

卒業論文 2023 年度（令和 5 年度）

Linux Netfilter の SRv6 への統合

慶應義塾大学 環境情報学部
澤田 開杜

Linux Netfilter の SRv6 への統合

本論文では、Linux の持つパケット操作機能である netfilter を、トラフィック制御技術の 1 つである SRv6 に統合する手法を提案する。昨今のデータセンタネットワークでは、汎用的なサーバや仮想マシン、コンテナ技術を使ってネットワークの機能 (NF) を仮想化する技術が一般化してきている。一連のルールに沿って NF を適用することを サービスファンクションチェイニング (SFC) という。SFC にデプロイされる NF はサービスファンクション (SF) と呼ばれ、SFC ではあるパケットを任意の順番で SF へ通過させる必要がある。従来のパケットルーティングでは、あるパケットを任意の順番で指定したノードを通過させる、ということとはできない。よって、SFC の実現のためには従来のパケットルーティングとは別の経路制御機構が必要である。SFC を実現できる経路制御技術の 1 つに Segment Routing over IPv6 (SRv6) という技術がある。SRv6 では、SRv6 header と呼ばれるヘッダで IP パケットをカプセル化する。また、SRv6 header にはパケットが通過するノードが順番に含まれる。これによって、IP 的なベストパスに関係なくパケットが通過するノードを指定可能であるから、任意のルールに従って SF を通る順番を指定できる。また、SRv6 はトラフィック制御だけでなく、トランジットするパケットに対して特定の操作を適用でき、この特定の操作の種類のことをビヘイビアという。

Linux カーネルには netfilter というパケット処理フレームワークが実装されている。netfilter を使うことで、パケットのフィルタリングや NAT, NAPT, その他のパケットマングリング操作を適用できる。しかし、SRv6 は SRv6 header でカプセル化されているため、カプセル化されている内部のパケットに対して netfilter を適用できない。そこで、本論文では End.AN.NF という新しい SRv6 ビヘイビアを提案する。End.AN.NF は Linux netfilter を Linux の SRv6 ルーティングインフラストラクチャへ統合する事ができる。End.AN.NF は、SRv6 を利用した SFC 環境において、Linux netfilter を SRv6 に対応した SF として扱えるようにする。End.AN.NF を利用する際、netfilter を利用して作成されたアプリケーションの実装を変える必要はなく、End.AN.NF は SRv6 の基本処理である End ビヘイビアを実行しながら、SRv6 でカプセル化された内部のパケットへ netfilter を適用できる。さらに、End.AN.NF は、パケットバッファにマークを付けることができる。したがって、netfilter を利用して作成されたアプリケーションは End.AN.NF がパケットバッファに付与したマークを照合することで、適用するルールを変更できる。本論文では、End.AN.NF を Linux カーネルに実装し、その性能評価を行った。計測の結果、End.AN.NF は End.DT4 と H.Encaps を使って SRv6 でカプセル化された内部パケットに netfilter を適用する方法に比べ、27% 高いスループット、及び 3.0 マイクロ秒低いレイテンシを実現した。

キーワード:

1. Service Function Chaining 2. Segment Routing 3. SRv6

慶應義塾大学 環境情報学部
澤田 開杜

Integrating Netfilter into SRv6 Routing Infrastructure of Linux as an SR-Aware Network Function

This paper proposes a method for integrating netfilter, a packet manipulation feature of Linux, with Segment Routing over IPv6 (SRv6), a traffic control technology. In modern data center networks, virtualization of Network Functions (NFs) using generic servers, virtual machines, and container technologies has become widespread. The application of NFs according to a set of rules is called Service Function Chaining (SFC). The NFs used in SFC are called Service Functions (SFs), and SFC requires that certain packets be routed through SFs in an arbitrary order. Traditional packet routing cannot route a packet through specific nodes in an arbitrary order. Therefore, a routing mechanism different from traditional packet routing is required to implement SFC. Segment Routing over IPv6 (SRv6) is one of the path control technologies that can implement SFC. In SRv6, IP packets are encapsulated with a header called the SRv6 header, which sequentially lists the nodes that the packet will traverse. This allows the specification of the nodes that a packet will traverse, regardless of the IP's best path, and thus the order of SF traversal can be arbitrary. In addition to controlling traffic, SRv6 can also apply specific operations to transit packets, these specific operations being called "behaviors".

The Linux kernel implements a packet processing framework called netfilter, which allows packet filtering, NAT, NAPT, and other packet mangling operations. However, netfilter cannot be applied to packets encapsulated in an SRv6 header. Therefore, this paper introduces a new SRv6 behavior called End.AN.NF. End.AN.NF allows Linux netfilter to be integrated into the Linux SRv6 routing infrastructure. It allows Linux netfilter to be treated as an SRv6 compatible SF in an SFC environment using SRv6. When using End.AN.NF, there's no need to modify the implementation of applications built with netfilter. End.AN.NF can apply netfilter to packets encapsulated in SRv6 while performing the basic processing of SRv6, known as End behavior. In addition, End.AN.NF can mark packet buffers, allowing applications built with netfilter to change the rules they apply by checking the mark End.AN.NF adds to the packet buffers. This paper implements End.AN.NF in the Linux kernel and evaluates its performance. The results show that End.AN.NF achieves 27% higher throughput and 3.0 microseconds lower latency compared to the method of applying netfilter to packets encapsulated in SRv6 using End.DT4 and H.Encaps.

Keywords :

1. Service Function Chaining 2. Segment Routing 3. SRv6

Keio University Bachelor of Arts in Environment and Information Studies
Kaito Sawada

目次

第1章	序論	1
1.1	本研究の概要	1
1.2	本論文の目的と構成	3
第2章	前提知識	4
2.1	Service Function Chaining	4
2.2	従来のパケットルーティングとトラフィックエンジニアリング	5
2.2.1	OpenFlow	6
2.2.2	NSH	7
2.2.3	MPLS	8
2.3	SRv6	9
2.3.1	SRv6 を利用した layer-3 VPN の構築例と SRv6 によるパケット転送の具体的な動作	10
2.3.2	SRv6 パケットと SID の構造	12
第3章	背景と問題提起	14
3.1	Linux と netfilter	14
3.2	SRv6 と SF としての netfilter 統合手法	15
3.2.1	アプリケーションの実装を変更する手法	16
3.2.2	SR-Proxy を利用する手法	16
3.3	問題提起	18
第4章	提案手法の設計と実装	19
4.1	提案手法	19
4.2	設計	19
4.3	実装	22
4.3.1	Linux における SRv6 ビヘイビアの実装	22
第5章	評価	27
5.1	計測の概要と予想	27
5.2	TRex	29
5.3	レシーブサイドスケーリング (RSS)	30
5.4	パケットサイズ毎のスループット性能	31
5.4.1	計測内容	31

5.4.2	評価	32
5.5	netfilter にインストールされたルール毎のスループット	33
5.5.1	nftables	34
5.5.2	計測内容	36
5.5.3	評価	36
5.6	レイテンシ	38
5.6.1	計測内容	38
5.6.2	評価	39
第 6 章	結論と展望	40
	謝辞	41
	付録	47

目 次

2.1	SFC アーキテクチャ	5
2.2	トラフィックステアリングが必要なネットワーク例	7
2.3	OpenFlow アーキテクチャ	8
2.4	NSH のパケットヘッダ構造	9
2.5	SRv6 アーキテクチャ	10
2.6	SRv6 を利用した layer-3 VPN の動作例	11
2.7	詳細な SRv6 のパケット構造	13
3.1	netfilter フックポイント (wiki.nftables.org より引用 [1])	15
3.2	SR-Proxy アーキテクチャ	17
4.1	Linux カーネルネットワークスタック内における End.AN.NF の動作の流れ	21
4.2	改良した Linux kernel が End.AN.NF の SID を IPv6 ルーティングエント リとして扱っている様子	21
5.1	End.DT4 と H.Encaps の組み合わせ と End.AN.NF のパケット処理プロ セスの差	28
5.2	DPDK アプリケーションの動作概要	30
5.3	レシーブサイドスケーリングの動作概要	31
5.4	SRv6 End ビヘイビア毎, 及び IPv4 のスループット	33
5.5	ベースチェーンのルールごとのスループットの	37
5.6	レギュラーチェーンのルールごとのスループットの	38
5.7	SRv6 End ビヘイビア毎, 及び IPv4 のレイテンシ	39

表 目 次

5.1 実験に使用したマシンの構成	29
-----------------------------	----

第1章 序論

1.1 本研究の概要

Service Function Chaining (SFC) は, Software Defined Network (SDN) 及び Network Function Virtualization (NFV) の文脈で研究されているトピックである [2, 3, 4, 5]. SFC では, サービスファンクション (SF) を通過する順序や SF のタイプに関する情報を事前に定義し, それらのルールをネットワーク機器に配布する必要がある. SFC ネットワークを構築するネットワーク機器は, 事前に決定されたルールに従って受信したパケットを SF に導く. パケットを NF へ導くためのルールは, SDN コントローラやルーティングプロトコルによってネットワーク機器に配布される. ネットワーク機器は IP ルーティング上の最短経路に関係なく, 配布された SFC ルールに従ってパケットを転送する次のホップを選択する必要がある. また, パケットのヘッダにこれらのルールに合致させるための特別な情報を埋め込む手法が取られることもある. SFC は, クラウドサービスプロバイダ (CSP), アプリケーションサービスプロバイダ (ASP) 及びインターネットサービスプロバイダ (ISP) にとって, 現在の静的な環境に代わる柔軟かつ経済的な選択肢を提供する [6].

SFC を実現可能な技術には, いくつかの候補が存在する. 例えば OpenFlow [7], Network Service Header (NSH) [8], MPLS [9] などである. これらの技術はどれも, 最短経路に関係なく, ルールに基づいて受信したパケットを意図した SF に導く, という要件を満たすことができる. OpenFlow では, 経路情報を管理する中央のコントローラが, 実際にパケットを転送する OpenFlow スイッチに対して明示的にパケット転送ルールを設定する. OpenFlow スイッチは, コントローラによって適切に管理されたルールに従い, パケットを意図した SF に転送する. OpenFlow のもつこのアーキテクチャは, 従来のルーティングプロトコルに基づかない柔軟な経路制御を可能にする. NSH は Service Path Identifier (SPI) と Service Index (SI) によって SF を識別する. NSH ノードは, パケットに付与された NSH 内の SPI, SI に基づいてパケットを転送する. NSH は, サービスプレーンと呼ばれる専用のオーバーレイネットワークを作成し, そのオーバーレイネットワーク内でサービスを転送する. このオーバーレイネットワークを構築する, というアーキテクチャにより, NSH では基礎となるネットワークトポロジを変更することなくサービス転送を可能にする. 一方, MPLS では, 直接 NSH を使用する代わりに, MPLS ラベルスタックを利用する. このラベルスタックには, パケットが通過すべきノードの順序がホップバイホップで含まれている. ラベルスタック内で表現されるノードはルータだけでなく, SF も含まれるため, そのラベルスタックに基づいてパケットを転送する事で SFC を実現できる. このアプローチもまた, 基礎となるネットワークトポロジを変更せずに SFC を実

現するために必要な、最短経路によらないパケット転送を達成する。

Segment Routing (SR), 特に Segment Routing over IPv6 (SRv6) もまた, SFC を実装するために使用される技術の 1 つである. SR では, リンク, ノード, サービスといったネットワーク内の各エンティティを**セグメント**として表現する. SRv6 パケットのヘッダ (SRH) には, セグメントリストと呼ばれる, そのパケットが通過すべきセグメントの順序を示したリストが含まれている. SRv6 では, セグメントを識別するための ID (SID) として, IPv6 アドレスを使用する. 言い換えれば, SRv6 は IPv6 ルーティングインフラをその基盤として利用し, SRH 内で定義された順序に従って, 任意のセグメントを経由してパケットを転送する. SRv6 は, SF が実行されるノードをセグメントとして表現し, SID を割り当て, 任意の順序で SF を通過するようにパケットを転送することで SFC を実現する.

SRv6 では, SF を SID で表し, セグメントリストに基づいて適切にパケットを転送することで, SRv6 を基盤とした SFC ネットワークを実現できる. しかし, SRv6 レイヤよりも上位にある SF の振る舞いと, 基盤となる IPv6 ルーティングインフラをどのように統合するかは明確でない. 例えば, IPv4 パケットのネットワークアドレストランスレータ (NAT) を SRv6 ネットワーク内の SF として考慮する場合を考える. SRv6 ネットワーク内において, IPv4 パケットは, SRv6 ヘッダ (SRH) を含む外部 IPv6 ヘッダでカプセル化される. SF で動作する NAT の実装が SRv6 に対応していない場合, SR-Proxy [10] が必要となり, ネットワーク構成や運用における複雑さが増加してしまう [11]. 実装が内部パケットへの NAT と SRv6 に則した転送動作を同時に実行できる場合, それはレイヤバイオレーションとなる. Linux には, SERA [12] という iptables を拡張したファイアウォールアプリケーションが存在する. SERA は SRH でカプセル化されたパケットについて, カプセル化された内部パケットのヘッダ情報にマッチする iptables のフィルタールールを適用できる. SRv6 での基本的な転送動作として, End と呼ばれる動作がある. SERA は iptables を拡張することで, この End 動作を処理する機能も実装されている. ただし, 既に Linux カーネルには IPv6 ルーティング, 及び SRv6 End 動作に関する処理が実装されている. SERA は, Linux カーネルに実装されている SRv6 機能を使わずに, 独自に改良した iptables アクションによって End 動作を処理する. つまり, SERA は Linux カーネル内で統合されている IPv6 ルーティングインフラと SRv6 処理機能を使わずに, 独自に拡張した iptables によって SRv6 とフィルタリングサービスとしての NF を統合している.

本論文では, 既存の netfilter を内部実装に利用する SF アプリケーションの実装を変更することなく SRv6 対応 SF として扱えるようにする, End.AN.NF 提案する. End.AN.NF は Linux netfilter を NF として扱えるようにしつつ, Linux に実装されている IPv6 ルーティングインフラを活用する. End.AN.NF は受信した SRv6 内部パケットに対して, netfilter のフックポイントを透過的するように設計されている. 本論文では End.AN.NF を Linux カーネル上で実装し, スループットとレイテンシを評価した. 評価の結果, End.AN.NF は End.DT 4 と H.Encaps の組み合わせによる SRv6 内部パケットへの netfilter 適用と比較して 27% 高いスループットと 3.0 マイクロ秒低いレイテンシを実現した. さらに, End.AN.NF のレイテンシは, End.DT4 と H.Encaps の組み合わせよりも 3.0 マイクロ秒

低い。また、End.AN.NF のレイテンシはマイクロ秒解像度で End 動作と同じである。

1.2 本論文の目的と構成

本論文は、Linux ルータを利用している管理者の負担を減らしつつ、高性能で柔軟な SFC を実現することを目的としている。Linux には netfilter という高機能なパケット操作フレームワークが実装されており、管理者が netfilter を内部実装に使っている iptables や nftables などのアプリケーションを使って FW などのサービスを行うことは一般的である。また、Linux には SRv6 の機能が実装されている。SRv6 は近年注目される新しい技術であり、新しい技術であるがゆえに SRv6 をサポートするベンダ機器は多くない。Linux を使うことで、新たなベンダ機器を導入するコストを抑えながら SRv6 ネットワークを構築することができる。本論文では、SRv6 ネットワークにおける SFC 環境で netfilter を利用可能にする方法を提案する。

本論文における以降の構成は次の通りである。1 章では、本論文の概要及び構成を述べる。2 章では、本論文を読むにあたって必要となる前提知識を説明する。3 章では、本論文の背景と取り組む問題について説明し、提案手法の概要を述べる。4 章では、本論文の提案する新たな SRv6 End behavior である End.AN.NF についての設計や詳細な動作、及び実装について述べる。5 章では、本論文の提案する End.AN.NF の性能が実用的であるか、また Linux カーネルのメインラインに実装されている手法の組み合わせに比べてどれだけパフォーマンスが改善されたのかをスループット及びレイテンシの観点から評価する。6 章では、本論文における結論と今後の展望について述べる。

第2章 前提知識

本章では、

2.1 Service Function Chaining

SFC とは、エンドツーエンド通信を提供するために必要なさまざまなサービスファンクション (SF) を決定及び順序付けし、それらを介するようにトラフィックを操作することを指す。図 2.1 に、SFC アーキテクチャの概略図を示す。SF には、ファイアウォールや IP ネットワークアドレストランスレータ (NAT) などのネットワークサービスファンクションや、アプリケーション固有の機能が含まれる。SFC アーキテクチャは、基礎となるネットワークトポロジから独立したトポロジを前提としている。この基礎となるネットワークトポロジをアンダーレイネットワークといい、独立した SFC のためのネットワークトポロジをオーバーレイネットワークという。図 2.1 では灰点線のパスが物理的なパスであるアンダーレイネットワークを示し、その他の実線が SFC として選択可能なパスの例であるオーバーレイネットワークを示している。SFC アーキテクチャでは、パケットは通信の入口となるノードで事前に定義されたポリシとパケット内の情報から分類され、SFC 対応ドメイン内で適用する SF のセットを決める。その後、任意の順番で各 SF でパケット処理が適用されるように転送される。例えば、Logging をした後 FW Service を適用、Filtering を適用するような場合は、図 2.1 における緑のパスを辿るようになる。

SFC アーキテクチャはネットワークの用途や運用計画などのコンテキストに依存しない汎用的な場面で利用可能な技術であり、SFC アーキテクチャは固定ネットワークやモバイルネットワーク、多くのデータセンターアプリケーションに適用できる。SFC の構築に関して、すべての SF が満たさなければならない標準の定義や特性は存在しない。各 SF は単に「パケットに対する特定の処理を適用できる要素」として扱われ、特定のネットワークで常に有効な SF を静的に列挙することはできない。なぜなら、適用する SF の集合はその瞬間に有効な SF であり、それらが有効かどうかはその時々ネットワーク環境によって異なる場合があるからである。SF のチェーンとそれら呼び出す基準は、SF 対応ドメインを運用する各ネットワーク毎に固有である。

SFC における SF は、受信したパケットの特定の処理を担当する機能である。SF はプロトコルスタックのさまざまなレイヤで動作し、論理的なコンポーネントとして仮要素として実現されることもあれば、物理的な筐体としてネットワークの中に組み込まれることもある。近年では、SF が動作するマシンは物理的な筐体ではなく、汎用マシンにインストールされたハイパーバイザ上の VM の中で動作することも多い。また、コンテナ技

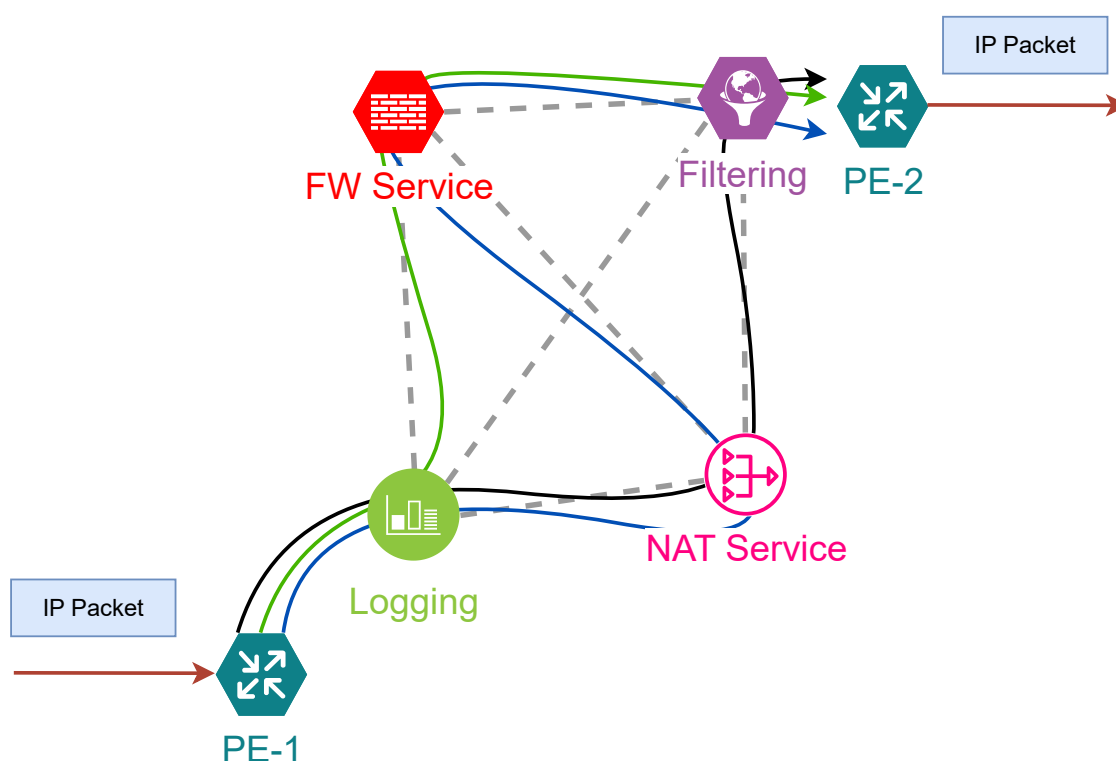


図 2.1: SFC アーキテクチャ

術の台頭により，SF 自体をコンテナに閉じ込めてデプロイすることも一般的になっている．さらに，それらのコンテナを kubernetes [13] に代表されるコンテナオーケストレータによって管理する手法も提案されている [14] このように，仮想化されたネットワーク上の機能 (Network Function / NF) を NFV という．SFC は NFV の利用例として NFV のコンテキストでも研究されてきた技術である．

2.2 従来のパケットルーティングとトラフィックエンジニアリング

章 2.1 で述べた通り，SFC を実現するためには，エンドツーエンド通信を提供するために必要な複数の SF を決定，及び順序付けし，それらを介するようにトラフィックを操作する必要がある．しかし，従来のパケットルーティングでこのようなトラフィック操作を実現することは難しい．

ルータが IP パケットを転送するとき，ルータは自身の持つルーティングテーブルを参照する．このルーティングテーブルは通常，BGP [15] や OSPF [16]，IS-IS [17] などのルーティングプロトコルを通じて交換した経路情報から作成される．一般的に，多くのルーティングプロトコルでは「経由するノードの数を最小にする経路を最も良いものとする」という基本設計をもとに，オペレータが任意に決定したコスト情報などを含めて最も良い経路を計算する．このように決定された最も良い経路をベストパスといい，ルーティ

ングテーブルには「ある宛先アドレスを持つパケットは次にどのノードに転送するのがベストパスなのか」が書かれている。ルータは、自身に接続されているノードやルーティングプロトコルを通じて受け取った経路情報が変更されたとき、その変更を近接ルータに通知し、自身のルーティングテーブルを更新する。ルーティングテーブルはオペレータが静的に構築することもできる。しかし、静的に経路を決定してしまうとノードの近接情報が変わるたびにオペレータ自身が設定し直す必要があり、これは手間がかかったりオペレーションミスを誘発したりする問題がある。そのため、静的な経路設定が利用される場面は限定的である。

図 2.2 に、あるネットワークのトポロジを示す。このネットワークにおいて、User-B の通信は青いパスを通るように、User-A から Server に向かう通信は Security appliance を経由させるような経路制御を行いたい。しかし、Router-B から Router-C までの経路は青いパスを通ると 1 hop だが、Security appliance を通る赤い経路は 2 hop である。つまり、Router-B から Server までの経路は青いパスの方が経由するノードの数が少ない。そのため、通常のルーティングプロトコルで経路を学習すると、Router-B のルーティングテーブルには Server に向かう経路として青いパスがベストパスとして採択される。ルーティングプロトコルの設定でコストを変更することで赤いパスをベストパスにすることは可能である。ベストパスを赤いパスに変更すると、User-A の通信は意図通り赤いパスを通るようになる。しかし、ベストパスを変更してしまうと User-B の通信についても赤いパスを通るようになり、これは意図した通信経路にならない。

ベストパスによらずに、また特定の通信毎に選択して経路を制御することを、トラフィックステアリング、トラフィックエンジニアリング (TE) という。TE が可能なパケット転送メカニズムとして、いくつかの候補が存在する。例えば OpenFlow [7], Network Service Header (NSH) [8], MPLS [18] などである。

2.2.1 OpenFlow

OpenFlow のによるネットワーク構成の概略図を図 2.3 に示す。OpenFlow では、OpenFlow スイッチと呼ばれる OpenFlow に対応した専用のネットワーク機器をパケットを転送するノードとして使用し、OpenFlow コントローラと呼ばれる専用のマシンが OpenFlow スイッチの経路情報を集中管理する。OpenFlow のアーキテクチャは一般のパケットルーティングアーキテクチャとは大きく異なる。一般的なネットワークでは先に挙げた BGP や OSPF などのルーティングプロトコルを利用して経路情報を交換し、ルーティングテーブルを作成する。そして、ルータは自身が作成したルーティングテーブルに基づいてパケットを転送する。対照的に、OpenFlow では BGP や OSPF などのルーティングプロトコルを利用してルーティングテーブルを作成することはしない。経路情報は OpenFlow コントローラが集中管理し、OpenFlow コントローラは自身が決定した経路情報を実際にパケットを転送する OpenFlow スイッチへインストールする。また、OpenFlow ではルーティングテーブル自体も一般的なルーティングで作成されるものとは異なり、OpenFlow で利用されるルーティングテーブルに対応するテーブルのことをフローテーブルという。OpenFlow のフローテーブルには、宛先の IP アドレスだけではなく、送信元アドレスや

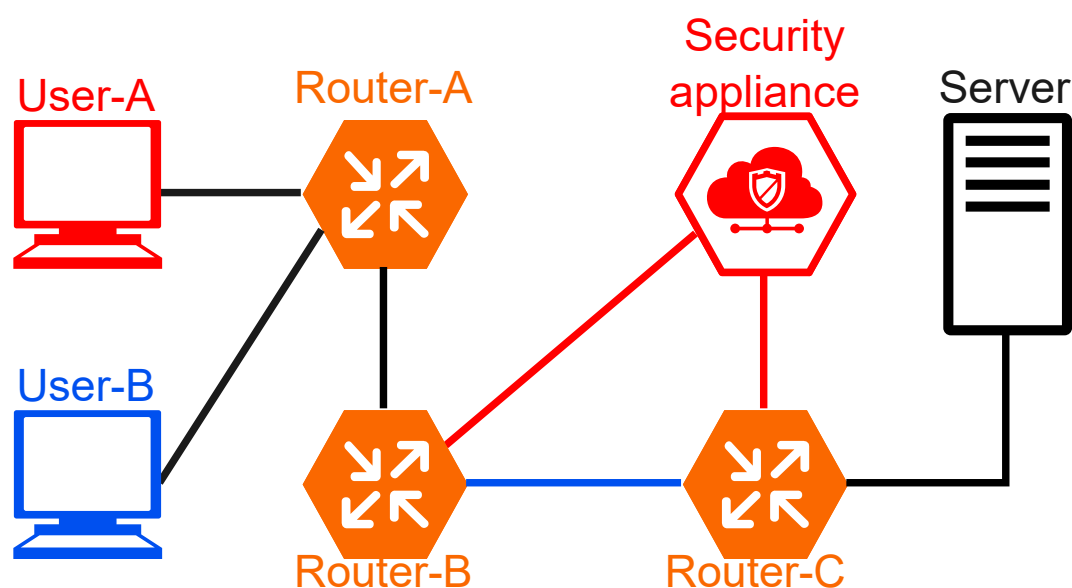


図 2.2: トラフィックステアリングが必要なネットワーク例

通信を受信したポート，独自定義の専用パケットヘッダのフィールドなどもマッチングルールとして含まれる．これにより，IP 的なベストパスによらずに，かつ同じ宛先であっても別のパスを選択できる．

2.2.2 NSH

NSH は，SFC を実現する 1 つの手法として考えられたプロトコルで，NSH と呼ばれるヘッダでパケットをカプセル化する．OpenFlow が汎用的なパケット転送アーキテクチャとして考案されたのとは対比的に，NSH は SFC を前提として考案された．NSH で転送されるパケットの構造を図 2.4 として示す．NSH パケットは，大きく分けて 3 のパートに分けられる．Original Packet は，実際のエンドツーエンドでやり取りするパケットのことを指す．そのパケットを，NSH でカプセル化している．NSH の中にはいくつかのフィールドが存在し，その中には SFC の中でどのパスを通過するのか通過するのかや，ユーザ定義のメタデータ，オプションなフィールドなどが含まれる．NSH でカプセル化されたパケットを，更に Transport Encapsulation というヘッダがカプセル化している．この Transport Encapsulation は特定のフォーマットである必要はなく，GRE，VXLAN などの一般的なトンネリングプロトコルや通常のイーサフレームである．Transport Encapsulation の目的は，オーバーレイネットワークを通じて適切な SF ノードまでパケットを転送することである．NSH に対応したノードでは，SF が適用されたパケットに対して，そのパケットの NSH を参照し次の SF のノードを決定する．次の SF ノードが決まったらそのノードに届くよう，対応する Transport Encapsulation でカプセル化することで TE ができる．

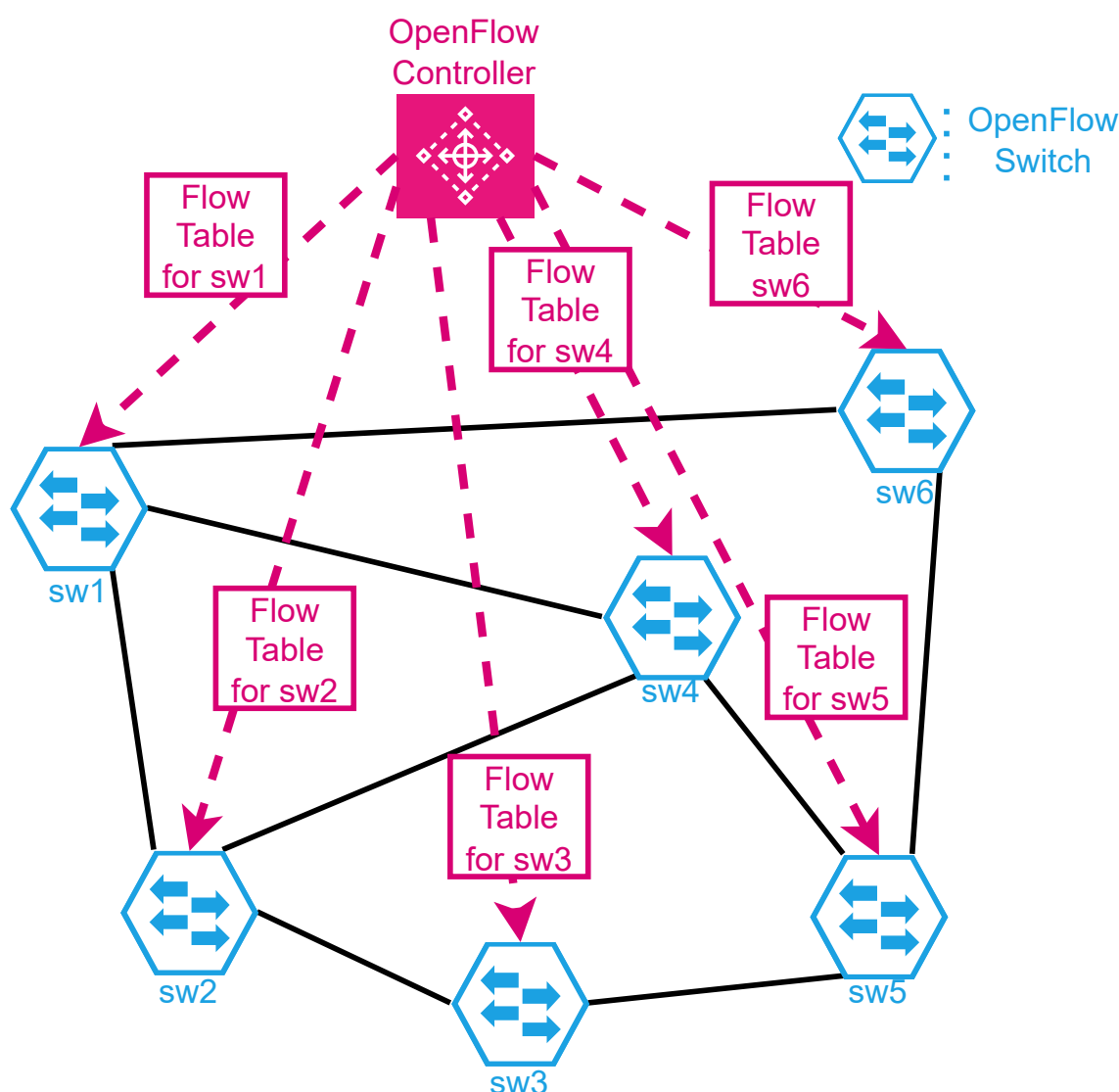


図 2.3: OpenFlow アーキテクチャ

2.2.3 MPLS

MPLS とは Multiprotocol Label Switching の略称であり、MPLS ヘッダに含まれるラベル情報に基づいてパケットを転送するプロトコルである。MPLS ヘッダは Layer 2 ヘッダと Layer 3 ヘッダの間に挿入され、MPLS ヘッダの中にラベル情報を始めとするいくつかのフィールドが埋め込まれる。MPLS では、宛先の IP アドレスではなく、MPLS ヘッダに含まれるラベル情報を参照してパケットを転送する。つまり MPLS では、MPLS ヘッダ内に埋め込まれたタグ情報を使ってパケットを転送するため、IP 的なベストパスによらないルールでパケットを転送できる。また、MPLS は古くから VPN を構成するために用いられてきた一般的なプロトコルである。そのため、多くのネットワーク機器でサポートされている。OpenFlow は特別な機器やコントローラが必要であり、NSH も比較的新しいプロトコルで、かつ機能も SFC に特化しているため、NSH をサポートする機器は少

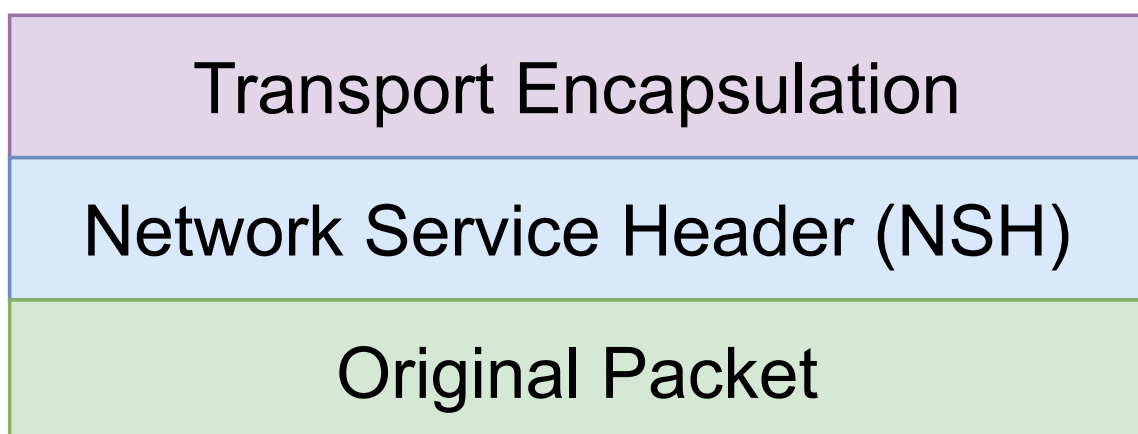


図 2.4: NSH のパケットヘッダ構造

ない。対象的に MPLS は既に多くの機器でサポートされているため、OpenFlow や NSH と比較して導入が容易であるという特徴を持つ。また、RFC8595 [9] のように、NSH と MPLS を組み合わせて SFC を実現する手法も提案されている。このアーキテクチャでは、NSH の Transport Layer として MPLS を利用している。

2.3 SRv6

先に挙げた技術だけではなく、Segment Routing over IPv6 (SRv6) も TE を適用できる技術の 1 つである。MPLS が独自のラベルを使って使ってパケットを転送するアーキテクチャであるのに対し、SRv6 では MPLS のラベルに対応する概念として IPv6 アドレスを利用する。SRv6 で利用される識別子はセグメント識別子 (SID) と呼ばれ、各 SID はネットワーク内の特定の場所で実行される特定の機能を表す。この SID は IPv6 と全く同じフォーマットをしている。SRv6 では、SRv6 ヘッダ (SRH) と呼ばれる IPv6 拡張ヘッダに SID の一連の集合からなるリストを埋め込むことで、ネットワークオペレータやアプリケーションはパケットが通過する中間地点を指定できる。

SRv6 ヘッダには通過するネットワーク上のノードの順番がリストとして埋め込まれ、ルータはそのリストに基づいてパケットを転送する。図 2.5 において、例えば SID リストの要素が **A**, **FW**, **C** である場合、パケットは緑色のパスを通るように転送される。また、SRv6 はあるパケットが経由するノードを指定できるだけでなく、パケットに対してパケットに対して特定の操作を適用できる。このようなパケット操作の種類のことを **SRv6 ビヘイビア** という。現在 RFC8986 [19] では 15 種類の End ビヘイビアが定義されている。

