Scalable node addressing and message routing for global computing

上田陽平

Peer-to-peer (P2P)

- Global computing の一種
- 世界中の多数の計算機が参加
- 中央に管理サーバがない
- 計算機は頻繁に参加・脱退

P2Pで解決すべき問題

- ノードへのアドレスの割り当て
- ノード間のmessage routing
- Fault tolerance

ノードへのアドレスの割り当て

- IP address では不十分
  - 全ノードの IP address を管理するのは困難
  - Private address が存在
- IP アドレスによらない addressing が必要
- Addressing は scalable であるべき

ノード間のmessage routing

- 巨大なノード数に対応できるroutingが必要
- Routing table はコンパクトであるべき
- ほとんどのノード間で通信が可能
  - ほとんどのノードが隣接している
- O(log N) hop 程度のroutingで十分
  - (N：参加ノード数)
- Minimum-hop や neighborly は必要ない

Fault tolerance

- 大規模になればなるほど、故障が多くなる

- Network failure の検出が必要
- Node failure からの復旧も必要

我々の方針

- Phoenix model[田浦'01]を一般のグラフに拡張したものを用いる

Phoenix model on General graphs

- ノードに整数の区間(interval)を割り当てる
  - e.g. [0,16)= {0,1,…,15}
- 各ノードの区間はdisjoint
- 全ノードの区間の和集合はひとつの区間
  - $[0,2^{64})$
- 各ノードの区間は1個 (例外あり)

Message routing

- メッセージには64bit整数のkeyをつける
- メッセージはkey を含む区間を持つノードまでroutingされる

Routing table の作り方(1)
区間の分割

- 最初のノードは$[0,2^{64})$が割り当てられる
- あらたに参加するノードは参加済みノードから区間を半分分けてもらう(例外あり)
- 分けてもらう相手は、ランダムにkeyを生成して、Joinメッセージを送ることで見つける

区間の分割 (例)

区間の分割 (例)

a

b

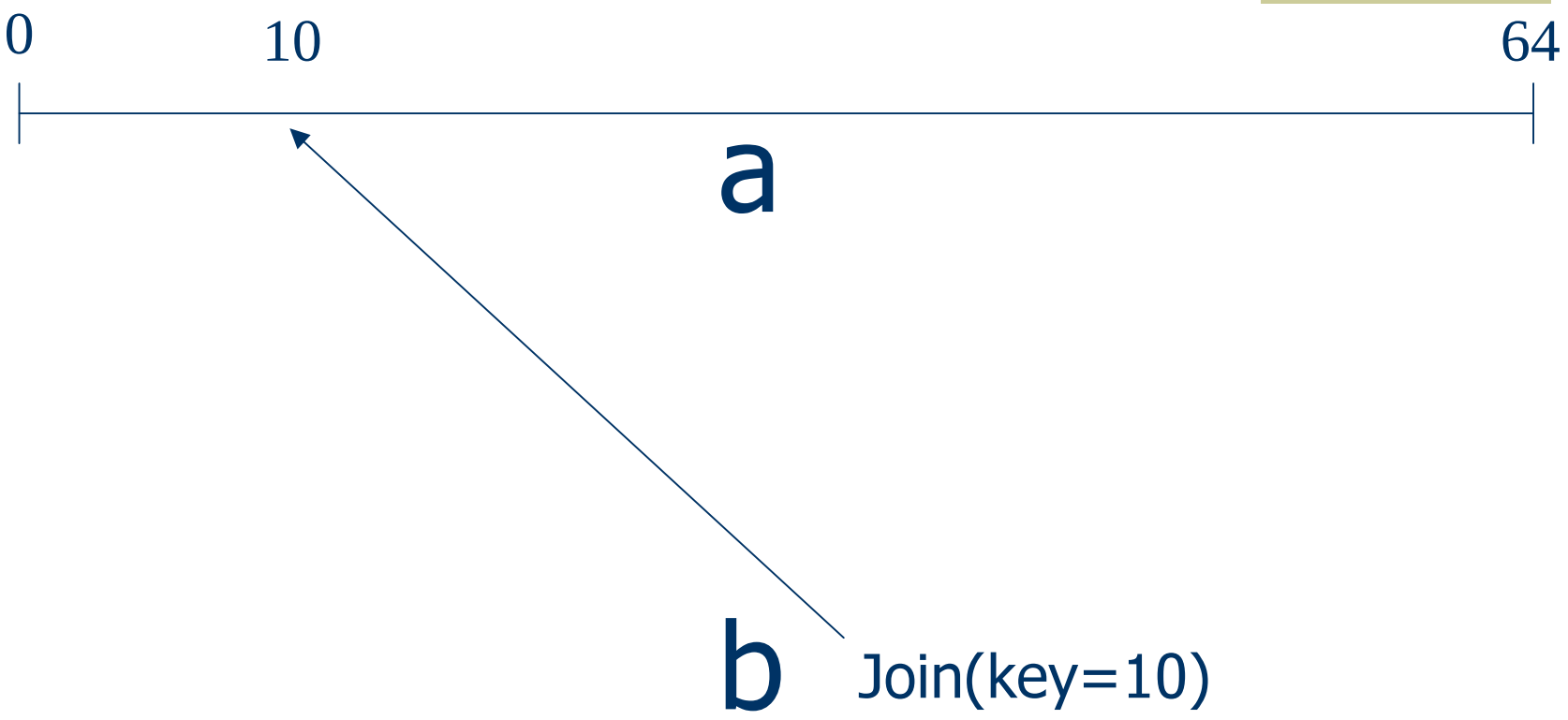
区間の分割 (例)

b Join(key=10)

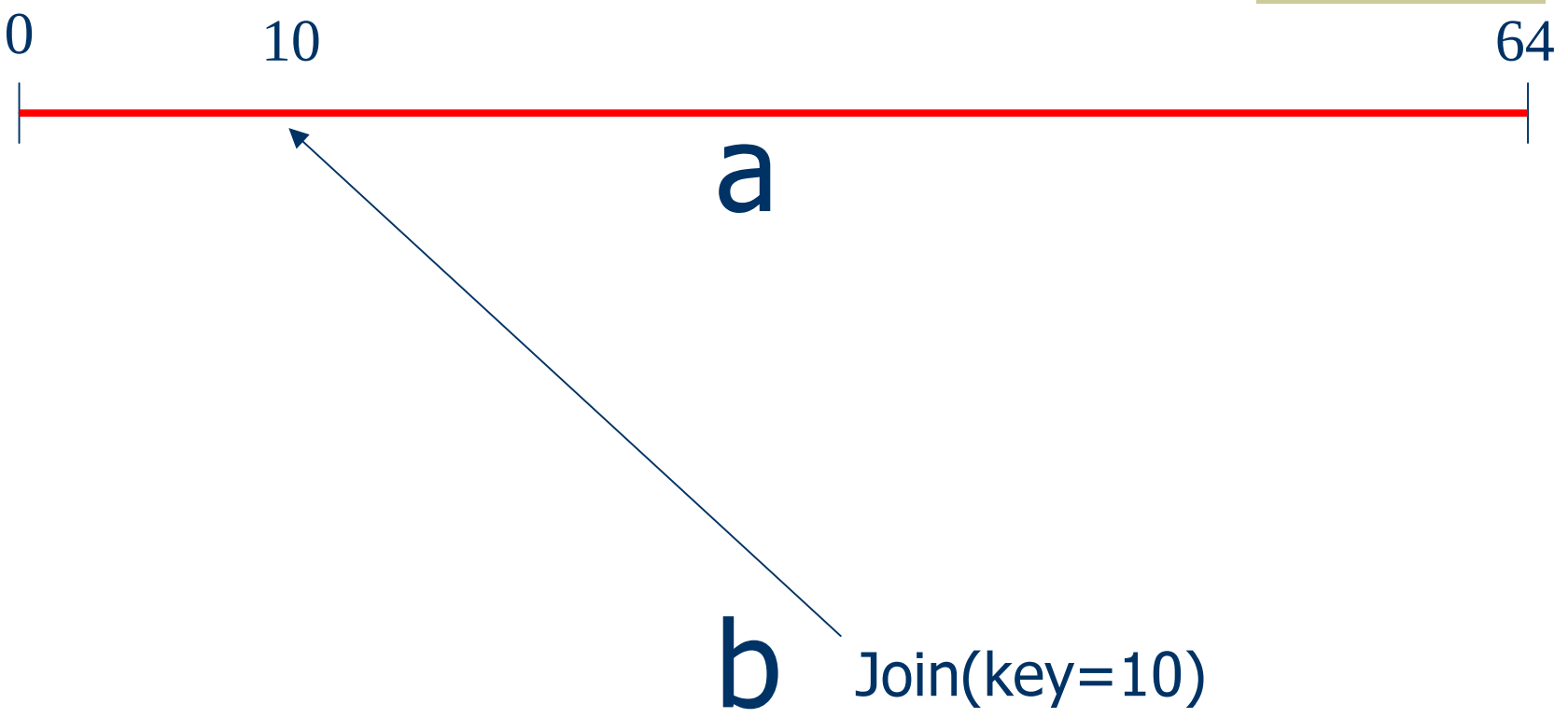
区間の分割 (例)

b Join(key=10)

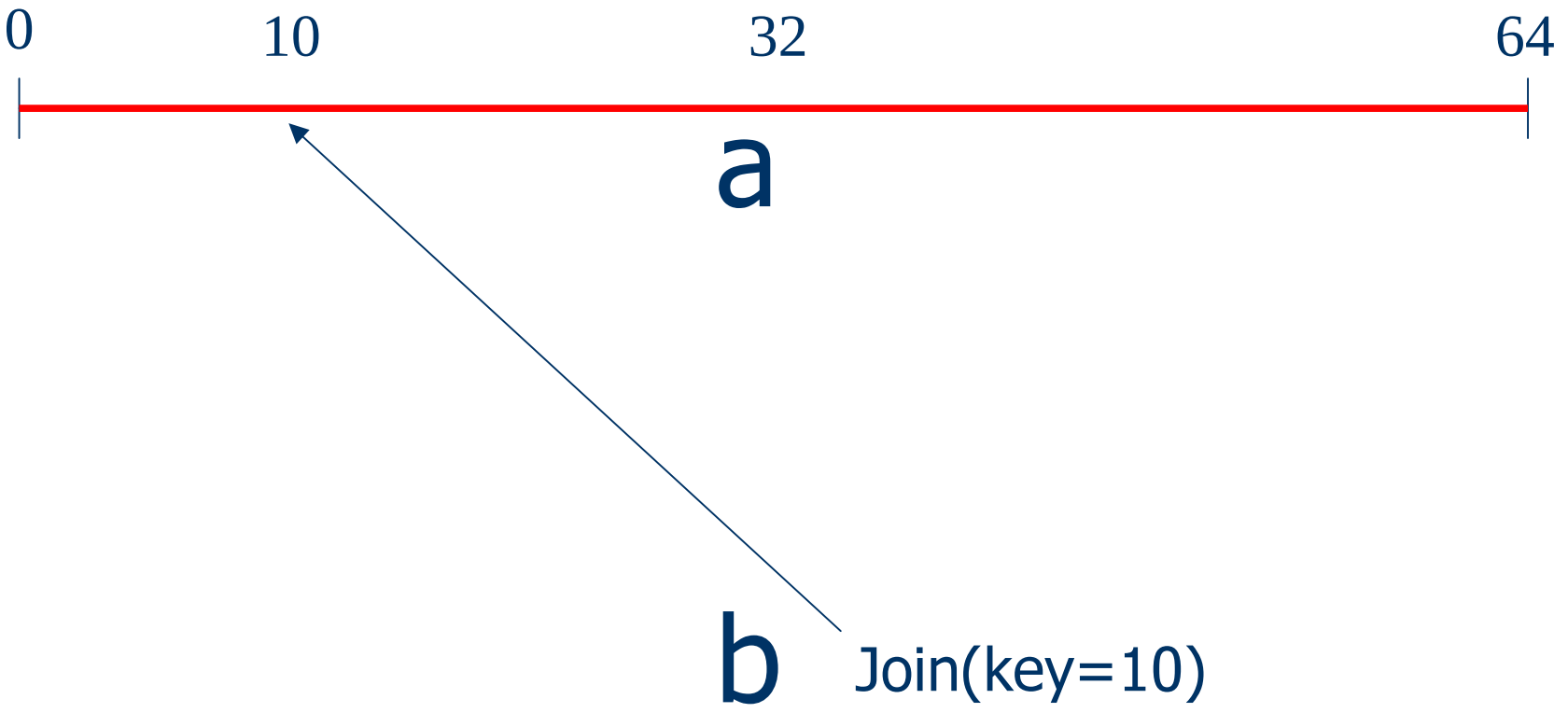
区間の分割 (例)



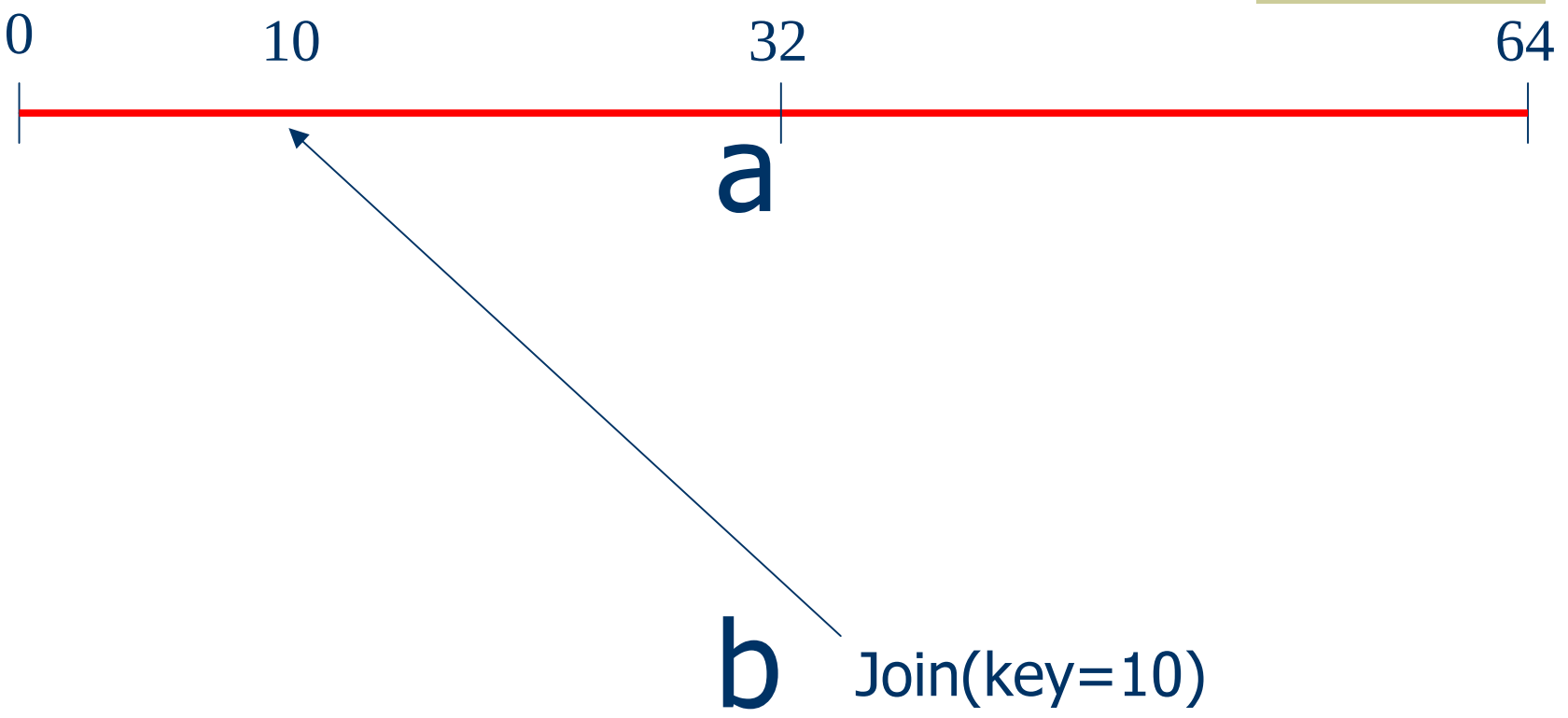
区間の分割 (例)



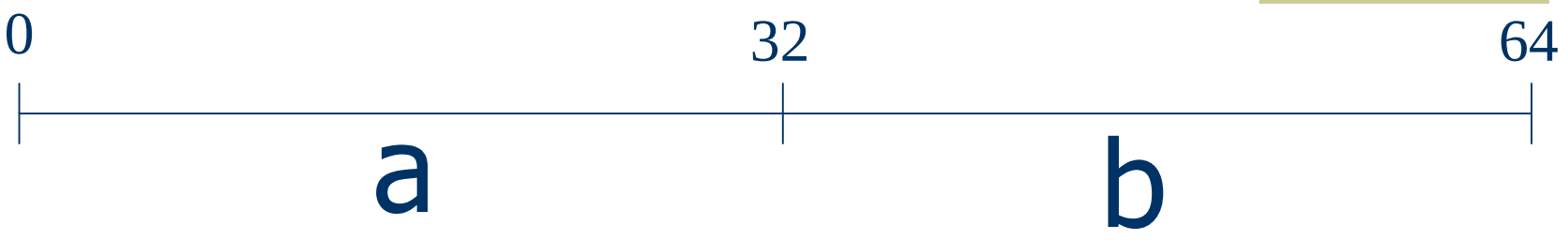
区間の分割 (例)



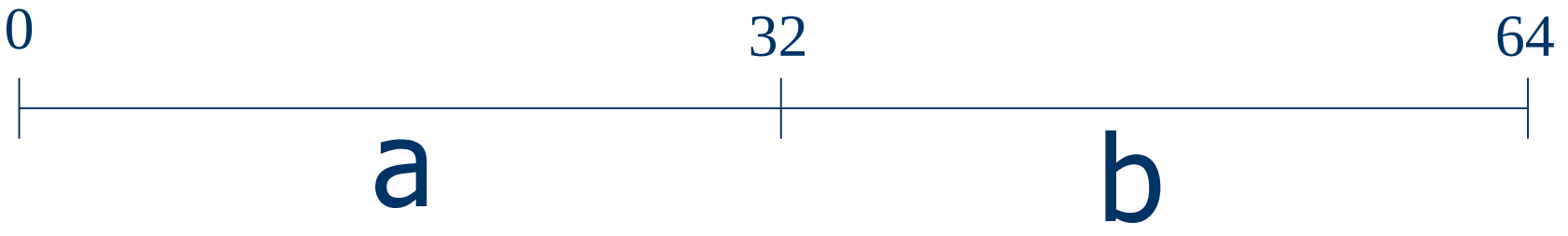
区間の分割 (例)



区間の分割 (例)

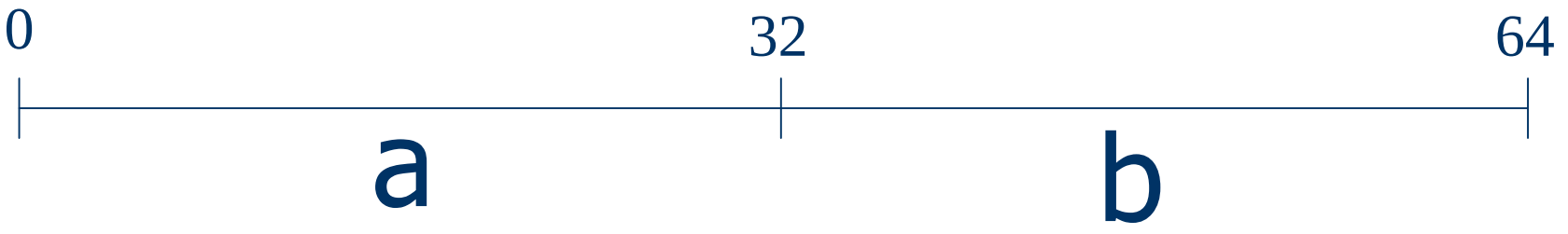


区間の分割 (例)



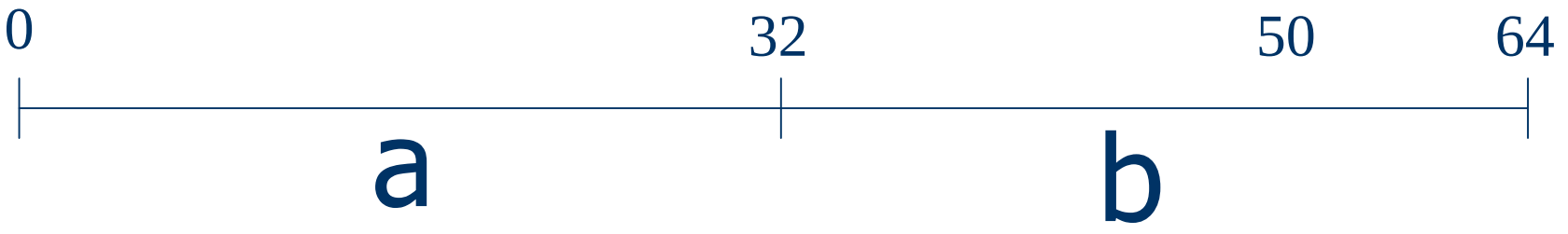
c

区間の分割 (例)



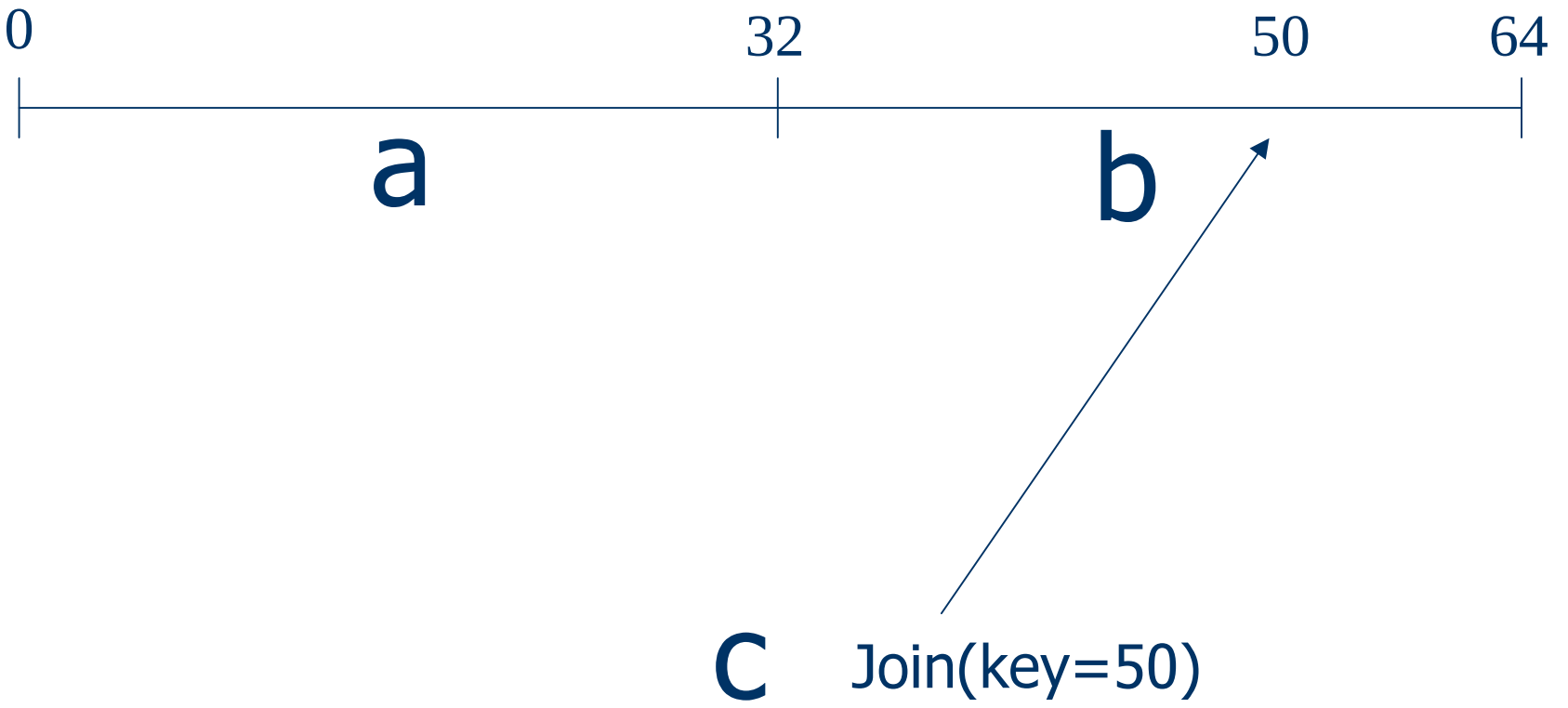
c　Join(key=50)

区間の分割 (例)

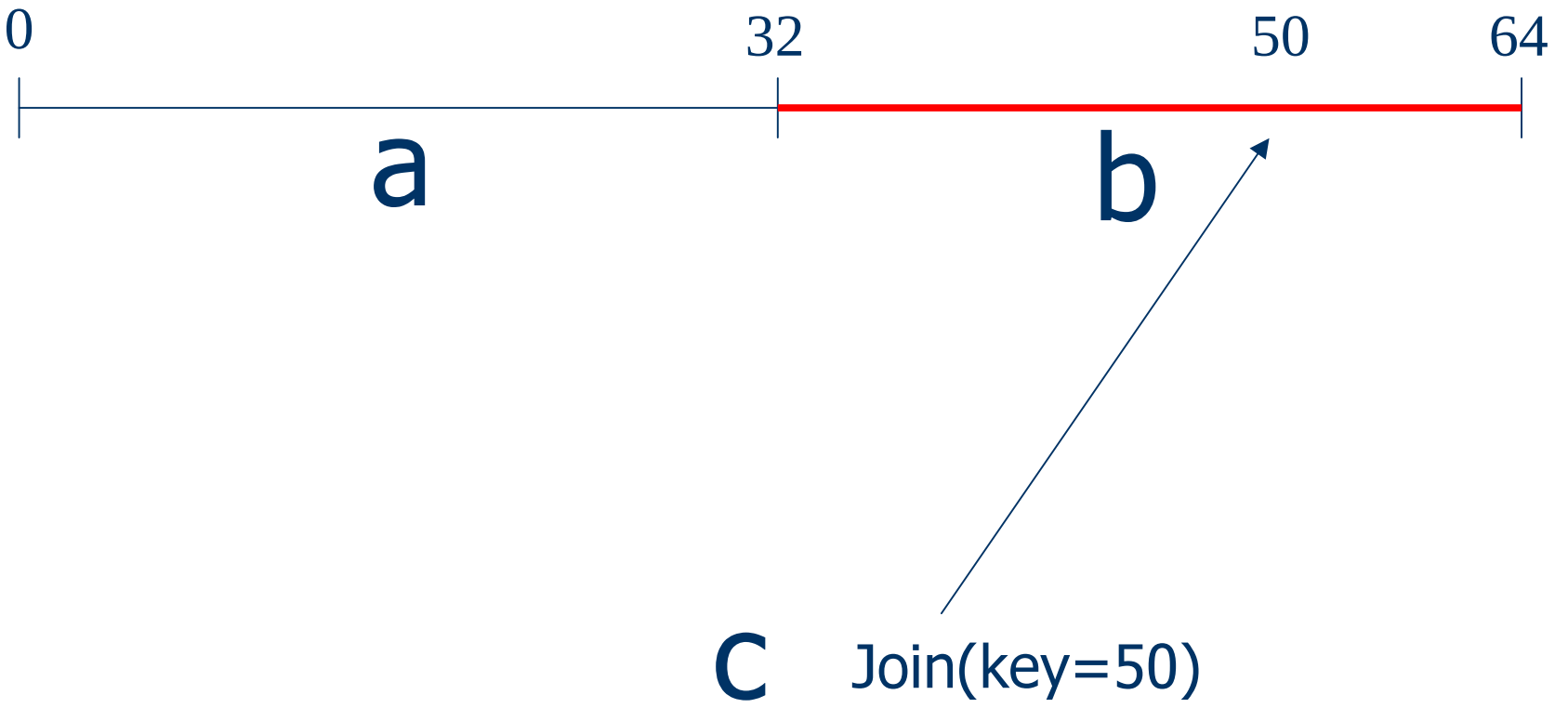


c　Join(key=50)

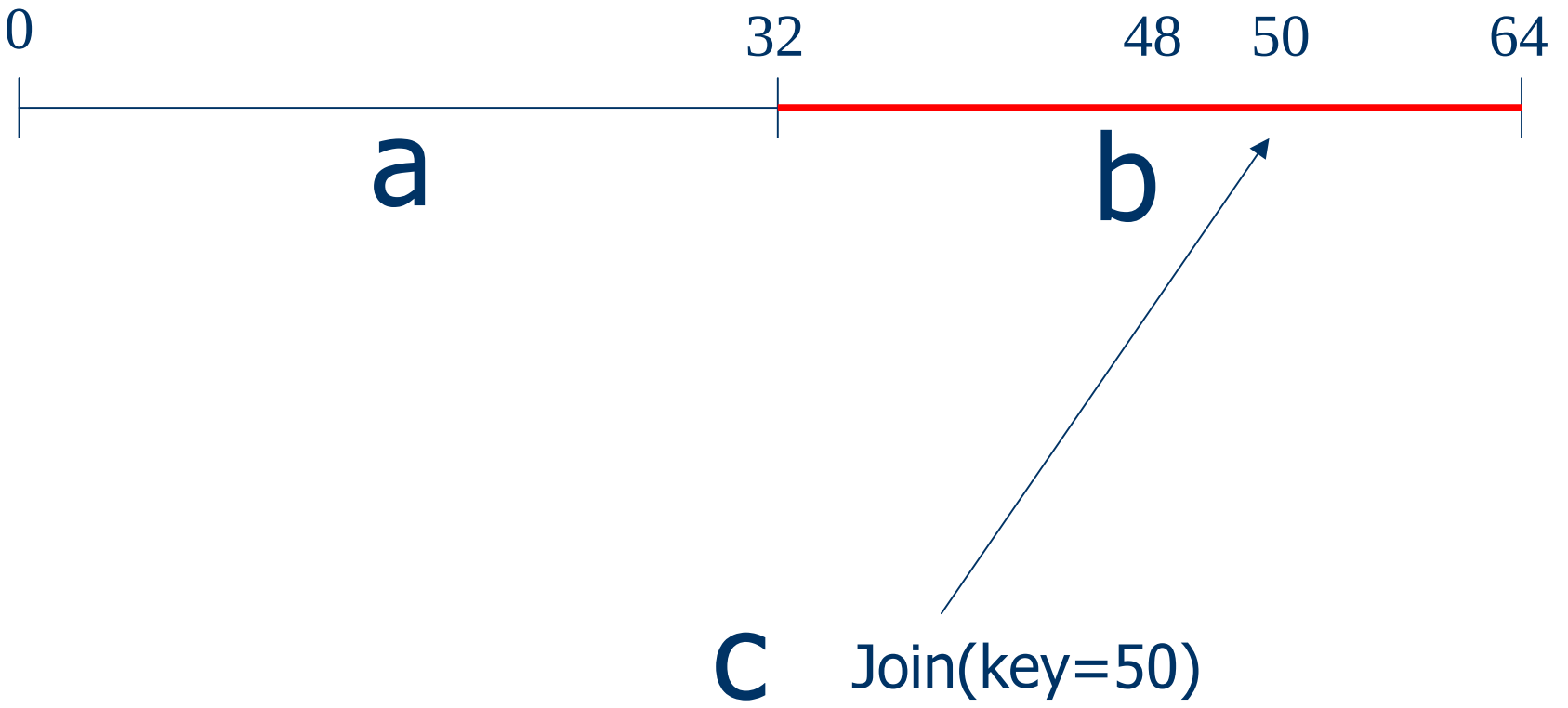
区間の分割 (例)



