

Sharding the object plane: topic bits turn one firehose into network traffic an operator can shape

The BEEF object plane carries application traffic: every overlay topic in existence can ride one multicast plane, and the topic space is a SHA-256 key space, effectively unbounded, with no registry and nothing to provision. This paper explains why such a plane is sharded at all, since an application only ever wants a sliver of it. It derives the group from the TopicID with the same top-bits rule the settlement plane uses, then shows what the resulting groups are for: they are the only unit of this traffic that the network's forwarding plane can see, join, route, bound and meter, which makes shard width the operator's traffic-shaping handle. It states what grows with what, and what bounds it: no component holds per-topic state, widening never needs a flag day, and the number of groups is set by a join budget, not by the address plan, so the plane stays narrow on purpose. It then follows the topic itself one step further, to topics named by transactions, scripts and derived keys, which turn the plane into a global addressing system by subscription. It places the design against the Internet's traditional distribution model, and it closes with how operators peer the plane, at whichever layer suits them.

Author: Jeff Harris · Lightweb Inc. · jeff@lightweb.net

Paper 010 · Version 02

Audience: multicast fabric operators, network engineers at operators and exchanges who may peer with one, protocol implementers, and application architects who want to know what a topic can be.

We = the 1BSV fabric; **you** = an operator, implementer or architect. Widths, address zones and frame fields are the published specifications; traffic figures are stated models, labeled as such.

The short version

- **The object plane carries application traffic, and applications consume selectively.** Every application's overlay objects can ride one plane, and each application wants its own topics and nothing else. That is the opposite of the settlement plane, whose consumers are miners and want everything, so the two planes shard by the same rule for different reasons: the settlement plane for parallelism, the object plane so that bytes flow only where someone elected a topic.
- **A topic's group is the leading bits of its TopicID.** `0x1000 + (SHA-256(name)[0:4] as uint32 BE) >> (32 - bits)`: no registry, no lookup, no state, and every component agrees without asking.
- **Groups are what the network can shape; topics are not.** Routers join, prune, bound, police and count multicast groups. They never see a topic name. Shard width is therefore the granularity of every operator handle on this traffic, and widening is a split that no subscription ever notices.
- **Width is a dial with a known exchange rate and a known bill.** At one million equally active topics, a service-side filter that wants ten of them ingests the whole plane at width 0 and about 3,900 times its elected bytes at width 8; each added bit halves the carriage of unwanted traffic. Each added bit also doubles every receiver's joins, and that is what sets the width.
- **The plane stays narrow by design.** A receiver's join state is at most bands × regions covered, because each band has one anycast source per region. A join budget of a few hundred (S,G) entries per receiver puts the operating range at tens to a few hundred groups; the address plan's room for 32,768 is reserve, not a plan.
- **Nothing in the network scales with the topic count.** Forwarding state is groups × sources; per-topic state lives only in the service-side filters, and only for topics their own subscribers elected. BGP carries source prefixes, never topics.
- **A topic is an address the receiver chooses.** A name can be a transaction, an outpoint, a locking script, a key, or a secret two parties derive; subscription by hash keeps the name private; multi-party rendezvous and identity binding follow without a registry.
- **This is a different distribution model, and a more securable one.** Receivers pull named sources down source-rooted trees; the sources are routed prefixes the routing system already validates; the objects prove themselves at every receiver.
- **Operators peer the plane at whichever layer suits them.** At the network layer with PIM-SSM and MP-BGP over a private interconnect, at the tunnel layer with GRE where native multicast could be absent at the border, or at the application layer as a transit-level consumer that takes every plane over a pair of direct connections. The same frames, records and repair protocol work at all three.

1. One plane, an unbounded key space, and selective consumers

1.1 What the plane carries

The object plane is the application plane of the network. An overlay topic is named by a string and identified on the wire by the SHA-256 of that string, its TopicID. Any name is a valid topic the moment someone publishes to it; there is no registry to apply to and nothing in the network that has to learn the name first. The topic space is therefore a 2^{256} key space, and the number of live topics is bounded only by the number of applications in the world. Overlay topics work over unicast today, and many will never reach the plane. Any of them can ride it: every BEEF object published to a topic on the plane travels verbatim, with its proofs intact, on one plane alongside the settlement plane and never mixed with it.

1.2 Who consumes it, and how little of it

An application consumes selectively. An exchange wants the topics of the tokens it lists. A game wants its own topic. A wallet backend wants the handful of topics its users touch. An aggregator that wants the whole plane exists, but it is the exception, and it is priced as one. For almost every consumer the plane's total rate is irrelevant; what matters is the rate of the topics it elected, and that is a sliver.

The settlement plane's consumers are the opposite: miners, and a miner wants everything. Sharding there exists to parallelize a stream every consumer drinks whole, so that no single receiver, link or cache carries all of it, and every miner joins every group. The same one-line rule serves both planes, for different reasons.

Between the plane and an application sits a **delivery edge**: the service-side component that terminates the plane for its subscribers and hands each one only the topics it elected. The term is ours; the function is a service-side filter, an optimization that nothing in the network depends on. This is the tension the rest of the paper resolves. The network replicates per **group**: a join builds a tree, a tree carries a group, and a router forwards or prunes whole groups. The application elects per **topic**. Everything between those two granularities is either carried for nothing or filtered by software at the delivery edge, and the distance between them is what shard width sets.

