Original Article

Capture of real-time data from electronic health records: scenarios and solutions

Nikola Kirilov^

Institute of Medical Informatics, Heidelberg University Hospital, Heidelberg, Germany

Correspondence to: Nikola Kirilov, BSc, MSc. Institute of Medical Informatics, Heidelberg University Hospital, Im Neuenheimer Feld 130.3 69120, Heidelberg, Germany. Email: Nikola.Kirilov@med.uni-heidelberg.de.

Background: The integration of real-time data (RTD) in the electronic health records (EHRs) is transforming the healthcare of tomorrow. In this work, the common scenarios of capturing RTD in the healthcare from EHRs are studied and the approaches and tools to implement real-time solutions are investigated.

Methods: Delivering RTD by representational state transfer (REST) application programming interfaces (APIs) is usually accomplished through a Publish-Subscribe approach. Common technologies and protocols used for implementing subscriptions are REST hooks and WebSockets. Polling is a straightforward mechanism for obtaining updates; nevertheless, it may not be the most efficient or scalable solution. In such cases, other approaches are often preferred. Database triggers and reverse proxies can be useful in RTD scenarios; however, they should be designed carefully to avoid performance bottlenecks and potential issues.

Results: The implementation of subscriptions through REST hooks and WebSocket notifications using a Fast Healthcare Interoperability Resources (FHIR) REST API, as well as the design of a reverse proxy and database triggers is described. Reference implementations of the solutions are provided in a GitHub repository. For the reverse proxy implementation, the Go language (Golang) was used, which is specialized for the development of server-side networking applications. For FHIR servers a python script is provided to create a sample Subscription resource to send RTD when a new Observation resource for specific patient id is created. The sample WebSocket client is written using the "websocket-client" python library. The sample RTD endpoint is created using the "Flask" framework. For database triggers a sample structured query language (SQL) query for Postgres to create a trigger when an INSERT or UPDATE operation is executed on the FHIR resource table is available. Furthermore, a use case clinical example, where the main actors are the healthcare providers (hospitals, physician private practices, general practitioners and medical laboratories), health information networks and the patient are drawn. The RTD flow and exchange is shown in detail and how it could improve healthcare.

Conclusions: Capturing RTD is undoubtedly vital for health professionals and successful digital healthcare. The topic remains unexplored especially in the context of EHRs. In our work for the first time the common scenarios and problems are investigated. Furthermore, solutions and reference implementations are provided which could support and contribute to the development of real-time applications.

Keywords: Real-time data (RTD); electronic health record (EHR); representational state transfer (REST); database trigger; reverse proxy

Received: 04 January 2024; Accepted: 19 March 2024; Published online: 03 April 2024.

doi: 10.21037/mhealth-24-2

View this article at: https://dx.doi.org/10.21037/mhealth-24-2

^ ORCID: 0000-0001-7668-2448.

Introduction

The coordination of patient care activities is one of the most complex and important services in every healthcare organization (1). Therefore, the information regarding patients has to be timely delivered to make right and prompt clinical decisions (2). In the recent years, the development of data sources and communication practices in healthcare has focused on the electronic health records (EHRs) (3,4). According to the National Academies of Medicine, EHRs have multiple functions such as capture of health data, support of the clinical decisions, health information exchange, electronic communication, etc. (5). Enhancing healthcare information systems with real-time data (RTD) is transforming the healthcare of tomorrow (6).

What is RTD?

RTD is defined as information that is continuously updated and provided right away after collection. Delay should be kept as minimal as possible. This offers the possibility of real-time computing and processing (7). RTD is often associated with the following characteristics: immediate availability (little to no delay in accessing the information), frequent updates (frequently updated data, often in milliseconds or seconds), streaming (continuous flow from the source to the destination without interruption), low latency and data processing in real-time using technologies like complex event processing, stream processing, and machine learning algorithms. Handling RTD can be challenging due to the volume, velocity, and variety of data sources. Scalability, data consistency, and security are important considerations (8). New artificial intelligence (AI) solutions are able to assess and evaluate data in real-time, due to the growing computational power (9).

Highlight box

Key findings

- This paper identifies the common scenarios of capturing real-time data (RTD) from the electronic health records. In this work the challenges of the process are studied and possible solutions are proposed.

What is known and what is new?

- It is known, that RTD in healthcare is important and this has been shown in many scientific papers. Nevertheless, the exact implementation and design of the systems has not been thoroughly discussed.
- The new in this work is the investigation of the common scenarios and problems capturing RTD. Furthermore, implementations of solutions are described and clinical example as use case is given.

What is the implication, and what should change now?

- The provided reference implementations could help guide the future development of real-time applications. This could lead to the enhancement of clinical decision making and the improvement of clinical research.

