Technical Disclosure Commons

Defensive Publications Series

August 22, 2016

PRESENCE INDICATION IN REAL-TIME CHAT APPLICATIONS

Johan Wikström

PRESENCE INDICATION IN REAL-TIME CHAT APPLICATIONS

ABSTRACT

This document describes techniques to display presence indicators in real-time chat applications. The techniques enable accurate presence indicators with low overheads. A server compares successive timestamps to determine whether a time threshold is met. If the time threshold is met, the server pushes an updated last-seen timestamp to clients. Otherwise, the server only selectively pushes the updated last-seen timestamp to clients. The client polls the server if the last-seen timestamp meets a threshold difference from the current time. The combination of pushing by a server and polling by a client to selectively update last-seen timestamps enables accurate and consistent representation of presence state for multiple users and client devices. The techniques save server resources, network costs, and reduce energy demands on client devices.

KEYWORDS

- Presence indicator
- Availability indicator
- Real-time chat
- Instant messaging

BACKGROUND

Real-time chat applications can indicate to users when their contacts are available for chatting. Such indication may utilize visual indicators, e.g., a green dot next the username of available (present) contacts and a white dot next to the username of unavailable contacts. In some applications, users may manually set their presence, e.g., as “Available”, “Busy”, “Offline”, etc. Some applications can automatically determine when a given user’s contacts are available or present, and generate the indicator without manual input. Automatic presence determination can be implemented by a service provider, e.g., the service provider that provides the chat application. In determining and indicating presence automatically, it is important that presence information of a given user is indicated consistently to their contacts. Consistency of presence information implies that all users see the same presence information for a given contact at a particular time. For example, a user Alice can see a contact Bob as present while another user Cindy who has the contact Bob also sees Bob as present at the same time.

A possible solution to maintain consistent display of presence indicators across chat applications associated with different users and executed on different devices is to ensure that each chat application always has an updated last-seen timestamp for each contact. To implement this solution, a server pushes the last-seen timestamp for a particular user to all users that have the particular user as a contact each time the timestamp associated with the particular user is updated. However, such push updates are bandwidth intensive. Further, for mobile clients, such a solution may be unsuitable, e.g., due to battery constraints.

DESCRIPTION

Figure 1 illustrates a diagram of an example environment 100 in which techniques of the present disclosure may be implemented. In figure 1, three client devices (102a, 102b, 102c) are shown. In different implementations, client devices 102 can include desktop computers, laptop computers, tablets, phones, wearable devices, etc. Respective users Alice (104a), Bob (104b), and Cindy (104c) use client devices to access a chat application 110, shown as client-applications 110a, 110b, and 110c respectively. In different implementations, the chat application is an application executing on respective device 102, e.g., a browser-based application, etc. The client devices are coupled to server 106 via network 108. The server enables the users to chat with each other using the chat application. The server maintains datastore 120 that stores last-seen timestamps for users of the chat application.

Each chat application displays usernames of the contacts of the respective user, e.g., usernames 120a, 120b, and 120c. In this example, each of Alice, Bob, and Cindy are contacts of the other two users. For example, usernames 120a shown on client device 102a include names of Alice's contacts (e.g., Bob and Cindy), usernames 120b shown on client device 102b include names of Bob's contacts (e.g., Alice and Cindy), and usernames 120c shown on client device 102c include names of Cindy's contacts (e.g., Alice and Bob). Further, chat application 110 shows a presence indicator next to each username displayed. The presence indicator indicates the availability or presence of the respective user. For example, on client device 102a of the user Alice, presence indicator 112a indicates that Bob is present and presence indicator 114a indicates that Cindy is not present; on client device 102b of the user Bob, presence indicator 116a indicates that Alice is present and presence indicator 114b indicates that Cindy is not present; and on client device 102c of the user Cindy, presence indicator 116b indicates that Alice is present and presence indicator 112b indicates that Bob is present.

In operation, the chat application accesses the server for a variety of purposes, e.g., to send messages, to update a focus state of the chat application, to indicate that the user (e.g., any of the users 104) is logged in, etc. The chat application sends remote procedure calls (RPCs) to the server. Further, the chat application accesses the server to obtain information for a user's contacts. In some examples, a contact is deemed as present if the contact used the chat application within a threshold amount of time, e.g., within the last 15 minutes. The server maintains information on when each user was last seen and provides it to the chat application. Based on the information from the server, the chat application determines that a contact is present, e.g., based on a determination that the contact was last seen within the threshold amount of time.