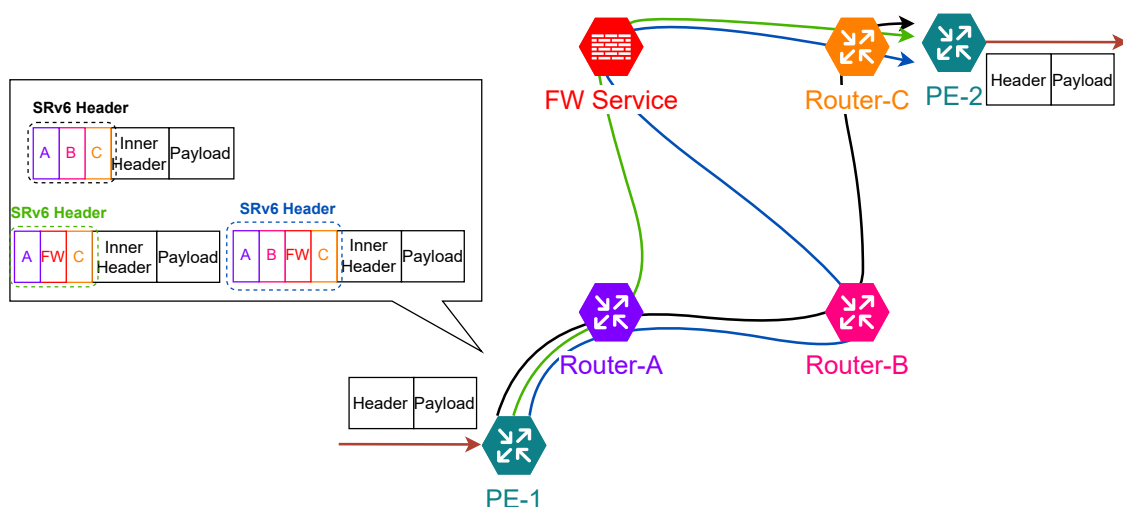


図 2.5: SRv6 アーキテクチャ

2.3.1 SRv6 を利用した layer-3 VPN の構築例と SRv6 によるパケット転送の具体的な動作

SRv6 ビヘイビアを組み合わせることで、layer-3 VPN を構成することもできる [20]. SRv6 を利用した layer-3 VPN の動作を図 2.6 に示す。

図 2.6 ① において、PE-1 は受信したパケットを SRH でカプセル化する。このように SRH でパケットをカプセル化する、という操作も SRv6 ではビヘイビアとして定義されており、この操作のことを H.Encaps という。ここでは、PE-1 は受信したパケットに対して **A**, **FW**, **C** を意味する SID リストを付加したものとする。このとき、SRv6 でカプセル化されたパケットの宛先アドレスは、内部パケットの宛先アドレスに関わらず、次に到達すべきノードを示す SID である FW Service になる。

PE-1 は H.Encaps で SRH を付加したパケットを Router-A へ送信する。このとき、パケットは宛先 IPv6 アドレスが Router-A である単なる IPv6 パケットとして扱われる。PE-1 は自身の持つルーティングテーブルを参照し、Router-A へのネクストホップを決定し、パケットを送出する。図 2.6 ② は、Router-A が受信したパケットに対して SID を 1 つ進めて FW Service にパケットを転送している様子を示している。リスト状になっている SID の中でどれが現在有効な SID であるかを指定するために、SRH にはセグメントレフト (segleft) と呼ばれるフィールドが定義されている。segleft は SID リストのインデックスであり、(SID の合計) - 1 から始まり、0 で終わる。

Router-A は受信した SRv6 パケットの segleft を 1 つデクリメントし、FW Service の SID が次に有効な SID であることを示すようにする。また、Router-A はパケットの宛先アドレスを新しく有効になった FW Service の SID に書き換える。このように、segleft を 1 つ進め、宛先アドレスを新たに有効になった SID で書き換える動作のことを End ビヘイビアといい、これも SRv6 ビヘイビアの 1 つである。End ビヘイビアは SRv6 の中で最も基本的なビヘイビアである。

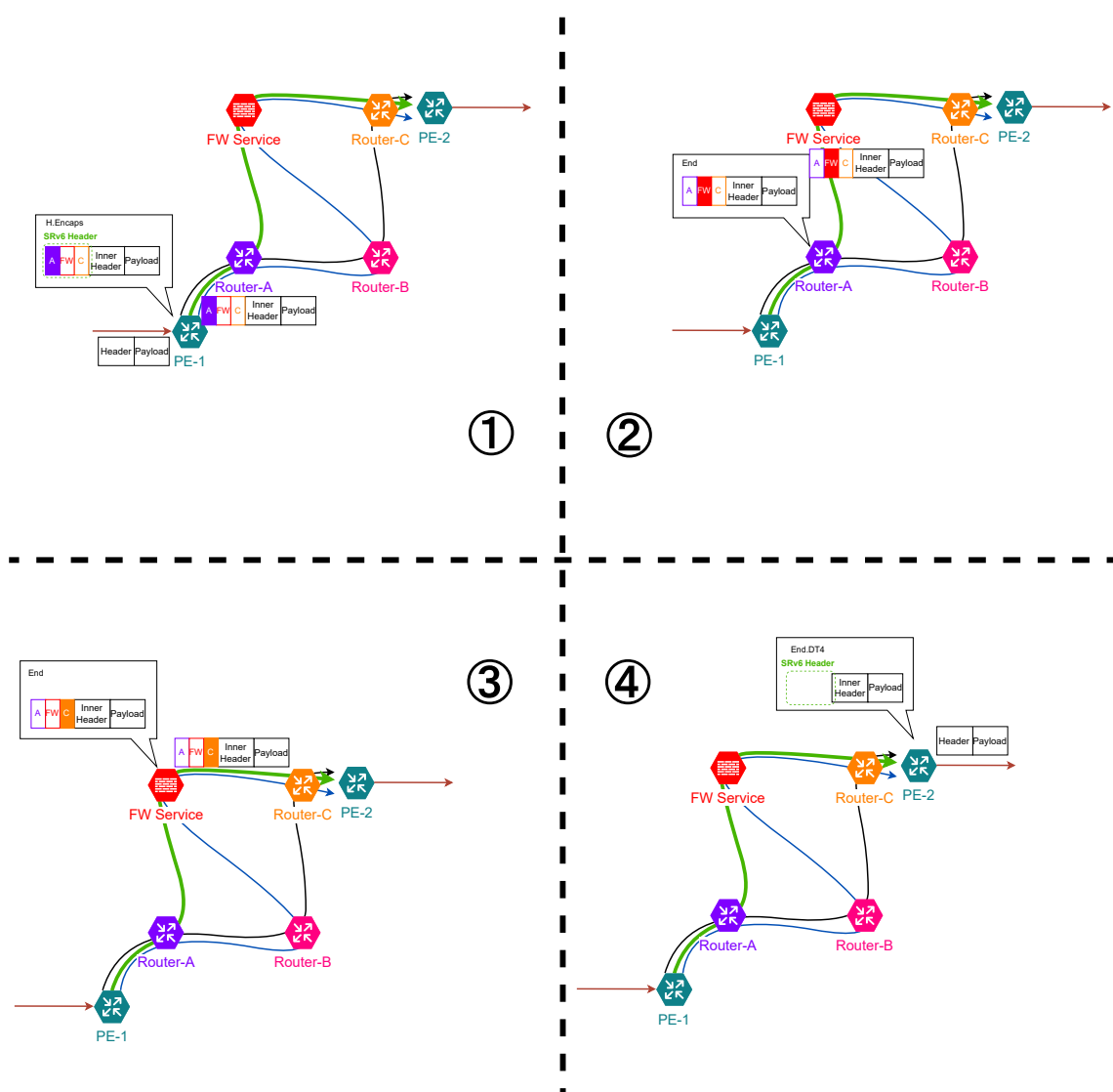


図 2.6: SRv6 を利用した layer-3 VPN の動作例

宛先アドレスを書き換えたあと、Router-A は自身のルーティングテーブルから新たな宛先アドレス (FW Service の SID) を検索し、ネクストホップへ転送する。図 2.6 ③では、②と同様に FW Service が End ビヘイビアを実行して segleft デクリメントし、新たに有効になった SID に基づいて Router-C へパケットを転送している。図 2.6 ④では、PE-2 が受信したパケットに対して End.DT4 というビヘイビアを実行している。このビヘイビアは、SRH を取り除き、特定の VRF を参照して SRv6 でカプセル化されていた内部パケットを転送する、という動作を実行する。End.DT4 により、パケットから SRH は取り除かれ、PE-1 でカプセル化される前のパケットを得ることができる。

2.3.2 SRv6 パケットと SID の構造

図 2.7 に、SRv6 パケットの詳細な構造を示す。図 2.6 では、簡単のためにパケット構造の図を簡略化して表現した。実際の SRv6 パケットは、一般的な IPv6 ヘッダの下に SRH が位置する。IPv6 ヘッダ内の Next Header (NH) には 43 が設定され、この 43 という数字は次に来るヘッダのタイプが IPv6 ルーティングヘッダであることを示す。

先に述べた通り、SID は SRv6 で利用される識別子で、SRH にはいくつかの SID がリスト状になって含まれている。現在どの SID が有効であるかは、segleft の値によって決まる。図 2.7 の例では、segleft は 1 である。よって、Segment List のインデックスが 1 である、最初から 2 番目の SID が有効化されている。宛先アドレスには現在有効な SID の値が設定されるため、このパケットの宛先アドレスは Segment List の 2 番目の SID となる。

図 2.7 で示されているように、SID は LOC:FUNCT:ARG という 3 つのパートに分かれている。また、SID は IPv6 アドレスであるため、それら 3 つパートの長さの合計は 128bit である。LOC はロケータを表す。ロケータとは、ある SID に対応するノードの場所を表す。FUNCT は SID に関連付けられた SRv6 ビヘイビアの識別子であり、ARG はビヘイビアの動作に必要な追加情報をエンコードする領域である。例えば、End.DT4 では ARG フィールドにはルックアップするべき VRF テーブルを識別するための情報がエンコードされる。

SRv6 の SID は IPv6 アドレスであるため、既存のルーティングプロトコルを用いて SID を経路情報として既存のルーティングプロトコルを利用して広告できる。SRv6 ビヘイビアは、SID の FUNCT パートで表現される。ただし、「特定の SRv6 ビヘイビアならば FUNCT パートはこの値である」というような一般的な定義は存在しない。例えば、IS-IS の TLV の Type フィールドの値は IANA によって策定されている [21]。それに対して、SRv6 ではビヘイビアに対応する値は標準化されず、オペレータが自由に決めることができる。図 2.6 ④ では、Router-C は受信したパケットに対して End.DT4 を実行している。Router-C は LOC:FUNCT:ARG のフォーマットに従って **[Router-C のロケータを示すブロック]:[Router-C 上で End.DT4 として定義されたブロック]:[利用可能な VRF table 番号]** を自由に決定し、それを経路情報として周りのノードへ広告する。

SRv6 ビヘイビアが実装されていないネットワーク機器が SRv6 でカプセル化されたパケットを受信した際、その機器は SRv6 パケットを IPv6 パケットとして転送できる。SRH の前には一般的な IPv6 ヘッダが挿入されているため、そのパケットは IPv6 パケットとして転送可能であるからである。つまり、SRv6 ビヘイビアを実行せず、単にパケットを現在有効な SID へ転送するだけであれば、IPv6 パケットフォワーディングが実装されていれば適切にパケットを転送できる。

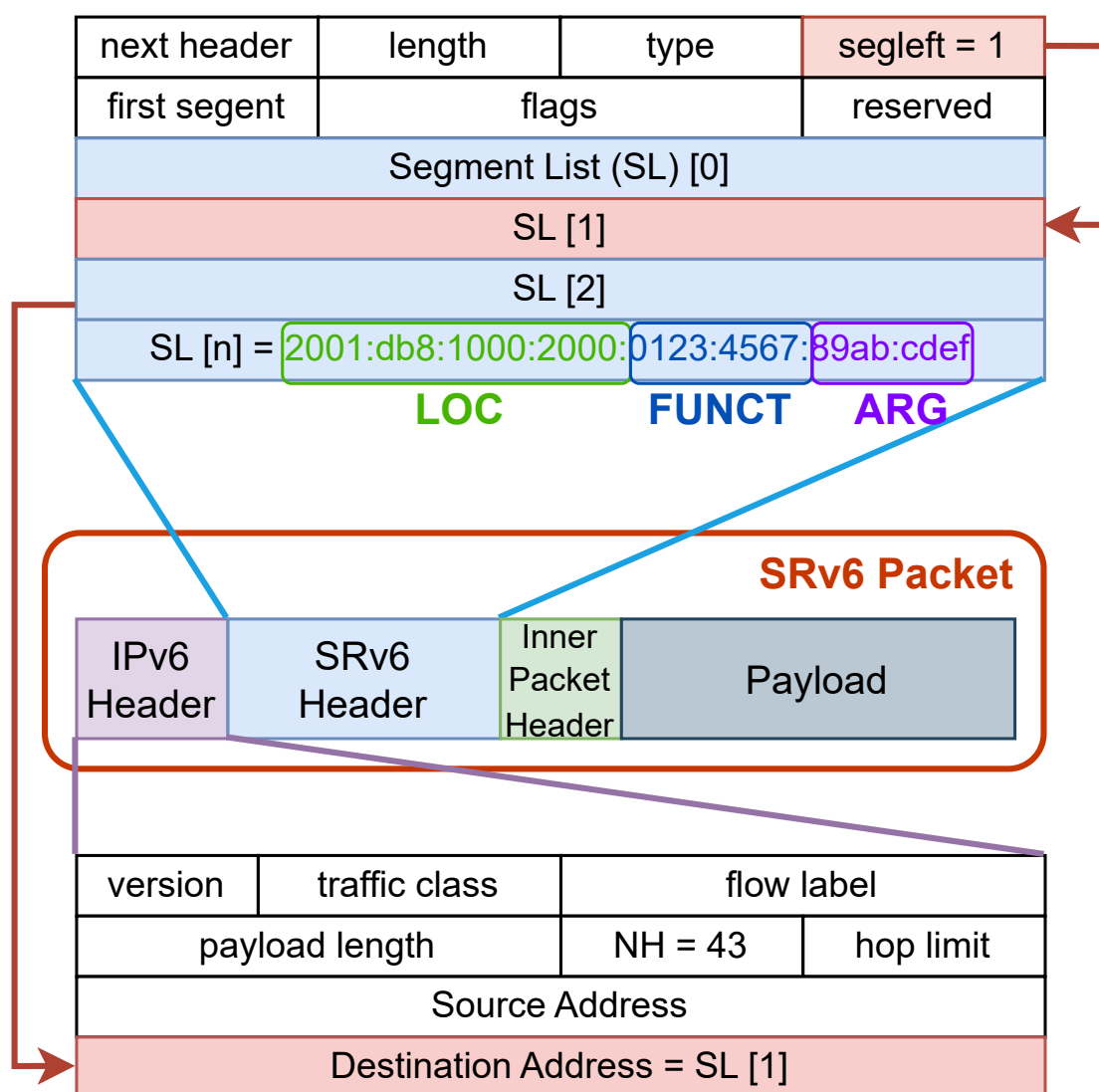


図 2.7: 詳細な SRv6 のパケット構造

第3章 背景と問題提起

章 1 では、SFC の概念と SFC を実現するための要件を満たせるいくつかのプロトコルを説明し、近年特に注目されている SRv6 について説明した。その知識を踏まえた上で、本章では背景と本論文で解決すべき問題を提起する。

3.1 Linux と netfilter

SFC をデプロイする上で、SF を動作させる環境として Linux を選択することは有用である。章 2.1 で述べたように、SFC 環境では SF を汎用サーバや仮想マシン、コンテナの中にデプロイすることが一般的になっている。Linux 上で任意のアプリケーションを開発して、そのバイナリを動作させることは容易であり、開発に必要な基盤や情報も十分に整っている。更に Linux カーネルはコンテナメカニズムもサポートしており、SF アプリケーションをコンテナの中で動作させることも可能である。また、Linux カーネルにはパケットフォワーディング機能が実装されている。IPv4 パケットや IPv6 パケットの転送はもちろん、MPLS や VXLAN、更にはいくつかの SRv6 ビヘイビアも実装されている。SRv6 では既存のルーティングプロトコルを使って SID を経路情報として他のルータへ広告することができる。Linux には FRR [22] や gobgp [23] などのルーティングソフトウェア実装が存在し、これらを利用して SID を広告できる。

Linux カーネルには netfilter [24] と呼ばれるパケット処理フレームも実装されており、これは SF アプリケーションを開発するのに有用である。netfilter は、パケットのフィルタリングやロギング、NAT、NAPT、やその他のパケットマングリングを可能にするフレームワークである。netfilter は、iptables や nftables といったパケット処理アプリケーションの内部の実装に使われている。iptables や nftables といった、netfilter を基盤として実装されたアプリケーションのことを、本論文では netfilter-based アプリケーションと呼ぶ。netfilter-based アプリケーションの例として、iptables や nftables 以外にも、conntrack-tools[25] や snort[26] といったアプリケーションも存在する。

netfilter の仕組みを使うと、任意のカーネルモジュールは Linux カーネルのネットワークスタック上で定義された特定の場所に場所にコールバック関数を設定できる。カーネルモジュールとは、Linux カーネルのソースコードそのものを書き換えず、かつマシンの再起動を必要としない Linux カーネルの機能を拡張するプログラムのことである。一般的なユーザ定義のアプリケーションはユーザ空間で動作するため、カーネル空間で特定のプログラムを実行することはできない。しかし、カーネルモジュールとして動作するプログラムは、カーネル空間で動作する。つまり、開発者はカーネルモジュールを開発すること

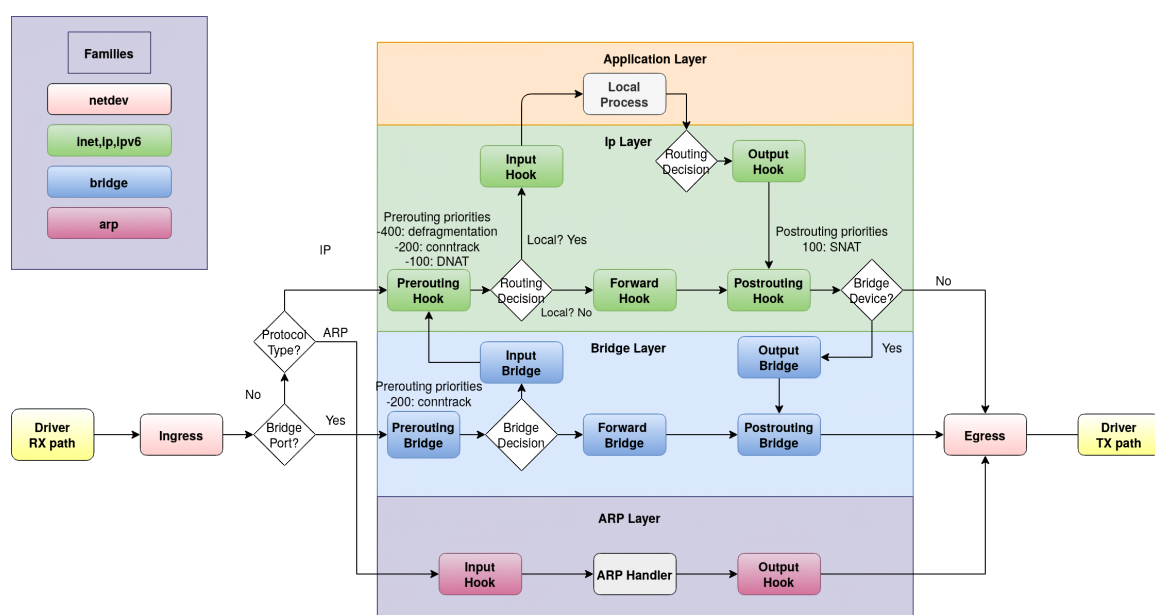


図 3.1: netfilter フックポイント (wiki.nftables.org より引用 [1])

で Linux カーネルに機能を追加することができる。

netfilter によってコールバック関数を設定できるポイントの一覧を、図 3.1 として示す。なお、この図は wiki.nftables.org より引用した図である。図 3.1 から読み取れるように、netfilter は Linux のネットワークスタックの様々な場所にコールバック関数を設定することができる。これらの netfilter がコールバック関数を設定できるポイントを、netfilter フックポイント、または単にフックポイント、フックという。いくつかのフックポイントは、現在処理しているパケットの宛先が自分自身かどうかによって通過するかしらないかが変わる。例えば、IP レイヤに存在するフックポイントでは、パケットの宛先アドレスが自分であれば Input フックポイントを通過するが、そうでない場合は Forward フックポイントを通過する。Forward フックにのみ特定のパケットをフィルタリングするコールバック関数を登録することで、自身宛のパケットはフィルタリングをかけないが、転送するパケットにのみフィルタリングを適用する、という使い方ができる。

3.2 SRv6 と SF としての netfilter 統合手法

Linux netfilter と SRv6 を利用して SFC をデプロイすることは有用な手法であるように思えるが、実際には問題が存在する。章 3.1 で述べたように、Linux カーネルには SRv6 の機能が実装されており、netfilter は SF アプリケーションの開発に有用である。しかし、netfilter は SRH でカプセル化された内部のパケットに対して netfilter フックポイントを適用できない。これは、例えば転送するパケットの送信元アドレスを書き換える NAT 操作を SRv6 パケットの内部に適用しようとしても、SRH でカプセル化されているために通常のパケットと同じ操作では NAT を適用することができないからである。つまり、一

一般的な IPv4 パケットに対して特定の操作を行うために実装された netfilter-based アプリケーションを SRv6 の内部 IPv4 パケットに対して適用する事はできない。

SRv6 に対応していない SF アプリケーションのことを、SR-unaware アプリケーションといい、対照的に SRv6 に対応している SF アプリケーションのことを SR-aware アプリケーションという。SR-unaware アプリケーションを SRv6 環境で SF として利用する方法はいくつか存在する。以降では、3 つの手法について解説する。

3.2.1 アプリケーションの実装を変更する手法

最も単純な方法は、アプリケーション自体の実装を変更し、SR-aware アプリケーションにすることである。SERA [12] は、Linux iptables に統合された SR-aware アプリケーションの実装である。SERA は Linux iptables を拡張し、SRH のフィールドと iptables のルールをマッチさせて、ファイアウォール用のフィルタリングルールを適用する。また、SERA は SRH でカプセル化された内部の IPv4 パケットに対してファイアウォールルールを適用する事もできる。更に SERA は、SRv6 End ビヘイビアのように、パケットを次の SID に転送する機能も持つ。しかし、SERA は iptables の拡張であるため、その他の netfilter-based アプリケーションを SR-aware にすることはできない。SERA のような手法で netfilter-based アプリケーションを SR-aware にするためには、アプリケーション毎にその実装を変える必要がある。また、SERA の採用した iptables を拡張するというデザインは、SERA に関連する SID を既存のルーティングインフラに統合することを困難にしている。iptables 内のフィルタリングルールとして利用するための SID の情報は、2.3 章で解説した layer-3 VPN の例とは異なり、既存のルーティングプロトコルを通じて広告することはできない。

3.2.2 SR-Proxy を利用する手法

もう 1 つの方法は、SR-Proxy と呼ばれる手法を適用することである。SR-Proxy の概念図を図 3.2 として示す。図 3.2 において、Router-B が SR-Proxy を適用するノードである。Router-B は、Router-A から受信した SRv6 パケットの SRH を一度取り外し、その状態をキャッシュしておく。Router-B は、取り外して得られた SRv6 の内部パケットを FW ノードへ送信する。FW ノードが受信するパケットは SRH でカプセル化されていない一般的な IP パケットであるため、FW ノードは受信したパケットを一般的な IP パケットとして解釈し、FW サービスを適用する。FW ノードは、FW サービスを適用したパケットを再び Router-B に送り返す。Router-B は FW サービスから受け取った非 SRv6 パケットを、キャッシュしておいた SRH で再びカプセル化する。Router-B は自身が再度カプセル化した SRv6 パケットを、次の転送先である Router-C へ送信する。

SR-Proxy を利用する方法は、汎用性が高い。SR-Proxy を実行するノードが SRH を取り外して SF ノードにパケットを転送するため、SF アプリケーションは SRH の存在を意識する必要がない。この手法であれば、iptables に限らず、あらゆる netfilter-based アプリケーションを SFC 中の SF として扱うことができる。

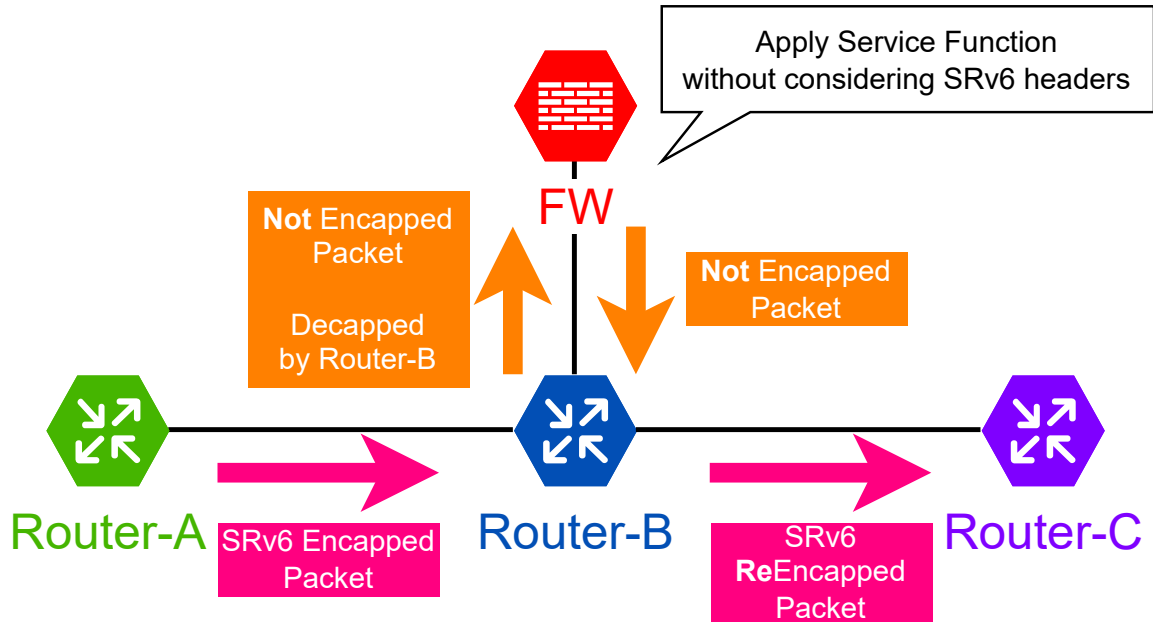


図 3.2: SR-Proxy アーキテクチャ

また, SR-Proxy は SRv6 のビヘイビアとしても提案されており [27], End.AS や End.AD や End.AM と呼ばれるビヘイビアがそれに該当する. これらは SRv6 ビヘイビアであるため, それら SID は IPv6 アドレスとして表現され, 既存のルーティングプロトコルを使ってそれらの SID を広告することができる. 2024 年 1 月現在, End.AD や End.AM は Linux カーネルのメインライン上では実装されておらず, これらの実装はワークインプログレス状態である. Linux で動作する SR-Proxy として, SRv6 ビヘイビア以外の実装も提案されている [28, 29, 30].

しかし, SR-Proxy を利用する方法には, SERA のような SR-aware アプリケーションには存在しないオーバーヘッドが存在する. それは, SR-Proxy が一度 SRv6 パケットから SRH をデカプセル化する際に生じるオーバーヘッド, 一度外部ノードに転送し, SF 適用後に再度受信するオーバーヘッド, そして受信したパケットを再度同じ SRH でカプセル化するオーバーヘッドである.

また, SR-Proxy は根本的にネットワークにさらなる複雑さをもたらす. SR-Proxy は SF から返されるパケットに付加する適切な SID リストを決定する必要がある. SF から返される内部パケットは任意の宛先と送信元を持つ可能性があるため, SR-Proxy が付けるべき SID リストは内部パケットによって異なる可能性がある. つまり, SR-Proxy は適切な SID リストを決定するための独自のメカニズムを実装する必要がある. 例えば, 静的な SID リストをアタッチする End.AS か, プロキシの内部で状態をキャッシュする必要がある [28]. さらに, SR-Proxy をデプロイするためにはいくつかの問題が存在する. 例えば, 特定の SR-Proxy タイプと共存できないサービスのタイプがあったり, サービスの有効性を検出する必要や SR-Proxy が送信する先の SF に関する SID 広告の問題など [11] が既にインターネットドラフトとして挙げられている.

3.3 問題提起

現在, Linux の持つ SRv6・IPv6 ルーティングインフラストラクチャを活用しながら Linux カーネルに実装されている netfilter という多機能なパケット処理機能を NF として利用する手法は, 確立されていない. 現在提案されている手法では, Linux の持つ IPv6 ルーティングインフラストラクチャを活用した SR-aware アプリケーションをシンプルに汎用的に実現することは難しい. また, Linux カーネルには netfilter という多機能なパケット処理機能が実装されているものの, SRv6 上で netfilter を直接 NF として扱う方法も確立されていない. SRv6 は SF を SID として表すことで SFC を実現可能なアーキテクチャであるものの, SID として表現された SRv6 上のノードとしての SF と, 実際のアプリケーションとしての SF を統合する方法は自明ではない. セクション 3.2.2 で述べた SR-Proxy を利用する方法では, SR-Proxy を導入することで生まれるオーバヘッドや運用上の問題が指摘されている. また, SRH でカプセル化されているパケットに対して, パケットをカプセル化したまま netfilter 自体を適用する手法も確立されていない. 本論文では, Linux netfilter を SRv6 を使って構築された SFC 上の SF として活用するための手法を提案する.

第4章 提案手法の設計と実装

本章では、前提となる Linux カーネルにおけるパケットフォワーディング処理, netfilter に関する知識を解説し、本論文の提案手法についての設計と実装について述べる。

4.1 提案手法

本論文では、netfilter を SR-aware SF として利用するための新しい実装として、Linux のルーティングインフラに netfilter によるパケットのフィルタリングとマングリング機能を統合した End.AN.NF を提案する。End.AN.NF は、End behavior of SR-aware Native function for NetFilter の略であり、これは SRv6 ビヘイビアの 1 つの種類である。End.AN.NF は netfilter-based アプリケーションを SR-aware にするのではなく、netfilter そのものを SR-aware にするという考えで設計されている。これにより既存の netfilter-based アプリケーションは、そのアプリケーションの実装を変更することなく、SRv6 で構築された SFC 環境で SF アプリケーションとして動作させることができる。End.AN.NF の実装は、Linux カーネルの IPv6 ルーティングスタックを活用するように設計されている。End.AN.NF の SID は IPv6 アドレスとして表現され、Linux 上では IPv6 のルーティングテーブルエントリとして扱われる。End.AN.NF を示す SID は、通常の IPv6 経路として既存のルーティングプロトコル、及びその実装を介して他のノードに透過的に広告される。End.AN.NF は、SRv6 パケットに対して、IPv6 パケットとして netfilter ルールを適用し、かつカプセル化されたインナーパケットに対しても同様に netfilter ルールを適用できる。End.AN.NF は、nftables[31] や iptables [32] などの netfilter-based アプリケーションを介して設定された、トラフィックに対する選択的なパケット破棄や NAT の適用などを SRH でカプセル化された内部のパケットに対しても適用できる。

4.2 設計

End.AN.NF は、トランジットするパケットを Linux ネットワークスタックの IPv6 パケット転送フローに既に存在する netfilter フックポイントを通過させ、かつ SRv6 レイヤに 3 つの netfilter フックポイントを持ち、異なるタイミングでパケットに netfilter ルールを適用する。図 4.1 は、トランジットパケットに適用される netfilter のフックのフローを示している。受信したあるパケットに対して End.AN.NF が動作する際、そのパケットには 2 段階の netfilter フックが適用される。1 段階目の適用では、SRH を含む外部 IPv6 ヘッダのついたカプセル化されたパケットに対して、その外部 IPv6 ヘッダをター

ゲットにして実行される。これは End.AN.NF が提供するものではなく、SRv6 パケットが IPv6 パケットとして解釈されて転送される際に適用されるものである。2つ目の適用では、SRH を含まない、カプセル化された内部パケットの IP ヘッダをターゲットにして実行される。まず、End.AN.NF カーネルに実装した Linux の SRv6 ノードが IPv6 パケットを受信すると、そのカーネルは受信したパケットに prerouting フックを適用し、通常通り宛先 IPv6 アドレスの最長プレフィックスマッチングを行う。宛先アドレスが自身の持つルーティングテーブル上で End.AN.NF の SID として定義されていた場合、カーネルはパケットを End.AN.NF の実装に渡し、そうでない場合、カーネルは IPv6 レイヤの forward フックと postrouting フックを適用しながら、SRv6 パケットを IPv6 パケットとして解釈し、対応するネクストホップに転送する。一方、End.AN.NF は、SRH でカプセル化されたインナーパケットに対して、再度、prerouting フック、forward フック、及び postrouting フックを適用する。この 3 つの netfilter フックポイントは、図 3.1 で示されている通り、カーネルがあるパケットをトランジットする際に IP レイヤで通過するフックポイントである。End.AN.NF の段階で netfilter が適用されている間、SRH は End.AN.NF によって隠されるので、netfilter は SRH の処理を考慮する必要がない。End.AN.NF が終了すると、外部 IPv6 ヘッダの宛先アドレスは次の SID に置き換えられ、カプセル化されたパケットは Linux の IPv6 パケットフォワーディングプロセスにおける通常の転送パスに戻る。

End.AN.NF は、パケットをマーキングするために SID の ARG フィールドを利用する。SRv6 の仕様上、ある End ビヘイビアがその End ビヘイビア固有の用途で ARG を利用することが許可されている。End.AN.NF では、SID の ARG がマークとしてカーネル空間におけるパケットバッファに付加される。netfilter-based アプリケーションは、パケットバッファ上のマーク部分を照合することで、適用するルールを変更することができる。したがって、オペレータが、単一の End.AN.NF SID しか定義されていない場合であっても、SF アプリケーションは ARG に基づいてトラフィックのルールを調整することが可能である。

アルゴリズム 1 は、End.AN.NF がパケットを netfilter のフックポイントに渡す方法を示した擬似コードである。まず、End.AN.NF は、ARG の長さがこの End.AN.NF SID に指定されている場合、受信したパケットの宛先アドレスから ARG 値を抽出する。抽出された ARG 値は、マークとしてパケットバッファに付加される。次に、End.AN.NF はパケットバッファの先頭を外側の SRH から内側のパケットに切り替え、バッファを netfilter フックに渡す。フックにインストールされたルールが内側のパケットに適用された後、End.AN.NF は、パケットバッファの先頭を内側のパケットから外側の SRH に復元し、パケットを次のプロセスに渡す。この手順は、図 4.1 の赤い長方形で示した 3 つのフックポイント、prerouting、forward、postrouting に対してそれぞれ適用する。

Linux カーネルは、End ビヘイビアを特定の SID を宛先とするルーティングテーブルエントリとして扱う。End.AN.NF は End ビヘイビアの 1 つであるため、その SID も同様にルーティングテーブルエントリにインストールされる。図 4.2 に示すように、カーネルは他の End ビヘイビアと同様に End.AN.NF を表す SID をルーティングテーブルエントリとして扱っていることが確認できる。ルーティングソフトウェアや iproute2 を用い

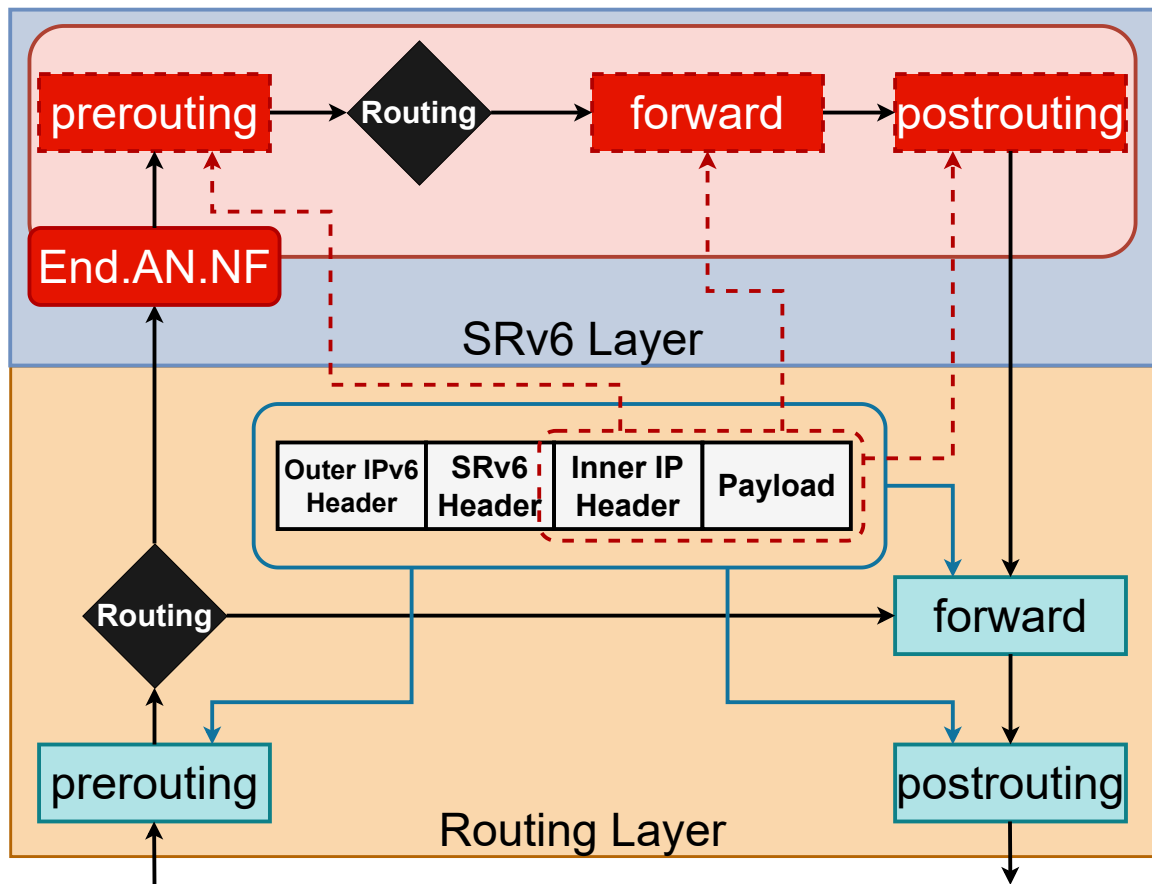


図 4.1: Linux カーネルネットワークスタック内における End.AN.NF の動作の流れ

```
$ ip -6 route | grep End
2001:db8:1::/96 encap seg6local action End.AN.NF arglen 32 dev eth0 metric 1024 pref medium
2001:db8:2::200 encap seg6local action End dev eth0 metric 1024 pref medium
2001:db8:3::300 encap seg6local action End.DX4 nh4 192.168.99.1 dev eth1 metric 1024 pref medium
```