Integration of the RTD in the EHRs

RTD integration and access are crucial aspects of the EHRs that can significantly enhance the quality of healthcare delivery (10). Immediate accessibility to patient information such as medical history, including diagnoses, medications, allergies, and test results allows healthcare professionals to make informed decisions quickly, especially in critical situations. EHRs are continuously updated with new information, ensuring that healthcare providers have the most recent data about a patient's health status (11). This can be particularly important for managing chronic conditions or monitoring patients in real time, such as in an intensive care unit (12,13). EHR systems often incorporate clinical decision support tools that use real-time patient data to send alerts and reminders to healthcare providers (14). For example, if a prescribed medication interacts negatively with another medication in the patient's record, the EHRs can generate an alert to prevent potential harm (15). RTD from wearable devices and remote monitoring tools can be integrated into EHRs. This allows healthcare providers to monitor patients' vital signs, activity levels, and other health metrics in real time, which can be valuable for patients with chronic conditions or those recovering from surgery (16). Through streamlined workflows RTD can improve the efficiency of healthcare. For instance, it can automate certain processes, such as notifying a pharmacy to prepare a prescription when a healthcare provider enters the order into the EHR (17). RTD is essential for telemedicine and virtual visits (18). Patients and healthcare providers can share information and communicate in real time, even when they are not physically present in the same location. RTD can be made available to patients through EHR patient portals (19). This empowers patients to actively manage their health by monitoring their progress and communicating with their healthcare team. RTD from EHRs can be analyzed to identify trends, outbreaks, and other insights that can support public health decisions. For example, during a disease outbreak, RTD from EHRs

can help authorities track the spread and severity of the disease (20). It's important to note that while RTD in EHRs offers numerous benefits, it also raises concerns about data security, privacy, and the need for robust data infrastructure to support its transmission and storage. Healthcare organizations must implement stringent security measures to protect sensitive patient information while ensuring timely access to authorized users (21).

Standards for EHRs

The beginning of the intensive research on EHRs dates back to the 1980s. The aim was to create the clinical, ethical and technical requirements for the standards, which were later published at the International Organization for Standardization (22). Standards for EHRs are essential to ensure interoperability, data exchange, as well as the secure and consistent use of EHRs across healthcare organizations and systems (23). Several standards and frameworks have been established to guide the development and implementation of EHRs (24).

Health Level Seven International (HL7) is a widely recognized organization that develops standards for the exchange, integration, sharing, and retrieval of electronic health information. HL7's standards, such as HL7 Version 2 (HL7v2) and HL7 Version 3 (HL7v3), provide a framework for structuring clinical and administrative data in EHRs. The Fast Healthcare Interoperability Resources (FHIR) standard developed by HL7 has gained significant traction for its modern and flexible approach to data exchange (25). Standards such as these of the HL7 family (v2 and v3) are designed to build complex clinical documents, but they do not specify communication protocol, nor do they ensure interoperability. On the other hand, the recently developed OpenEHR and HL7 FHIR standards focus on interoperability and data transmission between centers. Both are gaining popularity and are being adopted by many facilities and institutions. FHIR has its own data model with a definition of a set of resources, while OpenEHR is more flexible and offers data modelling utilizing an archetype hierarchy (26).

These standards and frameworks provide a foundation for the development and implementation of EHRs. Healthcare organizations and EHR vendors must adhere to these standards to ensure that patient data is accurately represented, securely transmitted, and easily shared between different systems and healthcare providers, ultimately improving patient care and safety.

The objective of this paper is to study the challenges of RTD captured from EHRs, typical scenarios, their solutions and to provide reference implementations, indicating a use case example.

Methods

In this work, the common scenarios of capturing RTD in the healthcare from EHRs are identified. In order to show the feasibility of the concepts the approaches and tools are investigated to implement the real-time solutions regardless of the use case. The eligibility and the needed modifications to fit into the context of EHRs are examined. To develop reference program implementations Python and Go language (Golang) were used. Python offers an abundancy of libraries for server/client applications and Golang was specially developed for server-side networking applications.

The modern EHR standards have embraced the Representational State Transfer (REST) architecture, due to its raising use in all web applications in the recent years, thus, multiple REST application programming interfaces (APIs) solutions are available for each of them. The fact that these implementations use the Hypertext Transfer Protocol (HTTP) for communication makes REST more preferable, due to the undisturbed traffic through institutional firewalls. Usually, these APIs consist of a webserver, which processes the requests and stores data in a database, e.g., PostgreSQL, MySQL, Microsoft SQL, etc. (27-29).

RTD in REST

REST is an architectural style and set of constraints for designing networked applications. It was introduced and defined by Roy Fielding in 2000. REST is widely used in the design of web services and APIs due to its simplicity, scalability, and compatibility with the HTTP protocol (30). In his work, Roy Fielding lays the key principles of REST.

Resources are key components of REST APIs, which can be a conceptual entity or a piece of data. Each resource is identified by a unique Uniform Resource Identifier. RESTful interactions are performed using standard HTTP methods, also known as HTTP verbs. The primary HTTP methods used in REST are: GET (retrieve data or information about a resource), POST (create a new resource or submit data to a resource), PUT (update or replace an existing resource), DELETE (remove a resource). REST allows the use of intermediary components (proxies, gateways, caches) to improve system performance,

scalability, and security. Each component only needs to understand the immediate request/response, not the entire application (31).

RESTful APIs have become the dominant approach for building web services and have found widespread use in web and mobile applications due to their simplicity and compatibility with the HTTP protocol. They offer a scalable and loosely coupled way to design distributed systems on the web.

