(2) N の routing table を作る (c)
(N initialize its routing table)
  (1)で通信したノードで level p を埋める
  (fill level p using nodes contacted in step 1)

  level p より上を埋める
  (fill levels lower than p)
  つぎに, それらのノードからそのレベルの back pointerをとりよせ
  その back pointerの集合がつぎの neighbor set になる
  (use backpointers to update neighbor set)

  さらに, 通信を受けたノードはその際に自分の表も改善 (d)
  (contacted nodes in the process will improve their tables)

同時参加の場合 (simultaneous insertion) (SPAA 4.5)
  ほとんどの場合は, 相互に関連があることはない
    それぞれの表が重ならない
    (tables does not intersect usually)

  場合によっては, 相手が表に入るべきケースあり
  (in some cases, intersection may occur)
    よりよいエントリとして (as a better entry)
      性能の劣化 (performance degrade)
      時間をおいて表を再構成すればよい
      (rerun table building)
    唯一のエントリとして (as the only possible neighbor)
      検索に支障をきたす可能性
      (queries may fail)

  後者の問題が起きないように acknowledged multicast を行なう
  (keep state in the multicast)

2)自発的な離脱 (voluntary deletion) (SPAA 5.1)
  back pointer を使って通知
  (notify neighbors using back pointers)
  自分の代わりになるノードも知らせる
  (along with displacement node from its routing table)
    自分の表を使って
    必ずしも最適とは限らない
    あとで改善
  自分が root になっている object の参照を新 root にまわす
  (adjust object references)
    re-publish

3)非自発的な離脱 (involuntary node deletion) (SPAA 5.2)
  routing table エントリの冗長化
  (redundancy in routing tables)
  定期的な beacon で detect
    repair の起動
    object の redistribution

IV. Tapestry Node Architecture and Implementation

A. Component Architecture
Figure 6 - functional layering

router
  routing and location messages
  Figure 7 - object location process

  upcall - message の到着(自分宛以外でも)を知らせる
    (notify application layer of the arrival of messages)
    必要性については後述 (explained in the next section)
  object pointer
    subscribe で記録されているもの (P.4 Fig.5)
    保持者への直接リンク (direct link to the copies)
  nextHop

Fig. 3

dynamic node management
  node arrival and depature

  routing table と object pointer database は常に変化
  dynamic node management による
    arrival, departure
  neighbor link による
    リンクの状態やlatencyの変化

neighbor link
  secure but unreliable datagram
  IP addr and port で open
  may periodically close some connections

  continuous link monitoring
    soft-state keep-alive messages
    latency and loss rate

transport
  TCP/IP, UDP/IP のふたつがある

B. Tapestry Upcall Interface
API は III-A で紹介済み
  PublishObject
  UnPublishObject
  RouteToObject
  RouteToNode
アプリケーションによってはそれ以外にも必要
(other API is required by specific applications)
  例えば multicast
Deliver
  自ノードへのメッセージの到着
  (invoked on incoming message destined for the local node)
Forward
  upcall-enabled message の到着
Route
  次ノードへの転送を指示
  (invoked by application to forward the a message)

C. Implementation
Java
57,000 lines
(詳細省略)

V. Evaluation

A. Evaluation Methodology
環境は三通り (three platforms)

micro-benchmark
  local cluster

PlanetLab
  network testbed
  100 machine
  北米, 欧州, アジア, 豪州
  Figure 10 - ペアごとの ping
  constant load, data loss, node failures

local network simulation
  Simple OceanStore Simulator(SOSS)
  network delay を simulate
    ただし packet loss などはない

B. Performance in a Stable Network
1) Micro benchmarks

2ノードでメッセージの処理性能を計測
(measure message processing overhead between two nodes)
メッセージサイズを変えながらlatencyを測定
(measure the latency for various message sizes)

まず, 一台のマシン上でポートだけ変えて測定
(on the same machine, to elimitate the network delay)
  network delay を除外
  Figure 11
  メッセージサイズが小さいと定数
  (constant for small messages)
  大きくなると比例
  (linear relative to the message size for large messages)
    copy のコスト
  プロセッサの性能比とよく合っている
  (proportional with the increase in processor speed)

スループットも測定 (throughput)
  Figure 12
  100Mbitのイーサネットでも測定
  (also measured through a 100Mbit ethernet)
    4KBメッセージで上限
    (maximum bandwidth utlization for 4KByte message)

2) Routing Overhead to Nodes and Objects
400 nodes
62 PlanetLab machines
RDP(Relative Delay Penalty)
  overlayを使ったlatency / shortest IP distance (ping)
90%点, 中間値(median), 最小値(minimum)

to Nodes (Figure 13)
  RDP: 全てのペア間の往復時間 / ping 往復時間
  (relative Delay Penalty: (routing on overlay) / (RTT by ping)
  中間値は 3 からはじまり 1 に近づいていく
  (median value start at 3, decrease to 1)

  近いノード間でのRDPの異常
  (inflated RDP for short latency paths)
  同一マシン上での複数ノードが問題
  (incurred by multiple nodes on a machine)
    OS のスケジューリング遅延
    (scheduling delay by the OS)
    ping は近いし
    (unrealistically low RTT by ping)

to Objects (Figure 14)
  一台のノードに10000個のobjectを置き
  (10000 objects located on a single node)
  399ノードがそれらにメッセージを送る
  (399 nodes send messages to each of the objects)

