What‘s Broken in XMPP?

And how can we fix it?

2018-01-30.00
Ge0rG on
xsf@muc.xmpp.org

Agenda

- Client Sessions: RTTs, client-side complexity
- Message Routing
  - It’s broken
  - The fixes (Carbons, MAM) are broken
  - Multi-Client / Mobile is broken
  - The Big Picture
  - Message Routing 2.0
- Other Issues:
  - “Smart” notifications
  - MUC: self-ping, directed presence

Sessions

- Initial Design:
  - long-living stable TCP connection (stationary client and server)
  - Expensive setup (many RTTs for hello, TLS, auth, session bind)
- Bolt-on optimizations:
  - Roster versioning (delta transmission, still full presence flood)
  - Stream Management (limited to ~5mins, simulates availability, destruction of zombie sessions is challenging WRT messages)
- Problems:
  - Stale/dead sessions “consume” messages
  - Client-side session setup/resumption complexity

Client Session Setup

- Establish Secure Connection (DNS/SRV, TCP, TLS+Validation)
- Authenticate
- Attempt Session Resumption
  - On success: send queued stanzas, receive server's buffer → done!
- Bind new session, request roster delta
- Subscribe to Carbons
- Request missed messages from MAM
- Send initial presence
  - Receive offline messages (duplicates from MAM) or use XEP-0013
  - Receive presence flood from all roster items
- Join MUCs (Obtain MUC history, query MUC-MAM, send queued MUC(-PM) messages)
- Request Caps

Solution: Session 2.0

- Merge session resumption, resource binding, offline sync
  - Less round-trips
  - Move most complex logic to server-side
- Client sends ISR / bind 2.0 stanza with payloads:
  - Last 0198 session-id+h (allows immediate stream resumption)
  - Last used resource (server can kill stale session / rebind)
  - Last known MAM-ID (for delta update on messages)
  - Initial presence
- Server responds with
  - <resume/> plus whatever was queued, if possible
  - new bound session and MAM result set, otherwise
- TODOs: MUC auto-join (maybe account-proxy); reduce incoming presence flood

Message Routing

- Initial Design:
  - All messages are (more or less) ephemeral
  - “Most available” client captures all messages
  - Other / off-line clients get nothing
  - Resource locking + presence-show/-prio determine “most available”
- Routing Rules (depend on message type):
  - Bare-JID messages sent to (implementation-defined) subset of clients
  - Full-JID messages sent to single client (type=chat: fallback to „most available“)
  - Full-JID messages can leverage receiving client‘s capabilities (Resource locking)
- Bolt-on „optimizations“:
  - Carbons to copy some messages to other clients
  - MAM to re-sync clients coming online

Message Types

- Initial design: each message type has different semantics
  - „chat“ = typical IM
  - „normal“ = kind of like email
  - „headline“ = annoying pop-up
  - „groupchat“ = reserved for XEP-0045
  - „error“ = things gone wrong
- „chat“ and „normal“ qualify for offline storage
- Routing depends on message type

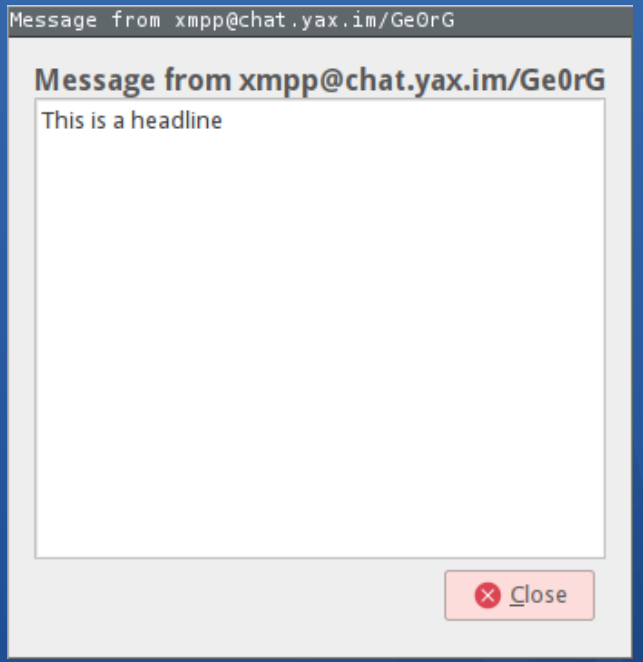
Message Type Routing

Type	To bare/online	To bare/offline	To full/online	To full/offline
`chat`	most available	store/error	deliver	store / m.-a.
`normal`	most available	store/error	deliver	ignore/error(*)
`headline`	non-negative	ignore	deliver	ignore/error
`groupchat`	error	error	deliver	ignore/error
`error`	ignore	ignore	deliver	ignore

- „most available“ = implementation defined, non-negative presence – can be none, one or any subset of clients
- (*) ejabberd: „most available“ because it broke some clients

Message Type Problems

- Routing depends on message type
- Routing depends on full/bare to-JID
- No message type for „system“ messages like MAM results
- XEPs and developers don‘t specify message type / use type incorrectly (e.g. XEP-0184)
- Focus-stealing headline pop-up = UX nightmare (screenshot from MUC-PM)



Message ID Problems

- Sender-defined identifier for a message
  - Optional
  - May be low-entropy - „1“, „2“, „3“, „1“…
  - May be rewritten by MUCs
- XEP-0359: Unique and Stable Stanza IDs
  - MUST be unique, high-entropy value
  - Who is responsible for generation?
  - <stanza-id> vs. <origin-id> vs message ‚id‘
- Last Message Correction, 0184 ACKs – which ID to reference?