1.3 The one-group posture, stated at its strongest

Width 0 puts the whole plane in one group. If the argument against it were weak, it would be the right answer, and for a young plane it is the right answer. One group means no generation transitions ever, the minimum possible join and forwarding state, and a single filter at every delivery edge that cuts the stream to each subscriber's topics. The published specification calls it a deliberate operating point rather than a degenerate one, and says when to leave it: widen when per-group delivery capacity, not topic count, becomes the binding constraint.

What the one group costs is carriage. Every delivery edge, every interconnect and every peering link carries the whole plane whether or not anyone behind it elected anything, and the service-side filter discards most of what arrives. The rest of this paper is about spending bits against that cost, one at a time, and what an operator gains with each one.

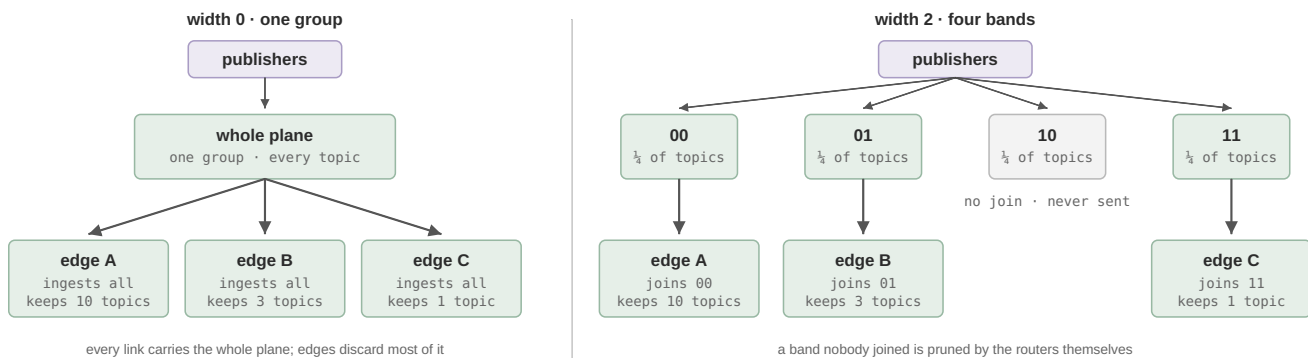


Figure 1 - the same three delivery edges at width 0 and at width 2: with bands, an edge receives only the bands its elected topics hash into, and the network does the first cut of the filtering by not forwarding what nobody joined.

2. The derivation: a topic's group is the front of its hash

The companion paper *Shard bit extraction* walks the arithmetic bit by bit for the settlement plane. The object plane applies the same rule to a different key and adds a plane base:

TopicID	= SHA-256(UTF-8 topic name)	· 32 bytes
shardIndex	= (TopicID[0:4] as uint32 BE) >> (32 - bits)	· 0 at width 0
IDX	= 0x1000 + shardIndex	· the BEEF plane base
group	= FF3E::B:IDX	· global SSM

Read the first four bytes of the TopicID as one big-endian word, keep its top **bits**, and add the plane base **0x1000**, which is where the BEEF object plane lives in the 16-bit index space the address plan zones. Width 0 is special-cased to index 0, so the whole plane occupies the single group **FF3E::B:1000**. The plane grows upward from its base and the address plan lets it widen to 15 bits, 32,768 groups, before it meets any other allocation, without ever moving its base address. That is address space, not an operating range: §4.4 gives the join budget that bounds the width a plane actually runs at, and the lower rows of the table below are what the plan permits rather than what an operator should expect to use.

bits	Groups	Index for tm_uhrp_files	IDX	Reads as
0	1	0	0x1000	the whole plane, one group
4	16	0x4	0x1004	first hex digit
8	256	0x4B	0x104B	first two hex digits
12	4,096	0x4BD	0x14BD	first three hex digits
15	32,768	0x25EA	0x35EA	top 15 bits; runway cap

_1. SHA-256("tm_uhrp_files") begins **4bd434d6**, word **0x4BD434D6**; every index is a prefix of the one below it: widening reveals bits, it never moves a topic to an unrelated group._

2.1 Why the topic and not the object

Every object on the plane also has a content identifier, the hash of its bytes, and it would be natural to shard on that as the settlement plane shards on the TxID. It would also defeat the purpose. Content hashes are uniform over objects, so every topic's objects would land in every group, and a subscriber to one topic would have to join the entire plane to be sure of receiving it. Keying the group on the TopicID instead puts all of a topic's objects in one group, so the group itself becomes the coarse topical filter, and a subscriber joins exactly one group per elected topic. The content identifier keeps its own jobs, identity and reassembly, and is never consulted for addressing.

The TopicID is then used twice, at two widths, by two different parties (Figure 2). Its leading **bits** are the band: the network shards on them, and every router on the path forwards or prunes by them without knowing that a topic exists. All 256 bits are the topic: whatever filters matches on the whole identifier, and topics that share a band differ only past the band's bits. Between the two sits a demarcation. On the service side, the network shards by band, and the delivery edge applies each subscriber's election as an optimization, so that a subscriber's connection carries only the topics it elected. On the consumer side, the consumer's own topic manager matches the TopicID of what arrives and decides what to admit, exactly as it would over unicast. A consumer that takes bands directly, an aggregator or a peer at the network layer, does all of the topic filtering on its own side of the line.

