が全員を知っている
  完全結合 (fully connected)

Fig.7 のようにリング状に並んだと考えて
  定期的に right neighbor を probe
  (each node in this circle probe its right neighbor periodicallyi)
反応がなければ, fail として, 代理で leave を報告
(Once a node detects the failure of its right neighbor,
 it immediately reports the event to one of its top nodes
 and redirects its probe to the next right neighbor)
  top node は共通だから代理が勤まる

concurrent failure があっても大丈夫
(resilient to concurrent failures)

ただし, 同じ固有prefixのノードが一つだけだと
(a special case: only one node with a certain prefix eigenstring)
  これはうまくいかない
  次節の refreshing で対応

? A と B の間の通信だけができなくなったらどうなるか
Aは「B が fail」と誤解して, top nodeに報告してしまう
  top node は, まず B 自身の生存を確認するのかな

C. Refreshing
multicast は不完全
prefix table に誤りの可能性 - two types
absent pointer
  ノードが存在するのに table に載ってない
  (an existing node is not in the table)
  そのノードが leave する時に正しくなる
  (would be automatically revised when the node leaves)
stale pointer
  table に載っているのにノードが存在しない
  multicast で返事がないことで発覚して修正

誤りを蓄積させないために - refreshing mechanism
(to guard against the accumulation of errors)

prefix routing table 中のノードの寿命を計測
(measures the lifetime of all the nodes in its prefix table)
レベル毎の平均寿命を算出 LT_i
(calculate the average lifetime at each level)
レベル l のノードは 2*LT_l の間隔で自分の状態を audience にマルチキャスト
(a level-l node multicasts its state around its audience set every 2LT_l)
レベル m のポインタが 3*LT_m のあいだrefreshされなければ

(if a level-m pointer is not refreshed for 3LT_m,)
  (つまり上述のマルチキャストを受信しなければ)
  prefix routing table から除去
        (it is removed from the prefix table)
absent も stale も対処できる
error rate の上限を限る効果

平均寿命の2倍を生きるノードは少ないので
(lifetime of node is much shorter than twice the average)
  実際にはほとんど起動されない
        (most nodes never perform refreshing multicast)

D. Top-node List の維持
event-report を a top node に送信
  その確認応答として, T-1 個の top node list がついてくる
  (t-1 pointers to other top nodes are piggybacked on the ack)
  これを受けて自分のリストを更新できる
  (use them to update local list)
top-node list の全ノードがだめだったら
(if all the top nodes are unavailable)
  prefix table から他のノードを選んで, そいつに聞く
  (asks another node in the prefix table for his top-node list)

E. Suffix Routing Table の維持
prefix と全く同様に, 独立に行なう
(maintained independently)
change-event の際には, 二つの multicast が発生することになる
(results in two multicasts)

F. Joining and Level-adjustment
新規ノード X が参加
  bootstrap node B に接続
X の 各種設定に協力
  level
  prefix top node Tp, suffix top node Ts
level
  Bのレベル + log(Bの帯域/Xの帯域)
  (level_X = level_B + log(bandwidth_B / bandwidth_X)
Tp
  [Bのtop = Xのtop] なら, リストから適当に選んで渡す
  そうでないなら B は, suffix table から C を選んで任せる
  a) Cの固有prefix ⊂ Xの固有prefix
  b) Xの固有prefix ⊂ Cの固有prefix (論文typo)
  c) Xの固有prefix = Cの固有prefix
  C は X に Tp を渡す
Ts についても同様

routing table
  X は Tp, Ts からダウンロード (topは全員知ってるから)
  (X downloads its prefix/suffix routing table and top-node list from TP and TS)
  転送は複数のノードが並列に行なっても良い
  (They can ask some other X's super nodes to transmit it in parallel)

Tp, Ts は X の参加を multicast する

warm-up
  参加の際, レベルを低く設定しておけば
  (A joining node can also first set a low level)
    短時間で参加処理が完了
    (so as to start working in a relatively short time)
  完了後にレベルを上げる
  (After completing the back-ground downloading, it raises it level)
  起動時間の短縮

