KUPF フレームワークの実装と応用

垣内 正年 †　森島 直人 †　砂原 秀樹 †

インターネットにおけるサービスの多様化と高度化により，経路制御やファイアウォールなどにおいて，ポリシを反映する複雑な処理が必要となっている．このような処理はあらかじめ定義された規則に基づいて対象となるデータを分類するという手続きをともなう．筆者らは，このようなデータの分類における問題点を解明するために手続きを一般化し，KUPF アーキテクチャとして設計・実装してきた．本稿では，KUPF における規則表現のデータ構造，規則の管理構造，データを分類するための KUPF フレームワーク，およびその API についての概略を示し，その設計・実装およびその適用例をまとめる．さらに，今後の KUPF の方向性について議論する．

Implementation and Application of KUPF Framework

KAKIUCHI Masatoshi†　MORISHIMA Naoto†　SUNAHARA Hideki†

Diversity and advancement of services on the Internet need more complex way to control route and to filter packets, which reflects policy. These operations require procedures, which classify target data based on defined rules. We have proposed and implemented KUPF architecture to expose and resolve problem of data classification. In this paper, we describe data structure for rule expression and management structure for rules on KUPF, and summarize KUPF framework classifying data with API and application, to explain its design and implementation. Furthermore, we summarize its design, implementation and application, then discuss about the plan of future KUPF.

1　まえがき

近年の複雑化するネットワークにおいて，高度化・多様化するサービスを実現するためには，ルータ，アプリケーションサーバ等のサービスを提供するネットワーク機器は，複数のパラメータから構成される複雑なフィルタルールに基づいてサービスを処理することが要求される．また，複数のサービスが並行して運用されるため，1 つのフィルタルールが同時に 1 つの入力データに適合する場合がある．この場合，ネットワーク機器はサービス毎の競合を適切に解決し，入力データに適用する処理を決定する必要がある．しかし，サービスのポリシ毎に競合関係は異なるため，フィルタルールのみに基づいて一意に適用する処理を選択できない．

これらの問題に対して，くまプロジェクト[*1]は，KUMA's Universal Parameter Filter (KUPF) [1, 2] アーキテクチャを提案し，実装してきた．KUPF は，複数パラメータから構成される複雑なフィルタルールが処理可能なパラメータフィルタを実現した．さらに，KUPF-VR [3] は，多次元空間検索手法を用いることで KUPF の検索速度高速化を実現した．

本稿では，KUPF における規則表現のデータ構造，規則の管理構造，データを分類するための KUPF フレームワーク，およびその API についての概略を示し，その設計・実装をまとめる．さらに，このフレームワークの適用例として実装した，IP パケット入力フィルタについて説明する．最後に，今後の KUPF の方向性について議論する．

2　KUPF の構成

一般的に，入力データを処理する場合，入力データに対応する処理を選択するための規則およびその処理自体の記述が必要となる．KUPF では，前者の規則をフィルタルール，後者の記述をアクション，さらにその対を分類レコードと呼ぶ．KUPF モデルは，分類レコードを選択する機構であるパラメータフィ

† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

*1 くまプロジェクト – http://www.kuma-project.net/

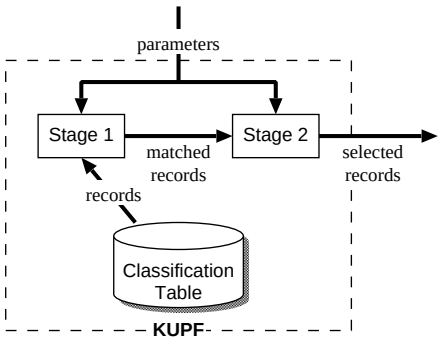

図 1　KUPF 概要

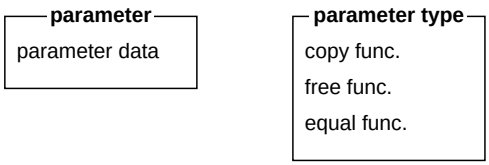

図 2　パラメータとパラメータ型

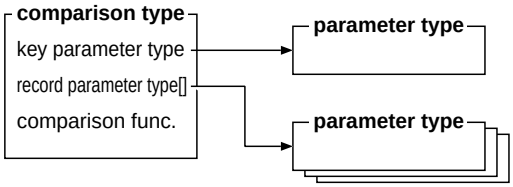

図 3　比較型

ルタを，サービス非依存処理とサービス依存処理の 2 段階に分割する．パラメータフィルタを 2 段階に分割したことで，以下が実現できた．第 1 ステージは，個々の分類レコードに対してフィルタルールの条件が入力データを満たすか否かの判定をするのみで，サービスに依存せずに処理を行うため，汎用的に設計・実装できる．第 2 ステージは，第 1 ステージの結果からサービス毎のポリシに従って，入力データに適用するアクションを持つ分類レコードを選択する．サービス依存処理は第 2 ステージで完結しているため，第 2 ステージ置き換えのみによりパラメータフィルタを異なるサービスに応用できる．

図 1 に KUPF の概要を示す．分類レコードは，分類表で管理される．第 1 ステージは，入力データのパラメータを入力すると，パラメータに一致するフィルタを持つ分類レコードを出力する．第 2 ステージは，入力データのパラメータと分類レコードを入力すると，サービス毎のポリシに従って分類レコードを選択し，出力する．

2.1　パラメータと比較

ポート番号，IP アドレス等の各値をパラメータと呼ぶ．KUPF では，これらの値を符号なし整数，固定長ビット列等の基本的なパラメータ型で表現し，ポート番号，IP アドレス等の特定の意味を持つ型を用いない．

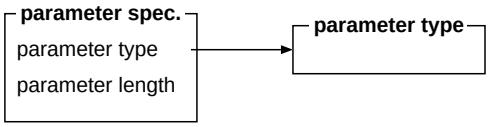

図 4　パラメータ仕様

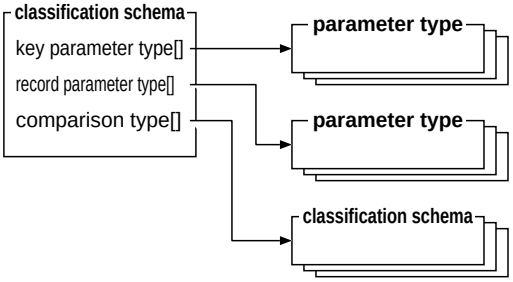

図 5　分類スキーマ

図 2 にパラメータとパラメータ型のデータ構造を示す．パラメータは，パラメータの値を表現するデータで構成する．パラメータは内部にパラメータ型に関する情報を保持しないため，パラメータだけでは特定の値を指し示さない．図中には示さないが，パラメータ型によって異なるバイト長のデータの取り扱いを容易にするため，パラメータないにパラメータのバイト長を保持する．パラメータ型は，パラメータを複製・解放する関数，2 つのパラメータの同一性を判定する関数で構成する．パラメータはパラメータ型と関連付けることで，複製・解放・同一性判定を行える．

あるパラメータとパラメータ群との関係を比較演算として定義した物を比較型と呼ぶ．比較型はパラメータの型，パラメータ群の型，比較手続きによって構成される．例えば，整数が範囲内に含まれていることを表す比較型は，

- 整数のパラメータ型（符号なし整数）
- 範囲の上限と下限のパラメータ型（それぞれ符号なし整数）
- 比較手続き（下限 $\leq$ 整数 $\leq$ 上限）

と記述できる．

図 3 に比較型のデータ構造を示す．比較型は，キーパラメータ（前記例の整数）のパラメータ型，レコードパラメータ群（前記例の上限・下限）のパラメータ型配列，比較手続きの比較関数で構成する．

2.2　パラメータ仕様と分類スキーマ

ビット列が IPv4 アドレスなどの意味を持つには，32 ビットという長さの属性が必要である．このパラメータ型に長さを加えた物をパラメータ仕様と呼ぶ．

表 1　分類スキーマの例

	終点アドレス	プロトコル番号	終点ポート番号の範囲
キーパラメータ仕様群	32bit 長ビット列	符号なし整数	符号なし整数
レコードメータ仕様群	32bit 長ビット列	符号なし整数	符号なし整数　符号なし整数
比較型	ビット列比較	符号なし整数比較	符号なし整数範囲比較