図 4.2: 改良した Linux kernel が End.AN.NF の SID を IPv6 ルーティングエントリとして扱っている様子

て SID をルーティングテーブルエントリとして追加すると、従来のルーティングプロトコルを用いてカーネルのルーティングテーブルにインストールされた経路を広告することが可能となる。実際に Linux 用のソフトウェアルータ実装である FRRouting [22] を使用し、カーネル内の End.AN.NF に関連付けられた SID を BGP 経由で他のルータに IPv6 経路として広告できることを確認した。End.AN.NF のアーキテクチャは、ルーティング制御に既存のルーティングプロトコルを使用できるため、既存の SF アプリケーションとの互換性が高い。このアーキテクチャは、Linux netfilter を用いた SR-aware SF の実現方法の 1 つである。

Algorithm 1 End.AN.NF がパケットを netfilter フックポイントへ通過させる処理の擬似コード

```

1: function PASSPACKETTOHOOK(packet)
2:   if the length of ARG is specified for this End.AN.NF SID then
3:     Extract the ARG value from the destination address of outer SRH
4:     Mark the ARG value on the packet buffer packet
5:   end if
6:   Switch the head of packet buffer packet from the outer SRH to the inner packet
7:   Pass packet to a netfilter hook
8:   Switch the head of packet buffer packet from the inner packet to the outer SRH
9: end function

```

4.3 実装

End.AN.NF は Linux カーネルで動作する SRv6 ビヘイビアである。End.AN.NF を実装するためには、Linux カーネルの SRv6 の実装を理解する必要がある。本セクションでは、End.AN.NF 自体の実装を解説しながら Linux カーネルにおける SRv6 の実装についても述べる。また、本論文では linux-5.15.106 を対象として End.AN.NF を実装する。ビルド時のカーネルコンフィグを付録としてとして本論文末尾に掲載する。本提案手法実装に利用した Linux フレーバーは、ubuntu のミニマムカスタムイメージである。カーネルコンフィグを編集し、動作に必要なネットワークドライバや VRF カーネルモジュールなどを有効にしている。また、本実装の動作検証はコンテナ技術を利用して仮想的なトポロジを作成して行った。コンテナを動作させるにあたり、mobyproject [33] の提供するカーネルコンフィグのチェックツール [34] を利用した。また、オーディオや GPIO サポートなど、本論文の提案手法の実装及び動作検証、計測に必要な項目は無効化している。

4.3.1 Linux における SRv6 ビヘイビアの実装

Linux における SRv6 End ビヘイビア や End.DT4 ビヘイビアなどの実装は主に、net/ipv6/seg6local.c に記述されている。章 3.1 で述べたように、Linux はカーネルモジュールという仕組みを使うことで Linux のカーネルのコードそのものを書き換えなくても機能を追加実装することができる。しかし、seg6local の実装についてはカーネルモジュールとして提供されておらず、Linux カーネルのソースコード内で直接 SRv6 ビヘイビアを実装する手法が取られているため、直接 Linux カーネルのプロトコルスタックの実装を変更する必要がある。

独自の SRv6 ビヘイビアを追加するためには、まず net/ipv6/seg6local.c で定義されている seg6_action_table の末尾に要素を追加する必要がある。End.AN.NF を実装するために追加した記述を、ソースコード 4.1 として実際のコードを抜粋したものを示す。この seg6_action_table は seg6_action_desc 構造体の配列である。seg6_action_desc のフィールドについて、特筆すべきは input フィールド及び slwt_ops フィールドである。受信したパケットの宛先アドレスが自身の持つルーティングテーブル上で、1 つの

SRv6 ビヘイビアとして表現されていた場合、そのパケットは Linux ネットワークスタックの SRv6 レイヤへ到達し、パケットは SRv6 ビヘイビア毎に固有の処理の実装に渡される。input フィールドには、最初に渡される SRv6 ビヘイビア毎に固有の関数へのポインタが代入される。ソースコード 4.1 では input フィールドに `input_action_end_nf` 関数へのポインタが設定されている。この関数の実装については以降で解説する。slwt_ops フィールドには、SID と SRv6 ビヘイビアを経路情報としてルーティングテーブルに設定するときに呼びされる関数へのポインタが代入される。例えば End.DT4 の実装では、slwt_ops フィールドにはパケットから SRH をデカプセル化したあとにルックアップする VRF を指定するための処理が定義された関数が設定されている。End.AN.NF の場合、SID の IPv6 アドレスの下位何 bit を ARG として利用するかを指定するための処理が定義された関数である `seg6_end_nf_build` へのポインタが設定されている。

End.AN.NF は、SID 内部の ARG 部をパケットバッファに埋め込んだ上で、SRH でカプセル化された内部パケットに対して netfilter フックポイントを通過させる。これらの処理は `input_action_end_nf` 関数内で行われる。ソースコード 4.2 に、`input_action_end_nf` 関数内で SID 内部の ARG 部をパケットバッファに埋め込む部分の処理を示す。slwt は `seg6_local_lwt` 構造体へのポインタであり、これは `input_action_end_nf` 関数呼び出し時に引数として渡される。seg6_local_lwt 構造体には SRv6 ビヘイビアの動作に必要な様々なフィールドが定義されている。パケットが入ってきたインターフェースを表す数値や、End.DT4 などを使うためのルーティングテーブル ID などが含まれている。End.AN.NF の実装のために、seg6_local_lwt 構造体へ `_u8` 型の `arg_len` というフィールドを追加した。このフィールドは SID の ARG 部分の長さを示しており、このフィールドはソースコード 4.1 の slwt_ops フィールドに設定された `seg6_end_nf_build` 関数によって設定される。mark の計算及びパケットバッファへの埋め込みは、ARG が定義されているときにのみ行う。End.AN.NF において、ARG フィールドの利用は任意である。ARG を利用してパケットバッファにマークを付ける必要がない場合、SID を定義する際に ARG の長さを 0 とすることで ARG は無効になる。なお、ARG の長さを負の値にすることはできない。Linux では、ユーザ空間からカーネルが管理するルーティングテーブルにエントリを追加する際、NETLINK メッセージでやり取りをする。End.AN.NF の SID を経路表に追加する際、特定のフォーマットで NETLINK メッセージを作成する。そのメッセージの中には ARG の流さを指定するフィールドが定義されており、受け取ったメッセージは `seg6_end_nf_build` 関数内でバリデーションされ、ARG の長さが負だった場合は経路情報としてルーティングテーブルに載らない。ソースコード 4.2 では、`arg_len` が 0 でない、すなわち ARG が有効であるときに if 文内部の処理が実行される。計算された結果は Linux 上のパケットバッファを示す `sk_buff` 構造体の mark フィールドに設定される。

`input_action_end_nf` 関数の中で、SRH でカプセル化されたパケットを実際に netfilter フックポイントへ通過させている処理を抜粋したコードを、ソースコード 4.3 として示す。End.AN.NF は、SRv6 パケットの SRH 部分を隠蔽して netfilter にパケットを通す。ソースコード 4.3 の中で、SRH を隠蔽する、という処理は `skb_pull` 関数と `skb_reset_network_header` 関数の呼び出しによって実現される。先に述べた通り、Linux カーネルではパケットバッファを `sk_buff` 構造体で管理している。Linux カーネルでパ

ケットバッファを参照して各ネットワークレイヤでパケット転送処理を実行する際、処理ごとに `sk_buff` 構造体内部の `head` ポインタを進める必要がある。 `head` ポインタは、パケットバッファの中で現在処理をしているポインタの位置を示すものである。例えば、Ether レイヤの処理をしているときはこのポインタは Ether フレームの先頭を指す。Ether レイヤの処理が終わったあとは、 `head` ポインタの位置を次のレイヤ、カプセル化されていない一般的なパケットであれば IP レイヤへずらす。通常このように `head` ポインタの位置を進める場合は、 `skb_pull` 関数を利用する。 `skb_pull` 関数の第一引数は `sk_buff` 構造体へのポインタであり、第二引数はどれだけ進めるかを整数値で渡す。ソースコード 4.3 では、第二引数に `offset` という変数を渡している。この変数には、予め SRH の先頭から内部パケットのヘッダまでの長さを計算して代入してある。 `skb_pull` 関数の呼び出し後は、 `skb_reset_network_header` 関数を使用することで、IP レイヤのヘッダ位置を再度アップデートする。

ソースコード 4.3 の 9 行目から 11 行目は実際に SRH でカプセル化された内部パケットを prerouting フックポイントへ通過させている処理である。 netfilter フックポイントへの通過は、 `NF_HOOK` マクロを呼び出すことで実現できる。 netfilter フックポイントを通過させた後は `skb_push` 関数を呼び出しており、この関数は `skb_pull` 関数とは対比的に指定した分 `head` ポインタを前に戻す関数である。 `skb_push` 関数の呼び出し後は、 `skb_pull` 関数呼び出し時と同様に `skb_reset_network_header` 関数を呼び出してヘッダの位置をもとに戻している。

ソースコード 4.3 の 19 行目、及び 21 行目では、SRv6 End ビヘイビアに対応する処理を行っている。 `advance_nextseg` 関数は、 `segleft` をデクリメントして宛先アドレスを新たな SID で書き換える処理を行う関数である。また、 `seg6_lookup_nexthop` 関数では、新たな SID で書き換えた宛先アドレスに対するネクストホップを決定している。SRv6 End ビヘイビアが行う転送処理はこの大きく分けてこの 2 つであり、この転送処理は一般的なパケット転送処理とは異なる。そのため、SRH でカプセル化された内部パケットにとってのフォワード操作、 netfilter の forward フックポイントを適用するタイミングには議論の余地がある。本論文では、 `segleft` のデクリメントと新たな SID による宛先アドレスの更新、及び新たな宛先アドレスのネクストホップの決定を、SRH でカプセル化された内部パケットにとってのフォワード操作として解釈し実装する。

ソースコード 4.3 の 26 行目、及び 27 行目では、ソースコード 4.2 と同じように ARG の値をパケットバッファのマークフィールドに設定している。パケットバッファのマークフィールドは、End.AN.NF に限らず、汎用的に利用されるフィールドである。汎用的であるため、 netfilter-based アプリケーションからその値を参照して処理内容を変えることができる。ただし、その反面汎用的であるがゆえに他の用途で利用されたり、値が書き換わったりすることがある。よって、ここでは forward フックポイントを通過する前に再度マークを付け直している。パケットを forward フックポイントへ通過させる処理以降は、ほとんど同じ処理でパケットを同様に postrouting フックポイントへ通過させる。

ソースコード 4.3 に示すように、SRv6 パケットに対して End.AN.NF を使って SRH でカプセル化された内部パケットを netfilter フックポイントへ通過させる処理は非常に単純である。処理の殆どがポインタの加算及び減算になるように考慮しており、オーバー

ヘッドがなるべく小さくなるようにしている。

ソースコード 4.1: seg6_action_table へ End.AN.NF の定義を追加する実装

```

1 static struct seg6_action_desc seg6_action_table[] = {
2     .
3     .
4     .
5     // その他のビヘイビアの定義
6     {
7         .action      = SEG6_LOCAL_ACTION_END_NF,
8         .attrs       = SEG6_F_ATTR(SEG6_LOCAL_NF),
9         .optattrs    = SEG6_F_LOCAL_COUNTERS,
10        .input       = input_action_end_nf,
11        .slwt_ops    = {
12            .build_state = seg6_end_nf_build,
13        },
14    },
15 };

```

ソースコード 4.2: ARG の値をパケットバッファに付加する処理の実装

```

1 static int input_action_end_fw(struct sk_buff *skb,
2                               struct seg6_local_lwt *slwt)
3 {
4     .
5     .
6     .
7     if (slwt->arg_len) {
8         memcpy(&daddr_segment, &outer_header->daddr.s6_addr32[3],
9                sizeof(daddr_segment));
10        arg = ntohl(daddr_segment);
11        mask = (1UL << slwt->arg_len) - 1;
12        arg &= mask;
13        skb->mark = arg;
14    }
15    .
16    .
17 }

```

ソースコード 4.3: SRv6 の内部パケットを netfilter フックポイントへ通過させる処理の実装

```

1 static int input_action_end_fw(struct sk_buff *skb,
2     struct seg6_local_lwt *slwt)
3 {
4     .
5     .
6     .
7     skb_pull(skb, offset);
8     skb_reset_network_header(skb);
9     ret = NF_HOOK(NFPROTO_IPV4, NF_INET_PRE_ROUTING,
10         dev_net(skb->dev), NULL, skb, skb->dev,
11         skb_dst(skb)->dev, dummy_okfn);
12
13     skb_push(skb, offset);
14     skb_reset_network_header(skb);
15
16     if (ret != 1)
17         return ret;
18
19     advance_nextseg(srh, &ipv6_hdr(skb)->daddr);
20
21     seg6_lookup_nexthop(skb, NULL, 0);
22
23     skb_pull(skb, offset);
24     skb_reset_network_header(skb);
25
26     if (slwt->arg_len)
27         skb->mark = arg;
28     ret = NF_HOOK(NFPROTO_IPV4, NF_INET_FORWARD,
29         dev_net(skb_dst(skb)->dev), NULL, skb, skb->dev,
30         skb_dst(skb)->dev, dummy_okfn);
31     if (ret != 1) {
32         skb_push(skb, offset);
33         skb_reset_network_header(skb);
34         return ret;
35     }
36
37     if (slwt->arg_len)
38         skb->mark = arg;
39
40     ret = NF_HOOK(NFPROTO_IPV4, NF_INET_POST_ROUTING,
41         dev_net(skb->dev), NULL, skb, skb->dev,
42         skb_dst(skb)->dev, dummy_okfn);
43
44     skb_push(skb, offset);
45     skb_reset_network_header(skb);
46
47     if (ret != 1)
48         return ret;
49
50     return dst_input(skb);
51     .
52     .
53     .
54 }

```

第5章 評価

本論文で実装した End.AN.NF の性能を評価するために、4つの実験を行った。本章では、3つの計測実験で得られた結果から、提案手法が実用上十分なスループット性能を持っているか、及び実用的なレイテンシに収まっているのかを確認するために Linux に実装されている既存のパケット転送メカニズムと比較し評価する。このうち2つはスループットについて、もう1つはレイテンシについて焦点を当てたものである。最初の実験では、パケットサイズに基づくスループットを計測し、2つ目の実験では、netfilter-based アプリケーションにおけるフィルタールール数を増加させた際のスループットの変化を評価した。3つ目の実験では、異なるパケット転送メカニズムに関連するレイテンシを計測した。本章では更に、計測用パケットの送信に利用したトラフィックジェネレータ、及び評価の際に考慮したレシーブサイドスケールリングについても解説する。また、計測時に netfilter-based アプリケーションとして利用した nftables についても同様に解説する。

5.1 計測の概要と予想

End.AN.NF の性能を、3つの転送メカニズムと比較する。比較対象は、End、End.DT4 と H.Encaps の組み合わせ、及び IPv4 である。IPv4 は Linux のパケットフォワーディング性能におけるベースラインとして参照する。図 4.1 に示すように、End.AN.NF が動作する場合、受信パケットは End と比較して2倍の数のフックポイントを通過する。したがって、End.AN.NF の性能は End に劣ることが予想される。一方で、End.AN.NF の性能は End.DT4 と H.Encaps の組み合わせよりも高いと予想される。

SRv6 でカプセル化されたパケットに netfilter のルールを適用する場合、バニラ Linux カーネルでの実用的なアプローチは End.DT4 と H.Encaps の組み合わせである。本論文執筆現在、Linux のメインラインには章 3.2.2 で示したような End.AS や End.AD などの SR-Proxy は実装されていない。本論文では、End.AS や End.AD などの SR-Proxy のかわりにバニラの Linux カーネルで動作する End.DT4 と H.Encaps の組み合わせを End.AN.NF の比較対象とする。図 5.1 に、End.AN.NF と End.DT4 と H.Encaps の組み合わせの動作の違いを示す。End.AN.NF も End.DT4 と H.Encaps の組み合わせも、どちらも SRv6 パケットとして受信したパケットの内部を netfilter フックポイントへ通過させることができる。章 4.3.1 で述べたように、End.AN.NF はパケットバッファ内で IP ヘッダを指し示す部分のポインタを操作することで SRH を隠蔽し、内部パケットを netfilter フックポイントへ通過させる。対して、End.DT4 と H.Encaps の組み合わせでは、End.DT4 が SRH を一度取り外し、netfilter を適用してから再度 H.Encaps でカプセ

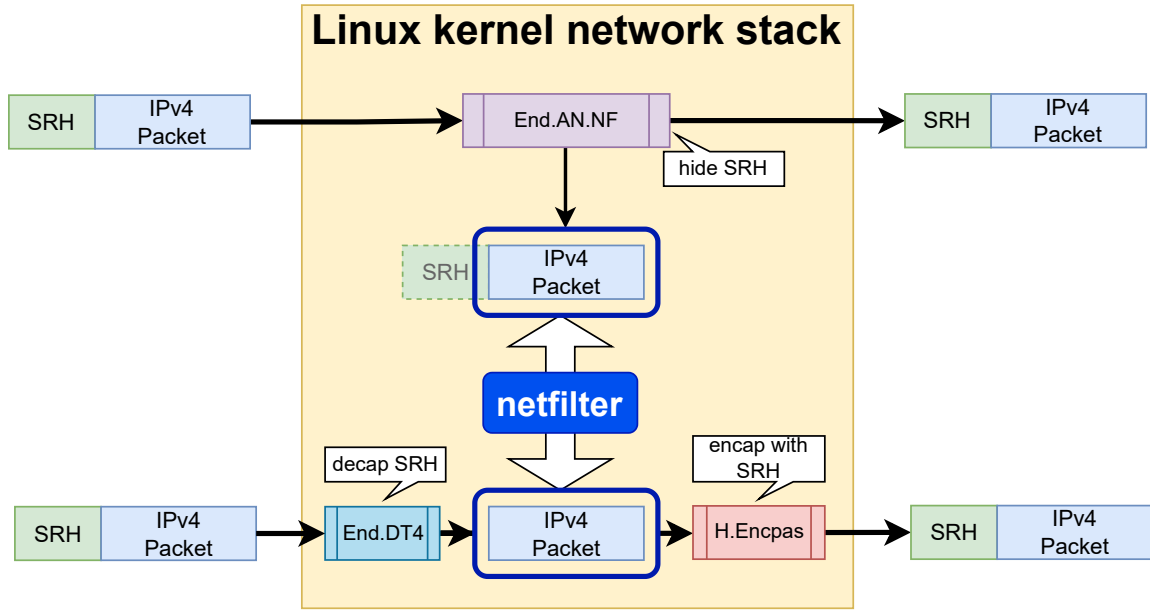


図 5.1: End.DT4 と H.Encaps の組み合わせ と End.AN.NF のパケット処理プロセスの差

ル化を行う。Linux カーネルのネットワークスタック的には、End.DT4 がデカプセル化を行うと、そのパケットは VRF で受信される。VRF で受信されたパケットは、一般的なパケットと同様に netfilter を通過する。そして、そのパケットの宛先アドレスを VRF 上でルックアップすると、経路情報として H.Encaps によって再度カプセル化されるように記述されているため、H.Encaps によってパケットはもう一度カプセル化される。つまり、End.DT4 と H.Encaps の組み合わせでは End.DT4 によるデカプセル化処理、デカプセル化されたパケットの VRF での受信、H.Encaps によるカプセル化のオーバーヘッドが存在する。したがって、このオーバーヘッドが性能の低下につながる事が予想されるため、End.AN.NF の性能は End.DT4 と H.Encaps の組み合わせよりも優れていると予想できる。

3 つの実験はすべて同じ構成、同じ環境で行った。マシンの構成を、表 5.1 に示す。100Gbps のリンクで直結された 2 台のマシンを用意した。2 台のマシンは同一仕様で、CPU には Intel(R) Xeon(R) Silver 4310 12 コア x2、メモリは 64GB DDR4-2666、NIC には Intel E810 100Gbps を搭載している。CPU のハイパースレディング機能は無効に設定した。1 台はトラフィック・ジェネレータとして、もう 1 台は System Under Test (SUT) として使用する。トラフィック生成マシンには Ubuntu 22.04 と TRex [35] をインストールし、テストトラフィックの生成に使用した。一方、SUT マシンには End.AN.NF を実装した Linux カーネル 5.15.106 をインストールし、End.AN.NF と、End.AN.NF の SID を設定するために独自に拡張した iproute2 コマンドを実装した。また、2 台のマシン間のリンクには 2 つの VLAN を設定し、テストトラフィックを送信するためのリンクと End.AN.NF 動作後に送信されるトラフィックが論理的に別のリンクになるようにした。VLAN は tag 付きで送信し、Linux カーネルのネットワークスタックが tag をほどく。

表 5.1: 実験に使用したマシンの構成

	トラフィックジェネレータ	System Under Test
カーネル	5.15.0-79-generic	customized 5.15.106
インストールツール	TRex v3.03	customized iproute2
CPU	Intel(R) Xeon(R) Silver 4310 12 コア x2	
メモリ	64GB DDR4-2666	
NIC	Intel E810 100Gbps	
OS	Ubuntu 22.04.2 LTS	

5.2 TRex

本論文では、スループット及びレイテンシの計測に TRex を利用した。TRex は Cisco System によって開発されたトラフィックジェネレータである。TRex は DPDK [36] というライブラリを使って開発されている。

DPDK はパケット処理をカーネル空間で行わない。Linux カーネルにはパケット転送メカニズムが実装されている。FRR などの Linux ソフトウェアルータ実装は、動作するルーティングプロトコル群がユーザ空間で動作し、NETLINK メッセージを通じて経路情報がカーネルのルーティングテーブルにインストールする。そして FRR は実際のパケット転送処理を Linux カーネルにまかせている。対して、図 5.2 に示すように DPDK では NIC で受信したパケットはカーネルをバイパスし、ユーザ空間で操作する DPDK ソフトウェアに渡される。よって、DPDK のパケット処理性能は Linux カーネルに実装されているパケット処理メカニズムの性能に依存しない。また、DPDK は CPU を独占する。DKDP アプリケーションは動作中、指定された CPU へポーリング常にを行う。これにより、コンテキストスイッチングを抑制して高速な処理を行うことができる。

TRex は柔軟なトラフィック生成が可能である。最も基本的な使い方は、元となるパケットキャプチャファイルを用意し、トラフィックごとに変更する部分を別途 yaml ファイルで定義するという方法である。この yaml ファイルには、例えば送信元 IP アドレスや宛先 IP アドレスを定義することができる TRex はこのファイルの内容に従って、元となるパケットキャプチャファイルの情報を変更し、パケットキャプチャファイルとは別の送信元 IP アドレスや宛先 IP アドレスを持つパケットを生成し、送信することができる。また、どれだけの時間、単位時間あたりにどれだけのパケットを送出するのかといったことも yaml ファイルに定義できる。

また、パケットキャプチャファイルを利用する方法以外にも、Python スクリプトを使ってパケットを生成し送出することができる。trex.stl.lib.api という Python ライブラリに様々な API が提供されているこのライブラリには SRv6 パケットを生成する関数も提供されており、本論文では、この Python スクリプトを使ってトラフィックを生成する手法でパケットを生成し計測した。本論文での計測実験では、IPv4 パケットを特定の SRH でカプセル化する。一部の実験時にはレシーブサイドスケーリングの仕組みを効率的に使うため、カプセル化する IPv4 パケットの送信先アドレスをインクリメントした。TRex はこのように、SRv6 のパケットの生成、及び内部パケット情報の操作も可能である。

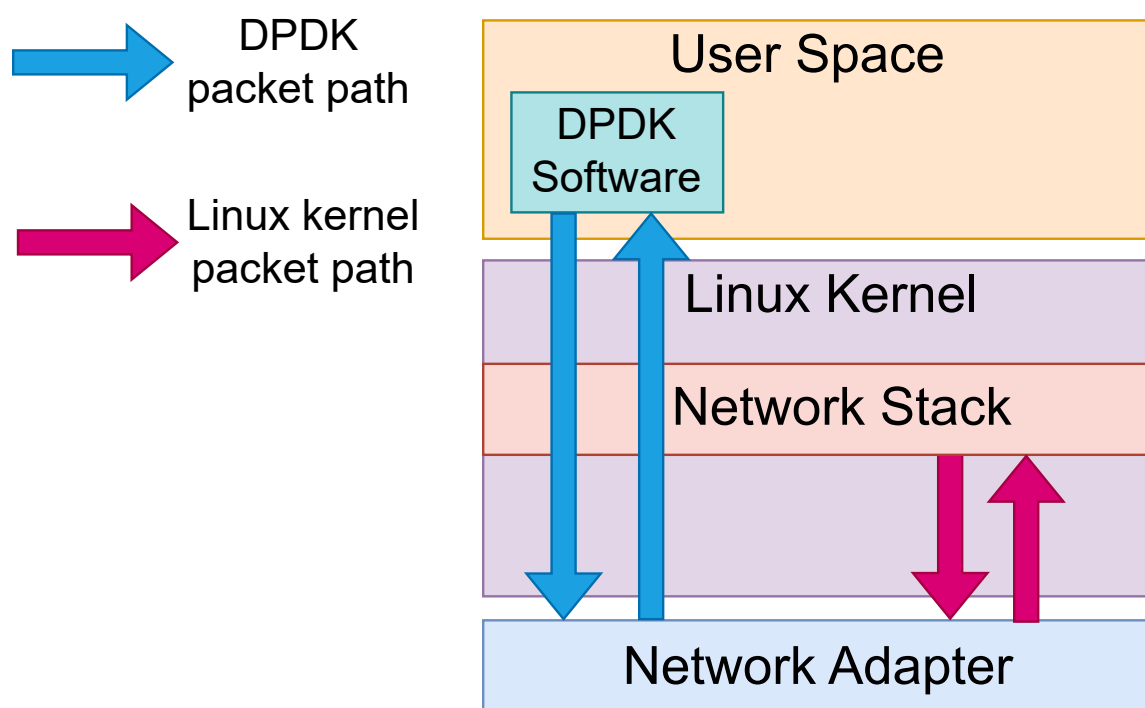


図 5.2: DPDK アプリケーションの動作概要

5.3 レシーブサイドスケーリング (RSS)

レシーブサイドスケーリング (RSS) とは、マルチコアプロセッサを搭載したシステムにおいて、パケットの受信処理を複数の CPU コアに分散させる技術である。これにより、CPU コアの負荷が均等に分散され、スループットの向上が期待できる。図 5.3 に示すように、RSS では受信したパケットからハッシュを計算し、その値をもとに RX キューを分散させる。RSS の実装は NIC のドライバに依存する。ハッシュの計算アルゴリズムは NIC のドライバの実装によって異なる上、パケットのどの部分からハッシュを計算するかも異なる。

一般的な SRv6 ネットワークではパケットを SRH でカプセル化するノードの数は限られるため、ドライバの実装が SRH の送信元及び宛先アドレスをキーにしてハッシュを計算する手法を取っている場合、ハッシュの値が偏ってしまう。本論文では、計測対象のパケットは SRv6 パケットである。SRv6 パケットは SRH でカプセル化されている。一般的な SRv6 パケットの送信元アドレスにはパケットを SRH でカプセル化するノードのループバックアドレスが割り当てられ、宛先アドレスには次の SRv6 ノードの SID が割り当てられる。つまり、SRv6 ビヘイビアを実行するノードが受け取るパケットの送信元アドレスは SRH でカプセル化するノードのループバックアドレスであり、宛先アドレスは自分自身の SID である。

本計測では、SRv6 パケットに対して RSS を有効化する必要がある際は、SRH でカプセル化された内部パケットの宛先アドレスを変更する。本計測で利用した環境では、NIC として E810 を利用した。実際にパケットキャプチャをして検証した結果、E810 では SRv6

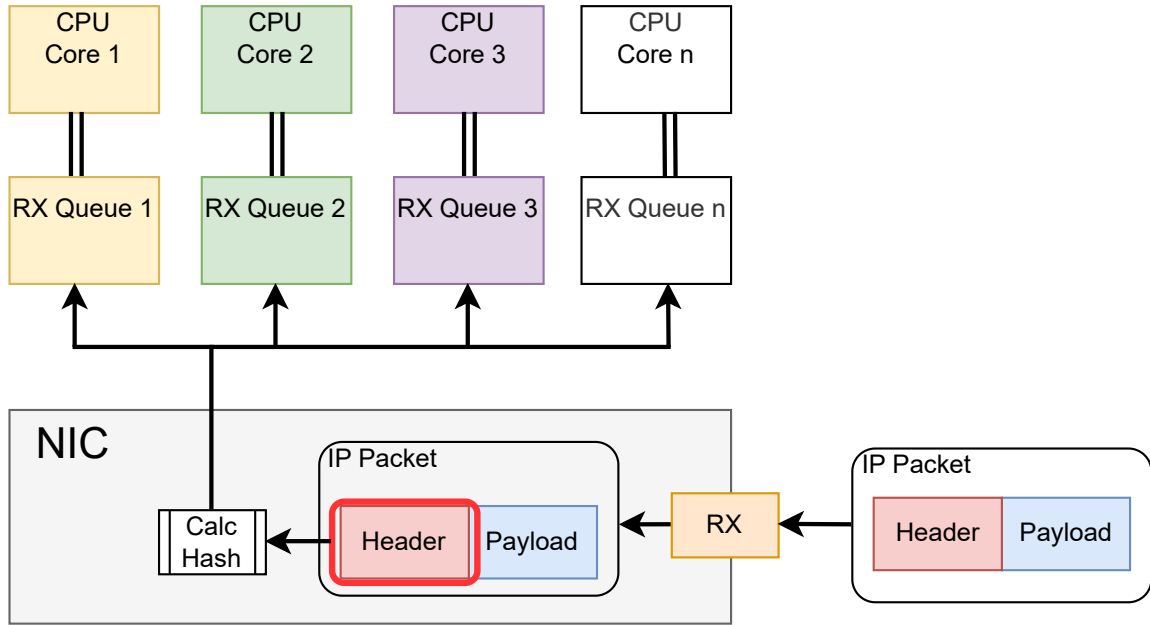


図 5.3: レシーブサイドスケーリングの動作概要

パケットに対して RSS を適用するためには内部パケットの宛先アドレスを変更すれば良いことがわかった。SRH でカプセル化された内部パケットの宛先アドレスは、そのパケットの送信元が実際に通信するエンドノードのアドレスである。したがって、実運用上、SRH でカプセル化された内部パケットの宛先アドレスがバラバラになることは自然なシナリオである。

5.4 パケットサイズ毎のスループット性能

End.AN.NF, End, IPv4, 及び End.DT4 と H.Encaps の組み合わせについて、パケットサイズを増加させながらスループットを測定した。この実験により、各パケット転送メカニズムにおけるパケットサイズによるスループットの変化が明らかになった。この実験では、netfilter フックポイント通過時に適用されるルールはからの状態で実施した。よって、netfilter のフックポイントは通過するものの、実際にパケットにフィルタやマングルルールが適用されることはない。End.AN.NF の、End に対するスループットの低下、及び End.DT4 と H.Encaps の組み合わせに対する性能の向上を評価した。

5.4.1 計測内容

トラフィック生成マシンで TRex によって生成されたトラフィックを、最小パケット長 126 バイトから最大パケット長 1518 バイトまでパケットサイズを変化させ、SUT マシンに送信した。測定時のパケット長は次のように計算した: $l = 174n + 126$ 。ここで l はパケット長、 n は測定回数である。 $n = 0$ から $n = 10$ まで、合計 10 回の測定を行った。

最小パケット長として 126 バイトを選択した理由は、SID リストの長さが 2 である際のタグ付き VLAN を持つ UDP パケットの最小長が 126 バイトだからである。End.AN.NF は、パケットの segleft をデクリメントするため、SID リスト長は少なくとも 2 である必要がある。これは SID リストの長さが 1 の場合、segleft は 0 から始まり、End.AN.NF でデクリメントすると負の値になってしまうからである。一方、End.DT4 は、segleft が 0 であることを必要とする。End.DT4 は SRv6 ネットワークの終点で SRH をでカプセル化するビヘイビアである。つまり、End.DT4 が動作するのは SID リストによって指定された最後のノードであるため、segleft はそれ以上デクリメントできない 0 である必要がある。そこで、End.DT4 と H.Encaps の組み合わせの測定では、TRex は SID リスト長が 2 のパケットを生成し、segleft を 0 に設定した。また、レシーブサイドスケーリング (RSS) の仕組みを効果的に使用するため、TRex でパケットを生成する歳に内側の IPv4 パケットの宛先アドレスと送信元アドレスの両方をインクリメントした。IPv4 パケットの計測の際は、SRv6 パケット長に合わせて UDP ペイロードにダミーデータを埋め込み、最小パケット長が 126 バイトから始まるようにした。End.DT4 と H.Encaps の組み合わせの計測と同様、RSS を効果的に活用するため、パケット生成時に宛先アドレスと送信元アドレスをインクリメントした。最大パケット長については、タグ付き VLAN ヘッダを含むイーサフレームの最大サイズが 1518 バイトであることから、今回の測定ではパケットサイズの上限を 1518 バイトに設定した。

5.4.2 評価

図 5.4 に、この実験の結果を示す。End.AN.NF のスループットは、すべてのパケット長において End と比較して 6% 以上の低下は見られない。パケット長が 1518 バイトのとき、End.AN.NF は End と比べた際のスループットの低下が最も少なく、その低下は約 1.7% である。対して、パケット長が 478 バイトのとき、End と比較した際の End.AN.NF のスループットの低下は最も大きく、その低下は約 5.6% である。パケット長とスループットには相関がなく、大きな変動が見られた。このスループットの低下は、End.AN.NF のパケットが End のパケットに比べて 2 倍の netfilter のフックポイントを通過することが原因として挙げられる。ただし、そのスループット低下のレベルは許容範囲内に留まっている。

End.AN.NF のスループットを End.DT4 と H.Encaps の組み合わせと比較した場合、End.AN.NF は予想通り、パケット長に関係なく一貫して優れた性能を示している。具体的には、End.AN.NF は End.DT4 と H.Encaps の組み合わせよりも、最大で 26.7% 高いスループットを達成している。グラフから、End.AN.NF と End.DT4 と H.Encaps の組み合わせとの間のスループットの差はパケット長の影響を受けていることが読み取れる。短いパケットでは相対的な性能格差が大きくなり、長いパケットではその差は縮まる。パケットサイズが小さくなるにつれて、1 秒あたりのパケット転送レート (pps) は増加する。結果として、パケットサイズが小さいほど、パケット転送のオーバーヘッドが顕著になる。

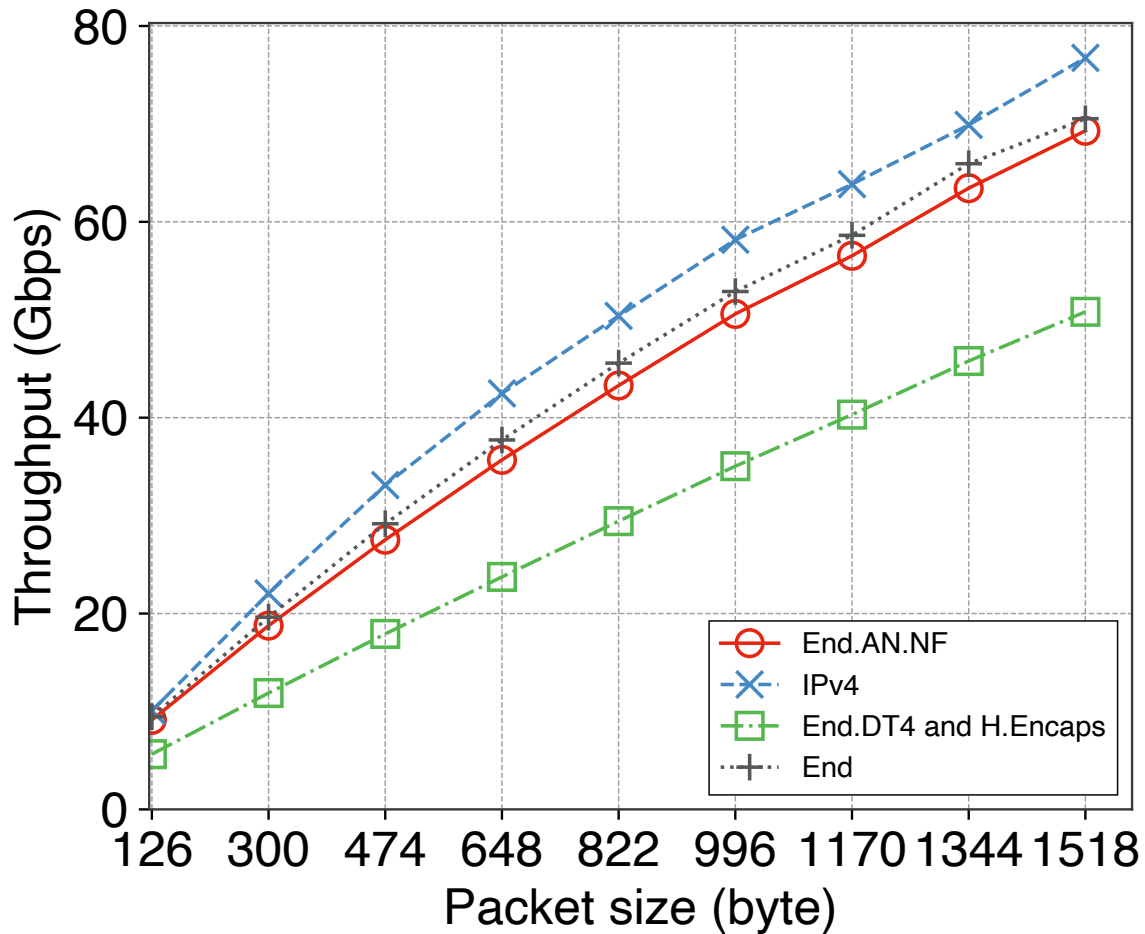


図 5.4: SRv6 End ビヘイビア毎, 及び IPv4 のスループット

5.5 netfilter にインストールされたルール毎のスループット

次に, End.AN.NF, IPv4, および End.DT4 と H.Encaps の組み合わせについて, netfilter にインストールするフィルタールールの数を変更しながらスループットを測定した. フィルタールールのインストールには, netfilter-based アプリケーションとして nftables を使用した. nftables では, ルールはチェーンの集合として表現され, チェインにはベースチェーンとレギュラーチェーンの 2 種類がある. nftables は, 他のチェーンがレギュラーチェーンを参照している場合のみ, レギュラーチェーンを使用する. 実験では, チェインの種類ごとにカウントを増やしながらスループットを測定した. パケット転送メカニズムに関わらず, フィルタールールの追加によりスループットが低下することが予想される. この実験は, フィルタールールによる各パケット転送メカニズムのスループット低下の特徴を明らかにすることを目的とする.