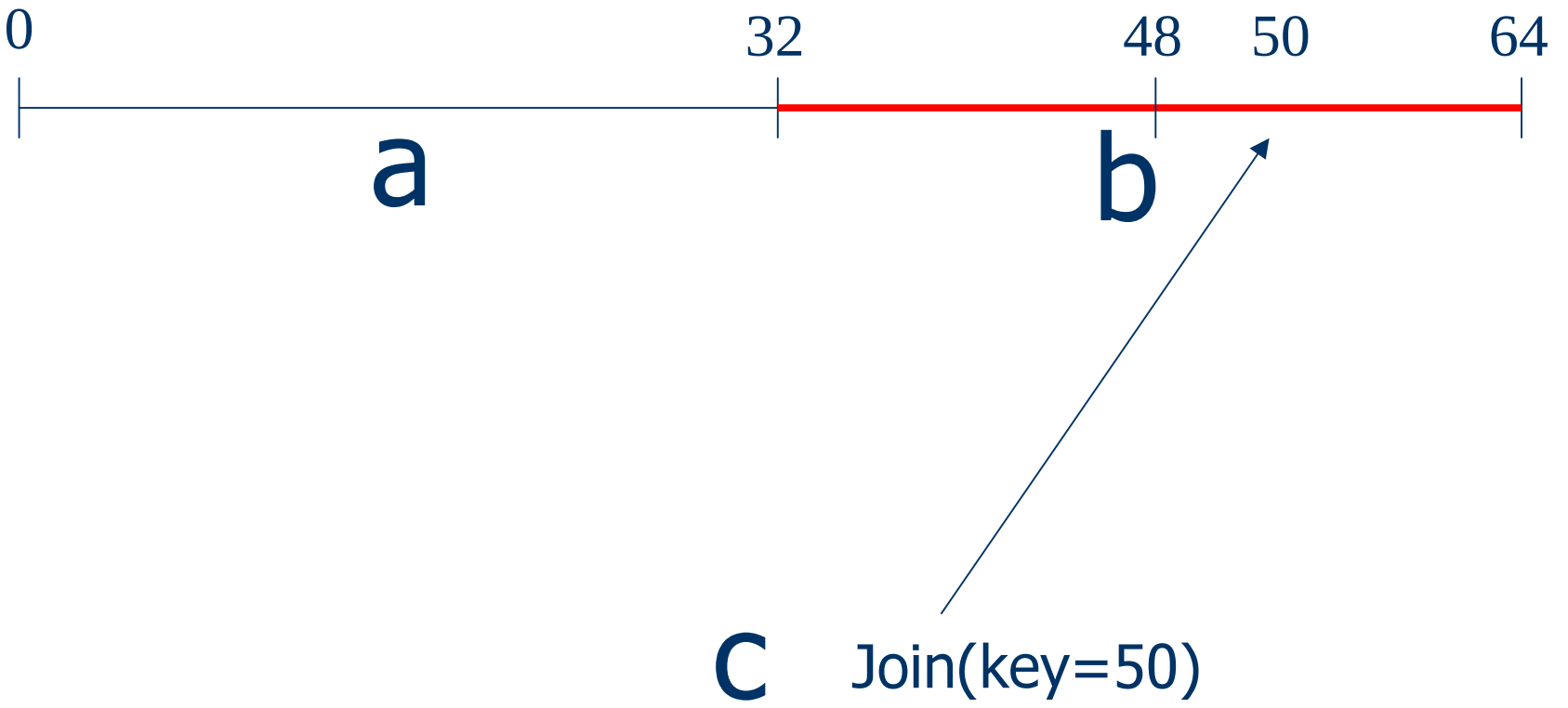
区間の分割 (例)



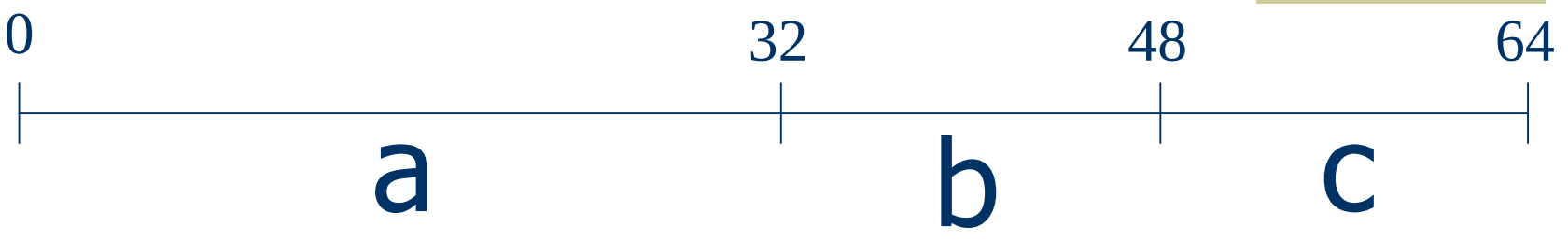
区間の分割 (例)



区間の分割 (例)



区間の分割 (例)



Routing table の作り方(2) cluster

- Level 0 のcluster
  - ノード全体
- Level i のcluster
  - Level $i - 1$ のclusterの区間をb等分した区間のひとつに属するノード全体

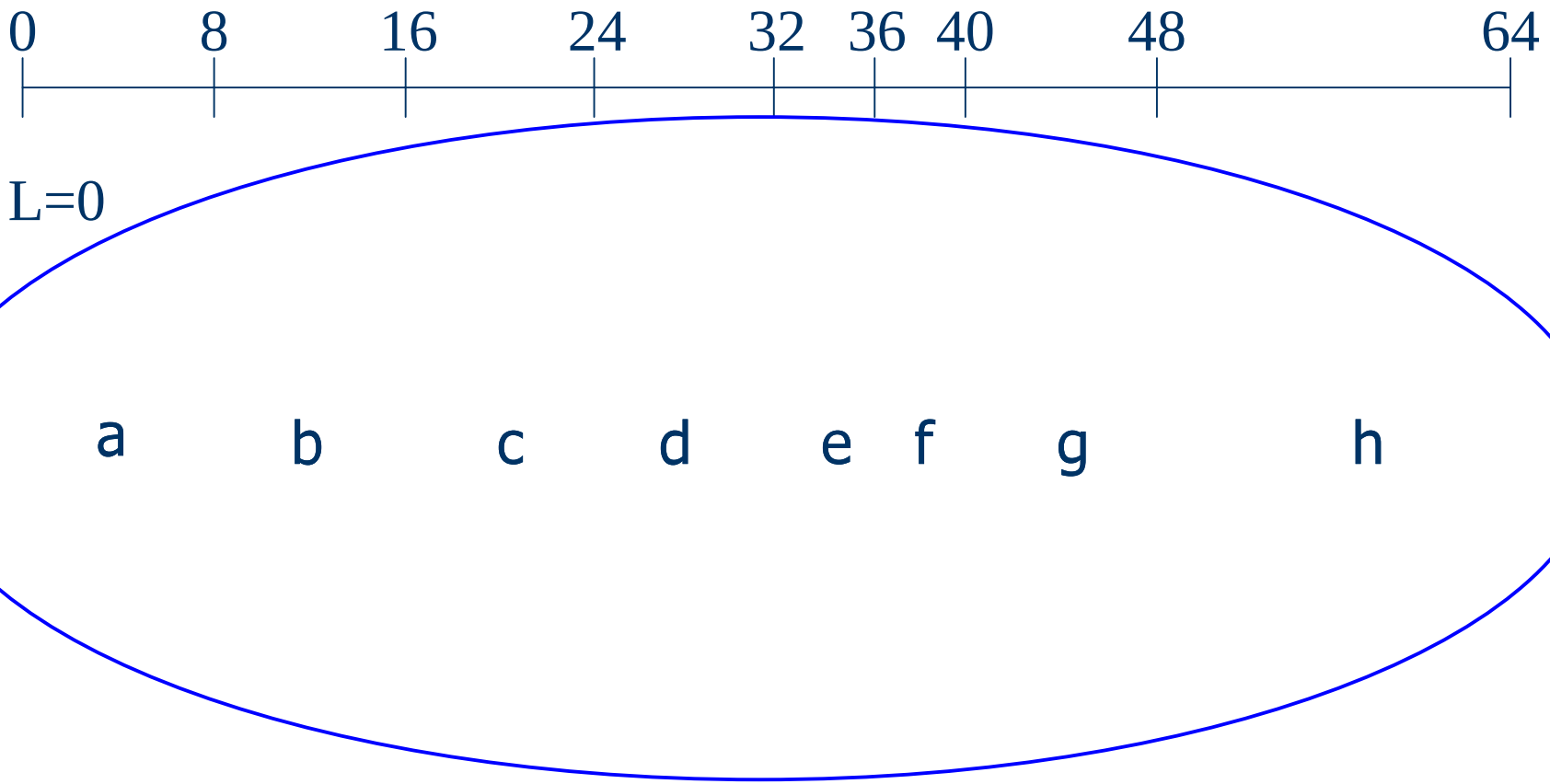
cluster (例)

- $b = 2$

a b c d e f g h

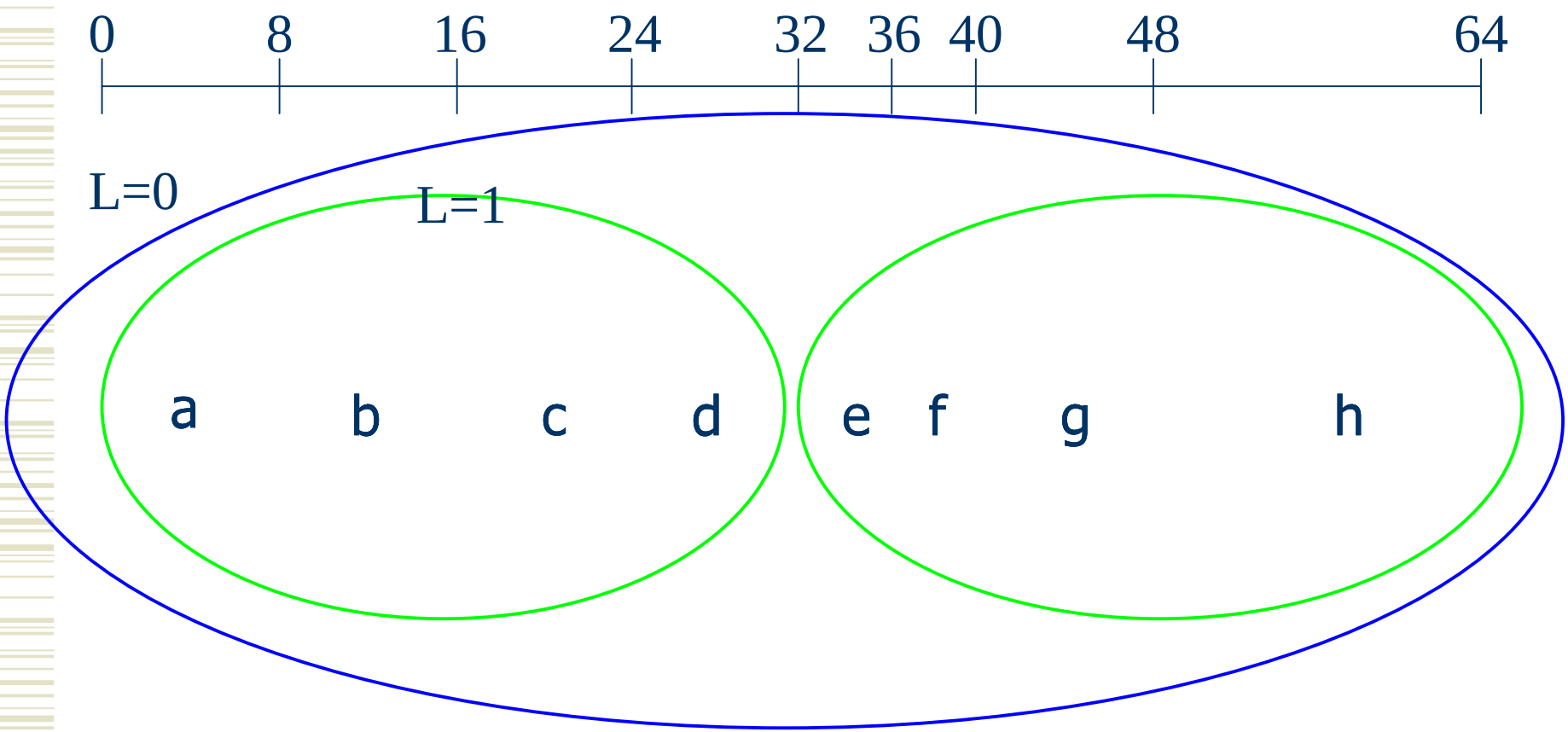
cluster (例)

- $b = 2$



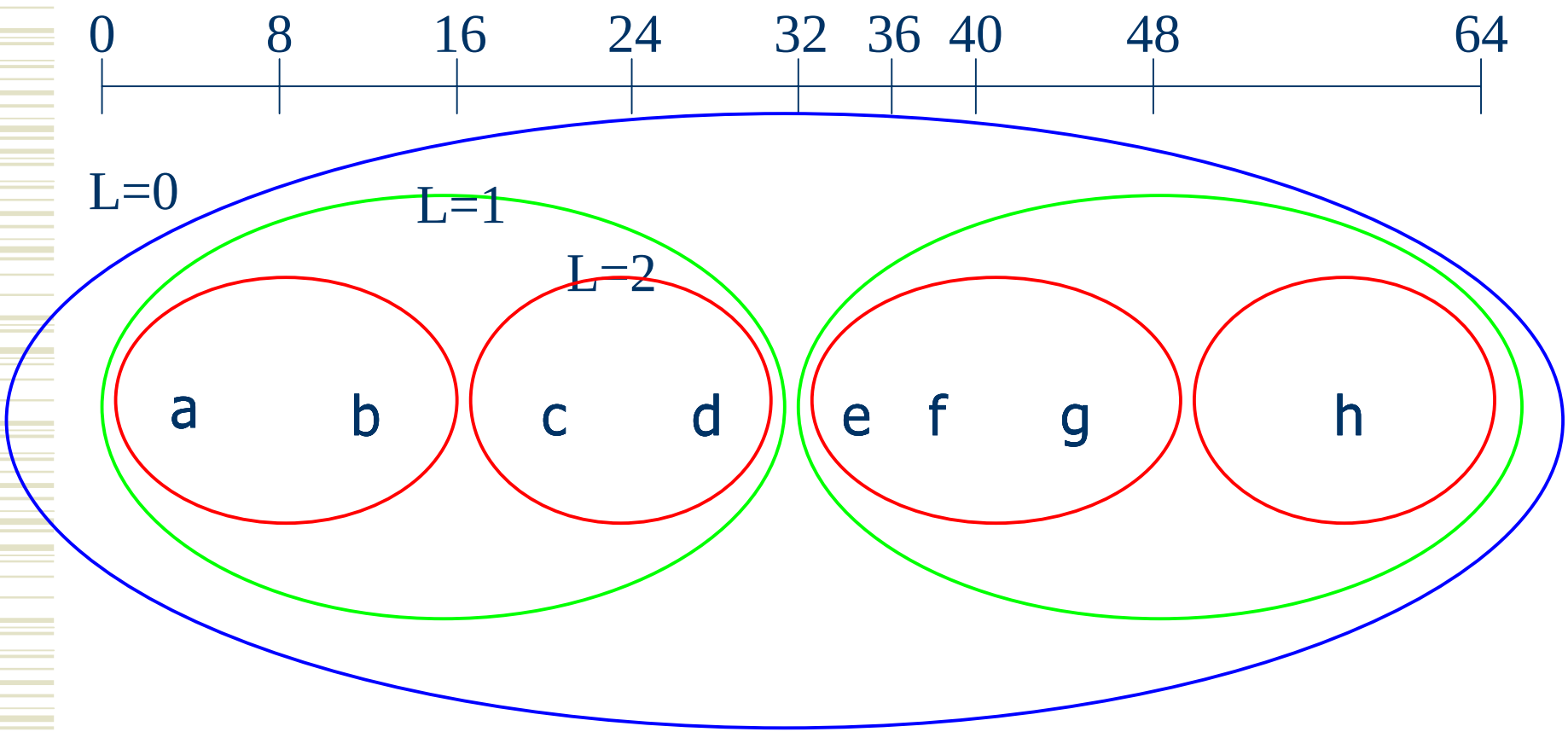
cluster (例)

- $b = 2$



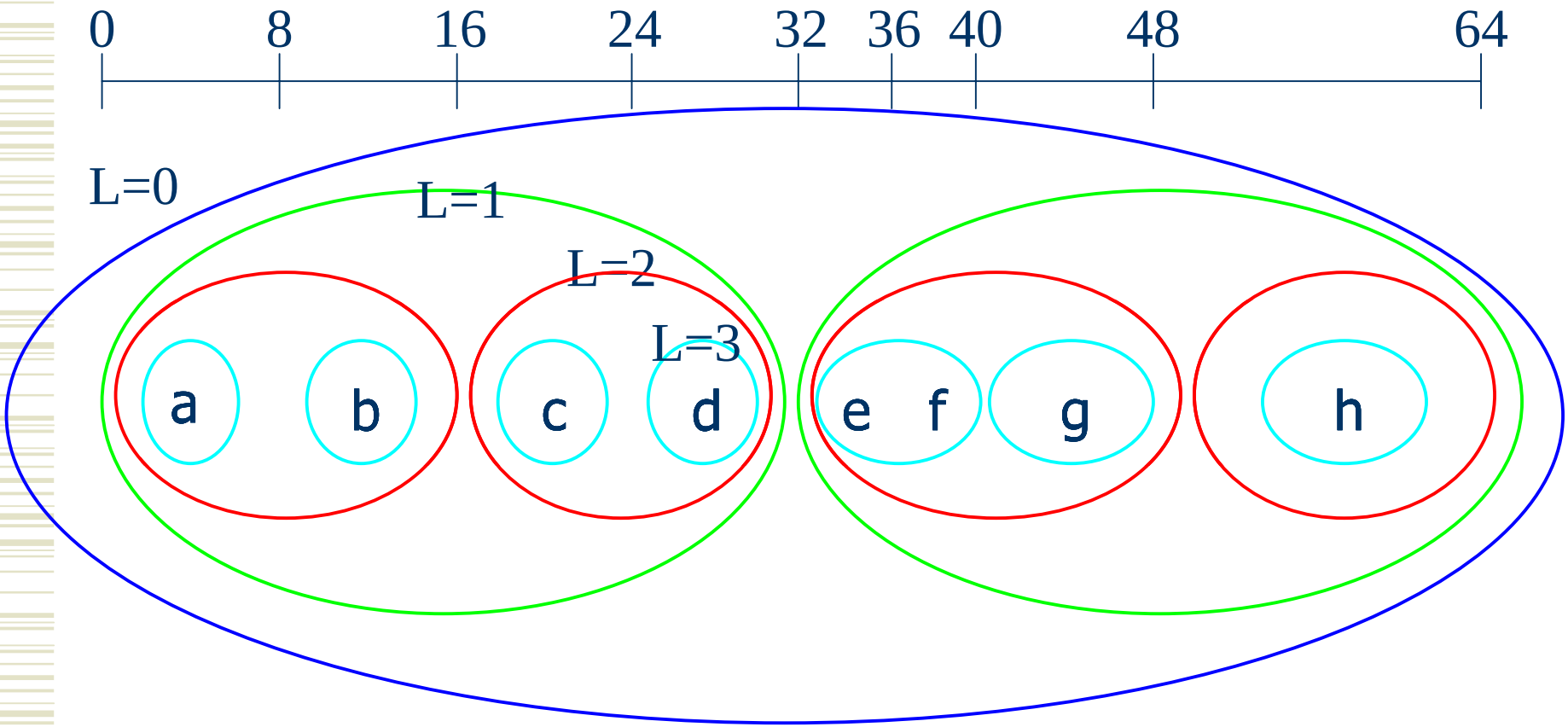
cluster (例)

- $b = 2$



cluster (例)

- $b = 2$

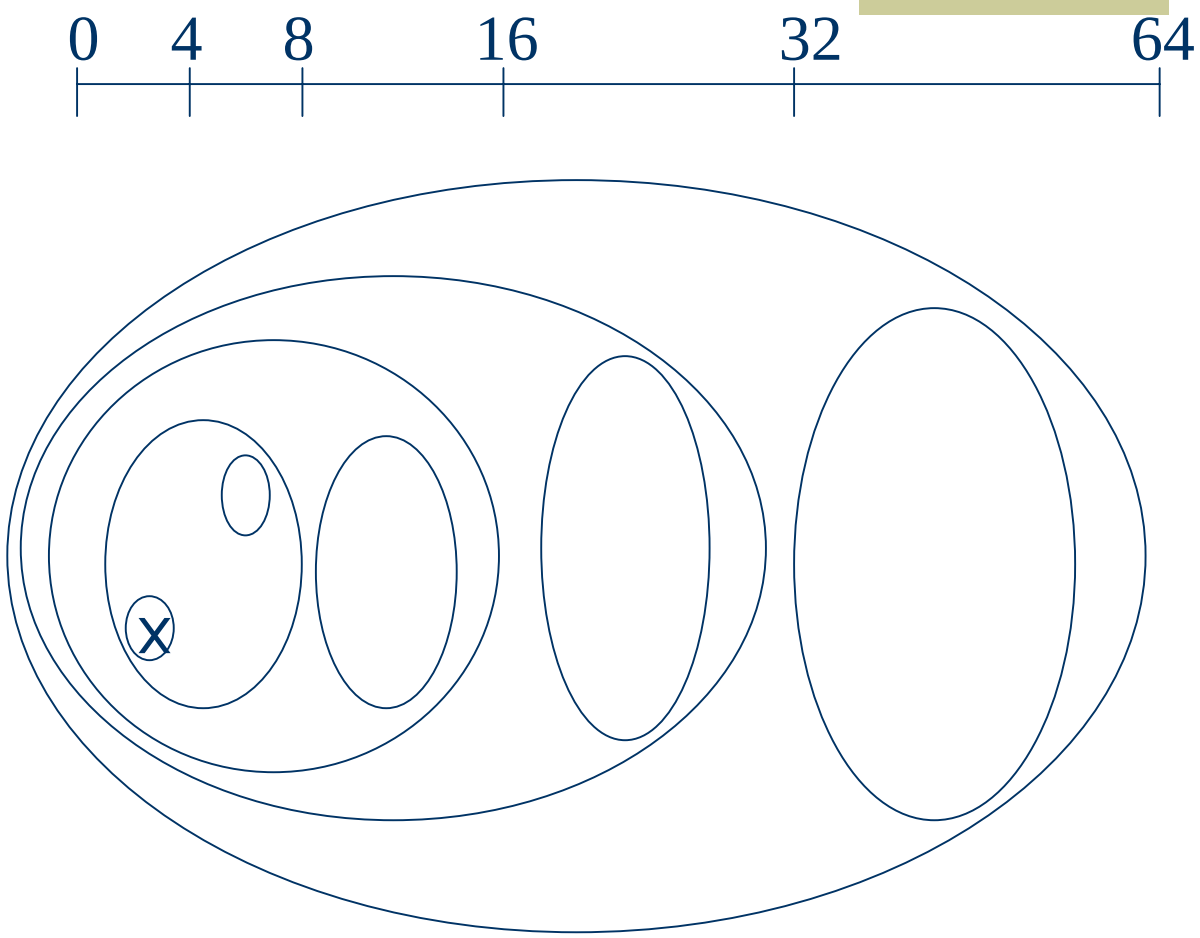


Routing table の作り方(3)
routing table のエントリ

- 自身が属するlevel i のclusterにおいて
  - b個のlevel $i+1$の subclusterの内、
    自身が属さない$b-1$個のsubclusterから
    ノードをひとつずつランダムに選ぶ
- Routing table の i 行目は選んだ$b-1$個の
  ノードの以下の情報のペアの列
  - ノードが属するlevel $i+1$の subclusterの区間
  - ノードのIP address

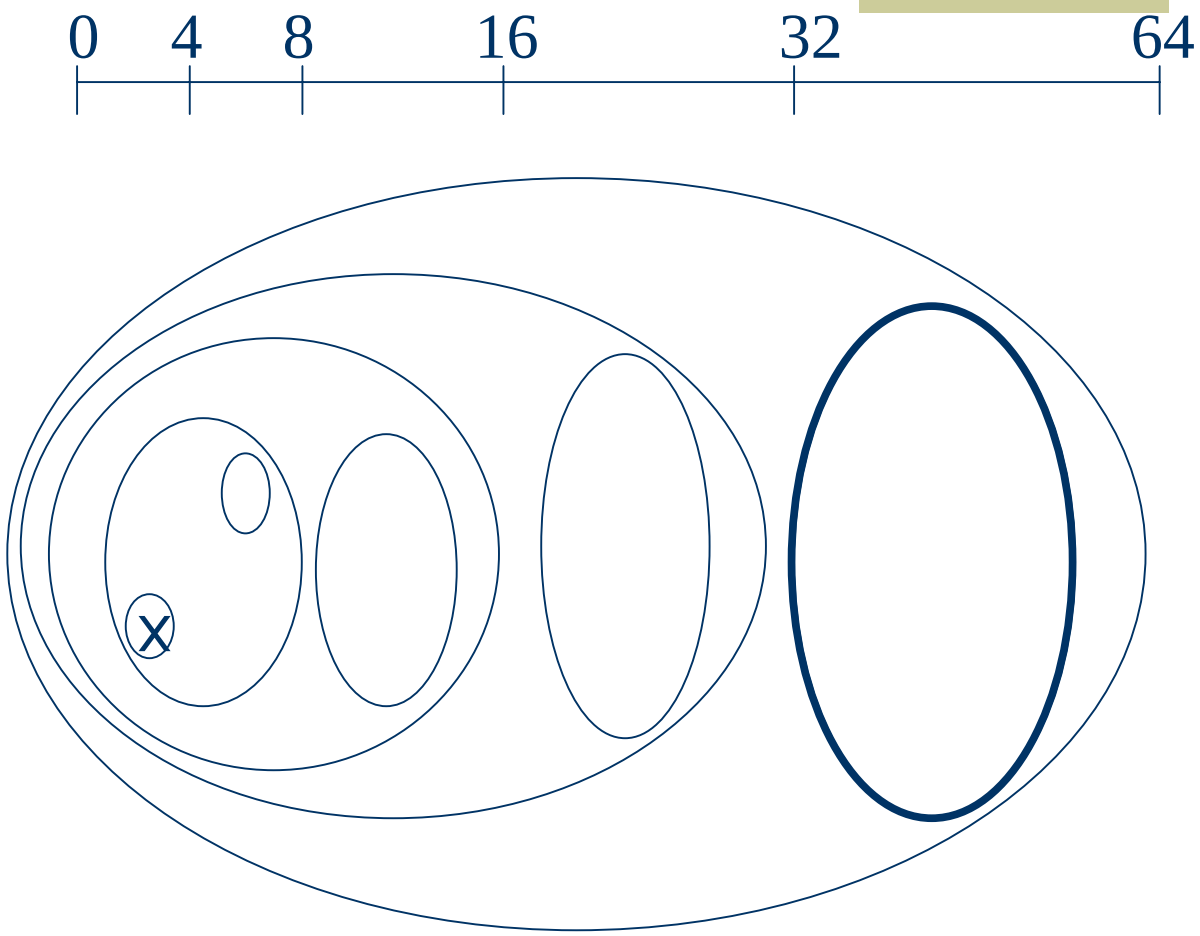
routing tableのエントリ(例1)

- $b = 2$



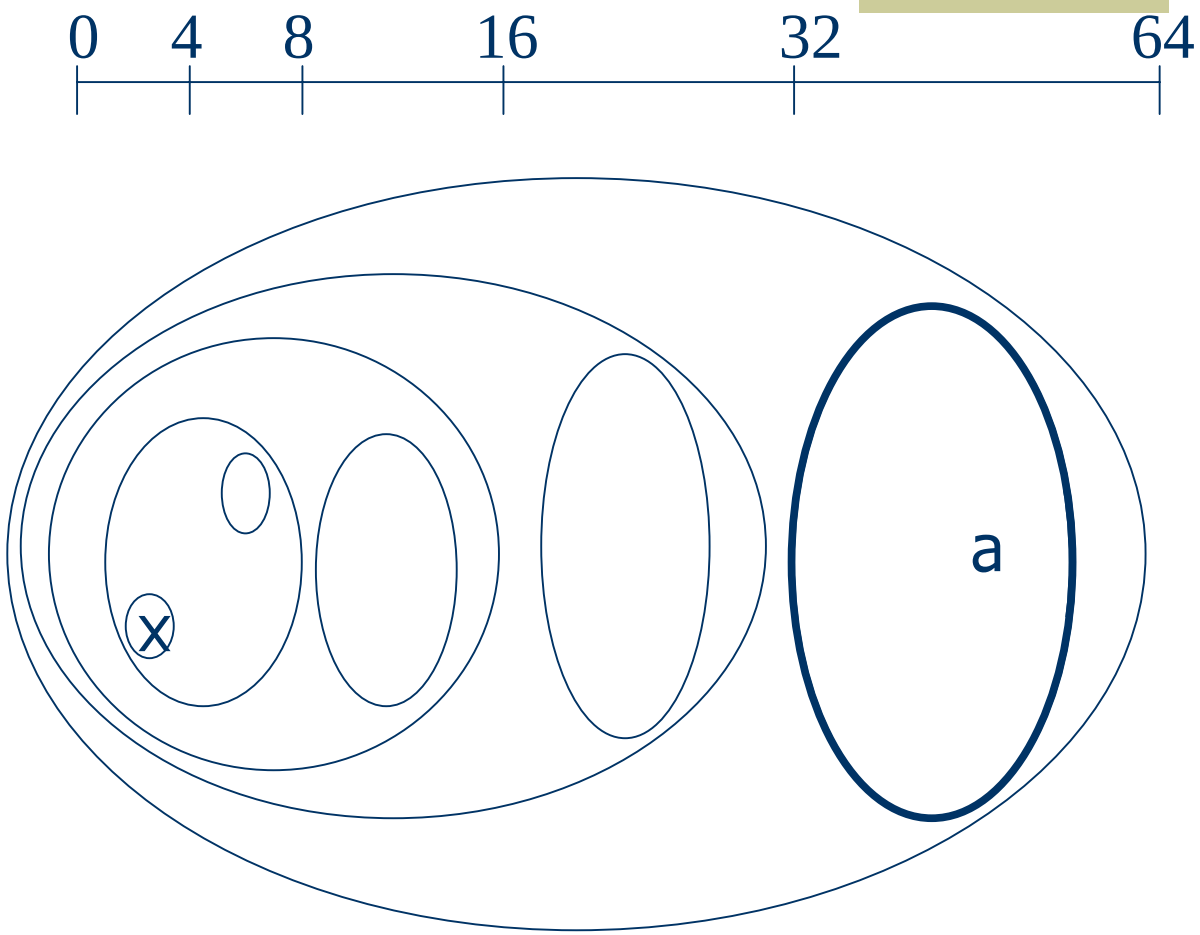
routing tableのエントリ(例1)