Delivering RTD to users by REST APIs is usually accomplished through a Publish-Subscribe approach. This allows clients to receive updates or notifications about resource changes in a real-time or event-driven manner (32). Clients subscribe to specific resources or events on the server. This subscription is typically established by the client sending a request to the server, expressing interest in certain resources or types of events (channels). The server acknowledges the subscription and associates the client with the specified resource or event. As resources change or events occur that are relevant to the client's subscription, the server detects these changes and sends notifications (33).

Common technologies and protocols used for implementing subscriptions are REST hooks, WebSocket and Server-Sent Events (34). The choice of technology depends on the specific requirements of the application and the desired level of real-time interaction between the server and the client. A REST server that does not support these approaches only follows a standard request-response model. In such a system, the only way to receive data is by making a HTTP request. The practices of HTTP polling could be considered a predecessor to the above-mentioned technologies and could enable real-time interactions. Despite of its limitations polling has remained for a long time on the web. The adoption of all these approaches is widespread and can be found in many of the web and mobile applications (35).

RTD independent of REST

Real-time communication is also possible outside of REST APIs. In this section, the approaches to separate the RTD delivery and management from the APIs are presented.

Many applications including REST APIs usually use a database solution to store the resources. Database triggers are a feature of relational database management systems and are usually used to automate actions or enforce data integrity rules within the database itself. Database trigger is a procedure which is executed as soon as an event in a database table occurs. This includes creation, deletion or modification of entries. Triggers are supported by most of the popular databases: Microsoft SQL, PostgreSQL, MySQL, SQLite, etc. (36). Database triggers can also be deployed in RTD scenarios to ensure data integrity, propagate changes, and automate certain actions when RTD events occur (37).

A reverse proxy is a server or software application which is located between the client devices and the backend servers. It acts as an intermediary, receiving requests from clients and forwarding them to the appropriate backend server. The reverse proxy then sends the server response back to the client. The key distinction between a reverse proxy and a regular (forward) proxy is the direction of the traffic flow. Reverse proxies are often used to enhance security, improve performance, and provide additional features for web applications (38). Currently a few solutions which offer an implementation for real time client interaction exist. These manage RTD delivery separating the implementation of the real-time service from the main backend API (39,40).

Results

Implementation description

RTD in REST

HL7 FHIR Rest APIs implement the REST hook. A Subscription resource needs to be created, which includes the criteria used to determine when to generate notifications and the rest hook channel. The criteria follow the same logic as if they were to be written as a uniform resource locator (URL) to search the resources of interest. When a resource meeting the criteria is created or updated the server sends a POST request with empty body or a PUT request with the full resource. In this scenario the recipient of the data has to be an HTTP endpoint which accepts the POST or PUT requests with a JavaScript Object Notation (JSON) body. After that if needed the data can be forwarded to other systems which need it in real-time allowing for a change in protocol and domain (*Figure 1*).

HL7 FHIR servers offer the use of WebSockets to send notifications to clients when an event occurs. To use WebSockets a Subscription resource has to be created like the REST hook mechanism, where the criteria and the WebSocket channel is specified. The recipient in this case is a client which connects to the server using the HTTP WebSocket technology. As soon as connection is

