Supporting Evacuation Missions with Ontology-based SPARQL Federation

Audun Stolpe and Jonas Halvorsen
STIDS 12-15 November 2013
Table of contents

1. The Nato Network Enabled Capability (NNEC) concept
2. Example: Evacuation planning
3. System overview
4. Source Selection
5. Reducing the size of the cropping
The NNEC Concept

- Official policy for Command and Control in NATO forces
- Goal: Achieve better mission effectiveness
- Builds upon extensive information sharing
- Shift from “Need to know” to “Responsibilitiy to share”
NNEC objectives

• Some technical aims:
  • support extensive information sharing,
  • provide a robust scheme for information integration,
  • standardise exchange and storage formats

• Some strategic aims:
  • create a high degree of shared situational awareness,
  • based on timely data,
  • to support decision making during operations
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
  - standardise exchange and storage formats
- Some strategic aims:
  - create a high degree of shared situational awareness,
  - based on timely data,
  - to support decision making during operations
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
  - standardise exchange and storage formats
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
  - standardise exchange and storage formats

- Some strategic aims:
NNEC objectives

• Some technical aims:
  • support extensive information sharing,
  • provide a robust scheme for information integration,
  • standardise exchange and storage formats

• Some strategic aims:
  • create a high degree of shared situational awareness,
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
  - standardise exchange and storage formats

- Some strategic aims:
  - create a high degree of shared situational awareness,
  - based on timely data,
NNEC objectives

- Some technical aims:
  - support extensive information sharing,
  - provide a robust scheme for information integration,
  - standardise exchange and storage formats

- Some strategic aims:
  - create a high degree of shared situational awareness,
  - based on timely data,
  - to support decision making during operations
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- Our assumptions:
  - data models are standardised
  - HTTP can be assumed
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- Our assumptions:
  - data models are standardised
  - HTTP can be assumed
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members

- Our assumptions:
  - data models are standardised
  - HTTP can be assumed
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear

- Our assumptions:
  - data models are standardised
  - HTTP can be assumed
Presuppositions

- NNEC assumptions:
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- Our assumptions:
  - data models are standardised
  - HTTP can be assumed
Presuppositions

- **NNEC assumptions:**
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- **Our assumptions:**
Presuppositions

- **NNEC assumptions:**
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- **Our assumptions:**
  - data models are standardised
Presuppositions

- **NNEC assumptions:**
  - fragile ICT infrastructure, typically IP radio
  - low bandwidth + latency issues
  - information is typically distributed across systems
  - systems are contributed by different coalition members
  - systems may appear and disappear
  - information may be mission critical

- **Our assumptions:**
  - data models are standardised
  - HTTP can be assumed
Desiderata

We want an approach that:
1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests,
3. allows the relevant sources to be discovered at run-time,
4. guarantees the soundness/completeness of q.a.

• Points to ontology based data access + rewriting
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests,
3. allows the relevant sources to be discovered at run-time,
4. guarantees the soundness/completeness of q.a.

• Points to ontology based data access + rewriting
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests
We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests
3. allows the relevant sources to be discovered at run-time
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests
3. allows the relevant sources to be discovered at run-time
4. guarantees the soundness/completeness of q. a.
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests
3. allows the relevant sources to be discovered at run-time
4. guarantees the soundness/completeness of q. a.
Desiderata

We want an approach that:

1. allows a user to access available sources in a unified way,
2. minimises the number of required HTTP requests
3. allows the relevant sources to be discovered at run-time
4. guarantees the soundness/completeness of q. a.

- Points to ontology based data access + rewriting
1. The Nato Network Enabled Capability (NNEC) concept
2. Example: Evacuation planning
3. System overview
4. Source Selection
5. Reducing the size of the cropping
Case description

- JOC — Joint Operations Center

Several medical evacuation missions are being monitored by the analyst. These missions are particularly those that are threatened by enemy activity. The information need expressed by the query is to find all medical evacuation missions and friendly units such that:

1. The mission can be classified as being threatened;
2. The friendly unit can handle the specific type of threat that the enemy poses.
Case description

- JOC — Joint Operations Center
- analyst is monitoring medical evacuation missions
Case description

- JOC — Joint Operations Center
- analyst is monitoring medical evacuation missions
- particularly missions that are threatened by enemy activity
Case description

- JOC — Joint Operations Center
- analyst is monitoring medical evacuation missions
- particularly missions that are threatened by enemy activity
- information need expressed by the query
Case description