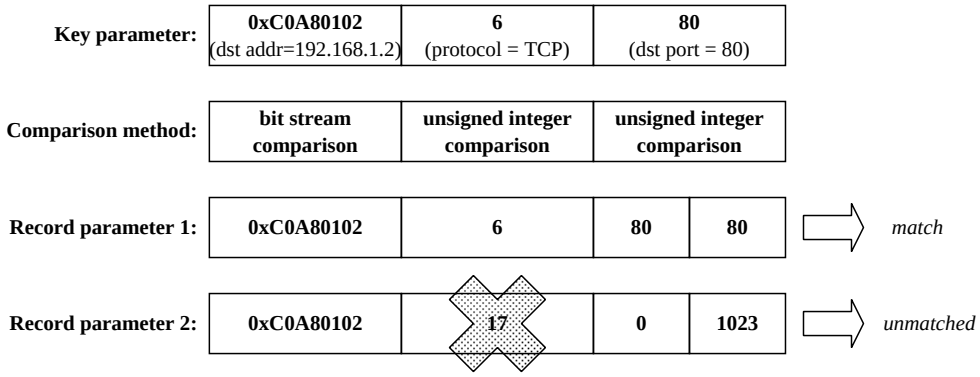

図 6　パラメータ比較

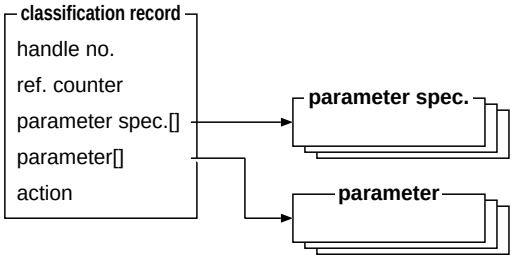

図 7　分類レコード

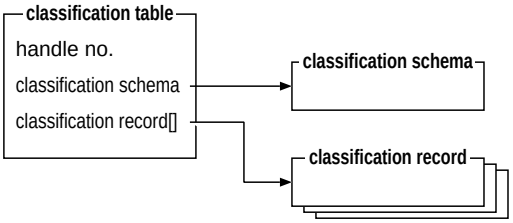

図 8　分類表

図 4 にパラメータ仕様のデータ構造を示す.

具体的なデータがある条件を満たしているか検証するためには，入力データのパラメータ仕様，条件となるパラメータ群の仕様，およびそれらを比較するための比較型が必要となる．複数の入力データに対する条件の集合，すなわち入力データ毎の上記 3 要素の集合を分類スキーマと呼ぶ．図 5 に分類スキーマのデータ構造を示す.

例えば，IPv4 パケットを終点アドレス，プロトコル番号，終点ポート番号の範囲の 3 条件で分類する場合，分類スキーマは表 1 で表現できる．分類スキーマに示された全ての条件についてキーパラメータ群とレコードパラメータ群が条件を満たすとき，キーパラメータ群とレコードパラメータ群は条件を満たすと判定する．(図 6)

分類スキーマを構成するパラメータ仕様と比較型は KUPF フレームワーク利用者が新たに定義することで拡張可能である．そのため，KUPF は任意の分類スキーマに柔軟に対応可能である.

2.3　分類レコード

分類レコードの構造を図 7 に示す．条件となるパラメータ群とアクション以外に，ハンドル番号，参照カウンタを保持する．ハンドル番号は分類表内で分類レコードを識別する．分類表に分類レコードを保持する間等，分類レコードを参照するとき参照カウンタを 1 増やす．図に示したものの他に，分類レコードの使用回数を保持するカウンタを持つ.

2.4　分類表

分類表の構造を図 8 に示す．分類スキーマ，分類レコード群以外に分類表を識別するハンドル番号を保持する．図には示したものの他に，フレームワーク利用者が名付けた名前で分類表を識別できる分類表名・ユニット番号，分類回数，レコード追加・削除回数等の統計情報を保持する.

3　プログラムインタフェイス

KUPF プログラムインタフェイスは，主に表 2 のパラメータ操作，分類表操作，分類レコード操作，分類操作の 4 種類で構成される．以下にこれらのインタフェイスの概略を説明する.

