Security Considerations for Concast

Ken Calvert
University of Kentucky
Laboratory for Advanced Networking
ActiveCast Project—Sponsored by DARPA
http://www.dcs.uky.edu/~acast/

Outline

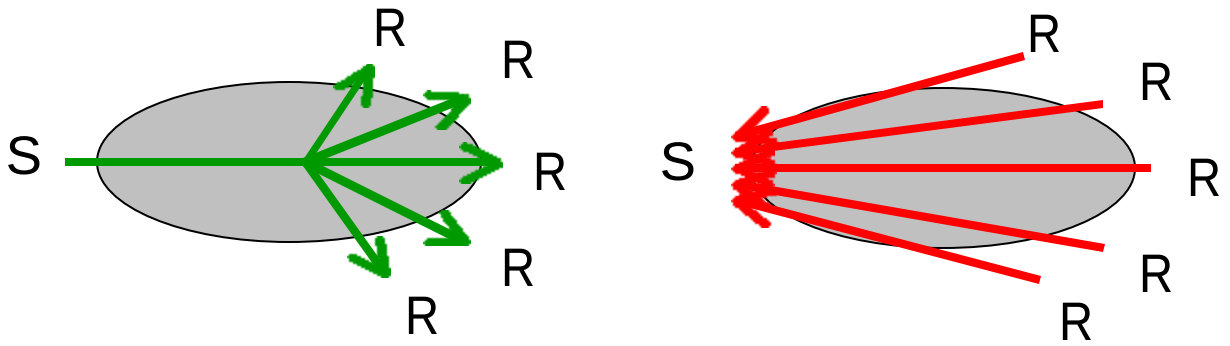
0. The Issue
1. Concast Overview
   - Service
   - Flow Setup (Signaling Protocol)
2. Trust/Policy Considerations for Concast

Issue: Infrastructure/User Trust

- Infrastructure services
  - Plain old unicast forwarding
  - Multicast forwarding
  - Nack suppression
  - Caching
  - Data-dependent forwarding
  - Aggregation (e.g. feedback)
- Increasing sophistication of services $\Rightarrow$ harder trust considerations
  - Example: routers munging with payloads $\Rightarrow$ no end-to-end security

Concast: The Problem

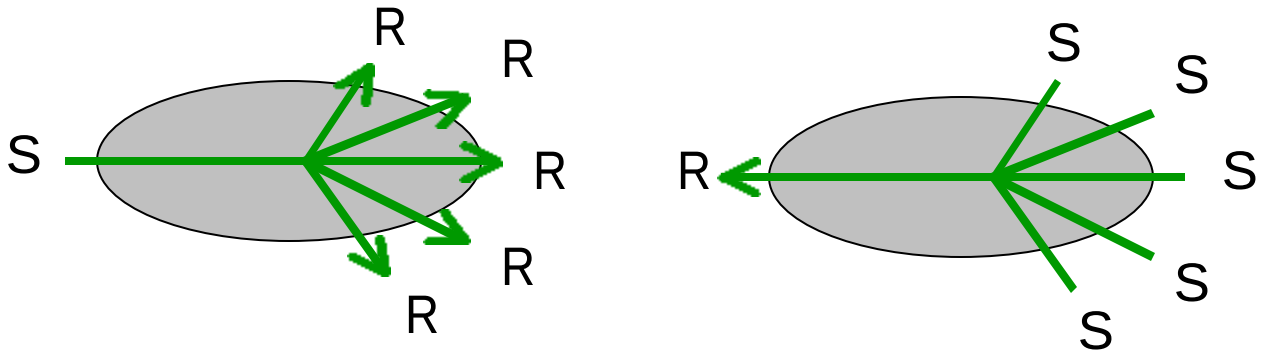
- Many multicast applications involve <u>feedback:</u>
  - Retransmission requests for reliability
  - Loss rates for congestion avoidance



- Sending feedback via existing channels is ugly
  - Sender deals with individual receivers, destroying abstraction
  - <u>Implosion</u> limits scalability
- Need a Many-to-One channel

Concast Service Overview

- Scalability through abstraction
  - Single identifier (concast group ID) represents an arbitrary number of senders
  - Network merges messages en route to receiver
    - Semantics specified by receiver

