

Multicast over a unicast Internet: the fabric is an overlay

"The Internet does not do multicast" is true, and it is the most common reason given for why a multicast distribution network cannot work. This paper takes the objection at its strongest and then shows why it does not apply: the fabric never asks the Internet to forward a multicast packet. Every multicast hop is a link between two routers one operator runs, and where that link has to cross the Internet it is a point-to-point tunnel carrying ordinary unicast datagrams. It defines the fabric by its three tiers, states what each router must be able to do, shows what a consumer receives and what a peer network may take, maps every public seam to the open standard that governs it, prices the overlay form honestly against native multicast, explains why this multicast can scale globally where earlier attempts could not, and closes with the migration path: every link can be upgraded in place, from tunnel to cross-connect to native multicast peering, without changing anything above it.

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

Paper 012 · Version 01

Audience: network engineers at ISPs, exchanges and mining operations, and anyone who has been told that multicast cannot cross the Internet.

We = Lightweb Inc., author of this architecture and operator of the first fabric built on it; nothing here depends on that operator, and the components are open for any other.

You = a network engineer deciding whether this can run on your network, or next to it.

Every figure uses the same colors: gray for the public Internet and anything a third party runs, green for the fabric's routers and the trees between them, purple for consumers. A dashed gray line is a unicast tunnel across the Internet; a solid green line is a multicast tree; a teal line is a repair.

The short version

- **The objection is correct about the Internet.** Transit providers do not forward multicast between customers, exchanges do not switch it, and access networks drop it. Nothing in this paper depends on that changing.
- **The fabric asks the Internet for unicast only.** Every multicast hop is a link between two routers one operator runs. Across the Internet that link is a GRE tunnel, and the Internet sees unicast datagrams between two fixed addresses, which is all it has ever carried.
- **A tunnel is a link.** Multicast routing runs on interfaces, not wires. A tree that spans a tunnel is built by the same protocol, in the same way, as a tree that spans a cable. The IETF's own inter-domain multicast practice names this case.
- **The fabric is three tiers of routers.** Ingress is the door a sender reaches, the spine is where trees branch and capacity is spent, and the edge is where a tree ends and delivery begins. Links between them may be of any kind.
- **A consumer never has to run multicast.** Delivery to a miner or an application is a unicast TCP stream inside a tunnel the consumer holds the keys to. A peer network may instead take the trees natively at a border, and both are consumers of the same network.
- **The overlay costs one copy per downstream tunnel, and that is the whole cost.** It is still one send from the publisher, replication at branch points instead of at the source, and only the groups a receiver joined. The paper states where native multicast would do better.
- **What rides on it is why it scales.** Earlier global multicast had no economics, no admission control and no way to check content. Here every frame carries a Bitcoin transaction that cost its sender a fee, every object proves itself to the receiver, delivery is paid for by the byte, and admission is bounded at the door by proof and by budget.
- **Every link can be upgraded in place.** Groups are global and static, sources are routed prefixes, and the frames and the repair protocol are the same at every layer, so a tunnel becomes a cross-connect, a cross-connect a jumbo link, and a tunneled peering a native one, with nothing above the link changing.

1. The objection, at its strongest

Here is the case against it, as a network engineer would put it. IP multicast was designed in the late 1980s and the experimental Mbone that carried it across research networks in the 1990s did not survive contact with the commercial Internet. Transit providers do not forward multicast between customers. Internet exchanges switch unicast and treat anything else as a fault. Consumer access networks either drop multicast or confine it to the provider's own television service. The one standard written to bridge the gap, Automatic Multicast Tunneling, has a relay in the Linux kernel and a handful of public relays run by research networks, and almost nobody has heard of it. Where multicast is deployed at scale today it is inside one operator's network, for that operator's own traffic, and it stops at that operator's border.

Every sentence of that is true, and a paper that pretended otherwise would lose its reader on the first page. If the fabric needed the public Internet to forward multicast, the objection would be fatal and the correct answer would be to build something else.

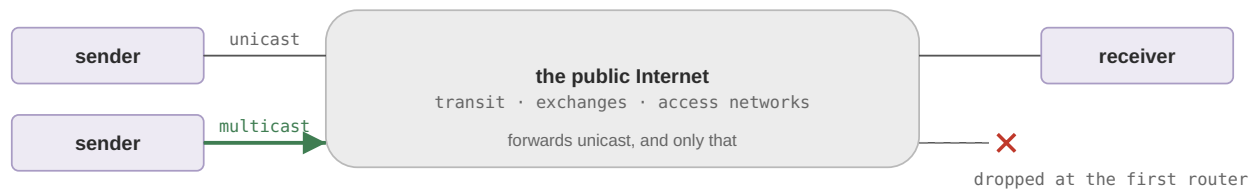


Figure 1 - the objection. The Internet forwards a unicast datagram from one address to another and drops a multicast one at its first router. This paper does not dispute it.

2. What a multicast network actually needs

Strip multicast to its requirements and there are four. A set of routers that run a multicast routing protocol between them. Links that connect those routers. Receivers that can tell their nearest router which traffic they want, by joining a group. And a way for the routers to know where that traffic comes from, so a tree can be built from the receivers back toward the source.

None of the four says the links must be physical, and none says the routers must belong to a transit provider. A link is anything a router can send a packet into and have it come out at the other end; the routing protocol sees an interface and does not care what is behind it. That observation is the whole of this paper, and the rest of it works through the consequences.