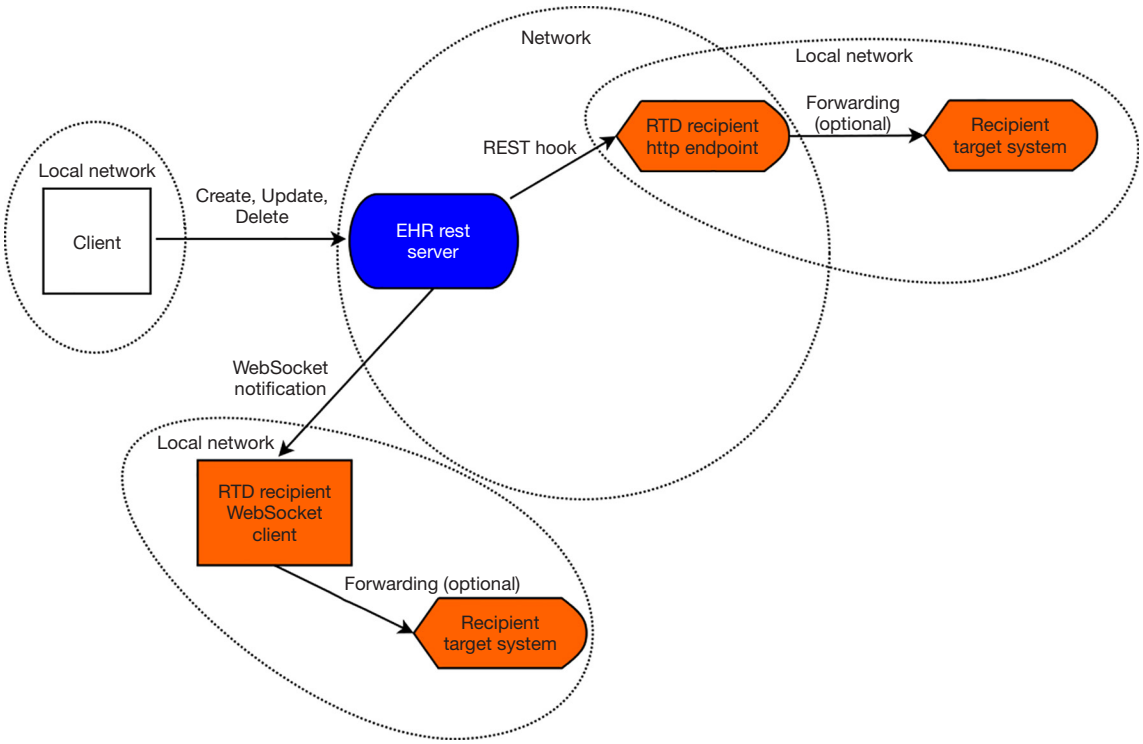

Figure 1 Diagram of the RTD delivery using Subscriptions over WebSocket or with REST hook. RTD, real-time data; REST, representational state transfer; EHR, electronic health record; HTTP, HyperText Transfer Protocol.

established the client sends a message "bind id" where id is the id of the subscription resource for which the client needs to be notified. The server responds than with a "bound id" message. From this moment the client receives a message "ping id" each time new data is available and has to query the resources and eventually forward them to a target system. The delivery of the data happens in real-time (*Figure 1*).

The HL7 FHIR describes two different approaches: polling a single record or polling across records. The polling of a single record is applicable only when there is an interest in the modification of an existing record. If the resource does not yet exist, polling across records should be used filtering by timestamp to identify new resources. Polling could be utilized with any other EHR standard, e.g., openEHR. The delay depends solely on the frequency of the queries and if it is short enough almost real-time delivery is achievable (*Figure 2*).

The polling interval defines how frequently the client sends requests to the server. The choice of polling interval depends on the specific requirements of the application. Shorter intervals provide more real-time updates but may increase server load and network traffic.

RTD independent of REST

Database triggers are applicable to all EHR servers using databases like OpenEHR and FHIR. The procedure code is written and inserted in the database server using a structured query language (SQL) query. When the defined event occurs the server notifies all clients that are listening on the particular channel. The recipient has to be connected to the database server and has to be listening for notify events. These can then be forwarded to another system. The delivery in this case is in real-time (*Figure 3*).

It's important to note that while database triggers can be useful in RTD scenarios, they should be designed carefully to avoid performance bottlenecks and potential issues, especially in high-velocity RTD environments. Additionally, the choice of database system and its support for real-time processing can impact the effectiveness of using triggers in such scenarios.

Reverse proxies can be used to track and forward requests to the EHR servers. The RTD capturing could be achieved by notifying a recipient when an HTTP request using the

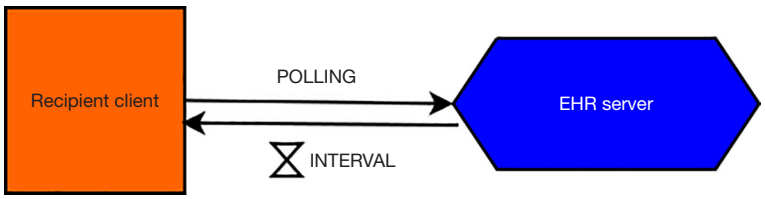

Figure 2 Diagram of the RTD delivery using Polling. EHR, electronic health record; RTD, real-time data.

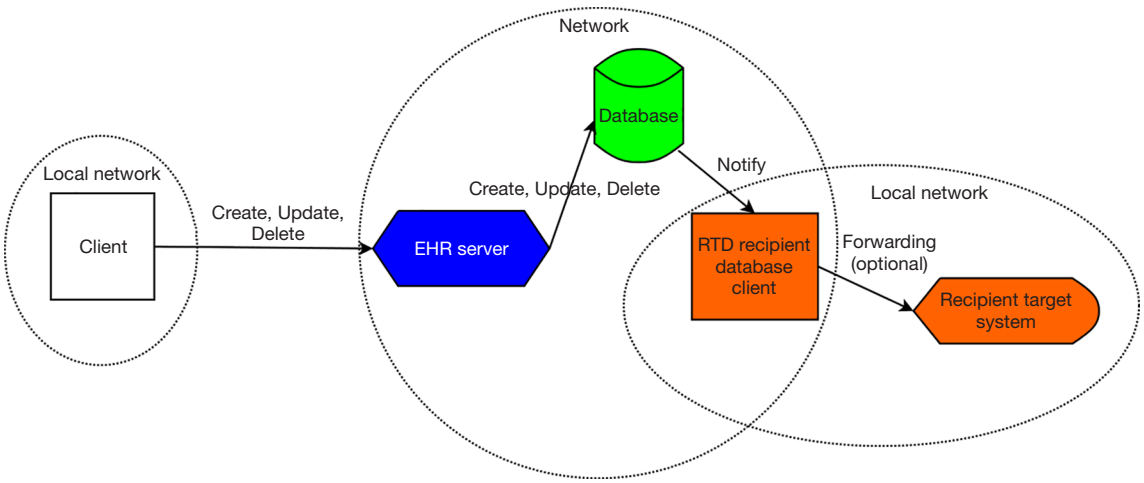

Figure 3 Diagram of the RTD delivery using database triggers. EHR, electronic health record; RTD, real-time data.

