Federation Time

Manuel Esménio
Ponder Source
December 2022

***Abstract*— [1] This paper pretends to cover the theoretical grounding for a federated system in the context of task tracking, by addressing...**

I. Background

A. Federation

A **federation** is a group of systems which share data in peer-to-peer transmissions where information flows from a node to its neighbours, while ensuring each individual node maintains **data sovereignty**, or the ability to choose the access rights of other nodes in federation, thus being able to arbitrarily veto external accesses and modifications to its own data.

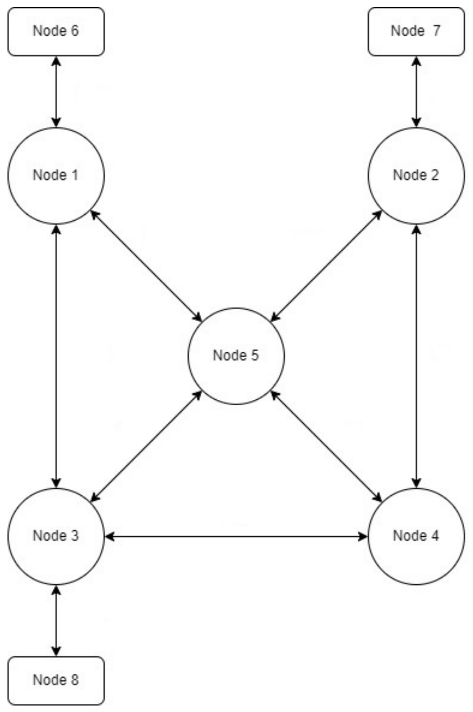

Fig. 1: Example of a federated system

A federation can have both federated and non-federated members. A **Non-federated member** must have only one neighbour (nodes 6-8 in Fig. 1), and acts as a repository of data (e.g: an API). A **federated member** must have at least two neighbours (nodes 1-5 in Fig. 1) to forward data as part of a chain of messages without causing duplication issues.

B. Format translation

Part of the problem when building a federated system is the assumption that every node can choose its own syntax. Consider the problem where we want to transfer information between two data repositories, where each uses one of the formats shown in figures 2 and 3, which were copied from the federated timesheets project presented in section II:

- **Customer** — "Customer1"
- **Project** — "Development"
- **Task** — " "
- **Start time** — "2022-03-30 13:01:00"
- **Stop time** — "2022-03-30 13:25:38"
- **Minutes** — "24"
- **Note** — " "

Fig. 2: Data fields in format "jiffy-CSV"

- **Date** — "2022-03-30"
- **From** — "14:32"
- **To** — "15:00"
- **Duration** — "1680"
- **Rate** — "KYD 10.00"
- **Internal rate** — "0"
- **User** — "admin"
- **Name** — "admin"
- **Customer** — "Flatley, Purdy and Gerlach"
- **Project** — "Dolore sit"
- **Activity** — "Ea illo"
- **Description** — "test"
- **Exported** — "no"
- **Billable** — "yes"
- **Tags** — "a1,v2"
- **Hourly rate** — "KYD 12.00"
- **Fixed rate** — "KYD 10.00"
- **Type** — "timesheet"
- **label.category** — "work"
- **Account** — "C-37165956"
- **VAT-ID** — "6011598336445140"
- **Order number** — "P-91111562"

Fig. 3: Data fields in format "kimai-CSV"

From these formats it is observable that: Some fields can be copy pasted between formats, such as "customer" and "project"; some fields can be reconstructed, for example one can use the "Date" and "From" fields in kimai-CSV and assume the task always starts at 0 seconds to get the "Start time" field from jiffy-CSV, which would probably be acceptable in most use cases in spite of the loss of information; some fields have to be discarded due to one of the formats not containing that information, like the billing information fields which are present in kimai-CSV and not in jiffy-CSV. Since we can't make any assumptions about the language, we can't even be sure *a priori* that the fields "customer" and "project" are meant to contain the same information, even though they have the same name.