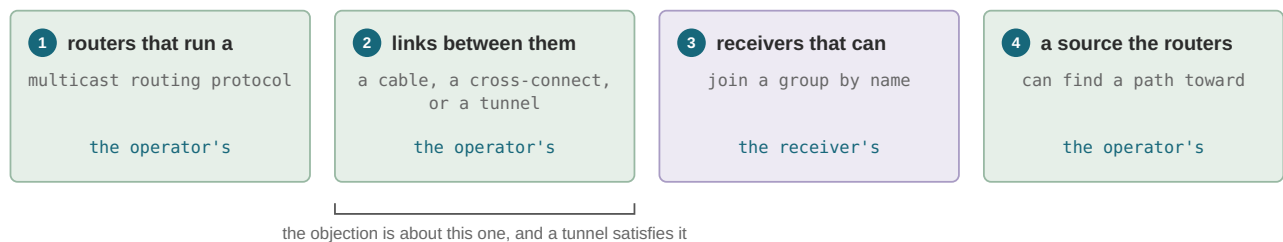


Figure 2 - the four requirements of a multicast network. The second is the one the objection is about, and it is satisfied by a tunnel as well as by a cable.

3. A tunnel is a link

Generic Routing Encapsulation wraps one packet inside another. A router at one end takes an IPv6 multicast datagram, puts a GRE header and a new IPv6 unicast header in front of it, and sends the result to the router at the other end, which strips both and forwards the inner datagram as if it had arrived on a cable. The outer packet is unicast from one fixed address to another, and the Internet forwards it the way it forwards every other packet. The routers at each end treat the tunnel as an interface, run their multicast routing protocol across it, and build trees through it. GRE over IPv6 is a published standard, and the inter-domain multicast practice the IETF documents as best current practice names the GRE tunnel as its second scenario, for the case where a peering point is not multicast-capable.

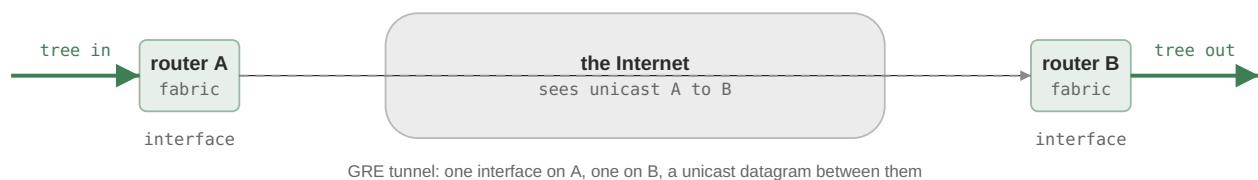


Figure 3 - a tunnel is a link. The tree enters an interface on router A and leaves an interface on router B. Between them the Internet carries a unicast datagram and never sees the multicast inside it.

What the Internet carries is shown in Figure 4. The inner frame is the multicast datagram: an IPv6 header addressed to a group, a UDP header, and one of the fabric's frames. The outer header is an IPv6 unicast header addressed to the far router, and a GRE header between them says what is inside. The encapsulation costs 52 bytes on a 1500-byte path, which is why the fabric fragments and bundles the way it does; *Frames on a 1500-byte Internet* is about that cost and what jumbo links do to it.

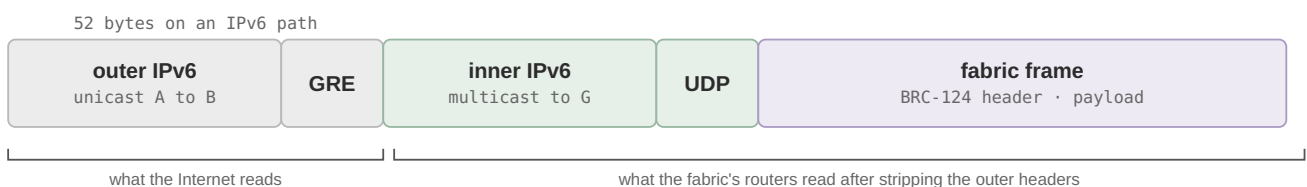


Figure 4 - the datagram on the wire. Routers on the Internet read the outer header and nothing else. The fabric's routers strip it and read the inner one.

Nothing about this is specific to this fabric. It is the mechanism every operator uses to carry multicast between two of its own sites across a provider that does not, and it has been in production use for as long as GRE has existed. The fabric's contribution is not the tunnel; it is what runs over it, and the fact that the whole fabric is built so that tunnels are an acceptable link rather than a special case.

4. The fabric defined

The fabric is a set of routers one operator runs, in three tiers named by what they do. The tiers connect by links, and a link is a cable, a cross-connect at a facility, or a tunnel across the Internet, chosen per link by what is available and what it costs. Every router runs the same multicast routing protocol; the tiers differ in role, not in software.

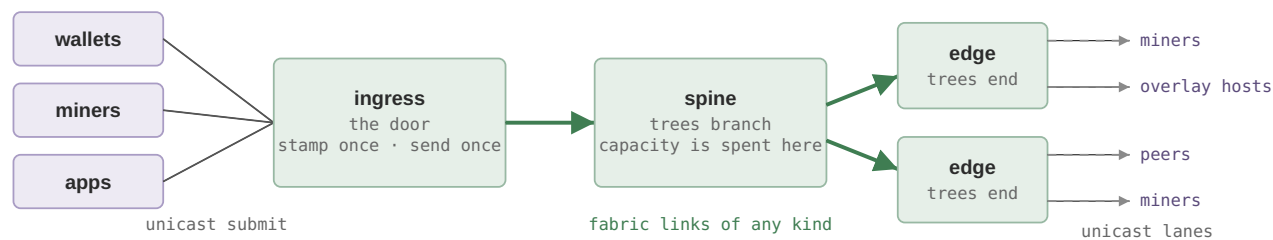


Figure 5 - the three tiers by role. A frame enters once at an ingress, is replicated at the spine wherever a tree branches, and leaves the last tree at an edge, which delivers it over unicast to each consumer.