POST, PUT or DELETE method is passed (*Figure 4*). In this scenario, the encryption and authentication should be handled by the proxy as well as the identity of the requesting client. It is important to note that the server should be aware of the URL of the reverse proxy in order to adjust the resource URLs (base URL).

Reference implementation

A reference implementation of the solutions for each scenario is made available in a GitHub repository. For HL7 FHIR a python script is provided to create a sample Subscription resource to send RTD when a new Observation resource for specific patient id is created. The sample WebSocket client is written using the "websocket-client" python library. The sample HTTP endpoint is created using the "Flask" framework. For database triggers a sample SQL query for Postgres is made available to create a trigger when an INSERT or UPDATE operation is executed on the FHIR "hfj_resource" table. The trigger notifies the connected clients about the operation, resource type and resource id. The database client is implemented using the "psycopg2" library. Finally, for the reverse proxy Golang's "httputil" package is used. It allows the modification of the response before it reaches the requesting client. In this case the response is not modified and the information is used to check the request method that created the response and the status code. If the request method is "POST", "PUT" or "DELETE" and if the status code is successful, an operation producing data update on the server is identified. In this step a new request to a RTD recipient, e.g., an endpoint is executed. More information regarding the implementation and usage could be found in the README.MD file.

Use case clinical example

In this use case example of RTD capture from the EHRs the main actors are the healthcare providers [hospitals, physician private practices, general practitioners (GPs), medical laboratories], health information networks (HIN) and patients. All of the above-mentioned settings have their own private networks with EHR servers and databases protected by a firewall. In the hospital network a reverse

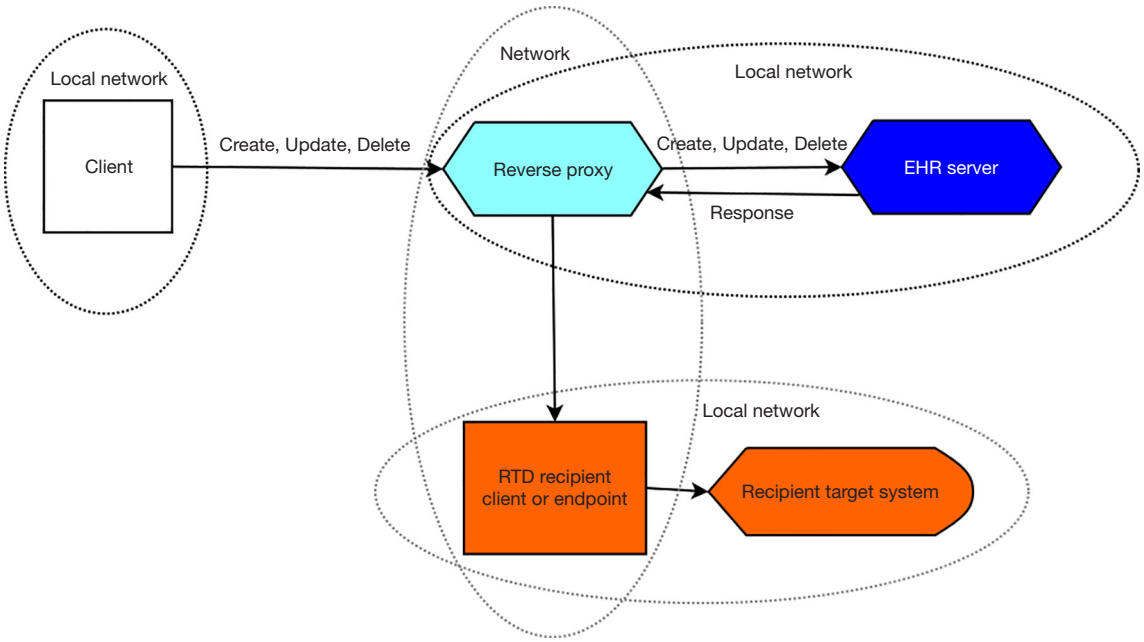

Figure 4 Diagram of the RTD delivery using reverse proxy. EHR, electronic health record; RTD, real-time data.

proxy could be placed in front of the EHR server and when new records are generated these will be sent to a HIN in real-time. The RTD delivery from a physician private practice could be implemented using a database trigger. Medical laboratories could also deliver RTD by connecting a REST hook to the HIN. As soon as new information is received by the HIN, patients can receive RTD through a WebSocket mobile client and the GP will be informed about the new patient data through REST hook to a HTTP Endpoint positioned in the demilitarized zone behind the firewall. GPs could send RTD to the HIN as well. The selection of the implementation approach and technology here is just an example and in real-life this process should be assessed according to the needs and design of the network and information technology infrastructure of the facility (*Figure 5*).

