Multicast and Socket Attributes in Unicon

Jafar Al-Gharaibeh

Unicon Technical Report: 28

August 2026

Abstract

Unicon could open TCP and UDP sockets, but applications had no way to join a multicast group, pick an interface, or set TTL from the language. This report describes trailing name=value attributes on open(), the same style as SSL and graphics windows, plus Attrib() after open() for mutation and f["name"] / key(f) for status. Binding a UDP socket to a group address joins that group; source@group or source= selects source-specific multicast. Defaults cover reuse, all-interface IPv4 joins, and outbound interface selection so the common cases need no attributes at all.

Keywords: Unicon, multicast, SSM, UDP, socket attributes, Attrib, runtime, technical report.

Unicon Project
http://unicon.org
University of Idaho
Department of Computer Science
Moscow, ID, 83844, USA

1 1. Introduction

This report describes IP multicast [1] and the socket-attribute facility it sits on. Trailing arguments to `open()` on network modes are `name=value` strings applied with `setsockopt(2)`. After `open()`, `Attrib()` uses the same names. The language-facing reference lives in *Programming with Unicon* [2] (Chapter 5, "Multicast and Socket Attributes") and the language reference (`open` trailing attributes, `Attrib`, subscript peek, `key()`). Automated tests live in `tests/posix/mcast.icn` and `tests/posix/sockpeek.icn`.

2 2. Motivation

Unicon's `open()` already constructed TCP and UDP handles. Multicast – any-source (ASM) and source-specific (SSM) [3] – is ordinary UDP plus membership and hop-limit options. Without those options, a Unicon program could not join a multicast group like `239.1.1.1`, restrict a join to one NIC, or send with a limited TTL like `ttl=3`. These options were not supported.

3 3. Design principles

Attributes are `name=value`, like graphics windows. Unknown names fail `open()` with error 1310 (`bad socket attribute`). Booleans are exactly `yes` or `no`, matching `verifyPeer`.

```
procedure main()
   if open(":5110", "nua4", "bogus=yes") then
      write("oops")
   else
      write("bogus: ", &errortext)
   if open(":5110", "nua4", "reuseaddr=yes") then
      write("oops")
   else
      write("numeric boolean: ", &errortext)
end
```

Output:

```
bogus: bad socket attribute
numeric boolean: bad socket attribute
```

DWIM for the common cases. Binding a UDP socket to a multicast group joins that group. Listeners default to `reuseaddr=yes`. No `iface=` means join every UP IPv4 interface and send out the first non-loopback IPv4 address (else loopback). Sending to `255.255.255.255` enables `SO_BROADCAST`. Explicit attributes always override.

`source@group`**.** The address form matches VLC, ffmpeg, and RFC 4607 `(S,G)` order. Empty either side is invalid.

`ttl` **and** `iface`**, not multicast-only names.** Hop limit and interface are used by unicast, multicast, and raw sockets [4]. `ttl=` sets both `IP_TTL` and `IP_MULTICAST_TTL` (or the IPv6 hop options). `iface=` accepts an IPv4 address, a numeric index, or a name (`en0`, `lo0`; Windows also maps `lo` / `lo0` to `loopback_0`).

Order matters. `iface=` before `join=` selects the interface used by that join. `Attrib()` applies a batch of assignments together for the same reason.

Cached listeners are aliases; attributes get a new socket. A second `open()` of the same listener address with no socket attributes returns another file for the cached fd. An open that supplies `join=` / `iface=` / and so on creates an independent socket so it cannot retune earlier aliases. Use `Attrib()` to change options on a live shared handle.

4 4. Opening a multicast socket

IP multicast uses UDP (modes `nu` / `nua`, optionally with `4` or `6` for family). A minimal receiver and sender:

```
procedure main()
   local f, r
```

```
   f := open("239.1.1.1:5000", "nua") | stop(&errortext)
   while r := receive(f) do
      write(r.addr, "  ", r.msg)
end
```

```
procedure main()
   local f, i
   f := open("239.1.1.1:5000", "nu") | stop(&errortext)
   every i := 1 to 5 do
      writes(f, "hello-" || i) & delay(500)
end
```

The receiver binds the group address, so the runtime joins that group (ASM) and enables port reuse. The sender's destination is the group; without `iface=`, outbound multicast uses the first non-loopback IPv4 address. Kernel multicast loopback stays on, so a local receiver still sees the packets.

The same exchange in one process over loopback (this is the shape of `tests/posix/mcast.icn`). `iface=127.0.0.1` keeps the datagram on loopback even when the host has no multicast route:

```
procedure main()
   local f, s, r
   f := open("239.42.42.99:5199", "nua", "iface=127.0.0.1") |
        stop(&errortext)
   s := open("239.42.42.99:5199", "nu", "iface=127.0.0.1") |
        stop(&errortext)
   writes(s, "hello-multicast")
   if *select(f, 2000) > 0 then {
      r := receive(f)
      write(r.msg)
      }
   else
```

```
      write("timeout")
   close(s)
   close(f)
end
```

Output:

```
hello-multicast
```

`r.addr` is the sender (host plus an ephemeral port).
A wildcard bind plus explicit joins is the other shape:

```
f := open(":5000", "nua",
          "iface=192.168.1.10",
          "join=239.1.1.1",
          "join=232.1.1.1,10.0.0.5")
```

`join=group` is ASM; `join=group,source` is SSM. Attributes may be repeated. Mixing ASM and SSM in one call is allowed. If `open()` has no `4` / `6` flag, a `join=` group address picks the family so an IPv4 join is not attempted on an IPv6 wildcard socket.

