DESIGN · FRAMES · MTU

Frames on a 1500-byte Internet: split the large, pack the small, and size to the smallest link

A multicast frame travels in one datagram, and the common path across the Internet carries 1500 bytes of it, 52 fewer inside a tunnel. This paper does the arithmetic of what fits: where the bytes go, the ceiling above which an object is silently lost, how an object larger than the ceiling is split into fragments that are frames in their own right and repaired one at a time, how many small transactions are packed into one datagram to cut packets per second, why a frame must be sized to the smallest link on its path and not to the sender's interface, and what a jumbo-frame link buys. It closes with what a consumer sees, which is whole objects and none of this.

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

Paper 013 · **Version** 01

Audience: protocol implementers, and network engineers sizing a link or a tunnel that a fabric crosses.

We = Lightweb Inc., author of this architecture and operator of the first fabric built on it; nothing here depends on that operator. **You** = an implementer, or the operator of a link the fabric crosses. Every header width is the published specification's; every path size is stated.

Every figure uses the same colors: gray for the Internet and for outer headers, green for the fabric's own headers and trees, purple for the payload a consumer finally receives. A teal line is a repair; a red cross is a loss.

The short version

- **One frame, one datagram.** A datagram is the unit multicast replicates and the unit a sequence number names, so a frame never spans two.
- **140 bytes of every datagram are headers.** IPv6 takes 40, UDP 8, the frame header 92. A tunnel adds 52 outside. On a 1500-byte path that leaves 1360 bytes for an object, or 1308 inside a tunnel.
- **Above the ceiling an object vanishes without a trace.** No error at the sender, none at the receiver, no counter anywhere. Fragmentation is not an optimization; it is the difference between delivery and silence.
- **A fragment is a frame.** Each slice carries its own sequence number, so a lost slice is noticed and repaired exactly as a lost frame is, and the receiver rebuilds the object against its hash.
- **Many small transactions ride one datagram.** A bundle packs transactions of one group into a single frame, so a thousand small transactions cross a 1500-byte fabric in about 200 packets, and a 9000-byte fabric in about 29.
- **Size to the smallest link on the path.** A frame that fits the sender's interface and not a tunnel on the way dies at the tunnel. The smallest link sets the fragment size for every tree that crosses it.
- **A jumbo link carries an object in a sixth of the datagrams.** At 9000 bytes a fragment holds 8848 bytes against 1348, so a 1 MB object takes 114 fragments instead of 742, with a sixth of the sequence numbers and a sixth of the repairs.

1. One frame, one datagram

The fabric carries every transaction, subtree, block and overlay object as a frame inside a single UDP datagram. It does not stream. The reason is what multicast replicates: a router copies a datagram into each downstream link, and that is the only unit it knows. A frame that spanned datagrams would be replicated as pieces that no router could relate, and a sequence number that named a stream position rather than a datagram could not tell a receiver which datagram it lacked. So a frame is a datagram, every datagram carries one sequence number in one flow, and a lost datagram is a lost sequence number, noticed at the next.

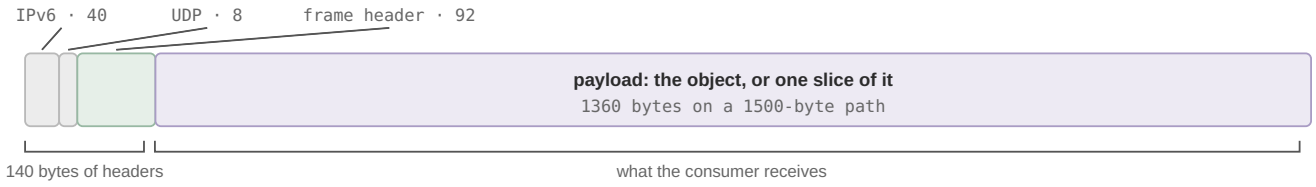

Figure 1 - a frame on the wire. Headers are fixed; the payload takes whatever the path leaves.