- JOC — Joint Operations Center
- analyst is monitoring medical evacuation missions
- particularly missions that are threatened by enemy activity
- information need expressed by the query

“Find all medical evacuation missions and friendly units such that a) the mission can be classified as being threatened; and b) that the friendly unit can handle the specific type of threat that the enemy poses.”
Information sources

The query involves three operational information systems:

1. JOCWatch: a log of events reported from the field
2. MedWatch: medical mission planning and tracking
3. Track Source: friendly units: capabilities, location.
Information sources

The query involves three operational information systems:

1. **JOCWatch**: a log of events reported from the field
2. **MedWatch**: medical mission planning and tracking
3. **Track Source**: friendly units: capabilities, location.
Information sources

The query involves three operational information systems:

1. **JOCWatch**: a log of events reported from the field
Information sources

The query involves three operational information systems:

1. **JOCWatch**: a log of events reported from the field
2. **MedWatch**: medical mission planning and tracking
Information sources

The query involves three operational information systems:

1. JOCWatch: a log of events reported from the field
2. MedWatch: medical mission planning and tracking
3. Track Source: friendly units: capabilities, location.
Ontology-based data integration
Conceptual relationship between data sources

[Diagram showing the relationships between data sources (JOCWatch, MedWatch, TrackSource).]
Rewriting: A simple example

Query: ThreatenedMission(x)?

\[
\top \rightarrow \text{Mission} \\
\text{Mission} \rightarrow \text{Incident} \\
\text{Incident} \rightarrow \text{HostileOp} \\
\text{ThreatenedMis} \rightarrow \text{ThreateningInc} \\
\text{ThreateningInc} \rightarrow \text{Mission}(x) \land \text{overlaps}(x, y) \land \text{Incident}(y) \land \text{partOf}(y, z) \land \text{HostileOp}(z)
\]

Diagram:
- Mission
- Incident
- HostileOp
- ThreatenedMis
- ThreateningInc

Relationships:
- is-a
- partOf
- overlaps
Query: ThreatenedMission(x)?
Query: ThreatenedMission(x)?
Rewriting: A simple example

Query: ThreatenedMission(x)?

Mission

ThreatenedMis

HostileOp

Incident

ThreateningInc

Mission(x) & overlaps(x, y) &
Query: ThreatenedMission(x)?
Rewriting: A simple example

Query: ThreatenedMission(x)?

Mission(x) & overlaps(x, y) & Incident(y) & partOf(y,z) & HostileOp(z)
<table>
<thead>
<tr>
<th></th>
<th>The Nato Network Enabled Capability (NNEC) concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Example: Evacuation planning</td>
</tr>
<tr>
<td>3</td>
<td>System overview</td>
</tr>
<tr>
<td>4</td>
<td>Source Selection</td>
</tr>
<tr>
<td>5</td>
<td>Reducing the size of the cropping</td>
</tr>
</tbody>
</table>
System overview
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$

Diagram:

1. **Rewrite** $Q$ and $\Sigma$ to $Q_\Sigma$.
2. **Distribute** $Q_\Sigma$ to sources $R_1, R_2, \ldots, R_n$.
3. **Crop** snapshots from sources.
4. **Evaluate** cropped snapshots.
5. **Answer** final result.
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- Federator selects from available sources
- Splits $Q$ into subqueries tailored for each source, and distributes subqueries to the sources
- Results of subqueries yield snapshots that are jointly sufficient for answering $Q_\Sigma$
- Snapshots are joined to form the cropping $A$
- Against which, $Q_\Sigma$ is finally evaluated

Answer
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- Federator selects from available sources
- According to signature of $Q_\Sigma$, then
- Splits $Q_\Sigma$ into subqueries tailored for each source, and
- Distributes subqueries to the sources
- Results of subqueries yield snapshots, that
- Are jointly sufficient for answering
- Snapshots are joined to form the cropping
- Against which, $Q_\Sigma$ is finally evaluated
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- federator selects from available sources
- according to signature of $Q_\Sigma$, then
- splits $Q$ into subqueries tailored for each source, and

![Diagram of system overview](image-url)
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- federator selects from available sources
- according to signature of $Q_\Sigma$, then
- splits $Q$ into subqueries tailored for each source, and
- distributes subqueries to the sources