実行中のレベル変更 (adjust level at runtime)
  新たに必要となる table エントリを
    supernodes から入手
    ?? supernode がいない場合、つまり自分がtop

?? おそらくsuffix tableを使うのだろう
その後で top node へ level-shift event 報告 -> multicast

下げる時は table エントリを捨ててから報告

G. Backup Routing Table の維持
参加処理で prefix/suffix tableを埋めた後
(after downloading its pre-fix/suffix routing table)
suffix table の適当なものを backup table に入れる
virtual pointer
これで完全に埋まらない場合は
(If there are some items that cannot be satisfied)
physical pointer
a super node (or a top node) に要求

維持のために, physical pointer を定期的に probe
30秒に一回
stale が見つかったら
super node から代わりを入手

普通はphysical pointerはないから
probe のオーバヘッドは少ない

H. Summary
table の維持に必要な帯域
(bandwidth required for maintenance of routing table)
1) event-distribution multicast (join, leave, level change)
2) failure detection (ring)
3) backup routing table

1 が主要な帯域
several messages per second (weak nodes)
hundreds of messages (powerful node)
2 は固定
0.2 messages / sec
3 は普通はゼロ
10個で 0.5 messages / sec

レベルによって帯域の制御ができることを示している
(adjust the size of its prefix/suffix routing table
to control its bandwidth cost)

IV. LOCALITY-AWARE ROUTING
routing protocol で, 2-hop の時に, 次のノードを選ぶところで自由度
(many candidates when choosing the next hop from the suffix table)
ここで, network distance が近いものを選ぶと有利
たった2-hop のうちの一つだから

network distance
suffix table が大きいと, それぞれの測定は困難
distance prediction
いろいろあるらしい
GNP (Global Network Positioning) を採用

16個の well-known nodes
各ノードはそれらへの距離を測定
(Every node measures its network distances to them)
その16次元の値を, 自分のポインタ情報に付加
(attaches the 16-dimension coordinates into its pointers)

複数候補から近いものを選ぶ際には
(When there are many candidates for the next hop)
16次元空間で自分と近いものを選ぶ
(choose the one whose coordinates are the closest
to its own in the 16-dimension Euclidean space)

候補が多いと, CPU時間を要する
1,000,000ノードの例では

  平均 5,000 以上の候補がある

  routing efficiency とのトレードオフ

Figure 8
  候補の一部だけを選ぶ方法
  乱数で出発点を決めて
  (from a random start point)
  予定した候補数(N_{cand})だけ調べる
  (The search continues until it has collected N_cand valid candidates)

  N_{cand} は各ノードが適当に決める

V. EXPERIMENT RESULTS
1,000,000 Tourist ノードのシミュレーション
  実際のp2pシステムの寿命分布にあわせて
  (lifetime distribution is consistent with the measurement result
   of real peer-to-peer systems)
16-server cluster
ONSP
  a general platform for overlay protocol simulation
  parallel discrete event
  MPI
time scale 1/26 (26倍の時間を要する)

巨大なルーティングテーブル (keeping large routing table)
  同じ固有prefix/suffixなら, tableも同じ
  ひとつ作ってそれを共有する
  (store a "correct" routing table for each eigenstring and share it)
  error items だけをノード毎に保持
  (each node only stores the error items)

transit-stub model
  120 transit domains
  4 transit nodes
  5 stub domains
  4 stub nodes
  Tourist nodes

  intra-stub 1ms
  stub-to-transit 10~30ms
  transit-to-transit 50~150ms

シミュレーション (simulation)
  1,000,000ノード
  寿命分布 [26]の figure 6 (lifetime distribution)
  新ノードの参加と離脱 (同率)
  (joining rate and leaving rate is equal)
    ポアソン分布 (Poisson distribution)
    1,000,000 nodes/2.3 hour
    2.3 hour は平均寿命

帯域 (bandwidth)
  各ノードは帯域の上限をもつ
  (each node sets an upper threshold)
  利用可能帯域の 1% (1% of its total input bandwidth)
  500bps 以上
  帯域の分布も[26]にあわせた(figure 3)
  (distribution is also consistent with measurement)