2. Where the bytes go

The widths are fixed by the published formats. An IPv6 header is 40 bytes, a UDP header 8, and the multicast frame header 92 (BRC-124), which carries the object's hash, the flow key and sequence number the ingress stamped, and the subtree the frame belongs to. A fragment header (BRC-130) is 104, the same 92 with twelve bytes of slice bookkeeping. A bundle header (BRC-142) is 66. A GRE tunnel over IPv6 adds 52 bytes outside all of that: 40 for the outer IPv6 header, 4 for GRE and 8 for the encapsulation-limit option that IPv6 tunnels carry.

Path	Bytes a datagram may hold	Whole object (BRC-124)	Per fragment (BRC-130)	Bundle members (BRC-142)
IPv6 minimum	1280	1140	1128	1166
common Internet path	**1500**	**1360**	**1348**	**1386**
the same, inside a tunnel	1448	1308	1296	1334
jumbo link	**9000**	**8860**	**8848**	**8886**
jumbo link, inside a tunnel	8948	8808	8796	8834

1. Whole object = path less 140; per fragment = path less 152; bundle members = path less 114. A tunnel takes 52 from the path first.

Two numbers from that table carry the rest of the paper. The unfragmented ceiling on the common path is 1360 bytes, and a fragment on the same path holds 1348. Almost every transaction fits under the first. No subtree, block or overlay object of any size does.

3. The cliff

A datagram larger than a link can carry is dropped by that link. That is the correct behavior for a router, and it is silent: IPv6 routers do not fragment on behalf of a sender, and a multicast datagram has no one to return an error to. The sender's own link accepted it, the receiver never saw it, and no counter on either side moved. A fabric that did not fragment would not "send objects whole"; it would have a hard ceiling at 1360 bytes on the common path, under which every transaction arrives and above which every subtree and block disappears, with the transaction lane looking perfectly healthy throughout.

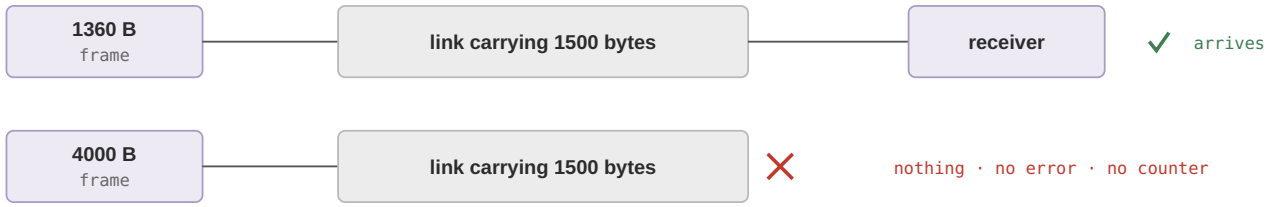

Figure 2 - the cliff. Under the ceiling a frame arrives; over it the frame is dropped at the first link that cannot carry it, and nothing reports the loss.

The diagnosis, when it happens, is always the same: everything below some size works and everything above it does not, with a healthy connection at the submitter and nothing in any log. The size is the tell. If the threshold matches the path less 140 for a plausible path, the frames are being sent whole over a link that cannot carry them.

4. Fragmentation

An object larger than the ceiling is cut into equal slices at the ingress, each slice becomes a datagram with a fragment header, and the receiver reassembles them. The header carries the whole object's hash, the slice's index and the total count, and the object's full length, so a receiver can allocate the buffer on the first slice it sees, in any order, and place every later slice by index. When every slice is present the receiver has the object and may check it against the hash in the header, which is the object's own name. A slice that does not fit the grid the others imply is rejected with the whole object, rather than concatenated into a payload of the wrong length.