Rewrite

$Q_\Sigma$

Distribute

$R_1$
$R_2$
$R_n$

Crop

Evaluate

Answer
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- Federator selects from available sources according to signature of $Q_\Sigma$, then
- Splits $Q$ into subqueries tailored for each source, and
- Distributes subqueries to the sources
- Results of subqueries yield snapshots, that
- Snapshots are joined to form the cropping against which, $Q_\Sigma$ is finally evaluated.
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- Federator selects from available sources
- According to signature of $Q_\Sigma$, then
- Splits $Q$ into subqueries tailored for each source, and
- Distributes subqueries to the sources
- Results of subqueries yield snapshots, that
- Are jointly sufficient for answering $Q$
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- federator selects from available sources according to signature of $Q_\Sigma$, then
- splits $Q$ into subqueries tailored for each source, and
- distributes subqueries to the sources
- results of subqueries yield snapshots, that are jointly sufficient for answering $Q$
- snapshots are joined to form the cropping $A$
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- federator selects from available sources
- according to signature of $Q_\Sigma$, then
- splits $Q$ into subqueries tailored for each source, and
- distributes subqueries to the sources
- results of subqueries yield snapshots, that
- are jointly sufficient for answering $Q$
- snapshots are joined to form the cropping $A$
- against which, $Q_\Sigma$ is finally evaluated
System overview

- Input: SPARQL query $Q$ and an ontology $\Sigma$
- $Q$ rewritten to $Q_\Sigma$
- Federator selects from available sources according to signature of $Q_\Sigma$, then
- Splits $Q$ into subqueries tailored for each source, and
- Distributes subqueries to the sources
- Results of subqueries yield snapshots, that
- Are jointly sufficient for answering $Q$
- Snapshots are joined to form the cropping $A$
- Against which, $Q_\Sigma$ is finally evaluated
Sample query

Query pattern:


Routed to MedWatch:

CONSTRUCT {
?_1 medics:missionType medics:Evac.
?_1 medics:jocwIncident ?_2.
?_3 jocw:status ?_4.
} WHERE {
{ ?_1 medics:missionType medics:Evac.
  ?_1 medics:jocwIncident ?_2.}
UNION
{ ?_3 jocw:status ?_4.}}

Routed to JOCWatch:

CONSTRUCT {
?_1 jocw:instigator ?_2.
?_3 jocw:status ?_4.
} WHERE {
{ ?_1 jocw:instigator ?_2.}
UNION
{ ?_3 jocw:status ?_4.}}

Each query is built as a union of exclusive/nonexclusive groups
Soundness/completeness

- Note the renaming of variables in the queries
- necessary for the soundness of query answering
- CONSTRUCT queries are defined to adhere to a logical form
- guarantees soundness/completeness in the sense:

\[
\text{Distributing a query } Q \text{ over a set of sources } R \text{ yields the exact same answer, as if } Q \text{ were evaluated directly against a single repository containing the union of } R.
\]
Table of contents

1. The Nato Network Enabled Capability (NNEC) concept
2. Example: Evacuation planning
3. System overview
4. Source Selection
5. Reducing the size of the cropping
Adapting to a dynamic network topology

- Important design goals:
  - do not hard-wire a query to a predefined set of sources
  - sources should be selected once per query
  - as sources come and go, the federator adapts

- Facilitated by FOL-rewritable ontology languages
  - FOL-rewritability decouples reasoning from data access
  - queries are precompiled (potentially cached)
  - selection and distribution performed afterwards
  - thus, network topology allowed to change
Detecting available source

- Discovery mechanism

Advantages:
- addresses the NNEC needs
- it is independent of a central registry,
- eliminates the issue of network fragmentation.
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,

Advantages:
- approach addresses the NNEC needs
- it is independent of a central registry,
- eliminates the issue of network fragmentation.
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,
  - DNS-SD content description and endpoint location
Detecting available source

• Discovery mechanism
  • mDNS for broadcasting and discovering sources,
  • DNS-SD content description and endpoint location
  • SPARQL 1.1 SDs and VoID descriptions of source content.

• Advantages:
  • approach addresses the NNEC needs
  • it is independent of a central registry,
  • eliminates the issue of network fragmentation.
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,
  - DNS-SD content description and endpoint location
  - SPARQL 1.1 SDs and VoID descriptions of source content.

- Advantages:
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,
  - DNS-SD content description and endpoint location
  - SPARQL 1.1 SDs and VoID descriptions of source content.

- Advantages:
  - approach addresses the NNEC needs
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,
  - DNS-SD content description and endpoint location
  - SPARQL 1.1 SDs and VoID descriptions of source content.