Ingress is the door. A sender reaches the nearest one and hands over a transaction or an object, once. The ingress stamps it with a flow key and a sequence number, derives its group from its hash by the public rule, and sends it into the fabric a single time. **Spine** routers carry the trees between regions and fan them out toward the edges; this is where trees branch, where the fabric's capacity is spent, and where a faster link pays for itself first. **Edge** routers are where a tree ends. Beside each edge runs the delivery component that takes frames off the trees and writes them to consumers over unicast. Sections 6 to 8 take the tiers in turn.

Each tier scales on its own axis, and none waits on another. Ingress scales with senders: add a door where senders are, and the nearest-door routing sends them to it; doors hold no state about each other, so the hundredth is as easy as the second. The spine scales with geography and volume: add a router where trees need to branch, or a link where a tree needs more capacity, and the trees re-form across it. The edge scales with consumers: add an edge where subscribers are, and it joins the groups they want. A region with many senders and few subscribers has many doors and few edges; a region with the opposite has the opposite; the spine between them is sized to the trees that cross it. No tier has to be over-built to let another grow.

Tier	Grows with	Add one when	Independent of
ingress	senders	a sender population lacks a near door	how many consumers or edges exist
spine	regions and volume	a tree needs a new branch or more capacity	how many doors or consumers exist
edge	consumers	subscribers lack a near edge	how many senders or doors exist

1. Each tier is added where its own demand is, and the trees re-form around the addition.

5. What each router must do

The requirements on a fabric router are standard router features, each with a published specification, and nothing proprietary sits in the routing plane. A fabric router runs IPv6. It runs Protocol Independent Multicast in its source-specific form, which builds a tree from each receiver back toward one named source along the reverse path. It answers Multicast Listener Discovery version 2 from receivers on its links, which is how a receiver names the source and group it wants. It needs a route toward every source, because the reverse path is computed from the unicast routing table; between domains those source routes travel in the multicast address family of multiprotocol BGP, so that a peer can carry them without accepting any general unicast routing. And it needs to know which group ranges are the fabric's, which are published.

Requirement	Standard	Who runs it
IPv6 forwarding	RFC 8200	every fabric router, every peer
Source-specific multicast routing	RFC 7761 (PIM-SM, SSM mode), RFC 4607	every fabric router
Receiver joins with a named source	RFC 3810 (MLDv2)	the router nearest each receiver
Source routes for reverse-path lookup	RFC 4760, multicast address family	border routers between domains
Tunnels where the link is the Internet	RFC 7676 (GRE over IPv6)	both ends of each such link
Group ranges	BRC-129	configuration, not protocol

1. Every row is a feature a router operating system already ships. The fabric adds no routing protocol of its own.

Two things are deliberately absent. There is no rendezvous point and no shared tree, because the fabric runs source-specific multicast everywhere; *One source, not any source* explains why that choice is what makes inter-domain operation possible at all. And there is no requirement that any link be multicast-capable on its own: a link that is not becomes a tunnel, and the router at each end does not know the difference.

6. Ingress

A sender submits a transaction in Bitcoin's extended format, which carries the inputs' previous outputs alongside the transaction so the door can check it without a lookup, or a complete overlay object with its proofs, to the nearest ingress over ordinary unicast. The raw format without that context is not accepted at the door. Submission is open and free; *The cost of being reachable* explains why admission is priced at nothing and delivery is the product. The ingress does three things to what it receives. It stamps the frame with a flow key and a sequence number, so that every copy of it anywhere in the fabric is the same numbered frame and a missing one can be noticed. It derives the frame's group from the front of its hash, the one-line rule *Shard bit extraction* walks through, so that any party can compute which group a transaction will travel on. And it sends the frame into the fabric once.

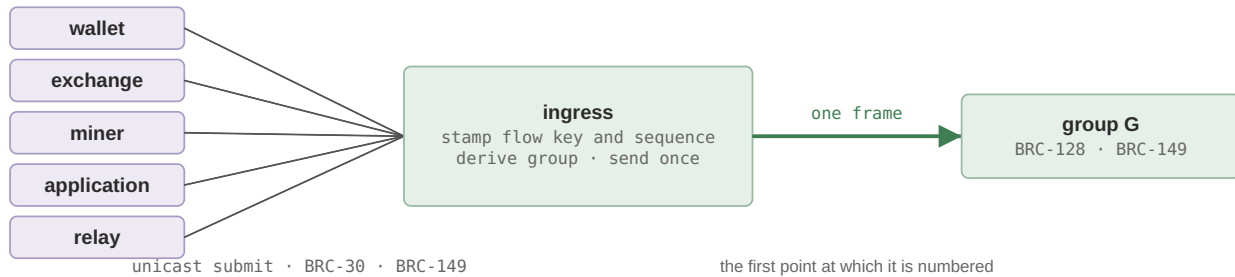


Figure 6 - many senders, one frame out. The ingress is the last point at which a transaction is unicast and the first at which it is numbered.