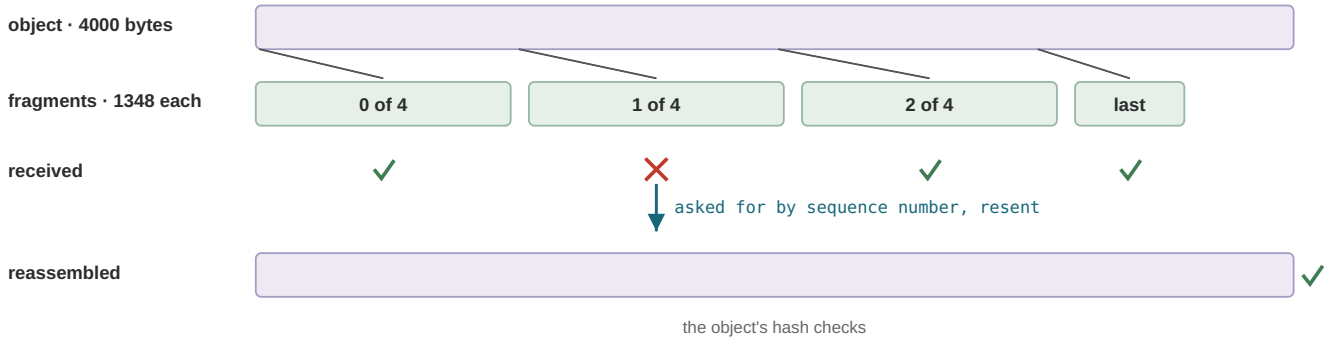

Figure 3 - an object cut into fragments, one lost in transit, repaired by the ordinary mechanism, and reassembled against its hash.

The count of fragments is the object's length divided by the per-fragment payload, rounded up, and the per-fragment payload is the path less 152. Because the index and the count are sixteen-bit fields, one object may have at most 65,535 fragments, which bounds a single object at about 88 MB on a 1500-byte path and about 580 MB on a 9000-byte one.

Object	Fragments at 1500	Inside a tunnel at 1500	Fragments at 9000	Inside a tunnel at 9000
100 kB	75	78	12	12
1 MB	**742**	772	**114**	114
10 MB	7,419	7,717	1,131	1,137

1. Payload per fragment 1348, 1296, 8848 and 8796 bytes respectively, from the table in §2; counts rounded up.

5. Why fragments are frames

The fragment header is laid out so that its first 92 bytes read as an ordinary frame header, and each fragment is stamped with its own sequence number in the same flow as every other frame from that ingress. That one decision is what keeps the rest of the fabric unchanged. A receiver's gap detector sees fragments as frames and notices a missing one at the next. A cache stores fragments as frames and answers a request for one by its number. The repair protocol (BRC-126) needs no rule for fragments because it never sees one; it sees sequence numbers. And a lost slice costs the receiver one request for one datagram, never the object.

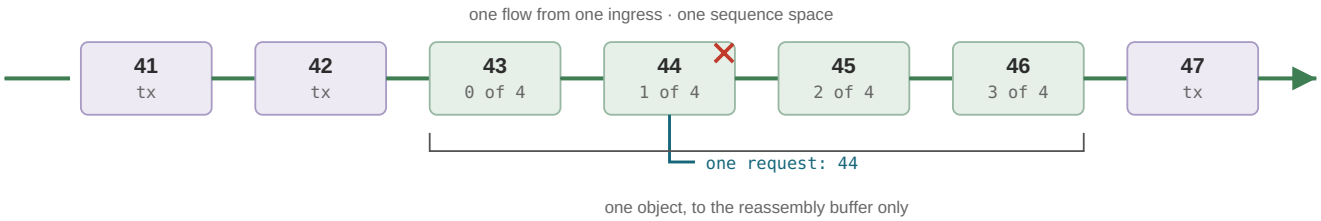

Figure 4 - fragments share the flow's sequence space with whole frames. The gap detector and the cache see sequence numbers; only the receiver's reassembly buffer knows that 43 to 46 are one object.

This is also what makes fragmentation safe across a lossy path. On a link losing one datagram in a thousand, a 100 kB object on the common path arrives with one or more slices missing about seven times in a hundred, and each such case is closed by one small request. The object is late by one round trip to a cache, not lost. The companion paper on repair gives the mechanism; here the point is only that fragments made no new work for it.