- $b = 2$



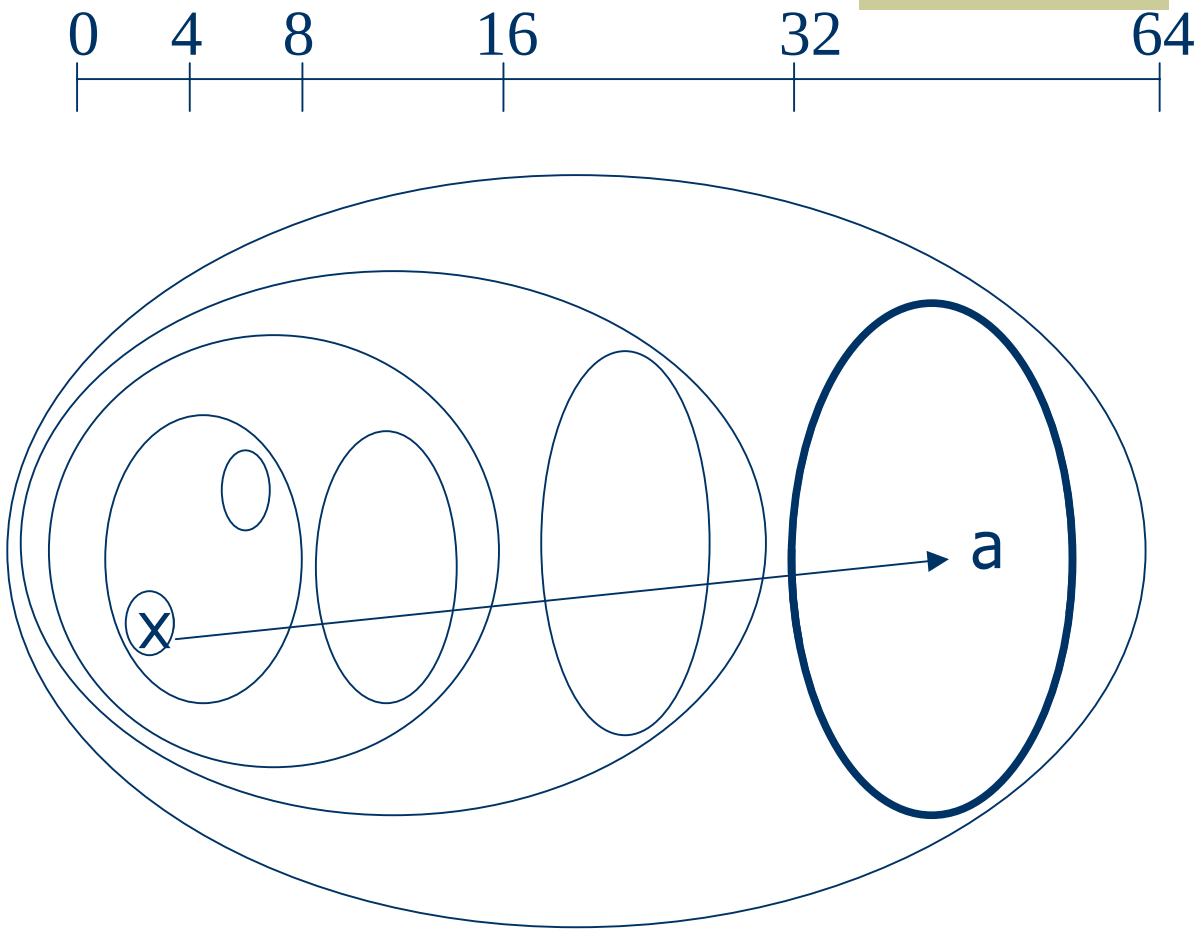
routing tableのエントリ(例1)

- $b = 2$



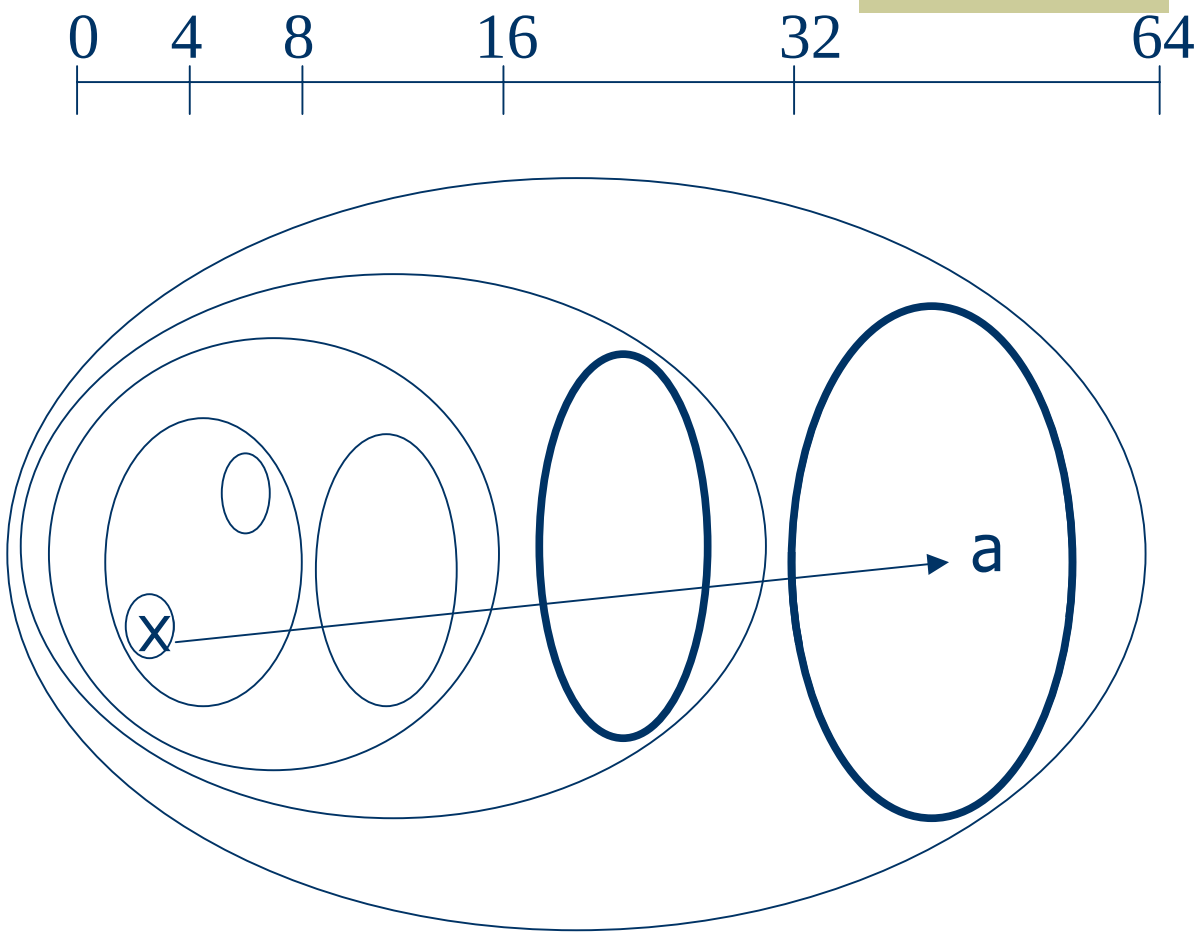
routing tableのエントリ(例1)

- $b = 2$



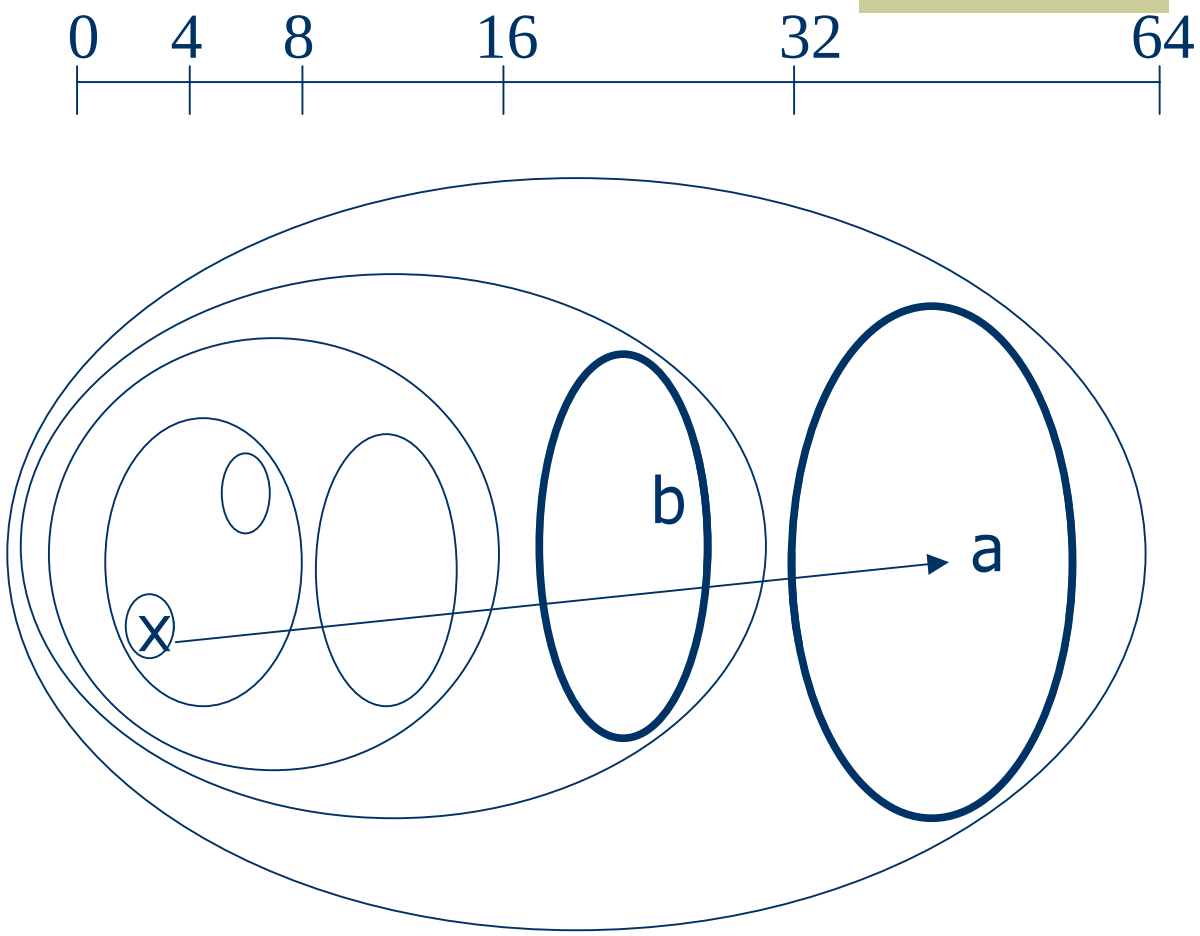
routing tableのエントリ(例1)

- $b = 2$



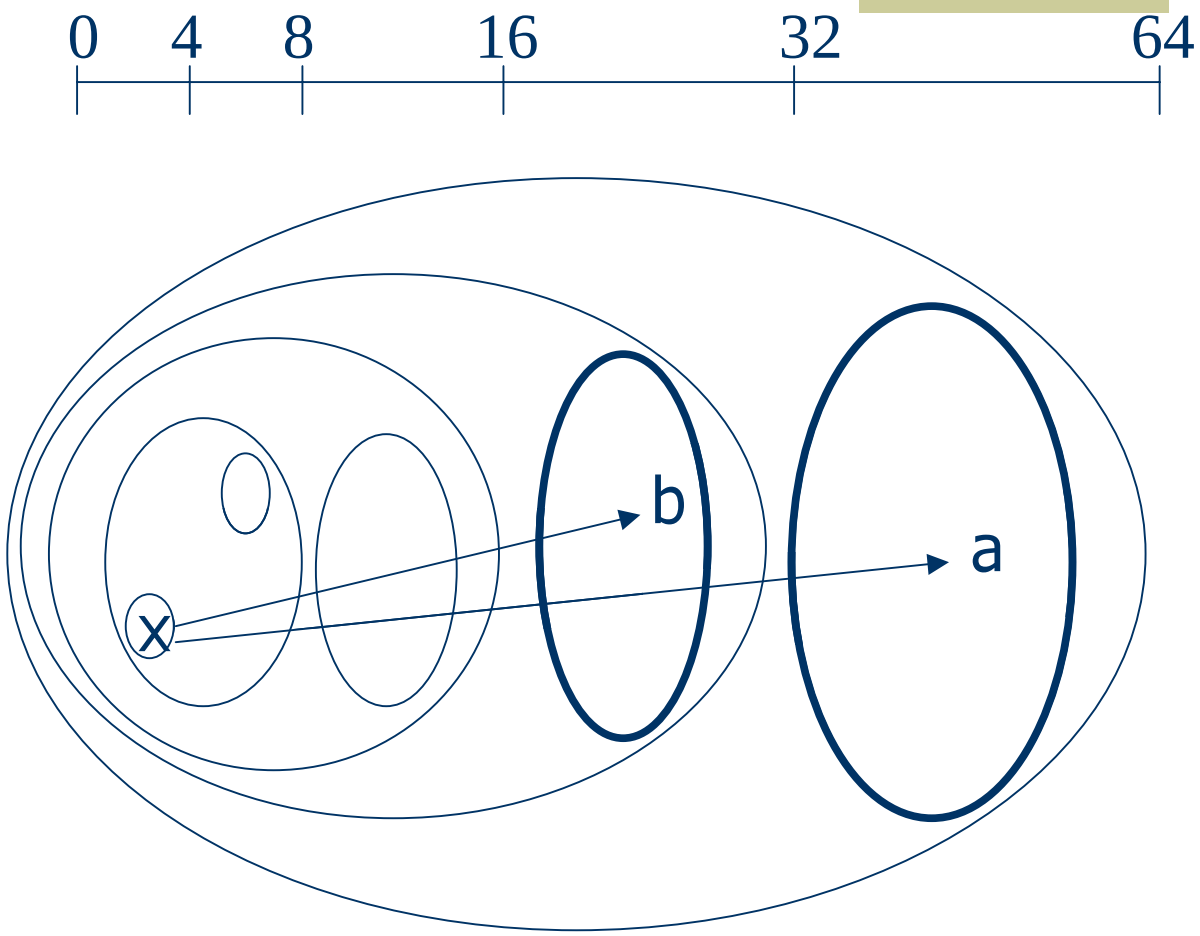
routing tableのエントリ(例1)

- $b = 2$



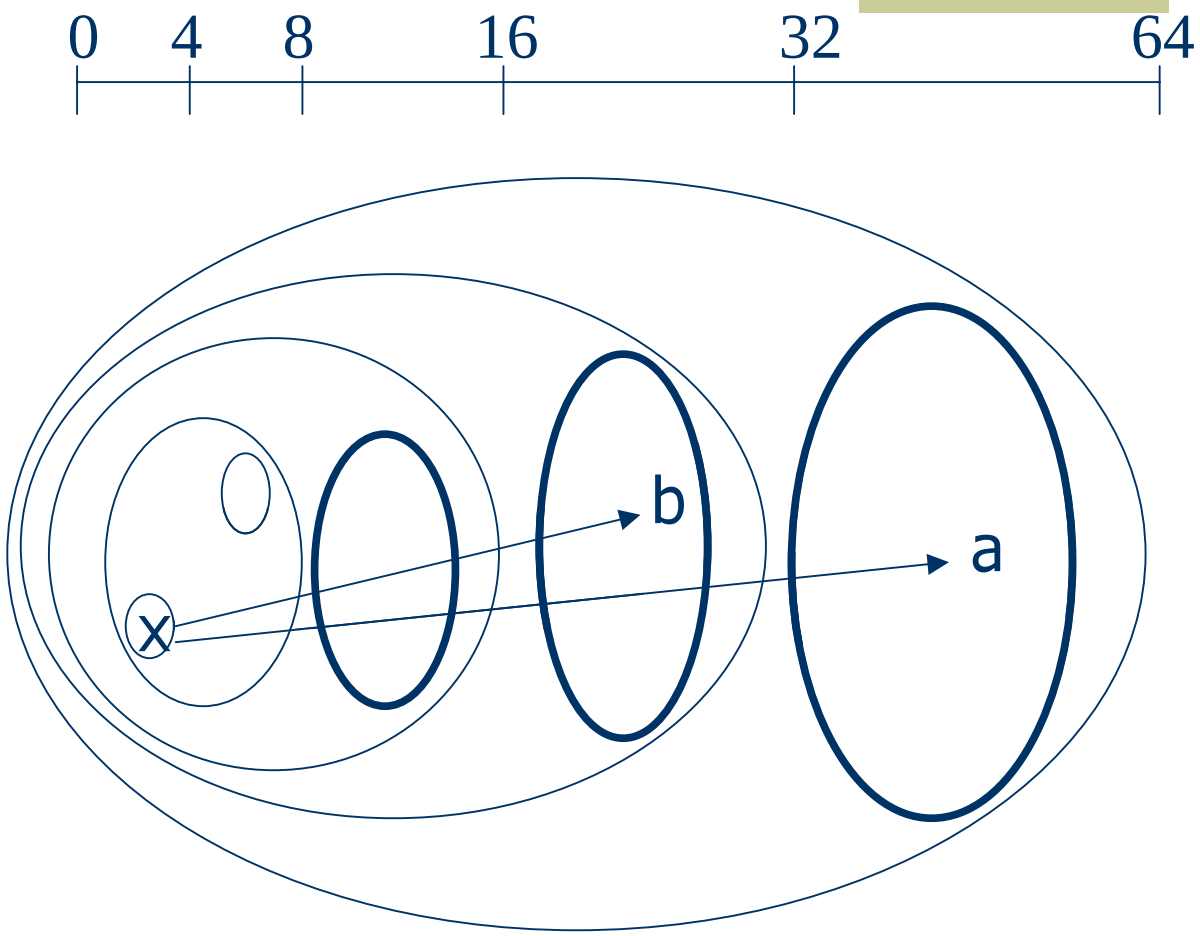
routing tableのエントリ(例1)

- $b = 2$



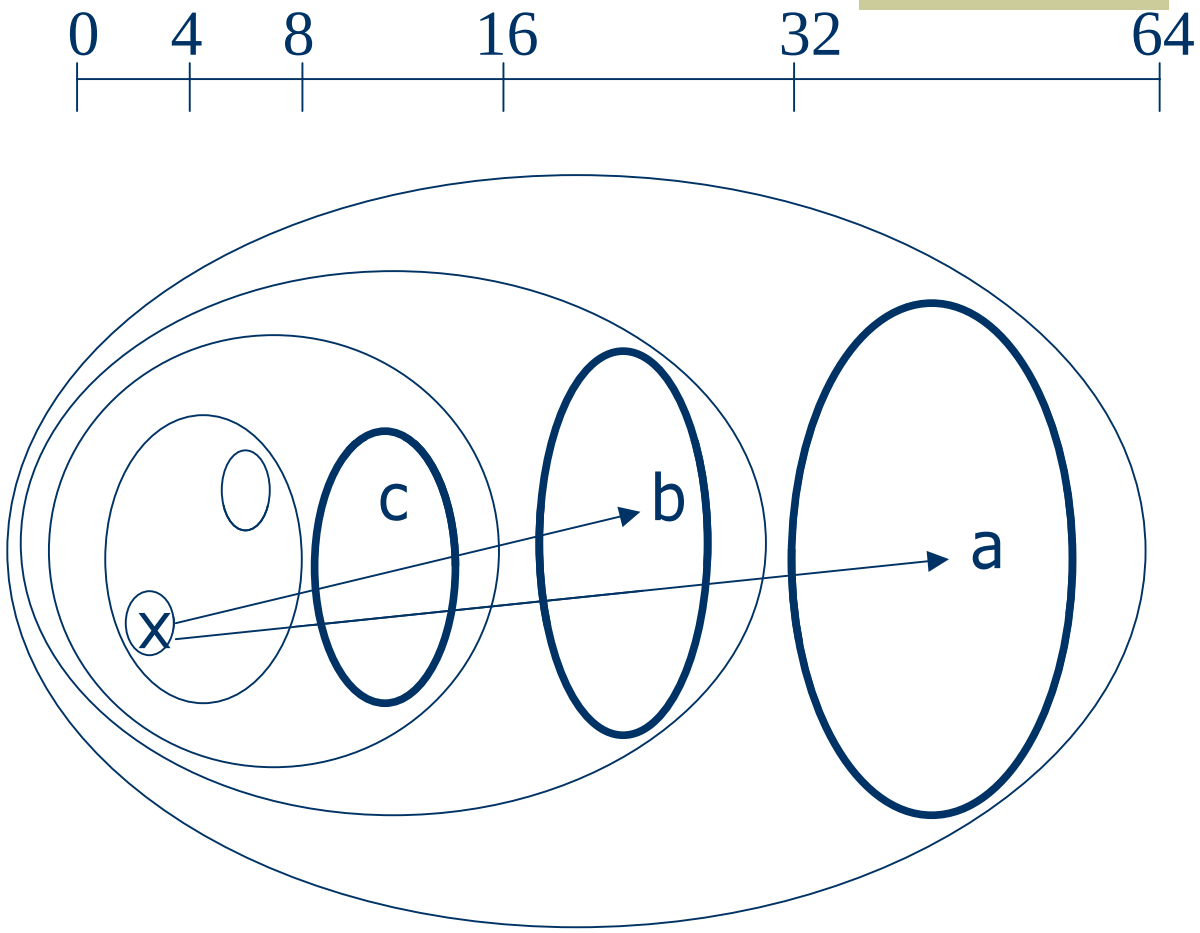
routing tableのエントリ(例1)

- $b = 2$



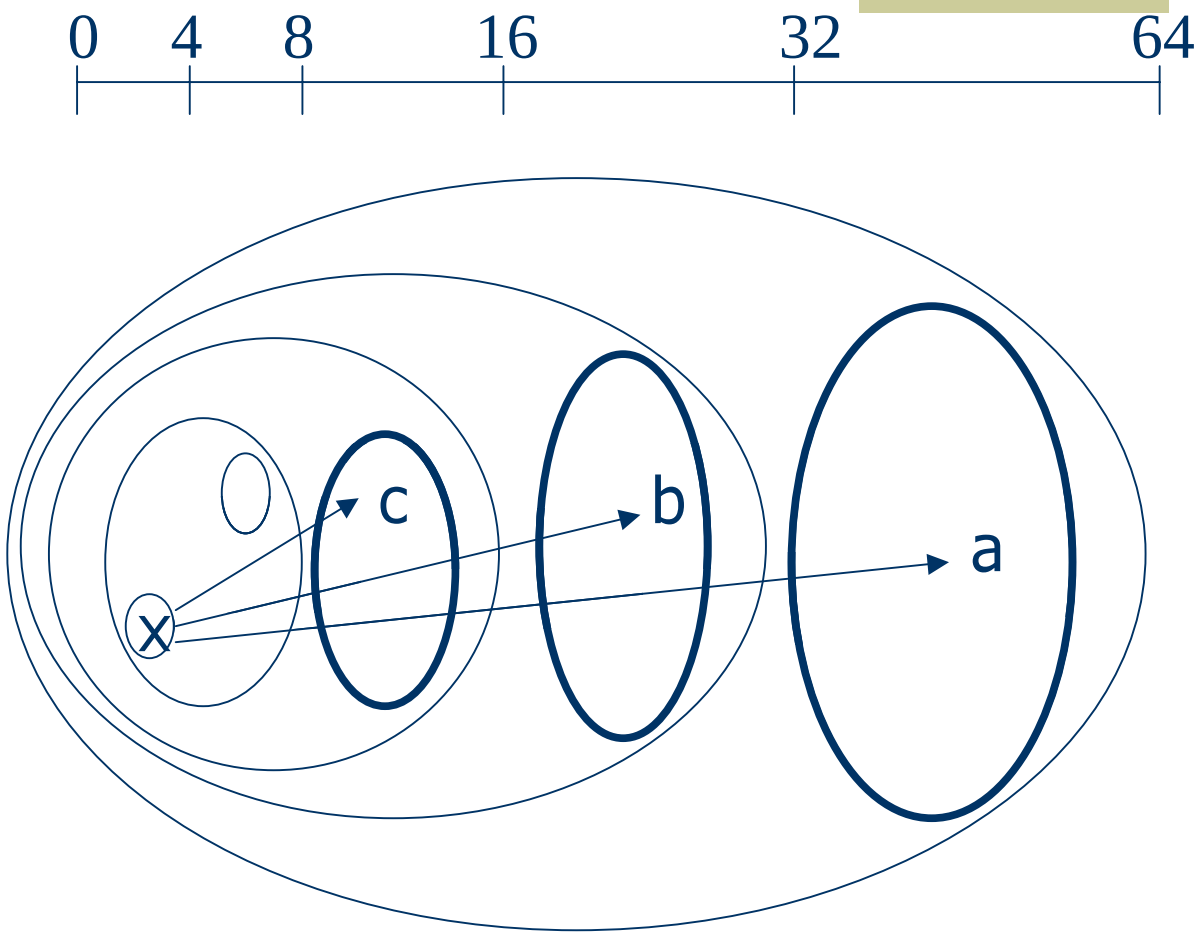
routing tableのエントリ(例1)

- $b = 2$



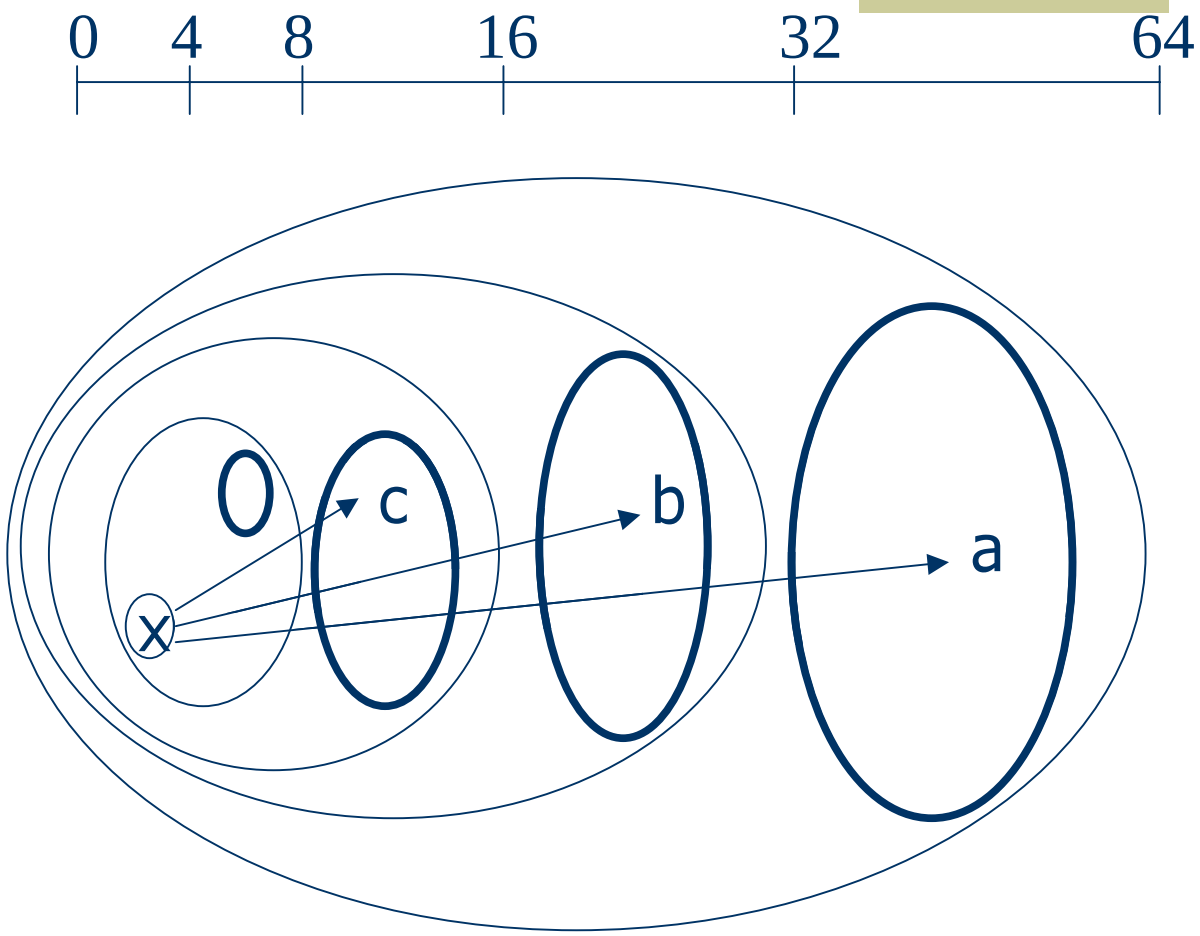
routing tableのエントリ(例1)

- $b = 2$



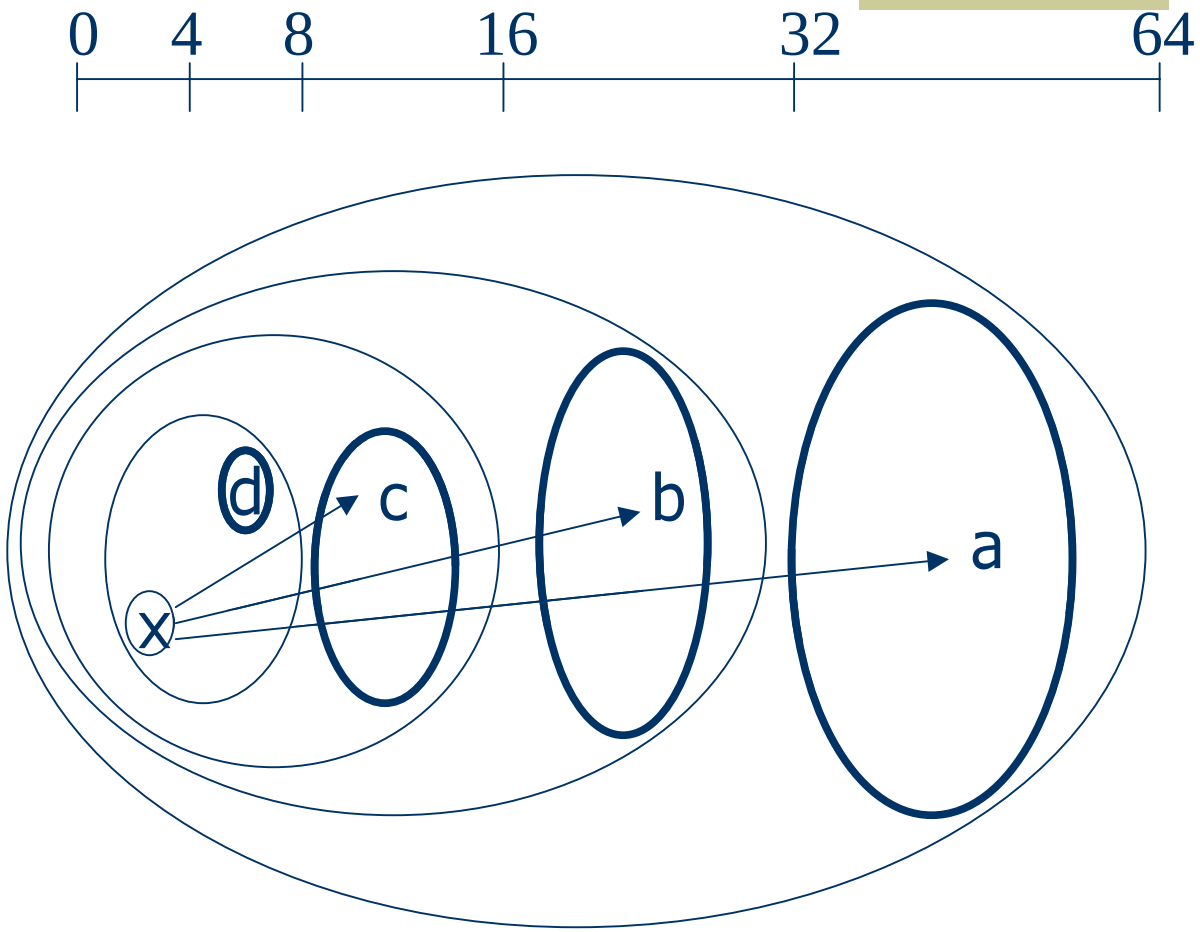
routing tableのエントリ(例1)