From the sender's side there is nothing to learn. The same bytes a wallet would hand to any node are handed to the ingress, and the ingress is reached the way any server is reached. The multicast begins behind the door.

7. Spine

The spine is the tier that carries trees between regions and fans them out toward the edges. A frame that entered at one ingress has to reach every edge with a subscriber for its group, and the spine is where that fan-out happens: at each spine router the tree branches once per downstream link that leads to a subscriber, and the router sends one copy into each. The spine is therefore where the fabric's capacity is concentrated and where the cost of the overlay form is paid, as §13 prices. It is also where a better link pays for itself first: a cross-connect in place of a tunnel, or a jumbo-frame link in place of a 1500-byte one, raises the ceiling for every tree that crosses it. *Frames on a 1500-byte Internet* covers what jumbo frames buy.

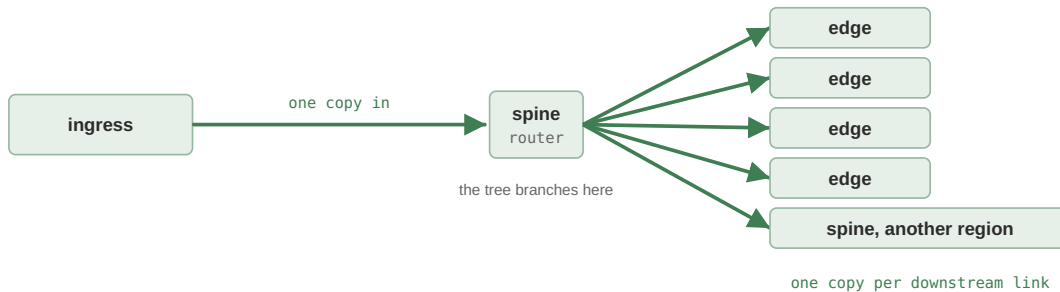


Figure 7 - a tree branching at the spine. One copy arrives; one copy leaves on each link toward a subscriber. Links to other regions are usually the long ones and the ones upgraded first.

Nothing about the spine's role depends on how its links are built. A spine whose links are all tunnels across the Internet and a spine whose links are all dark fiber run the same protocol and build the same trees; they differ in how many bytes each copy costs and in how large a frame fits, which is the subject of §13 and §15.

8. Edges and what leaves them

An edge is where a tree ends. Beside each edge router runs the delivery component that joins the groups its consumers asked for, takes the frames off the tree, notices and repairs anything missing (§12), and writes whole objects to each consumer. Delivery is unicast: a TCP stream per object class, inside a tunnel the consumer requested and holds the keys to, with a second path for failover. *Pushed delivery into a Teranode cluster* and *Landing in Kubernetes* specify those lanes and how a mining cluster receives them; from the consumer's side the whole fabric reduces to reliable streams arriving at addresses they named.

The multicast stops at the edge on purpose. The fabric does not extend a group over a consumer's tunnel, because doing so would put group joins, multicast routing state and loss repair on the consumer's side of the line, and a consumer who wanted none of that would have to run all of it. The edge absorbs it instead. The consumer runs no multicast routing, joins no group, and sees no fragment, bundle or repair: those are resolved before the stream is written.

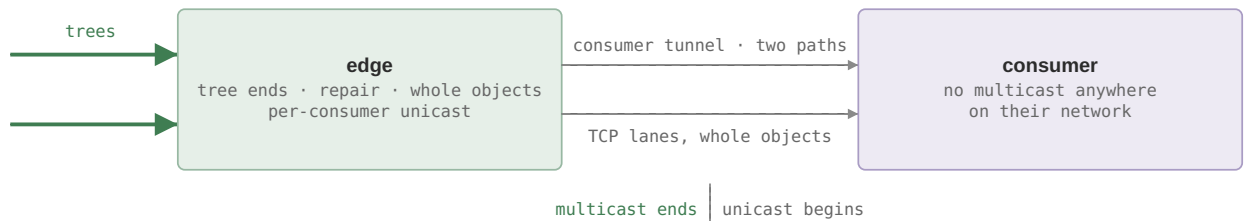


Figure 8 - the demarcation. Multicast lives entirely between the fabric's routers. A consumer receives unicast TCP over a tunnel, and nothing about multicast reaches their network.

9. Two ways downstream can look

The edge in §8 is the shape most consumers want: nothing to run, nothing to learn, a stream arriving where they said. It is not the only shape. A network that already runs multicast, or that wants to, may take the trees themselves at a border, as §10 describes, and carry them onward to its own receivers. Both are consumers of the same fabric. Both are metered on what they take. The difference is where the multicast stops: at the fabric's edge for the first, somewhere inside the peer's network for the second.



Figure 9 - two downstream shapes. On the left the consumer runs no multicast. On the right the peer's routers join the fabric's trees and replicate inside the peer's own network, and that network's receivers never touch the fabric's.

The right-hand shape is where the efficiency the objection says is impossible actually appears on someone else's network: a peer with a thousand receivers takes one copy across the border and replicates it itself. The left-hand shape is where most consumers start, and nothing stops one from becoming the other.

10. Peering points

A peer network takes the fabric at one of three layers, and *Sharding the object plane* gives the full treatment with the reference table; this section is the picture. The frames, the groups and the repair protocol are the same at every layer. What differs is the mechanism that carries them across the border.