5 5. Source-specific multicast

Name the source before the group, or pass `source=` on a socket already bound to the group. Both IPv4 and IPv6 are supported. For IPv6 SSM, pass an explicit `iface=` (name or index) so the join uses a single interface:

```
f := open("192.168.1.1@239.1.1.1:5000", "nua")
f := open("239.1.1.1:5000", "nua", "source=192.168.1.1")
f := open("::1@ff3e::1:5000", "nua6", "iface=lo0")
f := open("ff3e::1:5000", "nua6", "iface=lo0", "source=::1")
```

On send or connect, `source@group:port` keeps only the group so the destination is the group address. The source is a receive-side filter, not a send address.

IPv6 SSM is a single `MCAST_JOIN_SOURCE_GROUP` / `MCAST_LEAVE_SOURCE_GROUP` with no multi-interface walk. Those walks have triggered macOS kernel panics in IPv6 source filter teardown. Already-a-member and already-gone are treated as success.

6 6. Attributes

Attribute	Role
`join=`	Add membership: `group` (ASM) or `group,source`
`leave=`	Drop membership; same forms as `join`. Pass
`source=`	SSM source for a group bind (repeatable). Set-only.
`iface=`	IPv4 address, index, or name. Restricts join and
`ttl=`	Unicast and multicast hop limits. `f["ttl"]` peeks
`mcastloop=yes|no`	Whether the host receives its own multicasts. Kernel
`reuseaddr=` / `reuseport=`	Bind reuse. Listeners default to `reuseaddr=yes`
`broadcast=`	`SO_BROADCAST`. Opening `255.255.255.255` sets it.
`rcvbuf=` / `sndbuf=`	Buffer sizes in bytes.

7 7. `Attrib()` and status peek

```
procedure main()
   local f, g
   f := open(":5110", "nua4") | stop(&errortext)
   Attrib(f, "ttl=4")
   write("ttl=", f["ttl"])
   write("mcastloop=", f["mcastloop"])
   if g := f["groups"] then
      every write(!g)
   else
      write("groups: unpopulated")
   close(f)
end
```

Output:

```
ttl=4
mcastloop=yes
groups: unpopulated
```

Assignments in one call are applied together, preserving `iface` then `join` order. Status is peeked with `f["name"]`. Unknown names raise

1. An unpopulated field fails. Boolean fields that answered

succeed with `"yes"` or `&null`. `key(f)` generates every answerable field. `f["*"]` snapshots those fields under one lock. Peeking a closed handle is error 174. `key(f)` that has not yet produced a name also raises 174 if the handle is already closed.

TCP, UDP, multicast, and raw sockets share one peek table (`sock_peek`). `join`, `leave`, and `source` are verbs, not peek fields. `proto` is stored at

`socket()` time on raw sockets [4]. `f["groups"]` is a list of joined groups (SSM as `source@group`); a single group is still a list of length 1.

Name	Value
`ttl`	Hop limit (integer), typically `1` .. `255`
`iface`	Dotted IPv4 address (`127.0.0.1`) or IPv6 ifindex (`1`)
`groups`	List of joined groups, e.g. `239.1.1.1`; SSM as
`mcastloop`	`"yes"` or `&null`
`reuseaddr` / `reuseport`	`"yes"` or `&null` (`reuseport` may be unpopulated)
`broadcast`	`"yes"` or `&null`
`rcvbuf` / `sndbuf`	Buffer sizes in bytes (integers)
`proto`	IP protocol number: `1` (ICMP), `2` (IGMP), `89`
`hdrincl`	`"yes"` or `&null`; typical on raw sockets

```
f := open(":5110", "nua4") | stop(&errortext)
Attrib(f, "ttl=4")
Attrib(f, "iface=127.0.0.1", "join=239.1.1.1")
Attrib(f, "iface=127.0.0.1", "join=239.1.1.2")
write("ttl=", f["ttl"])
every g := !f["groups"] do
   write("joined ", g)
every k := key(f) do
   write("field ", k)
close(f)
```

Membership changes still use assignment form:

```
Attrib(f, "join=239.1.1.2")
Attrib(f, "iface=127.0.0.1", "leave=239.1.1.1")
```

On some Linux/musl builds `getsockopt(IP_MULTICAST_IF)` does not recover the interface used at join time. Pass `iface=` on `leave=` so `IP_DROP_MEMBERSHIP` matches the join. Already-not-a-member is success; the runtime also retries `INADDR_ANY` and each local IPv4 address when a single DROP fails.