For example, chat application 110a queries the server for a last-seen timestamp for Bob. In response, the server retrieves the last-seen timestamp for Bob from the datastore and returns it to chat application 110a. In some implementations, the server also automatically subscribes Alice to receive push notifications when Bob's presence changes. For example, the server subscribes Alice to such push notifications on a per-user basis, where all chat applications 110 associated with Alice (e.g., applications for which Alice is logged in) receive the push notifications. In another example, the server subscribes Alice to push notifications selectively, e.g., chat applications 110 on desktop clients are subscribed to the notifications and chat applications 110 that are mobile clients are not subscribed to the notifications. In another example, each chat application 110 indicates to the server a suitable type of subscription. The chat application displays presence indicators (e.g., presence indicators 112, 114, and 116) based on the last-seen timestamp associated with the particular user.

Techniques of this disclosure enable consistent behavior of presence indicators across multiple users and client devices. In implementing the techniques, server 106 and each of chat applications 110 store a time threshold (e.g., 15 minutes). If a user is active within the time threshold (e.g., logged in, sending messages, etc.) the user is deemed to be present.

Server Operation

Figure 2 illustrates an example method 200 for a server (e.g., server 106) to implement techniques of the present disclosure.

At 202, the server determines a last-seen timestamp T1 for a particular user. For example, the server determines the last-seen timestamp based on activity from a chat application associated with the user, e.g., Bob. In different examples, such activity may include the chat application accessing the server for a variety of purposes, e.g., to send messages from Bob, to update a focus state of the chat application, to indicate that Bob is logged in, etc.

At 204, the server looks up the previous last-seen timestamp T0 for Bob.

At 206, the server compares the time difference between T1 and T0 with the time threshold (e.g., X minutes). If the server determines that the user Bob was last seen more than X minutes ago, the method proceeds to 210. Else, the method proceeds to 208.

At 208, the server determines whether the time difference between T1 and T0 is less than a second time threshold Y. For example, the second time threshold Y is lower than X, e.g., 0.7X or another fraction of X. If the time difference is less than Y, the method proceeds to 212. Else, the method proceeds to 210.

At 210, the server pushes T1 as the last-seen timestamp for Bob to all subscribed clients (e.g., chat applications 110). The method proceeds to 212.

At 212, the server replaces T0 with T1 in its datastore (e.g., datastore 120) as the last-seen timestamp for Bob.

Client Operation

Figure 3 illustrates an example method 300 for a client (e.g., chat application 110) to implement techniques of the present disclosure. For example, method 300 can begin after the client device has obtained a timestamp T from a server. In some examples, the client device can obtain the timestamp in response to the client device querying for a timestamp from the server (e.g., when querying for a user's presence), or can obtain the timestamp as a received timestamp pushed to the client device by the server.

At 302, the client stores a timestamp T for a user (e.g., Bob) received from the server. If the client has no previous timestamp for the user, the client stores T as the last-seen timestamp. If the client has a previous timestamp for the user, the client replaces the previous timestamp with T.

At 304, the client compares the difference between the current time and T with the time threshold (e.g., X minutes). For example, the time threshold (e.g., X minutes) used at the client is the same time threshold (e.g., X minutes) used at the server described above. If the difference is less than the threshold, the client determines that the user is present and displays a presence indicator accordingly, e.g., a green dot. If the difference is not less than the threshold, the client determines that the user is not present and displays a presence indicator accordingly, e.g., a white dot.

At 306, the client determines if the difference between the current time and T matches a particular value, e.g., the difference is equal to X-1 minutes. If the difference matches the particular value, the method proceeds to 308, else the method returns to 304.

At 308, the client queries the server for Bob’s last-seen timestamp and proceeds to 302.

In situations in which the systems and methods discussed herein may collect or use personal information about users (e.g., user presence status or information, information about a user's social network, location, biometric information, and/or activities and demographic information), users are provided with one or more opportunities to control whether information is collected, whether the personal information is stored, whether the personal information is used, and how the information is collected about the user, stored and used. That is, the systems and methods discussed herein collect, store and/or use user personal information only upon receiving explicit authorization from the relevant users to do so. For example, a user is provided with control over whether programs or features collect user information about that particular user or other users relevant to the program or feature. Each user for which personal information is to be collected is presented with one or more options to allow control over the information collection relevant to that user, to provide permission or authorization as to whether the information is collected and as to which portions of the information are to be collected. For example, users can be provided with one or more such control options over a communication network. In addition, certain data may be treated in one or more ways before it is stored or used so that personally identifiable information is removed. In some examples, a user's identity or geographic location may be treated so that no personally identifiable information can be determined.

Advantages

The techniques of this disclosure utilize high queries per second (QPS) data (e.g., based on client activity) to obtain the last-seen timestamp for a user of a chat application with good granularity. Further, the techniques filter the high QPS data prior to pushing the last-seen timestamp to a client which enables lower push QPS. Pushing the last-seen timestamp (e.g., 208 to 212) for a user enables all subscribed contacts to see the user as present as soon as the user goes online. Further, the combination of pushing and polling (e.g., 210 and 308) enables all subscribed clients to display a consistent presence indicator for a user when the user goes offline. For example, by implementing the techniques of this disclosure, all subscribed clients show the user as offline X minutes after the last-seen timestamp on the server. The techniques of this disclosure enable consistency of presence indicator for a user across clients, e.g., no client displays a particular user as present if another client displays the particular user as absent at the same time. Further, when a client changes state to present, all subscribed clients instantly show the client as present, unlike techniques that are exclusively based on polling.