5.5.1 nftables

本セクションでは、nftables について解説する。nftables 公式 wiki [37] によると、nftables とはモダンな Linux カーネル向けのパケット分類フレームワークだという。nftables を使うと、パケットのフィルタリングや NAT, NAPT やその他のパケットマングリングを適用することができる。対象は自身宛のパケットだけでなく、自信がトランジットとなる通信や自身が送出するパケットなど様々な種類のパケットに対してルールを適用できる。nftables は iptables の後継フレームワークであり、iptables の抱えるいくつかの問題点を改善したフレームワークである。例えば、nftables のルール定義文法は iptables のルール定義文法に比べて、より構造化されている。自身の 80 番ポートに対する tcp 通信を拒否する、というフィルタールールを定義する設定ファイルを、nftables と iptables それぞれについて確認する。nftables 向けの定義ファイルをソースコード 5.1 に、iptables 向けの定義ファイルをソースコード 5.2 に示す。

nftables では、ルールセットの塊を `table` という単位で管理する。この `table` には予約されている特定の語句や記号を除き、任意の名前で定義することができる。テーブルを定義する際は、`table family table_name` のフォーマットで記述する。`family` には、IPv4 パケットを対象とするテーブルには `ip` が、IPv6 パケットが対象であれば `ip6` が、両方が対象であれば `inet` が入る。対して、iptables は IPv4 と IPv6 のルールを同じ枠組みで書くことができない。iptables は IPv4 向けのユーティリティであり、IPv6 には ip6tables を利用する必要がある。

ソースコード 5.1 2 行目では、`deny8080` という名前でチェーンが定義されているチェーンはテーブルの中に任意に数定義することができ、そのフォーマットは `chain chain_name` である。ソースコード 5.1 3 行目では、ネットフィルタフックポイントに登録する際に必要な情報が定義されている。`type filter` はチェーンのルールがフィルターであることを表している。`hook input priority filter` は、「フィルタールールのデフォルトの優先度で、netfilter の input フックポイントに対象のチェーンに登録する」ということを示している。図 3.1 で示したように、input フックポイントは通信のエンドポイントが自分自身である際に通過するフックポイントである。ソースコード 5.1 4 行目では、実際のルールが定義されている。通信プロトコルが `tcp` であり、宛先ポートが 8080 番であればドロップされる。

対して iptables では、実質的なルールの定義はソースコード 5.1 の 1 行目及び 2 行目だけである。`*filter` は対象がフィルタールールであることを示し、`-A INPUT` でルールを input フックポイントに設定することを示し、以降は宛先ポートが 8080 番の tcp 通信を拒否するように書かれている。iptables では、ルールの定義をトランザクションとして管理するため、定義しただけでは変更が反映せれず、`COMMIT` キーワードで明示的に変更の反映を示す必要がある。

nftables のルール定義は、iptables のルール定義に比べて記述量が多くなる傾向がある。一方で、nftables のルール定義は iptables のルール定義に比べて宣言的で構造化されているため、ルール数が増えたときに人間が読んで理解しやすいフォーマットになっている。2 つのルールの定義方法は違うが、現在の Linux では、iptables でルールを定義すると内

部的には nftables のルールに変換される。これは nftables の方が従来の iptables よりも性能面でのアドバンテージが大きいからである。

チェーンにはベースチェーンとレギュラーチェーンの 2 種類が存在し、nftables はひとつひとつのルールをチェーンという単位で管理する。ベースチェーンとはソースコード 5.3 の 10 行目から 14 行目までで定義されている `filter_rule` チェインを指す。ベースチェーンを宣言すると、そのチェーンに対応するルールが netfilter フックポイントに設定されるため、ソースコード 5.3 の `filter_rule` チェイン内のルールは、自身が IPv4 パケットの転送動作をする際は必ず適用される。一方で、ソースコード 5.3 中の `remote_access` チェインと `rate_limit` チェインはレギュラーチェーンと呼ばれるチェーンである。レギュラーチェーンには、ソースコード 5.3 11 行目のようなルールのタイプと netfilter フックポイントを指定する項目がない。レギュラーチェーンは、定義しただけではどの netfilter フックポイントにも結び付けられず、実行されない。レギュラーチェーンは他のチェーンからの参照によってのみ実行される。例えば、ソースコード 5.3 の 12 行目、13 行目では、パケットバッファのマークフィールドに特定の値が値が代入されていることを条件とし、`goto` キーワードを使って指定したチェーンを呼び出している。このように、レギュラーチェーンは `goto` キーワードで呼び出されて初めて実行される。ソースコード 5.3 ではベースチェーンからレギュラーチェーンを呼び出しているが、レギュラーチェーンはその処理の中で別のレギュラーチェーンを呼び出すこともできる。つまり、レギュラーチェーンは定義しただけでは実行されず、他のベースチェーンまたはレギュラーチェーンから参照されて初めて実行される種類のチェーンである。

ソースコード 5.1: nftables におけるシンプルなフィルタルールの定義例

```
1 table ip filter_sample {
2     chain deny8080 {
3         type filter hook input priority filter;
4         tcp dport 8080 drop
5     }
6 }
```

ソースコード 5.2: iptables におけるシンプルなフィルタルールの定義例

```
1 *filter
2 -A INPUT -p tcp --dport 80 -j DROP
3 COMMIT
```

ソースコード 5.3: nftables におけるレギュラーチェーンとベースチェーンを利用したルールの定義例

```
1 table ip fw01 {
2     chain remote_access {
3         tcp dport ssh drop
4         tcp dport telnet drop
5         accept
6     }
```

```

7   chain rate_limit {
8       ip protocol icmp icmp type echo-request limit rate over 1/
        second drop
9   }
10  chain filter_rule {
11      type filter hook forward priority filter;
12      meta mark 0x1111 goto remote_access
13      meta mark 0x2222 goto rate_limit
14  }
15 }

```

5.5.2 計測内容

トラフィック生成マシンで TRex が生成したトラフィックを SUT マシンに送信した。この測定では、パケット長を一貫して 126 バイトに設定した。パケット長を 126 バイトに設定した理由はセクション 5.4.1 で説明したものと同じで、SID リスト長が 2 の場合のタグ付き VLAN の UDP パケットの最小長が 126 バイトだからである。

5.5.3 評価

図 5.5 は、ベースチェインのルール毎のスループットを示している。全てのチェインルールがベースチェインのみで構成されるこれらのチェインルールは、nftables のチェインルールの定義の中でも最も性能の出ないルール定義の 1 つである。この測定では、netfilter のフォワードフックポイントにフィルタールールを設定した。netfilter は 1 つのフックポイントに複数のルールを設置できる。実験を通して、このフックポイントで適用される同一のカスケードルールの数を増加させた。すべてのパケット転送メカニズムにおいて、スループットはルール数の増加と共に低下する。ルール数が増加するにつれて、3 つのパケット転送メカニズムすべてのスループットは約 0.4 Mbps に収束することがわかった。End.AN.NF と IPv4 のスループットを比較すると、スループット低下における顕著な特性の違いは見られず、End.AN.NF は IPv4 に対して大きく劣るスループット低下特性を示さない。一貫して、End.AN.NF は End.DT4 と H.Encaps の組み合わせのスループットを上回る。しかし、このスループットの差はルール数が増えるにつれて縮小し、128 ルールではわずか 9% の差まで減少した。これは、ルール数が増加するに従ってパケット転送にけるオーバーヘッドの割合が変化したからだと考えられる。ルール数が少ないうちはパケットのカプセル化とデカプセル化にかかるオーバーヘッドが割合として大きく、ルール数が増えるに従って、netfilter の処理にかかるオーバーヘッドの割合が増える。netfilter の処理にかかるオーバーヘッドは End.AN.NF と End.DT4 と H.Encaps の組み合わせで同じであるため、スループットの差が縮まったと考えられる。結果として、レギュラーチェインのルール数が増加するにつれて、End.AN.NF の End.DT4 と H.Encaps の組み合わせに対する優位性は低下すると言える。

図 5.6 は、ベースチェインのルール数毎のスループットを示している。注目すべき点は、ルール数を増加させてもスループットの低下が認めれず、かつ End.AN.NF が一貫して

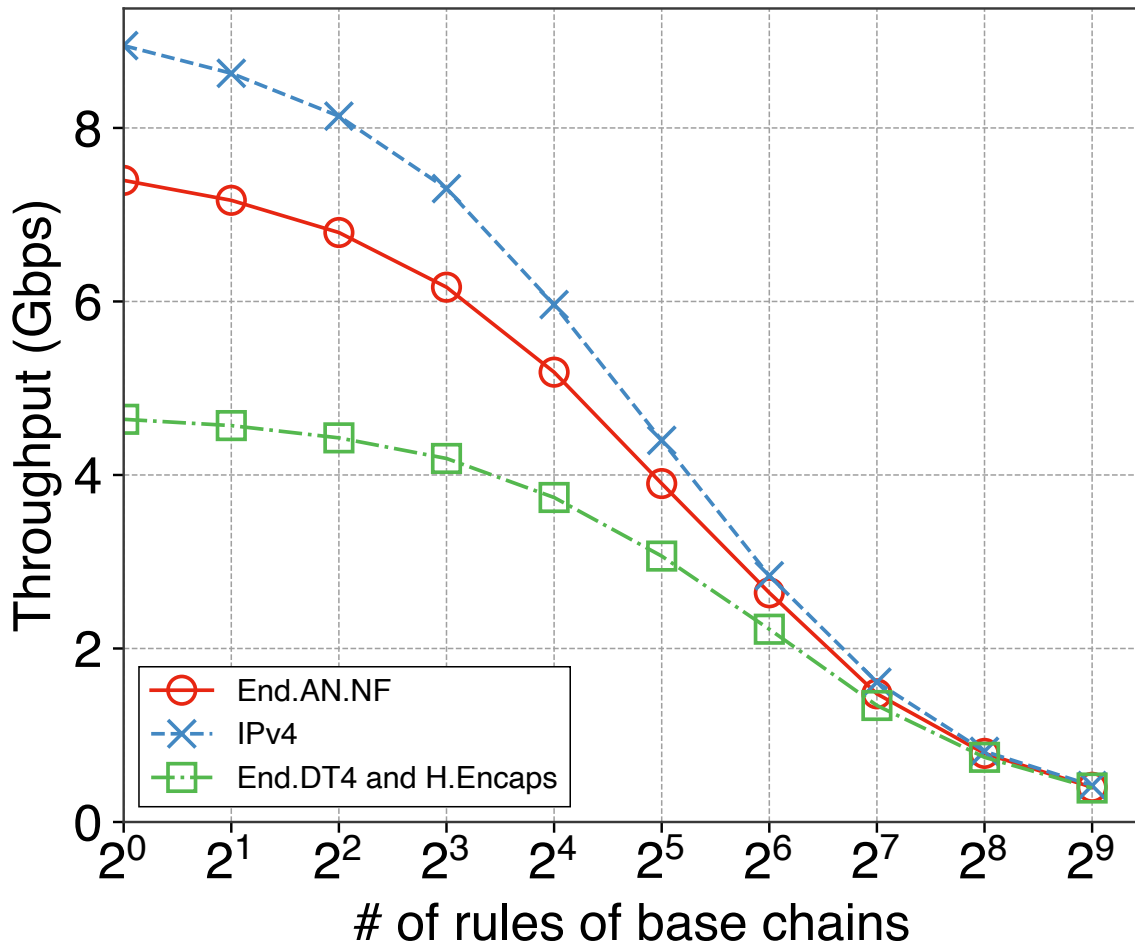


図 5.5: ベースチェーンのルールごとのスループットの

End.DT4 と H.Encaps の組み合わせを上回っていることである。レギュラーチェーンのフィルタールールは、ベースチェーンで測定した際と同じ構成である。通過するパケットをすべてアクセプトするルールが定義されており、一度受け入れるルールが適用されたあとも、事前に決めた回数同じ内容のルールが適用され続ける。しかし、レギュラーチェーンのみから成るこのようなルール構成では、定義されたレギュラーチェーンが他のチェーンから参照されていないため、実際にはルールがパケットへ適用されることはない。その結果、パケットが netfilter のフックポイントを通過する際に実際に適用されるルールの数は変わらない。

どちらの実験でも、End.AN.NF のスループットは一貫して End.DT4 と H.Encaps を組み合わせる手法を上回った。また、End.AN.NF のスループットは IPv4 には一貫して劣るものの、End.AN.NF のスループット低下特性は IPv4 と似ており、End.AN.NF のスループット性能は実用上問題ないことがわかった。

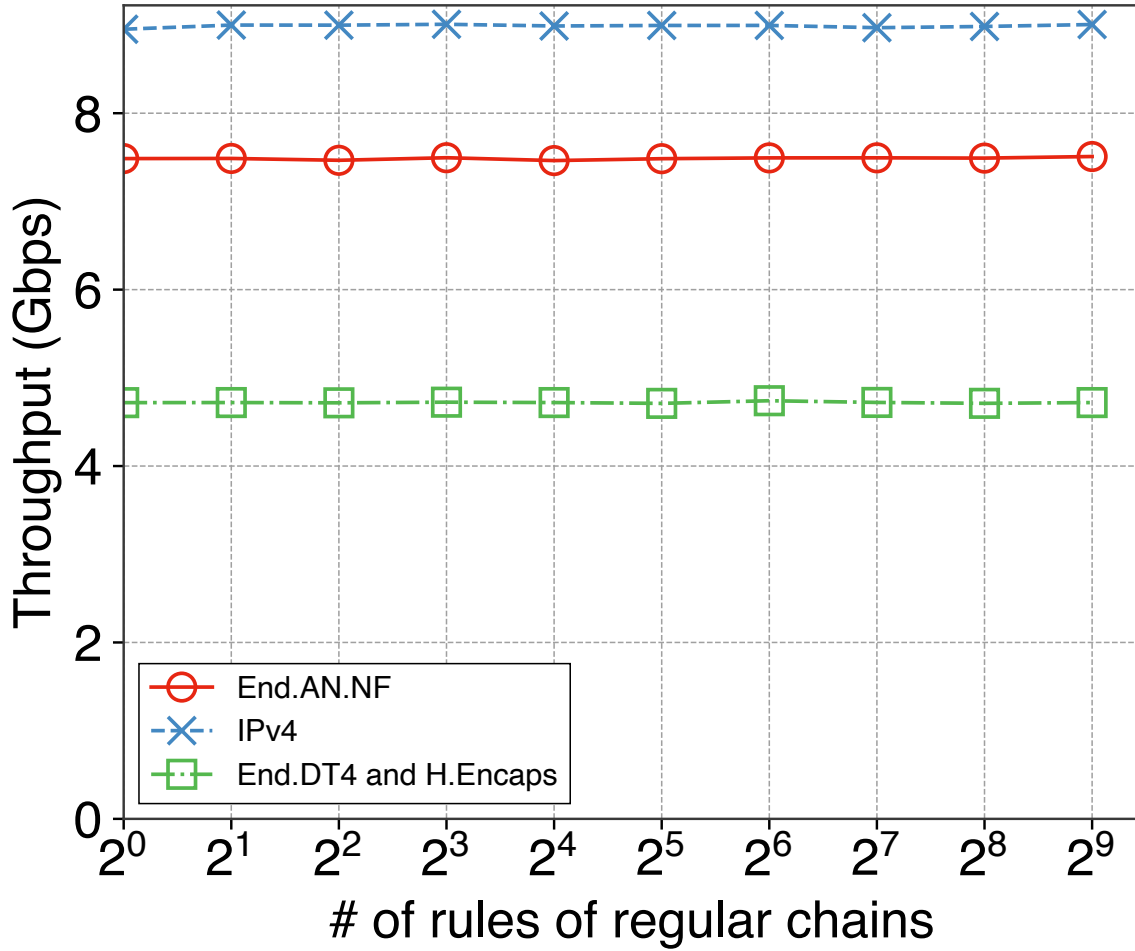


図 5.6: レギュラーチェインのルールごとのスループットの

5.6 レイテンシ

スループットと同様に End.AN.NF, End, IPv4, 及び End.DT4 と H.Encaps の組み合わせについて、レイテンシを測定した。この計測では特に、End.DT4 と H.Encaps の組み合わせと比較して、End.AN.NF のレイテンシがどれだけ改善されたのか評価することを目的としている。この評価では、ベースラインとして IPv4 のレイテンシを用いた。

5.6.1 計測内容

スループットと同様に、計測には TRex を使用し、パケット転送のレイテンシを測定した。TRex はパケットの送受信時間間隔をマイクロ秒単位で測ることが可能である。今回の測定は、パケット長を 142 バイトに設定した。142 バイトの内訳について、先頭 126 バイトはセクション 5.4.1 で説明した通りで、End.AN.NF がの動作要件を満たす最小のパケットとして必要だからである。追加の 16 バイトは、TRex がよるレイテンシを測定する際に利用するメタデータの埋め込みに使用される。実験中、トラフィック生成マシン

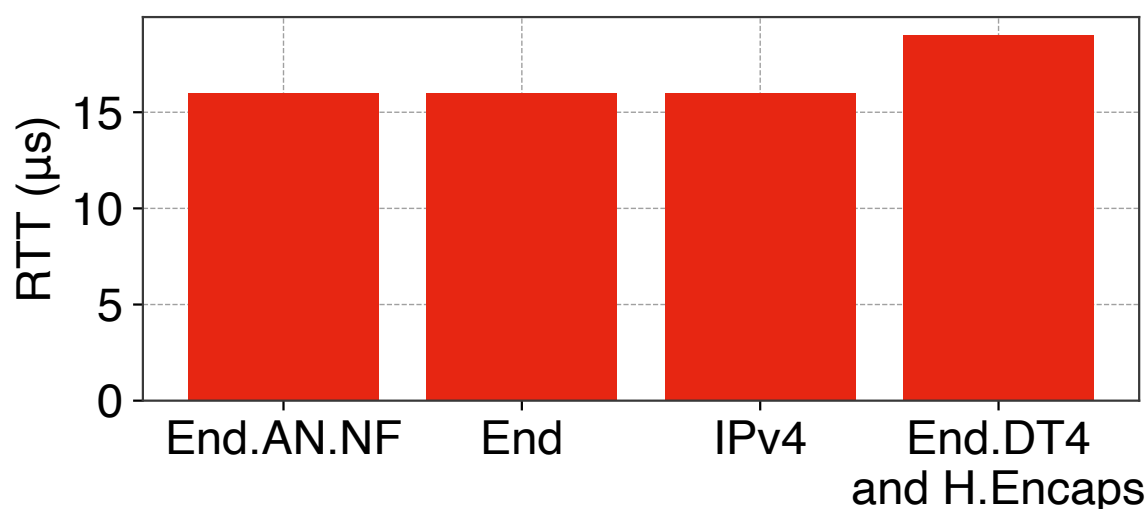


図 5.7: SRv6 End ビヘイビア毎, 及び IPv4 のレイテンシ

は SUT に対して毎秒 10000 個のパケットを 10 秒間送信した。今回のレイテンシ測定では、RSS を無効化するために送信元アドレスと宛先アドレスを変更しなかった。なぜなら、この規模の pps で RSS を使用してパケットを処理する CPU コアを分散させてしまうと、かえってレイテンシが悪化し、余分なジッタが発生することがあるからである。

5.6.2 評価

計測結果を図 5.7 に示す。これらのグラフの各データポイントは、100000 回のレイテンシ測定の平均値を表している。End.AN.NF, End, IPv4 のレイテンシはどれも 16.0 マイクロ秒である。対照的に、End.DT4 と H.Encaps の組み合わせは 19.0 マイクロ秒のレイテンシである。マイクロ秒単位での測定では、End.AN.NF のレイテンシは End と IPv4 のレイテンシと一致し、End.DT4 と H.Encaps の組み合わせのレイテンシよりも約 15.8% 高速であることが分かる。よって、End.AN.NF のレイテンシはマイクロ秒単位であれば IPv4 及び End と同じであり、End.DT4 と H.Encaps の組み合わせのレイテンシよりも約 15.8% 遅延が改善された。

第6章 結論と展望

本論文では、Linux netfilter を統合し、BGP などの既存のルーティングプロトコルとの共存を実現する新しい SRv6 End ビヘイビア、End.AN.NF を提案した。End.AN.NF は、SRv6 の内部のパケットに対して netfilter の 3 つのフックポイント prerouting, forward, postrouting を透過的に適用させることができる。netfilter のフックポイントを透過する事により、netfilter を実装に利用して作成されたアプリケーションは、その実装を変更せずに SR-aware アプリケーションとして機能させることができる。また、End.AN.NF はパケットをマークするために SID の ARG フィールドを活用する。このアプローチにより、netfilter を内部実装に利用した SF アプリケーションは、パケットバッファ上のマークをマッチングさせることによる動的なルール調整が可能となる。論文では End.AN.NF を Linux カーネルに実装し、その性能を評価した。評価の結果、提案手法の実装は、SRv6 インナーパケットに netfilter のルールを適用する方法である End.DT4 と H.Encaps の組み合わせと比較して、27% 高いスループットと 3.0 マイクロ秒低いレイテンシを実現した。さらに、End と End.AN.NF のスループットの差は 6% 未満であり、End.AN.NF のオーバーヘッドは最も基本的な End の動作と比較して許容範囲内であることを示している。

本論文の課題として、End.AN.NF について本論文で議論したのはデータプレーンの範疇に収まっている点が挙げられる。データプレーンとは、ネットワーク通信において、実際のユーザのパケットを処理して転送するメカニズムのことを指す。データプレーンと対になる概念として、コントロールプレーンが存在する。コントロールプレーンとは、ユーザパケットの通る経路やポリシーなどを制御する概念であり、BGP などのルーティングプロトコルがコントロールプレーンの要素の例である。End.AN.NF は SRv6 End ビヘイビアとして設計されているため、その SID を経路情報として広告することができる。ただし、本論文では具体的なコントロールプレーンの設計を提案して議論することはできていない。

例えば、ある netfilter-based アプリケーションを SF 利用するために End.AN.NF と組み合わせるとき、アプリケーションが想定しているパケットバッファのマークを具体的なルールと結びつけて他のノードに伝える手法は現状議論できていない。1 つのアイデアとしては、SDN 的な仕組みを使って SF アプリケーションの持つルールセットから経路情報を生成し、それをルートルフレクタを利用して iBGP で広告する手法が挙げられる。しかし、この手法では SF アプリケーションごとに別な SDN コントローラの実装が必要であり、End.AN.NF の提供する netfilter-based アプリケーションの実装を変更することなく利用可能、という利点を活かすことができない。今後は本研究を発展させてコントロールプレーンについても議論を行い、Linux と SRv6 による SFC の有用性について更に模索して行きたい。

謝辞

本論文を執筆するにあたり、ご指導賜りました慶應義塾大学教授 村井純博士，慶應義塾大学環境情報学部教授 中村修博士，同学部教授 楠本博之博士，同学部教授 高汐一紀博士，同学部教授 Rodney D. Van Meter 博士，同学部教授 植原啓介博士，同学部教授 三次仁博士，同学部教授 中澤仁博士，同学部教授 手塚悟博士，同学部教授 武田圭史博士，同学部准教授 大越匡博士，同大学政策・メディア研究科特任教授 鈴木茂哉博士，同研究科特任助教 工藤紀篤博士，同研究科特任講師 松谷健史博士に感謝いたします。

特に植原啓介博士には rgroot のファカルティとして、日頃から研究面や運用面で指導をしていただきました。また、1 月に参加した私にとって初めての国際会議に同伴していただき、緊張している私をサポートしていただきました。感謝いたします。

私がコンピュータネットワークの分野に進むきっかけを作っていただいた、慶應義塾大学大学院 豊田安信氏，元慶應義塾大学大学院 (現 NTT コミュニケーションズ) 深川祐太氏に感謝いたします。私は 2020 年秋学期に開講された、インターネットの設計と運用 という講義でネットワーク技術の面白さを知ることができました。豊田安信氏，深川祐太氏は TA/SA として私にネットワーク技術の面白さを伝えてくださりました。また、私を rgroot に誘ってくださったのもこのお二人でした。ありがとうございます。

東京大学准教授 中村遼博士に感謝いたします。中村遼博士には、研究面で多大な指導をしていただきました。研究ネタを一緒に考えてくださり、本論文のアイデアも中村遼博士からいただきました。また、中村遼博士に指導をしていただきながら執筆した論文は ICOIN 国際会議に採択していただくことができました。感謝いたします。

慶應義塾大学修士課程 石原匠氏に感謝いたします。石原匠氏は友人として私に接してくれながら、ときには先輩としてその背中を見せてくださりました。コロナ禍に入学した私には大学に友人が少なかったのも、先輩でありながら気軽に話せる存在は大変心の支えになりました。

東京大学大学院 伊藤広記氏，元東京大学大学院 (現 LINE ヤフー株式会社) 金谷光一郎氏に感謝いたします。伊藤広記氏，金谷光一郎氏は、当時の私と同様にネットワーク運用未経験者として WIDE Project の vSIX ワーキンググループに参加し、共に切磋琢磨しあって頂きました。伊藤広記氏，金谷光一郎氏は他大学の先輩でありながら、友人としても私に接して下さいました。ネットワークに入門して日が浅く右も左もわからないとき、わからないなりに共に考え、議論したことはとても良い経験になりました。

父の澤田裕司氏，母の澤田由紀氏に感謝いたします。家では口数の少ない私ですが、部屋に引きこもってパソコン作業を続けることができたのは家族のサポートあってこそでした。感謝いたします。

東京工業大学附属科学技術高等学校 13 期マイコン制御部 OB に感謝いたします。コロ

ナ禍で大学に通えず、また新たな友人を作る機会が殆どなかった当時、同期の皆さんと毎晩オンラインゲームに励んだことは心の支えでした。コロナ禍が明けた今でも、たまに飲みに行ったり、変わらずゲームをしたり、Twitter (X) 上で他愛もないコミュニケーションを取れることは大変嬉しいことです。本論文執筆に関しても、別の大学、別分野の研究でありながら、互いに鼓舞しあうことでモチベーションを高め合い、書き切ることができました。ありがとうございます。

最後に、全員の名前を書くことはできませんが、村井合同研、WIDE プロジェクト関係者全員に感謝いたします。私がネットワーク分野に興味を持ち、続けられたのは皆様の力あってこそでした。深く感謝申し上げます。

参考文献

- [1] The Netfilter’s webmasters. Netfilter hooks. https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks. Accessed: 2024-01-25.
- [2] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.
- [3] Karamjeet Kaur, Veenu Mangat, and Krishan Kumar. A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture. *Computer Science Review*, 38:100298, 2020.
- [4] Irena Trajkovska, Michail-Alexandros Kourtis, Christos Sakkas, Denis Baudinot, João Silva, Piyush Harsh, George Xylouris, Thomas Michael Bohnert, and Harilaos Koumaras. Sdn-based service function chaining mechanism and service prototype implementation in nfv scenario. *Computer Standards & Interfaces*, 54:247–265, 2017. SI: Standardization SDN&NFV.
- [5] Gianluca Davoli, Walter Cerroni, Chiara Contoli, Francesco Foresta, and Franco Callegati. Implementation of service function chaining control plane through openflow. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–4, 2017.
- [6] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. A survey on service function chaining. *Journal of Network and Computer Applications*, 75:138–155, 2016.
- [7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, mar 2008.
- [8] Paul Quinn, Uri Elzur, and Carlos Pignataro. Network Service Header (NSH). RFC 8300, January 2018.
- [9] Adrian Farrel, Stewart Bryant, and John Drake. An MPLS-Based Forwarding Plane for Service Function Chaining. RFC 8595, June 2019.

- [10] Francois Clad, Xiaohu Xu, Clarence Filsfils, Daniel Bernier, Cheng Li, Bruno Decraene, Shaowen Ma, Chaitanya Yadlapalli, Wim Henderickx, and Stefano Salsano. Service Programming with Segment Routing. Internet-Draft draft-ietf-spring-sr-service-programming-08, Internet Engineering Task Force, August 2023. Work in Progress.
- [11] Ryo Nakamura, Yukito Ueno, and Teppei Kamata. An Experiment of SRv6 Service Chaining at Interop Tokyo 2019 ShowNet. Internet-Draft draft-upa-srv6-service-chaining-exp-00, Internet Engineering Task Force, October 2019. Work in Progress.
- [12] Ahmed Abdelsalam, Stefano Salsano, Francois Clad, Pablo Camarillo, and Clarence Filsfils. Sera: Segment routing aware firewall for service function chaining scenarios. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 46–54, 2018.
- [13] The Kubernetes Authors. kubernetes. <https://kubernetes.io/ja/>. Accessed: 2024-01-23.
- [14] Abdullah Bittar, Ziqiang Wang, Amir Aghasharif, Changcheng Huang, Gauravdeep Shami, Marc Lyonnais, and Rodney Wilson. Service function chaining design & implementation using network service mesh in kubernetes. In Dhabaleswar K. Panda and Michael Sullivan, editors, *Supercomputing Frontiers*, pages 121–140, Cham, 2022. Springer International Publishing.
- [15] Yakov Rekhter, Susan Hares, and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.
- [16] John Moy. OSPF Version 2. RFC 2328, April 1998.
- [17] OSI IS-IS Intra-domain Routing Protocol. RFC 1142, February 1990.
- [18] Arun Viswanathan, Eric C. Rosen, and Ross Callon. Multiprotocol Label Switching Architecture. RFC 3031, January 2001.
- [19] Clarence Filsfils, Pablo Camarillo, John Leddy, Daniel Voyer, Satoru Matsushima, and Zhenbin Li. Segment Routing over IPv6 (SRv6) Network Programming. RFC 8986, February 2021.
- [20] Gaurav Dawra, Ketan Talaulikar, Robert Raszuk, Bruno Decraene, Shunwan Zhuang, and Jorge Rabadan. BGP Overlay Services Based on Segment Routing over IPv6 (SRv6). RFC 9252, July 2022.
- [21] iana: Internet Assigned Numbers Authority. IS-IS TLV Codepoints. <https://www.iana.org/assignments/isis-tlv-codepoints/isis-tlv-codepoints.xhtml>. Accessed: 2024-01-27.

- [22] FRRouting Project, a Linux Foundation Collaborative Project. Frrouting. <https://frrouting.org/>. Accessed: 2023-08-23.
- [23] GoBGP Community. gobgp. <https://osrg.github.io/gobgp/>. Accessed: 2024-01-25.
- [24] The Netfilter’s webmasters. netfilter: firewalling, NAT, and packet mangling for linux. <https://www.netfilter.org/>. Accessed: 2024-01-25.
- [25] Pablo Neira Ayuso. conntrack-tools: Connection tracking userspace tools for Linux. <https://conntrack-tools.netfilter.org/>. Accessed: 2024-01-25.
- [26] Pablo Neira Ayuso. conntrack-tools: Connection tracking userspace tools for Linux. <https://conntrack-tools.netfilter.org/>. Accessed: 2024-01-25.
- [27] Clarence Filsfils, Francois Clad, Pablo Camarillo, Ahmed Abdelsalam, Stefano Salsano, Olivier Bonaventure, Jakub Horn, and Jose Liste. SRv6 interoperability report. Internet-Draft draft-filsfils-spring-srv6-interop-02, Internet Engineering Task Force, March 2019. Work in Progress.
- [28] Marco Haerberle, Benjamin Steinert, Michael Weiss, and Michael Menth. A caching sfc proxy based on ebpf. In *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pages 171–179, 2022.
- [29] Andrea Mayer, Stefano Salsano, Pier Luigi Ventre, Ahmed Abdelsalam, Luca Chiaraviglio, and Clarence Filsfils. An efficient linux kernel implementation of service function chaining for legacy vnfs based on ipv6 segment routing. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 333–341, 2019.
- [30] Baosen Zhao, Yifang Qin, Wanghong Yang, Pengfei Fan, and Xu Zhou. Sra: Leveraging af_xdp for programmable network functions with ipv6 segment routing. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pages 455–462, 2022.
- [31] The Netfilter’s webmasters. The netfilter.org “nftables” project. <https://nftables.org/projects/nftables/index.html>. Accessed: 2023-09-18.
- [32] The Netfilter’s webmasters. The netfilter.org “iptables” project. <https://nftables.org/projects/iptables/index.html>. Accessed: 2023-09-18.
- [33] Moby Project. moby. <https://mobyproject.org/>. Accessed: 2024-01-27.
- [34] Moby Project. check-config.sh. <https://raw.githubusercontent.com/moby/moby/master/contrib/check-config.sh>. Accessed: 2024-01-27.
- [35] TRex Team. Trex: Realistic traffic generator. <https://trex-tgn.cisco.com/>. Accessed: 2023-08-25.

- [36] DPDK Project. Dpdk. <https://www.dpdk.org/>. Accessed: 2023-09-15.
- [37] nftables.org. What is nftables?
https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F. Accessed:
2024-01-27.