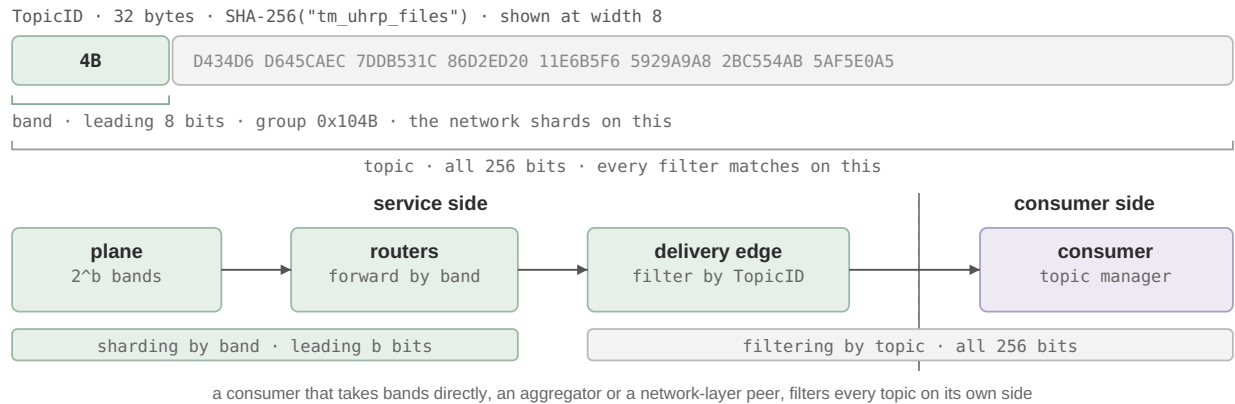


Figure 2 - one identifier, two widths: the leading bits select the band the network forwards, and the whole TopicID selects the topic. Sharding by band is service side; the delivery edge's per-subscriber filter is a service-side optimization; the consumer matches the full TopicID on its own side of the demarcation.

2.2 Why the top bits and not a modulo

The settlement-plane paper gives the full case: a modulo supports any group count but remaps about $K/(K+1)$ of all keys when K changes, while the top-bits rule turns a width change into a split, g into $2g$ and $2g+1$, with every index a prefix of the next. The property matters more here than there. A miner following a shard slice into its two children joins both. A topical subscriber following one topic joins only the child that inherits it, so a one-bit widening halves what it receives, at the price of two joins and one leave, and nothing else in the fleet is touched.

3. Why we shard: five things a band buys

A band is one group of the plane at the current width. Five properties follow from having more than one, and each is a reason to shard that has nothing to do with how much traffic there is today.

3.1 Selectivity the network enforces

Under source-specific multicast a tree extends only where a join came from. A band that nobody behind a link has elected is never forwarded onto that link, and the router's own pruning does it. This is the first and cheapest filter in the system, and it is enforced by the forwarding plane rather than by anything we run: a packet for a band with no downstream interest is not dropped at a delivery edge, it is never sent. The service-side filter then makes the exact cut, per subscriber, on what did arrive.

3.2 Parallelism by band

Spines, delivery edges and retransmission caches are assigned sets of bands, and a band carries about a $1/2^b$ share of the plane's topics. Capacity is added per band: a receiver for a quiet band is small, a receiver for a hot band is large, and when a band outgrows its hardware it is split in two and its halves land on different machines. Sets of bands are how a power-of-two group count is balanced across a fleet of any size, and the join model provides sets for free.

3.3 Bounded state

Multicast routing state is joined groups times sources. Ingress flow counters are kept per sender and group, and the TopicID is deliberately excluded from the flow key so that a sender's flow state cannot grow with the topics it publishes to. Retransmission caches key on flow and sequence number. A delivery edge holds one filter entry per topic its own subscribers elected, resolved by a constant-time lookup per frame. No component anywhere holds state proportional to the number of topics in the world, which is the property that lets millions of overlays coexist on one plane without any of them registering.

3.4 Growth without a flag day

Because the index is a prefix of the key, widening refines the partition instead of replacing it. The manifest protocol carries the transition: a successor width is announced per plane, emitters bridge both widths across a window, subscribers join each elected topic's child group before leaving its parent, and the plane settles at the new width with no key ever re-hashed to an unrelated group. One plane reshards without perturbing another, and the settlement plane never notices.

3.5 Legibility

An operator reads a topic's band off the first hex digits of its TopicID in a packet capture, a counter or a prefix-list, with no lookup service. A band is one contiguous block of indices and so is any set of its descendants, so reserving a range, bounding a range at an interface and exporting a range to a peer are all range arithmetic.

4. Sharding as traffic shaping

A multicast group is the unit the forwarding plane understands. Routers join and prune groups, hold forwarding entries per group and source, apply boundaries to group ranges, police and count per group, and export or refuse groups at a peering point. A topic is invisible to every one of those mechanisms. The shard width therefore sets the granularity at which an operator can see, steer, bound and meter this traffic, and choosing a width is choosing how finely the traffic can be shaped.

4.1 Three handles, three granularities

granularity	who acts	with what	sees
topic	delivery edge	per-subscriber election	TopicID
band (group)	every router	join · prune · boundary · policer · counter	FF3E::B:1xxx
source	BGP speakers	export · import · origin validation	S /128 in a /48

rows two and three exist in the network · row one exists only at an edge, for its own subscribers

Figure 3 - the three granularities an operator can act at. Only the two lower rows exist in the network; the top row exists only in a service-side filter, for its own subscribers.

The topic handle is ours and exact: a delivery edge delivers each subscriber the topics it elected and nothing else. The source handle belongs to the routing system: which emitters a network can reach is a matter of which source prefixes are advertised to it, and every tool BGP operators already use applies to that question unchanged. Between them sits the band, the only handle that both the network and the plane understand, and the one whose granularity the shard width sets.