- $b = 2$



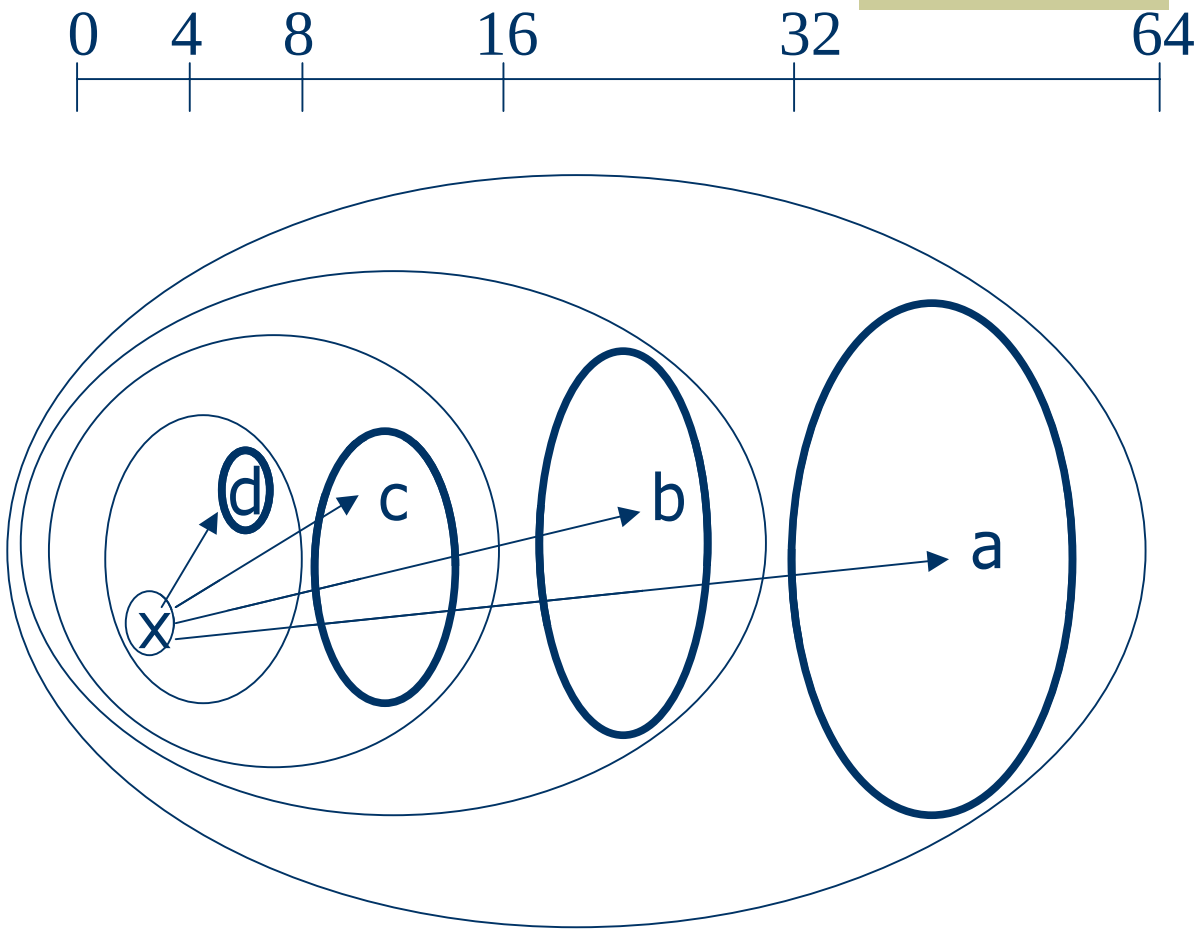
routing tableのエントリ(例1)

- $b = 2$



routing tableのエントリ(例1)

- $b = 2$

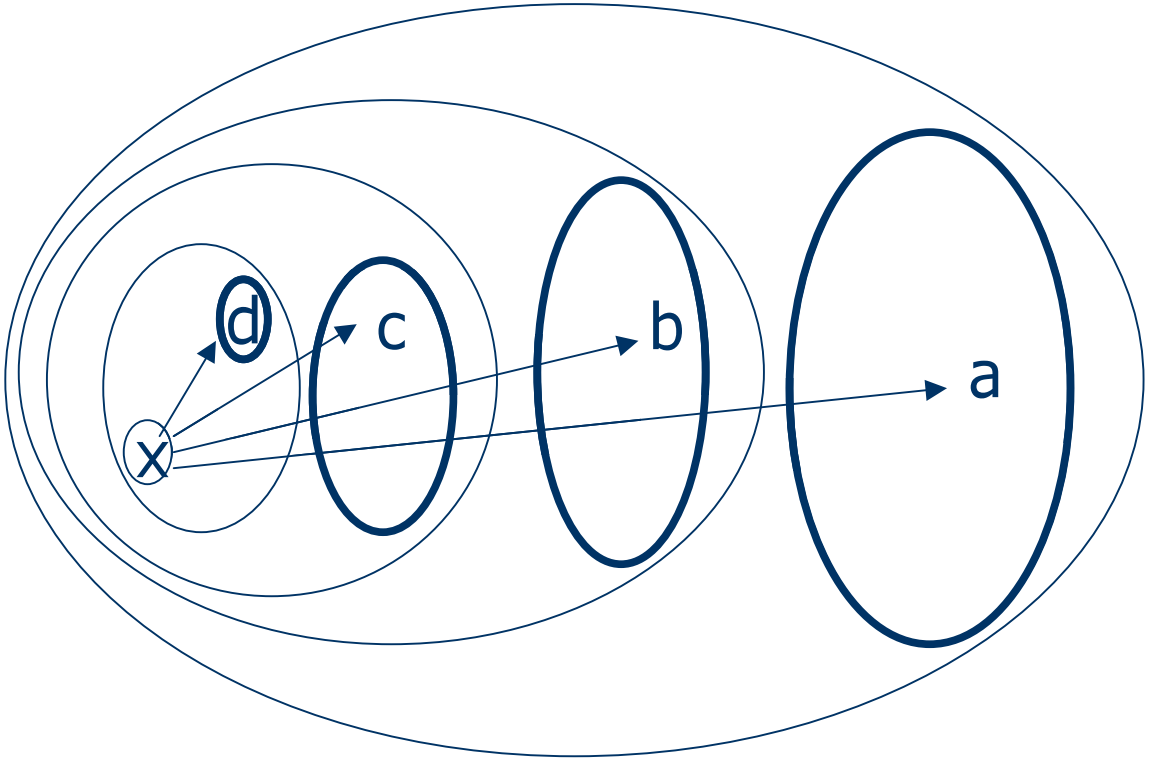


routing tableのエントリ(例1)

- $b = 2$

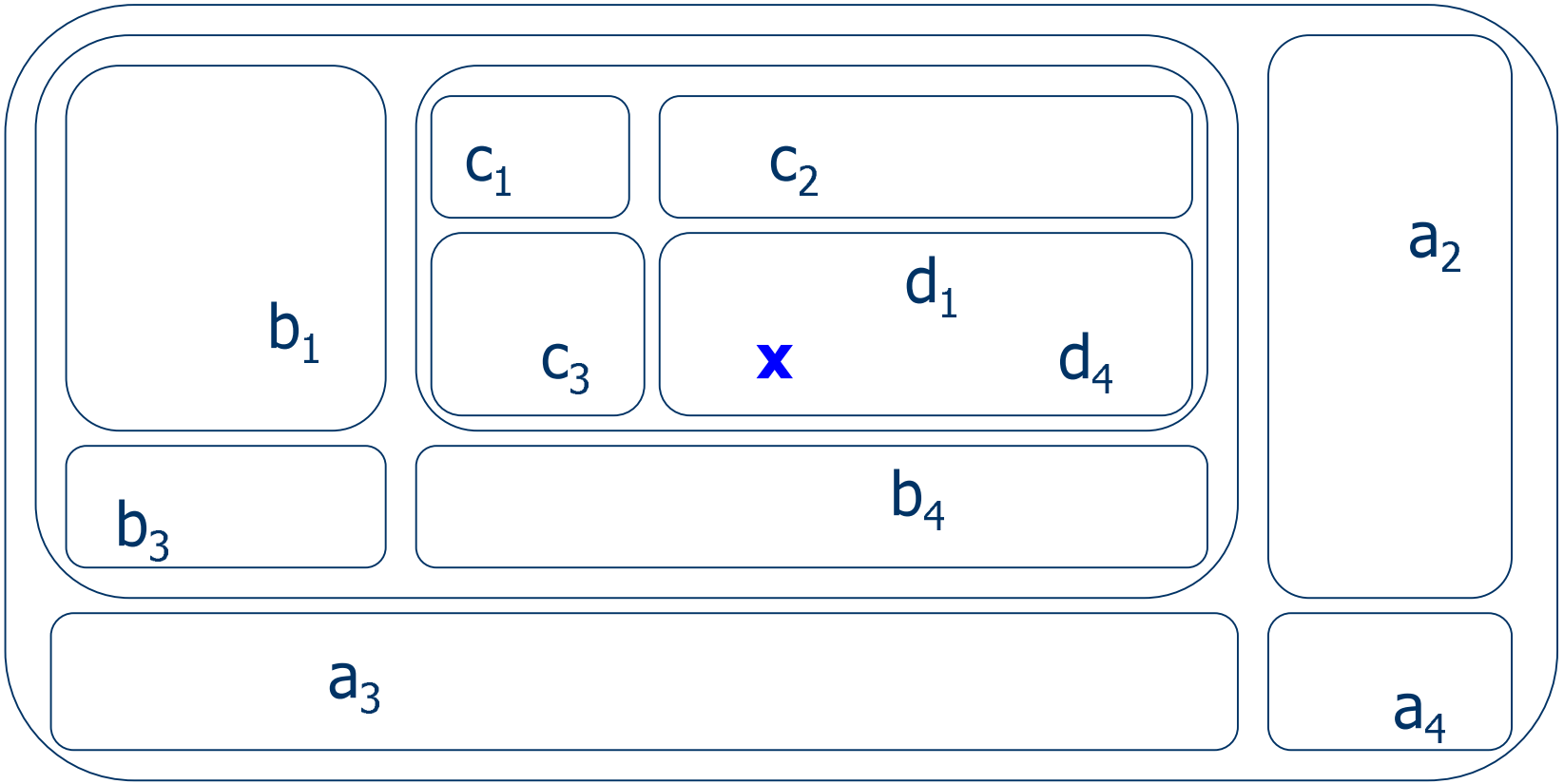
x のrouting table

[32,64) → a
[16,32) → b
[8,16) → c
[4,8) → d
[0,4) → x



routing tableのエントリ(例2)

b = 4, ノードx [30,31) の場合



routing tableのエントリ(例2)

$b = 4$, ノードx [30,31) のrouting table

	[64,128)→a_2	[128,192)→a_3	[192,256)→a_4
[0,16)→b_1		[32,48)→ b_3	[48,64)→b_4
[16,20)→c_1	[20,24)→c_2	[24,48)→c_3	
[28,29)→d_1	[29,30)→ d_1	**[30,31)→x**	[31,32)→ d_4

Routingの特徴

- Level i のroutingが行われると次のノードでは level i +1以上のroutingが行われる

- ホップ数: 平均 $\log N / \log b$
- Routing table の大きさ: 平均 $(b - 1) \log N / \log b$
- N はノード数

- $b = 16$, $N = 2^{20} = 1{,}048{,}576$ の時
  - ホップ数: 平均 5, Routing tableの大きさ: 平均 75

ノードの参加(1)

- IP address p のノードが参加する手順

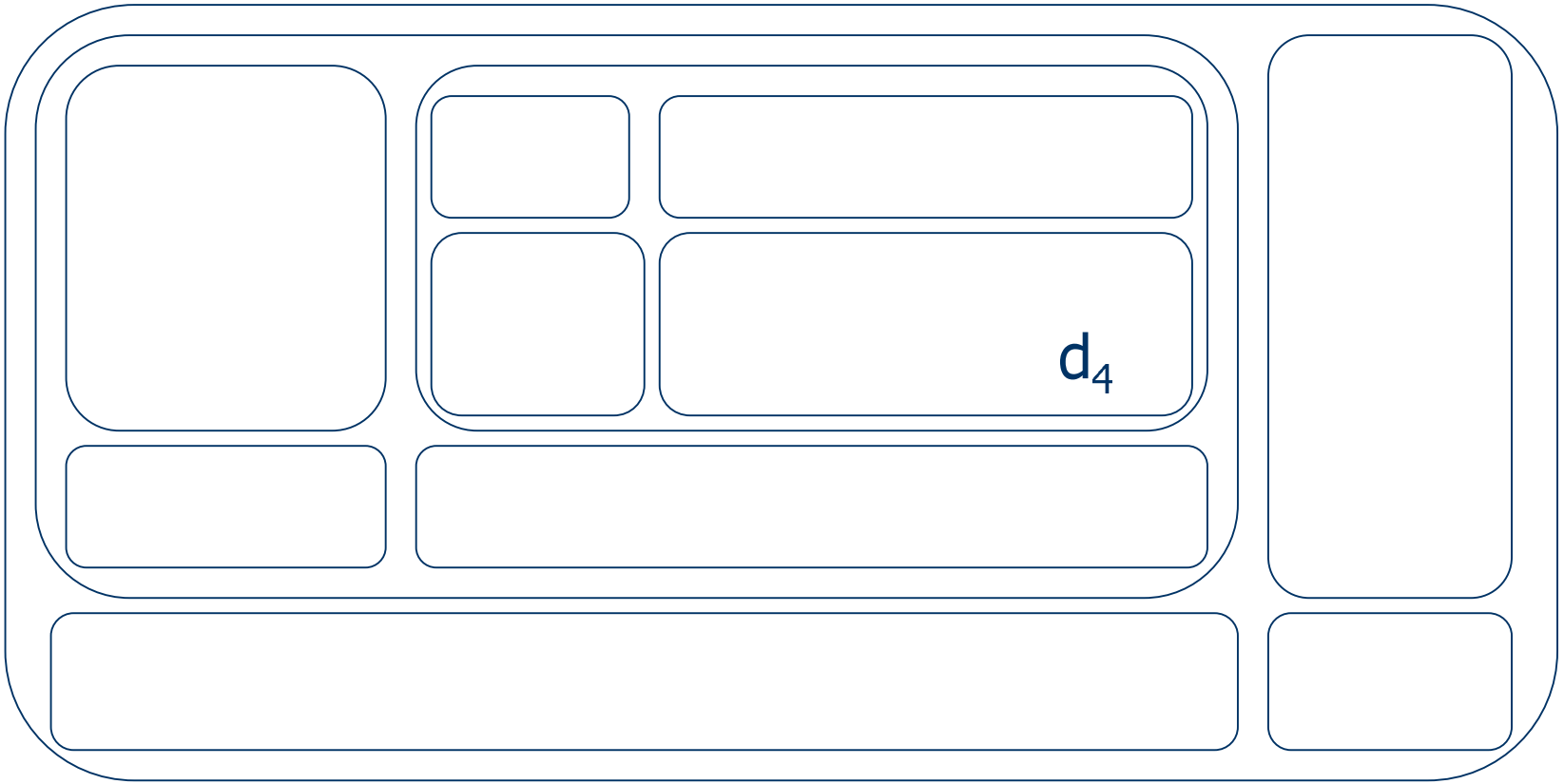
1. 乱数kをkeyとするJoin(k,p)メッセージを参加済みノードのひとつに送る
2. k を含む区間を持つノードから区間の半分を与えられる
3. Routing tableを作るためのメッセージNewNode(k,0,p)を区間をくれたノードに送る

ノードの参加(2)
NewNodeメッセージ

- NewNode (k,i,p) をroutingするノードは
  1. Level i + 1のsubclusterの区間の内、k を含まない区間からそれぞれひとつランダムに整数 l を選び, lをkeyとするメッセージNewNeighbor(l,p)をroutingする
  2. Level i + 1のsubclusterの区間の内、k を含む区間からひとつランダムに整数 m を選びメッセージ
     NewNode (m,i+1,p) をroutingする
- NewNeighbor (l,p)を受け取ったノードは
  1. 参加ノードp に自身のIP addressを知らせる
- 参加ノードはNewNeighborへの返事をもとにrouting tableを構築する

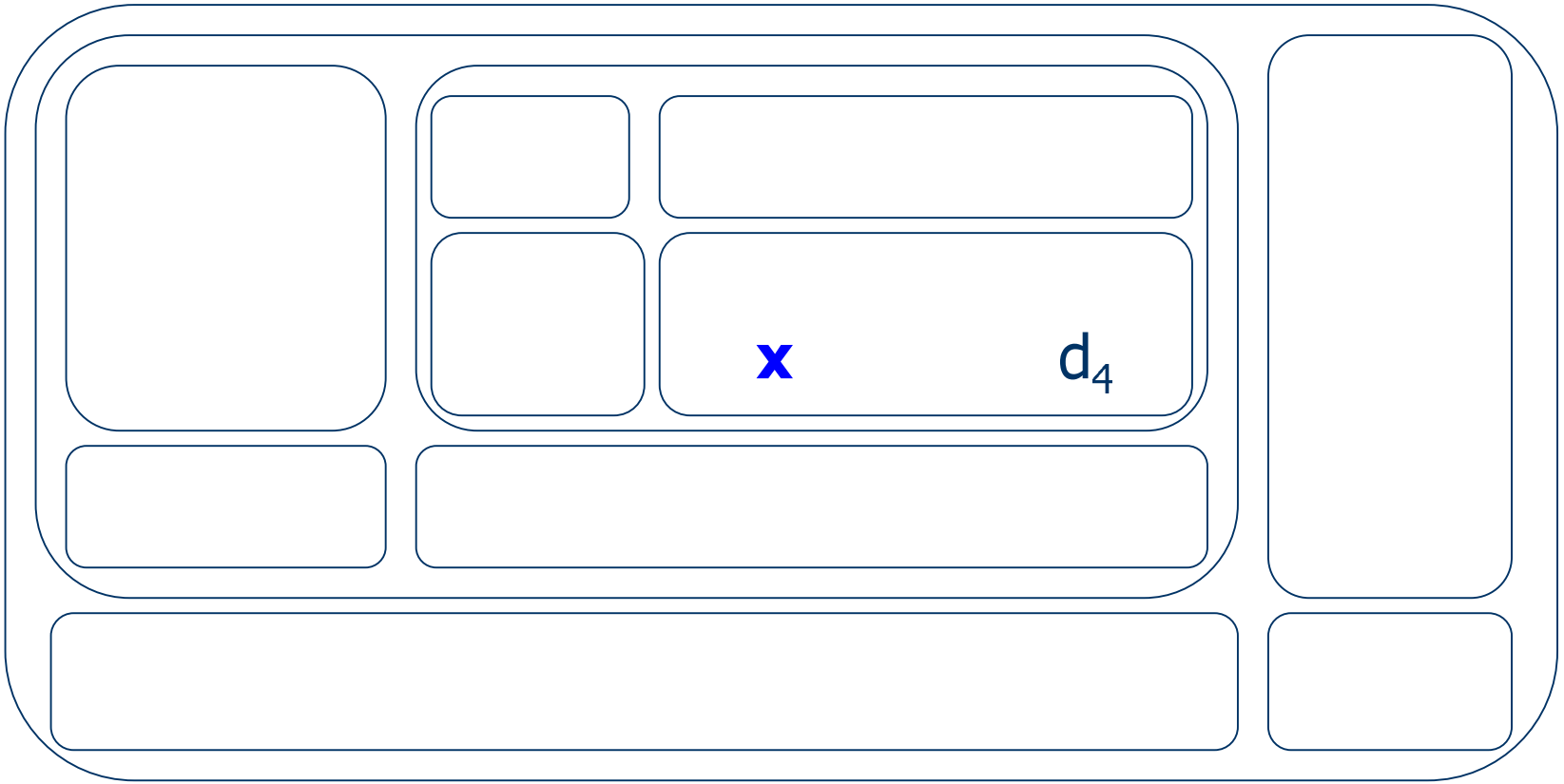
ノードの参加(例)

ノードxがd_4にJoin (k,p)を送る



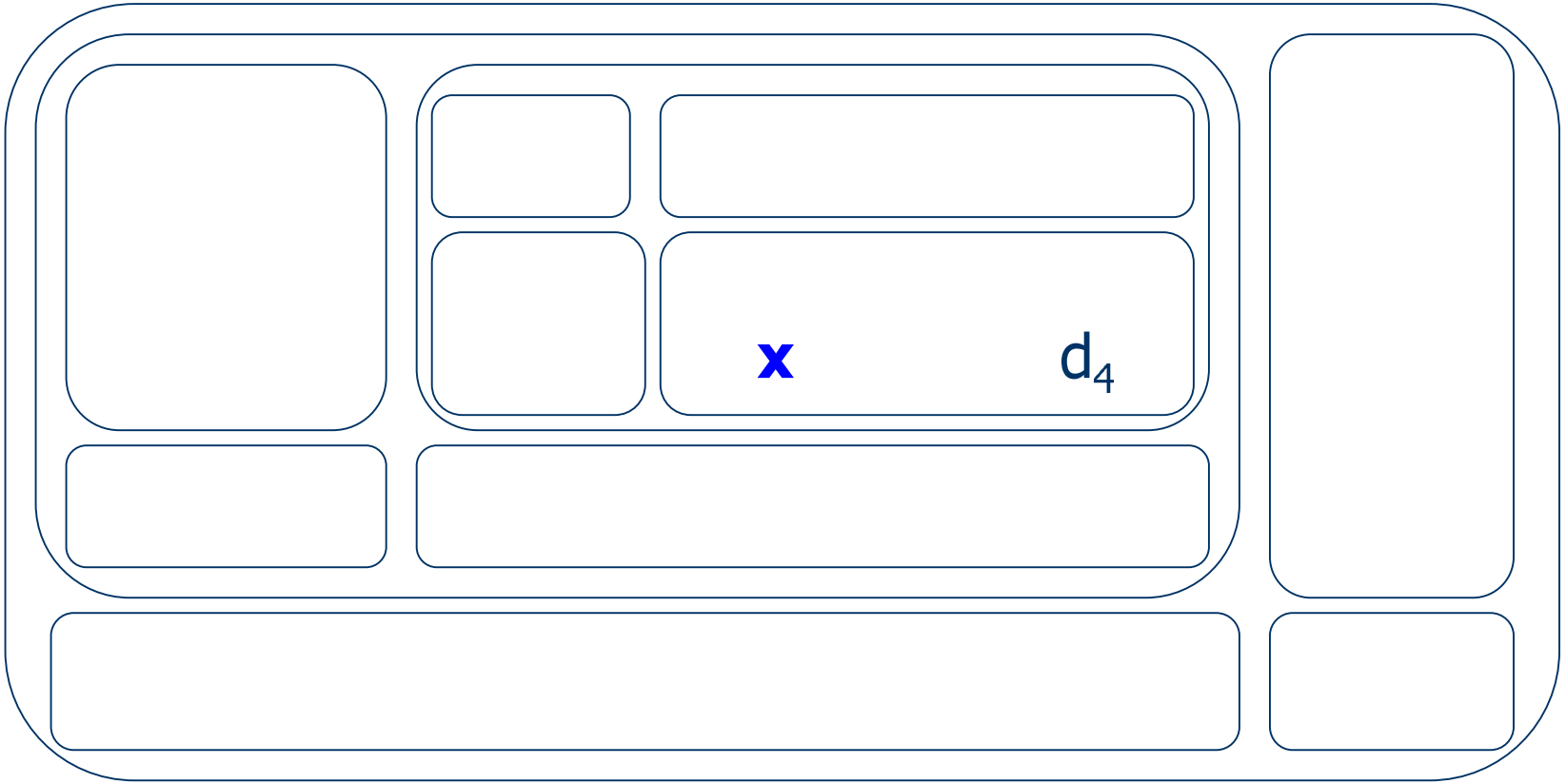
ノードの参加(例)

ノードxがd_4にJoin (k,p)を送る



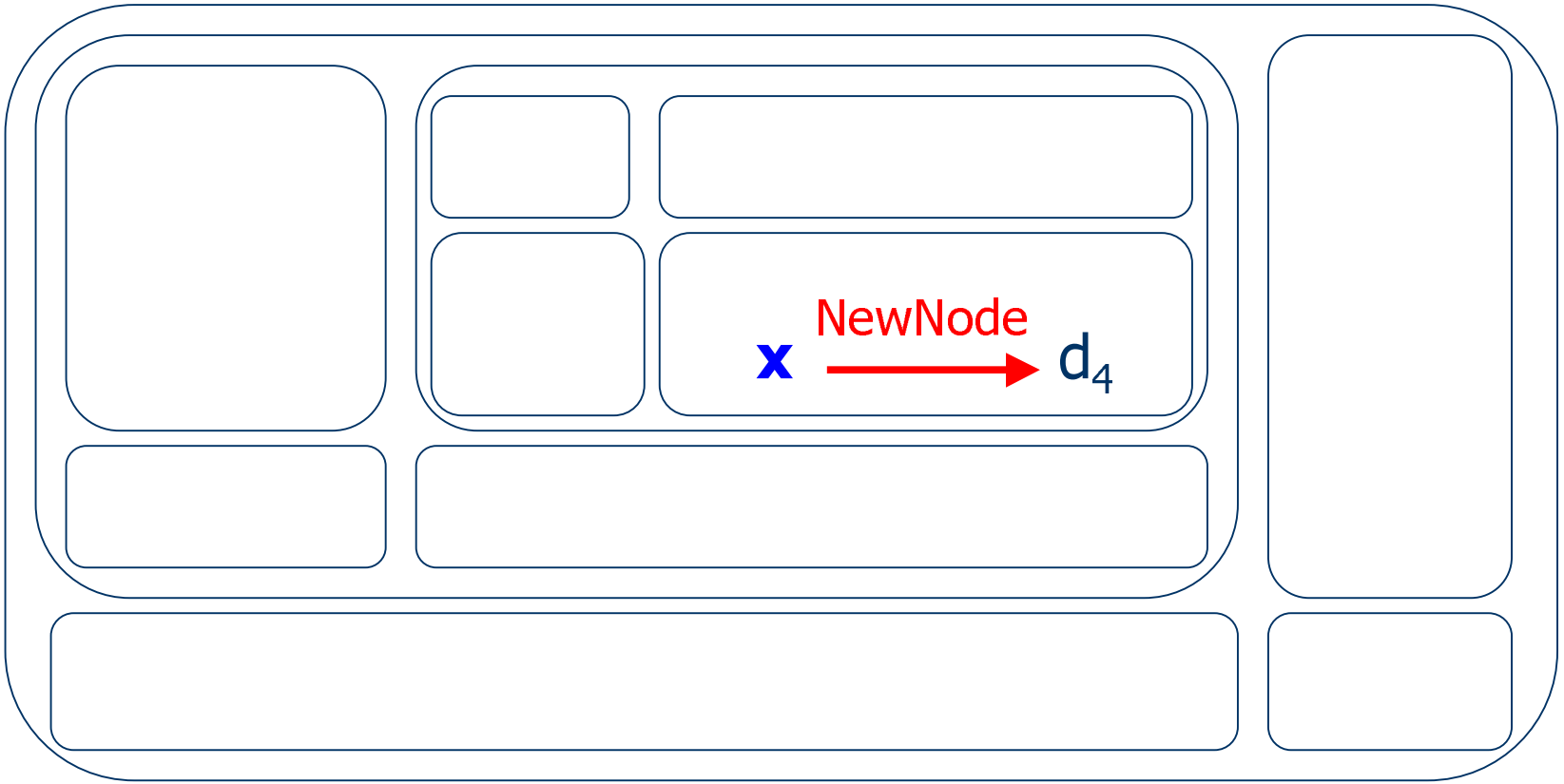
ノードの参加(例)

ノードxがd_4にNewNode(k,0,p)を送る



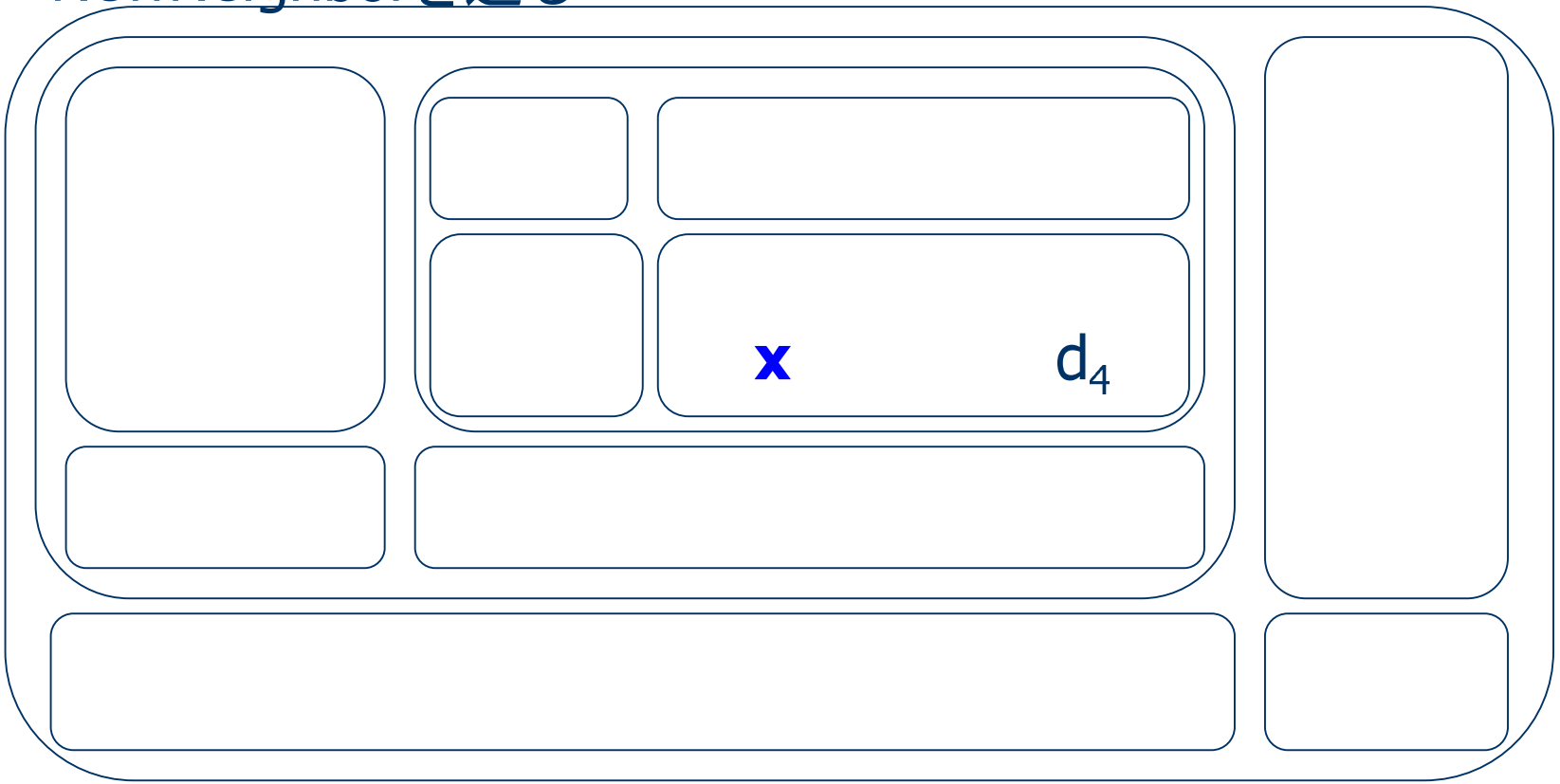
ノードの参加(例)