付録

ソースコード 1: kernel config

```
1 #
2 # Automatically generated file; DO NOT EDIT.
3 # Linux/x86 5.15.106 Kernel Configuration
4 #
5 CONFIG_CC_VERSION_TEXT="gcc (Ubuntu 11.3.0-1ubuntu1
6     ~22.04.1) 11.3.0"
7 CONFIG_CC_IS_GCC=y
8 CONFIG_GCC_VERSION=110300
9 CONFIG_CLANG_VERSION=0
10 CONFIG_AS_IS_GNU=y
11 CONFIG_AS_VERSION=23800
12 CONFIG_LD_IS_BFD=y
13 CONFIG_LD_VERSION=23800
14 CONFIG_LLD_VERSION=0
15 CONFIG_CC_CAN_LINK=y
16 CONFIG_CC_CAN_LINK_STATIC=y
17 CONFIG_CC_HAS_ASM_GOTO=y
18 CONFIG_CC_HAS_ASM_GOTO_OUTPUT=y
19 CONFIG_CC_HAS_ASM_GOTO_TIED_OUTPUT=y
20 CONFIG_CC_HAS_ASM_INLINE=y
21 CONFIG_CC_HAS_NO_PROFILE_FN_ATTR=y
22 CONFIG_PAHOLE_VERSION=0
23 CONFIG_IRQ_WORK=y
24 CONFIG_BUILDTIME_TABLE_SORT=y
25 CONFIG_THREAD_INFO_IN_TASK=y
26 #
27 # General setup
28 #
29 CONFIG_INIT_ENV_ARG_LIMIT=32
30 CONFIG_LOCALVERSION=""
31 CONFIG_BUILD_SALT=""
32 CONFIG_HAVE_KERNEL_GZIP=y
33 CONFIG_HAVE_KERNEL_BZIP2=y
34 CONFIG_HAVE_KERNEL_LZMA=y
35 CONFIG_HAVE_KERNEL_XZ=y
36 CONFIG_HAVE_KERNEL_LZO=y
37 CONFIG_HAVE_KERNEL_LZ4=y
38 CONFIG_HAVE_KERNEL_ZSTD=y
39 CONFIG_KERNEL_ZSTD=y
40 CONFIG_DEFAULT_INIT=""
41 CONFIG_DEFAULT_HOSTNAME="(none)"
42 CONFIG_SWAP=y
43 CONFIG_SYSVIPC=y
44 CONFIG_SYSVIPC_SYSCTL=y
45 CONFIG_POSIX_MQUEUE=y
46 CONFIG_POSIX_MQUEUE_SYSCTL=y
47 CONFIG_WATCH_QUEUE=y
48 CONFIG_CROSS_MEMORY_ATTACH=y
49 CONFIG_USELIB=y
50 CONFIG_AUDIT=y
51 CONFIG_HAVE_ARCH_AUDITSYSCALL=y
52 CONFIG_AUDITSYSCALL=y
53 #
54 # IRQ subsystem
55 #
56 CONFIG_GENERIC_IRQ_PROBE=y
57 CONFIG_GENERIC_IRQ_SHOW=y
58 CONFIG_GENERIC_IRQ_EFFECTIVE_AFF_MASK=y
59 CONFIG_GENERIC_IRQ_PENDING_IRQ=y
60 CONFIG_GENERIC_IRQ_MIGRATION=y
61 CONFIG_HARDIRQS_SW_RESEND=y
62 CONFIG_IRQ_DOMAIN=y
63 CONFIG_IRQ_DOMAIN_HIERARCHY=y
64 CONFIG_GENERIC_MSI_IRQ=y
65 CONFIG_GENERIC_MSI_IRQ_DOMAIN=y
66 CONFIG_IRQ_MSI_IOMMU=y
67 CONFIG_GENERIC_IRQ_MATRIX_ALLOCATOR=y
68 CONFIG_GENERIC_IRQ_RESERVATION_MODE=y
69 CONFIG_IRQ_FORCED_THREADING=y
70 CONFIG_SPARSE_IRQ=y
71 # end of IRQ subsystem
72
73
74
75 CONFIG_CLOCKSOURCE_WATCHDOG=y
76 CONFIG_ARCH_CLOCKSOURCE_INIT=y
77 CONFIG_CLOCKSOURCE_VALIDATE_LAST_CYCLE=y
78 CONFIG_GENERIC_TIME_VSYSCALL=y
79 CONFIG_GENERIC_CLOCKEVENTS=y
80 CONFIG_GENERIC_CLOCKEVENTS_BROADCAST=y
81 CONFIG_GENERIC_CLOCKEVENTS_MIN_ADJUST=y
82 CONFIG_GENERIC_CMOS_UPDATE=y
83 CONFIG_HAVE_POSIX_CPU_TIMERS_TASK_WORK=y
84 CONFIG_POSIX_CPU_TIMERS_TASK_WORK=y
85 #
86 # Timers subsystem
87 #
88 CONFIG_TICK_ONESHOT=y
89 CONFIG_NO_HZ_COMMON=y
90 CONFIG_NO_HZ_IDLE=y
91 CONFIG_NO_HZ=y
92 CONFIG_HIGH_RES_TIMERS=y
93 # end of Timers subsystem
94
95 CONFIG_BPF=y
96 CONFIG_HAVE_EBPF_JIT=y
97 CONFIG_ARCH_WANT_DEFAULT_BPF_JIT=y
98 #
99 # BPF subsystem
100 #
101 CONFIG_BPF_SYSCALL=y
102 CONFIG_BPF_JIT=y
103 CONFIG_BPF_JIT_ALWAYS_ON=y
104 CONFIG_BPF_JIT_DEFAULT_ON=y
105 CONFIG_BPF_UNPRIV_DEFAULT_OFF=y
106 CONFIG_USERMODE_DRIVER=y
107 CONFIG_BPF_LSM=y
108 # end of BPF subsystem
109
110 CONFIG_PREEMPT_VOLUNTARY=y
111 CONFIG_SCHED_CORE=y
112 #
113 # CPU/Task time and stats accounting
114 #
115 CONFIG_TICK_CPU_ACCOUNTING=y
116 CONFIG_BSD_PROCESS_ACCT=y
117 CONFIG_BSD_PROCESS_ACCT_V3=y
118 CONFIG_TASKSTATS=y
119 CONFIG_TASK_DELAY_ACCT=y
120 CONFIG_TASK_XACCT=y
121 CONFIG_TASK_IO_ACCOUNTING=y
122 CONFIG_PSI=y
123 # end of CPU/Task time and stats accounting
124
125 CONFIG_CPU_ISOLATION=y
126 #
127 # RCU Subsystem
128 #
129 CONFIG_TREE_RCU=y
130 CONFIG_SRCU=y
131 CONFIG_TREE_SRCU=y
132 CONFIG_TASKS_RCU_GENERIC=y
133 CONFIG_TASKS_RUDE_RCU=y
134 CONFIG_TASKS_TRACE_RCU=y
135 CONFIG_RCU_STALL_COMMON=y
136 CONFIG_RCU_NEED_SEGCBLIST=y
137 # end of RCU Subsystem
138
139 CONFIG_BUILD_BIN2C=y
140 CONFIG_IKCONFIG=m
141 CONFIG_LOG_BUF_SHIFT=18
142 CONFIG_LOG_CPU_MAX_BUF_SHIFT=12
143 CONFIG_PRINTK_SAFE_LOG_BUF_SHIFT=13
144 CONFIG_HAVE_UNSTABLE_SCHED_CLOCK=y
```

```

149 #
150 # Scheduler features
151 #
152 CONFIG_UCLAMP_TASK=y
153 CONFIG_UCLAMP_BUCKETS_COUNT=5
154 # end of Scheduler features
155
156 CONFIG_ARCH_SUPPORTS_NUMA_BALANCING=y
157 CONFIG_ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH=y
158 CONFIG_CC_HAS_INT128=y
159 CONFIG_ARCH_SUPPORTS_INT128=y
160 CONFIG_NUMA_BALANCING=y
161 CONFIG_NUMA_BALANCING_DEFAULT_ENABLED=y
162 CONFIG_CGROUPS=y
163 CONFIG_PAGE_COUNTER=y
164 CONFIG_MEMCG=y
165 CONFIG_MEMCG_SWAP=y
166 CONFIG_MEMCG_KMEM=y
167 CONFIG_BLK_CGROUP=y
168 CONFIG_CGROUP_WRITEBACK=y
169 CONFIG_CGROUP_SCHED=y
170 CONFIG_FAIR_GROUP_SCHED=y
171 CONFIG_CFS_BANDWIDTH=y
172 CONFIG_UCLAMP_TASK_GROUP=y
173 CONFIG_CGROUP_PIDS=y
174 CONFIG_CGROUP_RDMA=y
175 CONFIG_CGROUP_FREEZER=y
176 CONFIG_CGROUP_HUGETLB=y
177 CONFIG_CPUSETS=y
178 CONFIG_PROC_PID_CPUSET=y
179 CONFIG_CGROUP_DEVICE=y
180 CONFIG_CGROUP_CPUACCT=y
181 CONFIG_CGROUP_PERF=y
182 CONFIG_CGROUP_BPF=y
183 CONFIG_CGROUP_MISC=y
184 CONFIG_SOCK_CGROUP_DATA=y
185 CONFIG_NAMESPACES=y
186 CONFIG_UTS_NS=y
187 CONFIG_TIME_NS=y
188 CONFIG_IPC_NS=y
189 CONFIG_USER_NS=y
190 CONFIG_PID_NS=y
191 CONFIG_NET_NS=y
192 CONFIG_CHECKPOINT_RESTORE=y
193 CONFIG_SCHED_AUTOGROUP=y
194 CONFIG_RELAY=y
195 CONFIG_BLK_DEV_INITRD=y
196 CONFIG_INITRAMFS_SOURCE=""
197 CONFIG_RD_GZIP=y
198 CONFIG_RD_BZIP2=y
199 CONFIG_RD_LZMA=y
200 CONFIG_RD_XZ=y
201 CONFIG_RD_LZO=y
202 CONFIG_RD_LZ4=y
203 CONFIG_RD_ZSTD=y
204 CONFIG_BOOT_CONFIG=y
205 CONFIG_CC_OPTIMIZE_FOR_PERFORMANCE=y
206 CONFIG_LD_ORPHAN_WARN=y
207 CONFIG_SYSCTL=y
208 CONFIG_HAVE_UID16=y
209 CONFIG_SYSCTL_EXCEPTION_TRACE=y
210 CONFIG_HAVE_PCSPKR_PLATFORM=y
211 CONFIG_EXPRT=y
212 CONFIG_UID16=y
213 CONFIG_MULTIUSER=y
214 CONFIG_SGETMASK_SYSCALL=y
215 CONFIG_SYSFS_SYSCALL=y
216 CONFIG_FHANDLE=y
217 CONFIG_POSIX_TIMERS=y
218 CONFIG_PRINTK=y
219 CONFIG_BUG=y
220 CONFIG_ELF_CORE=y
221 CONFIG_PCSPKR_PLATFORM=y
222 CONFIG_BASE_FULL=y
223 CONFIG_FUTEX=y
224 CONFIG_FUTEX_PI=y
225 CONFIG_EPOLL=y
226 CONFIG_SIGNALFD=y
227 CONFIG_TIMERFD=y
228 CONFIG_EVENTFD=y
229 CONFIG_SHMEM=y
230 CONFIG_AIO=y
231 CONFIG_IO_URING=y
232 CONFIG_ADVISE_SYSCALLS=y
233 CONFIG_HAVE_ARCH_USERFAULTFD_WP=y
234 CONFIG_HAVE_ARCH_USERFAULTFD_MINOR=y
235 CONFIG_MEMBARRIER=y
236 CONFIG_KALLSYMS=y
237 CONFIG_KALLSYMS_ALL=y
238 CONFIG_KALLSYMS_ABSOLUTE_PERCPU=y
239 CONFIG_KALLSYMS_BASE_RELATIVE=y
240 CONFIG_USERFAULTFD=y
241 CONFIG_ARCH_HAS_MEMBARRIER_SYNC_CORE=y
242 CONFIG_KCMP=y
243 CONFIG_RSEQ=y
244 CONFIG_HAVE_PERF_EVENTS=y
245 CONFIG_PC104=y
246
247 #
248 # Kernel Performance Events And Counters
249 #
250 CONFIG_PERF_EVENTS=y
251 # end of Kernel Performance Events And Counters
252
253 CONFIG_VM_EVENT_COUNTERS=y
254 CONFIG_SLUB_DEBUG=y
255 CONFIG_SLUB=y
256 CONFIG_SLAB_MERGE_DEFAULT=y
257 CONFIG_SLAB_FREELIST_RANDOM=y
258 CONFIG_SLAB_FREELIST_HARDENED=y
259 CONFIG_SHUFFLE_PAGE_ALLOCATOR=y
260 CONFIG_SLUB_CPU_PARTIAL=y
261 CONFIG_SYSTEM_DATA_VERIFICATION=y
262 CONFIG_PROFILING=y
263 CONFIG_TRACEPOINTS=y
264 # end of General setup
265
266 CONFIG_64BIT=y
267 CONFIG_X86_64=y
268 CONFIG_X86=y
269 CONFIG_INSTRUCTION_DECODER=y
270 CONFIG_OUTPUT_FORMAT="elf64-x86-64"
271 CONFIG_LOCKDEP_SUPPORT=y
272 CONFIG_STACKTRACE_SUPPORT=y
273 CONFIG_MMU=y
274 CONFIG_ARCH_MMAP_RND_BITS_MIN=28
275 CONFIG_ARCH_MMAP_RND_BITS_MAX=32
276 CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MIN=8
277 CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MAX=16
278 CONFIG_GENERIC_ISA_DMA=y
279 CONFIG_GENERIC_BUG=y
280 CONFIG_GENERIC_BUG_RELATIVE_POINTERS=y
281 CONFIG_ARCH_MAY_HAVE_PC_FDC=y
282 CONFIG_GENERIC_CALIBRATE_DELAY=y
283 CONFIG_ARCH_HAS_CPU_RELAX=y
284 CONFIG_ARCH_HAS_FILTER_PGPROT=y
285 CONFIG_HAVE_SETUP_PER_CPU_AREA=y
286 CONFIG_NEED_PER_CPU_EMBED_FIRST_CHUNK=y
287 CONFIG_NEED_PER_CPU_PAGE_FIRST_CHUNK=y
288 CONFIG_ARCH_HIBERNATION_POSSIBLE=y
289 CONFIG_ARCH_NR_GPIO=1024
290 CONFIG_ARCH_SUSPEND_POSSIBLE=y
291 CONFIG_ARCH_WANT_GENERAL_HUGETLB=y
292 CONFIG_AUDIT_ARCH=y
293 CONFIG_HAVE_INTEL_TXT=y
294 CONFIG_X86_64_SMP=y
295 CONFIG_ARCH_SUPPORTS_UPROBES=y
296 CONFIG_FIX_EARLYCON_MEM=y
297 CONFIG_DYNAMIC_PHYSICAL_MASK=y
298 CONFIG_PGTABLE_LEVELS=5
299 CONFIG_CC_HAS_SANE_STACKPROTECTOR=y
300
301 #
302 # Processor type and features
303 #
304 CONFIG_SMP=y
305 CONFIG_X86_FEATURE_NAMES=y
306 CONFIG_X86_X2APIC=y
307 CONFIG_X86_MPPARSE=y
308 CONFIG_X86_CPU_RESCTRL=y
309 CONFIG_X86_EXTENDED_PLATFORM=y
310 CONFIG_X86_NUMACHIP=y
311 CONFIG_X86_UV=y
312 CONFIG_X86_INTEL_LPSS=y
313 CONFIG_X86_AMD_PLATFORM_DEVICE=y
314 CONFIG_IOSEF_MBI=y
315 CONFIG_IOSEF_MBI_DEBUG=y
316 CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
317 CONFIG_SCHED_OMIT_FRAME_POINTER=y
318 CONFIG_HYPERVISOR_GUEST=y
319 CONFIG_PARAVIRT=y
320 CONFIG_PARAVIRT_XXL=y
321 CONFIG_PARAVIRT_SPINLOCKS=y
322 CONFIG_X86_HV_CALLBACK_VECTOR=y
323 CONFIG_XEN=y
324 CONFIG_XEN_PV=y
325 CONFIG_XEN_512GB=y
326 CONFIG_XEN_PV_SMP=y
327 CONFIG_XEN_PV_DOMO=y
328 CONFIG_XEN_PVHVM=y
329 CONFIG_XEN_PVHVM_SMP=y
330 CONFIG_XEN_PVHVM_GUEST=y
331 CONFIG_XEN_SAVE_RESTORE=y
332 CONFIG_XEN_PVH=y
333 CONFIG_XEN_DOMO=y
334 CONFIG_KVM_GUEST=y
335 CONFIG_ARCH_CPUIDLE_HALTPOLL=y
336 CONFIG_PVH=y
337 CONFIG_PARAVIRT_CLOCK=y
338 CONFIG_JAILHOUSE_GUEST=y

```

```

339 CONFIG_ACRN_GUEST=y
340 CONFIG_GENERIC_CPU=y
341 CONFIG_X86_INTERNODE_CACHE_SHIFT=6
342 CONFIG_X86_L1_CACHE_SHIFT=6
343 CONFIG_X86_TSC=y
344 CONFIG_X86_CMPXCHG64=y
345 CONFIG_X86_CMOV=y
346 CONFIG_X86_MINIMUM_CPU_FAMILY=64
347 CONFIG_X86_DEBUGCTLMSR=y
348 CONFIG_IA32_FEAT_CTL=y
349 CONFIG_X86_VMX_FEATURE_NAMES=y
350 CONFIG_PROCESSOR_SELECT=y
351 CONFIG_CPU_SUP_INTEL=y
352 CONFIG_CPU_SUP_AMD=y
353 CONFIG_CPU_SUP_HYGON=y
354 CONFIG_CPU_SUP_CENTAUR=y
355 CONFIG_CPU_SUP_ZHAOXIN=y
356 CONFIG_HPET_TIMER=y
357 CONFIG_HPET_EMULATE_RTC=y
358 CONFIG_DMI=y
359 CONFIG_GART_IOMMU=y
360 CONFIG_MAXSMP=y
361 CONFIG_NR_CPUS_RANGE_BEGIN=8192
362 CONFIG_NR_CPUS_RANGE_END=8192
363 CONFIG_NR_CPUS_DEFAULT=8192
364 CONFIG_NR_CPUS=8192
365 CONFIG_SCHED_SMT=y
366 CONFIG_SCHED_MC=y
367 CONFIG_SCHED_MC_PRIO=y
368 CONFIG_X86_LOCAL_APIC=y
369 CONFIG_X86_IO_APIC=y
370 CONFIG_X86_REROUTE_FOR_BROKEN_BOOT_IRQS=y
371 CONFIG_X86_MCE=y
372 CONFIG_X86_MCELOG_LEGACY=y
373 CONFIG_X86_MCE_INTEL=y
374 CONFIG_X86_MCE_AMD=y
375 CONFIG_X86_MCE_THRESHOLD=y
376
377 #
378 # Performance monitoring
379 #
380 CONFIG_PERF_EVENTS_INTEL_UNCORE=y
381 CONFIG_PERF_EVENTS_INTEL_RAPL=m
382 CONFIG_PERF_EVENTS_INTEL_CSTATE=m
383 # end of Performance monitoring
384
385 CONFIG_X86_16BIT=y
386 CONFIG_X86_ESPFIX64=y
387 CONFIG_X86_VSYSCALL_EMULATION=y
388 CONFIG_X86_IOPL_IOPERM=y
389 CONFIG_MICROCODE=y
390 CONFIG_MICROCODE_INTEL=y
391 CONFIG_MICROCODE_AMD=y
392 CONFIG_X86_MSR=m
393 CONFIG_X86_SLEVEL=y
394 CONFIG_X86_DIRECT_GBPPAGES=y
395 CONFIG_AMD_MEM_ENCRYPT=y
396 CONFIG_NUMA=y
397 CONFIG_AMD_NUMA=y
398 CONFIG_X86_64_ACPI_NUMA=y
399 CONFIG_NODES_SHIFT=10
400 CONFIG_ARCH_SPARSEMEM_ENABLE=y
401 CONFIG_ARCH_SPARSEMEM_DEFAULT=y
402 CONFIG_ARCH_SELECT_MEMORY_MODEL=y
403 CONFIG_ARCH_MEMORY_PROBE=y
404 CONFIG_ARCH_PROC_KCORE_TEXT=y
405 CONFIG_ILLEGAL_POINTER_VALUE=0xdead000000000000
406 CONFIG_X86_PMEM_LEGACY_DEVICE=y
407 CONFIG_X86_PMEM_LEGACY=y
408 CONFIG_X86_CHECK_BIOS_CORRUPTION=y
409 CONFIG_X86_BOOTPARAM_MEMORY_CORRUPTION_CHECK=y
410 CONFIG_MTRR=y
411 CONFIG_MTRR_SANITIZER=y
412 CONFIG_MTRR_SANITIZER_ENABLE_DEFAULT=1
413 CONFIG_MTRR_SANITIZER_SPARE_REG_NR_DEFAULT=1
414 CONFIG_X86_PAT=y
415 CONFIG_ARCH_USES_PG_UNCACHED=y
416 CONFIG_ARCH_RANDOM=y
417 CONFIG_X86_SMAP=y
418 CONFIG_X86_UMIP=y
419 CONFIG_X86_INTEL_MEMORY_PROTECTION_KEYS=y
420 CONFIG_X86_INTEL_TSX_MODE_OFF=y
421 CONFIG_X86_SGX=y
422 CONFIG_EFI=y
423 CONFIG_EFI_STUB=y
424 CONFIG_EFI_MIXED=y
425 CONFIG_HZ_250=y
426 CONFIG_HZ=250
427 CONFIG_SCHED_HRTICK=y
428 CONFIG_KEXEC=y
429 CONFIG_KEXEC_FILE=y
430 CONFIG_ARCH_HAS_KEXEC_PURGATORY=y
431 CONFIG_KEXEC_SIG=y
432 CONFIG_KEXEC_BZIMAGE_VERIFY_SIG=y
433 CONFIG_CRASH_DUMP=y
434
435 CONFIG_KEXEC_JUMP=y
436 CONFIG_PHYSICAL_START=0x1000000
437 CONFIG_RELOCATABLE=y
438 CONFIG_RANDOMIZE_BASE=y
439 CONFIG_X86_NEED_RELOCS=y
440 CONFIG_PHYSICAL_ALIGN=0x200000
441 CONFIG_DYNAMIC_MEMORY_LAYOUT=y
442 CONFIG_RANDOMIZE_MEMORY=y
443 CONFIG_RANDOMIZE_MEMORY_PHYSICAL_PADDING=0xa
444 CONFIG_HOTPLUG_CPU=y
445 CONFIG_LEGACY_VSYSCALL_XONLY=y
446 CONFIG_MODIFY_LDT_SYSCALL=y
447 CONFIG_HAVE_LIVEPATCH=y
448 CONFIG_LIVEPATCH=y
449 # end of Processor type and features
450
451 CONFIG_CC_HAS_SLS=y
452 CONFIG_CC_HAS_RETURN_THUNK=y
453 CONFIG_SPECULATION_MITIGATIONS=y
454 CONFIG_PAGE_TABLE_ISOLATION=y
455 CONFIG_RETPOLINE=y
456 CONFIG_RETHUNK=y
457 CONFIG_CPU_UNRET_ENTRY=y
458 CONFIG_CPU_IBPB_ENTRY=y
459 CONFIG_CPU_IBRS_ENTRY=y
460 CONFIG_SLS=y
461 CONFIG_ARCH_HAS_ADD_PAGES=y
462 CONFIG_ARCH_MHP_MEMMAP_ON_MEMORY_ENABLE=y
463 CONFIG_USE_PERCPU_NUMA_NODE_ID=y
464
465 #
466 # Power management and ACPI options
467 #
468 CONFIG_ARCH_HIBERNATION_HEADER=y
469 CONFIG_SUSPEND=y
470 CONFIG_SUSPEND_FREEZER=y
471 CONFIG_HIBERNATE_CALLBACKS=y
472 CONFIG_HIBERNATION=y
473 CONFIG_HIBERNATION_SNAPSHOT_DEV=y
474 CONFIG_PM_STD_PARTITION=""
475 CONFIG_PM_SLEEP=y
476 CONFIG_PM_SLEEP_SMP=y
477 CONFIG_PM_WAKELOCKS=y
478 CONFIG_PM_WAKELOCKS_LIMIT=100
479 CONFIG_PM_WAKELOCKS_GC=y
480 CONFIG_PM=y
481 CONFIG_PM_DEBUG=y
482 CONFIG_PM_ADVANCED_DEBUG=y
483 CONFIG_PM_SLEEP_DEBUG=y
484 CONFIG_PM_TRACE=y
485 CONFIG_PM_TRACE_RTC=y
486 CONFIG_PM_CLK=y
487 CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
488 CONFIG_ENERGY_MODEL=y
489 CONFIG_ARCH_SUPPORTS_ACPI=y
490 CONFIG_ACPI=y
491 CONFIG_ACPI_LEGACY_TABLES_LOOKUP=y
492 CONFIG_ARCH_MIGHT_HAVE_ACPI_PDC=y
493 CONFIG_ACPI_SYSTEM_POWER_STATES_SUPPORT=y
494 CONFIG_ACPI_DEBUGGER=y
495 CONFIG_ACPI_DEBUGGER_USER=y
496 CONFIG_ACPI_SPCR_TABLE=y
497 CONFIG_ACPI_FPD=y
498 CONFIG_ACPI_LPIT=y
499 CONFIG_ACPI_SLEEP=y
500 CONFIG_ACPI_REV_OVERRIDE_POSSIBLE=y
501 CONFIG_ACPI_AC=y
502 CONFIG_ACPI_BATTERY=y
503 CONFIG_ACPI_BUTTON=y
504 CONFIG_ACPI_FAN=y
505 CONFIG_ACPI_DOCK=y
506 CONFIG_ACPI_CPU_FREQ_PSS=y
507 CONFIG_ACPI_PROCESSOR_CSTATE=y
508 CONFIG_ACPI_PROCESSOR_IDLE=y
509 CONFIG_ACPI_CPPC_LIB=y
510 CONFIG_ACPI_PROCESSOR=y
511 CONFIG_ACPI_IPMI=m
512 CONFIG_ACPI_HOTPLUG_CPU=y
513 CONFIG_ACPI_THERMAL=y
514 CONFIG_ACPI_CUSTOM_DSDT_FILE=""
515 CONFIG_ARCH_HAS_ACPI_TABLE_UPGRADE=y
516 CONFIG_ACPI_TABLE_UPGRADE=y
517 CONFIG_ACPI_DEBUG=y
518 CONFIG_ACPI_PCI_SLOT=y
519 CONFIG_ACPI_CONTAINER=y
520 CONFIG_ACPI_HOTPLUG_MEMORY=y
521 CONFIG_ACPI_HOTPLUG_IOAPIC=y
522 CONFIG_ACPI_HED=y
523 CONFIG_ACPI_BGRT=y
524 CONFIG_ACPI_NFIT=m
525 CONFIG_ACPI_NUMA=y
526 CONFIG_ACPI_HMAT=y
527 CONFIG_HAVE_ACPI_APEI=y
528 CONFIG_HAVE_ACPI_APEI_NMI=y
529 CONFIG_ACPI_APEI=y

```



```

529 CONFIG_ACPI_APEI_GHES=y
530 CONFIG_ACPI_APEI_PCIEAER=y
531 CONFIG_ACPI_APEI_MEMORY_FAILURE=y
532 CONFIG_ACPI_DPTF=y
533 CONFIG_ACPI_ADXL=y
534 CONFIG_PMIC_OPREGION=y
535 CONFIG_BYTCRC_PMIC_OPREGION=y
536 CONFIG_CHTCRC_PMIC_OPREGION=y
537 CONFIG_CHT_WC_PMIC_OPREGION=y
538 CONFIG_ACPI_VIOT=y
539 CONFIG_X86_PM_TIMER=y
540 CONFIG_ACPI_PRMT=y
541
542 #
543 # CPU Frequency scaling
544 #
545 CONFIG_CPU_FREQ=y
546 CONFIG_CPU_FREQ_GOV_ATTR_SET=y
547 CONFIG_CPU_FREQ_GOV_COMMON=y
548 CONFIG_CPU_FREQ_STAT=y
549 CONFIG_CPU_FREQ_DEFAULT_GOV_SCHEDUTIL=y
550 CONFIG_CPU_FREQ_GOV_PERFORMANCE=y
551 CONFIG_CPU_FREQ_GOV_POWERSAVE=y
552 CONFIG_CPU_FREQ_GOV_USERSPACE=y
553 CONFIG_CPU_FREQ_GOV_ONDEMAND=y
554 CONFIG_CPU_FREQ_GOV_CONSERVATIVE=y
555 CONFIG_CPU_FREQ_GOV_SCHEDUTIL=y
556
557 #
558 # CPU frequency scaling drivers
559 #
560 CONFIG_X86_INTEL_PSTATE=y
561 CONFIG_X86_PCC_CPUFREQ=y
562 CONFIG_X86_ACPI_CPUFREQ=y
563 CONFIG_X86_ACPI_CPUFREQ_CPB=y
564 CONFIG_X86_POWERNOW_K8=y
565 CONFIG_X86_SPEEDSTEP_CENTRINO=y
566
567 #
568 # shared options
569 #
570 # end of CPU Frequency scaling
571
572 #
573 # CPU Idle
574 #
575 CONFIG_CPU_IDLE=y
576 CONFIG_CPU_IDLE_GOV_LADDER=y
577 CONFIG_CPU_IDLE_GOV_MENU=y
578 CONFIG_CPU_IDLE_GOV_TEO=y
579 CONFIG_CPU_IDLE_GOV_HALTPOLL=y
580 # end of CPU Idle
581
582 CONFIG_INTEL_IDLE=y
583 # end of Power management and ACPI options
584
585 #
586 # Bus options (PCI etc.)
587 #
588 CONFIG_PCI_DIRECT=y
589 CONFIG_PCI_MMCONFIG=y
590 CONFIG_PCI_XEN=y
591 CONFIG_MMCONF_FAM10H=y
592 CONFIG_ISA_BUS=y
593 CONFIG_ISA_DMA_API=y
594 CONFIG_AMD_NB=y
595 # end of Bus options (PCI etc.)
596
597 #
598 # Binary Emulations
599 #
600 CONFIG_IA32_EMULATION=y
601 CONFIG_X86_X32=y
602 CONFIG_COMPAT_32=y
603 CONFIG_COMPAT=y
604 CONFIG_COMPAT_FOR_U64_ALIGNMENT=y
605 CONFIG_SYSVIPC_COMPAT=y
606 # end of Binary Emulations
607
608 CONFIG_HAVE_KVM=y
609 CONFIG_VIRTUALIZATION=y
610 CONFIG_AS_AVX512=y
611 CONFIG_AS_SHA1_NI=y
612 CONFIG_AS_SHA256_NI=y
613 CONFIG_AS_TPAUSE=y
614
615 #
616 # General architecture-dependent options
617 #
618 CONFIG_CRASH_CORE=y
619 CONFIG_KEXEC_CORE=y
620 CONFIG_HOTPLUG_SMT=y
621 CONFIG_GENERIC_ENTRY=y
622 CONFIG_KPROBES=y
623 CONFIG_JUMP_LABEL=y
624
625 CONFIG_OPTPROBES=y
626 CONFIG_KPROBES_ON_FTRACE=y
627 CONFIG_UPROBES=y
628 CONFIG_HAVE_EFFICIENT_UNALIGNED_ACCESS=y
629 CONFIG_ARCH_USE_BUILTIN_BSWAP=y
630 CONFIG_KRETPROBES=y
631 CONFIG_HAVE_IOREMAP_PROT=y
632 CONFIG_HAVE_KPROBES=y
633 CONFIG_HAVE_KRETPROBES=y
634 CONFIG_HAVE_OPTPROBES=y
635 CONFIG_HAVE_KPROBES_ON_FTRACE=y
636 CONFIG_HAVE_FUNCTION_ERROR_INJECTION=y
637 CONFIG_HAVE_NMI=y
638 CONFIG_TRACE_IRQFLAGS_SUPPORT=y
639 CONFIG_TRACE_IRQFLAGS_NMI_SUPPORT=y
640 CONFIG_HAVE_ARCH_TRACEHOOK=y
641 CONFIG_HAVE_DMA_CONTIGUOUS=y
642 CONFIG_GENERIC_SMP_IDLE_THREAD=y
643 CONFIG_ARCH_HAS_FORTIFY_SOURCE=y
644 CONFIG_ARCH_HAS_SET_MEMORY=y
645 CONFIG_ARCH_HAS_SET_DIRECT_MAP=y
646 CONFIG_HAVE_ARCH_THREAD_STRUCT_WHITELIST=y
647 CONFIG_ARCH_WANTS_DYNAMIC_TASK_STRUCT=y
648 CONFIG_ARCH_WANTS_NO_INSTR=y
649 CONFIG_HAVE_ASM_MODVERSIONS=y
650 CONFIG_HAVE_REGS_AND_STACK_ACCESS_API=y
651 CONFIG_HAVE_RSEQ=y
652 CONFIG_HAVE_FUNCTION_ARG_ACCESS_API=y
653 CONFIG_HAVE_HW_BREAKPOINT=y
654 CONFIG_HAVE_MIXED_BREAKPOINTS_REGS=y
655 CONFIG_HAVE_USER_RETURN_NOTIFIER=y
656 CONFIG_HAVE_PERF_EVENTS_NMI=y
657 CONFIG_HAVE_HARDLOCKUP_DETECTOR_PERF=y
658 CONFIG_HAVE_PERF_REGS=y
659 CONFIG_HAVE_PERF_USER_STACK_DUMP=y
660 CONFIG_HAVE_ARCH_JUMP_LABEL=y
661 CONFIG_HAVE_ARCH_JUMP_LABEL_RELATIVE=y
662 CONFIG_MMU_GATHER_TABLE_FREE=y
663 CONFIG_MMU_GATHER_RCU_TABLE_FREE=y
664 CONFIG_ARCH_HAVE_NMI_SAFE_CMPXCHG=y
665 CONFIG_HAVE_ALIGNED_STRUCT_PAGE=y
666 CONFIG_HAVE_CMPXCHG_LOCAL=y
667 CONFIG_HAVE_CMPXCHG_DOUBLE=y
668 CONFIG_ARCH_WANT_COMPAT_IPC_PARSE_VERSION=y
669 CONFIG_ARCH_WANT_OLD_COMPAT_IPC=y
670 CONFIG_HAVE_ARCH_SECCOMP=y
671 CONFIG_HAVE_ARCH_SECCOMP_FILTER=y
672 CONFIG_SECCOMP=y
673 CONFIG_SECCOMP_FILTER=y
674 CONFIG_HAVE_ARCH_STACKLEAK=y
675 CONFIG_HAVE_STACKPROTECTOR=y
676 CONFIG_STACKPROTECTOR=y
677 CONFIG_STACKPROTECTOR_STRONG=y
678 CONFIG_ARCH_SUPPORTS_LTO_CLANG=y
679 CONFIG_ARCH_SUPPORTS_LTO_CLANG_THIN=y
680 CONFIG_LTO_NONE=y
681 CONFIG_HAVE_ARCH_WITHIN_STACK_FRAMES=y
682 CONFIG_HAVE_CONTEXT_TRACKING=y
683 CONFIG_HAVE_CONTEXT_TRACKING_OFFSTACK=y
684 CONFIG_HAVE_VIRT_CPU_ACCOUNTING_GEN=y
685 CONFIG_HAVE_IRQ_TIME_ACCOUNTING=y
686 CONFIG_HAVE_MOVE_PUD=y
687 CONFIG_HAVE_MOVE_PMD=y
688 CONFIG_HAVE_ARCH_TRANSPARENT_HUGEPAGE=y
689 CONFIG_HAVE_ARCH_TRANSPARENT_HUGEPAGE_PUD=y
690 CONFIG_HAVE_ARCH_HUGE_VMAP=y
691 CONFIG_ARCH_WANT_HUGE_PMD_SHARE=y
692 CONFIG_HAVE_ARCH_SOFT_DIRTY=y
693 CONFIG_HAVE_MOD_ARCH_SPECIFIC=y
694 CONFIG_MODULES_USE_ELF_RELA=y
695 CONFIG_HAVE_IRQ_EXIT_ON_IRQ_STACK=y
696 CONFIG_HAVE_SOFTIRQ_ON_OWN_STACK=y
697 CONFIG_ARCH_HAS_ELF_RANDOMIZE=y
698 CONFIG_HAVE_ARCH_MMAP_RND_BITS=y
699 CONFIG_HAVE_EXIT_THREAD=y
700 CONFIG_ARCH_MMAP_RND_BITS=28
701 CONFIG_HAVE_ARCH_MMAP_RND_COMPAT_BITS=y
702 CONFIG_ARCH_MMAP_RND_COMPAT_BITS=8
703 CONFIG_HAVE_ARCH_COMPAT_MMAP_BASES=y
704 CONFIG_HAVE_STACK_VALIDATION=y
705 CONFIG_HAVE_RELIABLE_STACKTRACE=y
706 CONFIG_OLD_SIGSUSPEND3=y
707 CONFIG_COMPAT_OLD_SIGACTION=y
708 CONFIG_COMPAT_32BIT_TIME=y
709 CONFIG_HAVE_ARCH_VMAP_STACK=y
710 CONFIG_VMAP_STACK=y
711 CONFIG_HAVE_ARCH_RANDOMIZE_KSTACK_OFFSET=y
712 CONFIG_RANDOMIZE_KSTACK_OFFSET_DEFAULT=y
713 CONFIG_ARCH_HAS_STRICT_KERNEL_RWX=y
714 CONFIG_STRICT_KERNEL_RWX=y
715 CONFIG_ARCH_HAS_STRICT_MODULE_RWX=y
716 CONFIG_STRICT_MODULE_RWX=y
717 CONFIG_HAVE_ARCH_PREL32_RELOCATIONS=y
718 CONFIG_ARCH_USE_MEMREMAP_PROT=y
719 CONFIG_ARCH_HAS_MEM_ENCRYPT=y