6. Bundling, the inverse

Most transactions are small, a few hundred bytes, and the cost that dominates a forwarding plane is packets per second, not bytes. A fabric that sent each small transaction as its own datagram would spend most of its forwarding budget on headers. A bundle (BRC-142) packs many transactions of one group into one datagram, each prefixed by its length, under a header that carries one flow key and one sequence number for the whole. The bundle is a frame: it is numbered in the same sequence space, cached and repaired as one unit, and never larger than the path. Packing waits at most a short, bounded time, so a transaction that arrives alone is not held for company.



Figure 5 - a bundle. Five small transactions of one group become one numbered frame. The edge unpacks it before delivery, so the consumer never sees a bundle.

How many fit depends on the path and the transactions. With transactions of 250 bytes and two bytes of length prefix each, a 1500-byte path holds five per bundle and a 9000-byte path thirty-five. A thousand such transactions cross the fabric in about 200 datagrams on the first and about 29 on the second, against a thousand without bundling. The gain is a property of the traffic: transactions of one group arriving close together pack well, and a trickle packs not at all, which is why packing is bounded in time and never required.

Path	Members per bundle	Datagrams per 1,000 transactions	Against unbundled
1500	5	**200**	1,000
9000	35	**29**	1,000

1. Transactions of 250 bytes, stated; member capacity 1386 and 8886 bytes from §2; two bytes of length prefix per member.

Fragmentation and bundling are mutually exclusive by construction. A transaction that would not fit a bundle is sent alone, and one that would not fit a datagram is fragmented; a bundle is never fragmented.

7. Size to the smallest link on the path

A fabric router replicates a datagram into every downstream link, and a datagram sized to the sender's interface may be too large for one of them. A sender on a 1500-byte interface feeding a tree that crosses a tunnel sends 1360-byte objects that fit its own link and die at the tunnel, 52 bytes too long, with the silence of §3. The rule that follows is simple and easy to get wrong: the fragment size for a tree is set by the smallest link anywhere on that tree, not by anything the sender can see locally.

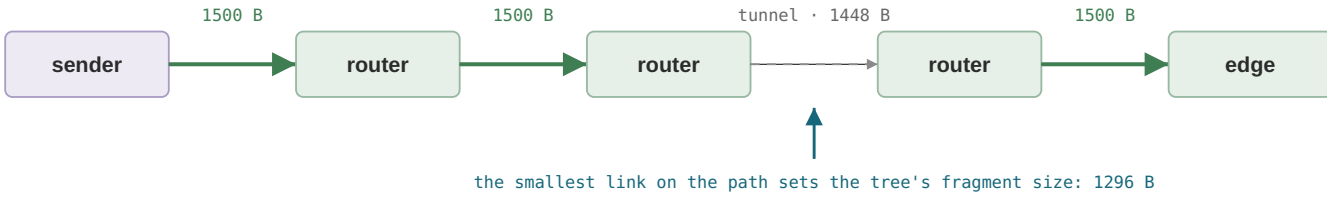

Figure 6 - one tree, four links, one tunnel. The tunnel is the smallest link on the path, and every fragment on this tree is sized to it, whatever the sender's own interface carries.

Two practices follow. A configured size is a claim, and a tunnel's claim is often wrong, because the kernel sizes a tunnel from the interface it was created on and not from the path the outer packets actually take; the true size is measured by sending probes of increasing length with fragmentation forbidden and noting where they stop. And the measurement belongs to the whole fabric, not to a node: a link an origin cannot see, such as an interconnect between regions, still bounds every tree that crosses it, so the fabric's fragment size is the minimum over all of its links and is pinned fabric-wide.