A hypothetical scenario could be a patient, who is admitted to a hospital and diagnosed with a chronic condition, which requires managing and regular visits at a physician private practice. The clinicians generate the new EHRs regarding this encounter and the information is sent in real-time to the HIN, which forwards the RTD to the GP. The GP can than expeditiously prepare and write a referral for specialist care and other documents to the HIN saving the patient troubles appointing a visit. The patient will receive the generated records on his phone as RTD and can visit the private practice of a specialist physician, who will examine the patient and assign additional laboratory testing, treatment and complete the EHR of the patient by sending RTD to the HIN. The GP is able to issue receipts and observe the laboratory results in real-time. This will be true for the patient as well and should reduce the amount of patient visits, waiting times and costs.

Discussion

The solutions to achieve RTD delivery depend mostly on the technology which the EHR server uses. HL7 FHIR uses a REST API and implements many of the notification features. The only work which discusses the use of the REST subscriptions mechanism in the context of EHRs presents a real-time link between a FHIR server to Observational Medical Outcomes Partnership (OMOP) and Patient-Centered Clinical Research Network (PCORnet) databases for COVID-19 research (41). The authors do not show in detail how this mechanism is implemented. REST hooks are effective delivering RTD, but need an endpoint address which is exposed and accessible from the EHR server. If both the server and the recipient of data are residing in the same local network then this may not be needed. On the contrary, WebSockets do not need outward-facing HTTP endpoint and relay on a Transmission

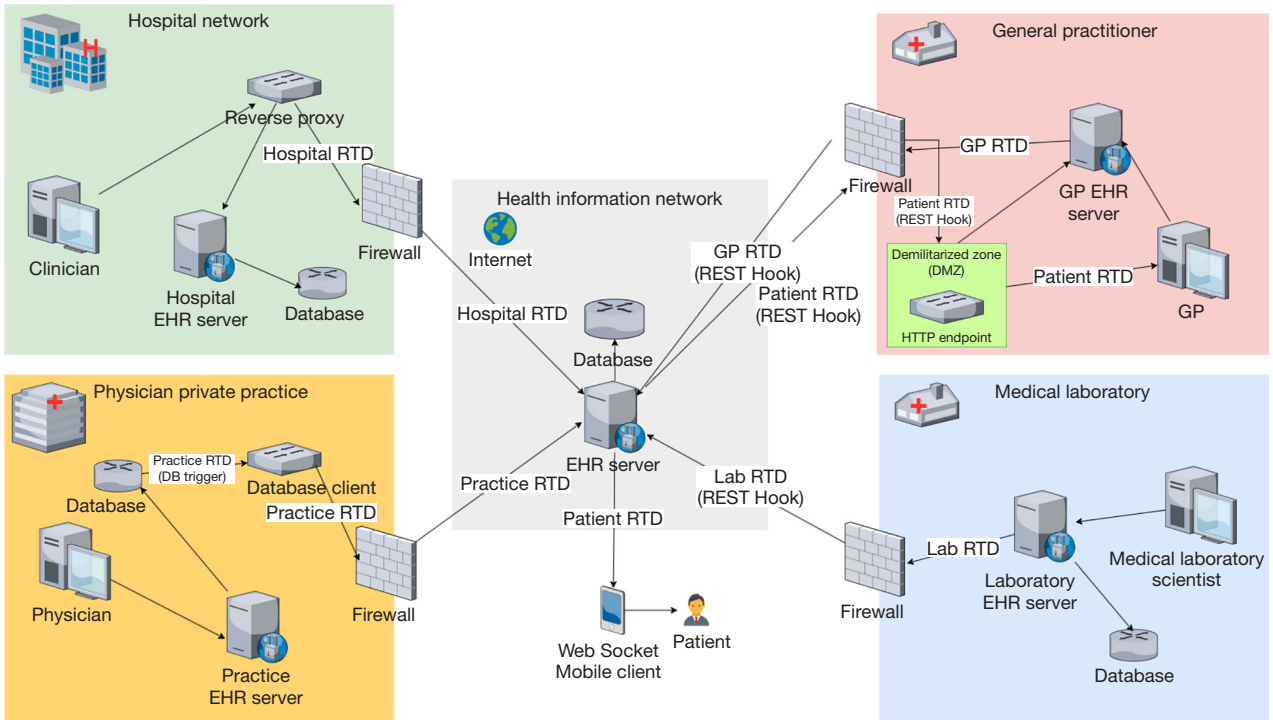

Figure 5 Use case model diagram: relationship and RTD exchange between healthcare providers (hospitals, physician private practices, GPs, medical laboratories), health information networks and patients. EHR, electronic health record; REST, representational state transfer; GP, general practitioner; Lab, laboratory; HTTP, HyperText Transfer Protocol; DB, database; RTD, real-time data.

Control Protocol/Internet Protocol (TCP/IP) connection on top of HTTP, which is established and maintained.