```

```

719 CONFIG_ARCH_HAS_CC_PLATFORM=y
720 CONFIG_HAVE_STATIC_CALL=y
721 CONFIG_HAVE_STATIC_CALL_INLINE=y
722 CONFIG_HAVE_PREEMPT_DYNAMIC=y
723 CONFIG_ARCH_WANT_LD_ORPHAN_WARN=y
724 CONFIG_ARCH_SUPPORTS_DEBUG_PAGEALLOC=y
725 CONFIG_ARCH_HAS_ELF_CORE_COMPAT=y
726 CONFIG_ARCH_HAS_PARANOID_L1D_FLUSH=y
727
728 #
729 # GCOV-based kernel profiling
730 #
731 CONFIG_ARCH_HAS_GCOV_PROFILE_ALL=y
732 # end of GCOV-based kernel profiling
733
734 CONFIG_HAVE_GCC_PLUGINS=y
735 # end of General architecture-dependent options
736
737 CONFIG_RT_MUTEXES=y
738 CONFIG_BASE_SMALL=0
739 CONFIG_MODULE_SIG_FORMAT=y
740 CONFIG_MODULES=y
741 CONFIG_MODULE_UNLOAD=y
742 CONFIG_MODVERSIONS=y
743 CONFIG_ASM_MODVERSIONS=y
744 CONFIG_MODULE_SRCVERSION_ALL=y
745 CONFIG_MODULE_SIG=y
746 CONFIG_MODULE_SIG_ALL=y
747 CONFIG_MODULE_SIG_SHA512=y
748 CONFIG_MODULE_SIG_HASH="sha512"
749 CONFIG_MODULE_COMPRESS_NONE=y
750 CONFIG_MODPROBE_PATH="/sbin/modprobe"
751 CONFIG_MODULES_TREE_LOOKUP=y
752 CONFIG_BLOCK=y
753 CONFIG_BLK_RQ_ALLOC_TIME=y
754 CONFIG_BLK_CGROUP_RWSTAT=y
755 CONFIG_BLK_DEV_BSG_COMMON=y
756 CONFIG_BLK_DEV_BSGLIB=y
757 CONFIG_BLK_DEV_INTEGRITY=y
758 CONFIG_BLK_DEV_INTEGRITY_T10=y
759 CONFIG_BLK_DEV_ZONED=y
760 CONFIG_BLK_DEV_THROTTLING=y
761 CONFIG_BLK_WBT=y
762 CONFIG_BLK_WBT_MQ=y
763 CONFIG_BLK_CGROUP_IOCOST=y
764 CONFIG_BLK_CGROUP_IOPRIO=y
765 CONFIG_BLK_DEBUG_FS=y
766 CONFIG_BLK_DEBUG_FS_ZONED=y
767 CONFIG_BLK_SED_OPAL=y
768 CONFIG_BLK_INLINE_ENCRYPTION=y
769 CONFIG_BLK_INLINE_ENCRYPTION_FALLBACK=y
770
771 #
772 # Partition Types
773 #
774 CONFIG_PARTITION_ADVANCED=y
775 CONFIG_AIX_PARTITION=y
776 CONFIG_OSF_PARTITION=y
777 CONFIG_AMIGA_PARTITION=y
778 CONFIG_ATARI_PARTITION=y
779 CONFIG_MAC_PARTITION=y
780 CONFIG_MSDOS_PARTITION=y
781 CONFIG_BSD_DISKLABEL=y
782 CONFIG_MINIX_SUBPARTITION=y
783 CONFIG_SOLARIS_X86_PARTITION=y
784 CONFIG_UNIXWARE_DISKLABEL=y
785 CONFIG_LDM_PARTITION=y
786 CONFIG_SGI_PARTITION=y
787 CONFIG_ULTRIX_PARTITION=y
788 CONFIG_SUN_PARTITION=y
789 CONFIG_KARMA_PARTITION=y
790 CONFIG_EFI_PARTITION=y
791 CONFIG_SYSV68_PARTITION=y
792 CONFIG_CMDLINE_PARTITION=y
793 # end of Partition Types
794
795 CONFIG_BLOCK_COMPAT=y
796 CONFIG_BLK_MQ_PCI=y
797 CONFIG_BLK_MQ_VIRTIO=y
798 CONFIG_BLK_MQ_RDMA=y
799 CONFIG_BLK_PM=y
800 CONFIG_BLOCK_HOLDER_DEPRECATED=y
801
802 #
803 # IO Schedulers
804 #
805 CONFIG_MQ_IOSCHED_DEADLINE=y
806 # end of IO Schedulers
807
808 CONFIG_ASN1=y
809 CONFIG_INLINE_SPIN_UNLOCK_IRQ=y
810 CONFIG_INLINE_READ_UNLOCK=y
811 CONFIG_INLINE_READ_UNLOCK_IRQ=y
812 CONFIG_INLINE_WRITE_UNLOCK=y
813 CONFIG_INLINE_WRITE_UNLOCK_IRQ=y
814
815 CONFIG_ARCH_SUPPORTS_ATOMIC_RMW=y
816 CONFIG_MUTEX_SPIN_ON_OWNER=y
817 CONFIG_RWSEM_SPIN_ON_OWNER=y
818 CONFIG_LOCK_SPIN_ON_OWNER=y
819 CONFIG_ARCH_USE_QUEUED_SPINLOCKS=y
820 CONFIG_QUEUED_SPINLOCKS=y
821 CONFIG_ARCH_USE_QUEUED_RWLOCKS=y
822 CONFIG_QUEUED_RWLOCKS=y
823 CONFIG_ARCH_HAS_NON_OVERLAPPING_ADDRESS_SPACE=y
824 CONFIG_ARCH_HAS_SYNC_CORE_BEFORE_USERMODE=y
825 CONFIG_ARCH_HAS_SYSCALL_WRAPPER=y
826 CONFIG_FREEZER=y
827
828 #
829 # Executable file formats
830 #
831 CONFIG_BINFMT_ELF=y
832 CONFIG_COMPAT_BINFMT_ELF=y
833 CONFIG_ELF_CORE=y
834 CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS=y
835 CONFIG_BINFMT_SCRIPT=y
836 CONFIG_BINFMT_MISC=m
837 CONFIG_COREDUMP=y
838 # end of Executable file formats
839
840 #
841 # Memory Management options
842 #
843 CONFIG_SELECT_MEMORY_MODEL=y
844 CONFIG_SPARSEMEM_MANUAL=y
845 CONFIG_SPARSEMEM=y
846 CONFIG_SPARSEMEM_EXTREME=y
847 CONFIG_SPARSEMEM_VMEMMAP_ENABLE=y
848 CONFIG_SPARSEMEM_VMEMMAP=y
849 CONFIG_HAVE_FAST_GUP=y
850 CONFIG_NUMA_KEEP_MEMINFO=y
851 CONFIG_MEMORY_ISOLATION=y
852 CONFIG_HAVE_BOOTMEM_INFO_NODE=y
853 CONFIG_ARCH_ENABLE_MEMORY_HOTPLUG=y
854 CONFIG_MEMORY_HOTPLUG=y
855 CONFIG_MEMORY_HOTPLUG_SPARSE=y
856 CONFIG_MEMORY_HOTPLUG_DEFAULT_ONLINE=y
857 CONFIG_ARCH_ENABLE_MEMORY_HOTREMOVE=y
858 CONFIG_MEMORY_HOTREMOVE=y
859 CONFIG_MHP_MEMMAP_ON_MEMORY=y
860 CONFIG_SPLIT_PTLOCK_CPUS=4
861 CONFIG_ARCH_ENABLE_SPLIT_PMD_PTLOCK=y
862 CONFIG_MEMORY_BALLOON=y
863 CONFIG_BALLOON_COMPACTION=y
864 CONFIG_COMPACTION=y
865 CONFIG_PAGE_REPORTING=y
866 CONFIG_MIGRATION=y
867 CONFIG_ARCH_ENABLE_HUGEPAGE_MIGRATION=y
868 CONFIG_ARCH_ENABLE_THP_MIGRATION=y
869 CONFIG_CONTIG_ALLOC=y
870 CONFIG_PHYS_ADDR_T_64BIT=y
871 CONFIG_VIRT_TO_BUS=y
872 CONFIG_MMU_NOTIFIER=y
873 CONFIG_KSM=y
874 CONFIG_DEFAULT_MMAP_MIN_ADDR=65536
875 CONFIG_ARCH_SUPPORTS_MEMORY_FAILURE=y
876 CONFIG_MEMORY_FAILURE=y
877 CONFIG_TRANSPARENT_HUGEPAGE=y
878 CONFIG_TRANSPARENT_HUGEPAGE_MADVISE=y
879 CONFIG_ARCH_WANTS_THP_SWAP=y
880 CONFIG_THP_SWAP=y
881 CONFIG_CLEANCACHE=y
882 CONFIG_FRONTSWAP=y
883 CONFIG_MEM_SOFT_DIRTY=y
884 CONFIG_ZSWAP=y
885 CONFIG_ZSWAP_COMPRESSOR_DEFAULT_LZO=y
886 CONFIG_ZSWAP_COMPRESSOR_DEFAULT="lzo"
887 CONFIG_ZSWAP_ZPOOL_DEFAULT_ZBUD=y
888 CONFIG_ZSWAP_ZPOOL_DEFAULT="zbud"
889 CONFIG_ZBUD=y
890 CONFIG_ZSMALLOC=y
891 CONFIG_GENERIC_EARLY_IOREMAP=y
892 CONFIG_PAGE_IDLE_FLAG=y
893 CONFIG_IDLE_PAGE_TRACKING=y
894 CONFIG_ARCH_HAS_CACHE_LINE_SIZE=y
895 CONFIG_ARCH_HAS_PTE_DEVMAP=y
896 CONFIG_ARCH_HAS_ZONE_DMA_SET=y
897 CONFIG_ZONE_DMA=y
898 CONFIG_ZONE_DMA32=y
899 CONFIG_ZONE_DEVICE=y
900 CONFIG_DEV_PAGEMAP_OPS=y
901 CONFIG_HMM_MIRROR=y
902 CONFIG_DEVICE_PRIVATE=y
903 CONFIG_ARCH_USES_HIGH_VMA_FLAGS=y
904 CONFIG_ARCH_HAS_PKEYS=y
905 CONFIG_ARCH_HAS_PTE_SPECIAL=y
906 CONFIG_SECRETMEM=y
907
908 #

```

```

909 # Data Access Monitoring
910 #
911 # end of Data Access Monitoring
912 # end of Memory Management options
913
914 CONFIG_NET=y
915 CONFIG_NET_INGRESS=y
916 CONFIG_SKB_EXTENSIONS=y
917
918 #
919 # Networking options
920 #
921 CONFIG_PACKET=y
922 CONFIG_UNIX=y
923 CONFIG_UNIX_SCM=y
924 CONFIG_AF_UNIX_OOB=y
925 CONFIG_XDP_SOCKETS=y
926 CONFIG_INET=y
927 CONFIG_IP_MULTICAST=y
928 CONFIG_IP_ADVANCED_ROUTER=y
929 CONFIG_IP_FIB_TRIE_STATS=y
930 CONFIG_IP_MULTIPLE_TABLES=y
931 CONFIG_IP_ROUTE_MULTIPATH=y
932 CONFIG_IP_ROUTE_VERBOSE=y
933 CONFIG_IP_MROUTE_COMMON=y
934 CONFIG_IP_MROUTE=y
935 CONFIG_IP_MROUTE_MULTIPLE_TABLES=y
936 CONFIG_IP_PIMSM_V1=y
937 CONFIG_IP_PIMSM_V2=y
938 CONFIG_SYN_COOKIES=y
939 CONFIG_INET_TABLE_PERTURB_ORDER=16
940 CONFIG_TCP_CONG_ADVANCED=y
941 CONFIG_TCP_CONG_CUBIC=y
942 CONFIG_DEFAULT_CUBIC=y
943 CONFIG_DEFAULT_TCP_CONG="cubic"
944 CONFIG_TCP_MD5SIG=y
945 CONFIG_IPV6=y
946 CONFIG_IPV6_ROUTER_PREF=y
947 CONFIG_IPV6_ROUTE_INFO=y
948 CONFIG_IPV6_MULTIPLE_TABLES=y
949 CONFIG_IPV6_SUBTREES=y
950 CONFIG_IPV6_MROUTE=y
951 CONFIG_IPV6_MROUTE_MULTIPLE_TABLES=y
952 CONFIG_IPV6_PIMSM_V2=y
953 CONFIG_IPV6_SEG6_LWTUNNEL=y
954 CONFIG_IPV6_SEG6_HMAC=y
955 CONFIG_IPV6_SEG6_BPF=y
956 CONFIG_IPV6_IOAM6_LWTUNNEL=y
957 CONFIG_NETLABEL=y
958 CONFIG_MPTCP=y
959 CONFIG_MPTCP_IPV6=y
960 CONFIG_NETWORK_SECMARK=y
961 CONFIG_NET_PTP_CLASSIFY=y
962 CONFIG_NETWORK_PHY_TIMESTAMPING=y
963 CONFIG_NETFILTER=y
964 CONFIG_NETFILTER_ADVANCED=y
965
966 #
967 # Core Netfilter Configuration
968 #
969 CONFIG_NETFILTER_INGRESS=y
970 CONFIG_NETFILTER_NETLINK=y
971 CONFIG_NETFILTER_NETLINK_HOOK=y
972 CONFIG_NETFILTER_NETLINK_ACCT=y
973 CONFIG_NETFILTER_NETLINK_QUEUE=y
974 CONFIG_NETFILTER_NETLINK_LOG=y
975 CONFIG_NETFILTER_NETLINK_OSF=y
976 CONFIG_NF_CONNTRACK=y
977 CONFIG_NF_LOG_SYSLOG=y
978 CONFIG_NETFILTER_CONNCOUNT=y
979 CONFIG_NF_CONNTRACK_MARK=y
980 CONFIG_NF_CONNTRACK_SECMARK=y
981 CONFIG_NF_CONNTRACK_ZONES=y
982 CONFIG_NF_CONNTRACK_PROCS=y
983 CONFIG_NF_CONNTRACK_EVENTS=y
984 CONFIG_NF_CONNTRACK_TIMEOUT=y
985 CONFIG_NF_CONNTRACK_TIMESTAMP=y
986 CONFIG_NF_CONNTRACK_LABELS=y
987 CONFIG_NF_CT_PROTO_DCCP=y
988 CONFIG_NF_CT_PROTO_SCTP=y
989 CONFIG_NF_CT_PROTO_UDPLITE=y
990 CONFIG_NF_NAT=y
991 CONFIG_NF_NAT_REDIRECT=y
992 CONFIG_NF_NAT_MASQUERADE=y
993 CONFIG_NETFILTER_SYNPROXY=y
994 CONFIG_NF_TABLES=y
995 CONFIG_NF_TABLES_INET=y
996 CONFIG_NF_TABLES_NETDEV=y
997 CONFIG_NFT_NUMGEN=y
998 CONFIG_NFT_CT=y
999 CONFIG_NFT_COUNTER=y
1000 CONFIG_NFT_CONNLIMIT=y
1001 CONFIG_NFT_LOG=y
1002 CONFIG_NFT_LIMIT=y
1003 CONFIG_NFT_MASQ=y

```

```

1004 CONFIG_NFT_REDIR=y
1005 CONFIG_NFT_NAT=y
1006 CONFIG_NFT_TUNNEL=y
1007 CONFIG_NFT_OBJREF=y
1008 CONFIG_NFT_QUEUE=y
1009 CONFIG_NFT_QUOTA=y
1010 CONFIG_NFT_REJECT=y
1011 CONFIG_NFT_REJECT_INET=y
1012 CONFIG_NFT_COMPAT=m
1013 CONFIG_NFT_HASH=y
1014 CONFIG_NFT_SOCKET=y
1015 CONFIG_NFT_OSF=y
1016 CONFIG_NFT_TPROXY=y
1017 CONFIG_NFT_SYNPROXY=y
1018 CONFIG_NF_DUP_NETDEV=y
1019 CONFIG_NFT_DUP_NETDEV=y
1020 CONFIG_NFT_FWD_NETDEV=y
1021 CONFIG_NFT_REJECT_NETDEV=y
1022 CONFIG_NF_FLOW_TABLE=y
1023 CONFIG_NETFILTER_XTABLES=y
1024 CONFIG_NETFILTER_XTABLES_COMPAT=y
1025
1026 #
1027 # Xtables combined modules
1028 #
1029 CONFIG_NETFILTER_XT_MARK=m
1030
1031 #
1032 # Xtables targets
1033 #
1034 CONFIG_NETFILTER_XT_TARGET_HL=y
1035 CONFIG_NETFILTER_XT_NAT=y
1036 CONFIG_NETFILTER_XT_TARGET_NETMAP=m
1037 CONFIG_NETFILTER_XT_TARGET_REDIRECT=m
1038 CONFIG_NETFILTER_XT_TARGET_MASQUERADE=y
1039
1040 #
1041 # Xtables matches
1042 #
1043 CONFIG_NETFILTER_XT_MATCH_HL=y
1044 # end of Core Netfilter Configuration
1045
1046 #
1047 # IP: Netfilter Configuration
1048 #
1049 CONFIG_NF_DEFRAG_IPV4=y
1050 CONFIG_NF_SOCKET_IPV4=y
1051 CONFIG_NF_TPROXY_IPV4=y
1052 CONFIG_NF_TABLES_IPV4=y
1053 CONFIG_NFT_REJECT_IPV4=y
1054 CONFIG_NF_REJECT_IPV4=y
1055 CONFIG_IP_NF_IPTABLES=m
1056 CONFIG_IP_NF_FILTER=m
1057 CONFIG_IP_NF_NAT=m
1058 CONFIG_IP_NF_TARGET_MASQUERADE=m
1059 CONFIG_IP_NF_TARGET_NETMAP=m
1060 CONFIG_IP_NF_TARGET_REDIRECT=m
1061 # end of IP: Netfilter Configuration
1062
1063 #
1064 # IPv6: Netfilter Configuration
1065 #
1066 CONFIG_NF_SOCKET_IPV6=y
1067 CONFIG_NF_TPROXY_IPV6=y
1068 CONFIG_NF_TABLES_IPV6=y
1069 CONFIG_NFT_REJECT_IPV6=y
1070 CONFIG_NF_REJECT_IPV6=y
1071 CONFIG_NF_LOG_IPV6=y
1072 CONFIG_IP6_NF_IPTABLES=y
1073 CONFIG_IP6_NF_MATCH_AH=y
1074 CONFIG_IP6_NF_MATCH_EUI64=y
1075 CONFIG_IP6_NF_MATCH_FRAG=y
1076 CONFIG_IP6_NF_MATCH_OPTS=y
1077 CONFIG_IP6_NF_MATCH_HL=y
1078 CONFIG_IP6_NF_MATCH_IPV6HEADER=y
1079 CONFIG_IP6_NF_MATCH_MH=y
1080 CONFIG_IP6_NF_MATCH_RPFILTER=y
1081 CONFIG_IP6_NF_MATCH_RT=y
1082 CONFIG_IP6_NF_MATCH_SRH=y
1083 CONFIG_IP6_NF_TARGET_HL=y
1084 CONFIG_IP6_NF_FILTER=y
1085 CONFIG_IP6_NF_TARGET_REJECT=y
1086 CONFIG_IP6_NF_TARGET_SYNPROXY=y
1087 CONFIG_IP6_NF_MANGLE=y
1088 CONFIG_IP6_NF_RAW=y
1089 CONFIG_IP6_NF_SECURITY=y
1090 CONFIG_IP6_NF_NAT=y
1091 CONFIG_IP6_NF_TARGET_MASQUERADE=y
1092 CONFIG_IP6_NF_TARGET_NPT=y
1093 # end of IPv6: Netfilter Configuration
1094
1095 CONFIG_NF_DEFRAG_IPV6=y
1096 CONFIG_BPFILTER=y
1097 CONFIG_NET_DSA=y
1098

```

1099	CONFIG_NET_DSA_TAG_OCELOT_8021Q=y	1194	CONFIG_PCI_REALLOC_ENABLE_AUTO=y
1100	CONFIG_VLAN_8021Q=y	1195	CONFIG_PCI_ATS=y
1101	CONFIG_NET_SCHED=y	1196	CONFIG_PCI_LOCKLESS_CONFIG=y
1102		1197	CONFIG_PCI_10V=y
1103	#	1198	CONFIG_PCI_PRI=y
1104	# Queueing/Scheduling	1199	CONFIG_PCI_PASID=y
1105	#	1200	CONFIG_PCI_LABEL=y
1106	CONFIG_NET_SCH_FQ_CODEL=m	1201	CONFIG_PCIE_BUS_DEFAULT=y
1107		1202	CONFIG_HOTPLUG_PCI=y
1108	#	1203	CONFIG_HOTPLUG_PCI_ACPI=y
1109	# Classification	1204	CONFIG_HOTPLUG_PCI_CPCI=y
1110	#	1205	CONFIG_HOTPLUG_PCI_SHPC=y
1111	CONFIG_NET_CLS=y	1206	
1112	CONFIG_NET_EMATCH=y	1207	#
1113	CONFIG_NET_EMATCH_STACK=32	1208	# PCI controller drivers
1114	CONFIG_NET_CLS_ACT=y	1209	#
1115	CONFIG_NET_ACT_NAT=y	1210	
1116	CONFIG_NET_TC_SKB_EXT=y	1211	#
1117	CONFIG_NET_SCH_FIFO=y	1212	# DesignWare PCI Core Support
1118	CONFIG_DCB=y	1213	#
1119	CONFIG_DNS_RESOLVER=y	1214	CONFIG_PCIE_DW=y
1120	CONFIG_MPLS=y	1215	CONFIG_PCIE_DW_HOST=y
1121	CONFIG_NET_SWITCHDEV=y	1216	CONFIG_PCIE_DW_EP=y
1122	CONFIG_NET_L3_MASTER_DEV=y	1217	CONFIG_PCIE_DW_PLAT=y
1123	CONFIG_NET_NCSI=y	1218	CONFIG_PCIE_DW_PLAT_HOST=y
1124	CONFIG_NCSI_OEM_CMD_GET_MAC=y	1219	CONFIG_PCIE_DW_PLAT_EP=y
1125	CONFIG_PCPU_DEV_REFCNT=y	1220	# end of DesignWare PCI Core Support
1126	CONFIG_RPS=y	1221	
1127	CONFIG_RFS_ACCEL=y	1222	#
1128	CONFIG_SOCK_RX_QUEUE_MAPPING=y	1223	# Mobiveil PCIe Core Support
1129	CONFIG_XPS=y	1224	#
1130	CONFIG_CGROUP_NET_PRIO=y	1225	# end of Mobiveil PCIe Core Support
1131	CONFIG_CGROUP_NET_CLASSID=y	1226	
1132	CONFIG_NET_RX_BUSY_POLL=y	1227	#
1133	CONFIG_BQL=y	1228	# Cadence PCIe controllers support
1134	CONFIG_BPF_STREAM_PARSER=y	1229	#
1135	CONFIG_NET_FLOW_LIMIT=y	1230	# end of Cadence PCIe controllers support
1136		1231	# end of PCI controller drivers
1137	#	1232	
1138	# Network testing	1233	#
1139	#	1234	# PCI Endpoint
1140	CONFIG_NET_DROP_MONITOR=y	1235	#
1141	# end of Network testing	1236	CONFIG_PCI_ENDPOINT=y
1142	# end of Networking options	1237	CONFIG_PCI_ENDPOINT_CONFIGFS=y
1143		1238	# end of PCI Endpoint
1144	CONFIG_HAMRADIO=y	1239	
1145		1240	#
1146	#	1241	# PCI switch controller drivers
1147	# Packet Radio protocols	1242	#
1148	#	1243	# end of PCI switch controller drivers
1149	CONFIG_STREAM_PARSER=y	1244	
1150	CONFIG_FIB_RULES=y	1245	CONFIG_RAPIDIO=y
1151	CONFIG_WIRELESS=y	1246	CONFIG_RAPIDIO_DISC_TIMEOUT=30
1152		1247	CONFIG_RAPIDIO_DMA_ENGINE=y
1153	#	1248	
1154	# CFG80211 needs to be enabled for MAC80211	1249	#
1155	#	1250	# RapidIO Switch drivers
1156	CONFIG_MAC80211_STA_HASH_MAX_SIZE=0	1251	#
1157	CONFIG_RFKILL=y	1252	# end of RapidIO Switch drivers
1158	CONFIG_RFKILL_LEDS=y	1253	
1159	CONFIG_RFKILL_INPUT=y	1254	#
1160	CONFIG_LWTUNNEL=y	1255	# Generic Driver Options
1161	CONFIG_LWTUNNEL_BPF=y	1256	#
1162	CONFIG_DST_CACHE=y	1257	CONFIG_AUXILIARY_BUS=y
1163	CONFIG_GRO_CELLS=y	1258	CONFIG_UEVENT_HELPER=y
1164	CONFIG_NET_SELFTESTS=y	1259	CONFIG_UEVENT_HELPER_PATH=""
1165	CONFIG_NET_SOCK_MSG=y	1260	CONFIG_DEVTMPFS=y
1166	CONFIG_NET_DEVLINK=y	1261	CONFIG_DEVTMPFS_MOUNT=y
1167	CONFIG_PAGE_POOL=y	1262	CONFIG_PREVENT_FIRMWARE_BUILD=y
1168	CONFIG_ETHTOOL_NETLINK=y	1263	
1169		1264	#
1170	#	1265	# Firmware loader
1171	# Device Drivers	1266	#
1172	#	1267	CONFIG_FW_LOADER=y
1173	CONFIG_HAVE_EISA=y	1268	CONFIG_FW_LOADER_PAGED_BUF=y
1174	CONFIG_EISA=y	1269	CONFIG_EXTRA_FIRMWARE=""
1175	CONFIG_EISA_VLB_PRIMING=y	1270	CONFIG_FW_LOADER_USER_HELPER=y
1176	CONFIG_EISA_PCI_EISA=y	1271	CONFIG_FW_LOADER_COMPRESS=y
1177	CONFIG_EISA_VIRTUAL_ROOT=y	1272	CONFIG_FW_CACHE=y
1178	CONFIG_EISA_NAMES=y	1273	# end of Firmware loader
1179	CONFIG_HAVE_PCI=y	1274	
1180	CONFIG_PCI=y	1275	CONFIG_WANT_DEV_COREDUMP=y
1181	CONFIG_PCI_DOMAINS=y	1276	CONFIG_ALLOW_DEV_COREDUMP=y
1182	CONFIG_PCIEPORTBUS=y	1277	CONFIG_DEV_COREDUMP=y
1183	CONFIG_HOTPLUG_PCI_PCIE=y	1278	CONFIG_HMEM_REPORTING=y
1184	CONFIG_PCIEAER=y	1279	CONFIG_SYS_HYPERVISOR=y
1185	CONFIG_PCIEASPM=y	1280	CONFIG_GENERIC_CPU_AUTOPROBE=y
1186	CONFIG_PCIEASPM_DEFAULT=y	1281	CONFIG_GENERIC_CPU_VULNERABILITIES=y
1187	CONFIG_PCIE_PME=y	1282	CONFIG_REGMAP=y
1188	CONFIG_PCIE_DPC=y	1283	CONFIG_REGMAP_I2C=y
1189	CONFIG_PCIE_PTM=y	1284	CONFIG_REGMAP_SPI=y
1190	CONFIG_PCIE_EDR=y	1285	CONFIG_REGMAP_MMIO=y
1191	CONFIG_PCI_MSI=y	1286	CONFIG_REGMAP_IRQ=y
1192	CONFIG_PCI_MSI_IRQ_DOMAIN=y	1287	CONFIG_DMA_SHARED_BUFFER=y
1193	CONFIG_PCI_QUIRKS=y	1288	# end of Generic Driver Options

1289		1380	#
1290	#	1381	# SCSI device support
1291	# Bus devices	1382	#
1292	#	1383	CONFIG_SCSI_MOD=y
1293	# end of Bus devices	1384	CONFIG_SCSI_COMMON=y
1294		1385	CONFIG_SCSI=y
1295	CONFIG_CONNECTOR=y	1386	CONFIG_SCSI_DMA=y
1296	CONFIG_PROC_EVENTS=y	1387	CONFIG_SCSI_PROC_FS=y
1297		1388	
1298	#	1389	#
1299	# Firmware Drivers	1390	# SCSI support type (disk, tape, CD-ROM)
1300	#	1391	#
1301		1392	CONFIG_BLK_DEV_SD=y
1302	#	1393	CONFIG_BLK_DEV_SR=y
1303	# ARM System Control and Management Interface Protocol	1394	CONFIG_CHR_DEV_SG=y
1304	#	1395	CONFIG_BLK_DEV_BSG=y
1305	# end of ARM System Control and Management Interface Protocol	1396	CONFIG_SCSI_CONSTANTS=y
1306		1397	CONFIG_SCSI_LOGGING=y
1307	CONFIG_EDD=y	1398	CONFIG_SCSI_SCAN_ASYNC=y
1308	CONFIG_EDD_OFF=y	1399	
1309	CONFIG_FIRMWARE_MEMMAP=y	1400	#
1310	CONFIG_DMIID=y	1401	# SCSI Transports
1311	CONFIG_DMI_SCAN_MACHINE_NON_EFI_FALLBACK=y	1402	#
1312	CONFIG_SYSFB=y	1403	# end of SCSI Transports
1313		1404	
1314	#	1405	CONFIG_SCSI_LOWLEVEL=y
1315	# EFI (Extensible Firmware Interface) Support	1406	CONFIG_MEGARAID_NEWGEN=y
1316	#	1407	CONFIG_MEGARAID_SAS=m
1317	CONFIG_EFI_VARS=y	1408	CONFIG_SCSI_DH=y
1318	CONFIG_EFI_ESRT=y	1409	CONFIG_SCSI_DH_RDAC=m
1319	CONFIG_EFI_VARS_PSTORE=m	1410	CONFIG_SCSI_DH_EMCM=m
1320	CONFIG_EFI_RUNTIME_MAP=y	1411	CONFIG_SCSI_DH_ALUA=m
1321	CONFIG_EFI_SOFT_RESERVE=y	1412	# end of SCSI device support
1322	CONFIG_EFI_RUNTIME_WRAPPERS=y	1413	
1323	CONFIG_EFI_GENERIC_STUB_INITRD_CMDLINE_LOADER=y	1414	CONFIG_ATA=y
1324	CONFIG_APPLE_PROPERTIES=y	1415	CONFIG_SATA_HOST=y
1325	CONFIG_RESET_ATTACK_MITIGATION=y	1416	CONFIG_PATA_TIMINGS=y
1326	CONFIG_EFI_RC12_TABLE=y	1417	CONFIG_ATA_VERBOSE_ERROR=y
1327	# end of EFI (Extensible Firmware Interface) Support	1418	CONFIG_ATA_FORCE=y
1328		1419	CONFIG_ATA_ACPI=y
1329	CONFIG_UEFI_CPER=y	1420	CONFIG_SATA_ZPODD=y
1330	CONFIG_UEFI_CPER_X86=y	1421	CONFIG_SATA_PMP=y
1331	CONFIG_EFI_DEV_PATH_PARSER=y	1422	
1332	CONFIG_EFI_EARLYCON=y	1423	#
1333	CONFIG_EFI_CUSTOM_SSDT_OVERLAYS=y	1424	# Controllers with non-SFF native interface
1334		1425	#
1335	#	1426	CONFIG_SATA_AHCI=m
1336	# Tegra firmware driver	1427	CONFIG_SATA_MOBILE_LPM_POLICY=3
1337	#	1428	CONFIG_SATA_AHCI_PLATFORM=m
1338	# end of Tegra firmware driver	1429	CONFIG_SATA_ACARD_AHCI=m
1339	# end of Firmware Drivers	1430	CONFIG_ATA_SFF=y
1340		1431	
1341	CONFIG_ARCH_MIGHT_HAVE_PC_PARPORT=y	1432	#
1342	CONFIG_PNP=y	1433	# SFF controllers with custom DMA interface
1343		1434	#
1344	#	1435	CONFIG_ATA_BMDMA=y
1345	# Protocols	1436	
1346	#	1437	#
1347	CONFIG_PNPACPI=y	1438	# SATA SFF controllers with BMDMA
1348	CONFIG_BLK_DEV=y	1439	#
1349	CONFIG_CDROM=y	1440	CONFIG_ATA_PIIX=y
1350	CONFIG_BLK_DEV_LOOP=y	1441	
1351	CONFIG_BLK_DEV_LOOP_MIN_COUNT=8	1442	#
1352	CONFIG_XEN_BLKDEV_FRONTEND=y	1443	# PATA SFF controllers with BMDMA
1353		1444	#
1354	#	1445	CONFIG_PATA_SIS=y
1355	# NVME Support	1446	
1356	#	1447	#
1357	# end of NVME Support	1448	# PIO-only SFF controllers
1358		1449	#
1359	#	1450	
1360	# Misc devices	1451	#
1361	#	1452	# Generic fallback / legacy drivers
1362	CONFIG_SRAM=y	1453	#
1363		1454	CONFIG_ATA_GENERIC=y
1364	#	1455	CONFIG_MD=y
1365	# EEPROM support	1456	CONFIG_BLK_DEV_MD=y
1366	#	1457	CONFIG_MD_AUTODETECT=y
1367	# end of EEPROM support	1458	CONFIG_MD_LINEAR=m
1368		1459	CONFIG_MD_RAID0=m
1369		1460	CONFIG_MD_RAID1=m
1370	#	1461	CONFIG_MD_RAID10=m
1371	# Texas Instruments shared transport line discipline	1462	CONFIG_MD_RAID456=m
1372	#	1463	CONFIG_MD_MULTIPATH=m
1373	# end of Texas Instruments shared transport line discipline	1464	CONFIG_BLK_DEV_DM_BUILTIN=y
1374		1465	CONFIG_BLK_DEV_DM=y
1375	CONFIG_INTEL_MEI=m	1466	CONFIG_DM_MULTIPATH=m
1376	CONFIG_INTEL_MEI_ME=m	1467	CONFIG_DM_INIT=y
1377	CONFIG_PVPANIC=y	1468	CONFIG_DM_UEVENT=y
1378	# end of Misc devices	1469	CONFIG_FUSION=y
1379		1470	CONFIG_FUSION_MAX_SGE=128
		1471	CONFIG_FUSION_LOGGING=y
		1472	
		1473	#
		1474	# IEEE 1394 (FireWire) support