8. What jumbo frames buy

A jumbo link carries 9000 bytes, and a fragment on it holds 8848, about six and a half times the 1348 of the common path. For an object that means a sixth of the datagrams, a sixth of the sequence numbers, and on a path with a given loss rate a sixth of the repairs; for the small-transaction stream it means seven times as many transactions per bundle. The gain is largest where the trees are largest, which is the spine of *Multicast over a unicast Internet*: the links between regions carry every tree that crosses them, so a jumbo link there raises the ceiling for the whole fabric below it. A jumbo cross-connect at a facility where two routers meet is the cheapest such link, and it replaces a tunnel in place.

What changes at 9000 bytes	1500	9000
payload per fragment	1,348	8,848
fragments for a 1 MB object	742	114
repairs per 1 MB object at one loss in 1,000	0.74	0.11
members per bundle, 250-byte transactions	5	35
largest single object	about 88 MB	about 580 MB

1. Repairs = fragments times the stated loss rate; the ceiling on object size follows from the sixteen-bit fragment count.

A jumbo link does not change a format, a group or a protocol; it changes how many datagrams an object needs, and that is the number everything else scales with.

9. Mixed paths

A fabric is rarely one size. A jumbo spine with 1500-byte tunnels to its edges fragments to the tunnels, because they are on the path, and gains from the spine only its lower loss and its bundles' density on the long links. The rule of §7 is what makes the mix safe: the smallest link anywhere on a tree sets the size for the whole tree, so adding a large link never breaks anything, and replacing a small one raises the size for every tree that crossed it. That is the migration path of *Multicast over a unicast Internet* seen from the frame's side: each tunnel that becomes a cross-connect, and each cross-connect that becomes a jumbo link, removes one small link from some set of trees, and the fragment size of those trees rises to the next smallest.

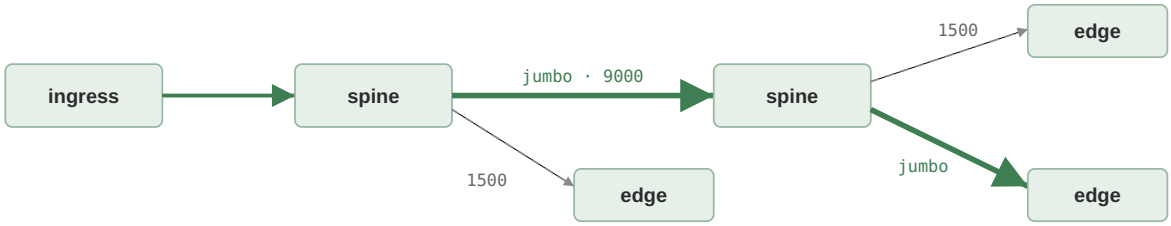

Figure 7 - a mixed fabric. The spine link is jumbo; two edges hang off tunnels and one off a jumbo link. Each tree is sized to its own smallest link.

10. What the consumer sees

None of this reaches a consumer. The edge reassembles fragments, unpacks bundles, completes repairs, and writes whole objects to a unicast stream inside the consumer's tunnel, as *Pushed delivery into a Teranode cluster* and *Landing in Kubernetes* specify. A consumer receives transactions, subtrees, blocks and overlay objects as complete objects on an ordered stream, and TCP inside the tunnel handles whatever the tunnel's own path does to the bytes. The fabric's frame sizes, fragment counts and bundle densities are the fabric's concern, and they can change, link by link, without a consumer noticing.

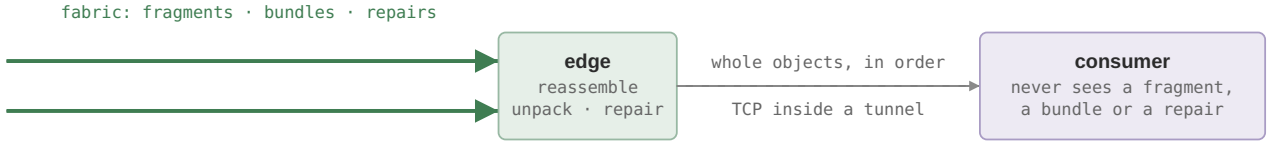

Figure 8 - the edge resolves every frame-level mechanism before the consumer's stream is written. A consumer never sees a fragment, a bundle or a repair.