  距離が遠い場合は良好
  (good at large distances)
  近距離では, 悪化するし, ばらつきも増大
  (diverges significantly at short distances)

  近距離での余分なホップの重みが増大するため
  (extraneous hops taken while routing at short distances)

3) Object Location Optimization
extra object pointer を object の近くに配置
  記憶量とのトレードオフ
  (tradeoff between space and faster routing)
publish path に対して
  k backup nodes of the next hop
  l nearest neighbors of the current hop
    (near - network distance)

  first m hops of the path
  (k + l) * m extra pointers

Figure 15
SOSS simulator, 1092 nodes
90%点の推移
非常に改善されている
(lower RDP significantly)

C. Convergence Under Network Dynamics
1) Single Node Insertion
1ノードが参加してから安定するまでの時間, 帯域
(time required for the network to stabilize and the control traffic bandwidth)
ノード総数を変えながら
(x axis: network size)
Figure 16 - latency
Figure 17 - bandwidth (横軸は対数)

ノードの持つrouting table は log(N)
結果もそうなっている
(latency scales sublineary)

リンクひとつひとつについては, 帯域はずっと低い
(lower bandwidth for single link)

2) Parallel Node Insertion
Figure 18
200ノードでスタート
(initias size is 200 nodes)
横軸は同時参加ノードの比率
(x axis: ratio of inserted nodes to the total nodes)
縦軸は収束時間
(y axis: convergence time)
中間値はほぼ線形
(median time scales)
90%点のバラツキが大きい
(90% value can fluctuate)
  とくに同時参加が10%以上のとき
  (greater or equal to 10%)
  node virtualization のためだろう
  (multiple nodes on a machine)
  scheduling delay

3) Continuous Convergence and Self-Repair
SOSS上で1000ノードで
測定したのは
  success rate of routing
  bandwidth
条件は二種 (two modes of network change)
  menbershipに大変化 (drastic change)
  slow constant churn (「かき回してバターを作る」)
routing の宛先は失われないようにしてある
(owner of the objects are guaranteed to stay alive)

大変化 (drastic change)
  20% nodes が fail
  50% nodes が inserted
Figure 19 (routing to Node)
Figure 20 (routing to Object)

success rate はすぐ 100% に戻る
大量参加の直後に帯域にスパイクが出るがすぐに戻る
(bandwidth spikes, then levels off)

churn
  確率的な参加と離脱 (probabilistic inserts and fais)
  参加 - ポアソン分布 (insert: Poisson distribution)
  離脱 - 指数分布 (failure: exponential distribution)

  churn 1(実験の前半) - 20 sec / 4 min
  churn 2(実験の後半) - 10 sec / 2 min
Figure 21 (routing to Node)
Figure 22 (routing to Object)
success rate
  まれに100%から少し下がるだけ
  パラメータに依存しない

PlanetLabでの実験
  大変化とchurn
Figure 23
reliability
  failure, join の後にshort dip
  churn の間はdipするが, 終わると回復

Tourist: Self-Adaptive Structured Overlay

Jinfeng Hu, Ming Li, Hongliang Yu, Weimin Zheng

I. Introduction
structured overlay
  nodeId(node), key(object)
  distance in numerical space (in many systems)

二つのカテゴリ
  O(logN)-hop
    small routing table
    routing table maintenance
      low cost
      explicit probing
    suitable for large and dynamic
  O(1)-hop
    large routing table
      tablesize: O(N) or O(sqrt(N))
      one-hop or two-hop
    routing table maintenance
      broadcast or multicast
      much larger cost
    suitable for small or stable
どちらを使うかを事前に予想するのは困難
二つの方法を動的に切替えることは困難
(it is difficult to select scheme beforehand,
 and to switch between these two schemes after the system started)

self-adaptivity - Tourist

1) small and stable system
  Tourist is one-hop
2) turns larger or more dynamic
  between one-hop and two-hop
3) very large and dynamic
  1,000,000 nodes, lifetime about 1 hour
  within two hops
4) extremely large and dramatically dynamic
  O(logN)-hop guarantee

node の heterogeneity を利用
  "level"
  larger bandwidth -> higher level
    larger routing table
    faster routing
    higher maintenance cost
level を 動的・自律的に変更
(change its level dynamically and autonomously)

II. Protocol Overview

nodeId, key
  128bit
distance
  XOR-based
  X, Y の距離は, X^Y で定義
    一致すれば 0 (0 if X = Y)
    上位bitが違うほど大きくなる (large penalty for higher bits)
routing
  key k -> root node
  XOR 的にもっとも近いノードへ
  (nearest in the sense of XOR distance)
二つの routing strucure
core structure
  one or two hop (O(1))
backup structure
  log(N) hop
  core structure で十分なときは使わない
    no real pointers -> no overhead (後述)
  Plaxtonの方式 (Pastryとかの方式)

A. Core Structure
level: 0,1,2,... (0: highest level)
  self-determined (autonomously)
two symmetric routing tables
  prefix routing table
  suffix routing table
l - 自分の level として
prefix routing table
  自分と l-bit prefix が共通の全ノードを保持
  (contain ALL nodes which share l-bit common prefix)
  上位が共通 - 自分と(id的に)近いノード
  (higher bits are common - nearby nodes)
suffix routing table
  l-bit common suffix
  下位が共通 - 全空間に散在
  (lower bits are common - scatter evenly in the space)
Figure 1

保持する他ノードの情報 (a node pointer consists of)
  IP, ポート番号 (port number)
  nodeId
  level