ノードxがd_4にNewNode(k,0,p)を送る



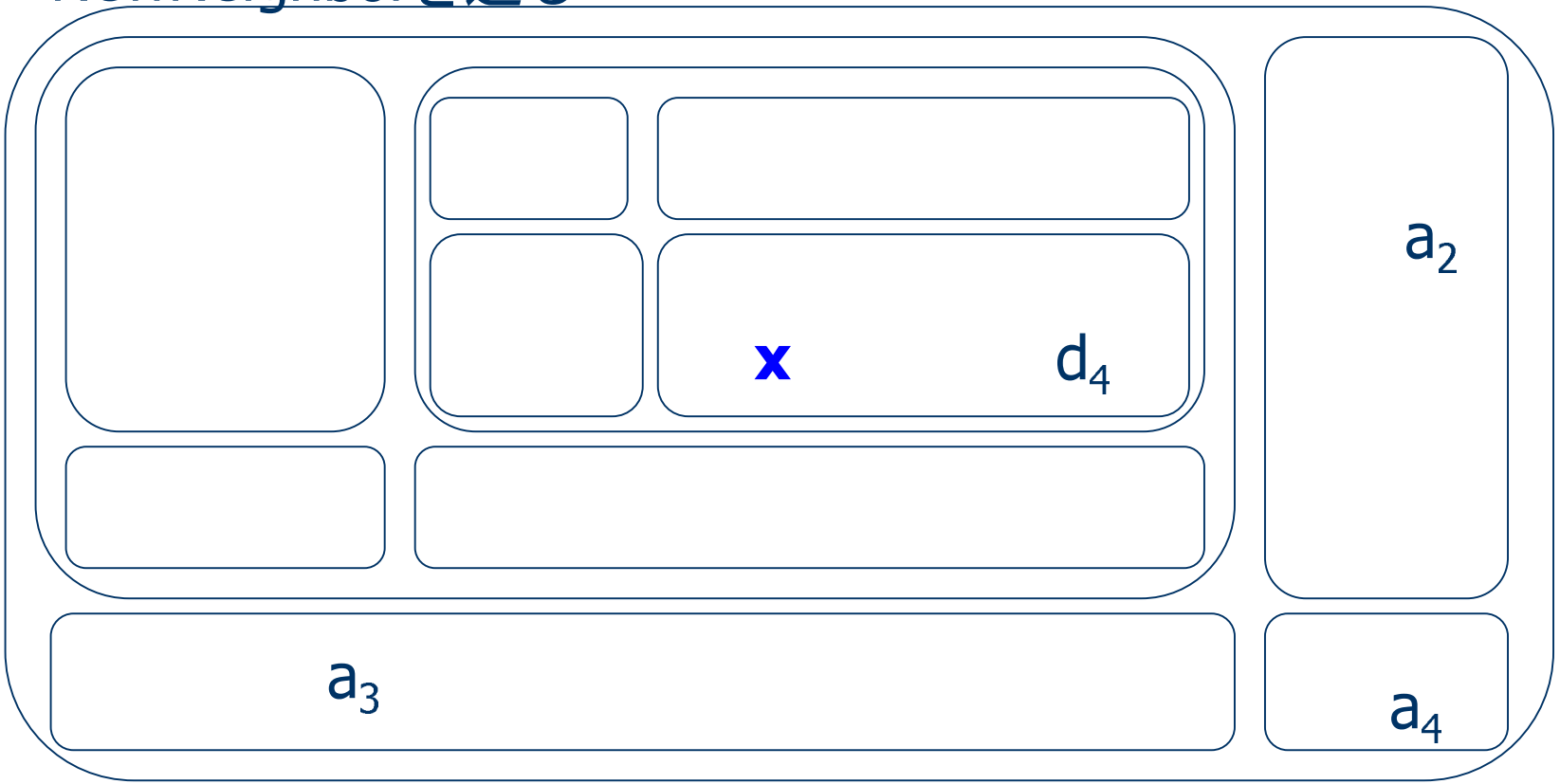
ノードの参加(例)

d_4はkが属さないLevel 1のsubcluserからノードを選びNewNeighborを送る



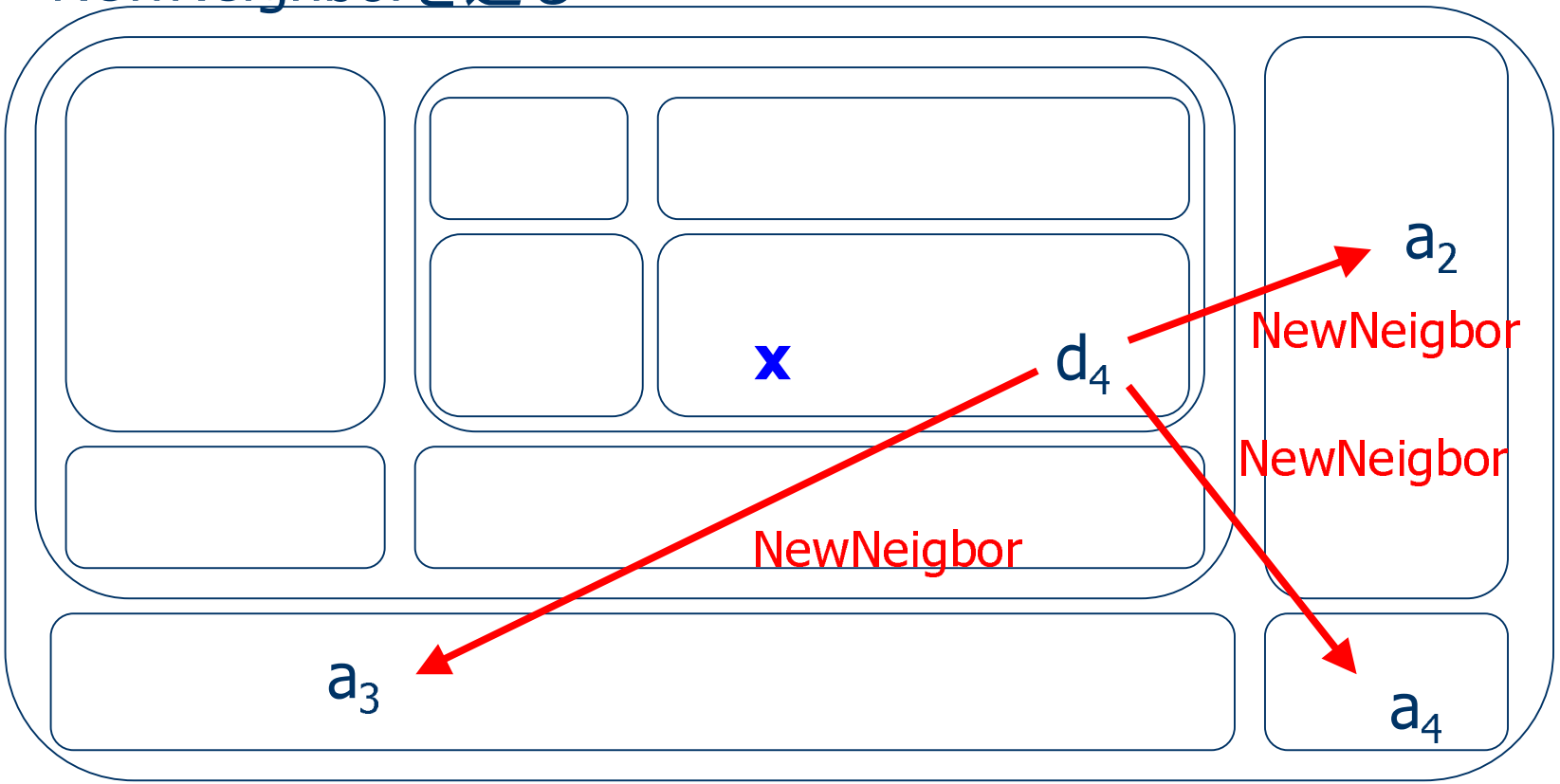
ノードの参加(例)

d_4は$\boldsymbol{k}$が属さないLevel 1のsubcluserからノードを選び
NewNeighborを送る



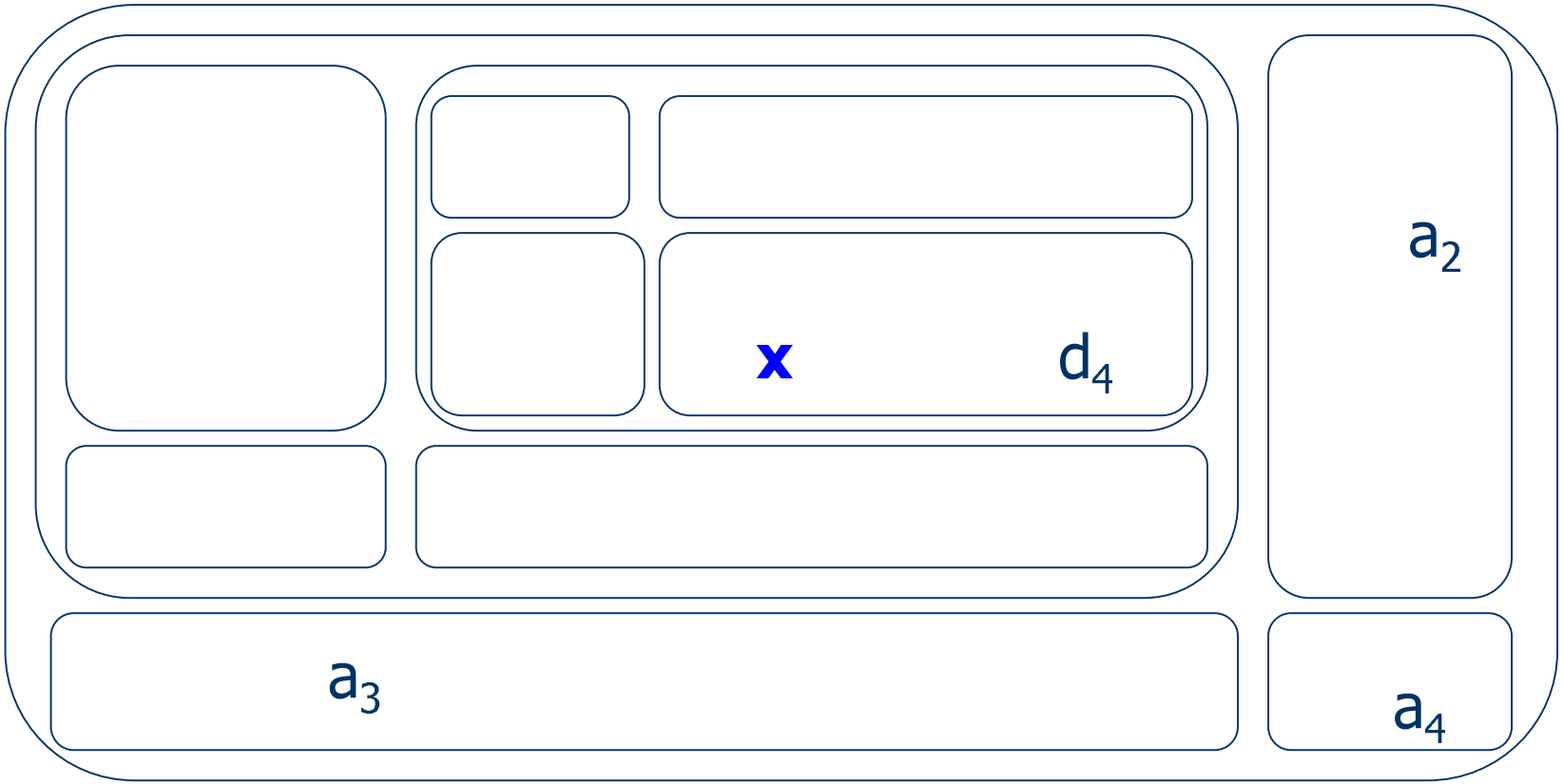
ノードの参加(例)

d_4はkが属さないLevel 1のsubcluserからノードを選び
NewNeighborを送る



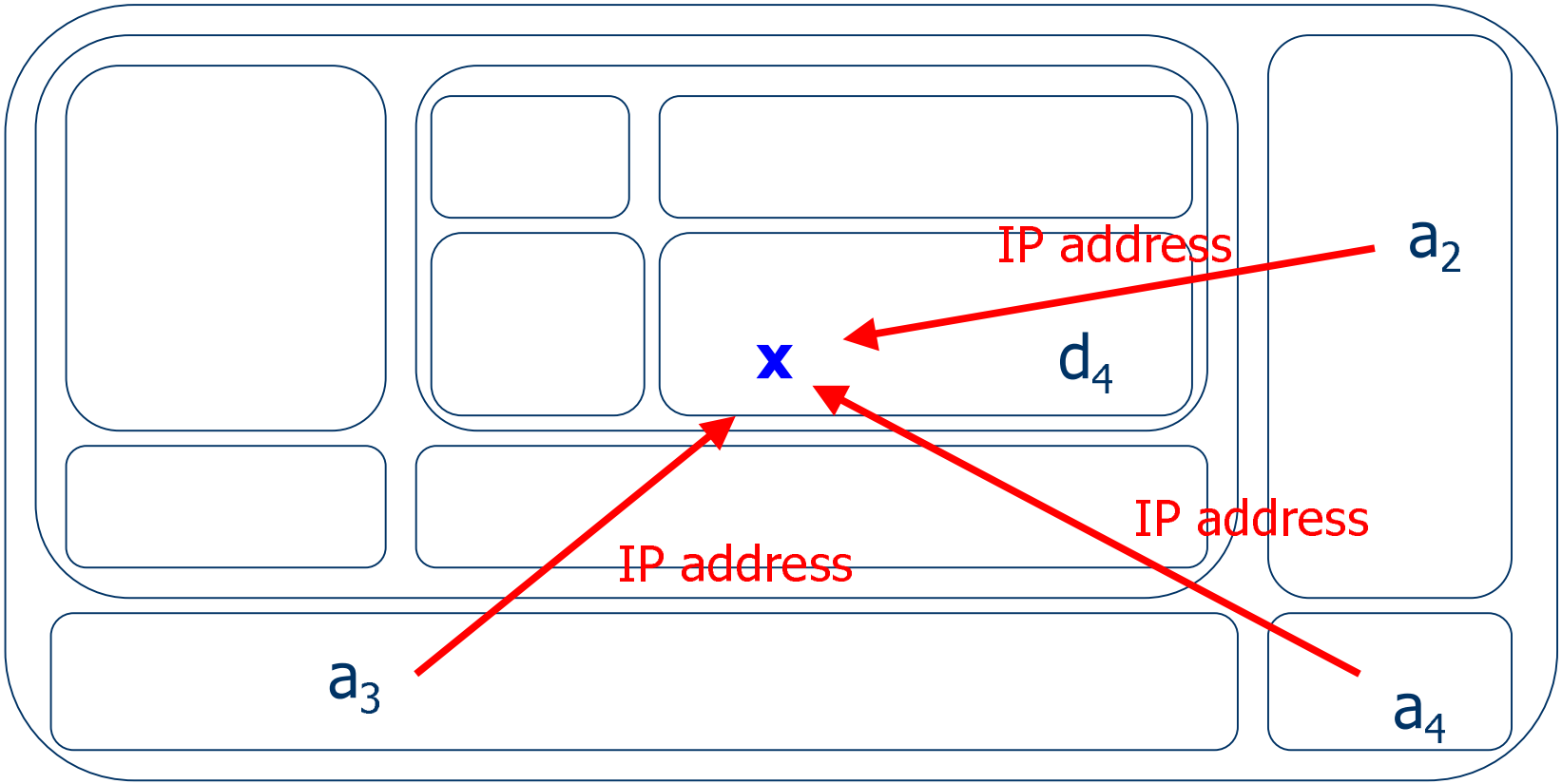
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



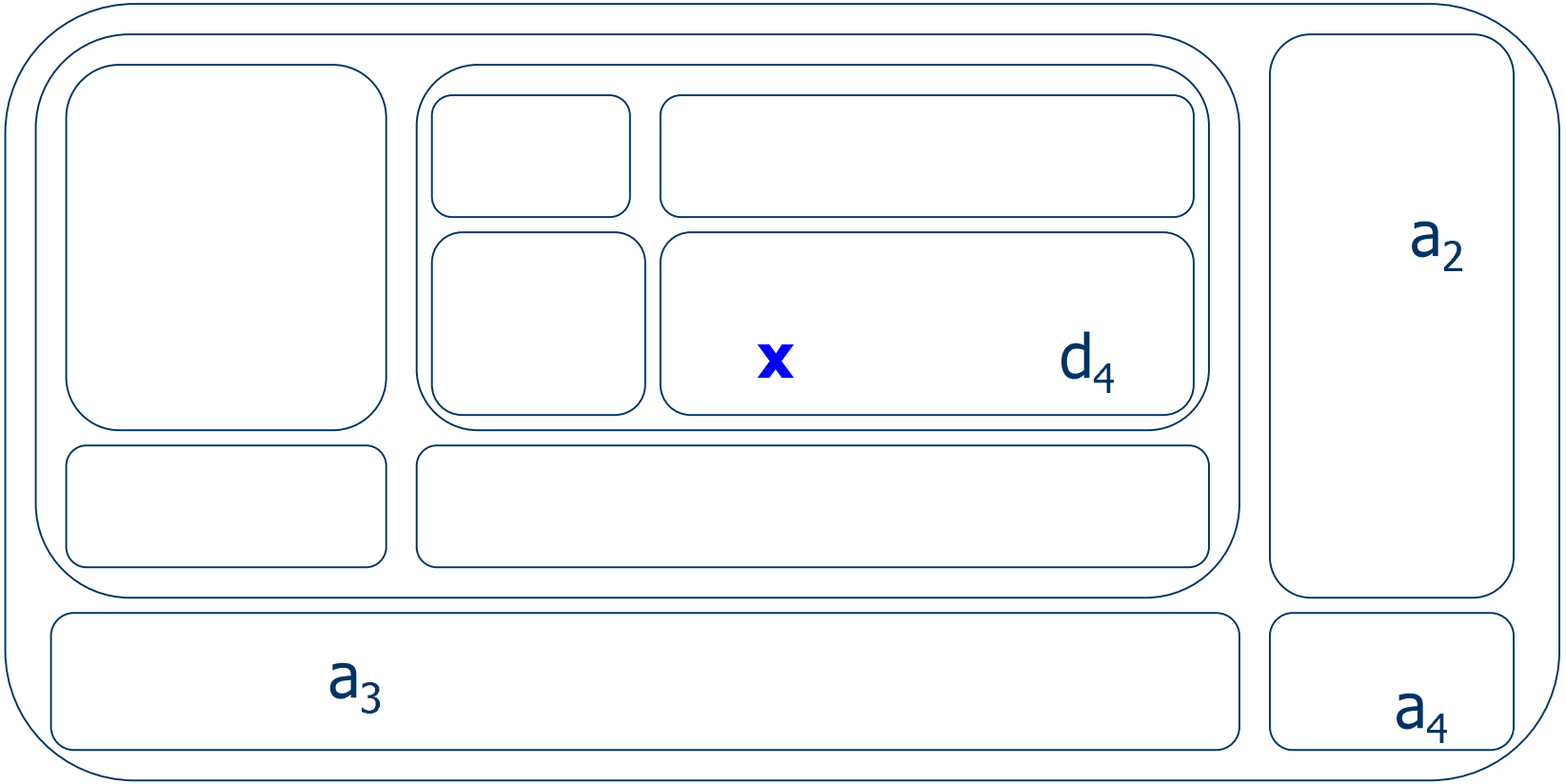
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



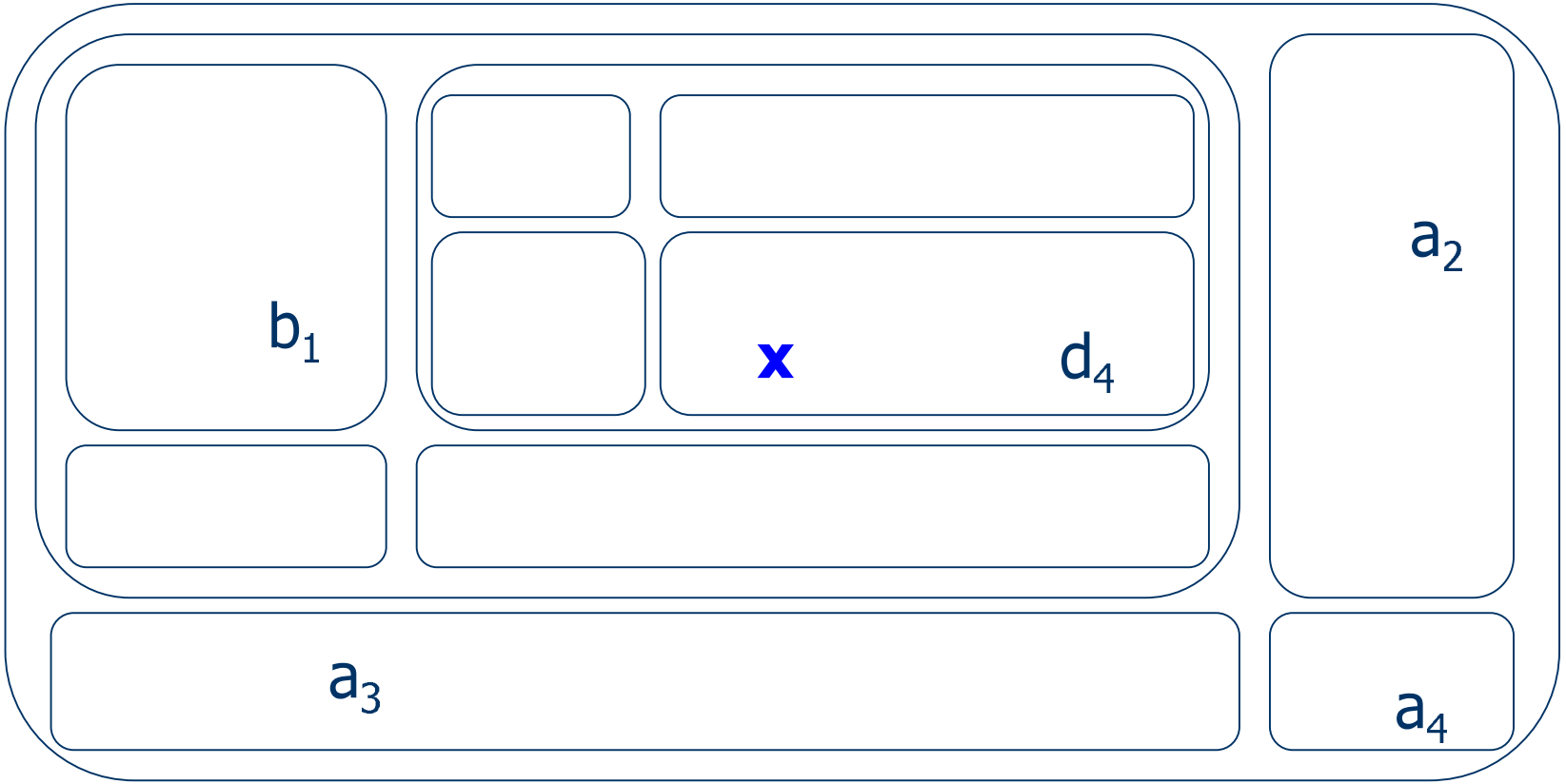
ノードの参加(例)

d_4は$\boldsymbol{k}$が属するLevel 1のsubcluserからノードを選び
NewNodeを送る



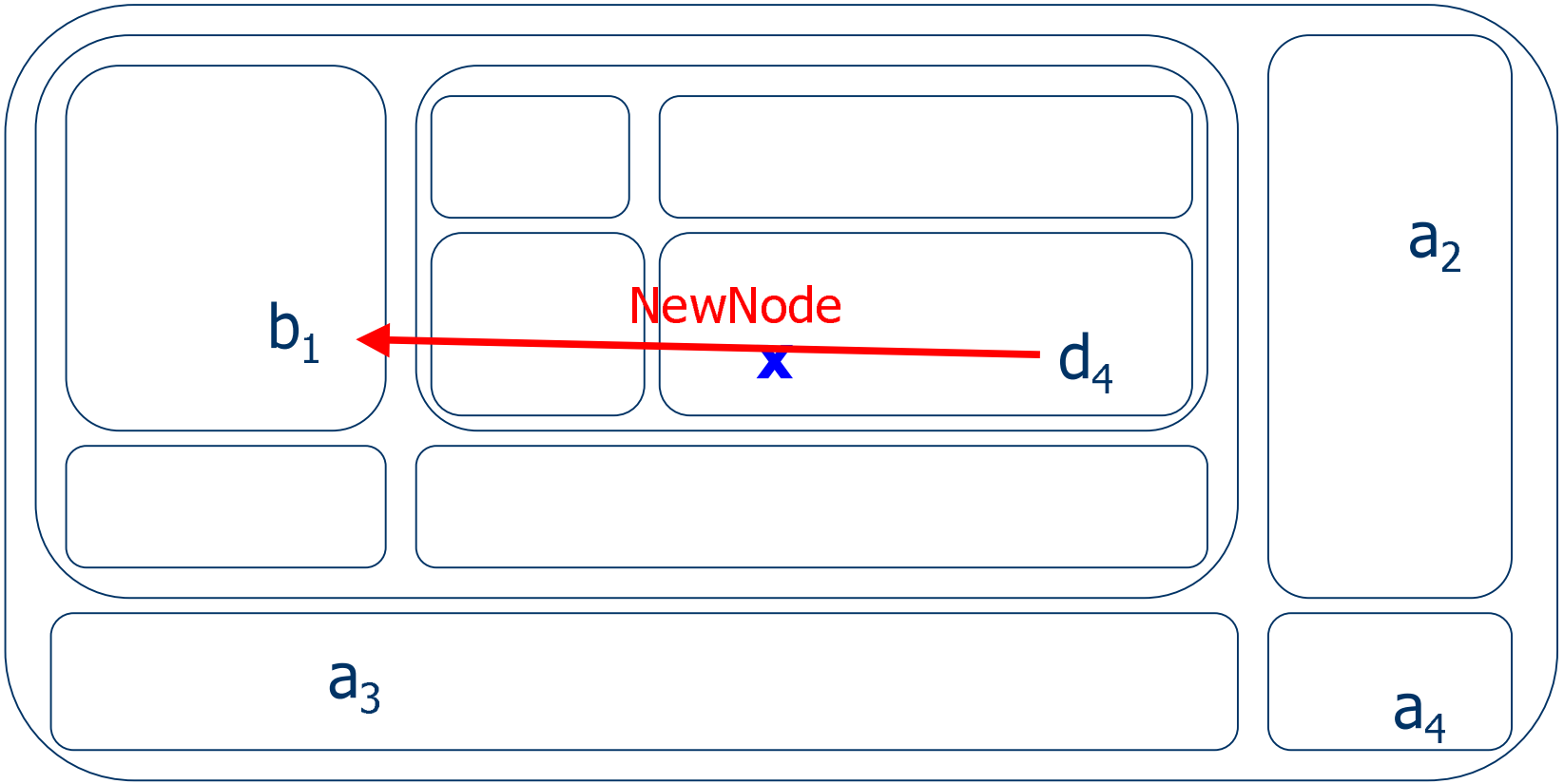
ノードの参加(例)

d_4は$\boldsymbol{k}$が属するLevel 1のsubcluserからノードを選び
NewNodeを送る



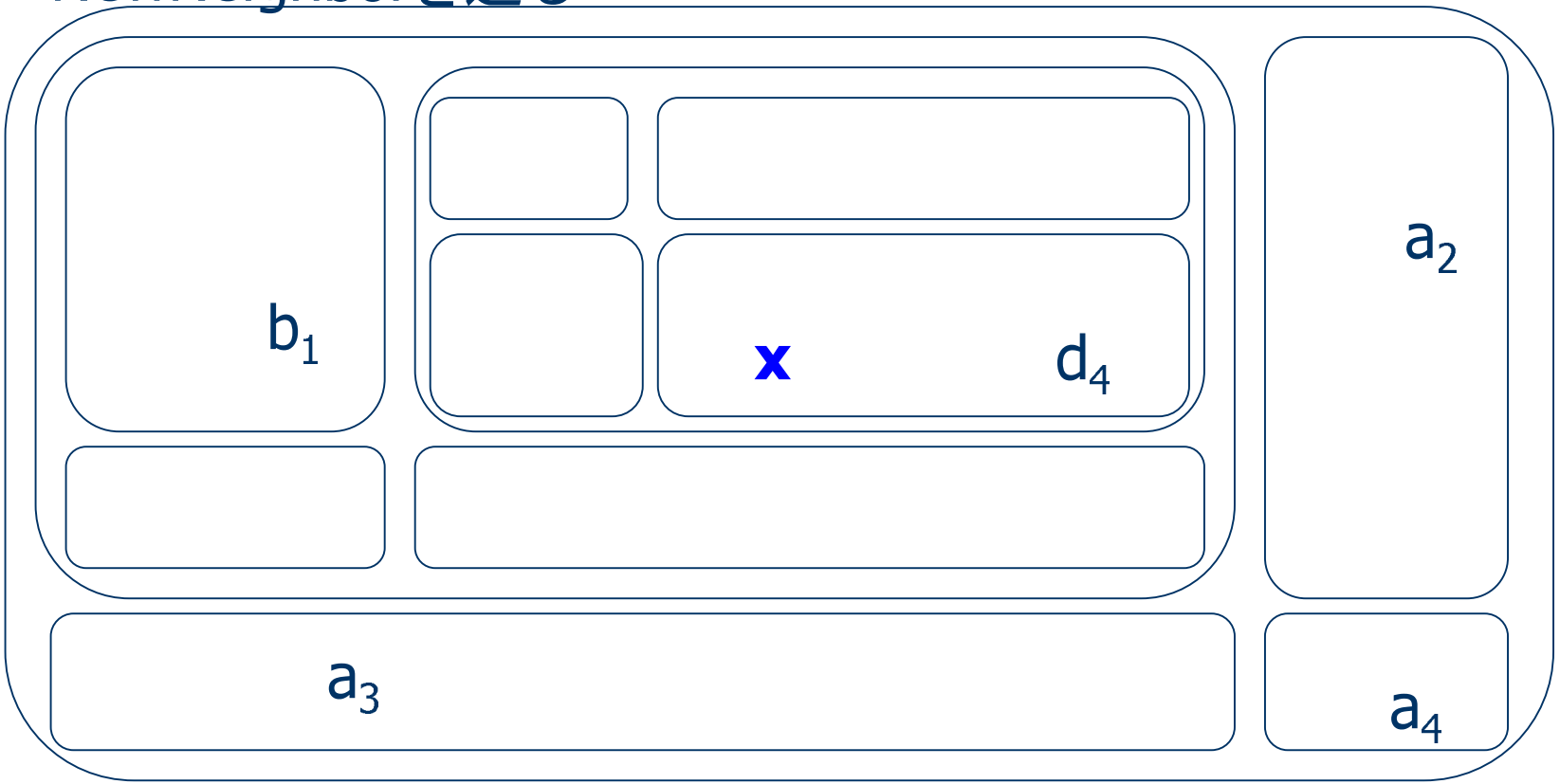
ノードの参加(例)