Conceptual Problems

- Users expect full history on all devices
- Users expect “smart” notifications
- Carbon-copy rules are vague, incomplete and overly complex
- MAM rules are incomplete, moderately complex and different from Carbons
- MAM does not give you the MAM-ID of sent messages
- MAM / offline messages / live messages → race condition on session setup!
- Battery saving (CSI) and push have different rules again

Practical Problems

- Problems with Carbons / MAM / offline messages:
  - Messages get re-routed / carbon-copied to incapable clients
  - Different rules → disjoint histories on devices
  - Message gets CCed, receipt/error response doesn‘t
  - MAM + MUC = madness (3rd-party hosted, NS versions!?)
  - Carbons + MUC(-PMs) = madness (is a <message/> MUC-related?)
  - OTR = madness
- Interop problems with other XEPs:
  - Which entity shall send 0184 ACKs? One client? All clients? MAM?
  - Implicit assumption: Body-less messages = ephemeral = unimportant (Chat Markers, Chat States, ACKs, OMEMO, …)

The Big Picture

- **This is a chat history database synchronization problem!**
- All “full-sync” clients should eventually arrive at the same chat history state, whether they were online, offline or in CSI battery saving
- Unified rules needed for all Message Routing XEPs:
  - Carbons, MAM, CSI, Push, …, error responses
- What affects Message Routing (for IM)?
  - Message **persistence** (will it affect the chat history DB?)
  - Message **urgency** (for immediate CSI and Push pass-through)
- We need explicit encoding for persistence and urgency!

S: Message Routing 2.0

- Persistence: Change Semantics of Bare-JID/Full-JID recipient
  - Bare-JID = persistent, delivered to all clients (allowed by RFC6121)
  - Full-JID = ephemeral, single-client, never re-routed (conflicts 6121)
  - Need for ephemeral all-online-clients routing (Bare-JID + <no-archive/>?)
- Burn (Message) Resource Locking with fire!
  - Or at least send all persistent messages to Bare-JID and ignore client caps for them
- Urgency: strawman proposal, discussion needed!
  - by default: persistent = urgent
  - use Hints XEP for deviations in either direction
  - How to handle MUC/MIX mentions (E2EE vs. meta-data leak)?

S: Message Routing 2.0

- Incompatible with current clients / servers? ☹
- Requires changes in client-server communication
  - Leverage bind2 → session2
  - Apply new Bare-/Full-JID semantics
  - “MAM subscription” instead of Carbons? (+ sent-msg ID reflection)
- Requires transition logic for legacy XMPP clients/servers
  - Use <no-archive/> + <private/> on messages leaving session2 domain
  - Apply combined legacy-{Carbons, MAM, CSI, Push} wisdom on messages entering session2 domain, derive Urgency and Persistence

Other Issues

- „Smart“ notifications
  - Need logic for audible / silent / no notification, depending on past and current activity on other devices
  - Possible approach: other device active → delay notification by 30s
    - Cancel notification if other device types / responds
  - Synchronize „read“ state between clients (Chat Markers, CSNs, sent-Carbons), cancel notification

MUC: am I there still?

- MUC designed for perfect networks and service uptime
- Problems:
  - Join treated as presence update
  - MUC service restarted (silently)
  - Client disconnected, kick presence lost
- Symptoms:
  - You only see a small subset of participants, messages from apparently offline users
  - You don‘t see anything happen, can‘t send messages

MUC: Self-Ping

- Client to periodically check if it is still joined
  - Send something („self-ping“)
  - Rejoin on error / timeout
- All self-ping mechanisms broken:
  - Presence to MUC: mistreated as GroupChat1.0 join, reflected to all participants → $O(N^2)$
  - Silent message to MUC: reflected to all participants → $O(N^2)$
  - PM to own participant-JID: might be forbidden by MUC config
  - IQ to own participant-JID: routed to „random“ MSN client, lost there → false positives → erroneous rejoin

MUC: rejoin-if-needed

- New <presence> flag: „rejoin-if-needed“, advertised in MUC Caps
- Client sends normal join presence
- Client sends presence updates with „rejoin-if-needed“ flag periodically or on-demand (<history/> allowed)
- Service handles as follows:
  - If client joined: only reflect if presence different (to avoid $O(N^2)$ flood)
  - If client not joined:
    - respond with presence-unavailable (to let the client flush participant list)
    - then respond with full join response

MUC: directed presence

- User‘s server tracks all outgoing directed presence (e.g. join)
- Problem with MUC-initiated participant nick change:
  - Client joins a MUC, is (later) renamed by MUC service
  - User‘s server still caches initial participant JID
  - Client exits
  - User‘s server sends directed presence-unavailable to **old** participant JID (old nickname)
- Possible solutions:
  - MUC accepts presence-unavailable from old/initial nickname
  - User‘s server needs to understand MUC semantics