

ネーミング(2)

分散システム

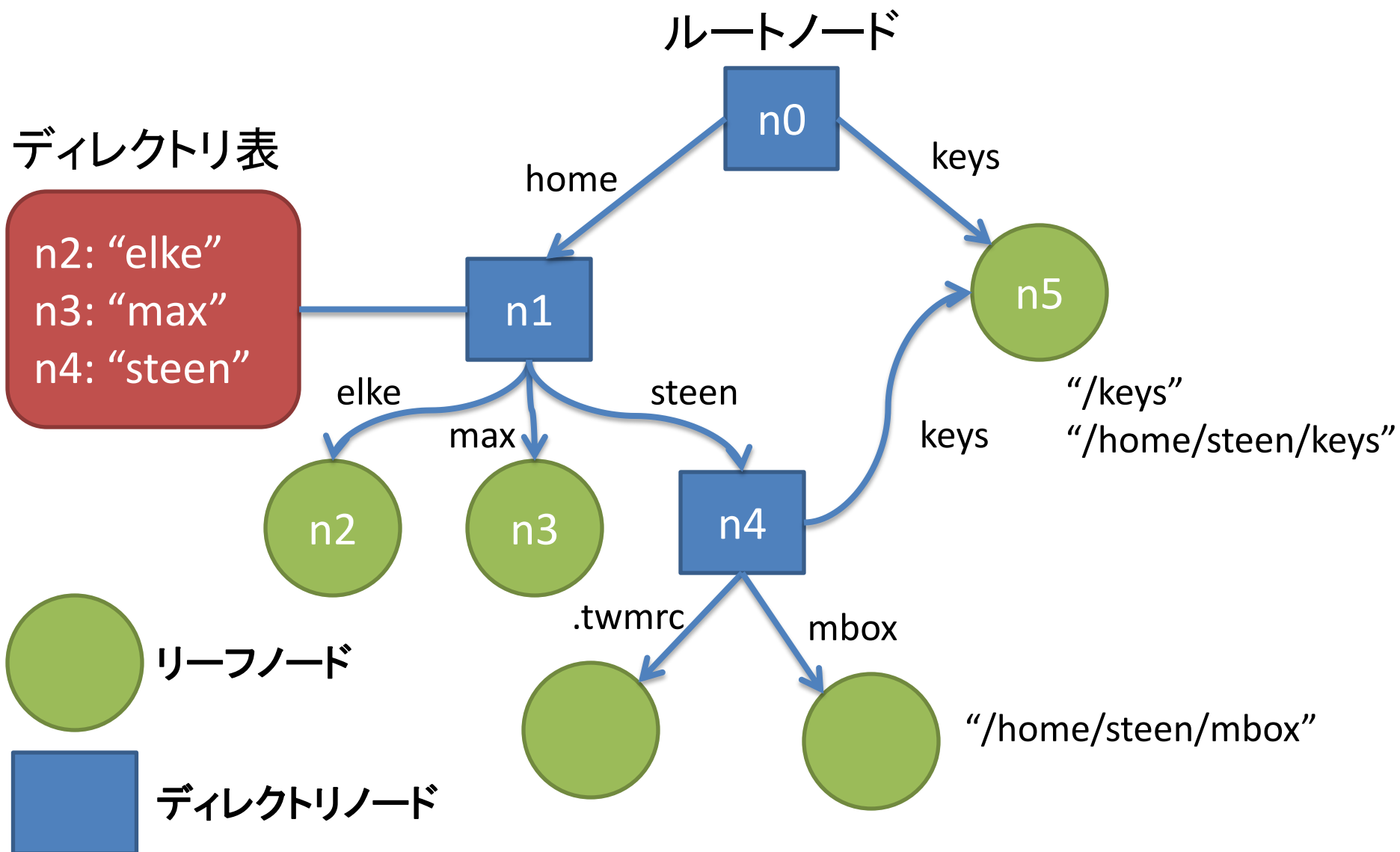
2012年1月31日

建部修見

構造化ネーミング (structured naming)