表 2　プログラムインタフェイス

パラメータ操作	
パラメータ作成	kupf_*xxx*_alloc(**パラメータ仕様，値，...**)
パラメータ解放	kupf_paramdata_free(**パラメータ仕様，パラメータ**)
分類表操作	
分類表作成	kupf_comp_table_alloc(**名前，ユニット番号，キーパラメータ仕様群，レコードパラメータ仕様群，比較型群**)
分類表削除	kupf_comp_table_free(**分類表**)
分類レコード操作	
分類レコード作成	kupf_entry_alloc(**レコードパラメータ仕様群，パラメータ群，アクション**)
分類レコード保持	kupf_entry_retain(**分類表**)
分類レコード解放	kupf_entry_free(**分類表**)
分類レコード追加	kupf_comp_table_entry_add(**分類表，分類レコード**)
分類レコード削除	kupf_comp_table_entry_remove(**分類表，分類レコード**)
分類レコード削除	kupf_comp_table_entry_remove_handle(**分類表，分類レコードハンドル番号**)
分類レコード全削除	kupf_comp_table_entry_flush(**分類表**)
分類操作	
第 1 ステージ分類操作	kupf_comp_table_collect(**分類表，キーパラメータ群**)
第 2 ステージ分類操作	kupf_select_*xxx*(**分類表，分類レコード群，...**)

表 3　パラメータ型

型名	型識別名	作成関数
符号なし整数	KUPF_PT_UINT	kupf_uint_alloc(パラメータ仕様, 整数値)
符号付き整数	KUPF_PT_SINT	kupf_sint_alloc(パラメータ仕様, 整数値)
固定長ビット列	KUPF_PT_FLBITS	kupf_flbits_alloc(パラメータ仕様, ビット値)
可変長ビット列	KUPF_PT_VLVITS	kupf_vlbits_alloc(パラメータ仕様, ビット長, ビット値)

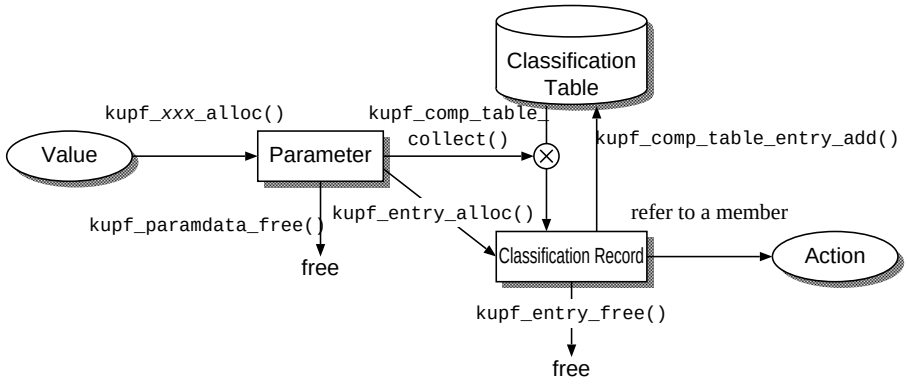

図 9　パラメータの変遷

3.1　パラメータ操作

パラメータ操作は，パラメータ作成・解放を行う．パラメータ作成は，パラメータのメモリを割当て，値を設定する．パラメータ作成操作は，パラメータ毎に別の関数を呼び出す．kupf_*xxx*_alloc の *xxx* はパラメータ型となる．例として，符号なし整数のパラメータ作成は，パラメータ仕様と整数値を引数として kupf_uint_alloc を呼び出す．可変長ビット列のパラメータ作成は，パラメータ仕様とビット長，ビット値を引数として kupf_vlbits_alloc を呼び出す．パラメータ解放は，パラメータに割当てられたメモリを解放する．

表 3 に KUPF フレームワークが用意するパラメータ型を示す．フレームワーク利用者はパラメータ作成関数とパラメータ型を作成することで，新たなパラメータ型をこの表のパラメータ型と同様にフレームワークで利用可能である．表 9 はフレームワークないにおけるパラメータの変遷を示している．パラメータ作成関数以外は，パラメータ型に依存しないインタフェイスであるため，新たなパラメータ型を