4.2 The exchange rate of a bit

Consider a plane of one million topics, each publishing at the same rate of 1 kB/s, so the plane as a whole runs at 1 GB/s. A delivery edge whose subscribers have elected ten topics receives at most ten bands, each carrying the topics that share it by hash. The last column is the other side of the ledger: the (S,G) joins a receiver that covers two regions holds if it wants the whole plane, which is what an aggregator, a transit peer or a retransmission cache does.

bits	Groups	Topics per group	Edge ingest, ten topics wanted	Carried per wanted byte	Joins, whole plane, two regions
0	1	1,000,000	1 GB/s (the whole plane)	100,000×	2
4	16	62,500	625 MB/s	62,500×	32
8	256	3,906	39.1 MB/s	3,906×	512
12	4,096	244	2.44 MB/s	244×	8,192
15	32,768	31	305 kB/s	31×	65,536

1. One million topics at 1 kB/s each (stated); ingest is the upper bound $\min(10, 2^b)$ bands $\times$ topics per group $\times$ rate; the wanted bytes are 10 kB/s in every row; joins are 2 regions $\times 2^b$ bands, one anycast source per band per region.

Every added bit halves the unwanted bytes a link carries and doubles the joins a whole-plane receiver holds; the first is why we widen, the second is why we stop, and the rows below width 8 are address space, not a plan.

The same arithmetic applies to any link in the tree. A regional interconnect carries the bands joined anywhere behind it, so at width 8 a region whose subscribers want a dozen topics pulls under fifty thousand topics' worth of traffic across the long-haul rather than a million.

4.3 What an operator does with bands

Move	Mechanism	What subscribers see
Plan capacity	a band is a predictable $1/2^b$ share of the plane's topics	nothing
Rebalance load	assign bands to spines and links by measured rate	nothing
Split a hot band	widen one bit; $2g$ and $2g+1$ land on different hardware	two joins, one leave
Keep long-haul quiet	a region's transit carries only the bands its edges joined	nothing
Fence a range	a prefix-list boundary on group ranges at an interface	nothing
Police or alarm	per-group policers and counters on every router	nothing
Localize repair	retransmission caches follow bands; repair load tracks a band's own loss	nothing
Export to a peer	a group-range and source-prefix policy at the peering point	nothing

1. Every row is a standard router or routing-policy feature applied to a group range; none of them requires a subscriber to change anything.

The split is the move that makes the others safe to try: a rebalancing that proves wrong is undone by moving bands back, with no subscription changed.

4.4 The join budget, and why the plane stays narrow

Width is paid for in joins, and joins are the scarcest state in a multicast network. Under source-specific multicast a receiver joins each source and group pair explicitly: there is no join for a range of groups and no join for "any source", so a receiver that wants G groups from S sources holds $S \times G$ entries, every router on the path holds a forwarding entry per pair with downstream interest, and the receiving host's kernel bounds the source filters it will hold per socket. A delivery edge that covered every spine of every region at a few thousand groups would hold hundreds of thousands of joins. That is not an operating point; it is the reason width has a limit.

The design bounds the source dimension first. Each band has one source per region, an anycast address shared by that region's spines with one live emitter and the rest standing by, so a receiver joins each band once per region it covers, never once per spine. A receiver's join state is therefore at most bands $\times$ regions covered, and regions covered is one or two for a delivery edge and small for anything else. What remains is the band dimension, and it is bounded by a join budget the operator sets: a few hundred (S,G) entries per receiver is comfortable on every router and host in the path. At two regions that budget admits width 7 or 8, 128 or 256 bands, which is also where the settlement plane sits: a few hundred groups already carry its design target, and administrative burden binds there long before bandwidth does. The object plane scales independently of the settlement plane, but to the same shape and for the same reason.

Three further costs are stated plainly. Routers pay forwarding entries per source and group with downstream interest, which the same budget bounds. The operator pays the choreography of a generation transition each time the width moves. And width does not help the one case that needs it most: a single very hot topic is one group however wide the plane, and the mechanism to spread one topic across sibling groups is reserved by the specification rather than defined. So planes stay narrow, and a plane widens by one bit when per-group delivery capacity binds and the join budget allows, never because topics have become many.

Could aggregation loosen the budget? Three strategies are open. Within a region, any-source multicast at site scope lets a receiver join a group once regardless of source count, with a rendezvous point doing the source discovery; the IETF deprecated it only between domains. A regional re-emit tier, which the spine already is, collapses every upstream source into one per region, and could collapse further into one per site. And the designated sub-ranges of §5.4 could let a hierarchy of bands be joined at the level a receiver needs rather than leaf by leaf. Each buys joins at the cost of state or latency somewhere else, none is specified here, and until one is, the join budget is the limit and the plane is narrow.

5. Why this design scales

5.1 What grows with what

State	Scales with	Never scales with
Multicast routing (MFC)	joined groups $\times$ sources	topics, subscribers
Ingress flow counters	senders $\times$ groups	topics, objects
Retransmission cache	flows $\times$ recovery window	topics
Delivery-edge filter	topics elected by its own subscribers	the global topic count
BGP (unicast + SAFI 2)	source prefixes, one aggregate per region	topics, groups
Per-frame work	deliverable names in the record (at most 3)	elected topics
A subscriber's joins	$\min(\text{elected topics}, 2^b) \times \text{regions covered}$	the plane's topic count

No component anywhere in the network holds state proportional to the number of topics in the world; that is the property that lets any name work, instantly, at any scale.

