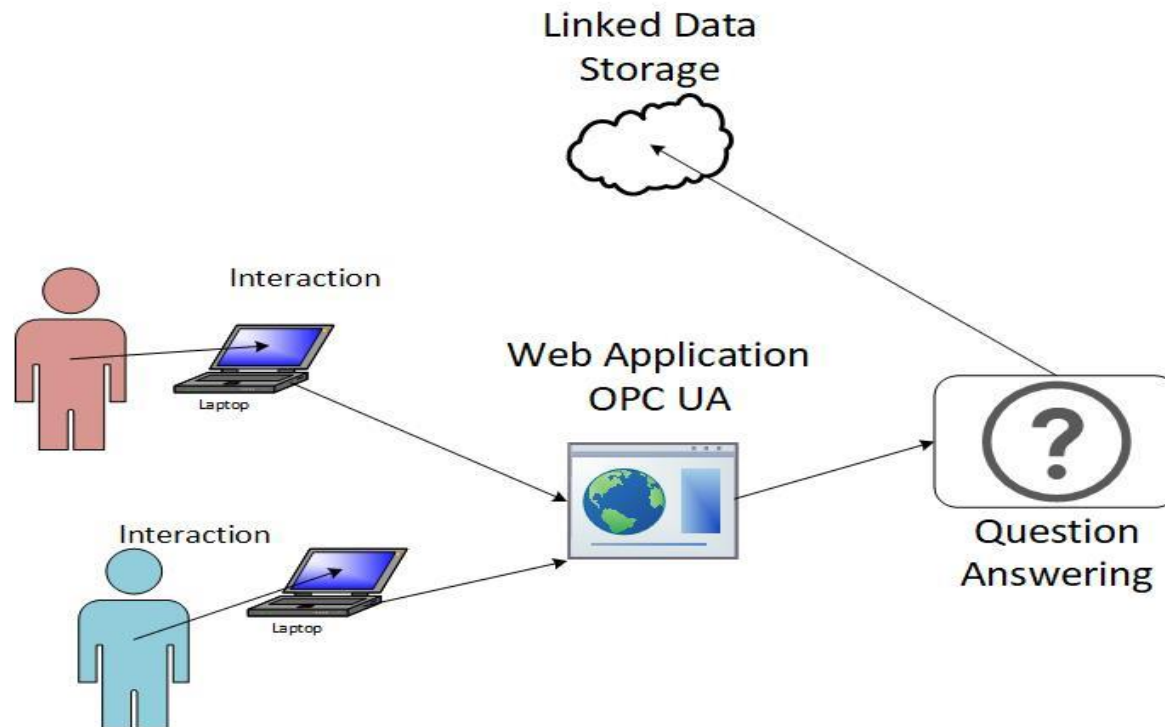


# Design and implementation of a Web-Based Software for the OPC UA Protocol integrated to a Semantic Question Answering System



# INTRODUCTION

- Motivation, Scope, and Goals
- Research Questions
- Introduction to OPC History
- The OPC UA Web-Based Application
- Serialization from OPC UA Protocol and Streamed Data
- The Semantic Question Answering
- Evaluation and Conclusion

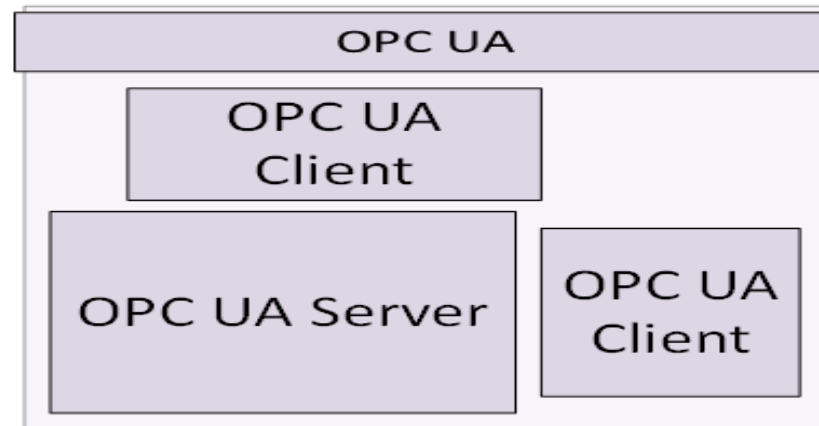
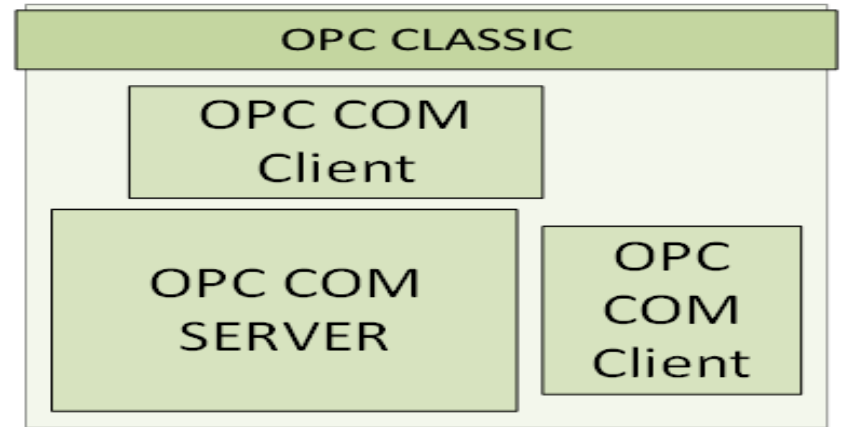
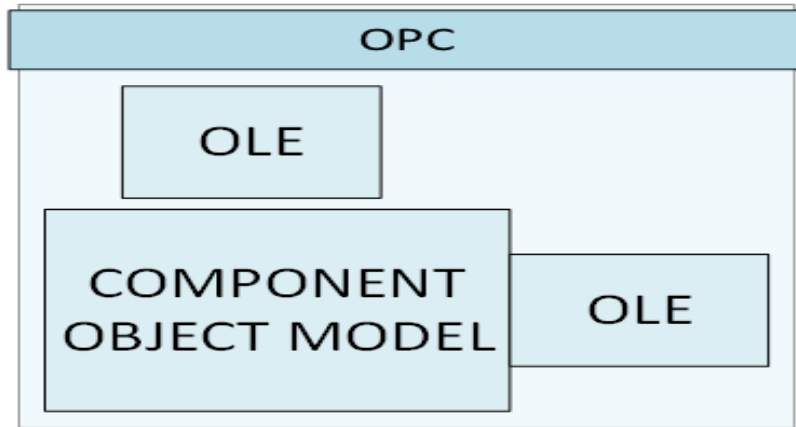
# MOTIVATION AND GOALS

- Implement the RESTful architecture aspect of a smart factory.
- Apply generated linked data by smart factories and regarding protocol suite.
- Create a minimum viable product a web-based tool for OPC-UA that integrated into a Semantic Question Answering to deploy in a smart factory.
- Evaluate the web-based application with predefined criterions

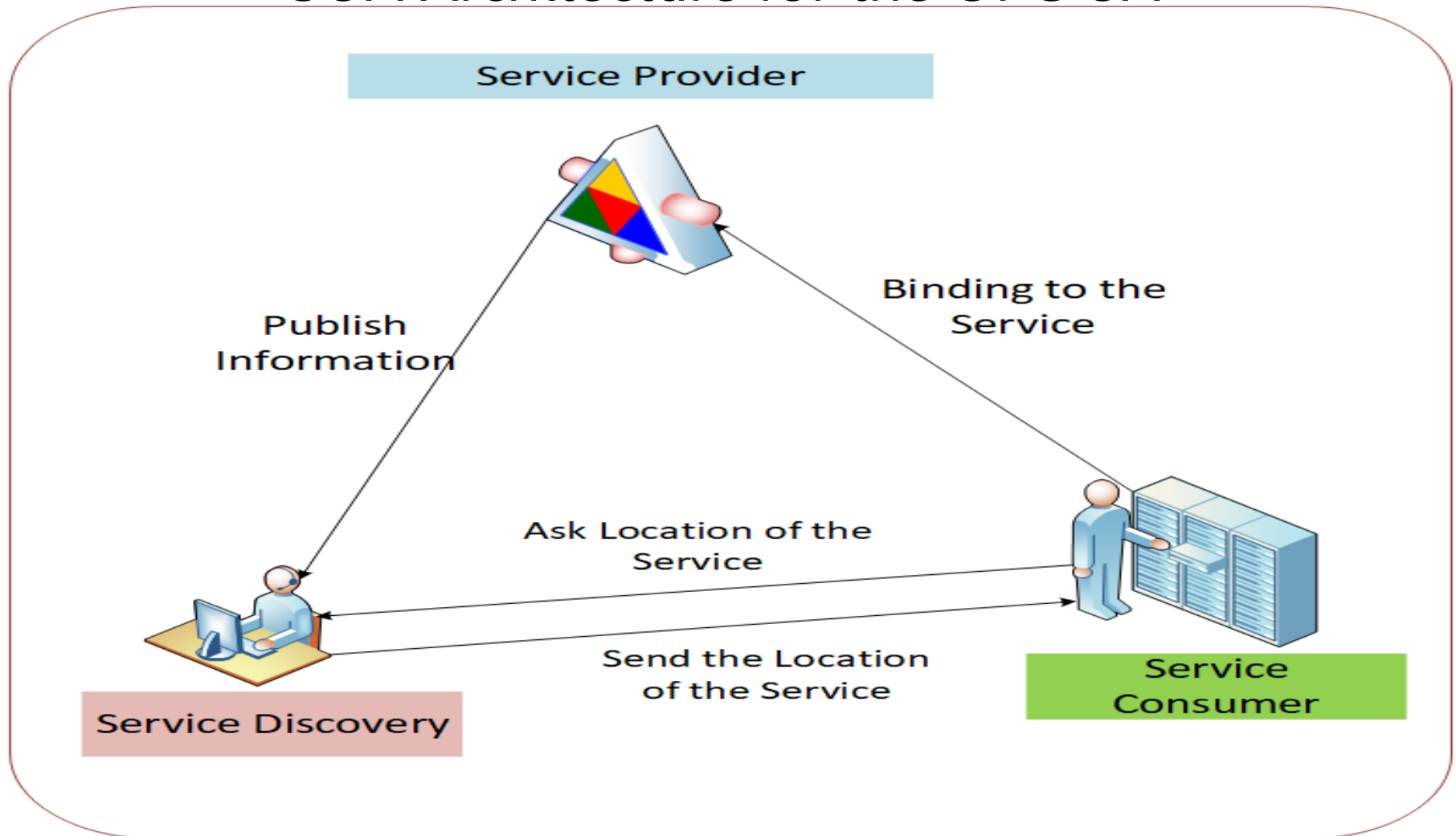
# RESEARCH QUESTIONS

1. How can a semantic question answering integrate to OPC UA Web-based software?
2. Is the research capable of establishing a new foundation for an OPC UA REST Integration and a Question Answering System
3. What are the requirements of OPC UA Web-based software integrated into the semantic question answering?
4. How does assess and compare an OPC UA Web-based software integrated into the semantic question answering?
5. What are the characteristics associated with the architecture of general application?
6. What are the key components of the approach and how did the research contribute to the research area?