Arguments
  failure detection - every 5 sec
    3 continuous failures -> node-leave event
  backup routing table - every 30 sec
    fail -> two more probes (once per 5 sec)
    ask another pointer
  multiple candidate in routing
    choose a random one (N_{cand} = 1) - no locality optimization

event-distribution message
  size: 1000 bit
  0.5 sec delay (for handling)

A. The Common Case
●figure 9 レベルの分布
  11レベルある
  二つのピーク
    5,6 (44%) ケーブル接続 (cable linked)
    10 (23%) モデム接続 (modem linked)
  heterogeneity を示している

●figure 10 レベル毎の帯域コスト
  レベル0で 350kbps
    全帯域は35Mbps以上 (1%以下の条件)
  レベルが増える毎に半分になる
  レベル10 は 340bps
    0.5 message / sec (1000bit message)

入力の方が出力よりも帯域を要する
(output bandwidth is lower than input)
レベル0だけは逆
(except the level-0 nodes)
  multicast のため
レベル0でも total bandwidthの 2%以下

●figure 11 ルーティングテーブルの不正確さ
very few errors in the routing tables
高レベルほどエラーが少ない
(higher-level nodes have fewer errors)
  multicast の方向が 高→低 だから
  (because the multicast direction is from higher to lower)
レベル10でも0.6%以下

理論値は (estimation)
  ノードが離脱すると15秒で発覚
  (node leave will detected after 15 secs)
  つづいてmulticast
  1,000,000ノードだと20ステップ
  1ステップ 250ms とすると
  15+(1+20)*(0.5+0.25) = 30.75 sec
  ノードの平均寿命は135分なので (average lifetime)
  30.75/(135*60) = 0.0037
  大体あってる

routing では ack をおこなう
  stale pointer を発見
  suffix table なら 別ノードへ
  prefix table なら (それがrootだったから) 新rootへ

routing efficiency
  200,000 messages from random node with random keys
  全部 2 hop 以下で完了
●figure 12 (論文の図が 13 と入れ替わっている)
  routing delay の分布
  1-hop, 2-hop を区別
●figure 13
  RDP (routing delay penalty)
    IP で直行したときとの比率 (ratio to IP-level latency)
  15% は 1-hop
  2-hop の50%が 2.0 以上
    locality aware が有効そう
    (sufficient room for locality-aware optimization)

●figure 14
  initiator node のレベル毎のホップ数
  レベル0は常に1-hop

次第に増えて, 最後はほぼすべて 2-hop

B. Locality-aware Routing
suffix table から次ホップを選ぶときに
figure 15
候補数の分布 (distribution of the number of candidates)
80%で2000以上
N_{cand}個の中から選ぶ

GNPを使用

●figure 16
N_{cand} を変えた時の delay
●figure 17
N_{cand} を変えた時の RDP (penalty)

2-hop メッセージのみを対象
N_{cand} が100でも大幅改善
2000より大きくしてもあまりかわらなくなる

C. Scalability
総ノード数を変化させてみる (change the system scale)
通信量の変化によって, レベルが低下するはず
(nodes lower their levels gradually)

●figure 18 ノード数とレベルの分布
500ノードだと全員 レベル0
ノード数が増えると低レベルがでてくる
(more levels and fewer nodes at high levels)

●figure 19 hop 数の分布
500ノードならすべて 1-hop
5M ノードでは
レベルは14まで
それでも全メッセージは 2-hop以下

D. Adaptivity
参加離脱 (churn rate) の変化

●figure 20
平均寿命の変化とレベル分布
●figure 21
平均寿命の変化とホップ数

平均寿命を common case の r 倍とした
ノード数は 1M

churn rate 高い
レベル分布が広がる

9.4日だと (平均の100倍)
90% がレベル0
90% が 1-ホップ

ノード数が多くても, 安定していれば 1-hop

1.35分だと (平均の1/100)
最低ノードはレベル17
レベル0はもはやいない
最高でもレベル3
メッセージの30%は 3-hop になった
first hop must be via backup routing table