The routing row is the one to notice. The plane's sources are a few spines per region, advertised as one aggregate prefix per region, so the entire routing footprint of a planet-scale plane is tens of routes. That is small enough to validate completely and police completely, a point §7 returns to.

5.2 The alternatives, at their strongest

One group forever. The simplest design, and the right one to start with. It fails only when per-group delivery capacity binds, and when it fails it fails everywhere at once, because every edge and every link carries the whole plane. What it legitimately buys is the absence of transitions, and the design keeps that benefit for as long as the plane can afford it.

A group per topic. Perfect selectivity: a subscriber joins exactly its topics and the network forwards exactly those. It also puts one forwarding entry per live topic in every router on the path and one join per topic per source at every subscriber, so the network's state grows with the topic population, which is unbounded. Name-based inter-domain routing research has spent the better part of two decades on the same wall: a routing table does not survive a billion prefixes. What a group per topic buys is nothing the service-side filter does not already provide, at a cost the network cannot pay.

A modulo. Covered in full by the companion paper: exact fit to a non-power-of-two fleet, in exchange for a flag day at every resize.

Choosing names for their hash. A publisher can grind a topic name until its TopicID lands in a chosen band, cheaply. This needs no protocol change and is harmless, but it is a convenience and not a mechanism: nothing enforces it, co-residence with other topics remains, and at width 0 it means nothing at all.

5.3 The ladder

Scale comes from four levers, and width is the smallest of them. Width splits groups within the join budget of \$4.4, so it shapes traffic more than it adds capacity. Hardware per band adds capacity: a band is a unit a receiver, a cache or a spine can be sized to, and a hot band is split once and lands on two machines. Regions multiply sources, one anycast source per band per region, so a region's emitters serve its own receivers and a receiver joins only the regions it covers. Topologies multiply by peering, which §8 describes, so that each operator's plane scales on its own hardware and its own policy, and a peer's receivers join the peer's sources, not ours. The address plan also holds thirteen more plane slots in reserve for planes not yet imagined, which are never spent for scale. Hardware per band, regions and topologies do the scaling; width does the shaping, within its budget.

5.4 The index space beyond the hash

The address plan's room for 32,768 groups is address space, not a plan. A plane sharded purely by hash will not run tens of thousands of bands, because the join budget of \$4.4 stops it at a few hundred. The more likely use of a wide index space is structured: sub-ranges designated rather than hashed, forming hierarchies within the plane, boundaries that third-party routers can fence by prefix, or shard methods other than the topic hash for adjunct infrastructure that lives beside the topical plane, such as proof services, archives and lookup tiers. How such a space should be laid out, and how topical filtering over a SHA-256 key space scales beside it, is open research. The design keeps the space contiguous and unallocated so that it stays open.

6. Topics as addresses

6.1 Addressing by subscription

A topic name is any string, and the network attaches no meaning to it. That makes the name an address the receiver chooses by computing it. Conventional overlay topics name communities, `tm_uhrp_files` for one. But a name can equally be a transaction ID, so that the topic is where that transaction's proof update and its spends are published; an outpost; the hash of a locking script, so that every payment to that script is a message on its topic; a public key; or a key derived for one relationship. Publisher and subscriber agree on a naming convention and nothing else, and the plane delivers whatever is published there, with proofs. This is addressing by subscription: the receiver's interest is the address, there is no registry, and a billion such addresses cost the network nothing, because nothing in it scales with topics.

The nearest existing thing is the Electrum protocol's script-hash subscription, in which a client sends a server the SHA-256 of a locking script and is notified when that script's status changes. It is the right idea and it has served wallets for years. What it leaves on the table is the structure: the server learns which scripts each wallet watches and does the watching for it, one server, request and response, and the notification is a status to be followed by a fetch. On the plane the hash is the group's key, delivery is publish-once multicast with the object and its proofs inside the notification, the only party that learns the subscription is the delivery operator, and that operator can be given the hash and never the name.

6.2 Derivation privacy

Because the TopicID is a hash, the name need never be disclosed: subscriptions are elected by TopicID. Two parties can derive a topic name from their keys under the standard key-derivation scheme, with a protocol identifier, a key identifier and the counterparty's key as inputs, so the name is known only to them and is rotated per invoice or per message. An observer sees a 32-byte identifier it cannot link to either party or to any earlier topic. On the wire the topic sits among its band's co-residents, so even the group an observer sees joined says little about who is talking to whom.

What the design does not hide is stated plainly. A topic named by a public fact, a transaction ID or a script hash, is derivable by anyone who knows the fact, so it is a public rendezvous and not a private channel. The delivery operator sees the elected TopicIDs of its own subscribers, if not their names. Traffic volume on a derived topic is visible to whoever can see the band. And the payloads are BEEF objects, which are public transactions; confidentiality of content is the application's job, as it always was.

6.3 Multi-party interaction

A shared topic is a rendezvous for any number of parties with no coordinator. Name it by the funding transaction of a multi-party contract, or derive it among the participants, and every state update published there, each a BEEF object and each provable, lands on every participant at the same moment, with the network repairing loss. Channel updates, multi-signature coordination, auctions and games take the same shape: the chain is the shared state, the topic is the bus, and nobody runs a server in the middle. A party that joins late backfills through the overlay synchronization protocols; the plane carries the present.