1475	#	1570	CONFIG_SWPHY=y
1476	# end of IEEE 1394 (FireWire) support	1571	CONFIG_LED_TRIGGER_PHY=y
1477		1572	CONFIG_FIXED_PHY=y
1478	CONFIG_MACINTOSH_DRIVERS=y	1573	
1479	CONFIG_MAC_EMUMOUSEBTN=m	1574	#
1480	CONFIG_NETDEVICES=y	1575	# MII PHY device drivers
1481	CONFIG_NET_CORE=y	1576	#
1482	CONFIG_NET_FC=y	1577	CONFIG_BCM84881_PHY=y
1483	CONFIG_TUN=y	1578	CONFIG_MDIO_DEVICE=y
1484	CONFIG_NET_VRF=y	1579	CONFIG_MDIO_BUS=y
1485		1580	CONFIG_FWNODE_MDIO=y
1486	#	1581	CONFIG_ACPI_MDIO=y
1487	# Distributed Switch Architecture drivers	1582	CONFIG_MDIO_DEVRES=y
1488	#	1583	
1489	# end of Distributed Switch Architecture drivers	1584	#
1490		1585	# MDIO Multiplexers
1491	CONFIG_ETHERNET=y	1586	#
1492	CONFIG_MDIO=y	1587	
1493	CONFIG_NET_VENDOR_3COM=y	1588	#
1494	CONFIG_NET_VENDOR_ADAPTEC=y	1589	# PCS device drivers
1495	CONFIG_NET_VENDOR_AGERE=y	1590	#
1496	CONFIG_NET_VENDOR_ALACRITECH=y	1591	# end of PCS device drivers
1497	CONFIG_NET_VENDOR_ALTEON=y	1592	
1498	CONFIG_NET_VENDOR_AMAZON=y	1593	CONFIG_PPP=y
1499	CONFIG_NET_VENDOR_AMD=y	1594	CONFIG_PPP_FILTER=y
1500	CONFIG_NET_VENDOR_AQUANTIA=y	1595	CONFIG_PPP_MULTILINK=y
1501	CONFIG_NET_VENDOR_ARC=y	1596	CONFIG_SLHC=y
1502	CONFIG_NET_VENDOR_ATHEROS=y	1597	CONFIG_WLAN=y
1503	CONFIG_NET_VENDOR_BROADCOM=y	1598	CONFIG_WLAN_VENDOR_ADMTEK=y
1504	CONFIG_TIGON3=m	1599	CONFIG_WLAN_VENDOR_ATH=y
1505	CONFIG_TIGON3_HWMON=y	1600	CONFIG_ATH5K_PCI=y
1506	CONFIG_NET_VENDOR_CADENCE=y	1601	CONFIG_WLAN_VENDOR_ATMEL=y
1507	CONFIG_NET_VENDOR_CAVIUM=y	1602	CONFIG_WLAN_VENDOR_BROADCOM=y
1508	CONFIG_NET_VENDOR_CHELSIO=y	1603	CONFIG_WLAN_VENDOR_CISCO=y
1509	CONFIG_NET_VENDOR_CIRRUS=y	1604	CONFIG_WLAN_VENDOR_INTEL=y
1510	CONFIG_NET_VENDOR_CISCO=y	1605	CONFIG_WLAN_VENDOR_INTERSIL=y
1511	CONFIG_NET_VENDOR_CORTINA=y	1606	CONFIG_WLAN_VENDOR_MARVELL=y
1512	CONFIG_NET_VENDOR_DEC=y	1607	CONFIG_WLAN_VENDOR_MEDIATEK=y
1513	CONFIG_NET_TULIP=y	1608	CONFIG_WLAN_VENDOR_MICROCHIP=y
1514	CONFIG_NET_VENDOR_DLINK=y	1609	CONFIG_WLAN_VENDOR_RALINK=y
1515	CONFIG_NET_VENDOR_EMULEX=y	1610	CONFIG_WLAN_VENDOR_REALTEK=y
1516	CONFIG_NET_VENDOR_EZCHIP=y	1611	CONFIG_WLAN_VENDOR_RSI=y
1517	CONFIG_NET_VENDOR_GOOGLE=y	1612	CONFIG_WLAN_VENDOR_ST=y
1518	CONFIG_NET_VENDOR_HUAWEI=y	1613	CONFIG_WLAN_VENDOR_TI=y
1519	CONFIG_NET_VENDOR_I825XX=y	1614	CONFIG_WLAN_VENDOR_ZYDAS=y
1520	CONFIG_NET_VENDOR_INTEL=y	1615	CONFIG_WLAN_VENDOR_QUANTENNA=y
1521	CONFIG_IXGBE=y	1616	CONFIG_WAN=y
1522	CONFIG_IXGBE_HWMON=y	1617	
1523	CONFIG_IXGBE_DCB=y	1618	#
1524	CONFIG_IXGBEVF=y	1619	# Wireless WAN
1525	CONFIG_I4OE=m	1620	#
1526	CONFIG_I4OE_DCB=y	1621	CONFIG_WWAN=y
1527	CONFIG_ICE=m	1622	# end of Wireless WAN
1528	CONFIG_NET_VENDOR_LITEX=y	1623	
1529	CONFIG_NET_VENDOR_MARVELL=y	1624	CONFIG_XEN_NETDEV_FRONTEND=y
1530	CONFIG_NET_VENDOR_MELLANOX=y	1625	CONFIG_ISDN=y
1531	CONFIG_NET_VENDOR_MICREL=y	1626	
1532	CONFIG_NET_VENDOR_MICROCHIP=y	1627	#
1533	CONFIG_NET_VENDOR_MICROSEMI=y	1628	# Input device support
1534	CONFIG_NET_VENDOR_MICROSOFT=y	1629	#
1535	CONFIG_NET_VENDOR_MYRI=y	1630	CONFIG_INPUT=y
1536	CONFIG_NET_VENDOR_NI=y	1631	CONFIG_INPUT_LEDS=m
1537	CONFIG_NET_VENDOR_NATSEMI=y	1632	
1538	CONFIG_NET_VENDOR_NETERION=y	1633	#
1539	CONFIG_NET_VENDOR_NETRONOME=y	1634	# Userland interfaces
1540	CONFIG_NET_VENDOR_8390=y	1635	#
1541	CONFIG_NET_VENDOR_NVIDIA=y	1636	CONFIG_INPUT_MOUSEDEV=y
1542	CONFIG_NET_VENDOR_OKI=y	1637	CONFIG_INPUT_MOUSEDEV_PSAUX=y
1543	CONFIG_NET_VENDOR_PACKET_ENGINES=y	1638	CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
1544	CONFIG_NET_VENDOR_PENSANDO=y	1639	CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
1545	CONFIG_NET_VENDOR_QLOGIC=y	1640	CONFIG_INPUT_JOYDEV=m
1546	CONFIG_NET_VENDOR_BROCADE=y	1641	CONFIG_INPUT_EVDEV=y
1547	CONFIG_NET_VENDOR_QUALCOMM=y	1642	
1548	CONFIG_NET_VENDOR_RDC=y	1643	#
1549	CONFIG_NET_VENDOR_REALTEK=y	1644	# Input Device Drivers
1550	CONFIG_NET_VENDOR_RENESAS=y	1645	#
1551	CONFIG_NET_VENDOR_ROCKER=y	1646	CONFIG_INPUT_KEYBOARD=y
1552	CONFIG_NET_VENDOR_SAMSUNG=y	1647	CONFIG_KEYBOARD_ATKBD=y
1553	CONFIG_NET_VENDOR_SEEQ=y	1648	CONFIG_INPUT_MOUSE=y
1554	CONFIG_NET_VENDOR_SILAN=y	1649	CONFIG_INPUT_JOYSTICK=y
1555	CONFIG_NET_VENDOR_SIS=y	1650	CONFIG_INPUT_TABLET=y
1556	CONFIG_NET_VENDOR_SOLARFLARE=y	1651	CONFIG_INPUT_TOUCHSCREEN=y
1557	CONFIG_NET_VENDOR_SMSC=y	1652	CONFIG_TOUCHSCREEN_ELAN=y
1558	CONFIG_NET_VENDOR_SOCIONEXT=y	1653	CONFIG_INPUT_MISC=y
1559	CONFIG_NET_VENDOR_STMICRO=y	1654	CONFIG_INPUT_UINPUT=y
1560	CONFIG_NET_VENDOR_SUN=y	1655	
1561	CONFIG_NET_VENDOR_SYNOPSYS=y	1656	#
1562	CONFIG_NET_VENDOR_TEHUTI=y	1657	# Hardware I/O ports
1563	CONFIG_NET_VENDOR_TI=y	1658	#
1564	CONFIG_NET_VENDOR_VIA=y	1659	CONFIG_SERIO=y
1565	CONFIG_NET_VENDOR_WIZNET=y	1660	CONFIG_ARCH_MIGHT_HAVE_PC_SERIO=y
1566	CONFIG_NET_VENDOR_XILINK=y	1661	CONFIG_SERIO_I8042=y
1567	CONFIG_FDDI=y	1662	CONFIG_SERIO_LIBPS2=y
1568	CONFIG_PHYLINK=y	1663	# end of Hardware I/O ports
1569	CONFIG_PHYLIB=y	1664	# end of Input device support

```

1665
1666 #
1667 # Character devices
1668 #
1669 CONFIG_TTY=y
1670 CONFIG_VT=y
1671 CONFIG_CONSOLE_TRANSLATIONS=y
1672 CONFIG_VT_CONSOLE=y
1673 CONFIG_VT_CONSOLE_SLEEP=y
1674 CONFIG_HW_CONSOLE=y
1675 CONFIG_VT_HW_CONSOLE_BINDING=y
1676 CONFIG_UNIX98_PTYS=y
1677 CONFIG_LEGACY_PTYS=y
1678 CONFIG_LEGACY_PTY_COUNT=0
1679 CONFIG_LDISC_AUTOLOAD=y
1680
1681 #
1682 # Serial drivers
1683 #
1684 CONFIG_SERIAL_EARLYCON=y
1685 CONFIG_SERIAL_8250=y
1686 CONFIG_SERIAL_8250_PNP=y
1687 CONFIG_SERIAL_8250_16550A_VARIANTS=y
1688 CONFIG_SERIAL_8250_FINTK=y
1689 CONFIG_SERIAL_8250_CONSOLE=y
1690 CONFIG_SERIAL_8250_DMA=y
1691 CONFIG_SERIAL_8250_PCI=y
1692 CONFIG_SERIAL_8250_NR_UARTS=48
1693 CONFIG_SERIAL_8250_RUNTIME_UARTS=32
1694 CONFIG_SERIAL_8250_EXTENDED=y
1695 CONFIG_SERIAL_8250_MANY_PORTS=y
1696 CONFIG_SERIAL_8250_SHARE_IRQ=y
1697 CONFIG_SERIAL_8250_RSA=y
1698 CONFIG_SERIAL_8250_RT288X=y
1699 CONFIG_SERIAL_8250_MID=y
1700
1701 #
1702 # Non-8250 serial port support
1703 #
1704 CONFIG_SERIAL_KGDB_NMI=y
1705 CONFIG_SERIAL_MAX310X=y
1706 CONFIG_SERIAL_CORE=y
1707 CONFIG_SERIAL_CORE_CONSOLE=y
1708 CONFIG_CONSOLE_POLL=y
1709 CONFIG_SERIAL_SCCNXP=y
1710 CONFIG_SERIAL_SCCNXP_CONSOLE=y
1711 # end of Serial drivers
1712
1713 CONFIG_SERIAL_MCTRL_GPIO=y
1714 CONFIG_SERIAL_NONSTANDARD=y
1715 CONFIG_HVC_DRIVER=y
1716 CONFIG_HVC_IRQ=y
1717 CONFIG_HVC_XEN=y
1718 CONFIG_HVC_XEN_FRONTEND=y
1719 CONFIG_SERIAL_DEV_BUS=y
1720 CONFIG_SERIAL_DEV_CTRL_TTYPORT=y
1721 CONFIG_TTY_PRINTK=y
1722 CONFIG_TTY_PRINTK_LEVEL=6
1723 CONFIG_VIRTIO_CONSOLE=y
1724 CONFIG_IPMI_HANDLER=m
1725 CONFIG_IPMI_DMI_DECODE=y
1726 CONFIG_IPMI_PLAT_DATA=y
1727 CONFIG_IPMI_DEVICE_INTERFACE=m
1728 CONFIG_IPMI_SI=m
1729 CONFIG_IPMI_SSIF=m
1730 CONFIG_HW_RANDOM=y
1731 CONFIG_DEVMEM=y
1732 CONFIG_DEVPORT=y
1733 CONFIG_HPET=y
1734 CONFIG_HPET_MMAP=y
1735 CONFIG_HPET_MMAP_DEFAULT=y
1736 CONFIG_TCG_TPM=y
1737 CONFIG_HW_RANDOM_TPM=y
1738 CONFIG_TCG_TIS_CORE=y
1739 CONFIG_TCG_TIS=y
1740 CONFIG_TCG_CRB=y
1741 CONFIG_RANDOM_TRUST_CPU=y
1742 CONFIG_RANDOM_TRUST_BOOTLOADER=y
1743 # end of Character devices
1744
1745 #
1746 # I2C support
1747 #
1748 CONFIG_I2C=y
1749 CONFIG_ACPI_I2C_OPREGION=y
1750 CONFIG_I2C_BOARDINFO=y
1751 CONFIG_I2C_COMPAT=y
1752 CONFIG_I2C_CHARDEV=y
1753 CONFIG_I2C_HELPER_AUTO=y
1754 CONFIG_I2C_SMBUS=m
1755 CONFIG_I2C_ALGOBIT=m
1756
1757 #
1758 # I2C Hardware Bus support
1759 #

```

```

1760
1761 #
1762 # PC SMBus host controller drivers
1763 #
1764 CONFIG_I2C_I801=m
1765
1766 #
1767 # ACPI drivers
1768 #
1769
1770 #
1771 # I2C system bus drivers (mostly embedded / system-
    on-chip)
1772 #
1773 CONFIG_I2C_DESIGNWARE_CORE=y
1774 CONFIG_I2C_DESIGNWARE_PLATFORM=y
1775 CONFIG_I2C_DESIGNWARE_BAYTRAIL=y
1776
1777 #
1778 # External I2C/SMBus adapter drivers
1779 #
1780
1781 #
1782 # Other I2C/SMBus bus drivers
1783 #
1784 # end of I2C Hardware Bus support
1785
1786 # end of I2C support
1787
1788 CONFIG_SPI=y
1789 CONFIG_SPI_MASTER=y
1790 CONFIG_SPI_MEM=y
1791
1792 #
1793 # SPI Master Controller Drivers
1794 #
1795
1796 #
1797 # SPI Multiplexer support
1798 #
1799
1800 #
1801 # SPI Protocol Masters
1802 #
1803 CONFIG_SPI_SLAVE=y
1804 CONFIG_SPI_DYNAMIC=y
1805 CONFIG_PPS=y
1806
1807 #
1808 # PPS clients support
1809 #
1810
1811 #
1812 # PPS generators support
1813 #
1814
1815 #
1816 # PTP clock support
1817 #
1818 CONFIG_PTP_1588_CLOCK=y
1819 CONFIG_PTP_1588_CLOCK_OPTIONAL=y
1820 # end of PTP clock support
1821
1822 CONFIG_PINCTRL=y
1823 CONFIG_PINMUX=y
1824 CONFIG_PINCONF=y
1825 CONFIG_GENERIC_PINCONF=y
1826 CONFIG_PINCTRL_AMD=y
1827 CONFIG_PINCTRL_SX150X=y
1828 CONFIG_PINCTRL_BAYTRAIL=y
1829 CONFIG_PINCTRL_CHERRYVIEW=y
1830 CONFIG_PINCTRL_INTEL=y
1831
1832 #
1833 # Renesas pinctrl drivers
1834 #
1835 # end of Renesas pinctrl drivers
1836
1837 CONFIG_GPIOLIB=y
1838 CONFIG_GPIOLIB_FASTPATH_LIMIT=512
1839 CONFIG_GPIO_ACPI=y
1840 CONFIG_GPIOLIB_IRQCHIP=y
1841 CONFIG_GPIO_SYSFS=y
1842 CONFIG_GPIO_CDEV=y
1843 CONFIG_GPIO_CDEV_V1=y
1844
1845 #
1846 # Memory mapped GPIO drivers
1847 #
1848 # end of Memory mapped GPIO drivers
1849
1850 #
1851 # Port-mapped I/O GPIO drivers
1852 #
1853 # end of Port-mapped I/O GPIO drivers

```

```

1854
1855 #
1856 # I2C GPIO expanders
1857 #
1858 # end of I2C GPIO expanders
1859
1860 #
1861 # MFD GPIO expanders
1862 #
1863 CONFIG_GPIO_CRYSTAL_COVE=y
1864 CONFIG_GPIO_PALMAS=y
1865 CONFIG_GPIO_RC5T583=y
1866 CONFIG_GPIO_TPS6586X=y
1867 CONFIG_GPIO_TPS65910=y
1868 # end of MFD GPIO expanders
1869
1870 #
1871 # PCI GPIO expanders
1872 #
1873 # end of PCI GPIO expanders
1874
1875 #
1876 # SPI GPIO expanders
1877 #
1878 # end of SPI GPIO expanders
1879
1880 #
1881 # USB GPIO expanders
1882 #
1883 # end of USB GPIO expanders
1884
1885 #
1886 # Virtual GPIO drivers
1887 #
1888 # end of Virtual GPIO drivers
1889
1890 CONFIG_POWER_RESET=y
1891 CONFIG_POWER_RESET_RESTART=y
1892 CONFIG_POWER_SUPPLY=y
1893 CONFIG_POWER_SUPPLY_HWMON=y
1894 CONFIG_CHARGER_MANAGER=y
1895 CONFIG_HWMON=y
1896
1897 #
1898 # Native drivers
1899 #
1900 CONFIG_SENSORS_CORETEMP=m
1901
1902 #
1903 # ACPI drivers
1904 #
1905 CONFIG_SENSORS_ACPI_POWER=m
1906 CONFIG_THERMAL=y
1907 CONFIG_THERMAL_NETLINK=y
1908 CONFIG_THERMAL_STATISTICS=y
1909 CONFIG_THERMAL_EMERGENCY_POWEROFF_DELAY_MS=0
1910 CONFIG_THERMAL_HWMON=y
1911 CONFIG_THERMAL_WRITABLE_TRIPS=y
1912 CONFIG_THERMAL_DEFAULT_GOV_STEP_WISE=y
1913 CONFIG_THERMAL_GOV_FAIR_SHARE=y
1914 CONFIG_THERMAL_GOV_STEP_WISE=y
1915 CONFIG_THERMAL_GOV_BANG_BANG=y
1916 CONFIG_THERMAL_GOV_USER_SPACE=y
1917 CONFIG_THERMAL_GOV_POWER_ALLOCATOR=y
1918 CONFIG_DEVFREQ_THERMAL=y
1919 CONFIG_THERMAL_EMULATION=y
1920
1921 #
1922 # Intel thermal drivers
1923 #
1924 CONFIG_INTEL_POWERCLAMP=m
1925 CONFIG_X86_THERMAL_VECTOR=y
1926 CONFIG_X86_PKG_TEMP_THERMAL=m
1927
1928 #
1929 # ACPI INT340X thermal drivers
1930 #
1931 # end of ACPI INT340X thermal drivers
1932
1933 CONFIG_INTEL_PCH_THERMAL=m
1934 # end of Intel thermal drivers
1935
1936 CONFIG_WATCHDOG=y
1937 CONFIG_WATCHDOG_CORE=y
1938 CONFIG_WATCHDOG_HANDLE_BOOT_ENABLED=y
1939 CONFIG_WATCHDOG_OPEN_TIMEOUT=0
1940 CONFIG_WATCHDOG_SYSFS=y
1941
1942 #
1943 # Watchdog Pretimeout Governors
1944 #
1945 CONFIG_WATCHDOG_PRETIMEOUT_GOV=y
1946 CONFIG_WATCHDOG_PRETIMEOUT_GOV_SEL=m
1947 CONFIG_WATCHDOG_PRETIMEOUT_GOV_NOOP=y
1948 CONFIG_WATCHDOG_PRETIMEOUT_DEFAULT_GOV_NOOP=y

```

```

1949
1950 #
1951 # Watchdog Device Drivers
1952 #
1953
1954 #
1955 # PCI-based Watchdog Cards
1956 #
1957
1958 #
1959 # USB-based Watchdog Cards
1960 #
1961 CONFIG_SSB_POSSIBLE=y
1962 CONFIG_BCMA_POSSIBLE=y
1963
1964 #
1965 # Multifunction device drivers
1966 #
1967 CONFIG_MFD_CORE=y
1968 CONFIG_MFD_AS3711=y
1969 CONFIG_PMIC_ADP5520=y
1970 CONFIG_MFD_AAT2870_CORE=y
1971 CONFIG_PMIC_DA903X=y
1972 CONFIG_PMIC_DA9052=y
1973 CONFIG_MFD_DA9052_SPI=y
1974 CONFIG_MFD_DA9052_I2C=y
1975 CONFIG_MFD_DA9055=y
1976 CONFIG_MFD_DA9063=y
1977 CONFIG_HTC_I2CPLD=y
1978 CONFIG_INTEL_SOC_PMIC=y
1979 CONFIG_INTEL_SOC_PMIC_CHTWC=y
1980 CONFIG_MFD_INTEL_PMT=m
1981 CONFIG_MFD_88PM860X=y
1982 CONFIG_MFD_MAX14577=y
1983 CONFIG_MFD_MAX77693=y
1984 CONFIG_MFD_MAX77843=y
1985 CONFIG_MFD_MAX8925=y
1986 CONFIG_MFD_MAX8997=y
1987 CONFIG_MFD_MAX8998=y
1988 CONFIG_EZX_PCAP=y
1989 CONFIG_MFD_RC5T583=y
1990 CONFIG_MFD_SYSCON=y
1991 CONFIG_MFD_LP8788=y
1992 CONFIG_MFD_PALMAS=y
1993 CONFIG_MFD_TPS65090=y
1994 CONFIG_MFD_TPS6586X=y
1995 CONFIG_MFD_TPS65910=y
1996 CONFIG_MFD_TPS65912=y
1997 CONFIG_MFD_TPS65912_I2C=y
1998 CONFIG_MFD_TPS65912_SPI=y
1999 CONFIG_MFD_TPS80031=y
2000 CONFIG_TWL4030_CORE=y
2001 CONFIG_MFD_TWL4030_AUDI0=y
2002 CONFIG_TWL6040_CORE=y
2003 CONFIG_MFD_WM8400=y
2004 CONFIG_MFD_WM831X=y
2005 CONFIG_MFD_WM831X_I2C=y
2006 CONFIG_MFD_WM831X_SPI=y
2007 CONFIG_MFD_WM8350=y
2008 CONFIG_MFD_WM8350_I2C=y
2009 # end of Multifunction device drivers
2010
2011 CONFIG_REGULATOR=y
2012 CONFIG_RC_CORE=m
2013 CONFIG_LIRC=y
2014 CONFIG_RC_DECODERS=y
2015 CONFIG_RC_DEVICES=y
2016 CONFIG_CEC_CORE=m
2017 CONFIG_MEDIA_CEC_RC=y
2018 CONFIG_MEDIA_CEC_SUPPORT=y
2019
2020 #
2021 # Graphics support
2022 #
2023 CONFIG_AGP=y
2024 CONFIG_AGP_AMD64=y
2025 CONFIG_AGP_INTEL=y
2026 CONFIG_AGP_VIA=y
2027 CONFIG_INTEL_GTT=y
2028 CONFIG_VGA_ARB=y
2029 CONFIG_VGA_ARB_MAX_GPUS=16
2030 CONFIG_VGA_SWITCHEROO=y
2031 CONFIG_DRM=m
2032 CONFIG_DRM_DP_AUX_CHARDEV=y
2033 CONFIG_DRM_KMS_HELPER=m
2034 CONFIG_DRM_FBDEV_EMULATION=y
2035 CONFIG_DRM_FBDEV_OVERALLOC=100
2036 CONFIG_DRM_LOAD_EDID_FIRMWARE=y
2037 CONFIG_DRM_DP_CEC=y
2038 CONFIG_DRM_GEM_SHMEM_HELPER=y
2039
2040 #
2041 # I2C encoder or helper chips
2042 #
2043 # end of I2C encoder or helper chips

```


2044		2139	
2045	#	2140	#
2046	# ARM devices	2141	# USB HID Boot Protocol drivers
2047	#	2142	#
2048	# end of ARM devices	2143	# end of USB HID Boot Protocol drivers
2049		2144	# end of USB HID support
2050	CONFIG_DRM_MGAG200=m	2145	
2051	CONFIG_DRM_PANEL=y	2146	#
2052		2147	# I2C HID support
2053	#	2148	#
2054	# Display Panels	2149	# end of I2C HID support
2055	#	2150	
2056	# end of Display Panels	2151	#
2057		2152	# Intel ISH HID support
2058	CONFIG_DRM_BRIDGE=y	2153	#
2059	CONFIG_DRM_PANEL_BRIDGE=y	2154	# end of Intel ISH HID support
2060		2155	
2061	#	2156	#
2062	# Display Interface Bridges	2157	# AMD SFH HID Support
2063	#	2158	#
2064	# end of Display Interface Bridges	2159	# end of AMD SFH HID Support
2065		2160	# end of HID support
2066	CONFIG_DRM_PANEL_ORIENTATION_QUIRKS=y	2161	
2067		2162	CONFIG_USB_OHCI_LITTLE_ENDIAN=y
2068	#	2163	CONFIG_USB_SUPPORT=y
2069	# Frame buffer Devices	2164	CONFIG_USB_COMMON=y
2070	#	2165	CONFIG_USB_LED_TRIG=y
2071	CONFIG_FB_CMDLINE=y	2166	CONFIG_USB_ARCH_HAS_HCD=y
2072	CONFIG_FB_NOTIFY=y	2167	CONFIG_USB=y
2073	CONFIG_FB=y	2168	CONFIG_USB_PCI=y
2074	CONFIG_FIRMWARE_EDID=y	2169	CONFIG_USB_ANNOUNCE_NEW_DEVICES=y
2075	CONFIG_FB_BOOT_VESA_SUPPORT=y	2170	
2076	CONFIG_FB_CFB_FILLRECT=y	2171	#
2077	CONFIG_FB_CFB_COPYAREA=y	2172	# Miscellaneous USB options
2078	CONFIG_FB_CFB_IMAGEBLIT=y	2173	#
2079	CONFIG_FB_SYS_FILLRECT=m	2174	CONFIG_USB_DEFAULT_PERSIST=y
2080	CONFIG_FB_SYS_COPYAREA=m	2175	CONFIG_USB_DYNAMIC_MINORS=y
2081	CONFIG_FB_SYS_IMAGEBLIT=m	2176	CONFIG_USB_AUTOSUSPEND_DELAY=2
2082	CONFIG_FB_SYS_FOPS=m	2177	
2083	CONFIG_FB_DEFERRED_IO=y	2178	#
2084	CONFIG_FB_MODE_HELPERS=y	2179	# USB Host Controller Drivers
2085	CONFIG_FB_TILEBLITTING=y	2180	#
2086		2181	CONFIG_USB_XHCI_HCD=y
2087	#	2182	CONFIG_USB_XHCI_DBGCAP=y
2088	# Frame buffer hardware drivers	2183	CONFIG_USB_XHCI_PCI=m
2089	#	2184	CONFIG_USB_XHCI_PCI_RENESAS=m
2090	CONFIG_FB_ASILANT=y	2185	CONFIG_USB_EHCI_HCD=y
2091	CONFIG_FB_IMSTT=y	2186	CONFIG_USB_EHCI_ROOT_HUB_TT=y
2092	CONFIG_FB_VESA=y	2187	CONFIG_USB_EHCI_TT_NEWSCHED=y
2093	CONFIG_FB_EFI=y	2188	CONFIG_USB_EHCI_PCI=y
2094	# end of Frame buffer Devices	2189	CONFIG_USB_EHCI_HCD_PLATFORM=y
2095		2190	CONFIG_USB_OHCI_HCD=y
2096	#	2191	CONFIG_USB_OHCI_HCD_PCI=y
2097	# Backlight & LCD device support	2192	CONFIG_USB_OHCI_HCD_PLATFORM=y
2098	#	2193	CONFIG_USB_UHCI_HCD=y
2099	CONFIG_BACKLIGHT_CLASS_DEVICE=y	2194	
2100	# end of Backlight & LCD device support	2195	#
2101		2196	# USB Device Class drivers
2102	CONFIG_HDMI=y	2197	#
2103		2198	
2104	#	2199	#
2105	# Console display driver support	2200	# NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD
2106	#		may
2107	CONFIG_VGA_CONSOLE=y	2201	#
2108	CONFIG_DUMMY_CONSOLE=y	2202	
2109	CONFIG_DUMMY_CONSOLE_COLUMNS=80	2203	#
2110	CONFIG_DUMMY_CONSOLE_ROWS=25	2204	# also be needed; see USB_STORAGE Help for more
2111	CONFIG_FRAMEBUFFER_CONSOLE=y		info
2112	CONFIG_FRAMEBUFFER_CONSOLE_DETECT_PRIMARY=y	2205	#
2113	CONFIG_FRAMEBUFFER_CONSOLE_ROTATION=y	2206	
2114	CONFIG_FRAMEBUFFER_CONSOLE_DEFERRED_TAKEOVER=y	2207	#
2115	# end of Console display driver support	2208	# USB Imaging devices
2116		2209	#
2117	# end of Graphics support	2210	CONFIG_USB_DWC2=y
2118		2211	CONFIG_USB_DWC2_HOST=y
2119		2212	
2120	#	2213	#
2121	# HID support	2214	# Gadget/Dual-role mode requires USB Gadget support
2122	#		to be enabled
2123	CONFIG_HID=m	2215	#
2124	CONFIG_HID_BATTERY_STRENGTH=y	2216	
2125	CONFIG_HIDRAW=y	2217	#
2126	CONFIG_HID_GENERIC=m	2218	# USB port drivers
2127		2219	#
2128	#	2220	
2129	# Special HID drivers	2221	#
2130	#	2222	# USB Miscellaneous drivers
2131	# end of Special HID drivers	2223	#
2132		2224	
2133	#	2225	#
2134	# USB HID support	2226	# USB Physical Layer drivers
2135	#	2227	#
2136	CONFIG_USB_HID=m	2228	# end of USB Physical Layer drivers
2137	CONFIG_HID_PID=y	2229	
2138	CONFIG_USB_HIDDEV=y	2230	CONFIG_USB_ROLE_SWITCH=y

2231	CONFIG_MMC=y	2324	#
2232	CONFIG_MMC_CRYPT0=y	2325	CONFIG_DMADEVICES=y
2233		2326	
2234	#	2327	#
2235	# MMC/SD/SDIO Host Controller Drivers	2328	# DMA Devices
2236	#	2329	#
2237	CONFIG_NEW_LEDS=y	2330	CONFIG_DMA_ENGINE=y
2238	CONFIG_LEDS_CLASS=y	2331	CONFIG_DMA_VIRTUAL_CHANNELS=y
2239	CONFIG_LEDS_BRIGHTNESS_HW_CHANGED=y	2332	CONFIG_DMA_ACPI=y
2240		2333	CONFIG_HSU_DMA=y
2241	#	2334	CONFIG_INTEL_LDMA=y
2242	# LED drivers	2335	
2243	#	2336	#
2244		2337	# DMA Clients
2245	#	2338	#
2246	# LED driver for blink(1) USB RGB LED is under	2339	CONFIG_ASYNC_TX_DMA=y
	Special HID drivers (HID_THINGM)	2340	
2247	#	2341	#
2248		2342	# DMABUF options
2249	#	2343	#
2250	# Flash and Torch LED drivers	2344	CONFIG_SYNC_FILE=y
2251	#	2345	CONFIG_SW_SYNC=y
2252		2346	CONFIG_UDMABUF=y
2253	#	2347	CONFIG_DMABUF_HEAPS=y
2254	# LED Triggers	2348	CONFIG_DMABUF_HEAPS_SYSTEM=y
2255	#	2349	# end of DMABUF options
2256	CONFIG_LEDS_TRIGGERS=y	2350	
2257	CONFIG_LEDS_TRIGGER_DISK=y	2351	CONFIG_AUXDISPLAY=y
2258	CONFIG_LEDS_TRIGGER_CPU=y	2352	CONFIG_CHARLCD_BL_FLASH=y
2259		2353	CONFIG_VFIO=y
2260	#	2354	CONFIG_VFIO_IOMMU_TYPE1=y
2261	# iptables trigger is under Netfilter config (LED	2355	CONFIG_VFIO_VIRQFD=y
	target)	2356	CONFIG_VFIO_NOIOMMU=y
2262	#	2357	CONFIG_VFIO_PCI_CORE=y
2263	CONFIG_LEDS_TRIGGER_PANIC=y	2358	CONFIG_VFIO_PCI_MMAP=y
2264	CONFIG_ACCESSIBILITY=y	2359	CONFIG_VFIO_PCI_INTX=y
2265		2360	CONFIG_VFIO_PCI=y
2266	#	2361	CONFIG_VFIO_PCI_VGA=y
2267	# Speakup console speech	2362	CONFIG_VFIO_PCI_IGD=y
2268	#	2363	CONFIG_IRQ_BYPASS_MANAGER=y
2269	# end of Speakup console speech	2364	CONFIG_VIRT_DRIVERS=y
2270		2365	CONFIG_VIRTIO=y
2271	CONFIG_INFINIBAND=m	2366	CONFIG_ARCH_HAS_RESTRICTED_VIRTIO_MEMORY_ACCESS=y
2272	CONFIG_INFINIBAND_USER_ACCESS=m	2367	CONFIG_VIRTIO_PCI_LIB=y
2273	CONFIG_INFINIBAND_USER_MEM=y	2368	CONFIG_VIRTIO_MENU=y
2274	CONFIG_INFINIBAND_ON_DEMAND_PAGING=y	2369	CONFIG_VIRTIO_PCI=y
2275	CONFIG_INFINIBAND_ADDR_TRANS=y	2370	CONFIG_VIRTIO_PCI_LEGACY=y
2276	CONFIG_INFINIBAND_ADDR_TRANS_CONFIGFS=y	2371	CONFIG_VIRTIO_BALLOON=y
2277	CONFIG_INFINIBAND_VIRT_DMA=y	2372	CONFIG_VIRTIO_MMIO=y
2278	CONFIG_INFINIBAND_IRDMA=m	2373	CONFIG_VIRTIO_MMIO_CMDLINE_DEVICES=y
2279	CONFIG_EDAC_ATOMIC_SCRUB=y	2374	CONFIG_VHOST_MENU=y
2280	CONFIG_EDAC_SUPPORT=y	2375	
2281	CONFIG_EDAC=y	2376	#
2282	CONFIG_EDAC_GHES=y	2377	# Microsoft Hyper-V guest support
2283	CONFIG_EDAC_I10NM=m	2378	#
2284	CONFIG_RTC_LIB=y	2379	# end of Microsoft Hyper-V guest support
2285	CONFIG_RTC_MC146818_LIB=y	2380	
2286	CONFIG_RTC_CLASS=y	2381	#
2287	CONFIG_RTC_HCTOSYS=y	2382	# Xen driver support
2288	CONFIG_RTC_HCTOSYS_DEVICE="rtc0"	2383	#
2289	CONFIG_RTC_SYSTOHC=y	2384	CONFIG_XEN_BALLOON=y
2290	CONFIG_RTC_SYSTOHC_DEVICE="rtc0"	2385	CONFIG_XEN_BALLOON_MEMORY_HOTPLUG=y
2291	CONFIG_RTC_NVMEM=y	2386	CONFIG_XEN_MEMORY_HOTPLUG_LIMIT=512
2292		2387	CONFIG_XEN_SCRUB_PAGES_DEFAULT=y
2293	#	2388	CONFIG_XEN_BACKEND=y
2294	# RTC interfaces	2389	CONFIG_XEN_SYS_HYPERVISOR=y
2295	#	2390	CONFIG_XEN_XENBUS_FRONTEND=y
2296	CONFIG_RTC_INTF_SYSFS=y	2391	CONFIG_XEN_GRANT_DMA_ALLOC=y
2297	CONFIG_RTC_INTF_PROC=y	2392	CONFIG_SWIOTLB_XEN=y
2298	CONFIG_RTC_INTF_DEV=y	2393	CONFIG_XEN_PRIVCMD=m
2299		2394	CONFIG_XEN_ACPI_PROCESSOR=y
2300	#	2395	CONFIG_XEN_MCE_LOG=y
2301	# I2C RTC drivers	2396	CONFIG_XEN_HAVE_PVMMU=y
2302	#	2397	CONFIG_XEN_EFI=y
2303		2398	CONFIG_XEN_AUTO_XLATE=y
2304	#	2399	CONFIG_XEN_ACPI=y
2305	# SPI RTC drivers	2400	CONFIG_XEN_HAVE_VPMU=y
2306	#	2401	CONFIG_XEN_UNPOPULATED_ALLOC=y
2307	CONFIG_RTC_I2C_AND_SPI=y	2402	# end of Xen driver support
2308		2403	
2309	#	2404	CONFIG_STAGING=y
2310	# SPI and I2C RTC drivers	2405	CONFIG_STAGING_MEDIA=y
2311	#	2406	
2312		2407	#
2313	#	2408	# Android
2314	# Platform RTC drivers	2409	#
2315	#	2410	# end of Android
2316	CONFIG_RTC_DRV_CMOS=y	2411	
2317		2412	CONFIG_UNISYSSPAR=y
2318	#	2413	CONFIG_X86_PLATFORM_DEVICES=y
2319	# on-CPU RTC drivers	2414	CONFIG_ACPI_WMI=m
2320	#	2415	CONFIG_WMI_BMOF=m
2321		2416	CONFIG_X86_PLATFORM_DRIVERS_DELL=y
2322	#	2417	CONFIG_DCDBAS=m
2323	# HID Sensor RTC drivers	2418	CONFIG_DELL_SMBIOS=m

2419	CONFIG_DELL_SMBIOS_WMI=y	2513	# end of NXP/Freescale QorIQ SoC drivers
2420	CONFIG_DELL_SMBIOS_SMM=y	2514	
2421	CONFIG_DELL_WMI_DESCRIPTOR=m	2515	#
2422	CONFIG_INTEL_PMC_CORE=y	2516	# i.MX SoC drivers
2423		2517	#
2424	#	2518	# end of i.MX SoC drivers
2425	# Intel Speed Select Technology interface support	2519	
2426	#	2520	#
2427	CONFIG_INTEL_SPEED_SELECT_INTERFACE=m	2521	# Enable LiteX SoC Builder specific drivers
2428	# end of Intel Speed Select Technology interface support	2522	#
2429		2523	# end of Enable LiteX SoC Builder specific drivers
2430	CONFIG_INTEL_TURBO_MAX_3=y	2524	
2431	CONFIG_INTEL_SCU_IPC=y	2525	#
2432	CONFIG_INTEL_SCU=y	2526	# Qualcomm SoC drivers
2433	CONFIG_INTEL_SCU_PCI=y	2527	#
2434	CONFIG_PMC_ATOM=y	2528	# end of Qualcomm SoC drivers
2435	CONFIG_CHROME_PLATFORMS=y	2529	
2436	CONFIG_MELLANOX_PLATFORM=y	2530	CONFIG_SOC_TI=y
2437	CONFIG_SURFACE_PLATFORMS=y	2531	
2438	CONFIG_HAVE_CLK=y	2532	#
2439	CONFIG_HAVE_CLK_PREPARE=y	2533	# Xilinx SoC drivers
2440	CONFIG_COMMON_CLK=y	2534	#
2441		2535	# end of Xilinx SoC drivers
2442	#	2536	# end of SOC (System On Chip) specific Drivers
2443	# Clock driver for ARM Reference designs	2537	
2444	#	2538	CONFIG_PM_DEVFREQ=y
2445	CONFIG_ICST=y	2539	
2446	CONFIG_CLK_SP810=y	2540	#
2447	# end of Clock driver for ARM Reference designs	2541	# DEVFREQ Governors
2448		2542	#
2449	CONFIG_HWSPINLOCK=y	2543	CONFIG_DEVFREQ_GOV_SIMPLE_ONDEMAND=y
2450		2544	CONFIG_DEVFREQ_GOV_PERFORMANCE=y
2451	#	2545	CONFIG_DEVFREQ_GOV_POWERSAVE=y
2452	# Clock Source drivers	2546	CONFIG_DEVFREQ_GOV_USERSPACE=y
2453	#	2547	CONFIG_DEVFREQ_GOV_PASSIVE=y
2454	CONFIG_CLKEVT_I8253=y	2548	
2455	CONFIG_I8253_LOCK=y	2549	#
2456	CONFIG_CLKBLD_I8253=y	2550	# DEVFREQ Drivers
2457	# end of Clock Source drivers	2551	#
2458		2552	CONFIG_PM_DEVFREQ_EVENT=y
2459	CONFIG_MAILBOX=y	2553	CONFIG_EXTCON=y
2460	CONFIG_PCC=y	2554	
2461	CONFIG_IOMMU_IOVA=y	2555	#
2462	CONFIG_IOASID=y	2556	# Extcon Device Drivers
2463	CONFIG_IOMMU_API=y	2557	#
2464	CONFIG_IOMMU_SUPPORT=y	2558	CONFIG_MEMORY=y
2465		2559	CONFIG_VME_BUS=y
2466	#	2560	
2467	# Generic IOMMU Pagetable Support	2561	#
2468	#	2562	# VME Bridge Drivers
2469	CONFIG_IOMMU_IO_PGTABLE=y	2563	#
2470	# end of Generic IOMMU Pagetable Support	2564	
2471		2565	#
2472	CONFIG_IOMMU_DEFAULT_DMA_LAZY=y	2566	# VME Board Drivers
2473	CONFIG_IOMMU_DMA=y	2567	#
2474	CONFIG_IOMMU_SVA_LIB=y	2568	
2475	CONFIG_AMD_IOMMU=y	2569	#
2476	CONFIG_DMAR_TABLE=y	2570	# VME Device Drivers
2477	CONFIG_INTEL_IOMMU=y	2571	#
2478	CONFIG_INTEL_IOMMU_SVM=y	2572	CONFIG_PWM=y
2479	CONFIG_INTEL_IOMMU_FLOPPY_WA=y	2573	CONFIG_PWM_SYSFS=y
2480	CONFIG_IRQ_REMAP=y	2574	CONFIG_PWM_CRC=y
2481	CONFIG_VIRTIO_IOMMU=y	2575	CONFIG_PWM_LPSS=y
2482		2576	CONFIG_PWM_LPSS_PCI=y
2483	#	2577	CONFIG_PWM_LPSS_PLATFORM=y
2484	# Remoteproc drivers	2578	
2485	#	2579	#
2486	CONFIG_REMOTEPROC=y	2580	# IRQ chip support
2487	CONFIG_REMOTEPROC_CDEV=y	2581	#
2488	# end of Remoteproc drivers	2582	# end of IRQ chip support
2489		2583	
2490	#	2584	CONFIG_RESET_CONTROLLER=y
2491	# Rpmc drivers	2585	
2492	#	2586	#
2493	# end of Rpmc drivers	2587	# PHY Subsystem
2494		2588	#
2495		2589	CONFIG_GENERIC_PHY=y
2496	#	2590	# end of PHY Subsystem
2497	# SOC (System On Chip) specific Drivers	2591	
2498	#	2592	CONFIG_POWERCAP=y
2499		2593	CONFIG_INTEL_RAPL_CORE=m
2500	#	2594	CONFIG_INTEL_RAPL=m
2501	# Amlogic SoC drivers	2595	CONFIG_IDLE_INJECT=y
2502	#	2596	CONFIG_DTPM=y
2503	# end of Amlogic SoC drivers	2597	CONFIG_DTPM_CPU=y
2504		2598	
2505	#	2599	#
2506	# Broadcom SoC drivers	2600	# Performance monitor support
2507	#	2601	#
2508	# end of Broadcom SoC drivers	2602	# end of Performance monitor support
2509		2603	
2510	#	2604	CONFIG_RAS=y
2511	# NXP/Freescale QorIQ SoC drivers	2605	CONFIG_RAS_CEC=y
2512	#	2606	
		2607	#