d_4は$\boldsymbol{k}$が属するLevel 1のsubcluserからノードを選び
NewNodeを送る



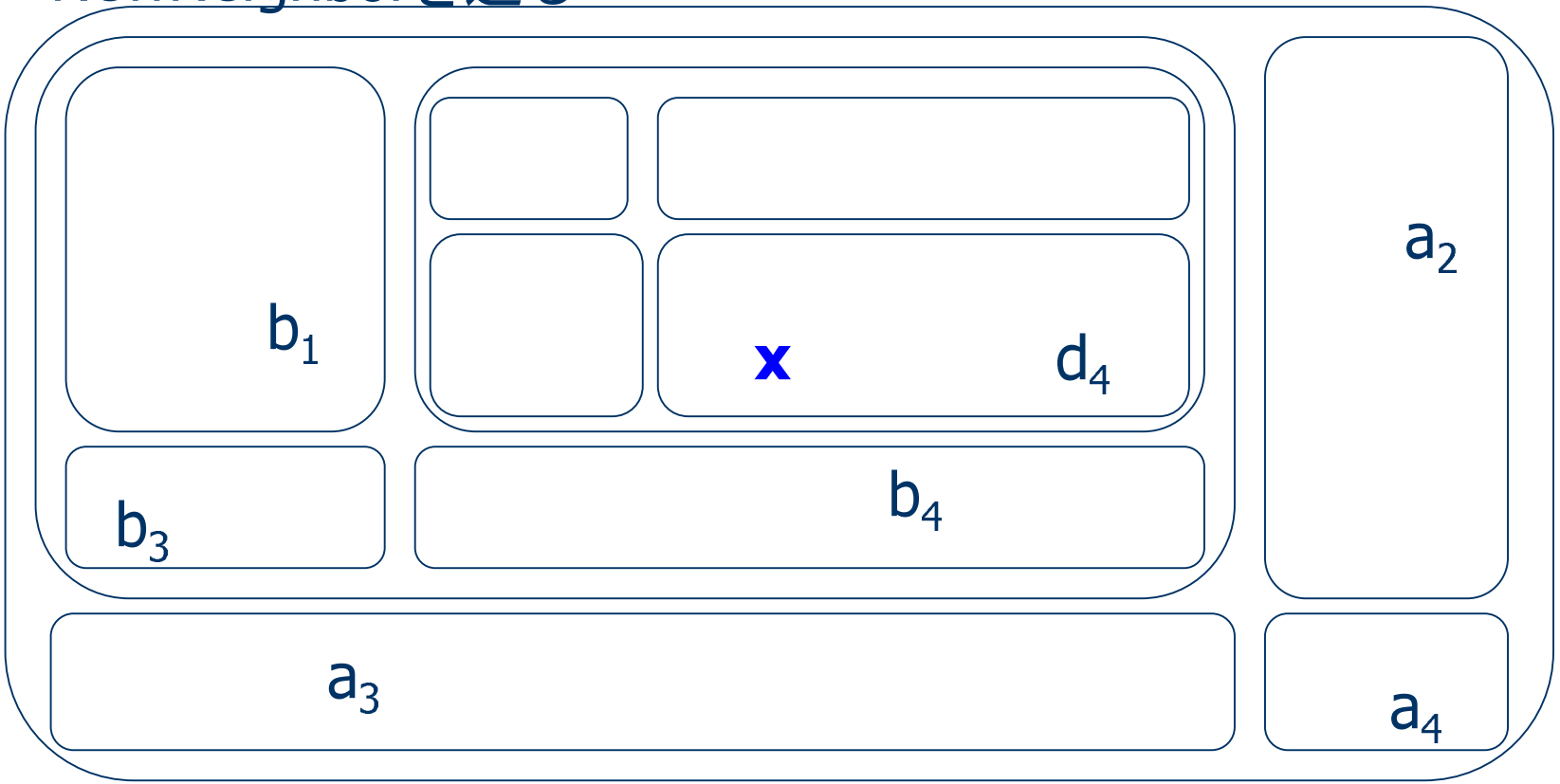
ノードの参加(例)

b_1はkが属さないLevel 2のsubcluserからノードを選びNewNeighborを送る



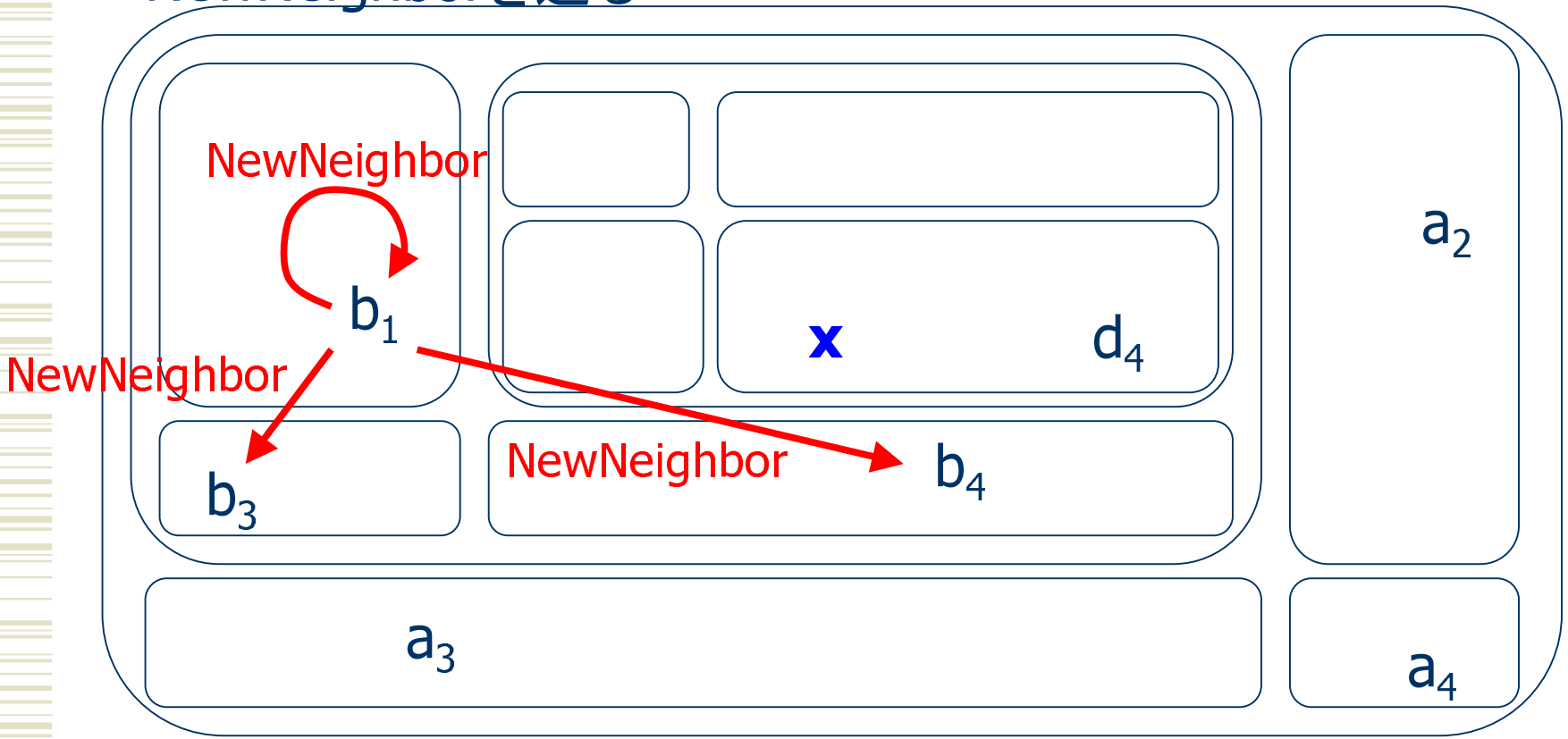
ノードの参加(例)

b_1はkが属さないLevel 2のsubcluserからノードを選びNewNeighborを送る



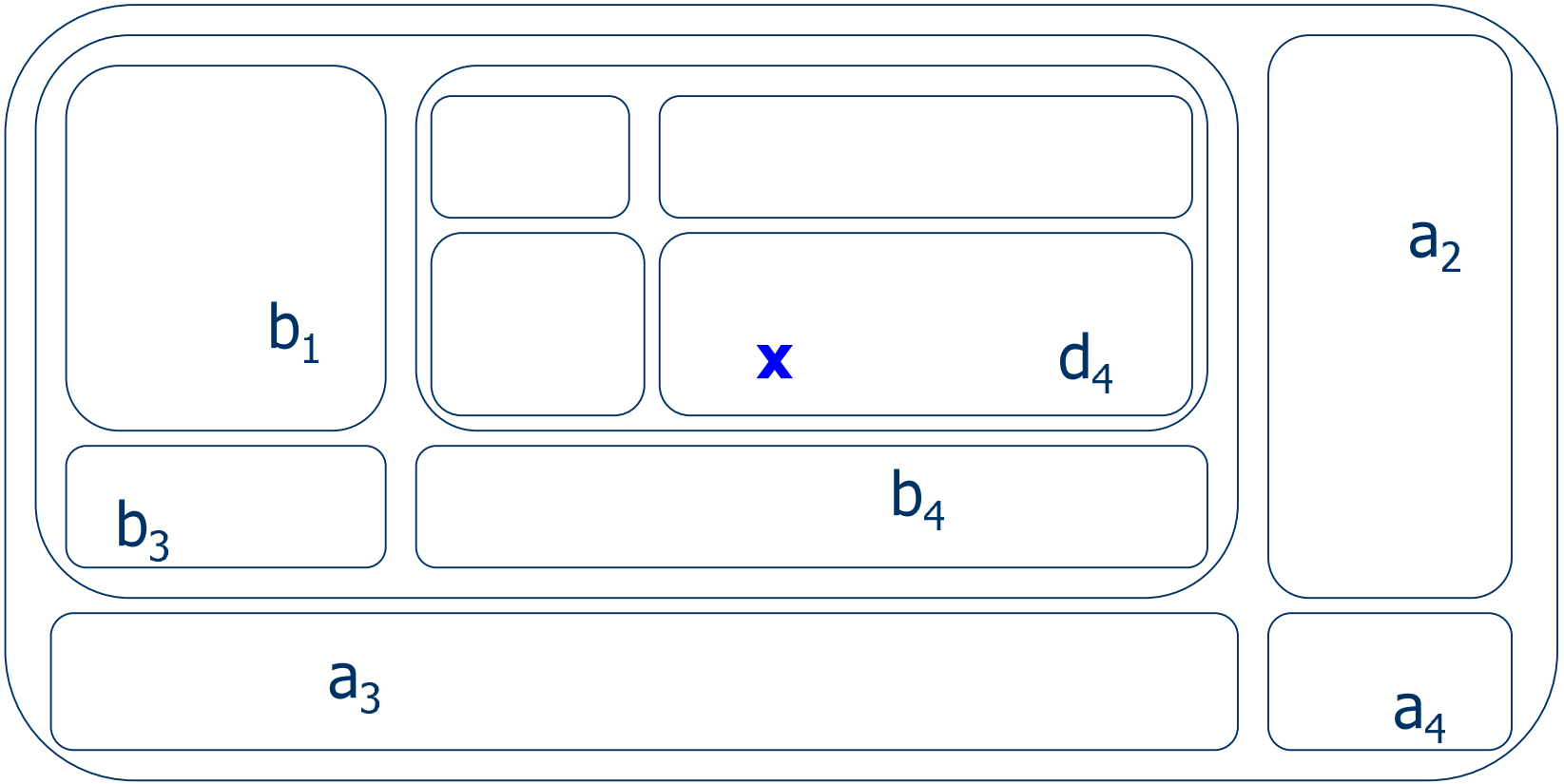
ノードの参加(例)

b_1はkが属さないLevel 2のsubcluserからノードを選び
NewNeighborを送る



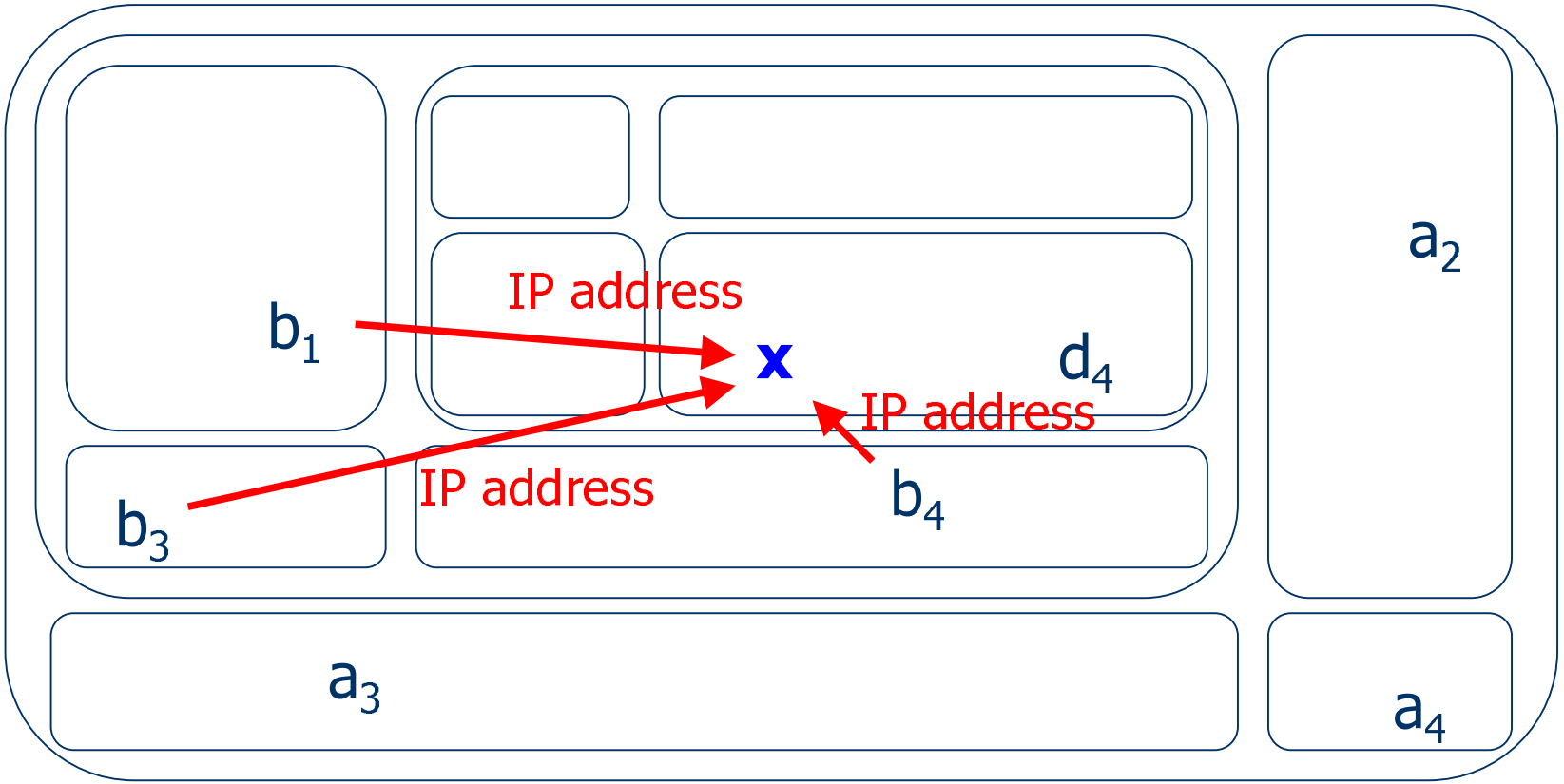
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



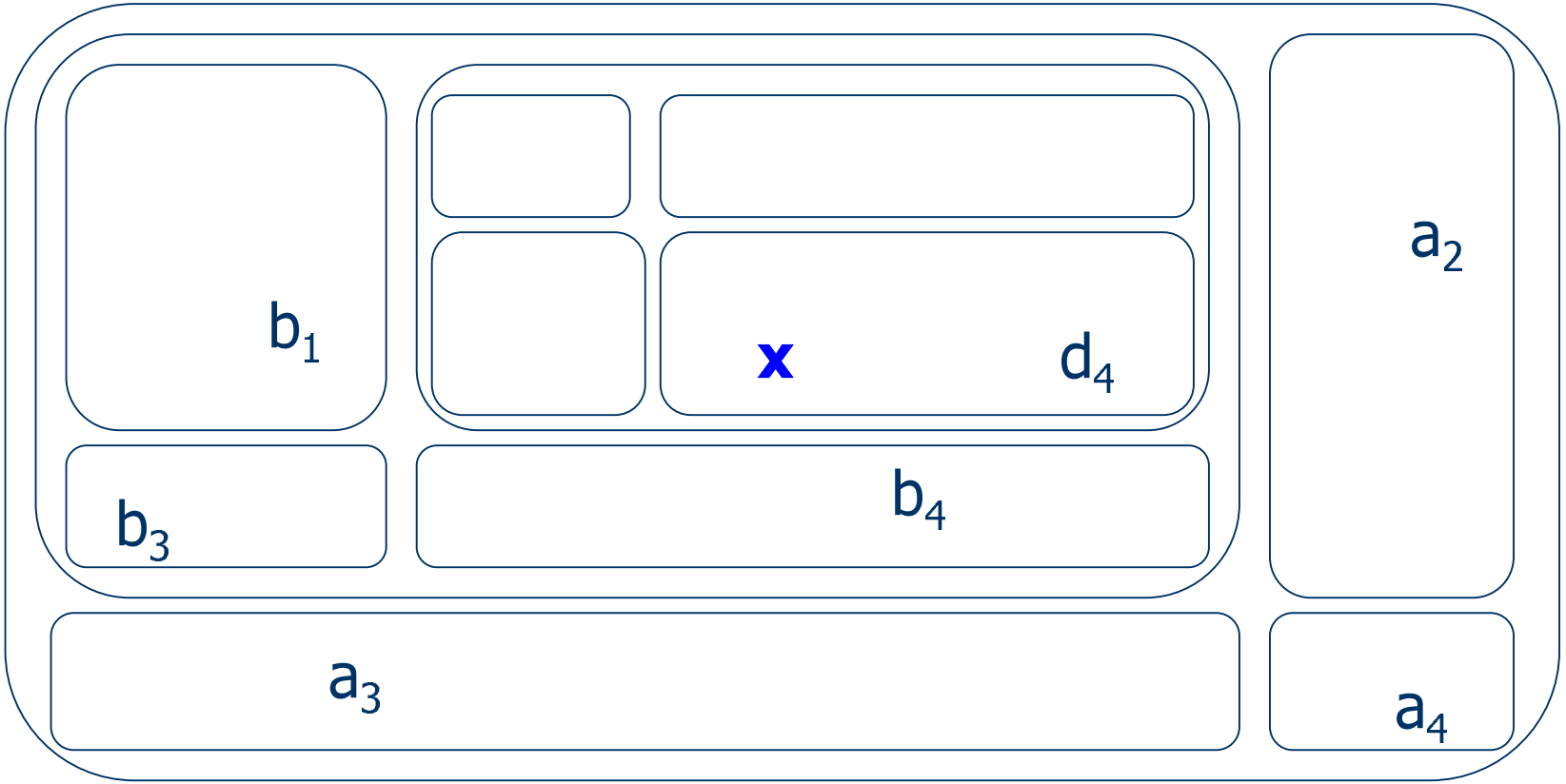
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



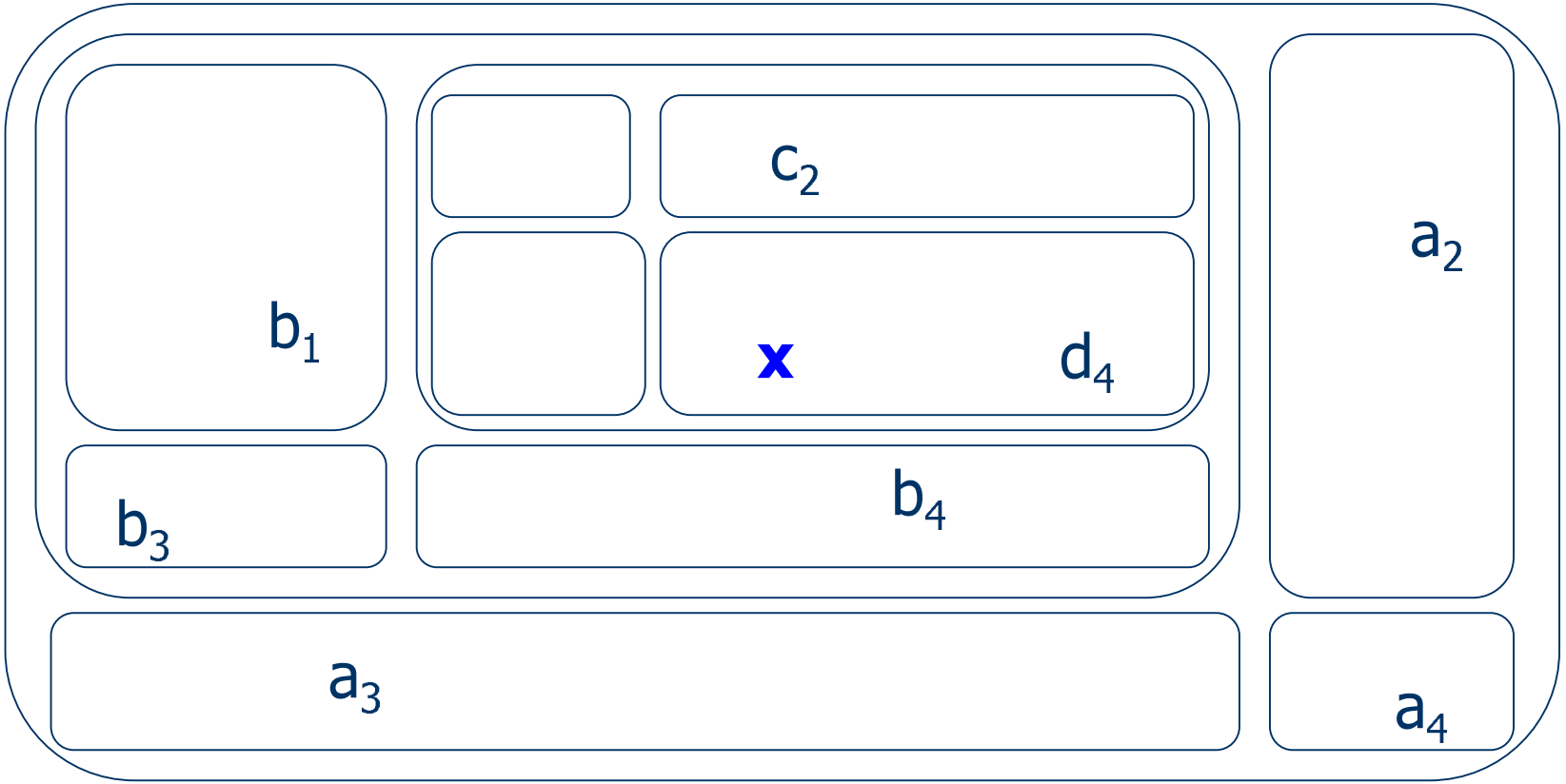
ノードの参加(例)

b_1は$\boldsymbol{k}$が属するLevel 2のsubcluserからノードを選び
NewNodeを送る



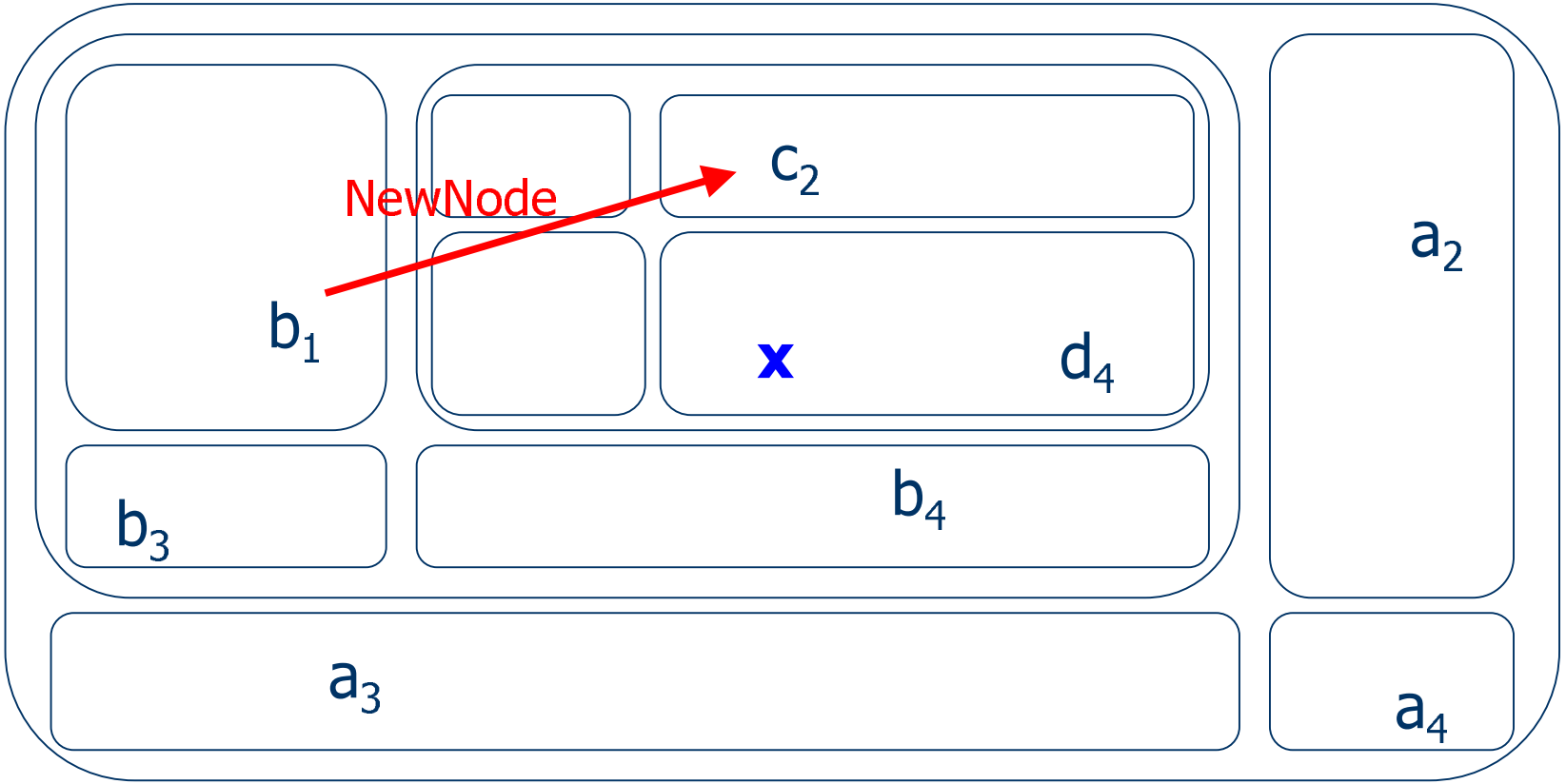
ノードの参加(例)

b_1は$\boldsymbol{k}$が属するLevel 2のsubcluserからノードを選び
NewNodeを送る



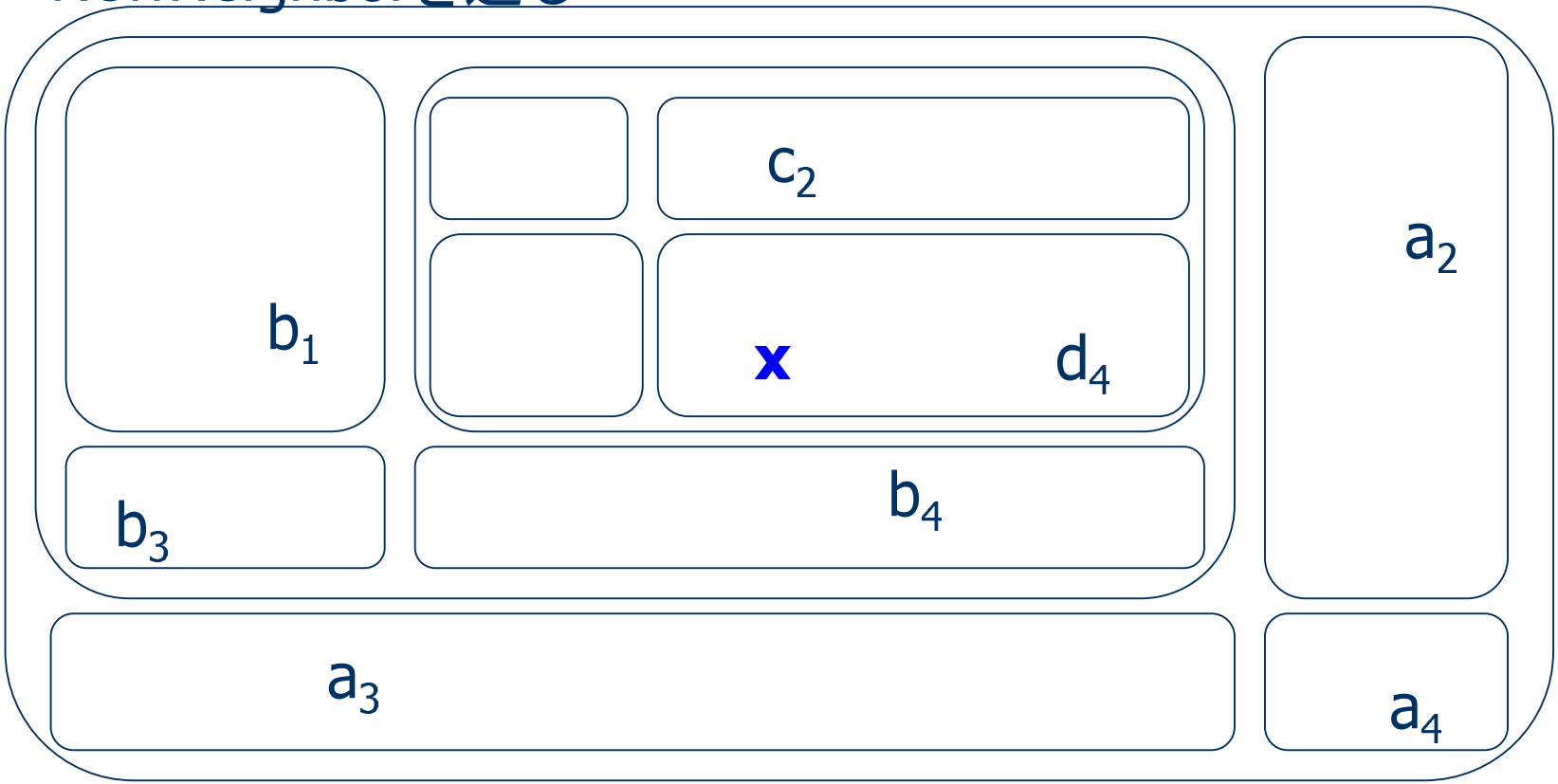
ノードの参加(例)

b_1は$\boldsymbol{k}$が属するLevel 2のsubcluserからノードを選び
NewNodeを送る



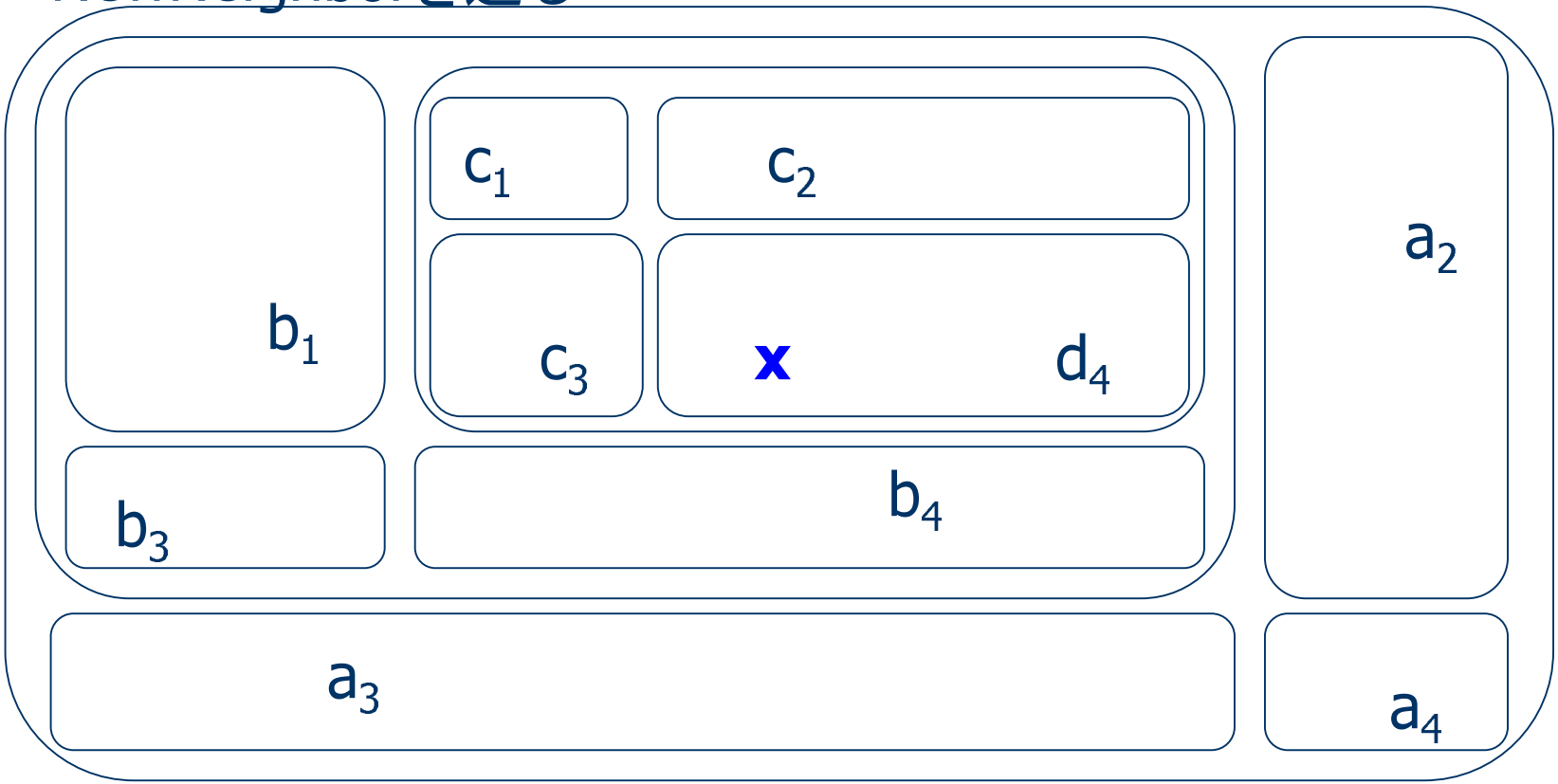
ノードの参加(例)

