

# **Title:** Remote Priority Queue: an Abstraction for Application-level Flow Control and Reliability in Push-based Communication

**Name:** Dmitrii Zagorodnov

**Email:** [dmitrii@cs.uit.no](mailto:dmitrii@cs.uit.no)

**Student:** no

We define push-based communication from an Internet user's point of view, as one in which data items arrive asynchronously, either unsolicited or in response to an earlier subscription. Email and Usenet are the classic examples of such communication; IP multicast, instant messaging, RSS feeds, and content-based publish/subscribe systems are the more recent ones. The asynchrony has two implications:

- The user may be unavailable to process the incoming data, so the user's proxy – the host entrusted with the task of receiving push-based communication on behalf of the user – has to decide what data to receive. We call this the problem of *application-level flow control*. Although transport-level flow control, such as provided by TCP, can induce the sender host to respect the speed of the receiver host, only the application can induce the sender host to respect the speed of the user. If the latter is not done, resources will be wasted on the data that the user has no time to process.
- The proxy host may be unavailable to receive the incoming data, due to a host or link failure. Ensuring that the user receives every wanted item is the *application-level reliability* problem. Again, TCP can ensure that the packets in an established connection are delivered reliably, but when communication is temporarily impossible, only the application can re-establish the transfer at a later time. Without an explicit scheme for transferring the messages missed during an outage, message loss is inevitable.

By and large, support for application-level flow control and reliability in contemporary push-based services is incomplete, tacit, or absent altogether. For example, flow control in email is very coarse-grained: a receiving mail server can block overly large messages or messages from “rogue” sources, but all others are filtered after they have been received; reliability is supported through repeated attempts to resend messages, up to a time limit chosen internally by the server. Usenet and RSS postings persist – and thus are available to a temporarily unavailable subscriber – until their hosting servers choose to expire them. Usenet newsgroups, RSS feeds, and pub/sub subscriptions are normally not flow controlled: their entire contents are downloaded. Message brokers in publish/subscribe systems typically act as routers, with no long-term buffering of messages for reliability.

The lack of flow control generally leads to waste (the worst offender being transfer of spam). The lack of explicit reliability support means push-based services cannot be trusted with important notifications.

We propose a conceptually simple abstraction for a session-layer interface of push-based services: a *Remote Priority Queue* (RPQ). By storing incoming data items for an explicit period (potentially indefinitely) and returning them ordered by some notion of priority – arrival time, spam score, relevance rank – such queues can serve as generic rendezvous points for producers and consumers of any type of data. Consumers choose how many items to dequeue per unit of time based on user preferences or activities, and thus achieve flow control. For reliability, consumers query past events with a timestamp-based interval, which allows them to efficiently retrieve all new items since the timestamp of their last contact with the queue. With the session-layer details invisible to the application, the RPQ interface abstracts away the mechanisms used for the delivery of notifications (e.g. polling, persistent connections, UDP or TCP-based callback). Due to its generality, any type of one-to-one or one-to-many message-based communication system can be built around an RPQ.

We are currently implementing a session-layer library based on the RPQ interface. We are also writing applications – several services and a client – on top of the library. The services range from a simple FIFO communication substrate to recommender systems driven by user feedback.