# ChordによるFIBルーティング

金光　永煥（Hidehiro KANEMITSU）

# 初期化処理: IDの設定

- ルーティング処理の本体は，net/gripps/ccn/fibrouting/ChordDHTRouting.java となる．
- CCNMgrが，ChordDHTRoutingを呼ぶ．
- initializeメソッド@CCNMgrから呼ばれるcalcIDメソッド@ChordDHTRroutingにて，ルータIDを決定．そしてrouterMapへ(ルータID, ルータ）の組としてputする@CCNMgr．
- ノードも同様．nodeMapへ(ノードID, ノード）をputする@CCNMgr
- ルータIDはハッシュ値に基づく値で，ノードは1, 2, 3..というように直に数値を割り当てている．

# 初期化処理: Face設定

- buildRouterFacesメソッド@CCNMgrにて，ルータ同士のface設定をする．
  - 隣り合うID同士のルータには，互いのfaceを入れる．
  - 例：B=next(A)であるとき，ルータAのfaceにB, ルータBのfaceにAを入れる．
  - これはinitializeメソッド@CCNMgrで行う．
- とびとびのIDのface設定（Chordのfinger Tableに相当）をする．これは，buildFacesメソッド@ChordDHTRouring　で行う（次ページで説明）．
- buildNodeFacesメソッド@CCNMgrにて，ノードとルータ間のface設定をする（2ページ後で説明）．

# 初期化処理: buildFacesメソッド

- 実体は，buildFingerFacesメソッド @ChordDHTRouting.
- あるルータIDをidとすると，fingerは

$$\mathrm{finger} = \left(\mathrm{id} + \mathrm{dist} \cdot 2^{k-1}\right) \% 2^{p},$$

$$1 \le k \le p,\ \mathrm{dist} = \left\lfloor \frac{2^{p}}{N} \right\rfloor = \left\lfloor \frac{\text{IDの最大値}}{\text{ルータの数}} \right\rfloor$$

であり，fingerに最も近いルータIDを，IDが「id」であるルータのfaceリストへ入れる.

- 互いのfaceを保持させる（faceの対称性）.

# 初期化処理: buildNodeFacesメソッド

- 実体はbuildNeighborRoutersメソッド @BaseRouting.
- ランダムに選ばれたルータを，ノードのrouterMapに入れる（<ルータID, ルータ>の組として）．
- 一方，追加されたルータ側では，当該ノードのfaceを保持する(face_nodeMapへputする）．

# 初期化処理: コンテンツとInterestパケット配備

- コンテンツを生成する(CCNContents).
- コンテンツに対し，size/type/prefix/ハッシュコードを設定.
- コンテンツ集合を，CCNMgr->cMapにて格納.
  - <prefix, CCNContents>の組として.

- buildInterestPacketsメソッド@CCNMgrにてInterestパケットを生成して各ノードのinterestQueueへ入れる.

# FIBの構築

- エッジルータ～ノード間のFIB構築
  - ノードNが持つコンテンツcにおいて，cのハッシュコードを取得（hcode(c)とする）
  - hcode(c)以上で，最も近いIDのルータを探す(rとする）
  - rのFIBにおいて，<prefix(c), faceリスト>へノードNのfaceを追加する.
- ルータ間の転送用のFIB構築
  - routingProcessメソッド@ChordDHTRoutingによってFIBを構築する.
  - この処理が大事なので，routingProcessメソッドの中身を読んで理解しておいてください.