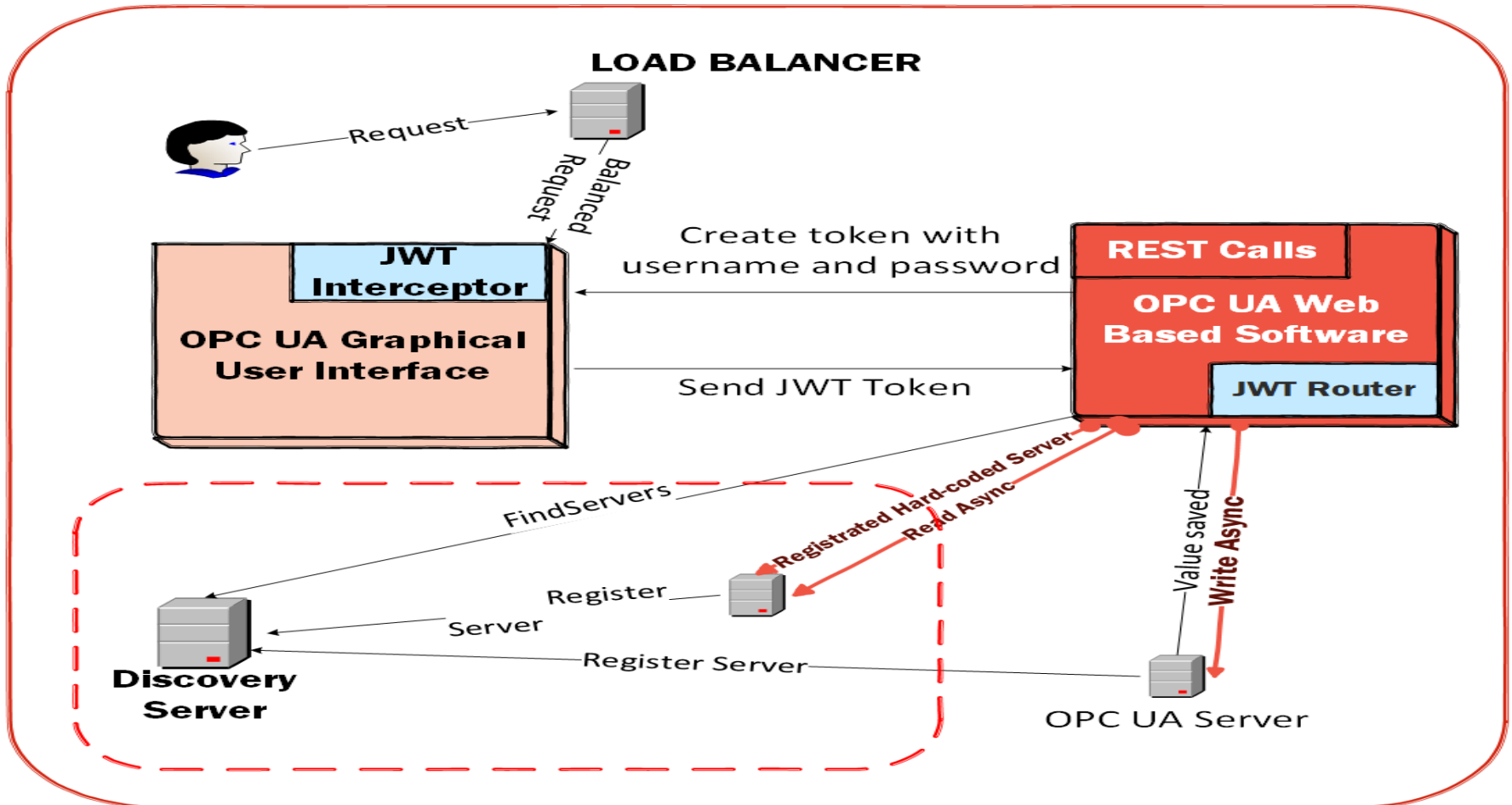
# INTRODUCTION TO OPC HISTORY



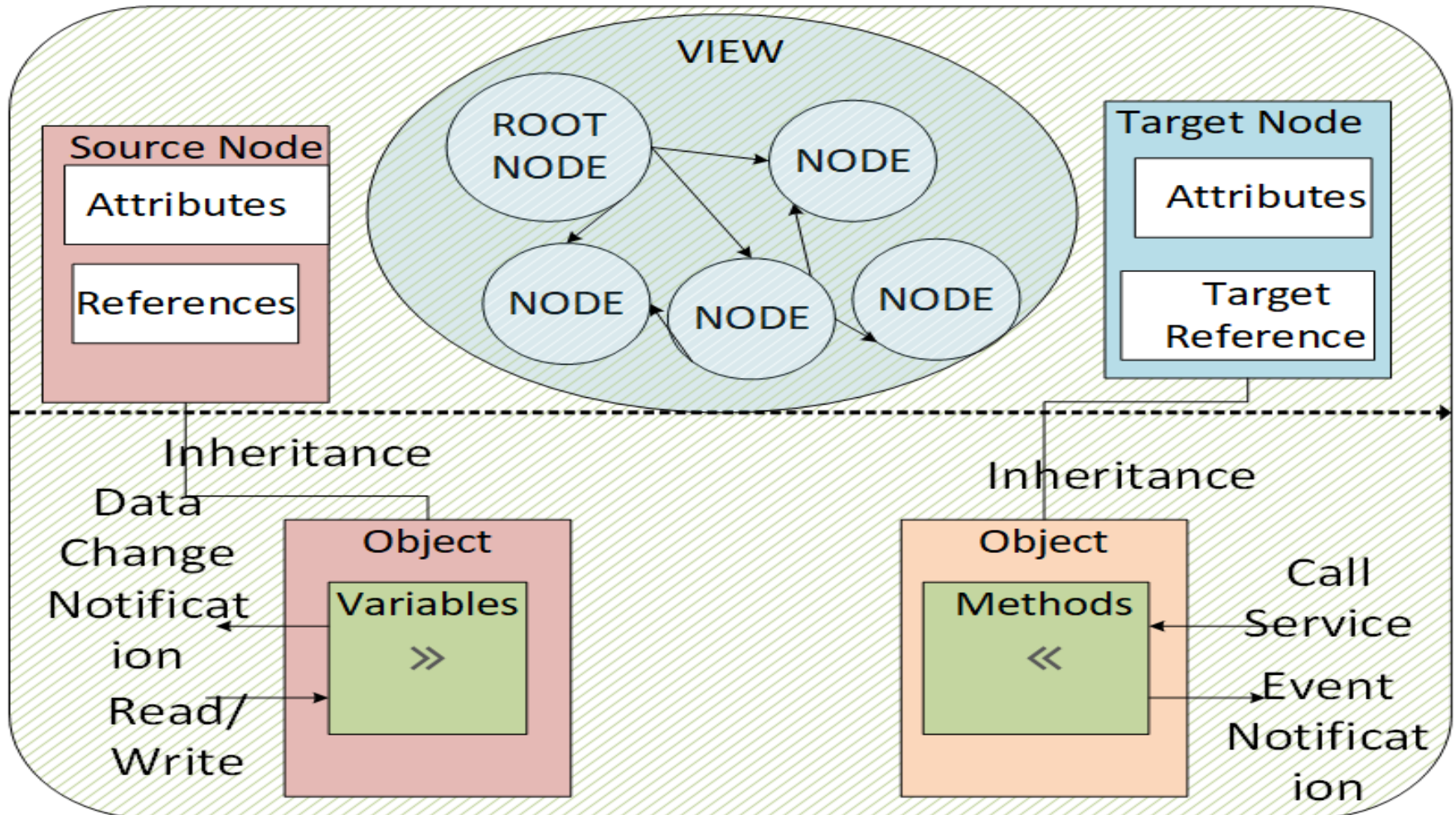
# SOA Architecture for the OPC UA



# The Web-based Application for the OPC UA

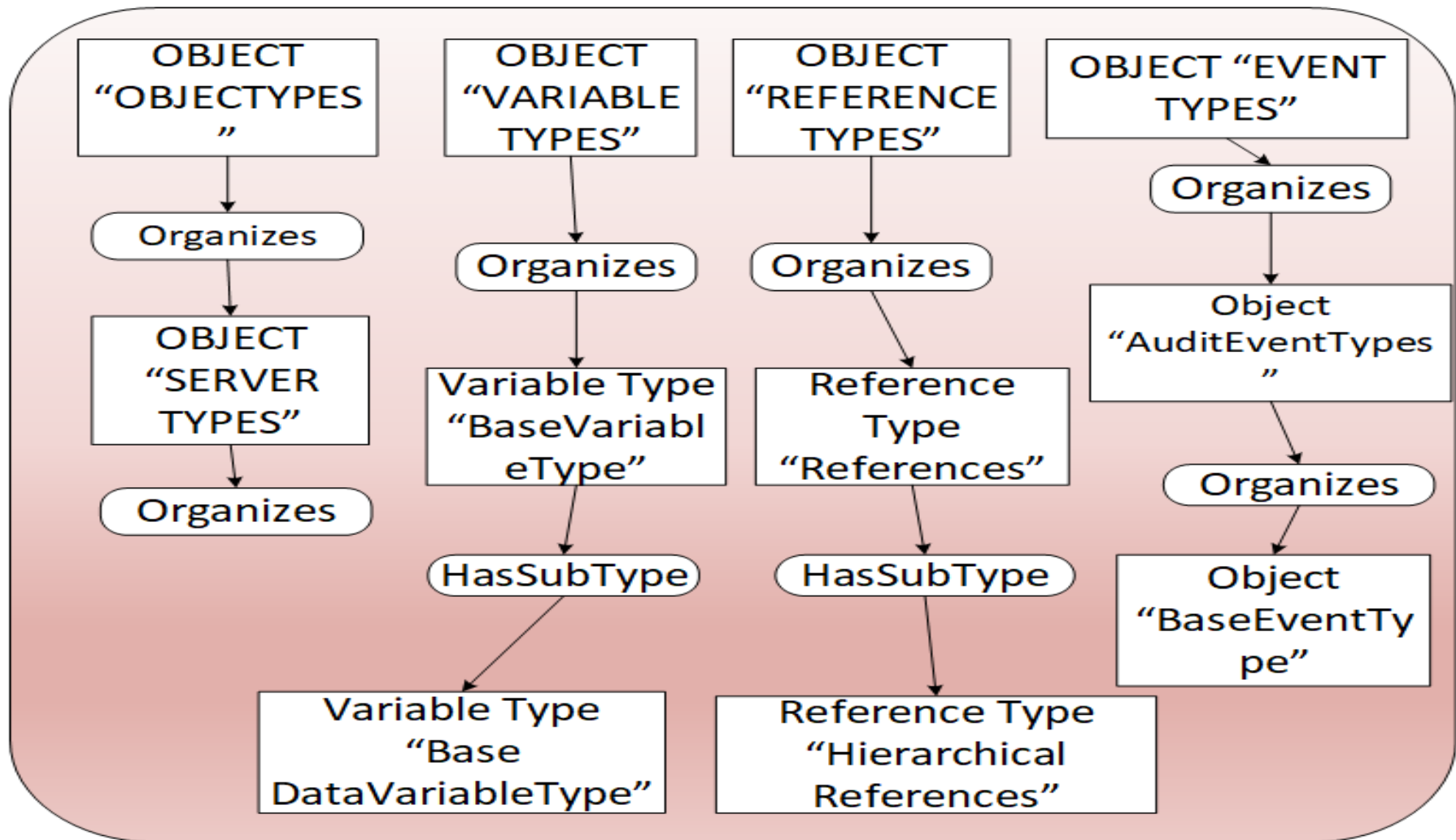