11. Why it scales

Every mechanism in this paper keeps the property that *Multicast over a unicast Internet* rests on: a frame is numbered once, at the door, and everything after that works on the number. Fragments inherit the stamp, so splitting an object makes no new work for the gap detector or the cache. Bundles carry one stamp for many transactions, so packing cuts the packets the forwarding plane handles without adding a case the repair protocol has to know. And the fragment size is a property of a tree's links, not of its traffic, so a fabric's largest objects scale with its links and nothing else. What a jumbo link buys is fewer datagrams per object, and datagrams per object is the one number that every cost in the fabric scales with.

12. Assumptions and limits

Claim	Basis	Note
IPv6 header 40, UDP 8, GRE over IPv6 52 with the option	**verified**	RFC 8200, RFC 768, RFC 7676
Frame header 92, fragment header 104, bundle header 66	**verified**	BRC-124, BRC-130, BRC-142
Fragment index and count are sixteen-bit	**verified**	BRC-130
Bundle members carry a two-byte length prefix	**verified**	BRC-142
Routers do not fragment IPv6 on a sender's behalf	**verified**	RFC 8200 §5
Path sizes 1280, 1500, 9000	**stated**	the IPv6 minimum, the common Internet path, a jumbo link
Transactions of 250 bytes	**stated**	the bundling example only
One loss in a thousand	**stated**	the repair example only
Fragment counts, bundle members, repairs per object	**structural**	arithmetic on the rows above

Limits, stated plainly:

1. **Bundling gains depend on traffic shape.** Transactions of one group arriving close together pack well; a trickle does not pack, and the bounded wait means it is not made to.
2. **The repair example assumes independent losses.** A burst loss takes several consecutive fragments at once and costs one request naming several numbers, which is cheaper per fragment, not dearer.
3. **A tunnel's configured size is not its real size.** The paper's tunnel rows assume the 52-byte overhead and a correctly measured path; a tunnel that rides another tunnel is smaller still, and only a probe finds out.
4. **The largest-object ceiling is per object, not per block.** A block larger than the ceiling on a path is carried as its subtrees, which the published formats already separate.
5. **Jumbo links are available where both ends are one operator's or at a facility that offers them.** The common Internet path stays 1500, and the paper's design holds there.

References

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-130: Multicast Transaction Frame Fragmentation: https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0130.md

BRC-142: Multicast Transaction Bundle Frame Format: https://github.com/bsv-blockchain/BRCs/blob/master/transactions/0142.md

IETF, *RFC 768: User Datagram Protocol*: https://www.rfc-editor.org/rfc/rfc768

IETF, *RFC 7676: IPv6 Support for Generic Routing Encapsulation (GRE)*: https://www.rfc-editor.org/rfc/rfc7676

IETF, *RFC 8200: Internet Protocol, Version 6 (IPv6) Specification*: https://www.rfc-editor.org/rfc/rfc8200

IETF, *RFC 8201: Path MTU Discovery for IP version 6*: https://www.rfc-editor.org/rfc/rfc8201

Pushed delivery into a Teranode cluster (Lightweb Inc., Paper 001): https://1bsv.net/papers/pushed-delivery-teranode.pdf

Landing in Kubernetes (Lightweb Inc., Paper 004): https://1bsv.net/papers/kubernetes-landing.pdf

Multicast over a unicast Internet (Lightweb Inc., Paper 012): https://1bsv.net/papers/multicast-over-unicast-internet.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.

© 2026 Lightweb Inc. All rights reserved. 1BSV and Unbounded Solutions for a Small World are trademarks of Lightweb Inc. This paper describes an architecture in rapid development: the design, its interfaces and this document are subject to change without notice, and later versions supersede this one. Nothing in this paper is a commitment to ship any feature, an offer of service, or professional advice.