6.4 Identity and verification

Objects are transactions, so every object on a topic is signed by the keys that spend its inputs, and a topic named after a key binds the two. A subscriber of a key's topic checks that each object is spent by, or carries a field signed by, that key, and that the object's proofs anchor to the chain: the topic is the address, the signature is the sender, and the proof is the time. Identity certificates and payment destinations fit the same shape. A payee's topic can be the destination derived from its identity key under the payment protocol, so a payment published there arrives, with its proof, at the one party who can spend it. The network checks none of this; the subscriber's own topic manager does, which is exactly what keeps the plane free of per-topic state.

6.5 What this opens

Three things follow and none of them is built here. A naming convention for transaction-, script- and key-derived topics belongs in a specification of its own. The interplay between such topics and the band structure, a few very public topics beside millions of quiet derived ones, is the same research question as §5.4 seen from the application side. And the privacy properties of §6.2 deserve a treatment with an adversary model rather than a paragraph. What this paper establishes is that the plane already supports all of it, because a topic was never more than a hash.

7. A different distribution model

7.1 Destination-routed unicast, and what it makes the publisher do

The Internet distributes by destination. A packet names where it is going; routers forward it toward that address; and a publisher that wants a thousand recipients to have the same bytes sends them a thousand times, or pays a content network to send them from a thousand nearer places. Fan-out is the sender's problem, its cost grows with audience, and the network sees only addresses: it cannot tell that the thousand packets are one object, so it can neither replicate them nor repair them as one.

Sharded, layered multicast inverts each of those. Replication happens at branch points in the network, once per link rather than once per recipient. Receivers pull: a tree exists only where a join asked for it, and it ends where interest ends. Flows are numbered end to end, so absence is observable and the network repairs its own loss. Bands make the traffic legible to every router on the path, so it can be bounded and shaped at the granularity §4 describes. And the layering puts each kind of traffic on its own plane, so settlement, headers and objects are operated, scaled and peered independently. *The overlay object plane* describes what this removes from an application's architecture; this section is about what it changes in the network's.

7.2 Why it is more securable

The security properties follow from the same structure, and most of them are inherited rather than added.

- **A receiver names the source it trusts.** Source-specific multicast has no rendezvous point and no any-source injection: a join is for a specific source and group, and only that source's traffic arrives on that tree. This is why the IETF deprecated any-source multicast between domains and why the plane runs source-specific everywhere.
- **Sources are routed prefixes, so routing security applies to them.** An emitter is a /128 inside an aggregate the operator originates in BGP. A Route Origin Authorization in the RPKI states which autonomous system may originate it, and prefix origin validation lets every peer reject a forged origin before any tree can be pulled toward it. BGPsec path validation and the provider-authorization work in progress at the IETF extend the same protection to the path. Content distribution inherits the routing system's security tools because its sources are routes.
- **Objects prove themselves.** Every object on the plane carries its own Merkle proofs and is verified at every receiver against headers the receiver obtained itself. The network is trusted to deliver, never to attest; a tampered object fails verification wherever it lands.
- **Repair is keyed to the ingress stamp.** The retransmission key is fixed when an object enters the network and does not depend on which source re-emits it, so a cache can serve repairs for a flow it did not originate, and a forged repair cannot match a flow it was not stamped into.
- **Blast radius is a band.** A flood, a misbehaving publisher or a compromised emitter is confined to the bands it reaches, and a band is a range a router can police or fence without touching the rest of the plane.
- **The routing footprint is small enough to secure completely.** The unicast default-free zone carries on the order of a million prefixes, and securing all of them is a decades-long project. The plane's sources are tens of aggregates. Every one can carry a Route Origin Authorization, every peer can validate every one, and every peering point can enumerate the permitted source and group pairs in full, which is exactly the control the inter-domain peering practice recommends.

The last point is where further work is possible. Which autonomous systems may originate a plane's sources, which may transit them and which may peer them are questions the RPKI can answer completely for a few dozen aggregates, in a way it cannot yet for the unicast Internet. Distribution built this way is as much an upgrade to how the Internet moves one object to many parties as it is to how applications are built on top of it.

8. Peering the plane

8.1 Two surfaces

The plane has two kinds of consumer. The first is the subscriber surface: a delivery edge hands each subscriber the topics it elected over a tunnel, metered on delivered bytes. The second is the peering surface: another network takes the plane as a network rather than as an application, at transit scale, and carries it onward to its own receivers. A peer of this kind may take every plane and every band at once over a pair of direct high-speed connections, or a few bands over an exchange port; either way it is a consumer at transit level, open to other multicast operators, to ISPs and to Internet exchanges directly. Its contract is deliberately left open in this paper beyond one commitment: it is built from standard tools and from the plane's own published formats, and it is a kind of consumer, so it is metered and governed the way a consumer is.

8.2 Three layers to take it at

The system, its formats and its protocols interoperate at different layers, and a peer chooses the layer that fits its network. The frames, the records and the repair protocol are the same at every layer; only the mechanism that carries them across the border changes.

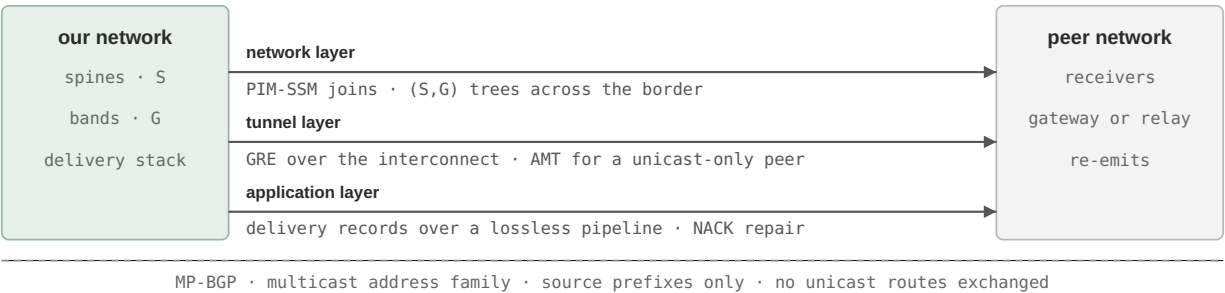


Figure 4 - one plane, three layers a peer can take it at. The frames and records are the same at each; what changes is whether the peer's routers join our trees, a tunnel carries them, or the peer runs the plane's own delivery components as a transit-level consumer and re-emits.