c_2は$\boldsymbol{k}$が属さないLevel 3のsubcluserからノードを選びNewNeighborを送る



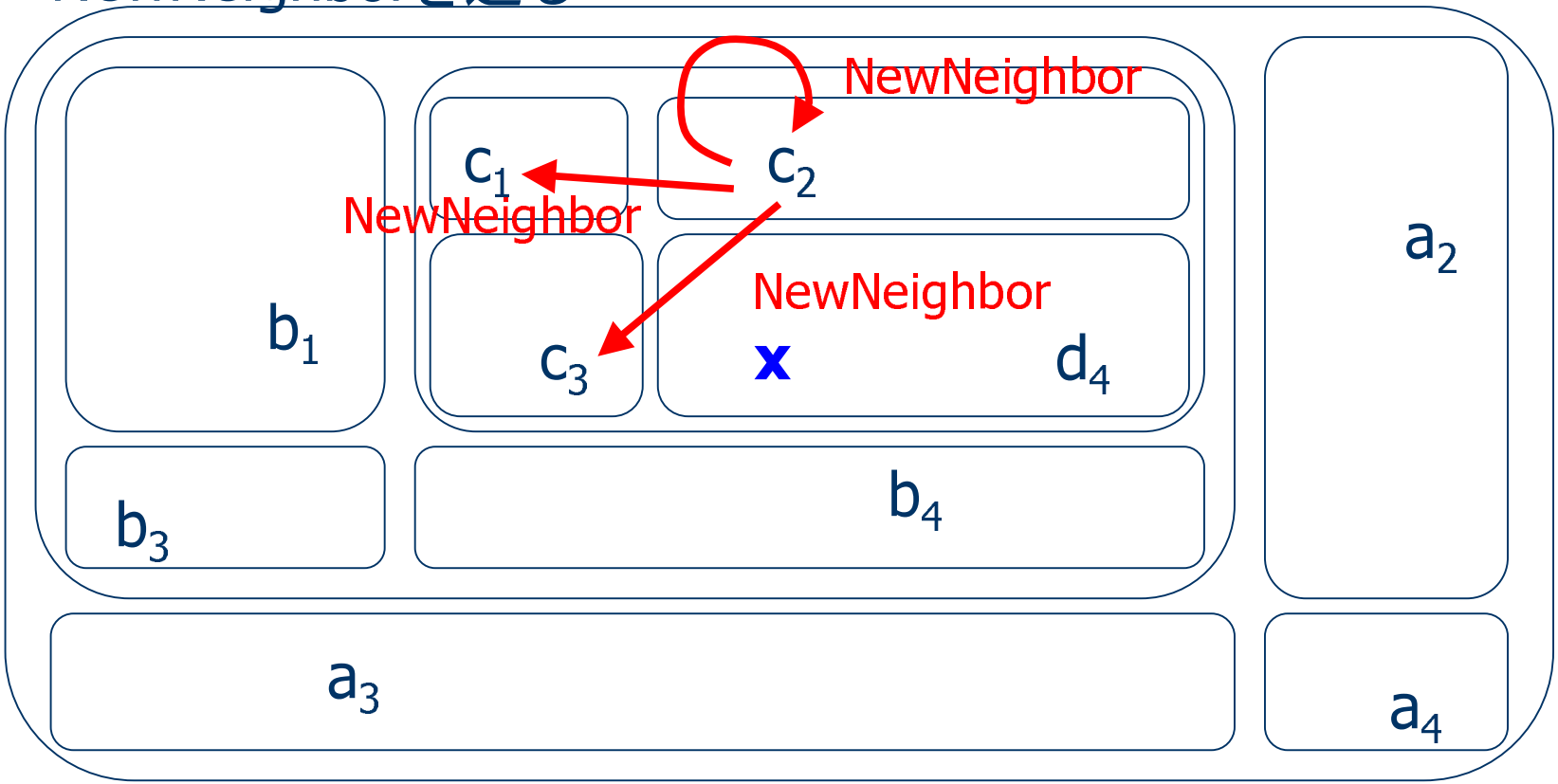
ノードの参加(例)

c_2は$\boldsymbol{k}$が属さないLevel 3のsubcluserからノードを選びNewNeighborを送る



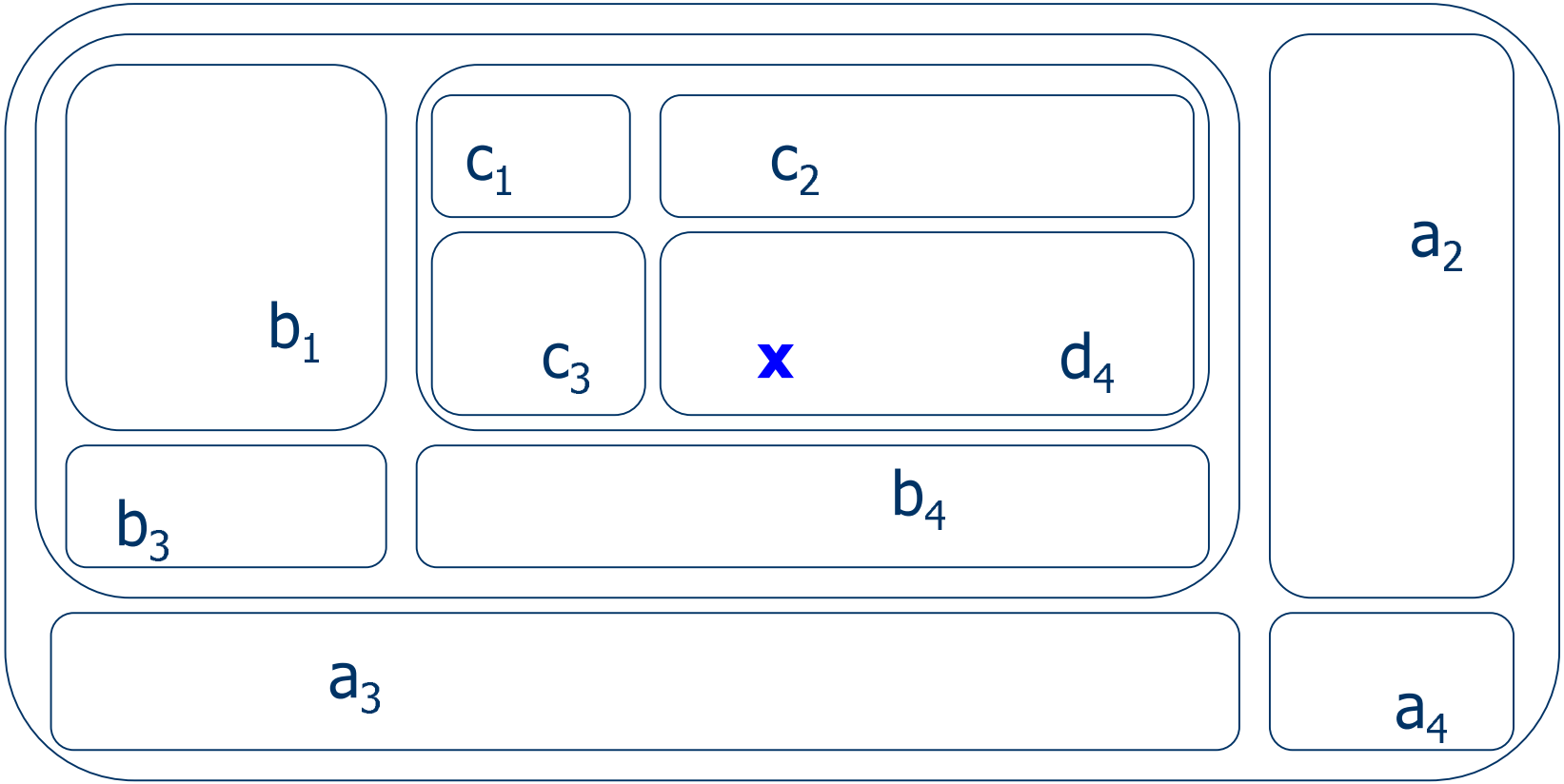
ノードの参加(例)

c_2はkが属さないLevel 3のsubcluserからノードを選びNewNeighborを送る



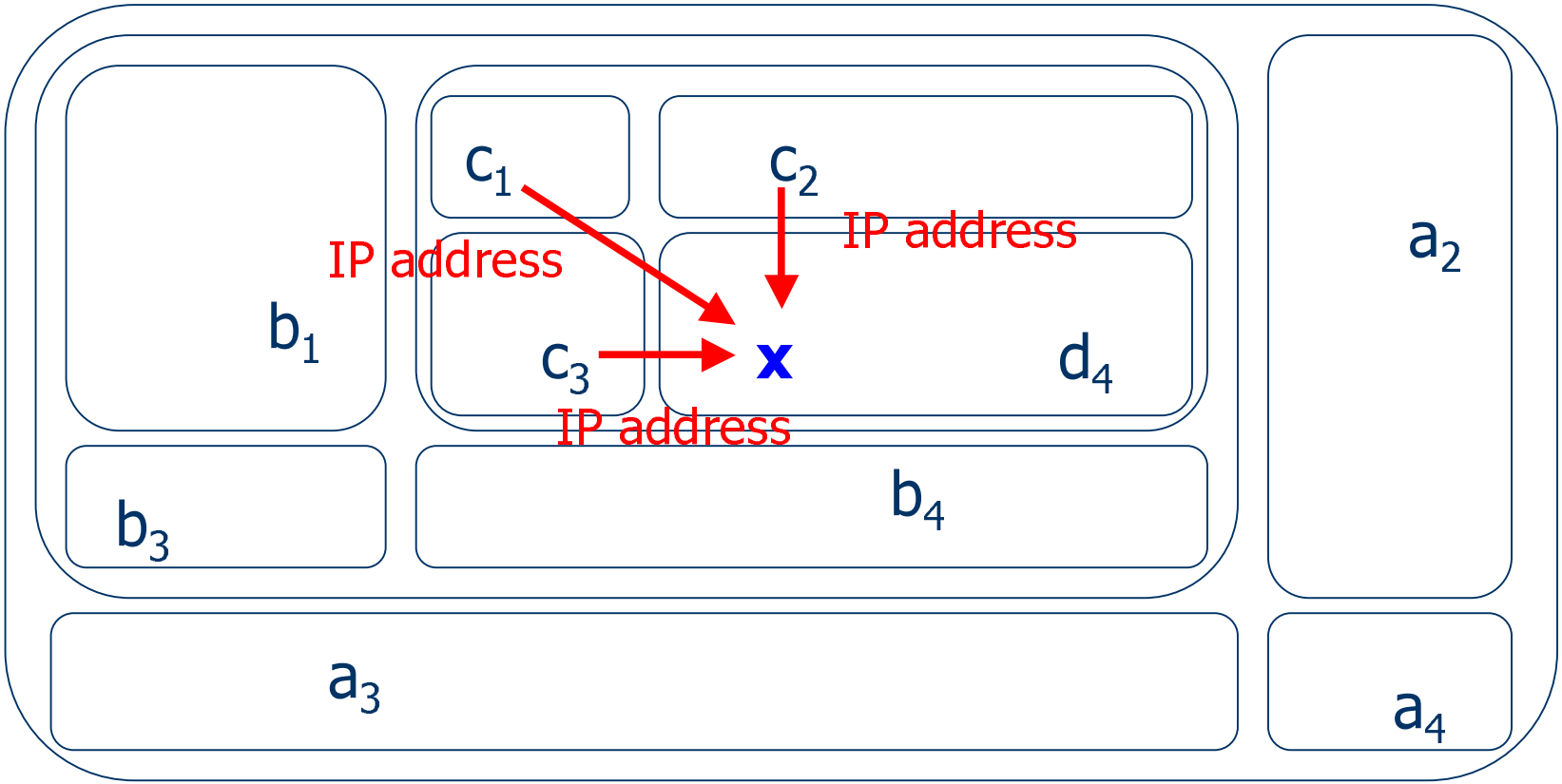
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



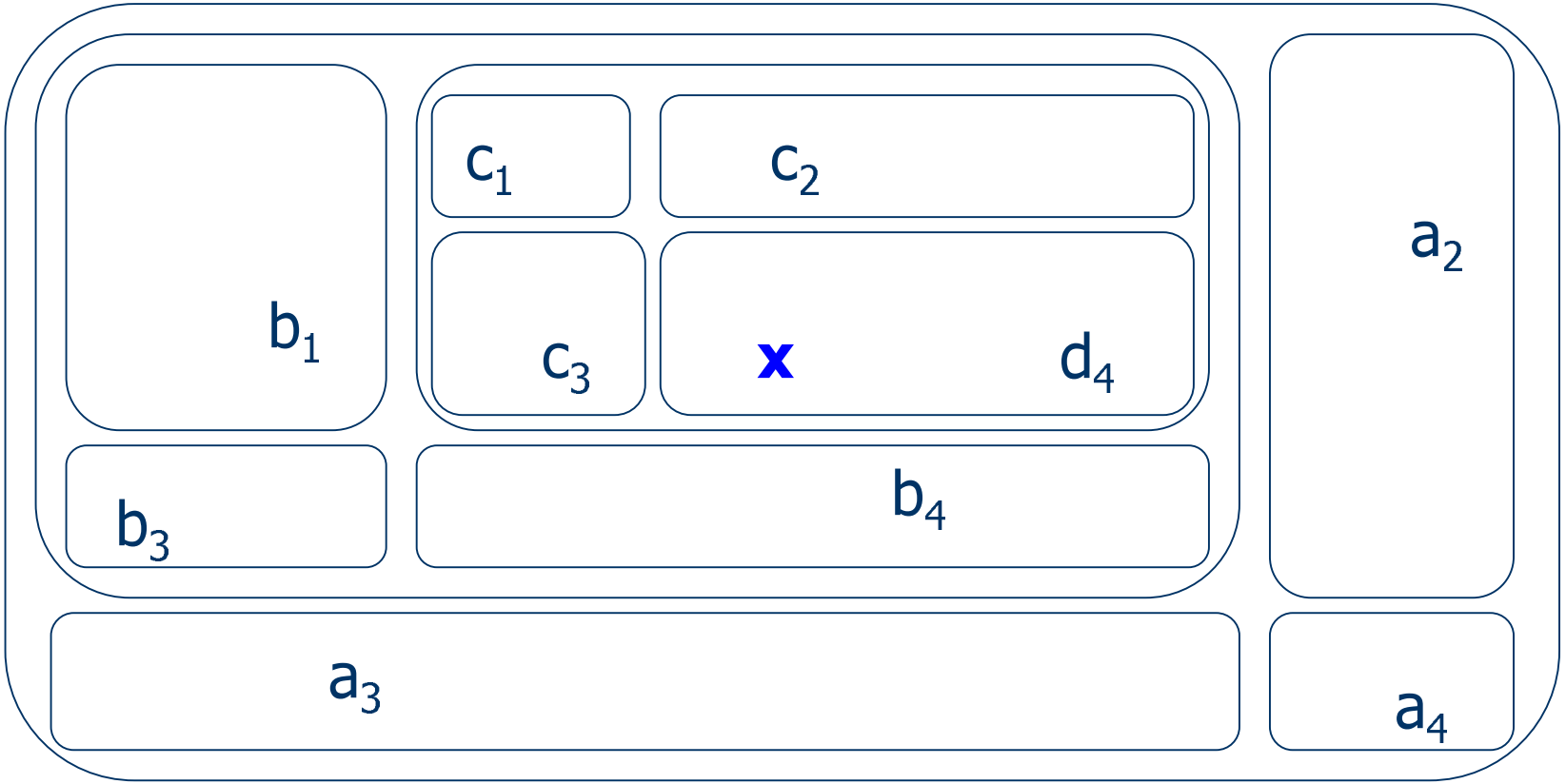
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



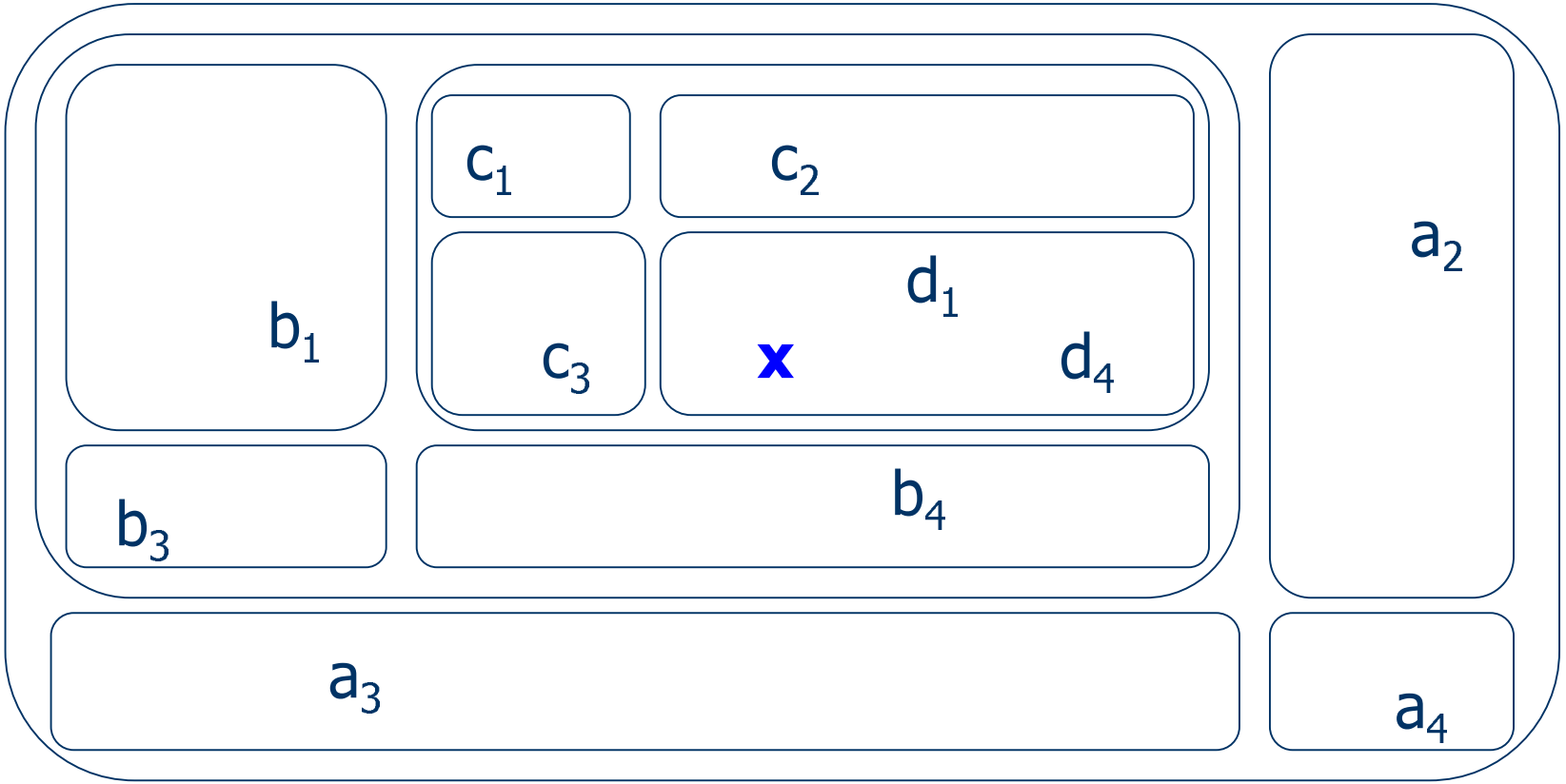
ノードの参加(例)

c_2はkが属するLevel 3のsubcluserからノードを選び
NewNodeを送る



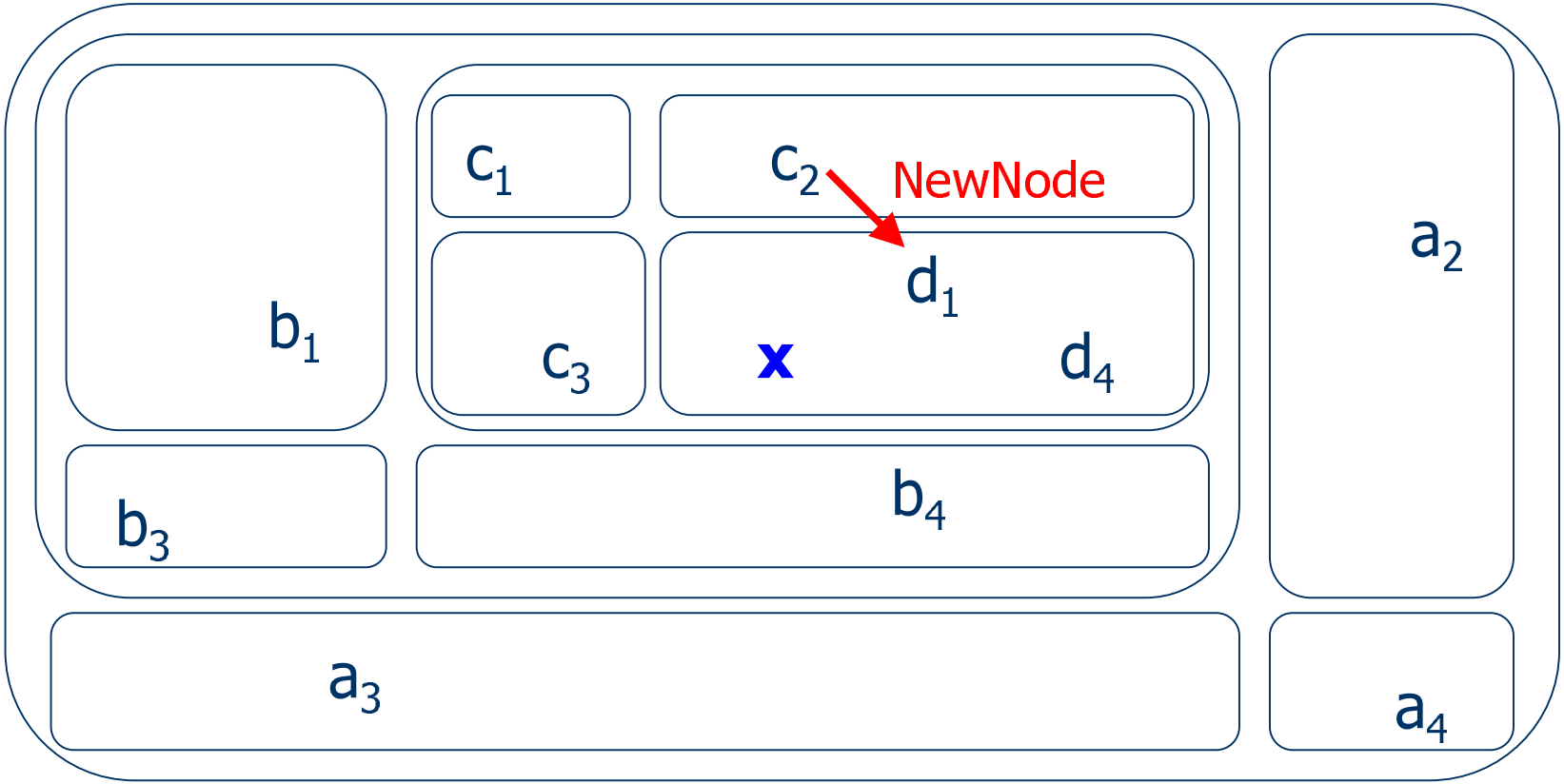
ノードの参加(例)

c_2はkが属するLevel 3のsubcluserからノードを選びNewNodeを送る



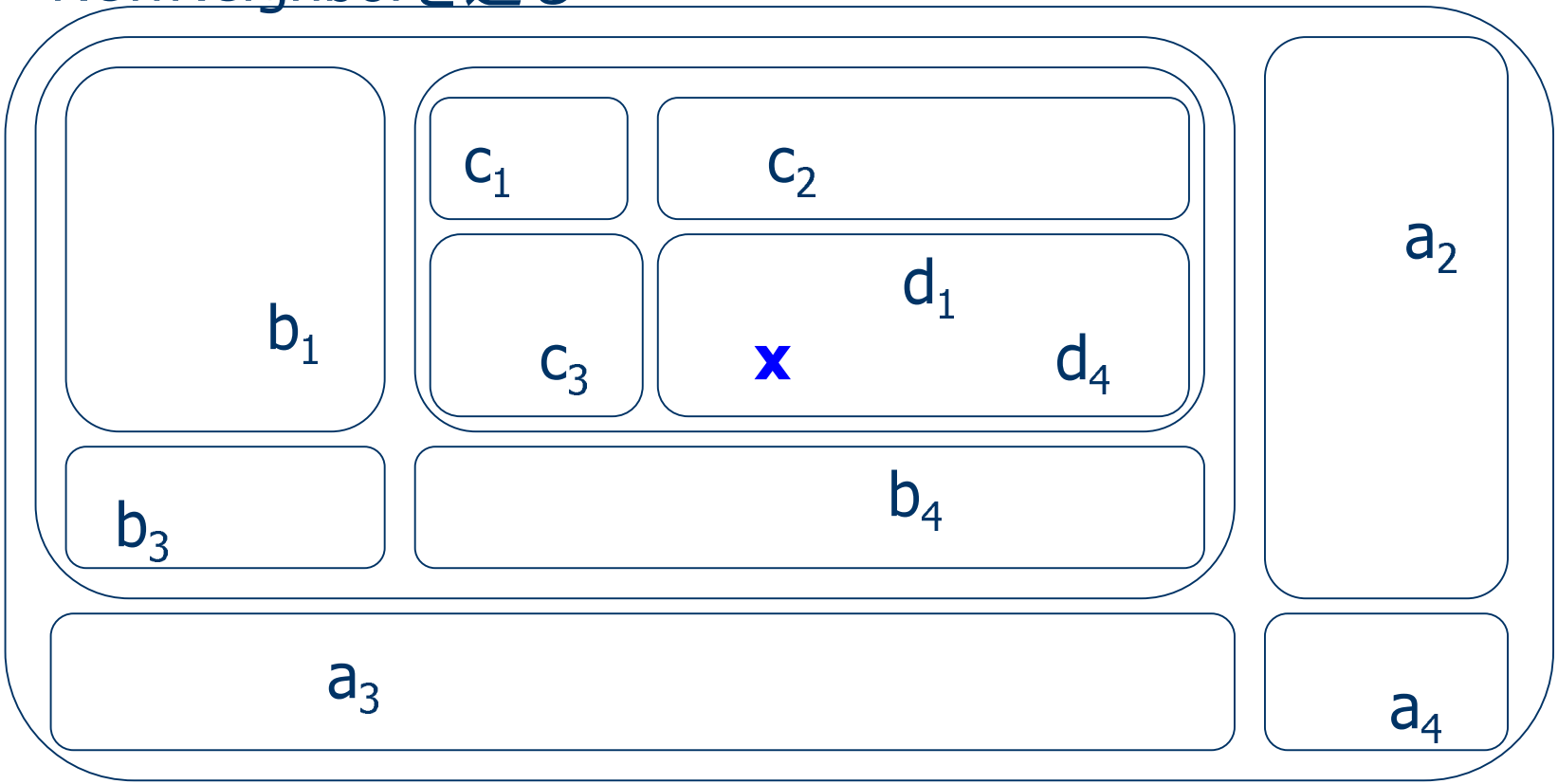
ノードの参加(例)

c_2はkが属するLevel 3のsubcluserからノードを選び
NewNodeを送る



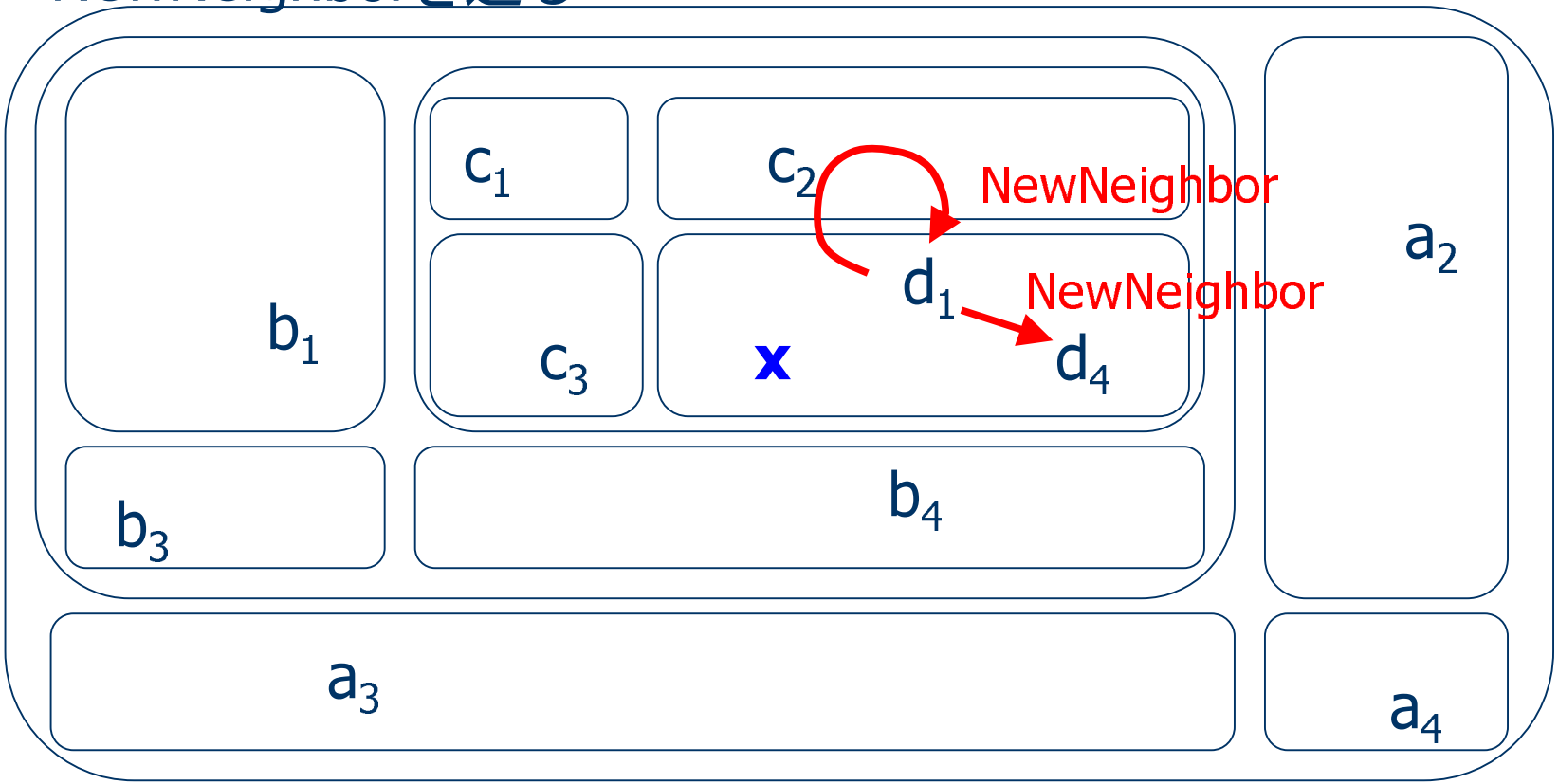
ノードの参加(例)