- フラットネーミングは計算機のため
 - 構造化ネーミングは人のため
 - ファイルの名前付け
 - ホスト名の名前付け
 - 構造化された名前
の名前空間 (name space) は
 - ラベル付けされた有向グラフ (DAG) = 名前グラフ (naming graph)
 - リーフノード = 出て行く枝 (outgoing edge) がない
 - ディレクトリノード
- で表される

単一ルートノードの名前グラフ



パス名

- 名前グラフのパスはノードから枝のラベルの列で表現できる
 - $N: \langle label_1, label_2, \dots, label_n \rangle$
 - N はパスの先頭ノード
 - 絶対パス名 = 先頭ノードがルートノード
 - 相対パス名
- 名前は名前空間により構成される
 - 必ずディレクトリノードからの相対名で定義
- グローバル名 = 必ず同じ実体を指す
- ローカル名 = 状況に依存

ファイルシステムのパス名

- リーフノードはファイル、ディレクトリノードはディレクトリを表す
- ルートディレクトリがある
 - ルートノードで名前グラフは表現される
- ラベルを"/"で区切る
- "/"で始めると絶対パス
 - n0: <home, steen, mbox> → /home/steen/mbox
- 同一ノードが複数のパス名で表現されることも
 - n5は /keys と /home/steen/keys で表される
- ブートブロック = 自動的にメモリにロードされOSをロード、スーパブロック = FS全体の情報、インデックスノード (inode)、ファイルデータブロック
 - ディレクトリは Inode のインデックス番号と名前の対応を保持
 - Inode のインデックス番号はノードの識別子



ブートブロック

スーパブロック

インデックスノード

ファイルデータブロック

ファイルシステムのディスクブロックでの表現

名前解決 (name resolution)

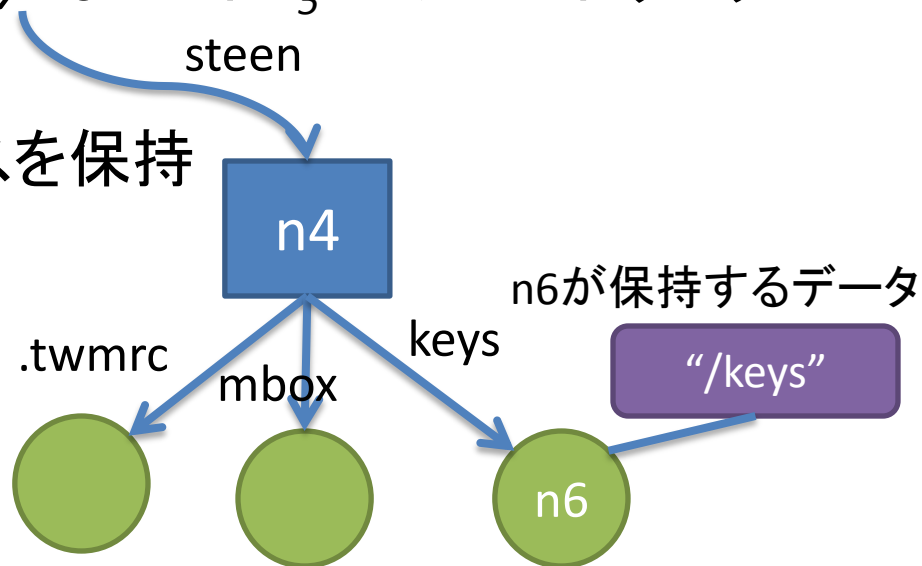
- 名前空間により、名前で情報にアクセスできるようになる
- 名前解決 = パス名によりノードに格納された情報を検索 (look up)
 - $N: \langle label_1, label_2, \dots, label_n \rangle$
 - ノード N から開始
 - ディレクトリ表の $label_1$ を検索、ノードの識別子を返す
 - そのノードのディレクトリ表で $label_2$ を検索、...
 - $label_n$ が参照するノードの内容を返して終了

クローージャ (closure)