At the **network layer**, your routers join the fabric's trees. A BGP session at the peering point exchanges nothing but source prefixes in the multicast address family, your receivers join the fabric's sources and groups, and one tree spans both networks. This is the inter-domain multicast practice the IETF documents, configured on routers you already own. At the **tunnel layer**, the same session and the same joins run inside a GRE tunnel, for a peering point that is not multicast-capable; where your network runs no multicast at all, an Automatic Multicast Tunneling gateway can request groups from a relay at the fabric's edge over plain unicast. At the **application layer**, you run the fabric's own open components beside your border: the delivery side takes the trees as a subscriber would, over a lossless connection from the fabric's edge, and the ingress side re-emits them under your own sources into your own trees, with a repair cache beside it. This is peering through the delivery wire by proxy, and it needs no multicast at the border at all.

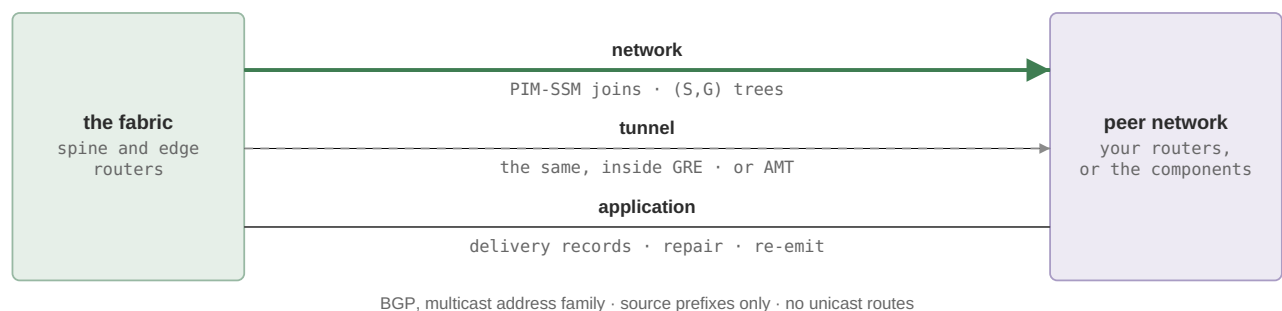


Figure 10 - three layers to take the fabric at. Your routers join the fabric's trees, or a tunnel carries the join, or the open components run on your side and re-emit. The frames are the same at each.

Layer	You run	Multicast at the border	Trees inside your network rooted at
network	PIM-SSM and MP-BGP on border routers	yes	the fabric's sources
tunnel	a GRE endpoint, or an AMT gateway	encapsulated	the fabric's sources
application	the open delivery and ingress stack	none	your own sources

1. A peer may start at one layer and move to another; nothing in the fabric changes when it does.

11. The public interoperability seams

Everywhere a party other than the operator touches the fabric, the thing it touches is an open document. A sender submits in Bitcoin's published extended transaction format, or in the published object format. A frame on the wire is one of a family of published frame formats, with a published rule for splitting a large one and a published rule for packing small ones. Groups are a published address plan. Repair is a published request and response. Discovery of groups and sources is a published announcement. Delivery to a consumer is a published set of lane formats. Peering is the IETF's published practice over the IETF's published protocols. Anyone can implement any seam from the document alone, and the open-source components exist as one implementation of each, not as the definition.

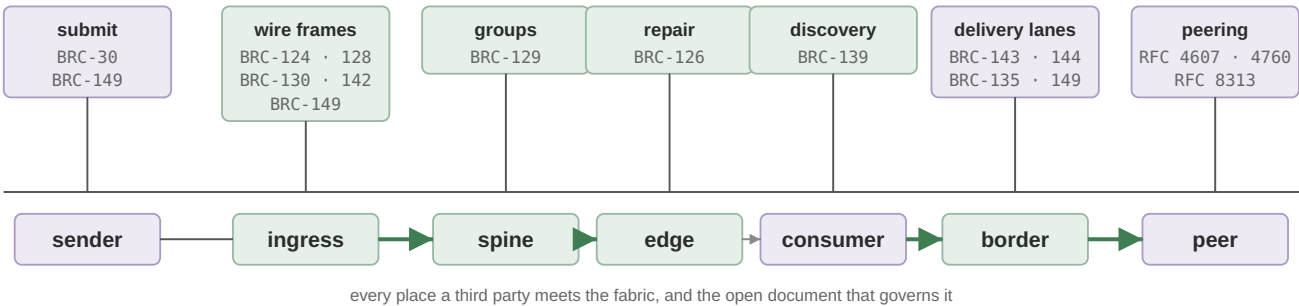


Figure 11 - the seam map. Each place a third party meets the fabric, and the open document that governs it. There is no seam without one.

This matters to the objection in a specific way. A critic who believes the network depends on something only one operator can run has not found it in the standards, because it is not there. The routing plane is standard, the formats are published, and the components are open. What any operator runs is a particular fabric built from them, the way an ISP operates a particular network built from BGP and OSPF.

12. Loss on the Internet, and why it does not matter

Tunnels across the Internet drop packets. So do cables, less often. A multicast tree has no acknowledgments, so a frame lost at a branch is lost to every receiver below that branch and nobody upstream knows. The fabric is built on the assumption that this happens constantly, and it notices.