Network layer. This is the inter-domain multicast practice the IETF documents as best current practice: source-specific multicast only, PIM-SSM joins across the peering point, and MP-BGP carrying source prefixes in the multicast address family so that reverse-path forwarding resolves toward our emitters. A private interconnect carries an eBGP session that exchanges nothing but what the plane needs: our source aggregates in one direction, the peer's in the other if it re-emits, and no general unicast routing. A router that resolves reverse-path forwarding from its unicast table alone takes the same few aggregates in the unicast address family, and still nothing beyond the plane's sources. Public peering at an exchange can add unicast routing exchange later if a reason arises; nothing in the plane requires it. The peer's receivers join our sources and groups directly, one tree spans both networks, and the peer shapes what it takes with the handles of §4: a group-range policy for bands, a prefix policy for sources, and the enumerated set of permitted source and group pairs the practice recommends for denial-of-service protection. This is configuration on routers the peer already owns.

Tunnel layer. Where the peering point itself is not multicast-enabled, the same session and the same joins run inside a GRE tunnel, which is the practice's own second scenario. Where the peer's network runs no multicast at all, Automatic Multicast Tunneling lets a gateway in the peer's network request groups from a relay at our edge over plain unicast. It is a standard with an implementation in the Linux kernel, in router operating systems, and in relays run by research networks; it is a niche tool rather than a common one, and the plane does not depend on it.

Application layer. A peer that wants neither can run the plane's own components: the delivery side that takes the plane as a subscriber would, over a lossless pipeline on the interconnect, and the ingress side that re-emits it under the peer's own sources, with the retransmission cache beside it. This is the transit-level consumer of §8.1 in its fullest form: every plane, every band, over a pair of direct connections, with the peer's internal trees rooted in its own network and repair across the border served by one proxied request to our cache. Nothing in it is new; it is the same stack a delivery operator runs, pointed at a network instead of an application.

Property	Network layer	Tunnel layer	Application layer
Peer runs	PIM-SSM and MP-BGP at the border	a GRE endpoint, or an AMT gateway	the plane's delivery and ingress stack
What crosses	the bands the peer joined	the bands the peer joined, encapsulated	delivery records for what the peer elected
Routing exchanged	source prefixes, multicast SAFI only	the same, inside the tunnel	none
Trees rooted	our spines, into the peer's network	our spines, into the peer's network	re-emitted under the peer's own sources
Repair	the retransmission protocol, end to end	the same	the same, with one proxied hop at the border
Reference	RFC 8313, RFC 4760, RFC 4607	RFC 8313 §3.2, RFC 7450	BRC-124, BRC-126, BRC-148, BRC-149

1. The three layers carry the same frames and records; a peer may start at one and move to another without the plane changing.

8.3 Borders a network can rely on

Because the handles are the routing system's, a border is drawn with the routing system's tools and holds for the routing system's reasons. A network that must keep certain traffic inside its territory advertises no sources for it outward, so no tree can be pulled across: enforcement is the physical absence of a path. A network that must admit only certain traffic imports only those sources and joins only those bands. Fences on group ranges at the border interfaces do the same for bands that must never leave or never enter. None of this is new machinery; it is the administratively scoped multicast that operators have fenced for twenty years, applied to ranges the plan makes meaningful. The settlement plane is unaffected by any of it: miners want everything everywhere, and it stays one global plane.

9. Assumptions and limits

Quantity	Value	Basis
Topic identifier	32 B, SHA-256 of the UTF-8 name	verified: BRC-148 topic identifier definition
Group derivation	$0x1000 + (\text{TopicID}[0:4] \text{ BE}) \gg (32 - \text{bits})$	verified: BRC-148 topical sharding; BRC-129 rule
Plane runway	width 0 to 15; 1 to 32,768 groups; $0x1000 - 0x8FFF$	verified: BRC-148 plane allocation and growth policy
Deliverable names per record	public door 1, authenticated up to 3; up to 15 named	verified: BRC-149 record; operator cap
Topics per group at 10^6 topics	$10^6 / 2^b$ (244 at 12, 31 at 15)	structural: uniform hash, statistical
Edge ingest bound	$\min(T, 2^b) \times (N / 2^b) \times r$	structural: one join per elected band
Per-topic rate in §4.2	1 kB/s; $N = 10^6$; $T = 10$	stated: model inputs, chosen for legibility
Split on widening	g becomes $2g, 2g+1$	structural: the index is a key prefix
Receiver join state	bands $\times$ regions covered; one anycast source per band per region	structural: SSM joins are per (S,G); anycast per band
Join budget and operating width	a few hundred (S,G) per receiver; width 7 to 8 at two regions	stated: operating posture; settlement plane precedent
BGP table for the plane	source aggregates per region, tens of routes	structural: BGP carries sources, never topics
Unicast default-free zone	on the order of 10^6 prefixes	stated: public routing-table statistics, order only
Script-hash subscription	SHA-256 of the locking script	verified: Electrum protocol, basics and methods
Inter-domain posture	SSM only; PIM-SSM + MP-BGP SAFI 2; GRE or AMT	verified: RFC 8815; RFC 8313 §1, §3
AMT availability	Linux kernel driver; router OS support; research-network relays	verified: kernel tree <code>drivers/net/amt.c</code> ; IETF deployment report