定義してもパラメータ作成後のインタフェイスを変更する必要がない．

3.2 分類表操作

分類表操作は，分類表作成・削除を行う．分類表作成は，分類表のメモリを割当て，分類スキーマ，その他の情報を設定する．分類表作成時にキーパラメータ仕様群，レコードパラメータ仕様群，比較型群を用いて分類スキーマを設定するため，他の呼び出しではこれらの情報を与える必要がない．分類表削除は，分類表が持つ全てのメモリ割当てを解放する．

表 3 に KUPF フレームワークが用意する主な比較型を示す．各比較型は，キーパラメータが定められた手続きでレコードパラメータ群の条件を満たすか否か判定する．フレームワーク利用者は比較型を作成することで，新たな比較型をこの表の比較型と同様にフレームワークで利用可能である．

3.3 分類レコード操作

分類レコード操作は，分類レコード作成・保持・解放，分類表への分類レコードの追加，分類表からの分類レコード削除を行う．

分類レコード作成は，分類レコードのメモリを割当て，レコードパラメータ仕様群，パラメータ群，アクションを設定する．分類表保持・分類表解放は，それぞれ分類レコードの参照カウンタを 1 増加・減少させる．減少させた結果，参照カウンタが 0 になれば，分類レコードのメモリ割当てを解放する．分類表等のモジュールは，分類レコードを参照する間分類レコードの参照カウンタを 1 増やす．参照するモジュールが存在しない分類レコードは参照カウンタが 0 となるため，メモリ割当てが解放される．この参照カウンタの仕組みにより，分類レコードを複製せずに複数のモジュールが同一の分類レコードを参照できる．

分類レコード追加は，分類表に分類レコードを追加し，分類表内で重複しない番号をハンドル番号として分類レコードに割り当てる．返値として割り当てたハンドル番号を返す．分類レコード削除・分類レコード全削除は，分類レコードから分類レコードを削除する．パラメータの種類は分類レコード内に隠蔽されるため，分類スキーマの種類にかかわらず分類レコード操作のインタフェイスは同一である．

3.4 分類操作

分類操作は，第 1 ステージ分類操作と第 2 ステージ分類操作に分かれる．第 1 ステージ分類操作は，分類表とキーパラメータ群を与えると，返値として，分類表から，キーパラメータ群を条件として満たすのレコードパラメータ群を持つ全分類レコードが返る．第 2 ステージ分類操作は，サービス毎に依存するため個別実装が必要である．KUPF は，サービスに依存しない実装として最善一致 `kupf_select_best` を提供する．最善一致による第 2 ステージ分類操作は，引数として分類表，第 1 ステージの結果，パラメータ間の優先順位を与える．優先順位は，最善一致を判定するパラメータの順序を指定する．返値として，最善一致の分類レコードを返す．比較型は分類表が保持するため，分類スキーマの種類にかかわらず分類操作のインタフェイスは同一である．

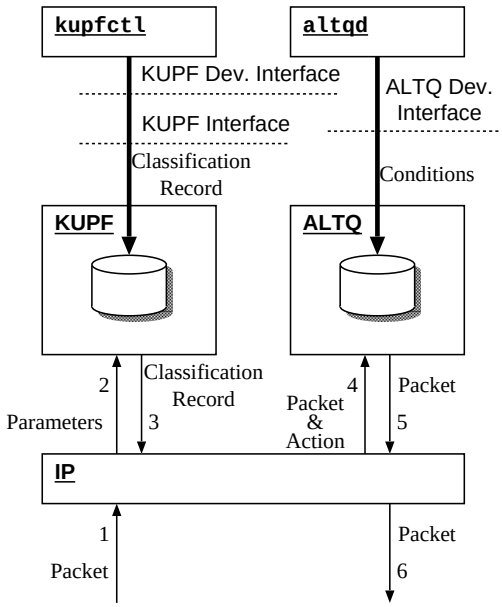

図 10　KUPF を用いた IP 入力フィルタ

4 IP パケット入力フィルタへの適用

KUPF フレームワークの適用例として，IP パケット入力フィルタを実装した．KUPF により得られたアクションで，Diffserv サービスを提供する ALTQ [4] を制御した．このシステムは図 10 に示すように，IP, KUPF, ALTQ のカーネル空間モジュールと `kupfctl`, `altqd` のユーザ空間プログラムから構成される．`kupfctl` は，デバイスインタフェイス経由で KUPF 分類レコード操作を呼び出し，分類レコードをカーネル内の KUPF に登録する．`altqd` は，ALTQ デバイスインタフェイスを呼び出し，Diffserv 制御の情報をカーネル内の ALTQ に登録する．IP パケットがこのシステムに入力されると，以下の手順で処理する．