d_1は$\boldsymbol{k}$が属さないLevel 4のsubcluserからノードを選び
NewNeighborを送る



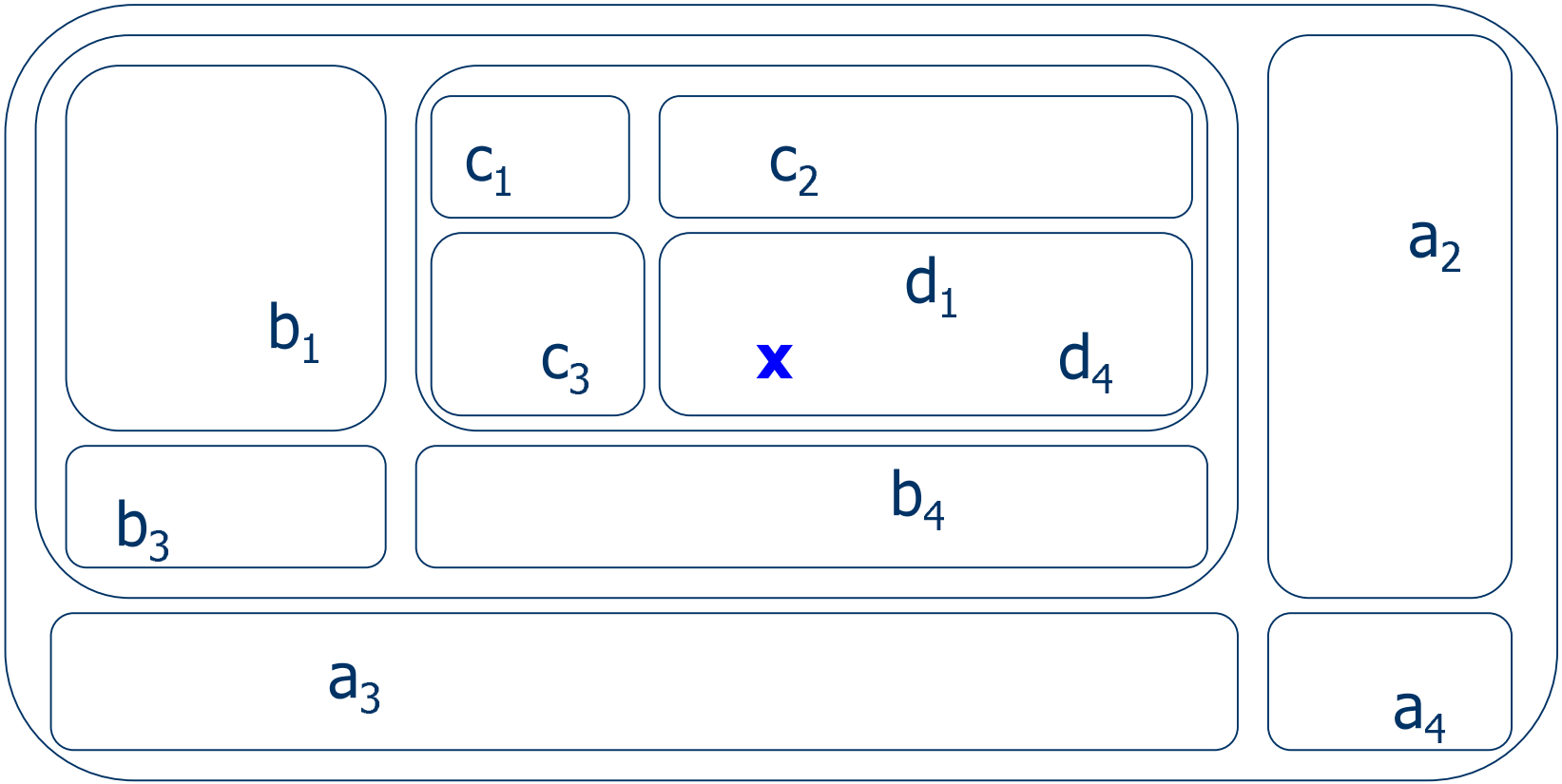
ノードの参加(例)

d_1は$\boldsymbol{k}$が属さないLevel 4のsubcluserからノードを選びNewNeighborを送る



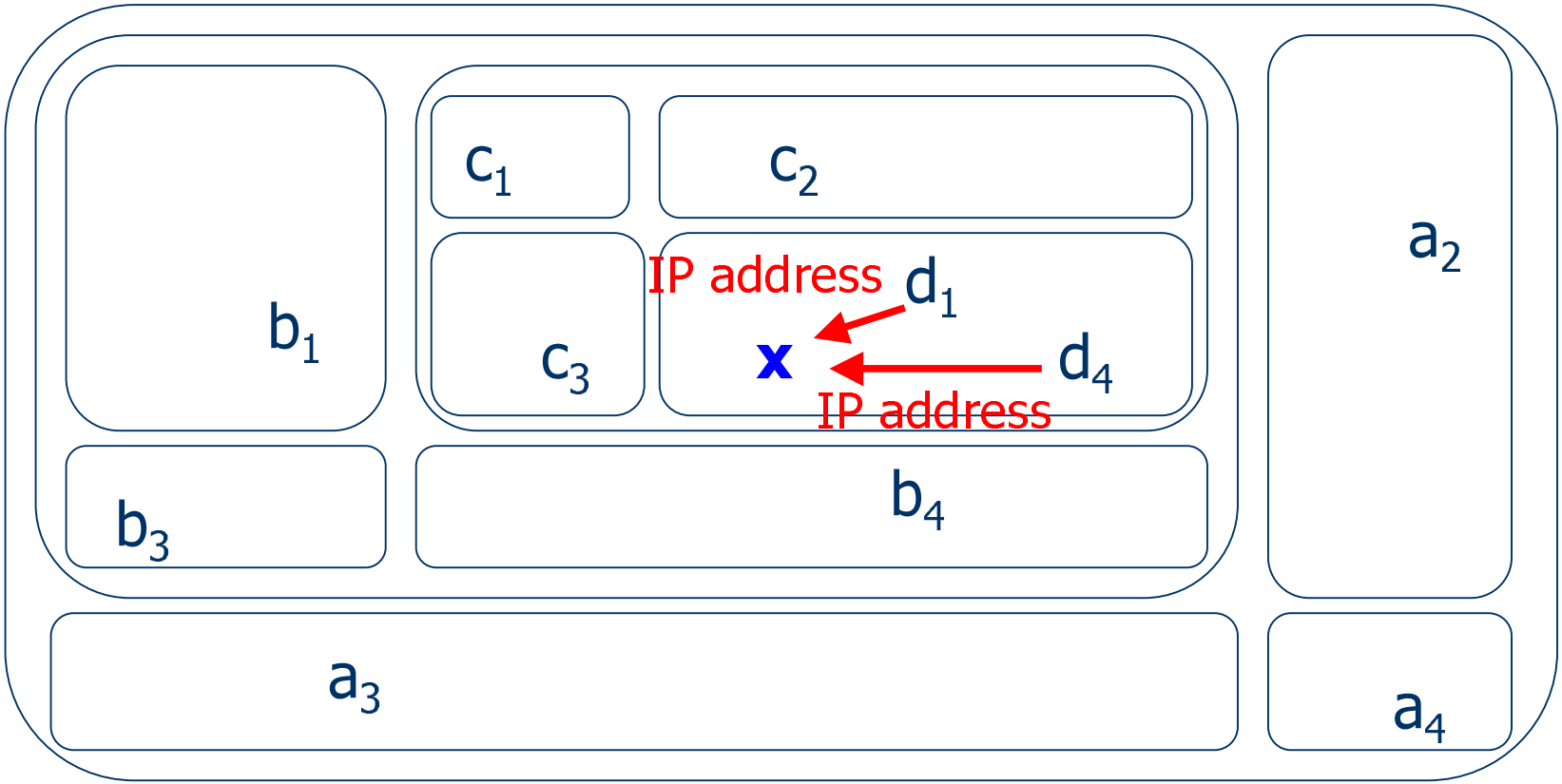
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



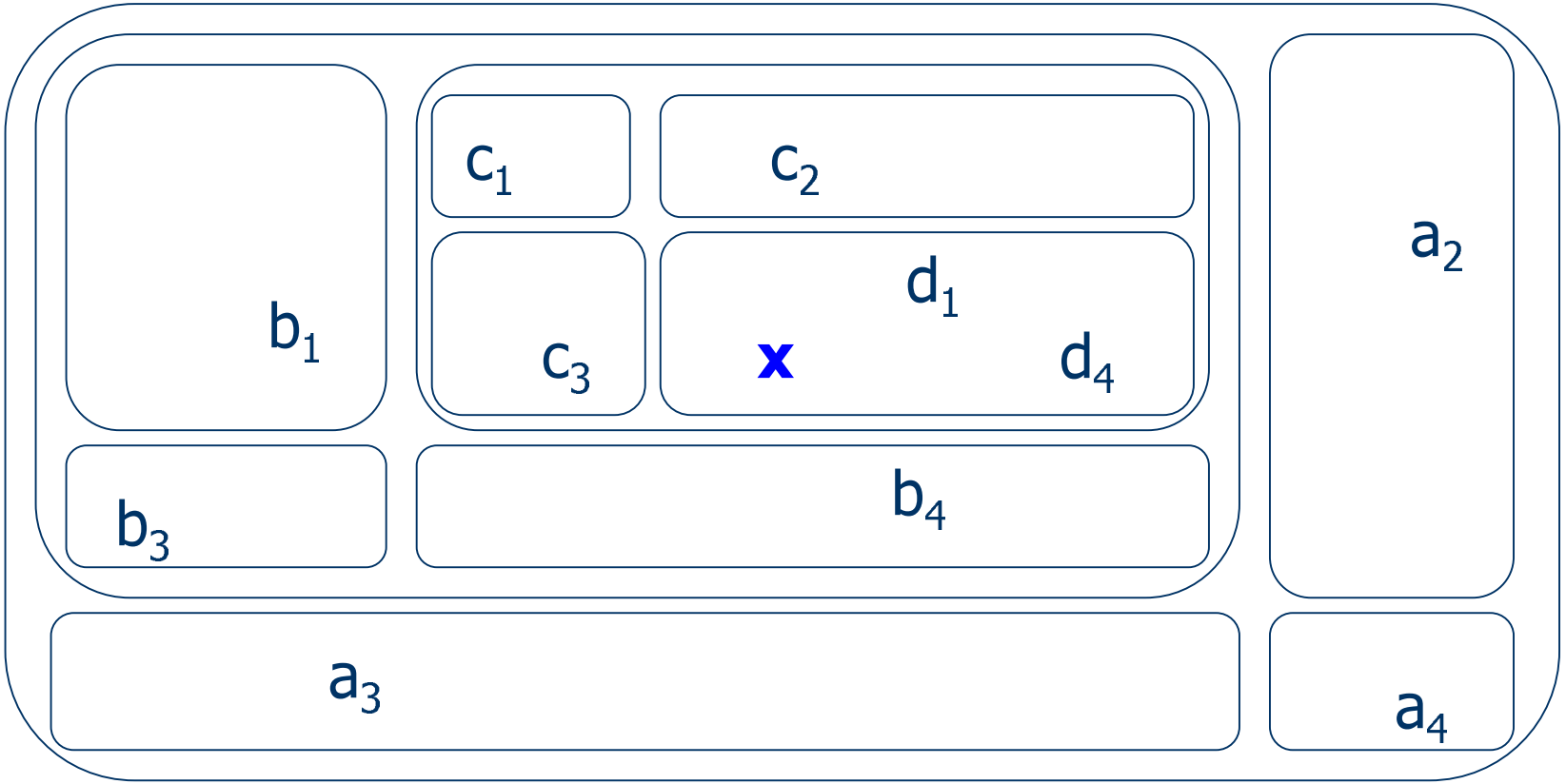
ノードの参加(例)

NewNeighborを受け取ったノードはIP addressを教える



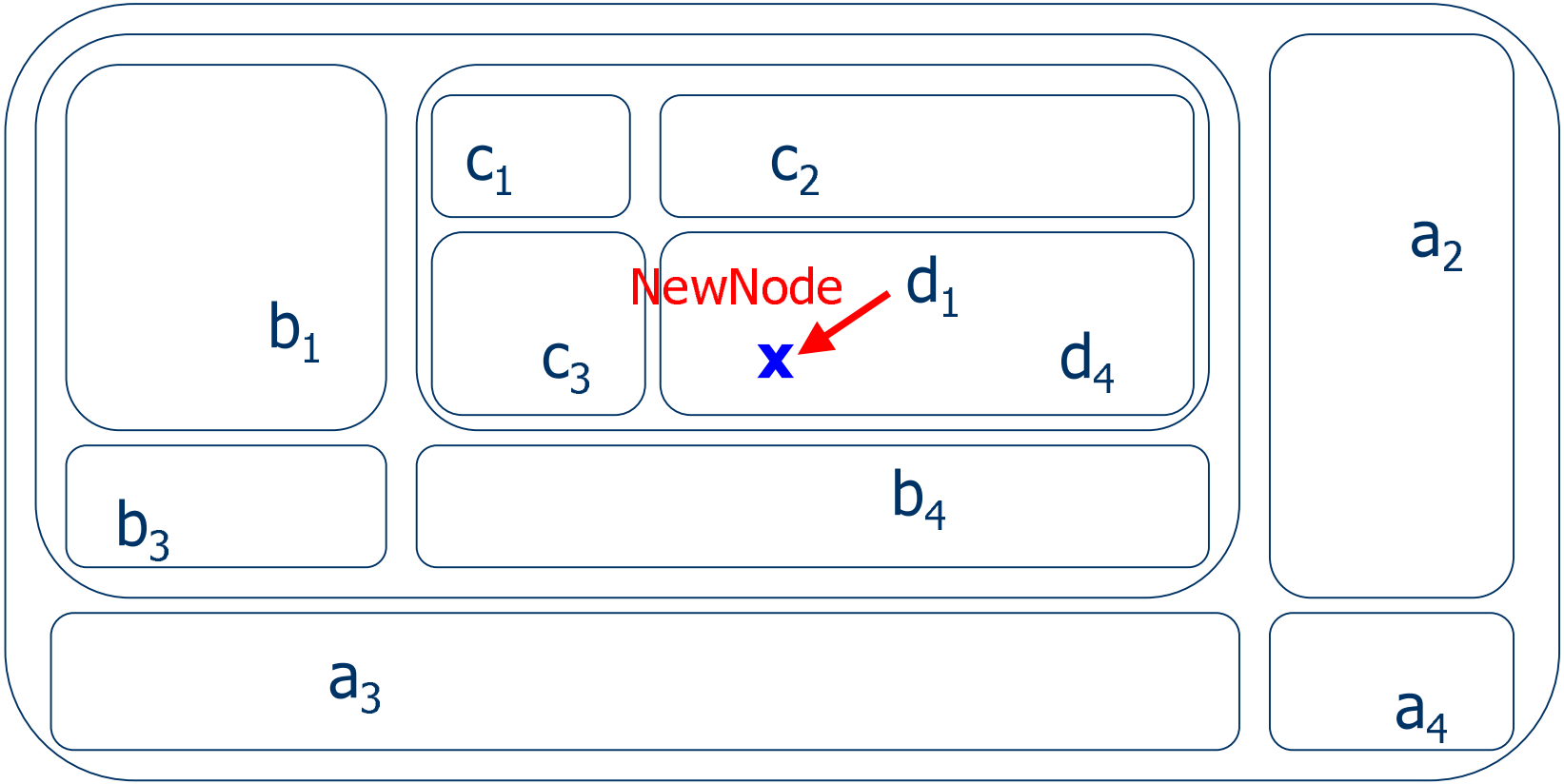
ノードの参加(例)

d_1は$\boldsymbol{k}$が属するLevel 4のsubcluserからノードを選び
NewNodeを送る



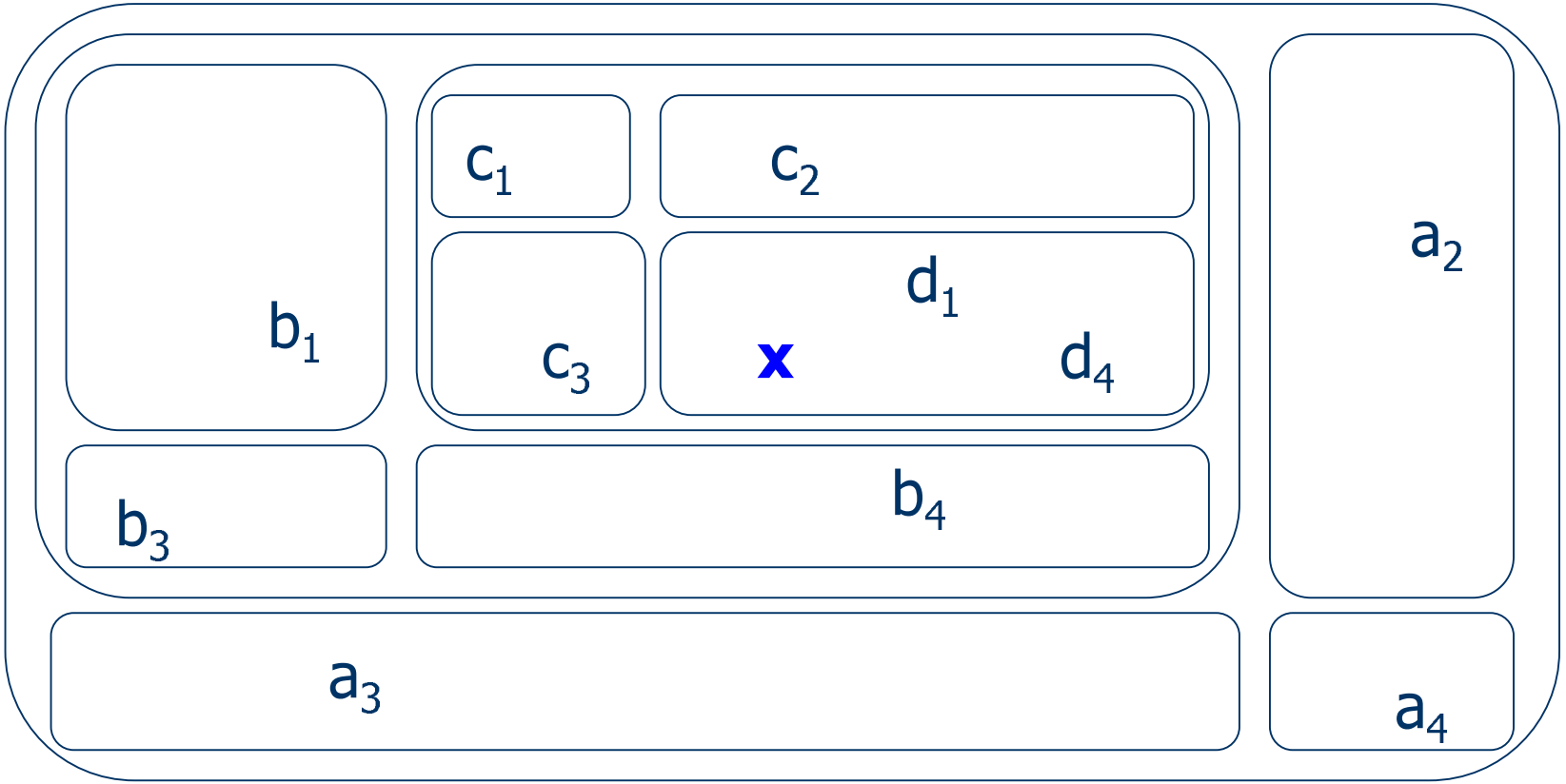
ノードの参加(例)

d_1は$\boldsymbol{k}$が属するLevel 4のsubcluserからノードを選び
NewNodeを送る



ノードの参加(例)

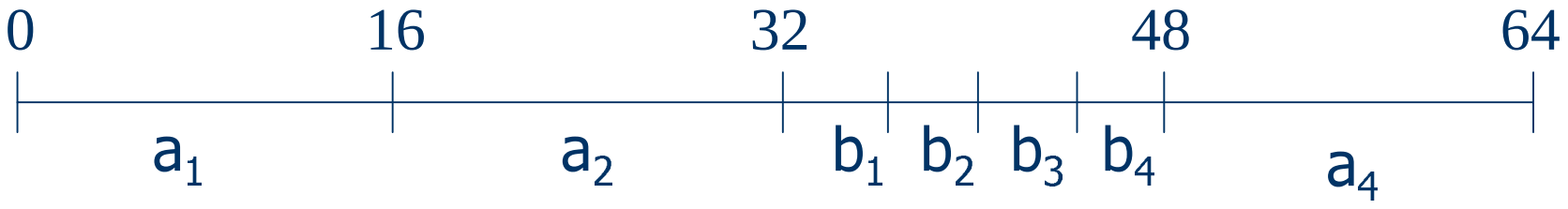
xは受け取ったIP addressをもとにrouting tableを構築する



ノードの脱退

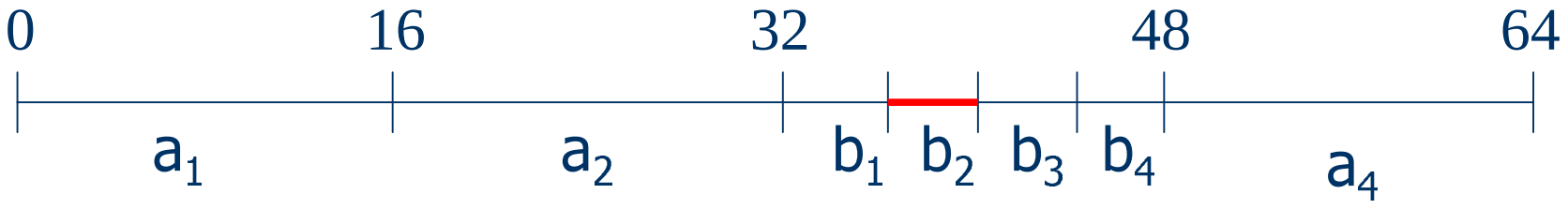
- 脱退ノードの区間を$[a,b]$で、level iとする
- 脱退ノードが属するlevel $i-1$のclusterの区間を$[\alpha,\beta]$とする
- 次の整数を含む区間のノードが脱退ノードの区間を引き取る
  - $a-1 \in [\alpha,\beta]$の時, $a-1$
  - そうでない時, $\beta-1$

ノードの脱退(例)

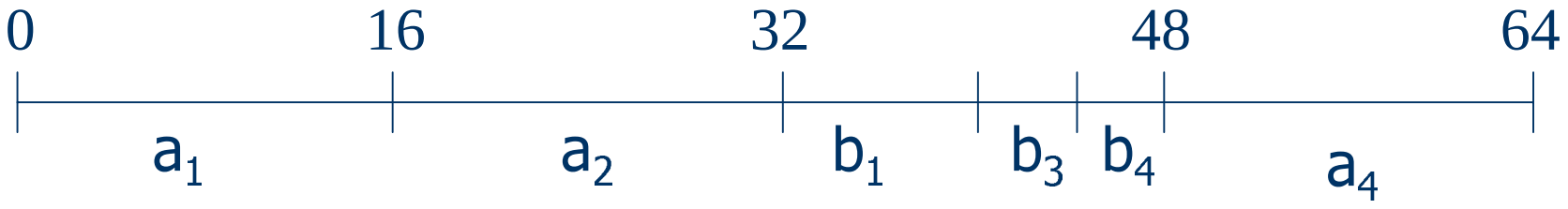


ノードの脱退(例)

b_2が脱退

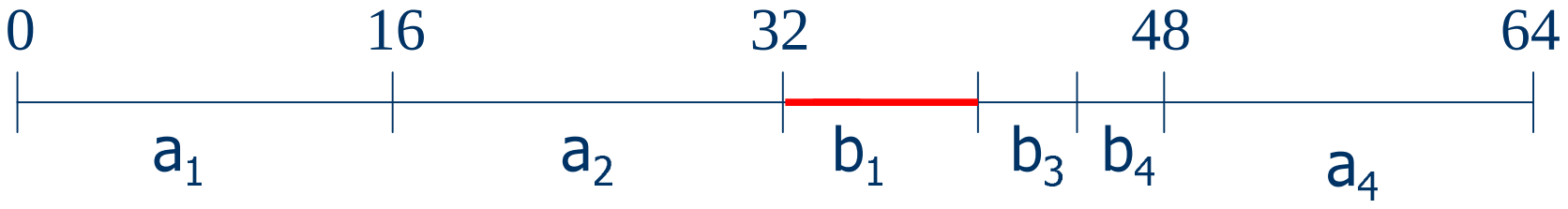


ノードの脱退(例)

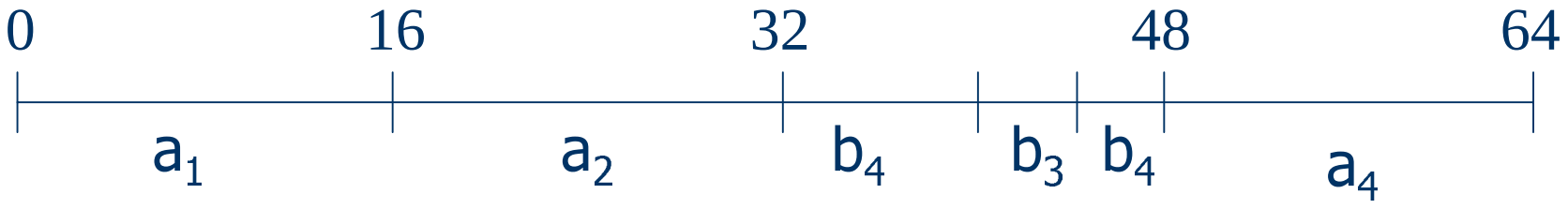


ノードの脱退(例)

b_1が脱退

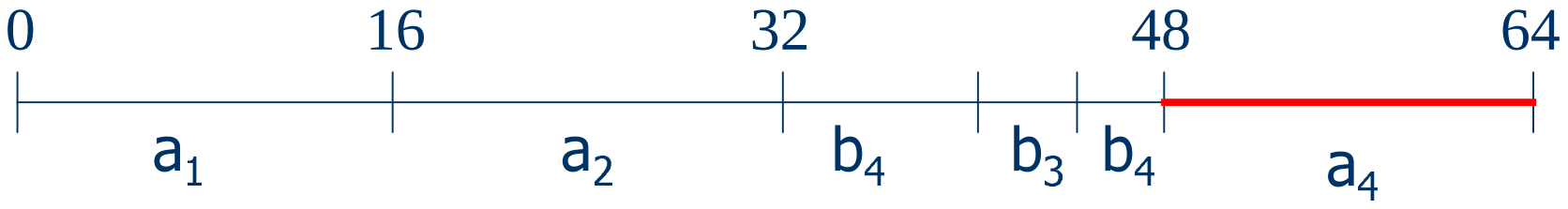


ノードの脱退(例)

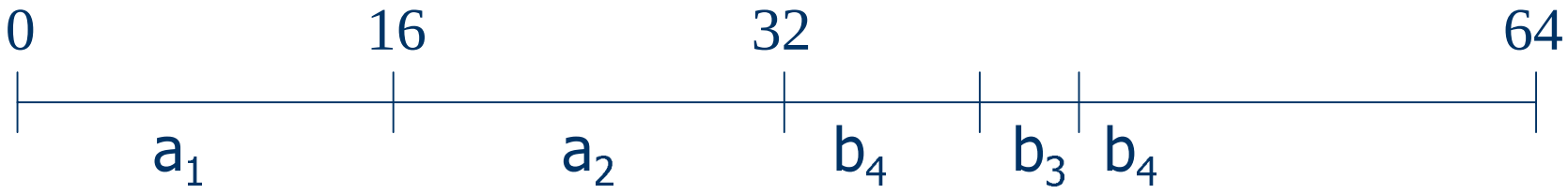


ノードの脱退(例)

a_4が脱退

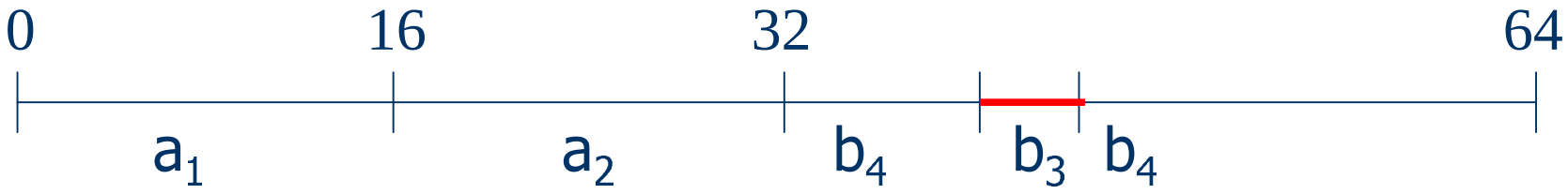


ノードの脱退(例)

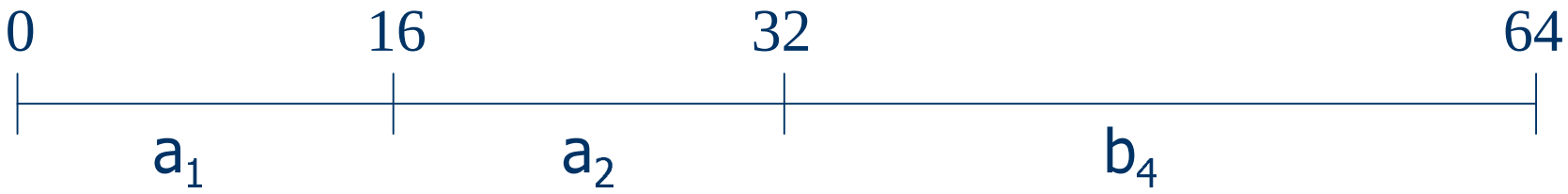


ノードの脱退(例)

b_3が脱退



ノードの脱退(例)



Fault tolerance

- Network failure
  - FailしたchannelのRouting tableのエントリの区間に属する乱数kを選び、kを含む区間のノードとchannelを確立しようとする
- Node failure
  - 考え中

関連研究

- Pastry [Rowstron et al. '01]
  - 128bitのランダムにノードIDを割り振る
  - メッセージをあて先の128bitのkeyにもっとも近いIDのノードへroutingする
  - 平均 $O(\log N)$ hops
- CAN [Ratnasamy et al. '01]
  - d次元トーラス[0,1]×・・×[0,1]上の領域を各ノードに割り当てる
  - メッセージのあて先はトーラス上の点
  - 平均 $O(N^{1/d})$ hops

まとめ

- Phoenix model on General graphs
  - 大規模 P2P システム
  - Scalable node addressing
  - Scalable message routing
    - 平均ホップ数: $O(\log N)$
    - 平均routing tableサイズ: $O(\log N)$
  - ノードの参加脱退可能
  - Network failureに対応(?)

課題

- 同レベルのcluster間が完全グラフ
  - エッジが比較的多くなる
  - Firewall や NAT が扱えない
  - 同levelのcluster間を2-hopくらいでroutingできれば解決できる?
- Node failure
  - 常に同レベルで隣の区間のノードが生存確認をしていて、死んだようだったら、区間を勝手に引き取ればいい?

TODO

- プロトコルの完成
- 実装
- シミュレーション
- プロトコルの正しさの証明
- 論文