# ADDRESS SPACE MODEL

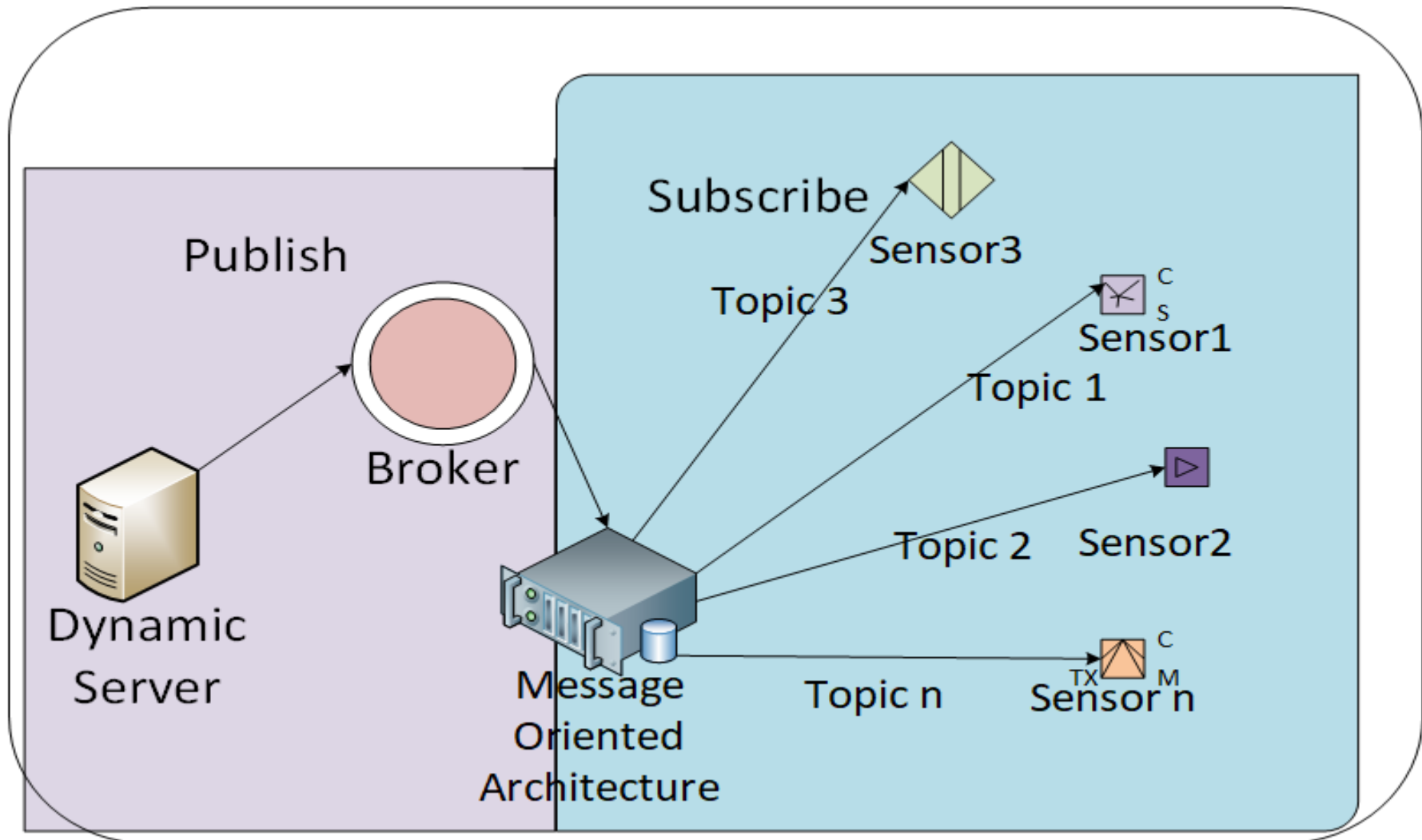




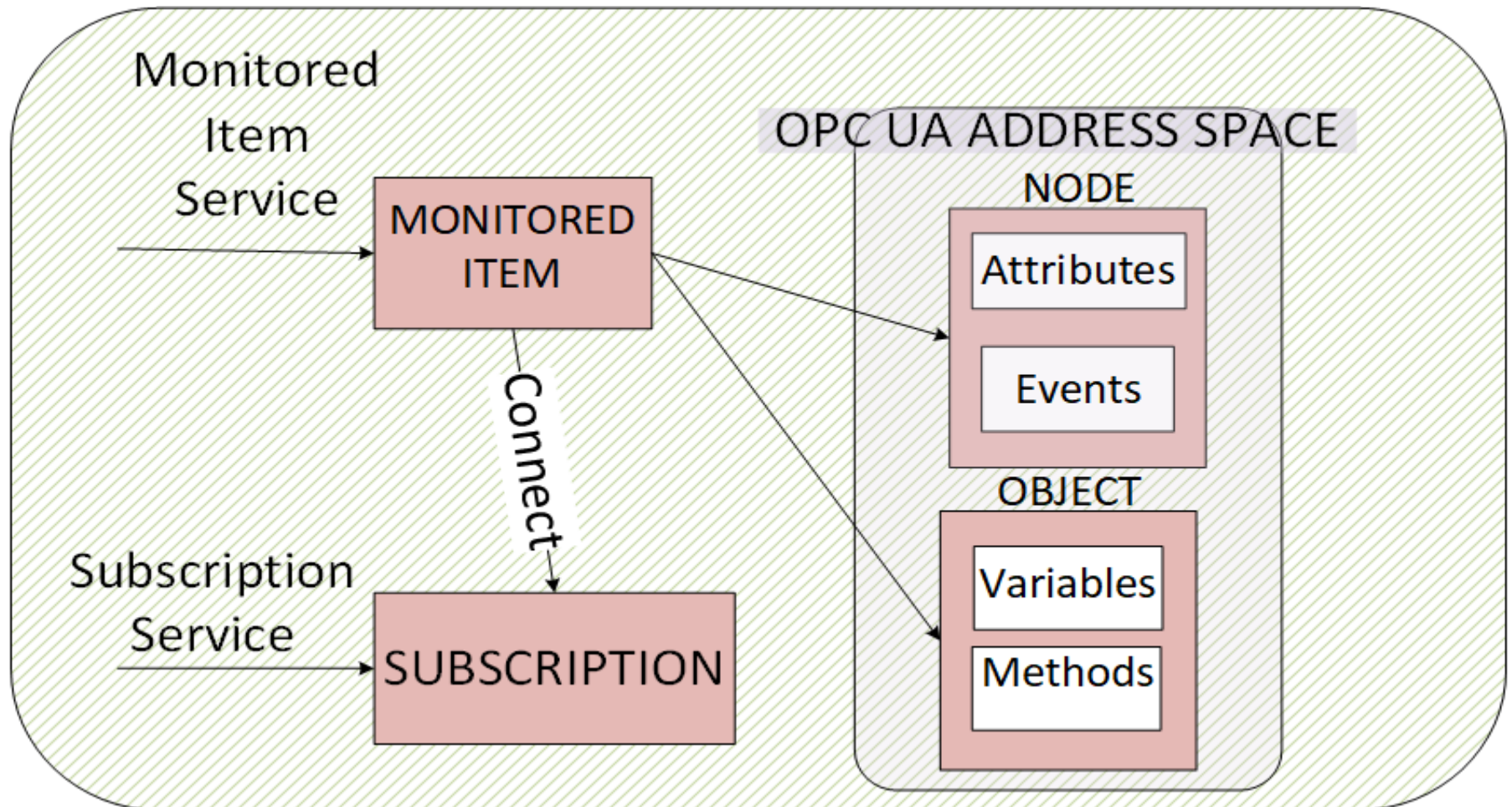
# INFORMATION MODEL



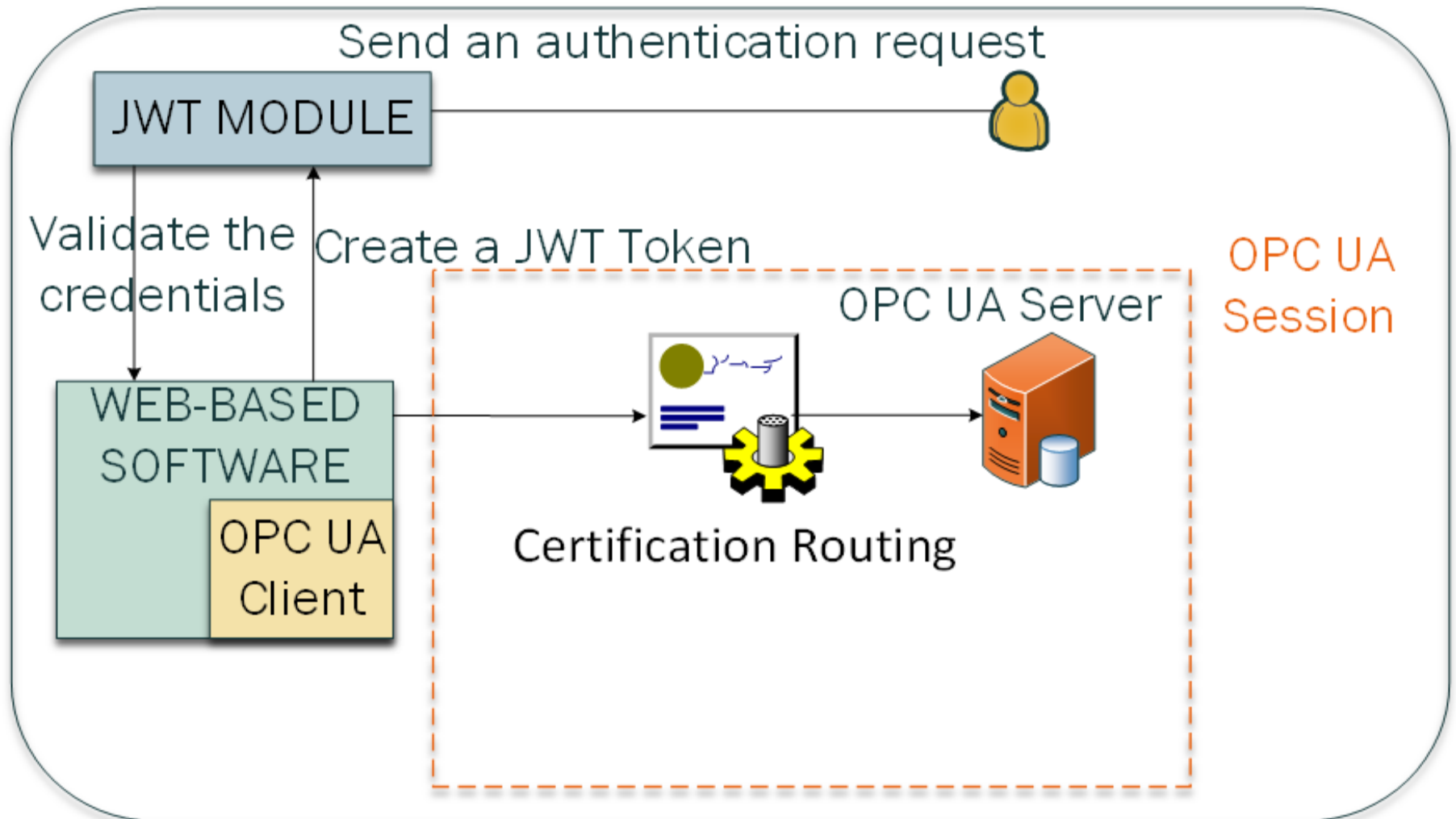
# MESSAGE BROKER SERVICE



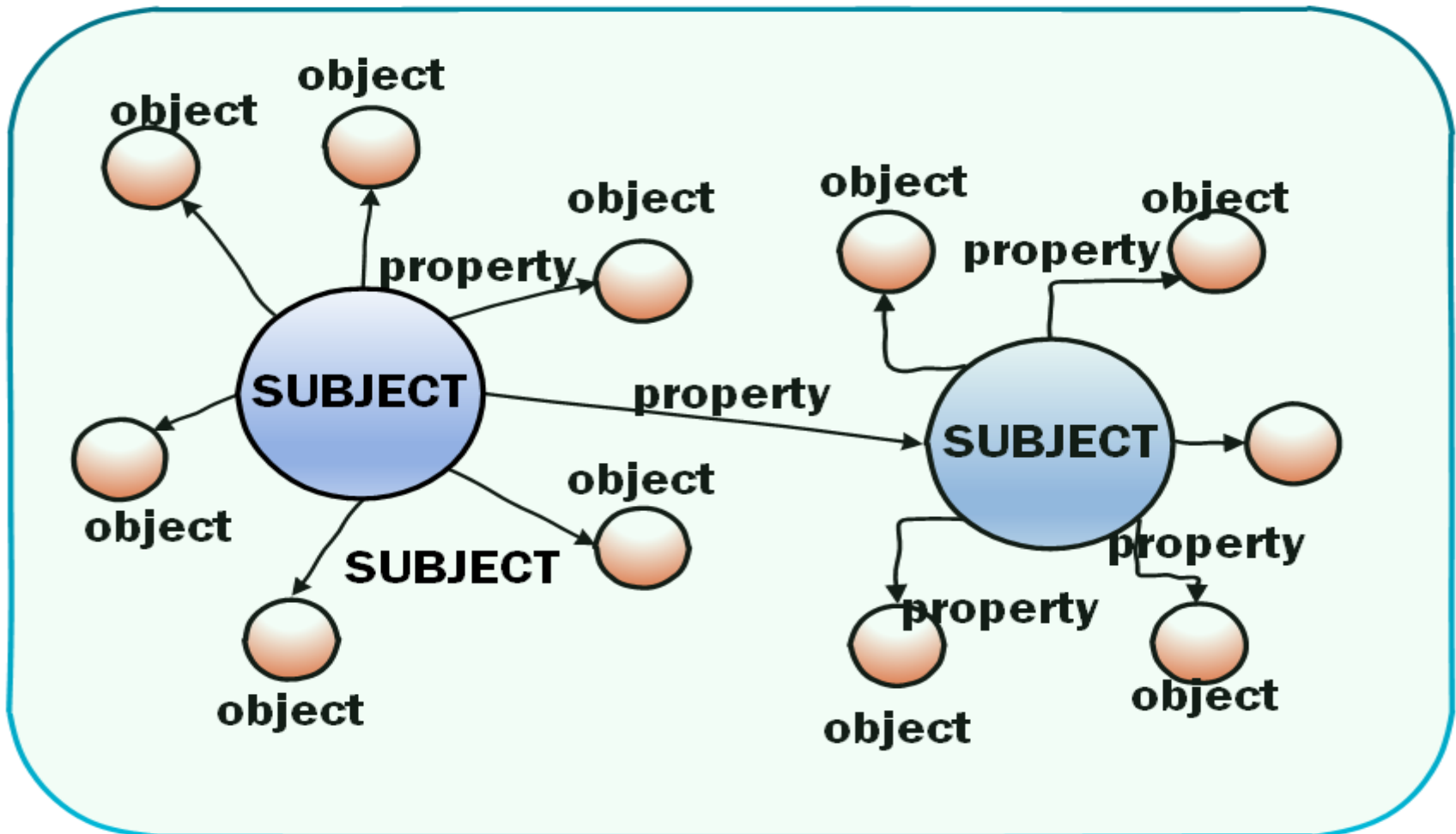
# SUBSCRIPTION SERVICE



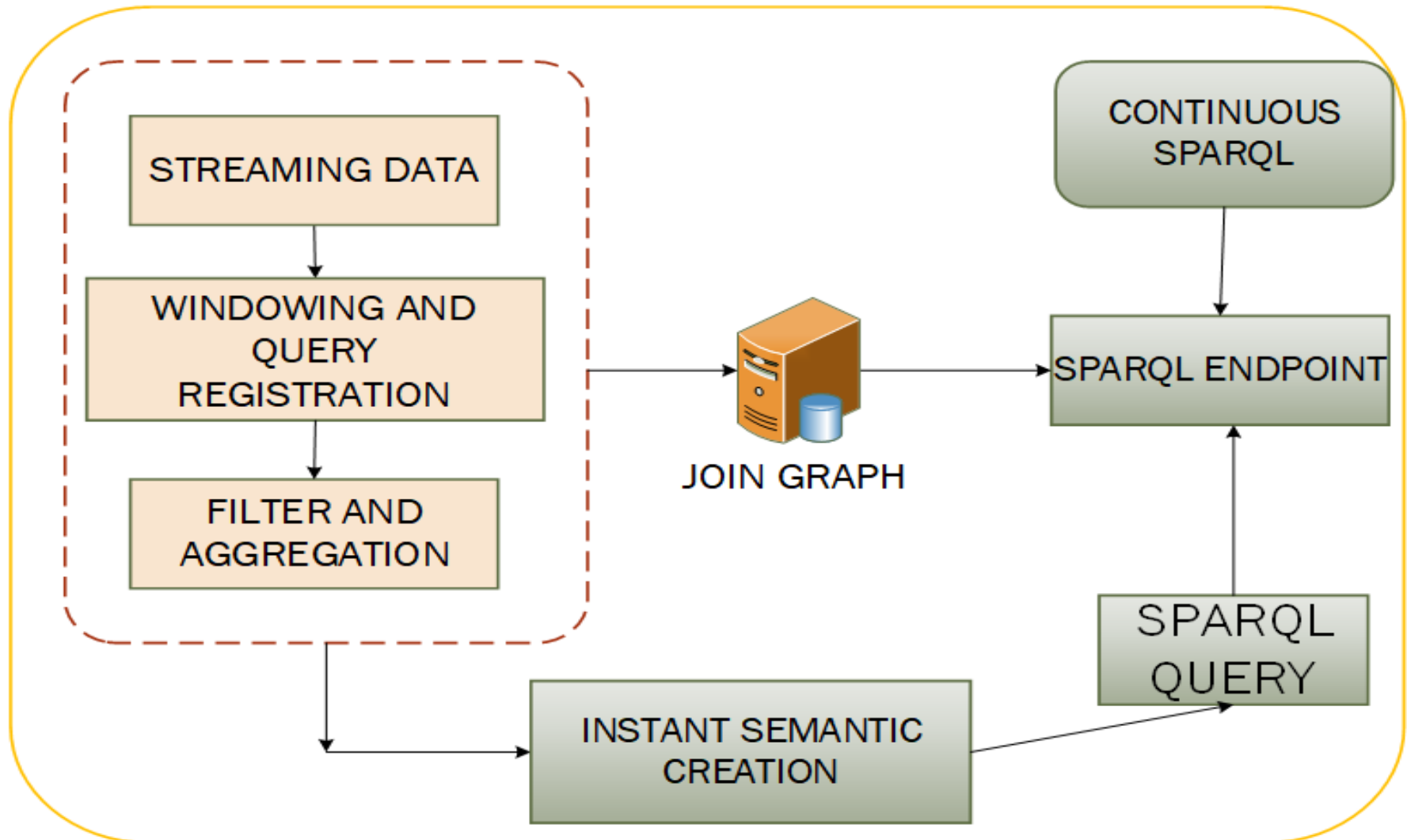