Every frame carries the flow key and sequence number the ingress stamped on it. A receiver that sees sequence 3 and then sequence 6 knows at that instant that 4 and 5 are missing, without a timer, and asks a nearby cache for exactly those two by name. The cache resends them, or says at once that it does not have them, in which case the receiver asks the next cache. A cache that is missing frames lost before they reached its region asks one cache upstream, once, and re-caches what it gets. A resend is the same numbered frame again, so it can never be mistaken for a new one. The protocol is published as BRC-126, and *Loss on the Internet, and the network that notices* gives it the full treatment. For this paper one point carries: the repair protocol does not know whether the frame it recovers was lost on a tunnel or on a cable, and a consumer on a unicast lane never sees it run.

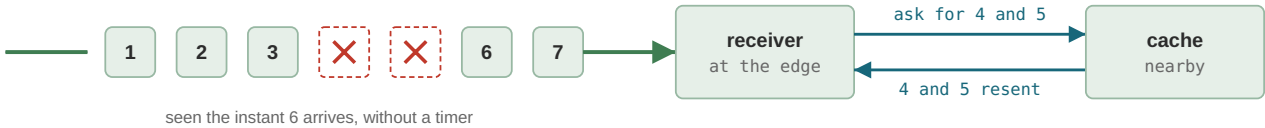


Figure 12 - a gap is seen, not guessed. The receiver notices 4 and 5 are missing the moment 6 arrives, asks a cache for them by number, and has them before the consumer's stream is written.

13. What the overlay costs, honestly

Native multicast on a shared segment sends one copy into the segment and every router on it receives it. A tree built over point-to-point tunnels cannot do that: a spine router with eight downstream tunnels sends eight copies, one into each, and its uplink carries all eight. That is the cost of the overlay form, and there is a second: each copy carries 52 bytes of encapsulation, and the path's usable size shrinks by the same amount. The case for the overlay is not that it is free. It is that the shape is right, and the cost is paid at a few branch points instead of at every publisher.

Take a worked example with stated numbers. A region has eight edges, each with a hundred subscribers to one group, and one spine router feeds them over tunnels. A publisher sending to all eight hundred by unicast sends eight hundred copies from one uplink. The fabric sends one copy to the spine, the spine

sends eight, and each edge writes a hundred unicast streams from a link built for exactly that. The publisher's cost went from eight hundred to one; the spine's cost is eight, and would be one on a native segment; the edges' cost is the same in either design, because delivery to a consumer is unicast by choice (\$8).

Design	Publisher sends	Spine link carries	Encapsulation per frame	Usable payload on a 1500-byte path
unicast to each receiver	800	800	0	1500 less headers
fabric over tunnels (this paper)	1	8	52 bytes	1448 less headers
fabric over native links	1	1	0	1500 less headers, 9000 on jumbo

1. Eight edges of one hundred subscribers each, one group, one spine router, all figures stated. "Less headers" is the IPv6, UDP and frame header cost that the companion paper on frames walks through.

The overlay keeps the one-send property that matters and pays for it at the spine, where it can be bought down link by link. Where the tunnels run over a path that is lossy or thin, the spine's eight copies cost eight times what one would; that is the honest case for upgrading a spine link, and \$15 is about doing so without touching anything else.

14. Why this multicast scales where the MBone did not

The critic's second objection is the stronger one. Even granting \$3, global multicast was tried, and it did not survive. The reasons are known. Any host could send to any group, so a group was a flood waiting to happen. Nobody paid for delivery, so no operator had a reason to carry it. A receiver had no way to tell a valid packet from a forged one, so filtering had to happen in the network, where it cannot scale. Routing state grew with every sender and every group. And there was no admission control at all: the network's only defense was to stop forwarding.

What rides on this fabric answers each of those, and most of the answers come from Bitcoin rather than from the network. They are the reason the design is a transaction network with multicast under it, and not a multicast network that happens to carry transactions.

Every frame cost its sender money. The payload is a Bitcoin transaction, or an overlay object built around one. A transaction carries a fee the sender pays a miner, and a valid signature over outputs the sender actually owns. There is no free way to make one, and a flood of them is a flood of fees. An invalid transaction is dropped at the door for the price of one signature check, which is the cheapest filter a network can have: the sender pays to produce what the receiver checks for almost nothing. Overlay objects carry the same skin in the game, because every one spends an output of a real transaction, as *The data carrier* walks through.

Every object proves itself. A frame does not ask to be trusted. A transaction is checked by its signature; an overlay object travels with its Merkle proofs and is verified by every receiver against block headers the receiver obtained for itself. The network is trusted to deliver and never to attest, which means nothing in the network has to inspect content to keep it honest, and a tampered or forged object fails verification wherever it lands. *The overlay object plane* and *The data carrier* give the mechanism. The consequence for scale is that validation is done once, at the receiver, by the party that cares, and the fabric's routers do nothing but forward.

Delivery is paid for by the byte. Admission is free and delivery is the product, priced on what reaches a consumer, as *The cost of being reachable* sets out. Every link in the fabric therefore has a revenue line behind it: a region with more subscribers earns more, and that is what buys the cross-connect or the jumbo link of \$15. The MBone had volunteers; this fabric has customers, and an operator who peers with it is one too.

Admission is bounded by proof, by tier and by budget. Three gates sit at the door and none of them needs cross-domain coordination. A block announcement must carry a header whose proof of work checks, which costs the forger real work and the checker one hash; the same check runs again at every edge, so an announcement that arrived from a peer network is verified identically without any shared registry. Subtree and block publication is a miner-tier right, exercised only through a provisioned tunnel, so the classes that would amplify most are the classes nobody can reach from the open Internet. Open publication of objects is budgeted per source and for the plane as a whole at the door, and again per consumer at the edge, so a flood spread across many doors is still bounded where it converges. None of these inspects content; each prices or bounds the right to emit.