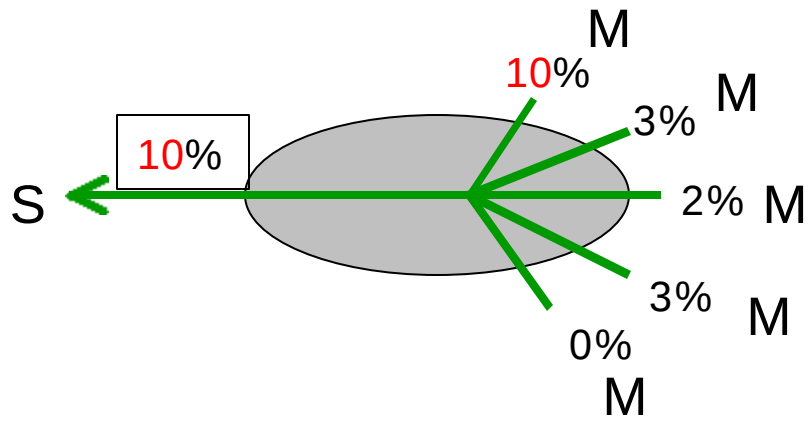


- Benefits both receiver and network
  - Multiple sends result in a single message delivery
  - Trade additional processing in routers for reduced bandwidth requirements

Example: Distilling Group Info

Scenario: Multicast sender S needs feedback from group members (M) to adapt its transmission rate

- E.g., maximum loss rate among all receivers
- Each receiver sends loss rate in a concast message
- Network computes maximum and delivers it



Flexible Merge Semantics

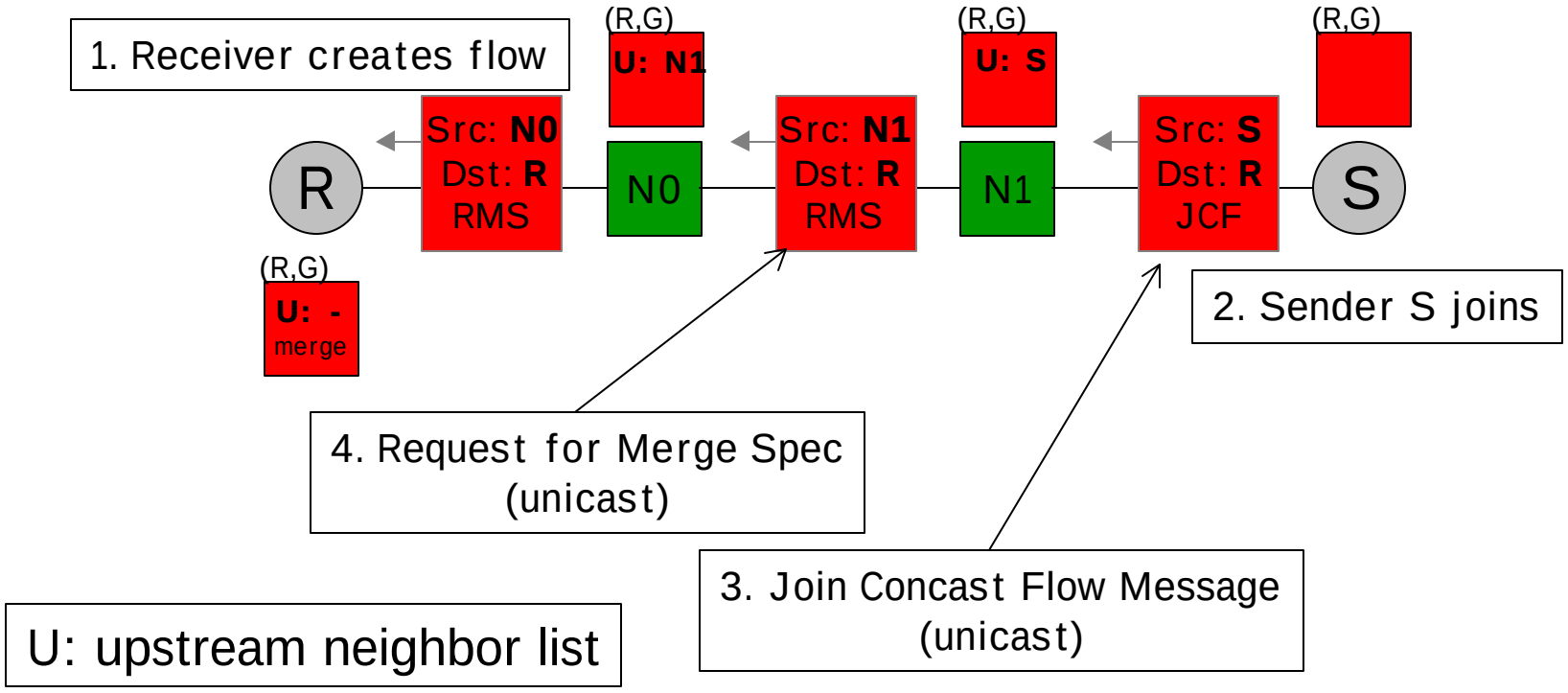
- Conservative: hardwire various merge semantics into the network, user selects at flow setup time
- Liberal: user specifies merge computation to be carried out by the network (intermediate systems)
  - E.g., by downloading Java bytecodes
- Challenge:
  - Allow customization of merge semantics
  - Within a practically-implementable framework that protects the safety of the network
  - Deal with trust issues implied by network support