If WebSockets or REST hooks are not supported by the server the recipient needs to resort to other approaches. In these scenarios database trigger or reverse proxy are the next solution which offers best RTD availability. By searching the literature, we did not find any application of database triggers or reverse proxies for RTD delivery in the context of EHRs. Database triggers require an access to the database system in order to execute the SQL query creating the trigger. Moreover, the database server needs to be accessible by the listening recipient. The reverse proxy solution calls for a more sophisticated intermediary to forward requests and handle encryption and authentication. The reliability of this solution might be higher than that of the database triggers avoiding bottleneck and performance issues. Furthermore, reverse proxies offer additional performance features. Although, reverse proxies that manage real-time features already exist, their use is mainly to separate the implementation of real-time messaging from the backend API or other microservices. This allows the backend to remain stateless, but requires the ability of the API to communicate with the reverse proxy and be aware of its existence (39,40). In our proposed reverse proxy design, there is no need for modification of the EHR Server and the solution can universally be deployed in many scenarios. Some reverse proxy servers offer a mirror module which mirrors the original request to another backend (42). These features do not access the response from the EHR server and can't guarantee the success of the request.

If none of these approaches can be used the only solution remains polling. To achieve almost RTD delivery, it is important to increase the frequency of the queries. This may increase bandwidth and consume processing power from the server. It is recommended to use polling if all other solutions are unavailable (35).

Conclusions

Capturing RTD is undoubtedly vital for health professionals and successful digital healthcare. The topic remains unexplored especially in the context of EHRs. In this work for the first time the common scenarios and problems are investigated. Furthermore, solutions and reference

implementations which could support and contribute to the development of real-time applications are provided. Ultimately, as RTD requirements of each scenario can vary significantly, it's advisable to carefully assess the specific needs of the application and choose appropriate technologies and strategies accordingly.

Acknowledgments

Funding: None.

Footnote

Data Sharing Statement: Available at https://mhealth.amegroups.com/article/view/10.21037/mhealth-24-2/dss

Peer Review File: Available at https://mhealth.amegroups.com/article/view/10.21037/mhealth-24-2/prf

Conflicts of Interest: The author has completed the ICMJE uniform disclosure form (available at https://mhealth.amegroups.com/article/view/10.21037/mhealth-24-2/coif). The author has no conflicts of interest to declare.

Ethical Statement: The author is accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

1. Hempel S, Ganz D, Saluja S, et al. Care coordination across healthcare systems: development of a research agenda, implications for practice, and recommendations for policy based on a modified Delphi panel. BMJ Open 2023;13:e060232.
2. Schlicher J, Metsker MT, Shah H, et al. From NASA to healthcare: real-time data analytics (mission control) is reshaping healthcare services. Perspect Health Inf Manag 2021;18:1g.
3. Evans RS. Electronic Health Records: Then, Now, and in the Future. Yearb Med Inform 2016;Suppl 1:S48-61.
4. Hämäläinen A, Hirvonen H. Electronic Health Records reshaping the socio-technical practices in Long-Term Care of older persons. Technology in Society 2020;62:101316.
5. Gliklich RE, Leavy MB, Dreyer NA, editors. Tools and Technologies for Registry Interoperability, Registries for Evaluating Patient Outcomes: A User's Guide, 3rd Edition, Addendum 2. Rockville (MD): Agency for Healthcare Research and Quality (US); 2019 Oct. Report No.: 19(20)-EHC017-EF.
6. Northern Kentucky University. Northern Kentucky University; c2024 [cited 2024 Jan 04]. Benefits of Real-Time Data in Health Information Systems. Available online: https://onlinedegrees.nku.edu/programs/business/informatics/mshi/real-time-data/
7. Shin KG, Ramanathan P. Real-time computing: a new discipline of computer science and engineering. Proceedings of the IEEE 1994;82:6-24.
8. Laska M, Herle S, Klamma R, et al. A Scalable Architecture for Real-Time Stream Processing of Spatiotemporal IoT Stream Data—Performance Analysis on the Example of Map Matching. ISPRS Int J Geo-Inf 2018;7:238.
9. Mintz Y, Brodie R. Introduction to artificial intelligence in medicine. Minim Invasive Ther Allied Technol 2019;28:73-81.
10. Paganelli AI, Mondéjar AG, da Silva AC, et al. Real-time data analysis in health monitoring systems: A comprehensive systematic literature review. J Biomed Inform 2022;127:104009.
11. Bouri N, Ravi S. Going mobile: how mobile personal health records can improve health care during emergencies. JMIR Mhealth Uhealth 2014;2:e8.
12. Fu MR. Real-time detection and management of chronic illnesses. Mhealth 2021;7:1.
13. Cruz R, Guimarães T, Peixoto H, et al. Architecture for Intensive Care Data Processing and Visualization in Real-time. Procedia Computer Science 2021;184:923-8.
14. Gold R, Sheppler C, Hessler D, et al. Using Electronic Health Record-Based Clinical Decision Support to Provide Social Risk-Informed Care in Community Health Centers: Protocol for the Design and Assessment of a Clinical Decision Support Tool. JMIR Res Protoc