- 名前解決には始点が必要
- 始点を知ること＝クローージャメカニズム
 - 暗黙に定義されることも、システムによりさまざま
- 例
 - ファイルシステムのルートディレクトリはスーパーブロックにハードコード (インデックス番号2)
 - 00312044784
 - HOME環境変数

リンク

- エイリアス (alias) = 同一エントリに対する別名
 - ハードリンク
 - 複数の絶対パス名で同一ノードを表現
 - `/keys`と`/home/steen/keys`はノード n_5 へのハードリンク
 - シンボリックリンク
 - リーフノードで絶対パスを保持



マウント

- 異なる名前空間を透明にマージ
 - 名前空間のマウント
- 例: ファイルシステムのマウント
- マウントポイント = 異なる名前空間 (外部名前空間) のディレクトリノードの識別子を保持
 - アクセスプロトコル
 - サーバ名
 - 外部名前空間のディレクトリノード (マウンティングポイント)
- それぞれの名前は解決できる必要がある
 - URLでの表現
 - `nfs://flits.cs.vu.nl//home/steen`

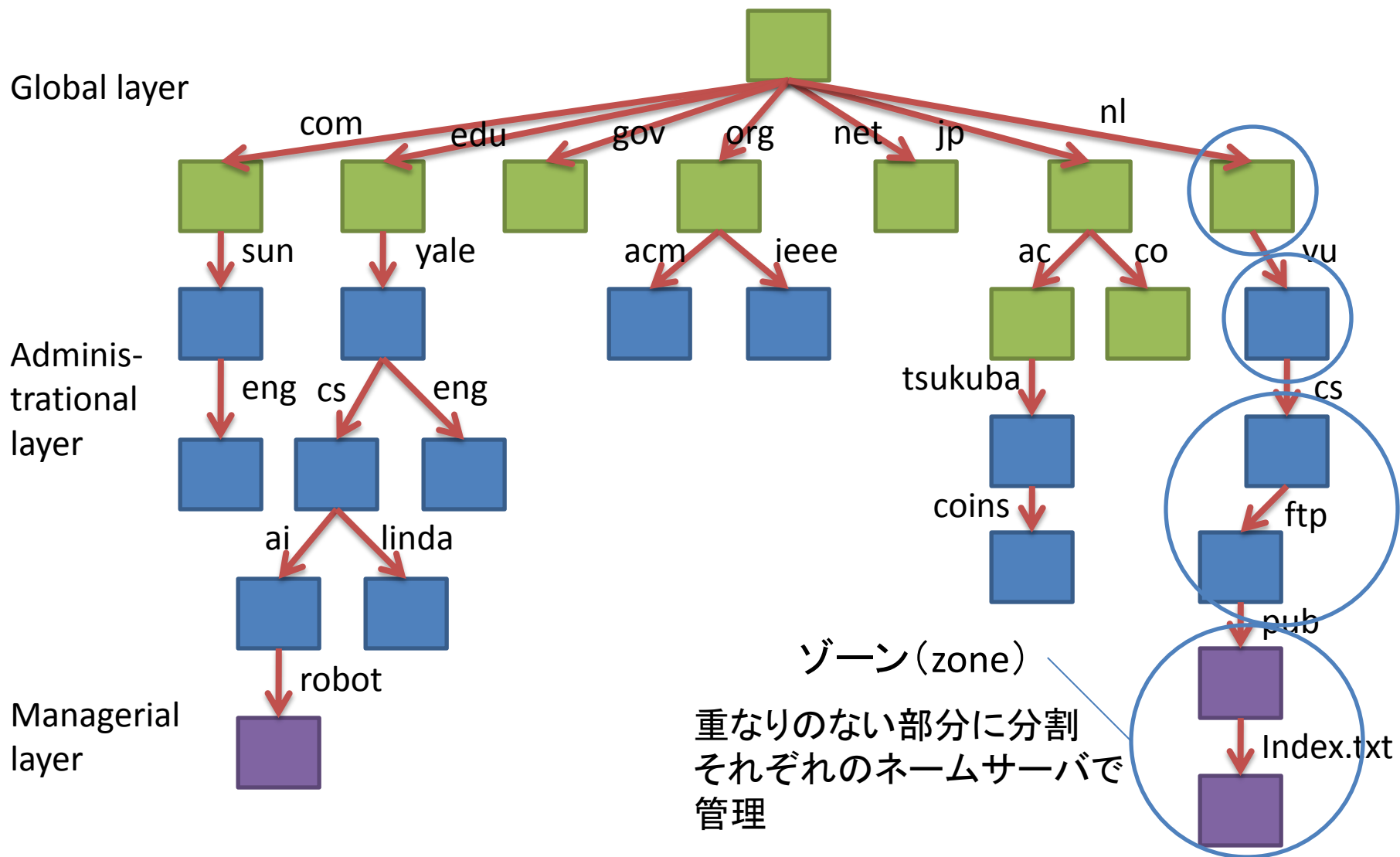
名前空間の実装

- ネーミングサービス = 名前の追加、消去、検索 (look up) を行う
- ネームサーバ (name server)
 - 単一ネームサーバ
 - 分散ネームサーバ

名前空間の分散

- 大規模(世界規模)で分散するシステムの名前空間はしばしば階層構造となる
- 単一ルートノードを仮定
- 三種類のレイヤに分類される
 - グローバルレイヤ(global layer)
 - ルートノードとその子ノード。滅多に変わらない
 - アドミニストレーションレイヤ(administrational layer)
 - 組織や管理ユニット。比較的変わらない
 - マネージャレイヤ(managerial layer)
 - ホスト、共有ファイル
 - よく変わる。管理者だけではなく利用者也変更

DNS名前空間の分割例



性能と可用性

- グローバルレイヤ
 - 高可用性が必須
 - ほとんど変更されないため、複製、クライアントキャッシュが有効
 - 応答時間よりスループットが重要
 - 変更は直ちに有効にならなくてもよい
- アドミニストレーションレイヤ
 - その組織にとってのみ重要
 - 複製、クライアントキャッシュが有効
 - 応答時間(数ミリ秒)が重要
 - 更新もそれなりに早いことが期待される
- マネージャレイヤ
 - 可用性より性能が重要