From this example I would like to draw the following conclusions:

- In order for two nodes of a federated system to communicate, there must be some logic that handles translating the data being communicated between them.
- The loss of information that can potentially arise from the communication between two nodes needs to be expected and accounted for.
- Even though an human can design and optimize translation functions, doing so requires some meta-knowledge of the task at hand, such as knowing these formats refer to timesheet tracking apps, which can make the automatization of translation logic for any given pair of nodes a convoluted task.

C. *Neighbourhood*

In a federated system, the neighbourhood of a node refers to all other nodes it can communicate with (in figure 1 the neighbourhood of node 1 is nodes 3,5,6). These communication channels are expected to allow two-way communication and each require their own translation logic, which to summarize what was said in subsection I-B means there is a variable cost to opening each channel.

Consider the problem where we have the federated system schematized in figure 1 and we want to transfer data from node 6 to node 7. Using the available communication channels and assuming the message never goes back to a node it has passed through, the possible routes are:

- $6 \to 1 \to 5 \to 2 \to 7$
- $6 \to 1 \to 5 \to 4 \to 2 \to 7$
- $6 \to 1 \to 5 \to 3 \to 4 \to 2 \to 7$
- $6 \to 1 \to 3 \to 4 \to 2 \to 7$
- $6 \to 1 \to 3 \to 4 \to 5 \to 2 \to 7$
- $6 \to 1 \to 3 \to 5 \to 4 \to 2 \to 7$

Fig. 4: Routes from node 6 to node 7

This means that using a strategy where each node forwards the information to all its neighbours, which would be the most resilient for ensuring the information eventually gets to the desired point, requires being able to uniquely identify data transmissions throughout the federation channels, so that the system can prevent the deriving issues of data duplication and infinite message looping.

Altough intuitively the shorter route would be more desirable, there is no guarantee of that, nor that node 5 will not randomly disconnect and require the federation to temporarily operate without it, meaning the neighbourhoods in federated systems have to be carefully designed to prevent geometries that rely on central (and fallible) nodes, while minimizing loss of information and average transmission times across the system.

D. *Open/Closed federation*

When one describes the functioning of a federated system one tends to describe a **open federation**:

- Decentralized data transfer system;
- Each node has data sovereignty;
- Variable and unknown members;
- Each node only has information about their neighbours;
- No rules, languages or protocols are applied on a global scale;
- All issues associated with forwarding data between nodes (duplication, translation, conflicting data, storms) are solved locally by the pairs of neighbours involved;

Fig. 5: Core ideas of open federation

When one goes to implement a federated system, technological challenges and time restraints often require a solution that does not implement all the requirements in figure 5, or a **closed federation** system: if there is a fixed and known set of members, we can assume they all trust eachother, which greatly simplifies the problems of transparency and security; if we assume a global rule where data conflicts are solved using "last-write wins" policy, then there is no need for nodes to write their own conflict resolution protocols for all their neighbours, nor their own format translation protocols, which can be hand-crafted for a known and finite number of members.

The concept of an open federation serve as direction and motivation for the projects currently being developed. While current limitations require our systems to partially operate as a closed federation, the reasoning behind the simplifications used should be explained as part of the implementation details, to set the ground for future work.

II. Previous work

III. Current iteration

IV. Future work

V. Glossary

A. *[PLACEHOLDER]*

In a peer-to-peer data sharing system it feels important to have a quantification of how reliable is the data being transmitted at any given point and time. Such a metric would enable individual nodes to choose between conflicting data (pick the one with highest value), or even to decide whether or not to accept a transmission of data (check if value is above/below a threshold value). In

References

[1] E. A. Feinberg, "On measurability and representation of strategic measures in markov decision processes," *Lecture Notes-Monograph Series*, vol. 30, pp. 29–43, 1996. [Online]. Available: http://www.jstor.org/stable/4355936