# AUTHENTICATION SERVICE



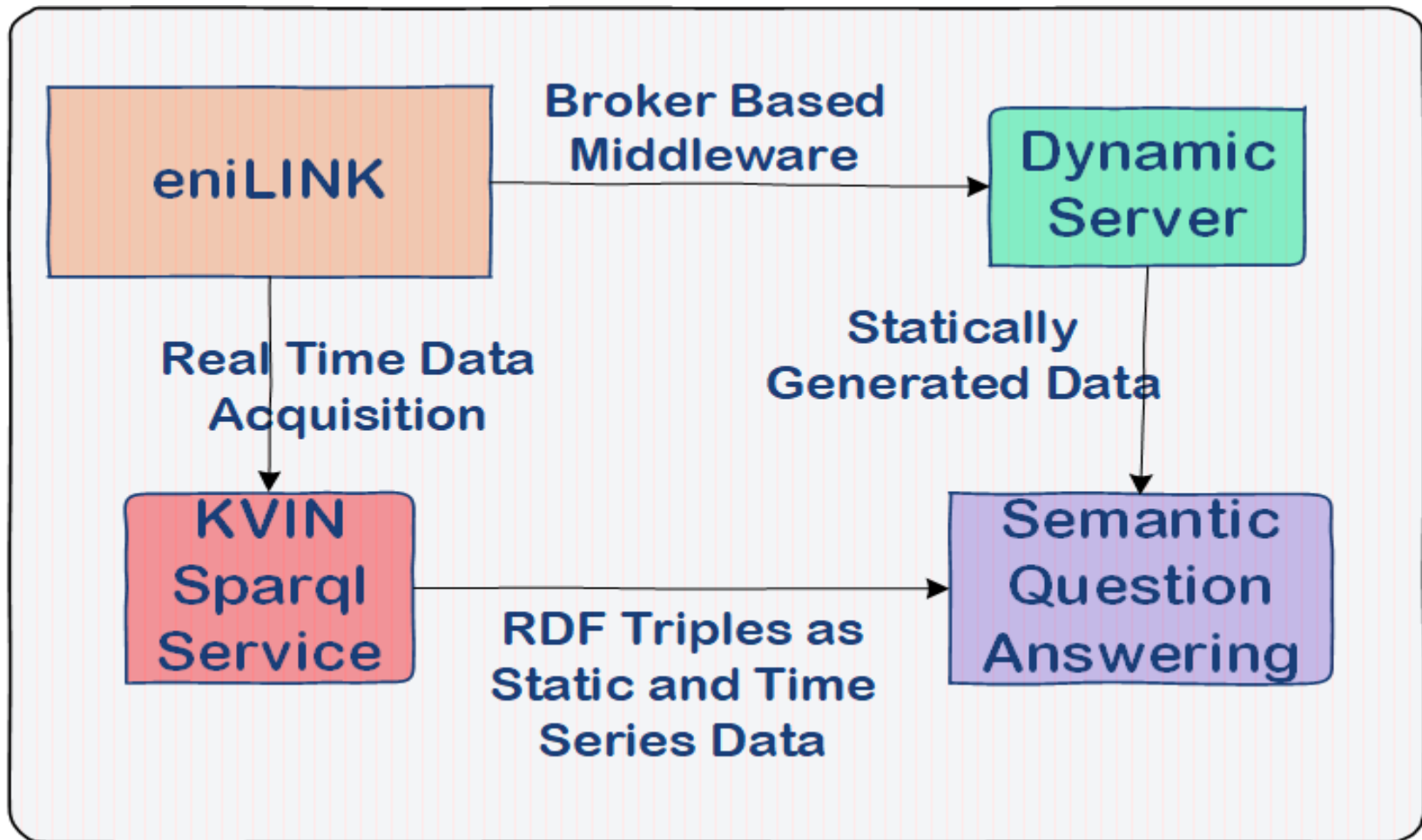
# LINKED DATA CONCEPT



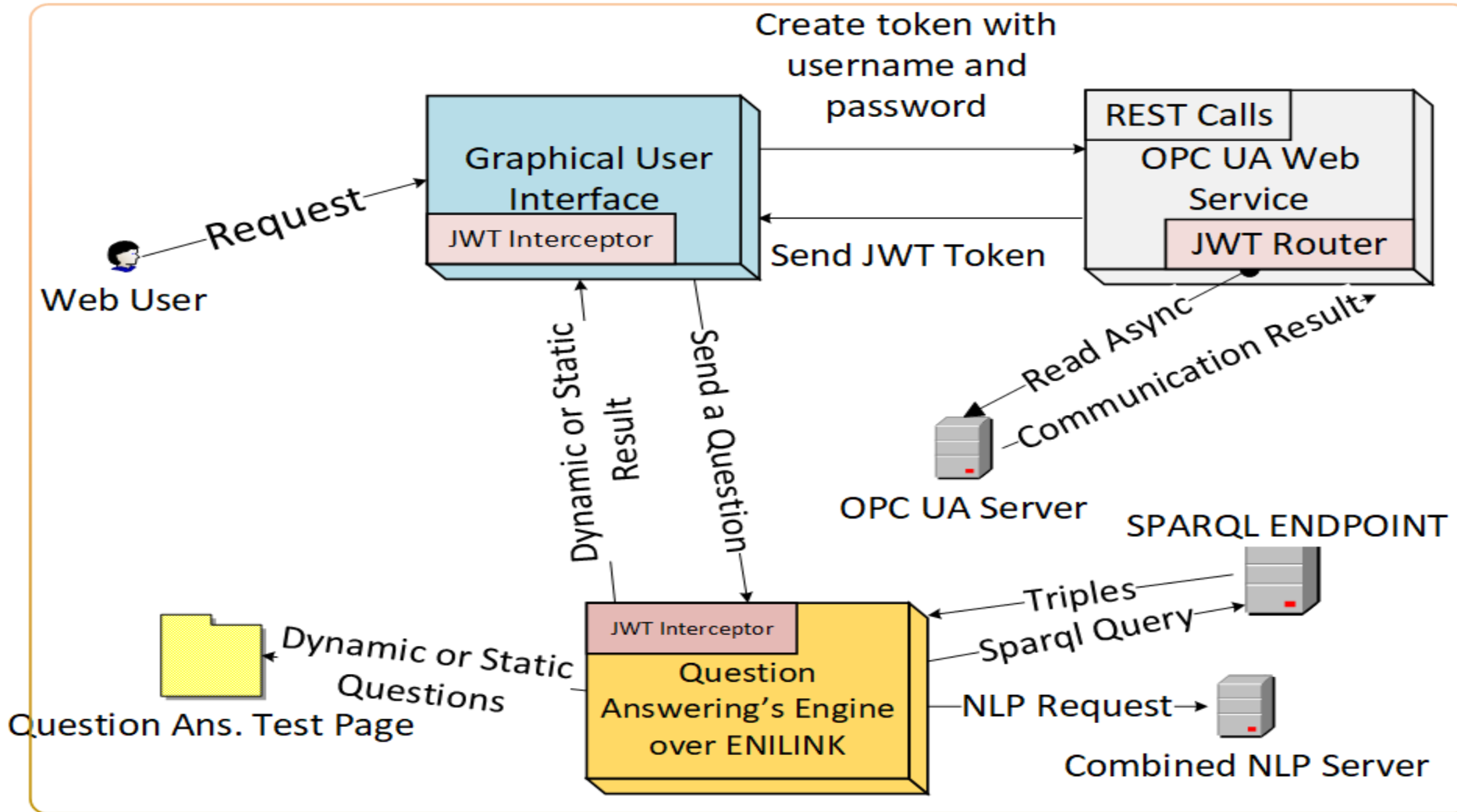
# SERIALIZATION INTO LINKED DATA



## KVIN SERVICE

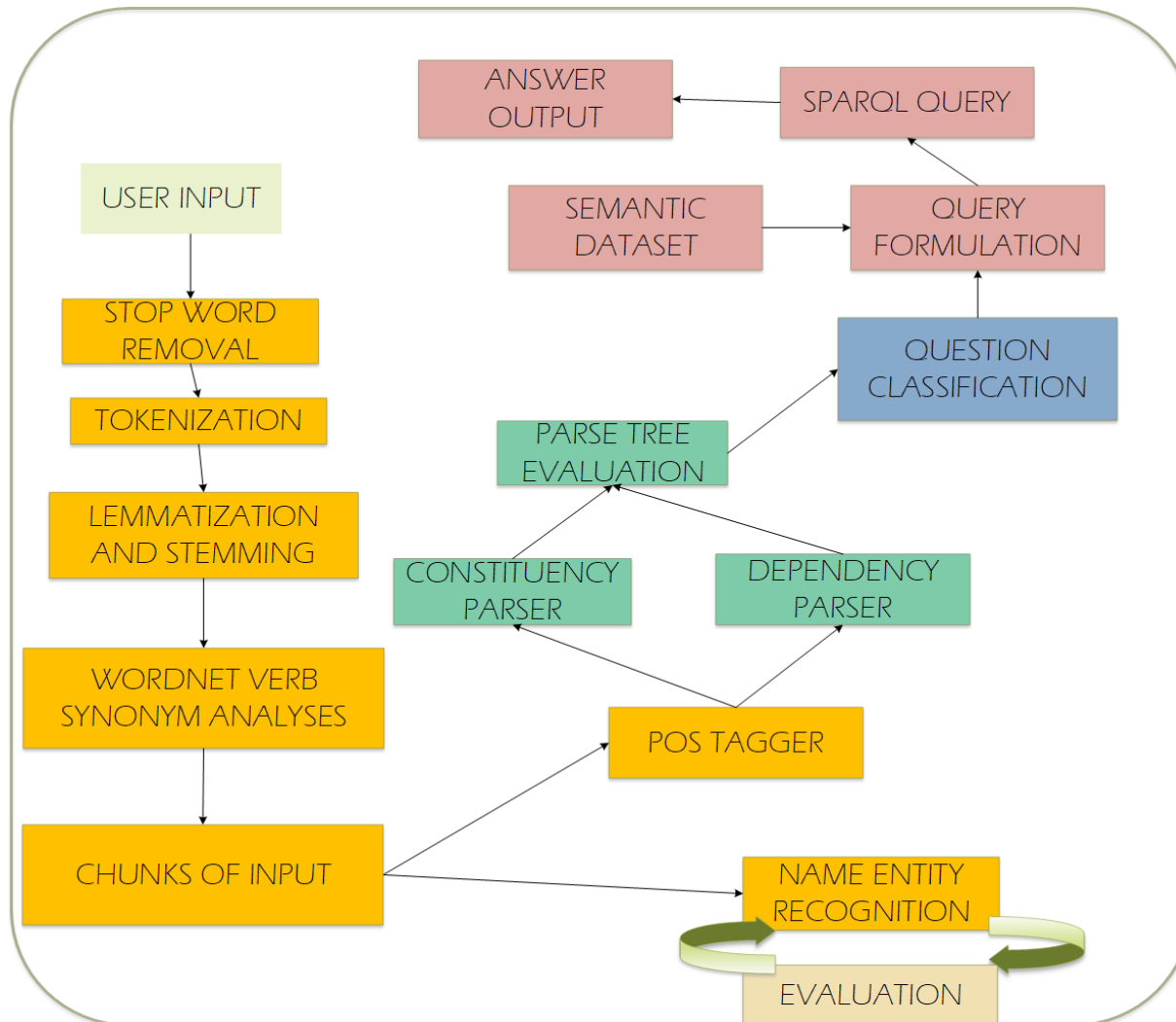