No one can send to a group. Source-specific multicast means a receiver joins a named source, and only that source's traffic arrives on its tree. The any-source injection that made the MBone's groups a flood risk does not exist here; *One source, not any source* is about that choice.

The front door is wide, and stateless. Ingress is reached by the nearest-door routing the Internet already provides, every door runs the same stamp-and-send and holds no state about any sender, and adding doors adds capacity. There is no single front gate to congest and no door that has to know about another; the public rule for deriving a group, which *Shard bit extraction* gives, is what lets every door agree without talking.

State is bounded by design. A receiver joins the groups it wants and nothing else. Groups are a fixed, published plan, not one per sender or one per application, so routing state is set by how many groups the plan has and how many sources feed them, and not by how many parties use the network. *Sharding the object plane* shows how the same property holds for an unbounded topic space.

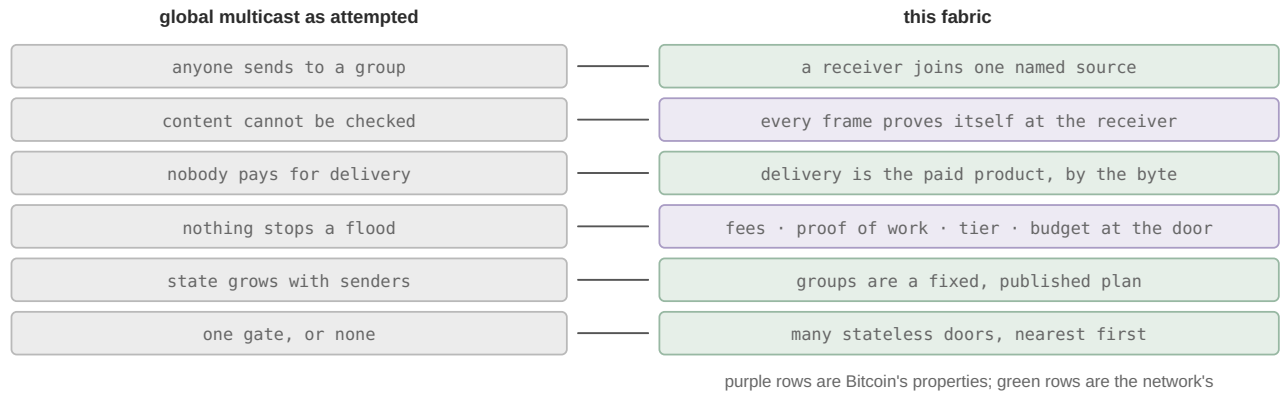


Figure 14 - what the earlier attempts lacked, and what sits in its place. Two of the six answers are Bitcoin's; the rest are the network's, and the network's would not hold without the two.

Property	Where it comes from	What it buys at scale
skin in the game	the transaction's fee and signature	a flood costs the flooders; invalid frames cost one check
self-proving content	BEEF and SPV against the receiver's headers	routers never inspect; validation is done once, by the receiver
paid delivery	metered bytes to each consumer	every link has revenue behind it
bounded admission	proof of work, miner tier, budgets	amplifying classes are gated; open classes are capped
one source per tree	source-specific multicast	no any-source flood; trees are receiver-pulled
wide stateless ingress	nearest-door routing, a public group rule	capacity grows by adding doors; doors never coordinate
fixed group plan	the published address plan	routing state does not grow with users

1. Skin in the game, self-proving content and proof of work are Bitcoin's properties; the fabric inherits them by carrying nothing else.

The fabric scales because what it carries is already expensive to forge, cheap to verify and paid for on delivery; the multicast only has to forward it.

15. The migration path

Every link in the fabric can be replaced in place with a better one, and the network above it does not notice. A tunnel across the Internet becomes a cross-connect at a facility where both routers have a presence. A 1500-byte cross-connect becomes a jumbo-frame link, and the fabric's largest objects cross it in a sixth of the datagrams. A GRE peering with a network that has since enabled multicast at the exchange becomes a native PIM-SSM peering. An AMT gateway in a network that has since deployed multicast becomes a plain join.

Three properties of the design make this true. The groups are global and static, published in BRC-129, so a receiver's join names the same group before and after a link changes. A source is a routed prefix, so a tree finds its way to it by whatever path the routing table now holds, tunnel or cable. And repair is keyed to the stamp the ingress put on the frame, so a frame recovered after a link change is the same frame as before it. The frames, the groups, the joins, the repair and the consumer's lanes are unchanged at every step; only the link is different, and only its properties, capacity, loss and frame size, improve.

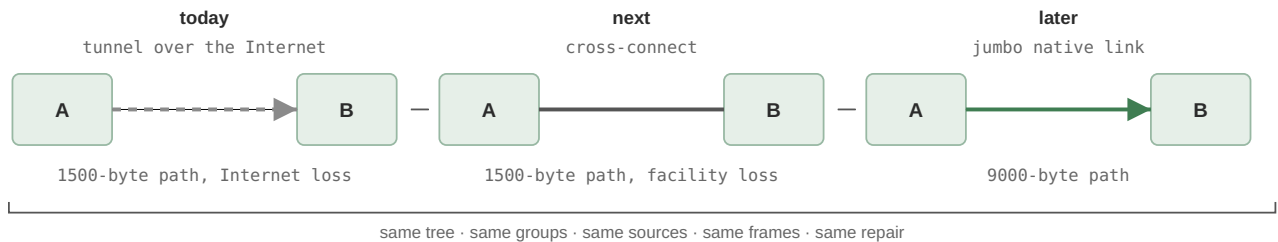


Figure 15 - one link, upgraded twice. The tree, the groups, the sources, the frames and the repair protocol are identical in all three; the link's capacity, loss and frame size are what changed.