Limits, stated plainly:

- Uniformity is statistical and per topic, not per byte.** Topics spread evenly across bands; traffic does not, because topics differ in rate by orders of magnitude. A hot band is split by widening; a hot topic is not, and the spread mechanism for one hot topic is reserved, not specified.
- Width is bounded by a join budget, not by the address plan.** Joins are per (S,G) and there is no range join; a few hundred per receiver is the operating budget, which puts the plane at tens to a few hundred groups. The strategies that could loosen it, any-source multicast within a region, a re-emit tier per site, hierarchical sub-ranges, are open research and not specified here.
- Selectivity by band is coarse by design.** A subscriber of one topic receives its whole band and filters at the delivery edge; the exact cut is never the network's job. The exchange rate of §4.2 is the whole promise.
- The wide index space is unplanned on purpose.** Tens of thousands of hash-sharded bands are a runway, not an expectation; structured uses of the space, and the scaling of topical filtering beside them, are open research (§5.4).
- A topic as an address is a convention, not a guarantee.** Public facts make public rendezvous; the delivery operator sees TopicIDs; payloads are public transactions. Privacy claims stronger than §6.2 need an adversary model this paper does not give.
- Native peering needs a multicast-capable peer.** PIM-SSM and MP-BGP at the border are ordinary router configuration; a network without them takes the plane through a tunnel or runs the plane's components, each of which adds a hop and something to operate.
- The peering surface's contract is open.** This paper commits it to standard tools, the plane's published formats and the consumer's shape; its terms are the operator's to set.
- Routing security protects the path, not the content.** Origin and path validation ensure a tree is pulled toward the right emitter; the object's own proofs ensure the bytes are right. Neither substitutes for the other, and the design relies on both.

References

BRC-29: Simple Authenticated BSV P2PKH Payment Protocol: <https://github.com/bsv-blockchain/BRCs/blob/master/payments/0029.md>
BRC-42: BSV Key Derivation Scheme (BKDS): <https://github.com/bsv-blockchain/BRCs/blob/master/key-derivation/0042.md>
BRC-43: Security Levels, Protocol IDs, Key IDs and Counterparties: <https://github.com/bsv-blockchain/BRCs/blob/master/key-derivation/0043.md>
BRC-52: Identity Certificates: <https://github.com/bsv-blockchain/BRCs/blob/master/peer-to-peer/0052.md>
BRC-124: Multicast Transaction Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0124.md>
BRC-126: Multicast Transaction NACK Retransmission Protocol: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0126.md>
BRC-129: IPv6 Multicast Group Address Assignments: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0129.md>
BRC-139: Multicast Shard Manifest Announcement Protocol: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0139.md>
BRC-148: Multicast Shard Domain Partitioning and the BEEF Object Plane: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0148.md>
BRC-149: Multicast BEEF Object Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0149.md>
Electrum protocol, Protocol Basics (script hashes) and Protocol Methods (`blockchain.scripthash.subscribe`): <https://electrumx.readthedocs.io/en/latest/protocol-basics.html>

IETF, RFC 4607: Source-Specific Multicast for IP: <https://www.rfc-editor.org/rfc/rfc4607>
IETF, RFC 4760: Multiprotocol Extensions for BGP-4: <https://www.rfc-editor.org/rfc/rfc4760>
IETF, RFC 6811: BGP Prefix Origin Validation: <https://www.rfc-editor.org/rfc/rfc6811>
IETF, RFC 7450: Automatic Multicast Tunneling: <https://www.rfc-editor.org/rfc/rfc7450>
IETF, RFC 8205: BGPsec Protocol Specification: <https://www.rfc-editor.org/rfc/rfc8205>
IETF, RFC 8313 (BCP 213): Use of Multicast across Inter-Domain Peering Points: <https://www.rfc-editor.org/rfc/rfc8313>
IETF, RFC 8777: DNS Reverse IP AMT Discovery: <https://www.rfc-editor.org/rfc/rfc8777>
IETF, RFC 8815 (BCP 229): Deprecating Any-Source Multicast (ASM) for Interdomain Multicast: <https://www.rfc-editor.org/rfc/rfc8815>
IETF MBONED, AMT relay deployment experience (Internet2 public relay): <https://datatracker.ietf.org/meeting/101/materials/slides-101-mboned-amt-relay-deployment-experience-01.pdf>
IETF SIDROPS, A Profile for Autonomous System Provider Authorization (Internet-Draft, work in progress): <https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-profile/>
Linux kernel, Automatic Multicast Tunneling driver ([drivers/net/amt.c](#)): <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/drivers/net/amt.c>
Shard bit extraction (Lightweb Inc., Paper 005): <https://1bsv.net/papers/shard-bit-extraction.pdf>
The overlay object plane (Lightweb Inc., Paper 006): <https://1bsv.net/papers/overlay-object-plane.pdf>

1BSV - BSV Layered Multicast - cash moves fast on 1BSV. · 1bsv.net

Lightweb Inc. · Jeff Harris · jeff@lightweb.net · github.com/lightwebinc

Unbounded Solutions for a Small World™ · © Lightweb Inc.