# SEMANTIC QUESTION ANSWERING





# FLOWCHART OF THE SEMANTIC QUESTION ANSWERING



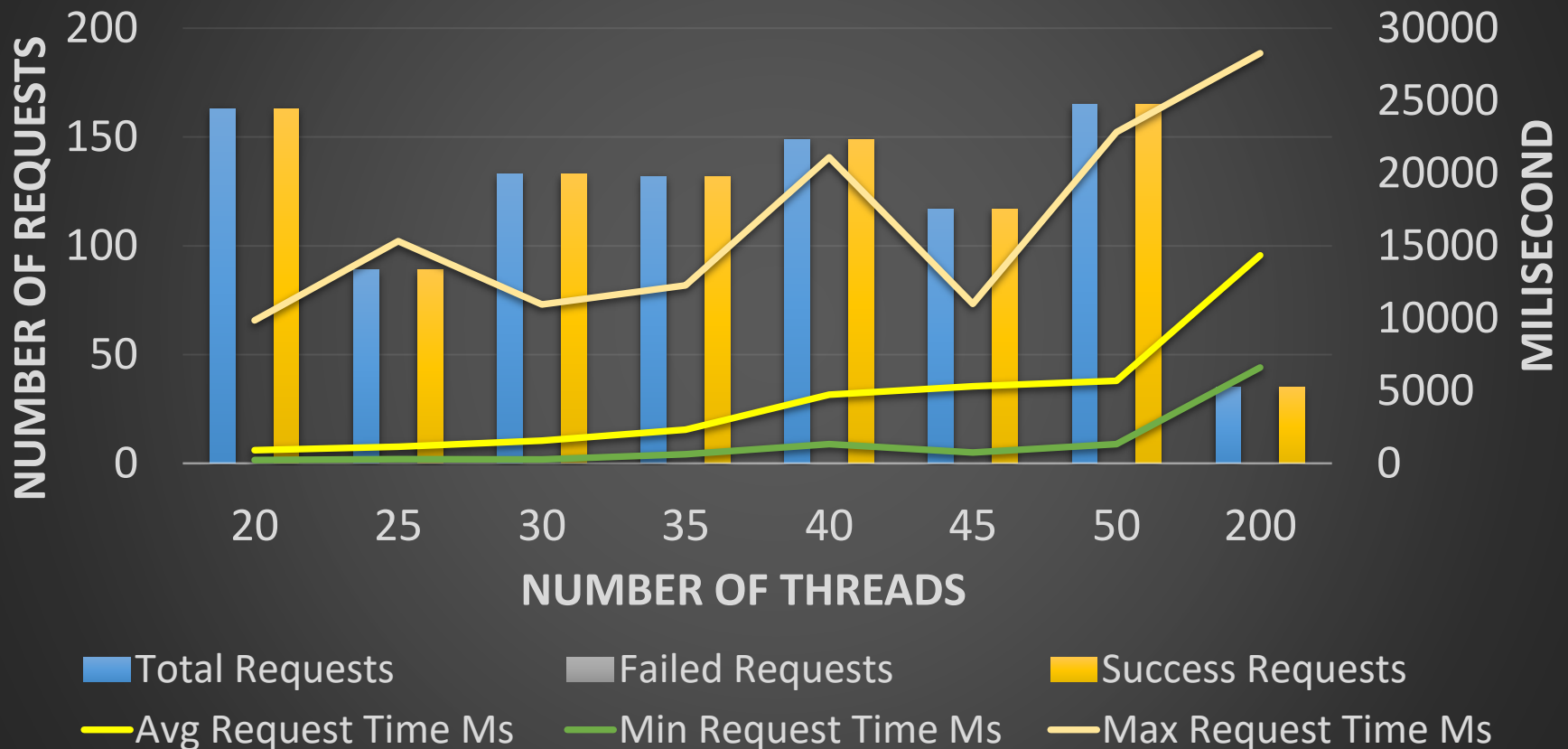
- **Green:** Syntactic Parsing
- **Blue:** Identify Questions
- **Orange:** Initial Steps for Identifying Chunks
- **Red:** Data Set Insertion and Semantic Mapping