1　IP にパケットが入力される

2　IP はパケットからパラメータ群を取り出し，

表 4　比較型

比較型名	型識別名	用途
符号なし整数比較	KUPF_CT_UINT_MATCH	プロトコル番号比較
固定長ビット列先頭比較	KUPF_CT_FLBITS_PREFIX_MATCH	IP アドレスプレフィックス比較
固定長ビット列部分比較	KUPF_CT_FLBITS_MASK_MATCH	IP ToS フィールドマスク付き比較

KUPF に送る

3　KUPF はパラメータに適合する分類レコードを IP に返す

4　IP はパケットと分類レコードが示すアクションを ALTQ に送る

5　ALTQ はアクションに対応する Diffserv 制御情報をパケットに適用し，IP に返す

6　IP は ALTQ で処理されたパケットを出力する

このように，KUPF を用いることでフィルタ処理がパケット制御から分離されるため，制御機構とは独立したフィルタ部分の高機能化が可能になる．また，複数の制御機構のフィルタ処理を KUPF で統合的に行うことで，フィルタ処理の効率化が計れる．

5　KUPF の発展

KUPF フレームワークはパラメータフィルタを抽象化したモデルで実装しているため，汎用性が高い反面，処理速度性能が犠牲となっている．処理速度性能を向上させる手法としては，KUPF 各ステージ毎の高速化，ステージ間連携による高速化，ハードウェア処理導入による高速化がある．

第 1 ステージ高速化は，KUPF-VR [3] において多次元空間検索手法を用いることで検索速度高速化を実現した．第 2 ステージはサービス毎に個別実装となるため，KUPF フレームワークとして第 2 ステージ単体での高速化はできない．

これに対し，第 2 ステージの呼び出しを先読み的に実行することで，ステージ間連携による高速化の余地がある．例えば，最善一致において，ある分類レコードが入力データの条件を満たす場合，この分類レコードより一致の度合いが低い分類レコード検索の省略が考えられる．さらに，複数のサービス処理機構の個別のパラメータフィルタを KUPF に集中することで，重複する処理をまとめて効率化が計れる．

KUPF-VR においては仮想包囲矩形導入によりパラメータ比較が単純化されている．そのため，ハードウェア処理導入はパラメータ型の柔軟性が高いにも関わらず比較的容易だと考えられる．

今後，これらの手法を組み合わせることで，KUPF の汎用性を維持しつつ高速性の向上に取り組む予定である．

6　むすび

本稿では，KUPF における規則表現のデータ構造，規則の管理構造，データを分類するための KUPF フレームワーク，およびその API についての概略を示した．また，適用例として IP フィルタ実装について説明した．KUPF の 2 段階処理モデルの利用により，KUPF フレームワークは，複数パラメータからなる複雑なポリシを扱うパラメータフィルタを汎用的なインタフェイスで実現した．また，パラメータフィルタとサービス処理機構を分離することで，各機能が独立して拡張可能となった．今後，KUPF の汎用性，高機能性とともに高速性の向上に取り組む予定である．

参考文献

[1] 垣内正年, 森島直人, 宇田仁, 中村豊, 砂原秀樹: 汎用的なパラメータフィルタの設計と実装, 電子情報通信学会技術研究報告, Vol. 102, No. 143, pp. 7–14 (2002).

[2] Kakiuchi, M., Morishima, N., Nakamura, Y., Fujikawa, K. and Sunahara, H.: KUPF: 2-Phase Selection Model of Classification Records, *Proc. 2003 Symposium on Applications and the Internet Workshops*, Orlando, United States, pp. 262–265 (2003).

[3] 垣内正年, 森島直人, 砂原秀樹: 仮想包囲矩形に基づくパラメータフィルタの設計と実装, 電子情報通信学会和文論文誌. （条件付き採録）.

[4] Cho, K.: A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers, *Proc. USENIX 1998 Annual Technical Conference*, New Orleans, United States (1998).