8 8. Listener cache

`sock_listen` still caches the first bind of an address so repeated `open()` of the same listener is cheap. That cache is a problem if a later `open(..., "join=...")` reused the fd and changed memberships for every live alias.

Rules:

- Cache hit, no socket attributes: another `File` for the same fd (owner refcount). Closing one alias leaves the others until the last handle is closed.
- Cache hit with attributes: release the pin and create an independent socket (`reuseaddr` makes the second bind succeed). `leave=` on that socket does not drop the cached alias's membership.
- Changing options on a shared handle: `Attrib()` on one of the aliases.

A cache-hit open that is not a pure alias only allows additive `join=` / `source=`; `leave=` and option changes on that path would retune every live `File`.

9 9. Runtime implementation

All of this lives in `src/runtime/rposix.r`, applied from `open()` in `fsys.r` and from `Attrib()` in `fmisc.r`.

Two passes. `reuseaddr` / `reuseport` must be set between `socket()` and `bind()`. Everything else runs after the socket is bound. `apply_sock_attrs()` takes a `prebind` flag.

Auto-join. After the post-bind pass, a UDP socket bound to a multicast group is joined unless an explicit `join=` took over. `source@group` or `source=` makes that join SSM.

IPv4 join without `iface=`. Walk `getifaddrs()` and `IP_ADD_MEMBERSHIP` on every UP IPv4 address so same-host and multi-homed receives work without naming a NIC. An explicit `iface=` joins only that one. Outbound multicast without `iface=` picks the first non-loopback IPv4 address via `IP_MULTICAST_IF`.

IPv6 ASM uses `IPV6_JOIN_GROUP` / `IPV6_LEAVE_GROUP` with an ifindex (zero means default). There is no all-iface walk like IPv4.

Errors. Unknown attributes and bad booleans are 1310. `setsockopt` failures fail `open()` / `Attrib()` with the system `&errortext`, except idempotent membership errors (already a member / already gone).

Mode `n` with `4` or `6` still selects the address family. `"nua4"` is useful when a join group must match an IPv4 wildcard.

10 10. Status and remaining work

The language surface described here is implemented: trailing attributes, implicit ASM/SSM joins, `Attrib()` join/leave, listener-cache isolation, IPv4 all-iface join, IPv6 SSM with an explicit interface, and the broadcast shortcut.

Membership query. `f["groups"]` is a list of groups joined through `open()` or `Attrib() join=` / `source=`. The kernel still has no portable membership dump; this is runtime-tracked state.

IPv6 all-interface join. IPv4 walks every NIC; IPv6 does not. Callers who need every IPv6 interface must join per `iface=`.

IPv6 SSM without `iface=`. Not recommended. The implementation refuses to walk interfaces for IPv6 source filters.

Loopback delivery. Alpine (especially under QEMU) often does not deliver loopback multicast even when the join succeeds. `tests/posix/mcast.icn` still checks membership and treats a missing datagram as success on Alpine so CI stays portable.

Windows IPv6 SSM. Membership calls are attempted; `loopback_0`

and UNIX `lo` / `lo0` names are mapped. Delivery and option support vary by stack; the test skips cleanly on `ENOPROTOOPT`.

Raw sockets [4] reuse `ttl` / `iface` / `join` on `SOCK_RAW`. That is a separate report.

11 Appendix: test coverage

`tests/posix/mcast.icn` runs in one process over loopback, one UDP port per case so a late datagram cannot land on the next receiver. It checks:

- unknown attributes and numeric booleans fail with 1310
- explicit `join=` plus `iface=127.0.0.1`
- implicit join by binding the group address
- SSM via `source@group` and via `source=`
- broadcast open of `255.255.255.255` (never `EACCES`)
- `Attrib() ttl=`, `join=`, `leave=`
- IPv6 SSM membership via `@`, `source=`, and `join=group,source` (delivery not required)
- cache-hit alias vs independent attr open

`tests/posix/sockpeek.icn` checks `f["ttl"]`, `f["groups"]` as a list after two joins, and `key(f)`.

Expected output is `tests/posix/stand/mcast.std`:

```
bogus attribute: bad socket attribute
numeric boolean: bad socket attribute
Received hello-multicast
Received hello-implicit
Received hello-ssm-at
Received hello-ssm-source
broadcast: ok
```

```
ttl=3
Received hello-attrib
leave: ok
ipv6-ssm-at: ok
ipv6-ssm-source: ok
ipv6-ssm-join: ok
cache-attr open: independent
Received hello-alias
```

12 References

References

[1] S. Deering. "Host Extensions for IP Multicasting". RFC 1112. 1989.

[2] Clinton Jeffery and Shamim Mohamed and Jafar Al-Gharaibeh. "Programming with Unicon". doc/book/. 2026.

[3] H. Holbrook and B. Cain. "Source-Specific Multicast for IP". RFC 4607. 2006.

[4] Jafar Al-Gharaibeh. "Raw Sockets and Packet Layouts in Unicon". doc/utr/utr29.rst. 2026.