# DEMO



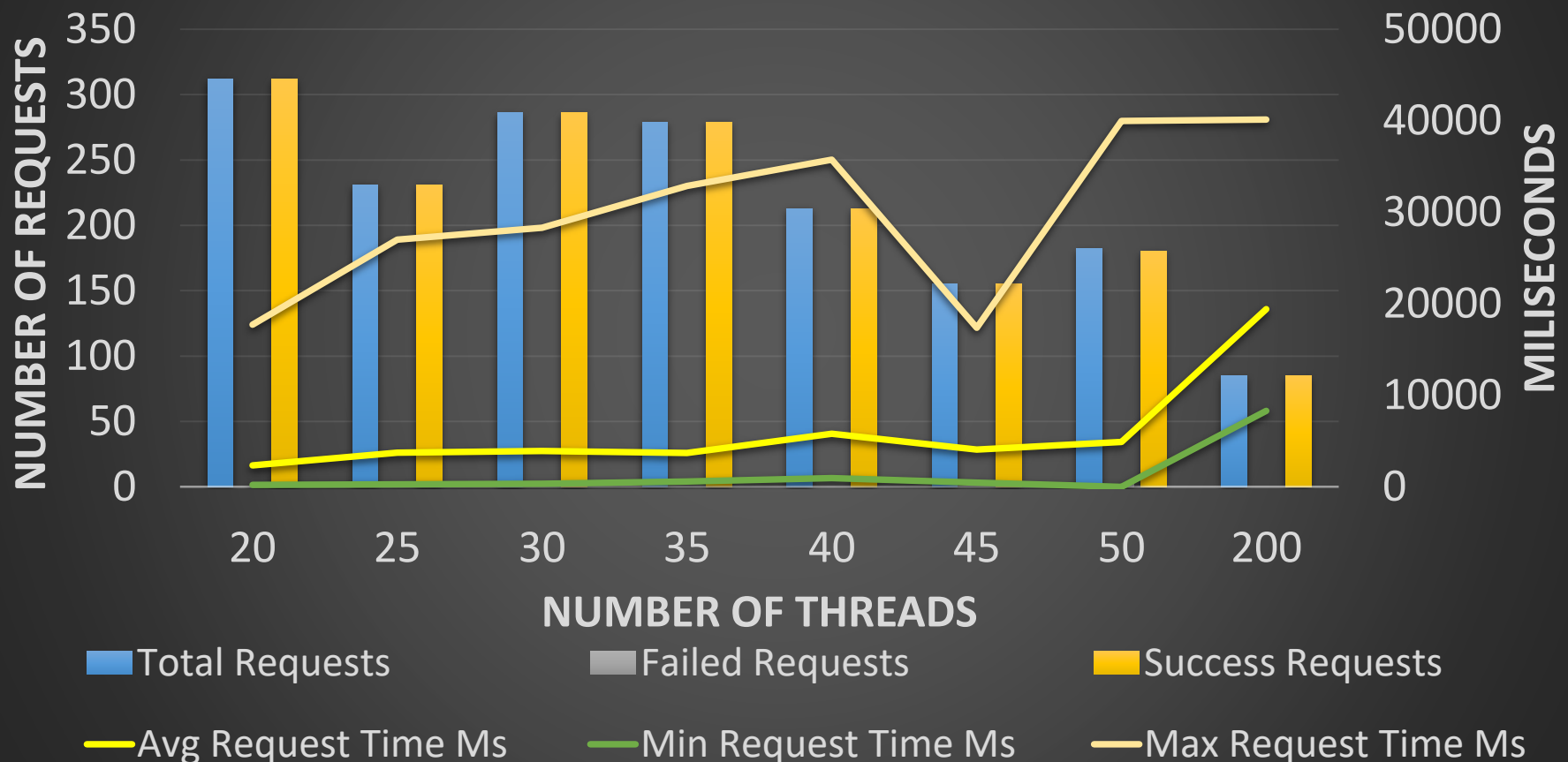
# EVALUATION

## 30s Single Instance



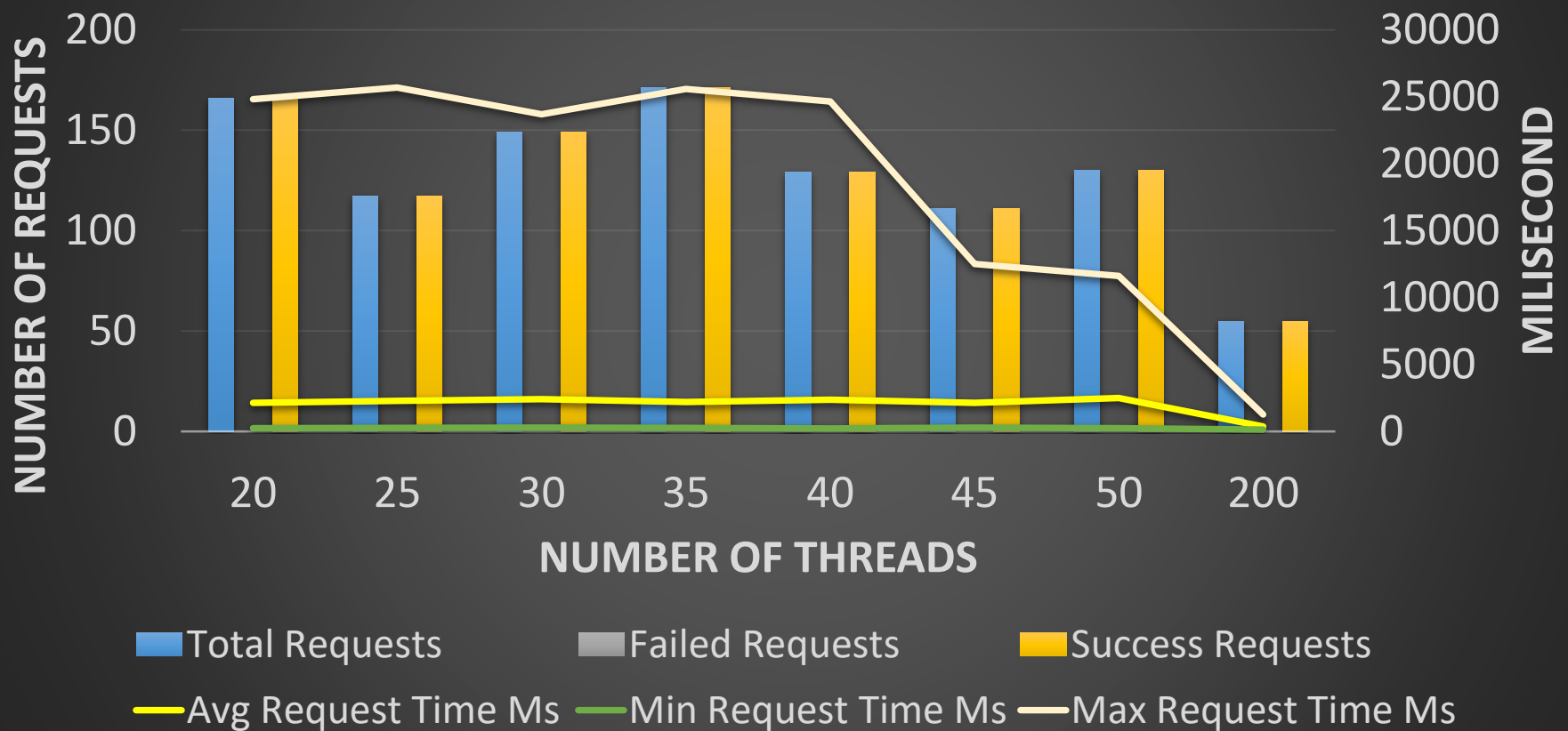
# EVALUATION

## 60s Single Instance



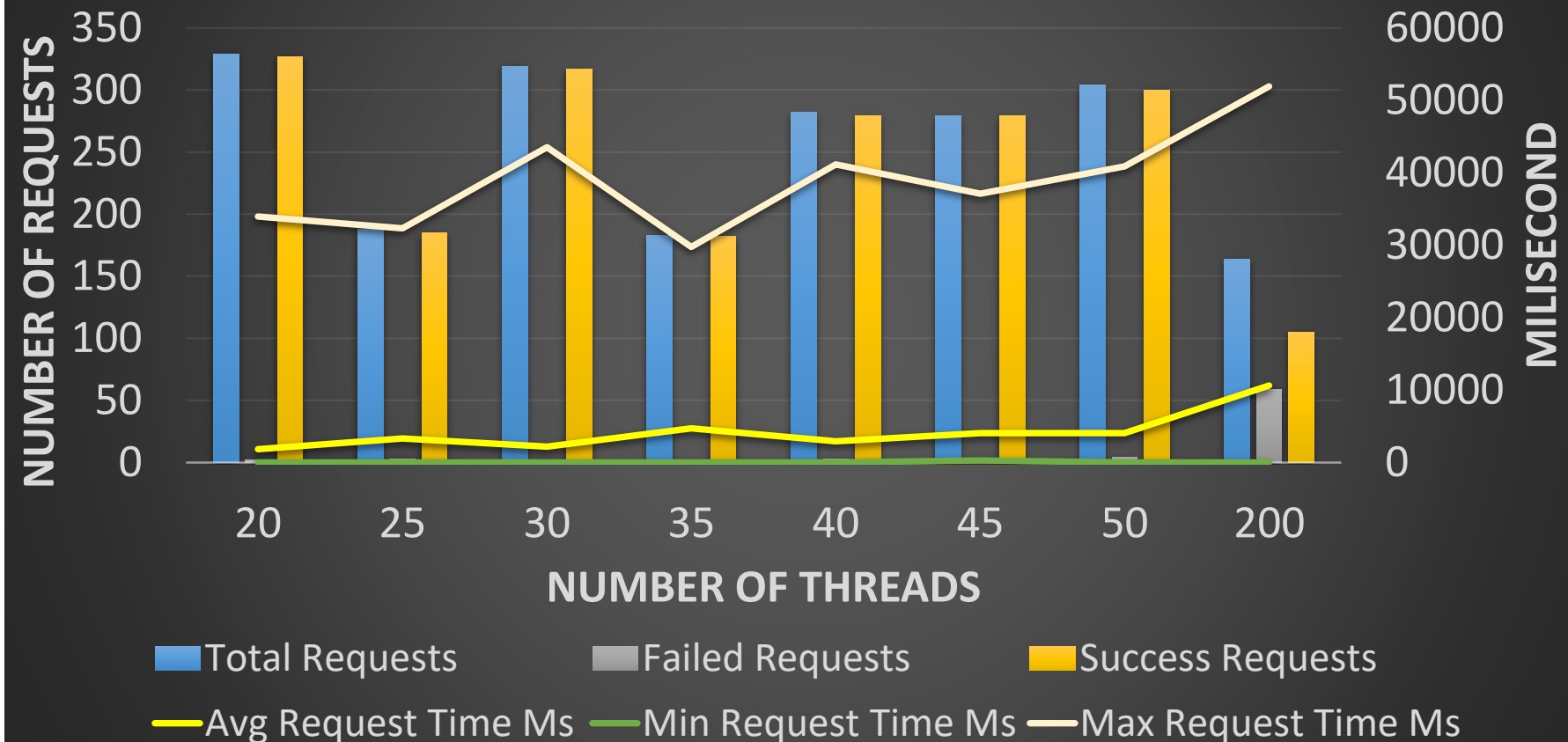
## EVALUATION

### 30s Round Robin Multiple Instances



## EVALUATION

### 60s Round Robin Multiple Instances



# QUESTION ANSWERING RESULTS

Evaluation Parameters	Properties
Answer Return Rate	Generated Data from OPC UA – 39.88 second – Consecutive Query of Generated Data 12.31 second Static query from RDF file of eniLINK – 19.33 second Dynamic Query – 17.48 second Open-Domain Question Answering Query – 20.55 s
Querying Style	Keyword-Based Search and Semantic Search
Coverage	eniLINK data, linkedfactory streaming data
Size	Static data relatively small size Continuous data relatively large size
Up-to-dateness	No update statement provided by SPARQL
Query Formulation Assistance	Voice Input Recognition, Spell Checker

# QUESTION ANSWERING RESULTS

Question Answering	True Positive	False Negative	False Positive	Precision	Recall	F1	Accuracy of the Model
Total Questions	34	13	3	%94.44	%72.34	%81.92	%68

1. **Precision** = True Positives / (True Positives + False Positives)
2. **Recall** = True positives / (True Positives + False Negatives)
3. **F1-Score** =  $2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$
4. **Accuracy of the Model** = (True Positive + True Negative) / (True Positive + False Negative + False Positive + True Negative)



## OUR FINDINGS

- It is possible to map a stateless architecture onto a stateful architecture.
- Average request time can only be reduced via load balancing and shared session pool.
- Semantic Question Answering can be customizable for other smart factories.
- A web-based application should balance the load and support to discovery servers.
- Question Answering System suffers from data quality and data size.
- Nevertheless, the application can provide us a basic level of monitoring, controlling and assisting with natural queries.
- Synchronous and Asynchronous calls might be a problem when used in a mixed way.
- The semantic question answering is not suitable for mission-critical system.

## CONCLUSION/FUTURE WORK

- OPC UA Web Based Software should not be compared with other desktop solutions. Desktop solutions handle request within the stateful connection.
- Our contribution is to assess the web-based application with empirical parameters and propose a novel architecture.
- Contributed the previous studies by researching gap of them about generating data from OPC UA and providing all in one web software (Quasi Microservices).
- An innovative implementation of the assistance system for human operators.
- A decision control system and monitor application

## FURTHER IMPROVEMENTS

- Global Discovery Service for inter-factories
- Deep Learning for subject-property-object triples
- Customized Named Entity Recognition
- Non-blocking IO Queue to balance internal messaging
- Reason induction compatible question answering system

Thank  
you!