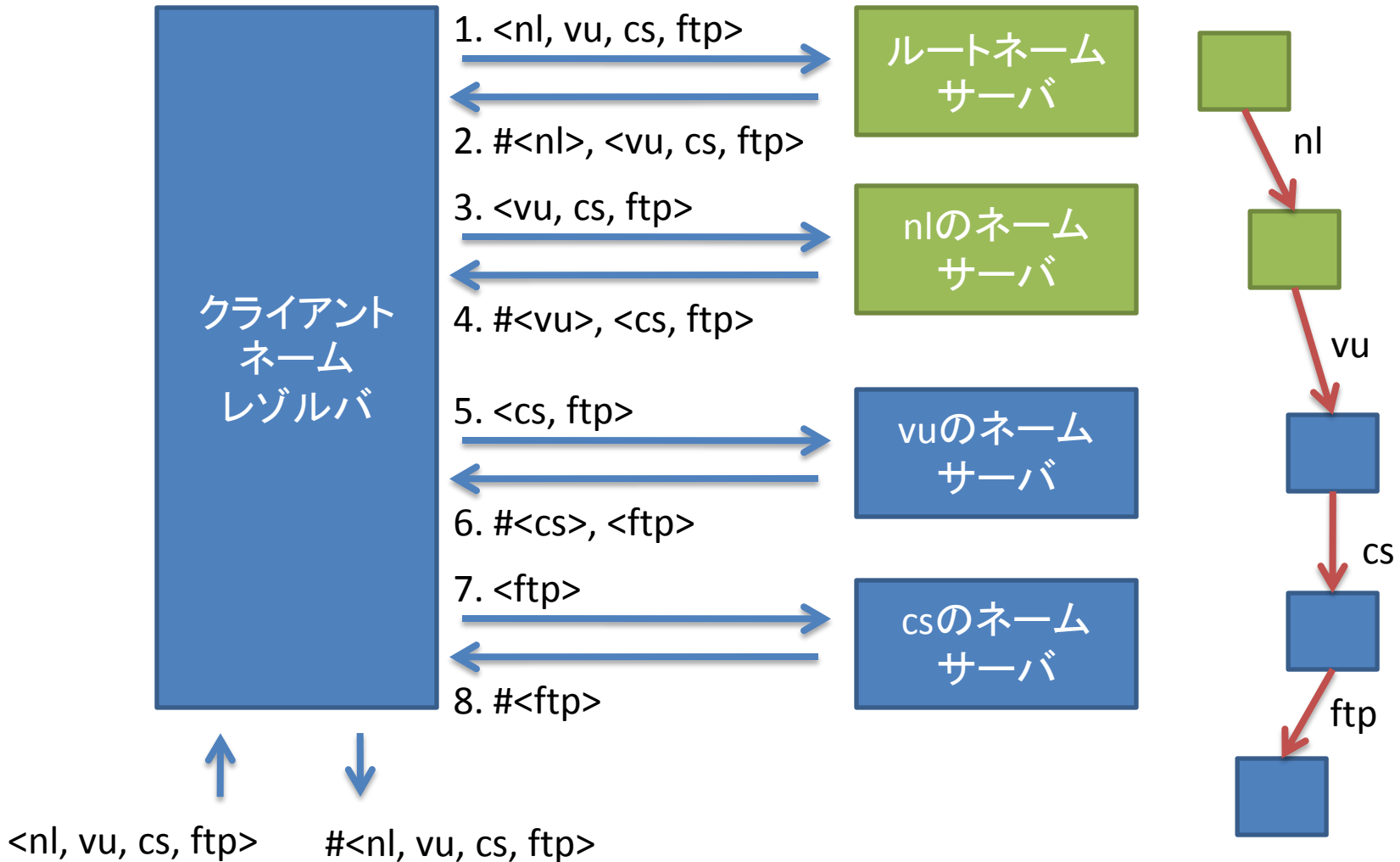
レイヤ別名前サーバの比較

要件	グローバル	アドミニストレーション	マネージャ
ネットワークの広がり	世界規模	組織規模	学部規模
総ノード数	少数	多数	膨大
Look upの応答時間	秒	ミリ秒	直ちに
更新の伝搬	遅延可 (lazy)	直ちに	直ちに
複製サーバ数	多数	なしor少数	なし
クライアントキャッシュ	有効	有効	しばしば有効