2608	# Android	2703	CONFIG_TMPFS_XATTR=y
2609	#	2704	CONFIG_TMPFS_INODE64=y
2610	CONFIG_ANDROID=y	2705	CONFIG_HUGETLBFS=y
2611	# end of Android	2706	CONFIG_HUGETLB_PAGE=y
2612		2707	CONFIG_HUGETLB_PAGE_FREE_VMEMMAP=y
2613	CONFIG_LIBNVDIMM=y	2708	CONFIG_MEMFD_CREATE=y
2614	CONFIG_ND_CLAIM=y	2709	CONFIG_ARCH_HAS_GIGANTIC_PAGE=y
2615	CONFIG_BTT=y	2710	CONFIG_CONFIGFS_FS=y
2616	CONFIG_NVDIMM_PFN=y	2711	CONFIG_EFIVAR_FS=y
2617	CONFIG_NVDIMM_DAX=y	2712	# end of Pseudo filesystems
2618	CONFIG_NVDIMM_KEYS=y	2713	
2619	CONFIG_DAX=y	2714	CONFIG_MISC_FILESYSTEMS=y
2620	CONFIG_NVMEM=y	2715	CONFIG_ECRYPT_FS=y
2621	CONFIG_NVMEM_SYSFS=y	2716	CONFIG_ECRYPT_FS_MESSAGING=y
2622		2717	CONFIG_SQUASHFS=y
2623	#	2718	CONFIG_SQUASHFS_FILE_DIRECT=y
2624	# HW tracing support	2719	CONFIG_SQUASHFS_DECOMP_SINGLE=y
2625	#	2720	CONFIG_SQUASHFS_XATTR=y
2626	# end of HW tracing support	2721	CONFIG_SQUASHFS_ZLIB=y
2627		2722	CONFIG_SQUASHFS_LZ4=y
2628	CONFIG_PM_OPP=y	2723	CONFIG_SQUASHFS_LZ0=y
2629	CONFIG_INTERCONNECT=y	2724	CONFIG_SQUASHFS_XZ=y
2630	# end of Device Drivers	2725	CONFIG_SQUASHFS_ZSTD=y
2631		2726	CONFIG_SQUASHFS_FRAGMENT_CACHE_SIZE=3
2632	#	2727	CONFIG_PSTORE=y
2633	# File systems	2728	CONFIG_PSTORE_DEFAULT_KMSG_BYTES=10240
2634	#	2729	CONFIG_PSTORE_DEFLATE_COMPRESS=y
2635	CONFIG_DCACHE_WORD_ACCESS=y	2730	CONFIG_PSTORE_COMPRESS=y
2636	CONFIG_VALIDATE_FS_PARSER=y	2731	CONFIG_PSTORE_DEFLATE_COMPRESS_DEFAULT=y
2637	CONFIG_FS_IOMAP=y	2732	CONFIG_PSTORE_COMPRESS_DEFAULT="deflate"
2638	CONFIG_EXT4_FS=y	2733	CONFIG_PSTORE_RAM=m
2639	CONFIG_EXT4_USE_FOR_EXT2=y	2734	CONFIG_PSTORE_ZONE=m
2640	CONFIG_EXT4_FS_POSIX_ACL=y	2735	CONFIG_PSTORE_BLK=m
2641	CONFIG_EXT4_FS_SECURITY=y	2736	CONFIG_PSTORE_BLK_BLKDEV=""
2642	CONFIG_JBD2=y	2737	CONFIG_PSTORE_BLK_KMSG_SIZE=64
2643	CONFIG_FS_MBCACHE=y	2738	CONFIG_PSTORE_BLK_MAX_REASON=2
2644	CONFIG_BTRFS_FS=m	2739	CONFIG_NETWORK_FILESYSTEMS=y
2645	CONFIG_BTRFS_FS_POSIX_ACL=y	2740	CONFIG_NLS=y
2646	CONFIG_FS_DAX=y	2741	CONFIG_NLS_DEFAULT="utf8"
2647	CONFIG_FS_DAX_PMD=y	2742	CONFIG_NLS_CODEPAGE_437=y
2648	CONFIG_FS_POSIX_ACL=y	2743	CONFIG_NLS_ISO8859_1=m
2649	CONFIG_EXPORTFS=y	2744	CONFIG_UNICODE=y
2650	CONFIG_EXPORTFS_BLOCK_OPS=y	2745	CONFIG_IO_WQ=y
2651	CONFIG_FILE_LOCKING=y	2746	# end of File systems
2652	CONFIG_FS_ENCRYPTION=y	2747	
2653	CONFIG_FS_ENCRYPTION_ALGS=y	2748	#
2654	CONFIG_FS_ENCRYPTION_INLINE_CRYPT=y	2749	# Security options
2655	CONFIG_FS_VERITY=y	2750	#
2656	CONFIG_FS_VERITY_BUILTIN_SIGNATURES=y	2751	CONFIG_KEYS=y
2657	CONFIG_FSNOTIFY=y	2752	CONFIG_KEYS_REQUEST_CACHE=y
2658	CONFIG_DNOTIFY=y	2753	CONFIG_PERSISTENT_KEYRINGS=y
2659	CONFIG_INOTIFY_USER=y	2754	CONFIG_TRUSTED_KEYS=y
2660	CONFIG_FANOTIFY=y	2755	CONFIG_ENCRYPTED_KEYS=y
2661	CONFIG_FANOTIFY_ACCESS_PERMISSIONS=y	2756	CONFIG_KEY_DH_OPERATIONS=y
2662	CONFIG_QUOTA=y	2757	CONFIG_KEY_NOTIFICATIONS=y
2663	CONFIG_QUOTA_NETLINK_INTERFACE=y	2758	CONFIG_SECURITY_DMESG_RESTRICT=y
2664	CONFIG_QUOTACTL=y	2759	CONFIG_SECURITY=y
2665	CONFIG_AUTOFS_FS=m	2760	CONFIG_SECURITYFS=y
2666	CONFIG_FUSE_FS=y	2761	CONFIG_SECURITY_NETWORK=y
2667		2762	CONFIG_SECURITY_INFINIBAND=y
2668	#	2763	CONFIG_SECURITY_PATH=y
2669	# Caches	2764	CONFIG_INTEL_TXT=y
2670	#	2765	CONFIG_LSM_MMAP_MIN_ADDR=0
2671	# end of Caches	2766	CONFIG_HAVE_HARDENED_USERCOPY_ALLOCATOR=y
2672		2767	CONFIG_HARDENED_USERCOPY=y
2673	#	2768	CONFIG_FORTIFY_SOURCE=y
2674	# CD-ROM/DVD Filesystems	2769	CONFIG_SECURITY_SELINUX=y
2675	#	2770	CONFIG_SECURITY_SELINUX_BOOTPARAM=y
2676	# end of CD-ROM/DVD Filesystems	2771	CONFIG_SECURITY_SELINUX_DEVELOP=y
2677		2772	CONFIG_SECURITY_SELINUX_AVC_STATS=y
2678	#	2773	CONFIG_SECURITY_SELINUX_CHECKREQPROT_VALUE=1
2679	# DOS/FAT/EXFAT/NT Filesystems	2774	CONFIG_SECURITY_SELINUX_SIDTAB_HASH_BITS=9
2680	#	2775	CONFIG_SECURITY_SELINUX_SID2STR_CACHE_SIZE=256
2681	CONFIG_FAT_FS=y	2776	CONFIG_SECURITY_SMACK=y
2682	CONFIG_VFAT_FS=y	2777	CONFIG_SECURITY_SMACK_NETFILTER=y
2683	CONFIG_FAT_DEFAULT_CODEPAGE=437	2778	CONFIG_SECURITY_SMACK_APPEND_SIGNALS=y
2684	CONFIG_FAT_DEFAULT_IOCHARSET="iso8859-1"	2779	CONFIG_SECURITY_TOMOYO=y
2685	# end of DOS/FAT/EXFAT/NT Filesystems	2780	CONFIG_SECURITY_TOMOYO_MAX_ACCEPT_ENTRY=2048
2686		2781	CONFIG_SECURITY_TOMOYO_MAX_AUDIT_LOG=1024
2687	#	2782	CONFIG_SECURITY_TOMOYO_POLICY_LOADER="/sbin/tomoyo-init"
2688	# Pseudo filesystems	2783	CONFIG_SECURITY_TOMOYO_ACTIVATION_TRIGGER="/sbin/init"
2689	#	2784	CONFIG_SECURITY_APPARMOR=y
2690	CONFIG_PROC_FS=y	2785	CONFIG_SECURITY_APPARMOR_HASH=y
2691	CONFIG_PROC_KCORE=y	2786	CONFIG_SECURITY_APPARMOR_HASH_DEFAULT=y
2692	CONFIG_PROC_VMCORE=y	2787	CONFIG_SECURITY_YAMA=y
2693	CONFIG_PROC_VMCORE_DEVICE_DUMP=y	2788	CONFIG_SECURITY_SAFESETID=y
2694	CONFIG_PROC_SYSCTL=y	2789	CONFIG_SECURITY_LOCKDOWN_LSM=y
2695	CONFIG_PROC_PAGE_MONITOR=y	2790	CONFIG_SECURITY_LOCKDOWN_LSM_EARLY=y
2696	CONFIG_PROC_CHILDREN=y	2791	CONFIG_LOCK_DOWN_KERNEL_FORCE_NONE=y
2697	CONFIG_PROC_PID_ARCH_STATUS=y	2792	CONFIG_SECURITY_LANDLOCK=y
2698	CONFIG_PROC_CPU_RESCTRL=y	2793	CONFIG_INTEGRITY=y
2699	CONFIG_KERNFS=y	2794	CONFIG_INTEGRITY_SIGNATURE=y
2700	CONFIG_SYSFS=y	2795	CONFIG_INTEGRITY_ASYMMETRIC_KEYS=y
2701	CONFIG_TMPFS=y		
2702	CONFIG_TMPFS_POSIX_ACL=y		

```

2796 CONFIG_INTEGRITY_TRUSTED_KEYRING=y
2797 CONFIG_INTEGRITY_PLATFORM_KEYRING=y
2798 CONFIG_LOAD_UEFI_KEYS=y
2799 CONFIG_INTEGRITY_AUDIT=y
2800 CONFIG_IMA=y
2801 CONFIG_IMA_MEASURE_PCR_IDX=10
2802 CONFIG_IMA_LSM_RULES=y
2803 CONFIG_IMA_NG_TEMPLATE=y
2804 CONFIG_IMA_DEFAULT_TEMPLATE="ima-ng"
2805 CONFIG_IMA_DEFAULT_HASH_SHA1=y
2806 CONFIG_IMA_DEFAULT_HASH="sha1"
2807 CONFIG_IMA_APPRAISE=y
2808 CONFIG_IMA_APPRAISE_BOOTPARAM=y
2809 CONFIG_IMA_APPRAISE_MODSIG=y
2810 CONFIG_IMA_TRUSTED_KEYRING=y
2811 CONFIG_IMA_MEASURE_ASYMMETRIC_KEYS=y
2812 CONFIG_IMA_QUEUE_EARLY_BOOT_KEYS=y
2813 CONFIG_EVM=y
2814 CONFIG_EVM_ATTR_FSUID=y
2815 CONFIG_EVM_EXTRA_SMACK_XATTRS=y
2816 CONFIG_EVM_ADD_XATTRS=y
2817 CONFIG_DEFAULT_SECURITY_APPARMOR=y
2818 CONFIG_LSM="landlock,lockdown,yama,integrity,
    apparmor"
2819
2820 #
2821 # Kernel hardening options
2822 #
2823
2824 #
2825 # Memory initialization
2826 #
2827 CONFIG_INIT_STACK_NONE=y
2828 CONFIG_INIT_ON_ALLOC_DEFAULT_ON=y
2829 CONFIG_CC_HAS_ZERO_CALL_USED_REGS=y
2830 # end of Memory initialization
2831 # end of Kernel hardening options
2832 # end of Security options
2833
2834 CONFIG_XOR_BLOCKS=m
2835 CONFIG_ASYNC_CORE=m
2836 CONFIG_ASYNC_MEMCPY=m
2837 CONFIG_ASYNC_XOR=m
2838 CONFIG_ASYNC_PQ=m
2839 CONFIG_ASYNC_RAID6_RECOV=m
2840 CONFIG_CRYPTD=y
2841
2842 #
2843 # Crypto core or helper
2844 #
2845 CONFIG_CRYPTD_ALGAPI=y
2846 CONFIG_CRYPTD_ALGAPI2=y
2847 CONFIG_CRYPTD_AEAD=y
2848 CONFIG_CRYPTD_AEAD2=y
2849 CONFIG_CRYPTD_SKCIPHER=y
2850 CONFIG_CRYPTD_SKCIPHER2=y
2851 CONFIG_CRYPTD_HASH=y
2852 CONFIG_CRYPTD_HASH2=y
2853 CONFIG_CRYPTD_RNG=y
2854 CONFIG_CRYPTD_RNG2=y
2855 CONFIG_CRYPTD_RNG_DEFAULT=y
2856 CONFIG_CRYPTD_AKCIPHER2=y
2857 CONFIG_CRYPTD_AKCIPHER=y
2858 CONFIG_CRYPTD_KPP2=y
2859 CONFIG_CRYPTD_KPP=y
2860 CONFIG_CRYPTD_ACOMP2=y
2861 CONFIG_CRYPTD_MANAGER=y
2862 CONFIG_CRYPTD_MANAGER2=y
2863 CONFIG_CRYPTD_MANAGER_DISABLE_TESTS=y
2864 CONFIG_CRYPTD_GF128MUL=y
2865 CONFIG_CRYPTD_NULL=y
2866 CONFIG_CRYPTD_NULL2=y
2867 CONFIG_CRYPTD_CRYPTD=m
2868 CONFIG_CRYPTD_SIMD=m
2869
2870 #
2871 # Public-key cryptography
2872 #
2873 CONFIG_CRYPTD_RSA=y
2874 CONFIG_CRYPTD_DH=y
2875
2876 #
2877 # Authenticated Encryption with Associated Data
2878 #
2879 CONFIG_CRYPTD_GCM=y
2880 CONFIG_CRYPTD_SEQIV=y
2881
2882 #
2883 # Block modes
2884 #
2885 CONFIG_CRYPTD_CBC=y
2886 CONFIG_CRYPTD_CTR=y
2887 CONFIG_CRYPTD_CTS=y
2888 CONFIG_CRYPTD_ECB=y
2889 CONFIG_CRYPTD_XTS=y
2890
2891 #
2892 # Hash modes
2893 #
2894 CONFIG_CRYPTD_HMAC=y
2895
2896 #
2897 # Digest
2898 #
2899 CONFIG_CRYPTD_CRC32C=y
2900 CONFIG_CRYPTD_CRC32C_INTEL=y
2901 CONFIG_CRYPTD_CRC32_PCLMUL=m
2902 CONFIG_CRYPTD_XHASH=m
2903 CONFIG_CRYPTD_BLAKE2B=m
2904 CONFIG_CRYPTD_BLAKE2S_X86=y
2905 CONFIG_CRYPTD_CRCT10DIF=y
2906 CONFIG_CRYPTD_CRCT10DIF_PCLMUL=m
2907 CONFIG_CRYPTD_GHASH=y
2908 CONFIG_CRYPTD_MD5=y
2909 CONFIG_CRYPTD_SHA1=y
2910 CONFIG_CRYPTD_SHA256=y
2911 CONFIG_CRYPTD_SHA512=y
2912 CONFIG_CRYPTD_GHASH_CLMUL_NI_INTEL=m
2913
2914 #
2915 # Ciphers
2916 #
2917 CONFIG_CRYPTD_AES=y
2918 CONFIG_CRYPTD_AES_NI_INTEL=m
2919
2920 #
2921 # Compression
2922 #
2923 CONFIG_CRYPTD_DEFLATE=y
2924 CONFIG_CRYPTD_LZO=y
2925
2926 #
2927 # Random Number Generation
2928 #
2929 CONFIG_CRYPTD_DRBG_MENU=y
2930 CONFIG_CRYPTD_DRBG_HMAC=y
2931 CONFIG_CRYPTD_DRBG_HASH=y
2932 CONFIG_CRYPTD_DRBG_CTR=y
2933 CONFIG_CRYPTD_DRBG=y
2934 CONFIG_CRYPTD_JITTERENTROPY=y
2935 CONFIG_CRYPTD_HASH_INFO=y
2936 CONFIG_CRYPTD_HW=y
2937 CONFIG_CRYPTD_DEV_PADLOCK=y
2938 CONFIG_CRYPTD_DEV_CCP=y
2939 CONFIG_CRYPTD_DEV_KEYRING=y
2940 CONFIG_CRYPTD_PUBLIC_KEY_SUBTYPE=y
2941 CONFIG_X509_CERTIFICATE_PARSER=y
2942 CONFIG_PKCS7_MESSAGE_PARSER=y
2943 CONFIG_SIGNED_PE_FILE_VERIFICATION=y
2944
2945 #
2946 # Certificates for signature checking
2947 #
2948 CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
2949 CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
2950 CONFIG_SYSTEM_TRUSTED_KEYRING=y
2951 CONFIG_SYSTEM_TRUSTED_KEYS=""
2952 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
2953 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
2954 CONFIG_SECONDARY_TRUSTED_KEYRING=y
2955 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
2956 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
2957 CONFIG_SYSTEM_REVOCATION_LIST=y
2958 CONFIG_SYSTEM_REVOCATION_KEYS=""
2959 # end of Certificates for signature checking
2960
2961 CONFIG_BINARY_PRINTF=y
2962
2963 #
2964 # Library routines
2965 #
2966 CONFIG_RAID6_PQ=m
2967 CONFIG_RAID6_PQ_BENCHMARK=y
2968 CONFIG_LINEAR_RANGES=y
2969 CONFIG_PACKING=y
2970 CONFIG_BITREVERSE=y
2971 CONFIG_GENERIC_STRNCPY_FROM_USER=y
2972 CONFIG_GENERIC_STRNLEN_USER=y
2973 CONFIG_GENERIC_NET_UTILS=y
2974 CONFIG_GENERIC_FIND_FIRST_BIT=y
2975 CONFIG_RATIONAL=y
2976 CONFIG_GENERIC_PCI_IOMAP=y
2977 CONFIG_GENERIC_IOMAP=y
2978 CONFIG_ARCH_USE_CMPXCHG_LOCKREF=y
2979 CONFIG_ARCH_HAS_FAST_MULTIPLIER=y
2980 CONFIG_ARCH_USE_SYM_ANNOTATIONS=y
2981
2982 #
2983 # Crypto library routines
2984 #

```

```

2985 CONFIG_CRYPTOLIB_AES=y
2986 CONFIG_CRYPTOLIB_ARCH_HAVE_LIB_BLAKE2S=y
2987 CONFIG_CRYPTOLIB_BLAKE2S_GENERIC=y
2988 CONFIG_CRYPTOLIB_POLY1305_RSIZE=11
2989 CONFIG_CRYPTOLIB_SHA256=y
2990 # end of Crypto library routines
2991
2992 CONFIG_LIB_MEMNEQ=y
2993 CONFIG_CRC_CCITT=y
2994 CONFIG_CRC16=y
2995 CONFIG_CRC_T10DIF=y
2996 CONFIG_CRC32=y
2997 CONFIG_CRC32_SLICEBY8=y
2998 CONFIG_LIBCRC32C=y
2999 CONFIG_XHASH=y
3000 CONFIG_ZLIB_INFLATE=y
3001 CONFIG_ZLIB_DEFLATE=y
3002 CONFIG_LZO_COMPRESS=y
3003 CONFIG_LZO_DECOMPRESS=y
3004 CONFIG_LZ4_DECOMPRESS=y
3005 CONFIG_ZSTD_COMPRESS=m
3006 CONFIG_ZSTD_DECOMPRESS=y
3007 CONFIG_XZ_DEC=y
3008 CONFIG_XZ_DEC_X86=y
3009 CONFIG_XZ_DEC_POWERPC=y
3010 CONFIG_XZ_DEC_IA64=y
3011 CONFIG_XZ_DEC_ARM=y
3012 CONFIG_XZ_DEC_ARMTHUMB=y
3013 CONFIG_XZ_DEC_SPARC=y
3014 CONFIG_XZ_DEC_BCJ=y
3015 CONFIG_DECOMPRESS_GZIP=y
3016 CONFIG_DECOMPRESS_BZIP2=y
3017 CONFIG_DECOMPRESS_LZMA=y
3018 CONFIG_DECOMPRESS_XZ=y
3019 CONFIG_DECOMPRESS_LZO=y
3020 CONFIG_DECOMPRESS_LZ4=y
3021 CONFIG_DECOMPRESS_ZSTD=y
3022 CONFIG_GENERIC_ALLOCATOR=y
3023 CONFIG_REED_SOLOMON=m
3024 CONFIG_REED_SOLOMON_ENC8=y
3025 CONFIG_REED_SOLOMON_DEC8=y
3026 CONFIG_INTERVAL_TREE=y
3027 CONFIG_XARRAY_MULTI=y
3028 CONFIG_ASSOCIATIVE_ARRAY=y
3029 CONFIG_HAS_IOMEM=y
3030 CONFIG_HAS_IOPORT_MAP=y
3031 CONFIG_HAS_DMA=y
3032 CONFIG_DMA_OPS=y
3033 CONFIG_NEED_SG_DMA_LENGTH=y
3034 CONFIG_NEED_DMA_MAP_STATE=y
3035 CONFIG_ARCH_DMA_ADDR_T_64BIT=y
3036 CONFIG_ARCH_HAS_FORCE_DMA_UNENCRYPTED=y
3037 CONFIG_SWIOTLB=y
3038 CONFIG_DMA_COHERENT_POOL=y
3039 CONFIG_SGL_ALLOC=y
3040 CONFIG_IOMMU_HELPER=y
3041 CONFIG_CHECK_SIGNATURE=y
3042 CONFIG_CPU_MASK_OFFSTACK=y
3043 CONFIG_CPU_RMAP=y
3044 CONFIG_DQL=y
3045 CONFIG_GLOB=y
3046 CONFIG_NLATTR=y
3047 CONFIG_CLZ_TAB=y
3048 CONFIG_IRQ_POLL=y
3049 CONFIG_MPILIB=y
3050 CONFIG_SIGNATURE=y
3051 CONFIG_DIMLIB=y
3052 CONFIG_OID_REGISTRY=y
3053 CONFIG_UCS2_STRING=y
3054 CONFIG_HAVE_GENERIC_VDSO=y
3055 CONFIG_GENERIC_GETTIMEOFDAY=y
3056 CONFIG_GENERIC_VDSO_TIME_NS=y
3057 CONFIG_FONT_SUPPORT=y
3058 CONFIG_FONTS=y
3059 CONFIG_FONT_8x8=y
3060 CONFIG_FONT_8x16=y
3061 CONFIG_FONT_ACORN_8x8=y
3062 CONFIG_FONT_6x10=y
3063 CONFIG_FONT_TER16x32=y
3064 CONFIG_SG_POOL=y
3065 CONFIG_ARCH_HAS_PMEM_API=y
3066 CONFIG_MEMREGION=y
3067 CONFIG_ARCH_HAS_UACCESS_FLUSHCACHE=y
3068 CONFIG_ARCH_HAS_COPY_MC=y
3069 CONFIG_ARCH_STACKWALK=y
3070 CONFIG_SBITMAP=y
3071 # end of Library routines
3072
3073 CONFIG_PLDMFW=y
3074 CONFIG_ASN1_ENCODER=y
3075
3076 #
3077 # Kernel hacking
3078 #
3079
3080 #
3081 # printk and dmesg options
3082 #
3083 CONFIG_PRINTK_TIME=y
3084 CONFIG_CONSOLE_LOGLEVEL_DEFAULT=7
3085 CONFIG_CONSOLE_LOGLEVEL_QUIET=4
3086 CONFIG_MESSAGE_LOGLEVEL_DEFAULT=4
3087 CONFIG_BOOT_PRINTK_DELAY=y
3088 CONFIG_DYNAMIC_DEBUG=y
3089 CONFIG_DYNAMIC_DEBUG_CORE=y
3090 CONFIG_SYMBOLIC_ERRNAME=y
3091 CONFIG_DEBUG_BUGVERBOSE=y
3092 # end of printk and dmesg options
3093
3094 CONFIG_AS_HAS_NON_CONST_LEB128=y
3095
3096 #
3097 # Compile-time checks and compiler options
3098 #
3099 CONFIG_DEBUG_INFO=y
3100 CONFIG_DEBUG_INFO_DWARF_TOOLCHAIN_DEFAULT=y
3101 CONFIG_GDB_SCRIPTS=y
3102 CONFIG_FRAME_WARN=1024
3103 CONFIG_SECTION_MISMATCH_WARN_ONLY=y
3104 CONFIG_FRAME_POINTER=y
3105 CONFIG_STACK_VALIDATION=y
3106 CONFIG_VMLINUX_MAP=y
3107 # end of Compile-time checks and compiler options
3108
3109 #
3110 # Generic Kernel Debugging Instruments
3111 #
3112 CONFIG_MAGIC_SYSRQ=y
3113 CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE=0x01b6
3114 CONFIG_MAGIC_SYSRQ_SERIAL=y
3115 CONFIG_MAGIC_SYSRQ_SERIAL_SEQUENCE=""
3116 CONFIG_DEBUG_FS=y
3117 CONFIG_DEBUG_FS_ALLOW_ALL=y
3118 CONFIG_HAVE_ARCH_KGDB=y
3119 CONFIG_KGDB=y
3120 CONFIG_KGDB_HONOUR_BLOCKLIST=y
3121 CONFIG_KGDB_SERIAL_CONSOLE=y
3122 CONFIG_KGDB_LOW_LEVEL_TRAP=y
3123 CONFIG_KGDB_KDB=y
3124 CONFIG_KDB_DEFAULT_ENABLE=0x1
3125 CONFIG_KDB_KEYBOARD=y
3126 CONFIG_KDB_CONTINUE_CATASTROPHIC=0
3127 CONFIG_ARCH_HAS_EARLY_DEBUG=y
3128 CONFIG_ARCH_HAS_UBSAN_SANITIZE_ALL=y
3129 CONFIG_UBSAN=y
3130 CONFIG_CC_HAS_UBSAN_BOUNDS=y
3131 CONFIG_UBSAN_BOUNDS=y
3132 CONFIG_UBSAN_ONLY_BOUNDS=y
3133 CONFIG_UBSAN_SHIFT=y
3134 CONFIG_UBSAN_BOOL=y
3135 CONFIG_UBSAN_ENUM=y
3136 CONFIG_UBSAN_SANITIZE_ALL=y
3137 CONFIG_HAVE_ARCH_KCSAN=y
3138 CONFIG_HAVE_KCSAN_COMPILER=y
3139 # end of Generic Kernel Debugging Instruments
3140
3141 CONFIG_DEBUG_KERNEL=y
3142 CONFIG_DEBUG_MISC=y
3143
3144 #
3145 # Memory Debugging
3146 #
3147 CONFIG_PAGE_POISONING=y
3148 CONFIG_ARCH_HAS_DEBUG_WX=y
3149 CONFIG_DEBUG_WX=y
3150 CONFIG_GENERIC_PTDUMP=y
3151 CONFIG_PTDUMP_CORE=y
3152 CONFIG_HAVE_DEBUG_KMEMLEAK=y
3153 CONFIG_SCHED_STACK_END_CHECK=y
3154 CONFIG_ARCH_HAS_DEBUG_VM_PGTABLE=y
3155 CONFIG_ARCH_HAS_DEBUG_VIRTUAL=y
3156 CONFIG_HAVE_ARCH_KASAN=y
3157 CONFIG_HAVE_ARCH_KASAN_VMALLOC=y
3158 CONFIG_CC_HAS_KASAN_GENERIC=y
3159 CONFIG_CC_HAS_WORKING_NOSANITIZE_ADDRESS=y
3160 CONFIG_HAVE_ARCH_KFENCE=y
3161 CONFIG_KFENCE=y
3162 CONFIG_KFENCE_SAMPLE_INTERVAL=0
3163 CONFIG_KFENCE_NUM_OBJECTS=255
3164 CONFIG_KFENCE_STRESS_TEST_FAULTS=0
3165 # end of Memory Debugging
3166
3167 #
3168 # Debug Oops, Lockups and Hangs
3169 #
3170 CONFIG_PANIC_ON_OOPS_VALUE=0
3171 CONFIG_PANIC_TIMEOUT=0
3172 CONFIG_LOCKUP_DETECTOR=y
3173 CONFIG_SOFTLOCKUP_DETECTOR=y
3174

```

3175	CONFIG_BOOTPARAM_SOFTLOCKUP_PANIC_VALUE=0	3234	CONFIG_TRACING_SUPPORT=y
3176	CONFIG_HARDLOCKUP_DETECTOR_PERF=y	3235	CONFIG_FTRACE=y
3177	CONFIG_HARDLOCKUP_CHECK_TIMESTAMP=y	3236	CONFIG_BOOTTIME_TRACING=y
3178	CONFIG_HARDLOCKUP_DETECTOR=y	3237	CONFIG_FUNCTION_TRACER=y
3179	CONFIG_BOOTPARAM_HARDLOCKUP_PANIC_VALUE=0	3238	CONFIG_FUNCTION_GRAPH_TRACER=y
3180	CONFIG_DETECT_HUNG_TASK=y	3239	CONFIG_DYNAMIC_FTRACE=y
3181	CONFIG_DEFAULT_HUNG_TASK_TIMEOUT=120	3240	CONFIG_DYNAMIC_FTRACE_WITH_REGS=y
3182	CONFIG_BOOTPARAM_HUNG_TASK_PANIC_VALUE=0	3241	CONFIG_DYNAMIC_FTRACE_WITH_DIRECT_CALLS=y
3183	# end of Debug Oops, Lockups and Hangs	3242	CONFIG_DYNAMIC_FTRACE_WITH_ARGS=y
3184		3243	CONFIG_FUNCTION_PROFILER=y
3185	#	3244	CONFIG_STACK_TRACER=y
3186	# Scheduler Debugging	3245	CONFIG_SCHED_TRACER=y
3187	#	3246	CONFIG_HWLAT_TRACER=y
3188	CONFIG_SCHED_DEBUG=y	3247	CONFIG_MMIOTRACE=y
3189	CONFIG_SCHED_INFO=y	3248	CONFIG_FTRACE_SYSCALLS=y
3190	CONFIG_SCHEDSTATS=y	3249	CONFIG_TRACER_SNAPSHOT=y
3191	# end of Scheduler Debugging	3250	CONFIG_BRANCH_PROFILE_NONE=y
3192		3251	CONFIG_BLK_DEV_IO_TRACE=y
3193		3252	CONFIG_KPROBE_EVENTS=y
3194	#	3253	CONFIG_UPROBE_EVENTS=y
3195	# Lock Debugging (spinlocks, mutexes, etc...)	3254	CONFIG_BPF_EVENTS=y
3196	#	3255	CONFIG_DYNAMIC_EVENTS=y
3197	CONFIG_LOCK_DEBUGGING_SUPPORT=y	3256	CONFIG_PROBE_EVENTS=y
3198	# end of Lock Debugging (spinlocks, mutexes, etc ...)	3257	CONFIG_BPF_KPROBE_OVERRIDE=y
3199		3258	CONFIG_FTRACE_MCOUNT_RECORD=y
3200	CONFIG_STACKTRACE=y	3259	CONFIG_FTRACE_MCOUNT_USE_CC=y
3201		3260	CONFIG_TRACING_MAP=y
3202	#	3261	CONFIG_SYNTH_EVENTS=y
3203	# Debug kernel data structures	3262	CONFIG_HIST_TRIGGERS=y
3204	#	3263	CONFIG_TRACE_EVENT_INJECT=y
3205	# end of Debug kernel data structures	3264	CONFIG_SAMPLES=y
3206		3265	CONFIG_ARCH_HAS_DEVMEM_IS_ALLOWED=y
3207		3266	CONFIG_STRICT_DEVMEM=y
3208	#	3267	
3209	# RCU Debugging	3268	#
3210	#	3269	# x86 Debugging
3211	CONFIG_RCU_CPU_STALL_TIMEOUT=60	3270	#
3212	# end of RCU Debugging	3271	CONFIG_EARLY_PRINTK_USB=y
3213		3272	CONFIG_EARLY_PRINTK=y
3214	CONFIG_USER_STACKTRACE_SUPPORT=y	3273	CONFIG_EARLY_PRINTKDBG=y
3215	CONFIG_NOP_TRACER=y	3274	CONFIG_EARLY_PRINTK_USB_XDBC=y
3216	CONFIG_HAVE_FUNCTION_TRACER=y	3275	CONFIG_HAVE_MMIOTRACE_SUPPORT=y
3217	CONFIG_HAVE_FUNCTION_GRAPH_TRACER=y	3276	CONFIG_IO_DELAY_OXED=y
3218	CONFIG_HAVE_DYNAMIC_FTRACE=y	3277	CONFIG_X86_DEBUG_FPU=y
3219	CONFIG_HAVE_DYNAMIC_FTRACE_WITH_REGS=y	3278	CONFIG_UNWINDER_FRAME_POINTER=y
3220	CONFIG_HAVE_DYNAMIC_FTRACE_WITH_DIRECT_CALLS=y	3279	# end of x86 Debugging
3221	CONFIG_HAVE_DYNAMIC_FTRACE_WITH_ARGS=y	3280	
3222	CONFIG_HAVE_FTRACE_MCOUNT_RECORD=y	3281	#
3223	CONFIG_HAVE_SYSCALL_TRACEPOINTS=y	3282	# Kernel Testing and Coverage
3224	CONFIG_HAVE_FENTRY=y	3283	#
3225	CONFIG_HAVE_OBJTOOL_MCOUNT=y	3284	CONFIG_FUNCTION_ERROR_INJECTION=y
3226	CONFIG_HAVE_C_RECORDMCOUNT=y	3285	CONFIG_ARCH_HAS_KCOV=y
3227	CONFIG_TRACER_MAX_TRACE=y	3286	CONFIG_CC_HAS_SANCOV_TRACE_PC=y
3228	CONFIG_TRACE_CLOCK=y	3287	CONFIG_RUNTIME_TESTING_MENU=y
3229	CONFIG_RING_BUFFER=y	3288	CONFIG_ARCH_USE_MEMTEST=y
3230	CONFIG_EVENT_TRACING=y	3289	CONFIG_MEMTEST=y
3231	CONFIG_CONTEXT_SWITCH_TRACER=y	3290	# end of Kernel Testing and Coverage
3232	CONFIG_TRACING=y	3291	# end of Kernel hacking
3233	CONFIG_GENERIC_TRACER=y		