Merging Datagrams

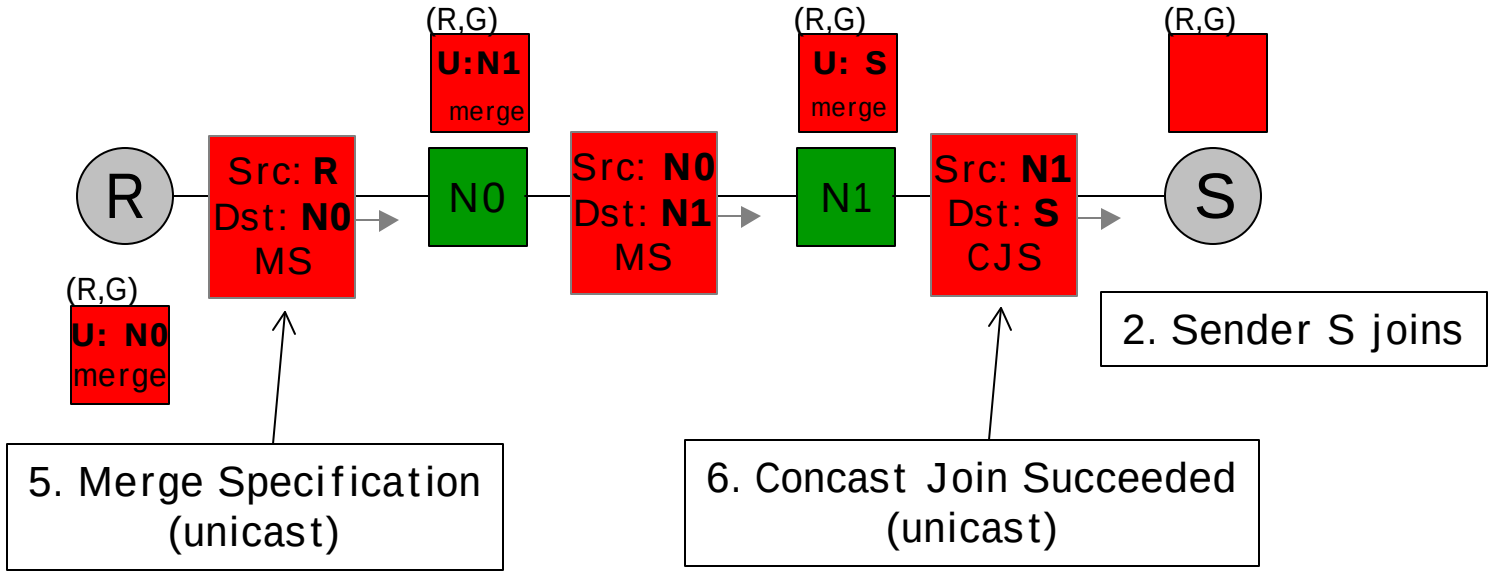
Receiver defines merge semantics via Merge Specification:

- Which messages to merge together
  - E.g. match a template based on fields of the message
- Mapping from incoming IP payloads to stored state
  - E.g. sum field contents into accumulator variable
- Mapping from stored state to forwarded IP payload
  - E.g. forward accumulator value
- Predicate indicating when forwarding should occur
- Security Aspects
  - Membership admission policy
  - Secrecy requirements
  - …

Concast Signaling Operation

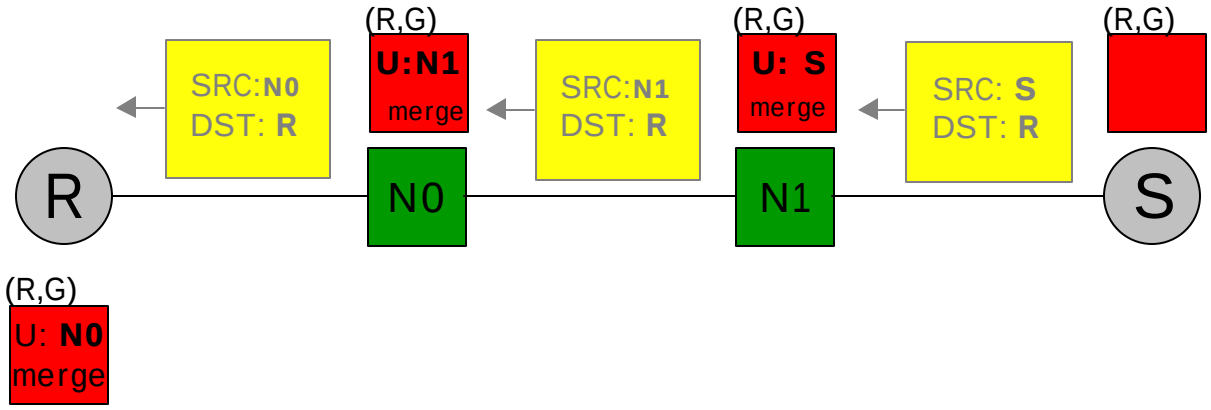


Concast Signaling Operation

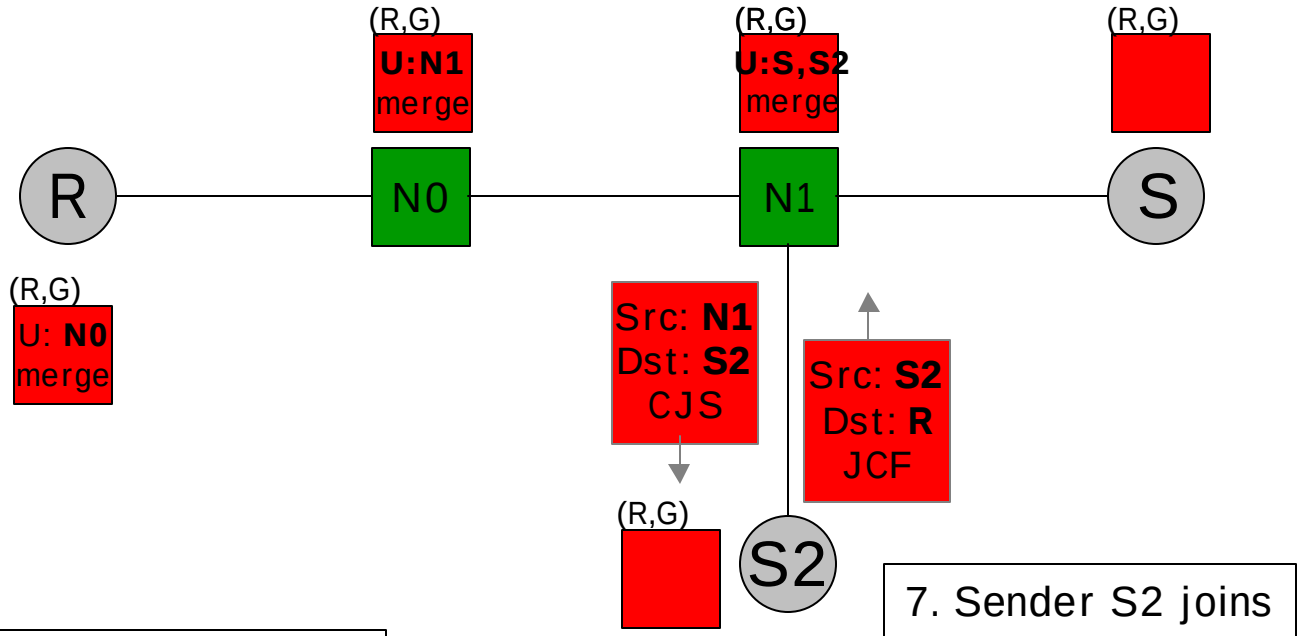


U: upstream neighbor list

Concast Signaling Operation



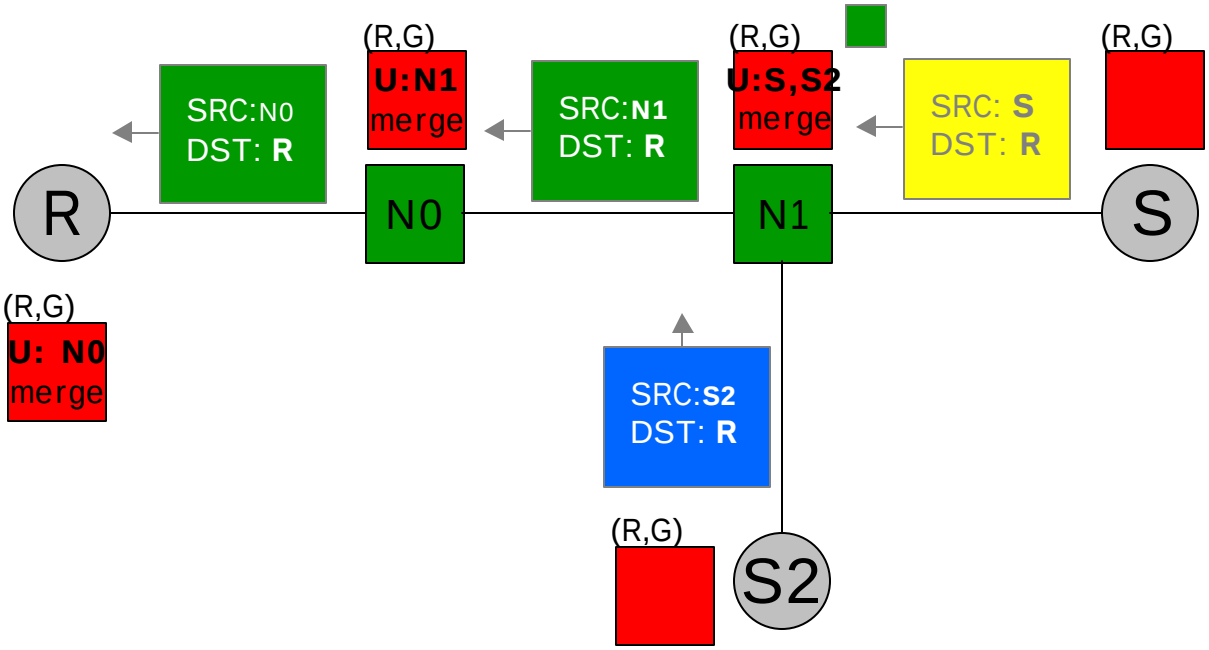
Concast Signaling Operation