2021;10:e31733.
15. Wright A, Aaron S, Seger DL, et al. Reduced Effectiveness of Interruptive Drug-Drug Interaction Alerts after Conversion to a Commercial Electronic Health Record. J Gen Intern Med 2018;33:1868-76.
16. Classen D, Li M, Miller S, et al. An Electronic Health Record-Based Real-Time Analytics Program For Patient Safety Surveillance And Improvement. Health Aff (Millwood) 2018;37:1805-12.
17. Porterfield A, Engelbert K, Coustasse A. Electronic prescribing: improving the efficiency and accuracy of prescribing in the ambulatory care setting. Perspect Health Inf Manag 2014;11:1g.
18. Kalid N, Zaidan AA, Zaidan BB, et al. Based on Real Time Remote Health Monitoring Systems: A New Approach for Prioritization "Large Scales Data" Patients with Chronic Heart Diseases Using Body Sensors and Communication Technology. J Med Syst 2018;42:69.
19. van Kuppenveld SI, van Os-Medendorp H, Tiemessen NA, et al. Real-Time Access to Electronic Health Record via a Patient Portal: Is it Harmful? A Retrospective Observational Study. J Med Internet Res 2020;22:e13622.
20. Satterfield BA, Dikilitas O, Kullo IJ. Leveraging the Electronic Health Record to Address the COVID-19 Pandemic. Mayo Clin Proc 2021;96:1592-608.
21. Menachemi N, Collum TH. Benefits and drawbacks of electronic health record systems. Risk Manag Healthc Policy 2011;4:47-55.
22. Kalra D. Electronic health record standards. Yearb Med Inform 2006;136-44.
23. Reisman M. EHRs: The Challenge of Making Electronic Data Usable and Interoperable. P T 2017;42:572-5.
24. Winter A, Takabayashi K, Jahn F, et al. Quality Requirements for Electronic Health Record Systems*. A Japanese-German Information Management Perspective. Methods Inf Med 2017;56:e92-104.
25. Ayaz M, Pasha MF, Alzahrani MY, et al. The Fast Health Interoperability Resources (FHIR) Standard: Systematic Literature Review of Implementations, Applications, Challenges and Opportunities. JMIR Med Inform 2021;9:e21929.
26. Kryszyn J, Smolik W, Wanta D, et al. "Comparison of OpenEHR and HL7 FHIR Standards." International Journal of Electronics and Telecommunications (2023): 47-52.
27. Mandel JC, Kreda DA, Mandl KD, et al. SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. J Am Med Inform Assoc 2016;23:899-908.
28. HAPI FHIR. HAPI FHIR Server Introduction. [cited 2024 Jan 04]. Available online: https://hapifhir.io/hapi-fhir/docs/server_plain/
29. OpenEHR. OpenEHR REST EHR API. [cited 2024 Jan 04]. Available online: https://specifications.openehr.org/releases/ITS-REST/Release-1.0.0/ehr.html
30. Fielding RT. Chapter 2: Network-based Application Architectures. Architectural Styles and the Design of Network-based Software Architectures (Ph.D.) 2000. University of California, Irvine. Archived from the original on 2014-12-16. Retrieved 2014-04-12.
31. Erl T, Carlyle B, Pautasso C, et al. SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST. Upper Saddle River. New Jersey: Prentice Hall, 2012.
32. Google. Google Cloud; c2024 [cited 2024 Jan 04]. What is Pub/Sub? Available online: https://cloud.google.com/pubsub/docs/overview
33. Google. Google Cloud; c2024 [cited 2024 Jan 04]. Overview of the Pub/Sub service. Available online: https://cloud.google.com/pubsub/docs/pubsub-basics
34. Zapier. Zapier Inc.; c2022-2024 [cited 2024 Jan 04]. What are webhooks? Available online: https://zapier.com/blog/what-are-webhooks/
35. Murley P, Ma Z, Mason J, et al. WebSocket Adoption and the Landscape of the Real-Time Web. Proceedings of the Web Conference 2021. Association for Computing Machinery, New York, NY, USA, 1192–1203. doi: 10.1145/3442381.3450063.
36. Berndtsson M, Mellin J. Database Trigger. In: Liu L, Özsu MT. editors. Encyclopedia of Database Systems. New York, NY: Springer; 2018.
37. Estuary. estuary.dev; c2023-2024 [cited 2024 Jan 04]. Realtime Database Triggers. Available online: https://estuary.dev/realtime-database-triggers/
38. Cloudflare. Cloudflare, Inc.; c2024 [cited 2024 Jan 04]. What is a reverse proxy? | Proxy servers explained. Available online: https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/
39. pushpin.org. Fastly, Inc; c2024 [cited 2024 Jan 04]. Available online: https://pushpin.org/
40. fanout.io. Fanout Now part of Fastly; c2024 [cited 2024 Jan 04]. Available online: https://fanout.io/
41. Lenert LA, Ilatovskiy AV, Agnew J, et al. Automated Production of Research Data Marts from a Canonical Fast Healthcare Interoperability Resource (FHIR) Data

Repository: Applications to COVID-19 Research. medRxiv [Preprint] 2021. doi: 10.1101/2021.03.11.21253384. Update in: J Am Med Inform Assoc 2021;28:1605-11.

42. Nginx. Nginx; c2024 [cited 2024 Jan 04]. Module ngx_http_mirror_module. Available online: https://nginx.org/en/docs/http/ngx_http_mirror_module.html

doi: 10.21037/mhealth-24-2
Cite this article as: Kirilov N. Capture of real-time data from electronic health records: scenarios and solutions. mHealth 2024;10:14.