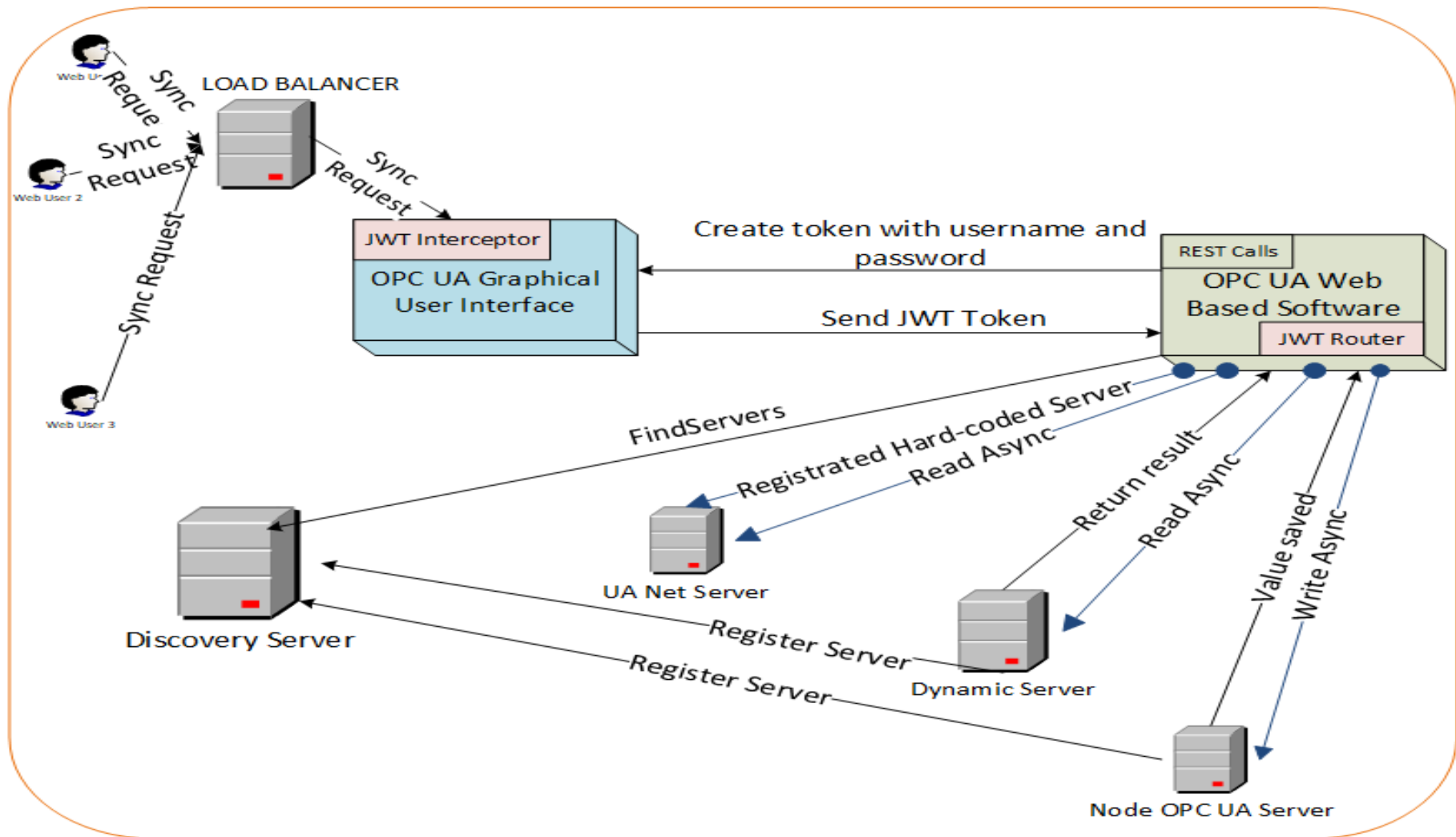
# Q&A



# ADDITIONAL SLIDES

The following slides are additional slides.

# OPC UA ARCHITECTURE



# Extraction Algorithm

---

## Algorithm 1 Node Extraction

---

```

1: function MAINFUNCTION()                                ▷ Starting point
2:   export = ServerExport(serverurl, filename)
3:   export.IMPORT NODES(serverurl)
4:   export.EXPORT FILE(outputFile, namespaces)
5: function BUILD NODE TREE(nodes)                        ▷ Node Formatting
6:   client ← GETENDPOINT()
7:   client ← CLIENT(serverurl)
8:   nodecumulated ← None
9:   nodeID ← 0
10:  for node < nodes do
11:    nodecumulated = node.nodeid.Namespaceindex
12:    for ref < node.getreferences() do
13:      nodecumulated.extend( ref.nodeid.Namespaceindex)
14:    nodecumulated = list(set(nodecumulated))             ▷ Clear duplicates
15:  return nodeID                                          ▷ Return node id list
16: function IMPORT NODES(serverurl)                       ▷ Traverse Node
17:   client = Client(serverurl)
18:   client.connect()
19:   for ns < client.getNamespaces() do
20:     namespaces[client.getNamespaceIndex(ns)] = ns
21:   root = client.getRootNode()
22:   child = client.iterateChildNodes()
23: function EXPORT FILE(outputFile, namespaces = None)    ▷ Export into
XML
24:   If namespaces != None then
25:     for node != nodes do
26:       If node.nodeid.namespaceindex is namespaces
27:         nodes = [node]
28:       else
29:         nodes = list(nodes)
30:
31:   export = XmlExport(client) then
32:   export.BUILD NODE(nodes)
33:   export.appendXML(outputFile)

```



## DEFINITIONS OF REST API

- **GET /api/authenticate No Authentication Bearer:** This http get call used for authenticating predefined users in order to obtain a Json Web Token.
- **GET /api/serverconf Authentication Bearer {JWT}:** This http get call used for configuring a OPC UA server endpoint and name
- **GET /api/serverconf/{DataSet:int}/allnodes/{node\_id}:** Authentication Bearer {JWT}: This http get call used for obtaining OPC UA methods, variables, objects and references by means of namespaces and IDs.

## DEFINITIONS OF THE REST API

- **POST /api/serverconf/{DataSet:int/allnodes/{node\_id} value {"message"} Authentication Bearer {JWT}**: This http post call used for updating or inserting a new value which resides in any node.
- **GET /integratedstaticmessage/{question} Authentication Bearer {JWT} Body of Request**: Angular Front-end Static Message HTTP Request
- **GET /integrateddynamicmessage/{question} Authentication Bearer {JWT} Body of Request**: Angular Front-end Static Message HTTP Request
- **GET /api/serverconf/DataSetID/subscribeNodes/monitor\_id Authentication Bearer {JWT}**: HTTP Call for Monitorable Nodes

# BINARY CLASSIFICATION

**True Positives:** Data points labeled as positive that are actually positive

**False Positives:** Data points labeled as positive that are actually negative

**True Negatives:** Data points labeled as negative that are actually negative

**False Negatives:** Data points labeled as negative that are actually positive

# eniLINK FRAUNHOFER IWU

Dashboard / linkedfactory / demofactory / machine5 / sensor2

<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/demofactory/machine5/sensor2>

Children

Search...



Graph

Grid

No children found.

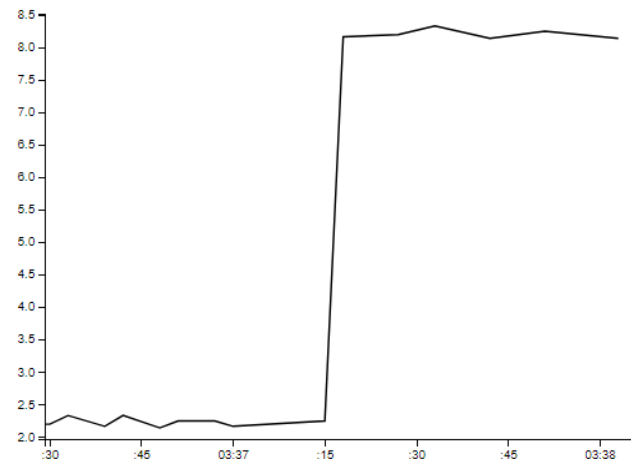
02/13/2019 7:17 AM



02/13/2019 3:37 PM



value



☐ relative time

# MAIN FEATURES OF NATURAL LANGUAGE PROCESSING

**Tokenization** : Each sentence must tokenize with Natural Language Processing algorithms. So we can implement further algorithms on combine tokens. For instance; Friends, Romans, Countrymen, lend me your ears – Tokenization – "Friends", "Romans", "Countrymen", "lend", "me", "your", "ears"

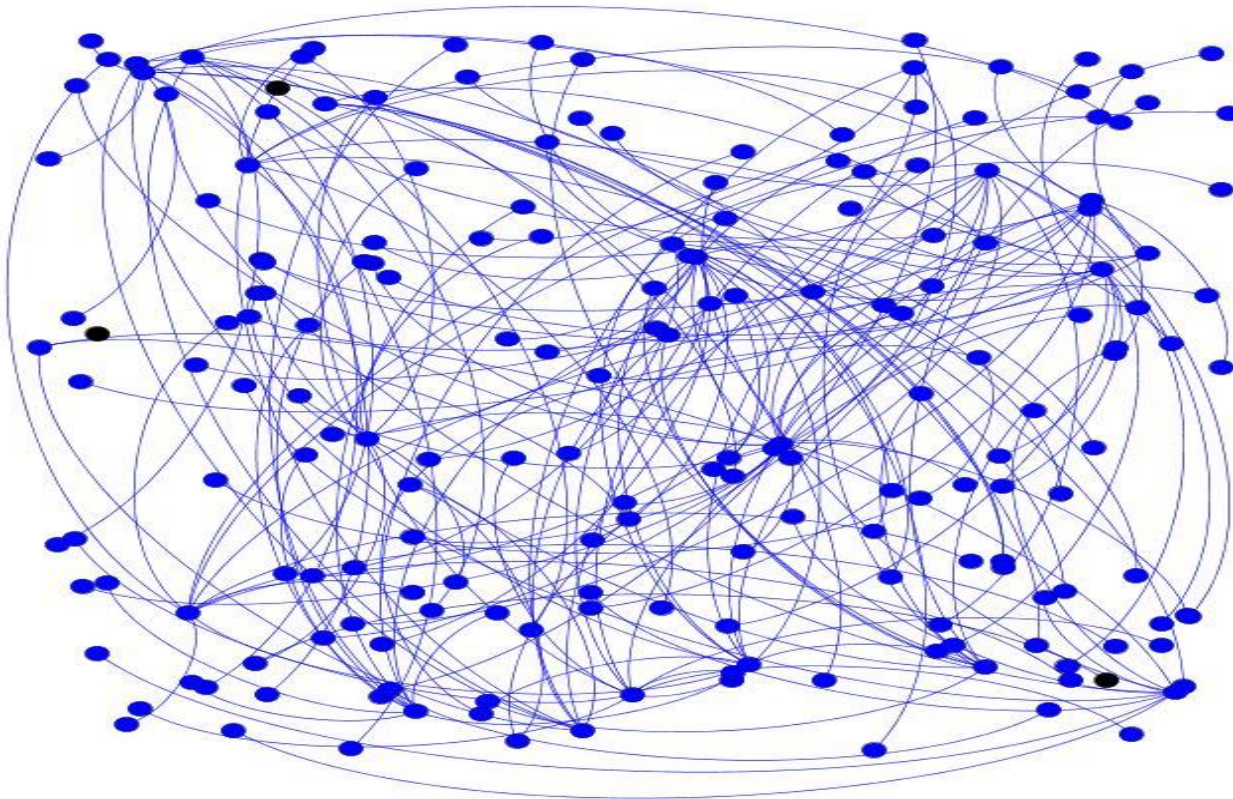
**Stemming and Lemmatizing** : Word stemming means removing affixes from words and returning the root word (which may not be a real word). Lemmatizing is similar to stemming, but the difference is that the result of lemmatizing is a real word.