7. Sender S2 joins

U: upstream neighbor list

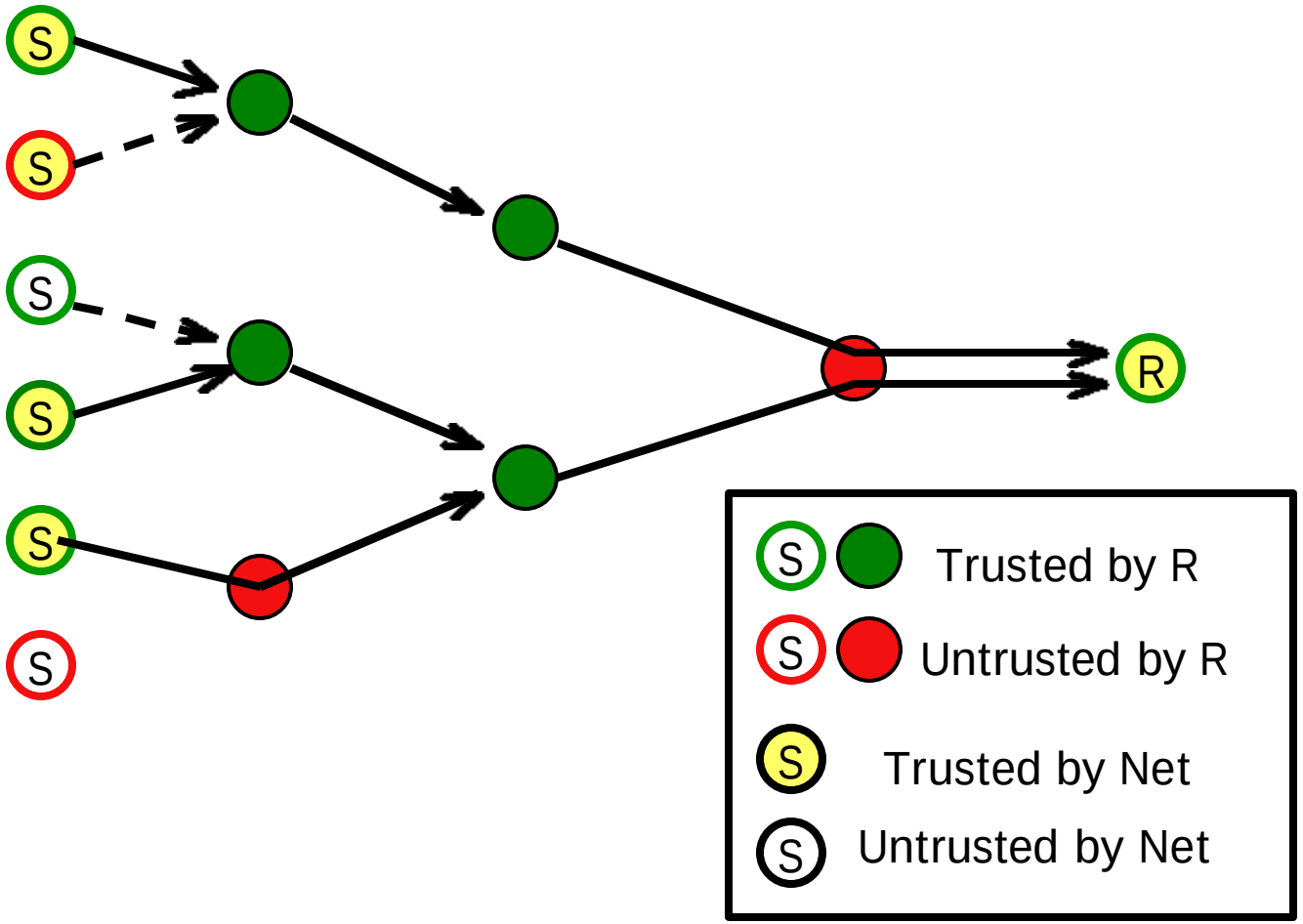
Concast Signaling Operation



Policy/Trust Considerations

- Access control
  - Who gets to join the flow
- Flow creation (specifying merge semantics)
  - Who gets to create a flow
- Implementation of Merge semantics
  - Who is trusted to implement merging correctly
  - The way concast does tree establishment means that this includes access control
- Confidentiality (for payload-dependent merge semantics)

Example


S
S
S
S
S
S
R
S
Trusted by R
S
Untrusted by R
S
Trusted by Net
S
Untrusted by Net

Per-Flow Policies

- Group access (Receiver defines)
  - Who can join as sender
- Merge Spec distribution (Receiver defines)
  - Which nodes perform operations on behalf of users
  - Which nodes can extend the flow (i.e. apply membership policy)

Per-Node/Domain Policies

- Group access
  - Who can be a sender (flow-independent)
- Flow creation
  - Who can supply merge spec
    - Users
    - Downstream routers
- Flow extension
  - Who can apply the membership policy
  - Currently tied to Merge Spec

Observation

- Concast gets “overlay” for free
  - State goes only on the Sender-Receiver path
  - Hard to separate execution and flow extension
  - Other services (explicit tree construction, etc.) may have different tradeoffs
    - e.g. better separation with higher overhead