名前解決の実装

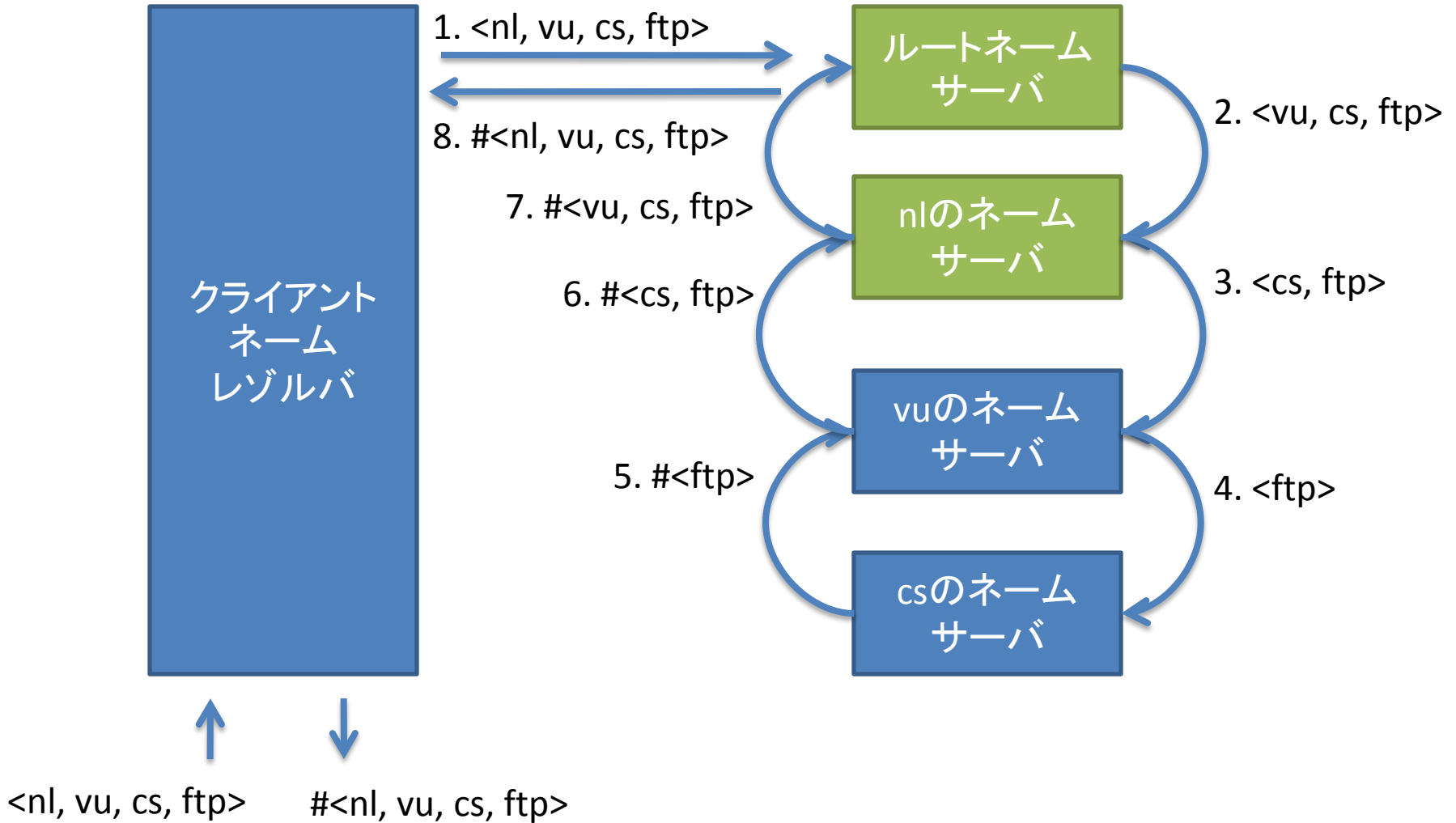
- 名前空間は複数のネームサーバに分散
- 簡単のため、複製、クライアントキャッシュは考えない
- クライアントはローカルのネームレゾルバ (name resolver) にアクセス
 - 反復名前解決 (iterative name resolution)
 - 再帰名前解決 (recursive name resolution)