This is the answer to the objection in its final form. The fabric does not wait for the Internet to do multicast, and it does not need the Internet to do multicast. It runs today over the Internet as it is, over consumer tunnels of the kind the public user guide describes, with the efficiency §13 states and no more. Where a better link exists, it takes it, and the design is built so that taking it changes nothing else. A critic who says the Internet will never do multicast may well be right, and the fabric will be no worse for it.

16. Assumptions and limits

Claim	Basis	Note
The Internet forwards unicast only between parties	stated	the objection, accepted
A GRE tunnel carries multicast between two routers	verified	RFC 7676; RFC 8313 §3.2 names the case
Encapsulation costs 52 bytes on IPv6	verified	40-byte outer IPv6, 4-byte GRE, 8-byte option as deployed; RFC 7676, RFC 8200
Frame header costs on the inner path	verified	BRC-124 (92 bytes), BRC-130 (104 bytes)
Groups are global and static	verified	BRC-129
Repair is keyed to the ingress stamp	verified	BRC-124 flow key and sequence number; BRC-126
One copy per downstream tunnel at a branch	structural	follows from point-to-point links
Eight edges, one hundred subscribers each	stated	the worked example of §13 only
A jumbo link carries about a sixth of the datagrams	structural	8848 against 1348 bytes of payload per fragment, BRC-130 at 9000 and 1500
Block announcements are gated by header proof of work	verified	the open-source fabric components, both ingress and edge
Open object publication is budgeted at door and edge	verified	<i>The overlay object plane</i> §4
AMT exists in router operating systems and the Linux kernel	verified	RFC 7450; the kernel's AMT driver

Limits, stated plainly:

1. **The overlay pays one copy per downstream tunnel at every branch.** Where a native multicast segment is available, it is strictly better on that hop, and the fabric takes it when it is.
2. **Tunnels shrink the path's usable frame size.** The fabric's largest objects cross a 1500-byte tunnel in many datagrams. The arithmetic and the remedy are in *Frames on a 1500-byte Internet*.
3. **Automatic Multicast Tunneling has few relays in public service.** The fabric offers it as one of three peering layers and depends on none of them.
4. **A consumer's tunnel is unicast by design.** The consumer's own network gains none of the fabric's replication. A consumer that wants that becomes a peer at one of the layers of §10.
5. **The peering contract is open** beyond the commitments stated in *Sharding the object plane*: built from standard tools and the published formats, and metered as a consumer.
6. **The worked example of §13 uses stated counts to show a shape.** Real fan-outs vary by region and by group, and the shape does not.

References

BRC-30: Transaction Extended Format (EF): <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0030.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-128: Multicast Extended Transaction Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0128.md>
BRC-129: IPv6 Multicast Group Address Assignments: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0129.md>
BRC-130: Multicast Transaction Frame Fragmentation: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0130.md>
BRC-135: Block Header Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0135.md>
BRC-139: Multicast Shard Manifest Announcement Protocol: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0139.md>
BRC-142: Multicast Transaction Bundle Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0142.md>
BRC-143: Subtree Data Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0143.md>
BRC-144: Block Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0144.md>
BRC-149: Multicast BEEF Object Frame Format: <https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0149.md>
IETF, RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for IPv6: <https://www.rfc-editor.org/rfc/rfc3810>
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 7450: Automatic Multicast Tunneling: <https://www.rfc-editor.org/rfc/rfc7450>
IETF, RFC 7676: IPv6 Support for Generic Routing Encapsulation (GRE): <https://www.rfc-editor.org/rfc/rfc7676>
IETF, RFC 7761: Protocol Independent Multicast - Sparse Mode (PIM-SM): <https://www.rfc-editor.org/rfc/rfc7761>
IETF, RFC 8200: Internet Protocol, Version 6 (IPv6) Specification: <https://www.rfc-editor.org/rfc/rfc8200>
IETF, RFC 8313 (BCP 213): Use of Multicast across Inter-Domain Peering Points: <https://www.rfc-editor.org/rfc/rfc8313>
IETF, RFC 8815 (BCP 229): Deprecating Any-Source Multicast (ASM) for Interdomain Multicast: <https://www.rfc-editor.org/rfc/rfc8815>
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)): <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/drivers/net/amt.c>
Pushed delivery into a Teranode cluster (Lightweb Inc., Paper 001): <https://1bsv.net/papers/pushed-delivery-teranode.pdf>
The cost of being reachable (Lightweb Inc., Paper 003): <https://1bsv.net/papers/ingress-economics.pdf>
Landing in Kubernetes (Lightweb Inc., Paper 004): <https://1bsv.net/papers/kubernetes-landing.pdf>
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>
Sharding the object plane (Lightweb Inc., Paper 010): <https://1bsv.net/papers/object-plane-sharding.pdf>
The data carrier (Lightweb Inc., Paper 011): <https://1bsv.net/papers/data-carrier.pdf>
1BSV User Guide (Lightweb Inc.): <https://1bsv.net/guide.html>

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.