The combination of pushing and polling also limits the queries per second (QPS) from the server to the client. Limiting the QPS saves server resources and network costs. For battery-powered clients such as mobile devices, the techniques enable savings in battery usage.

CONCLUSION

Presence indicators are valuable to users of real-time chat applications. Naïve solutions that push updates from the server to subscribing clients each time a user’s last-seen timestamp are bandwidth intensive. Further, such solutions are unsuitable for certain types of clients, e.g., mobile clients, since they are battery intensive. The techniques of this disclosure enable accurate presence indicators in real-time chat applications with low overheads. To update last-seen timestamps for a user’s contacts, the techniques use a combination of pushing by a server and polling by a client. Implementation of these techniques enables accurate and consistent representation of presence state for multiple users and client devices. The techniques save server resources, network costs, and reduce energy demands on client devices.

FIGURES

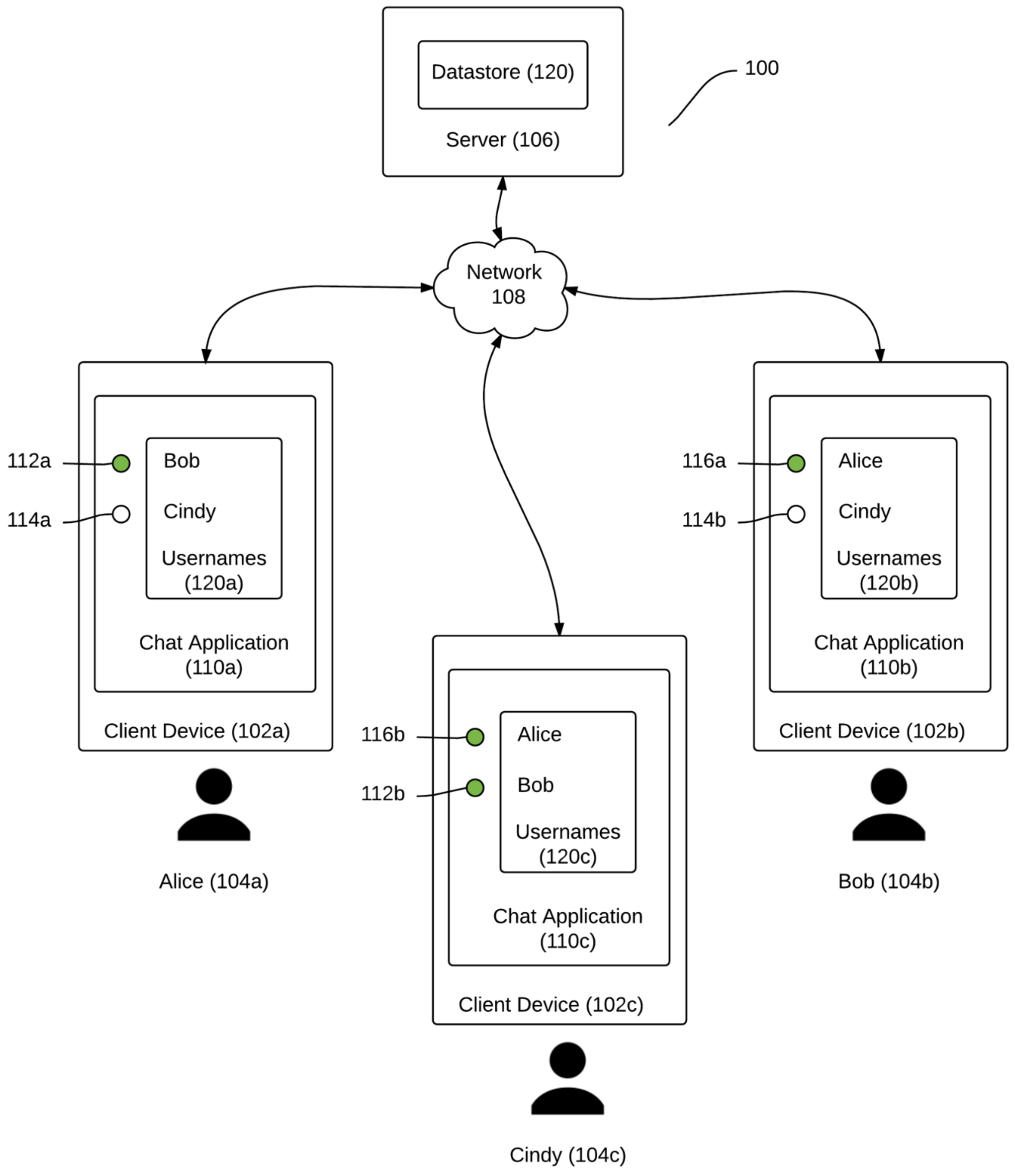

FIG. 1

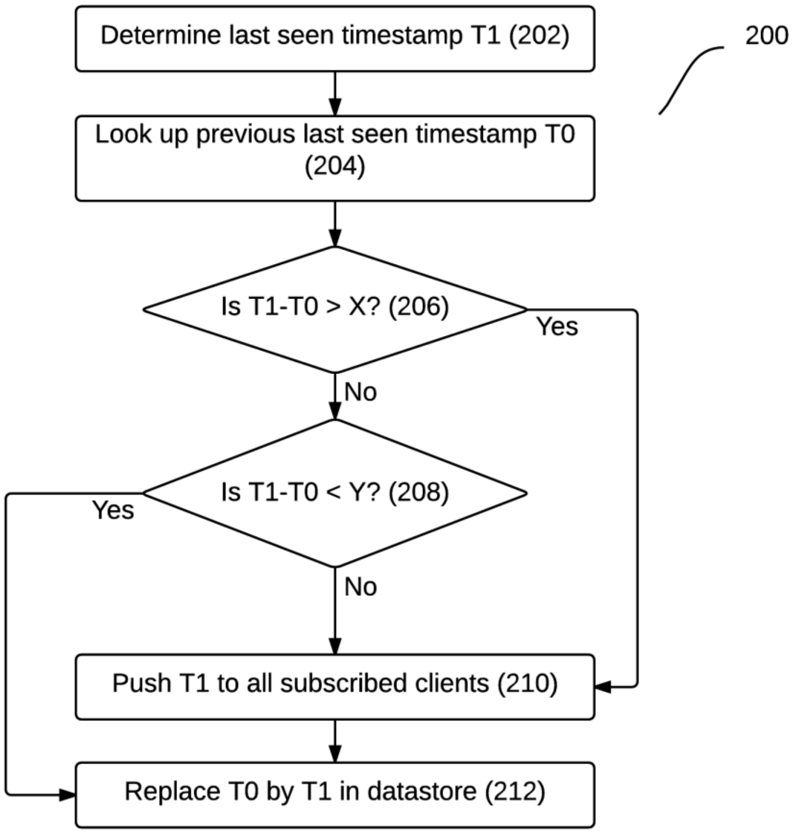
200
Determine last seen timestamp T1 (202)
Look up previous last seen timestamp T0 (204)
Is T1-T0 > X? (206)
Yes
No
Is T1-T0 < Y? (208)
Yes
No
Push T1 to all subscribed clients (210)
Replace T0 by T1 in datastore (212)

FIG. 2

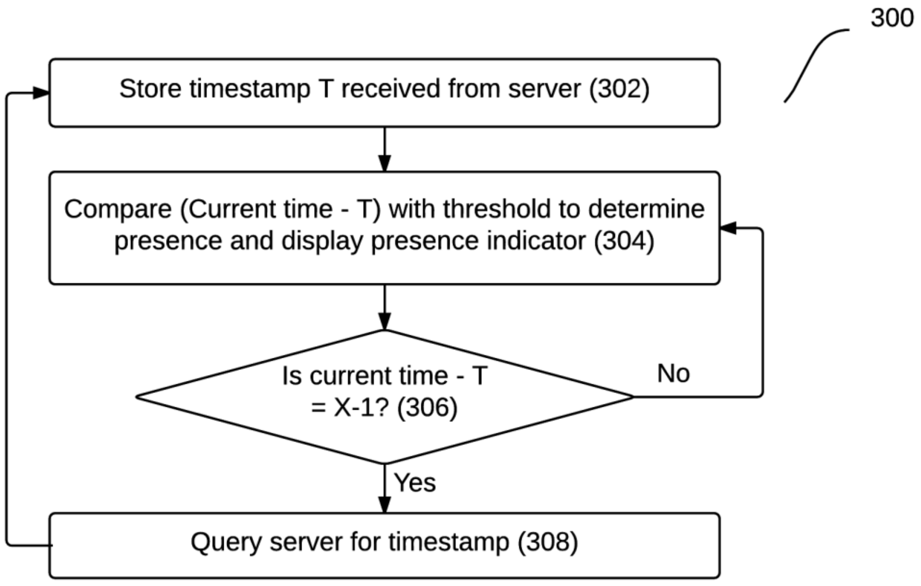
300
Store timestamp T received from server (302)
Compare (Current time - T) with threshold to determine presence and display presence indicator (304)
Is current time - T = X-1? (306)
No
Yes
Query server for timestamp (308)

FIG. 3