反復名前解決



#<name>はnameの名前サーバのアドレス

再帰名前解決



再帰名前解決について

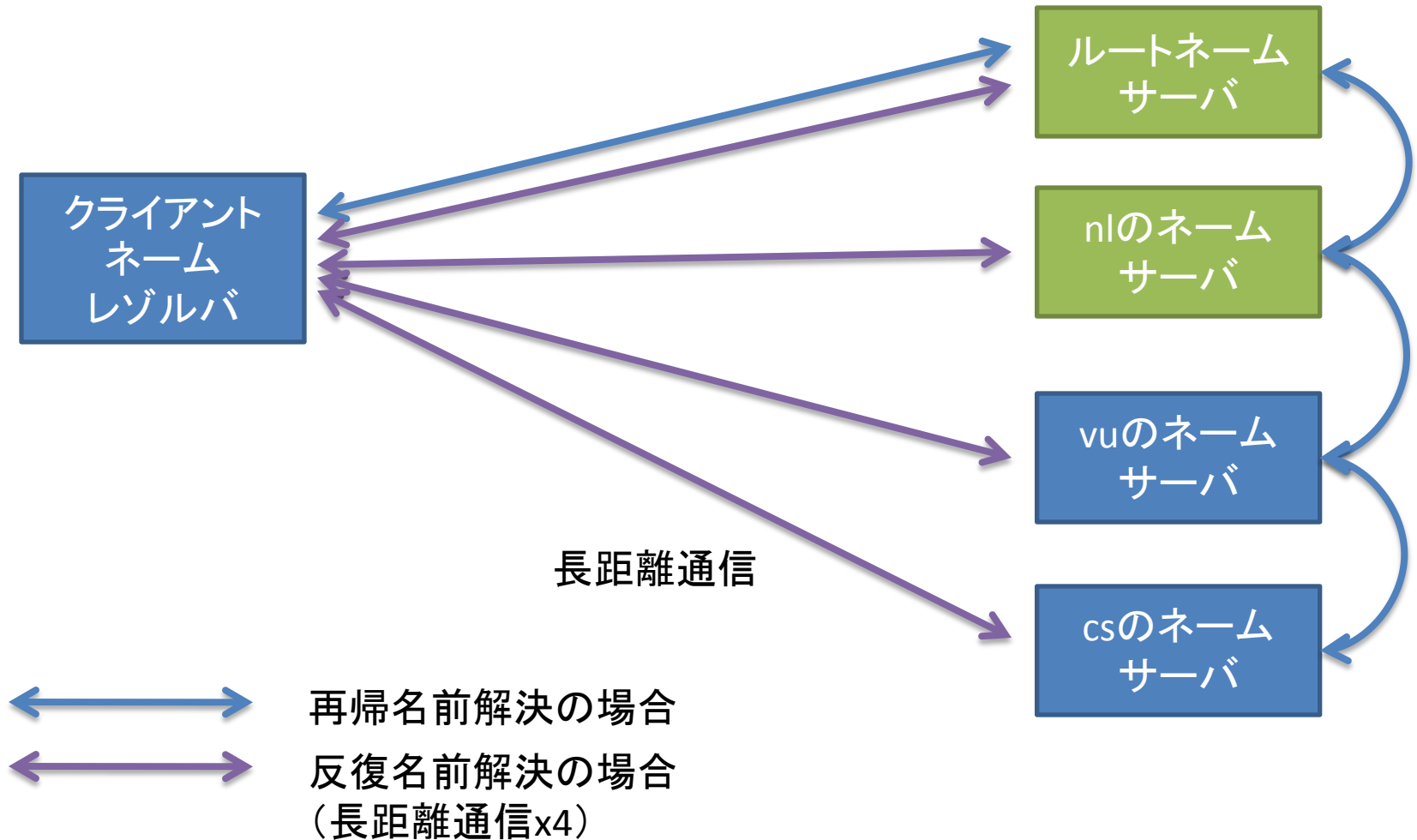
- 欠点
 - より性能要件が厳しくなる
 - グローバルレイヤでは高負荷になりすぎる
- 利点
 - キャッシュの効果が大きい
 - 通信コストを下げられる

再帰的名前解決におけるキャッシュ

サーバ	解決する名前	検索対象	子サーバに渡す	受取る、キャッシュする	応答
cs	<ftp>	#<ftp>	-	-	#<ftp>
vu	<cs, ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
nl	<vu, cs, ftp>	#<vu>	<cs, ftp>	#<cs> #<cs, ftp>	#<vu> #<vu, cs> #<vu, cs, ftp>
root	<nl, vu, cs, ftp>	#<nl>	<vu, cs, ftp>	#<vu> #<vu, cs> #<vu, cs, ftp>	#<nl> #<nl, vu> #<nl, vu, cs> #<nl, vu, cs, ftp>

反復名前解決ではクライアントごとにキャッシュ(効率悪い)
ローカルな中間ネームサーバでキャッシュし、共有する

通信コストの減少



まとめ

- 構造化された名前は名前空間で構成される
 - 名前グラフで表現
 - ディレクトリノードとリーフノード
 - 枝にはラベルが付けられる
 - しばしば単一ルート of DAG
- パス名で表現
- 名前解決は名前グラフを走査
- 分散サーバ構成では反復名前解決と再帰名前解決