**Part of Speech Tagger (POS Tagger)**: This is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc.

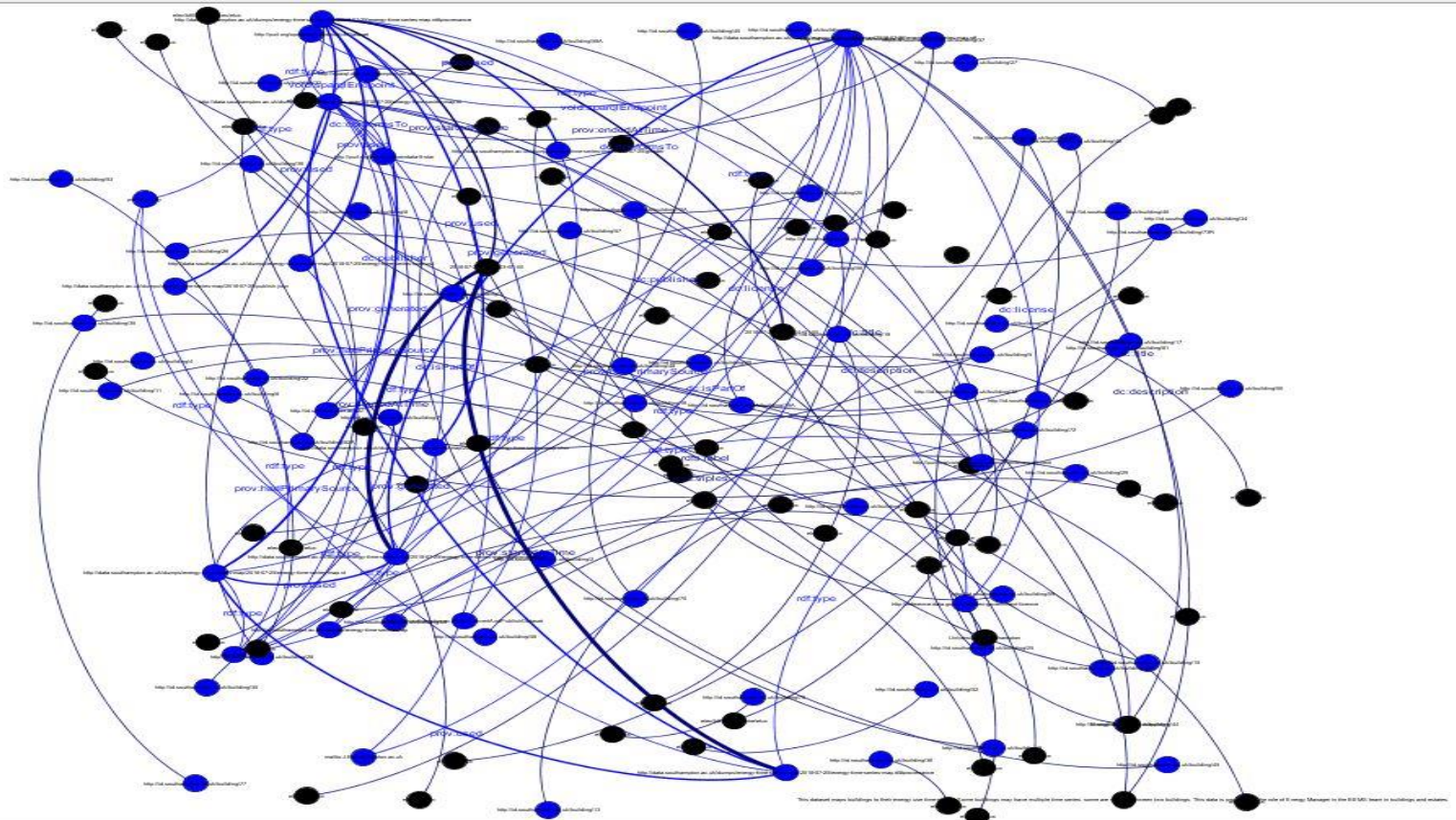
**Name-Entity Recognition (NER Tagger)**: This is a piece of software that reads noun, verb, adjective etc. to convert subject – predicate (verb) – object formats.

**Parser Tree**: This is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the **subject** or **object** of a verb.

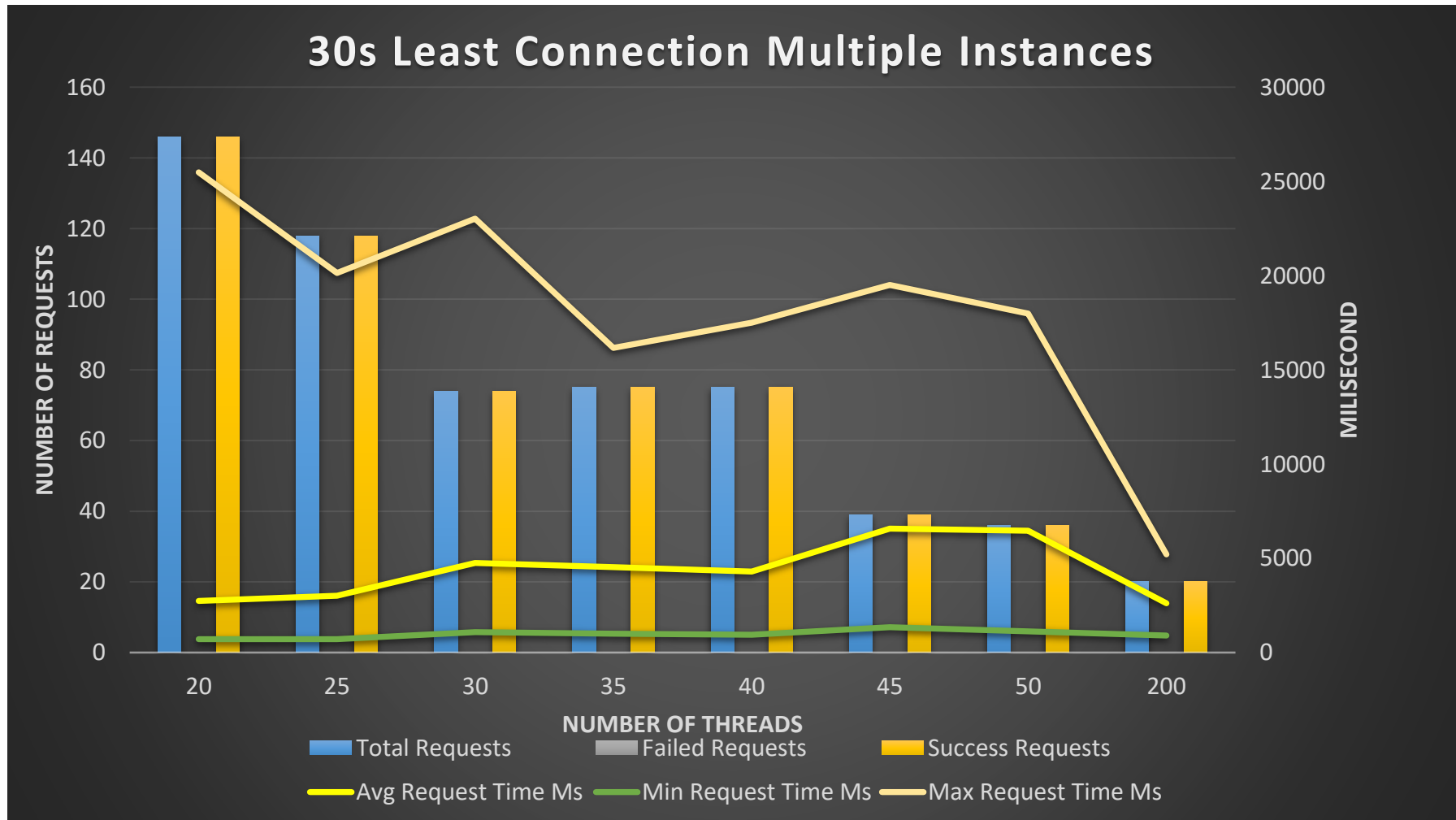
# LINKED FACTORY STATIC RDF NODES





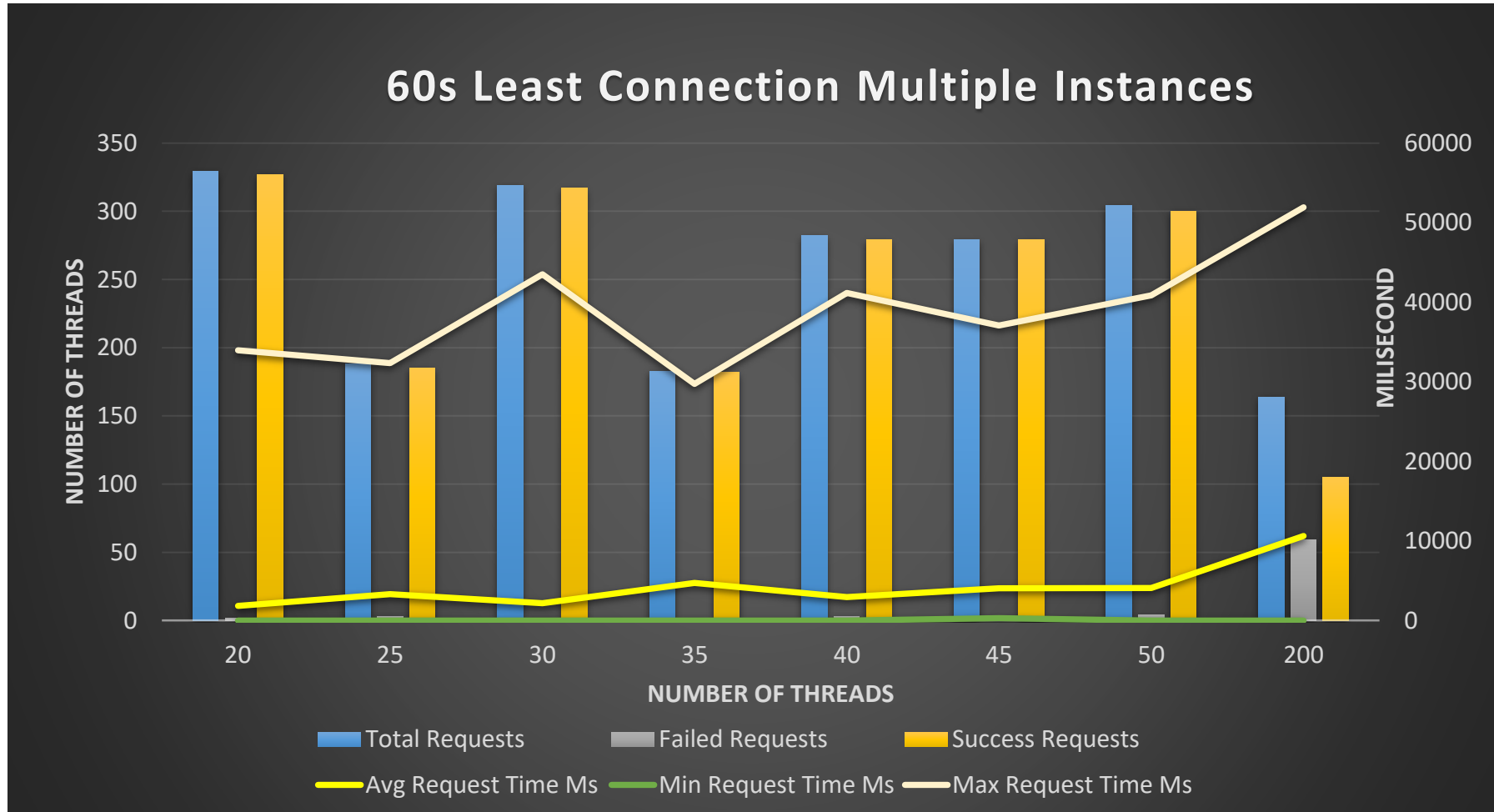


## RESULTS FOR LEAST CONNECTION





## RESULTS FOR LEAST CONNECTION



## REFERENCES

- [1] <http://archive.dnnsoftware.com/docs/85/developers/security/jwt/index.html>
- [2] <https://mentorphile.com/2018/09/14/demo-or-die/>
- [3] <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>