- Advantages:
  - approach addresses the NNEC needs
  - it is independent of a central registry,
Detecting available source

- Discovery mechanism
  - mDNS for broadcasting and discovering sources,
  - DNS-SD content description and endpoint location
  - SPARQL 1.1 SDs and VoID descriptions of source content.

- Advantages:
  - approach addresses the NNEC needs
  - it is independent of a central registry,
  - eliminates the issue of network fragmentation.
Table of contents

1 The Nato Network Enabled Capability (NNEC) concept
2 Example: Evacuation planning
3 System overview
4 Source Selection
5 Reducing the size of the cropping
Shortcomings

• Problem:

• CONSTRUCT queries are too unconstrained
• problem especially acute wrt. common predicates
• consider e.g. ?a rdf:type ?o
• this pattern will most likely be routed to every endpoint
• thus, downloading potentially huge amounts of data

• Proposed solution:
• assess the selectivity of triple patterns
• build the cropping incrementally
• within the confines of HTTP minimality
Shortcomings

- Problem:
  - *CONSTRUCT* queries are too unconstrained

  - This pattern will most likely be routed to every endpoint,
    thus, downloading potentially huge amounts of data.

- Proposed solution:
  - Assess the selectivity of triple patterns
  - Build the cropping incrementally
  - Within the confines of HTTP minimality
Shortcomings

- Problem:
  - `CONSTRUCT` queries are too unconstrained
  - problem especially acute wrt. common predicates
Shortcomings

- Problem:
  - `CONSTRUCT` queries are too unconstrained
  - problem especially acute wrt. common predicates
  - consider e.g. `?a rdf:type ?o`
Shortcomings

• Problem:
  • CONSTRUCT queries are too unconstrained
  • problem especially acute wrt. common predicates
  • consider e.g. ?a rdf:type ?o
  • this pattern will most likely be routed to every endpoint
Shortcomings

- Problem:
  - `CONSTRUCT` queries are too unconstrained
  - problem especially acute wrt. common predicates
  - consider e.g. `?a rdf:type ?o`
  - this pattern will most likely be routed to *every* endpoint
  - thus, downloading potentially huge amounts of data
Shortcomings

- Problem:
  - **CONSTRUCT** queries are too unconstrained
  - problem especially acute wrt. common predicates
  - consider e.g. `?a rdf:type ?o`
  - this pattern will most likely be routed to *every* endpoint
  - thus, downloading potentially huge amounts of data
Shortcomings

- **Problem:**
  - CONSTRUCT queries are too unconstrained
  - problem especially acute wrt. common predicates
  - consider e.g. `?a rdf:type ?o`
  - this pattern will most likely be routed to every endpoint
  - thus, downloading potentially huge amounts of data

- **Proposed solution:**
Shortcomings

• Problem:
  • `CONSTRUCT` queries are too unconstrained
  • problem especially acute wrt. common predicates
  • consider e.g. `?a rdf:type ?o`
  • this pattern will most likely be routed to *every* endpoint
  • thus, downloading potentially huge amounts of data

• Proposed solution:
  • assess the selectivity of triple patterns
Shortcomings

- **Problem:**
  - CONSTRUCT queries are too unconstrained
  - problem especially acute wrt. common predicates
  - consider e.g. ?a rdf:type ?o
  - this pattern will most likely be routed to every endpoint
  - thus, downloading potentially huge amounts of data

- **Proposed solution:**
  - assess the selectivity of triple patterns
  - build the cropping incrementally
Shortcomings

• Problem:
  • CONSTRUCT queries are too unconstrained
  • problem especially acute wrt. common predicates
  • consider e.g. ?a rdf:type ?o
  • this pattern will most likely be routed to every endpoint
  • thus, downloading potentially huge amounts of data

• Proposed solution:
  • assess the selectivity of triple patterns
  • build the cropping incrementally
  • within the confines of HTTP minimality
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
- \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)
- Less is less likely to produce large results
- Compare query patterns with the all-some lifting of \(\prec\)
- Nonexclusive groups more weight than exclusive groups
- Some patterns are preselected for the lowest level, e.g.
  
  \[
  ?s \quad ?p \quad ?o \quad ?s \quad rdfs\text{-}type \quad ?o
  \]
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
- \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)
- \(\prec\)-less is less likely to produce large results
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
- \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)
- \(\prec\)-less is less likely to produce large results
- compare query patterns with the all-some lifting of \(\prec\)
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
- \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)
- \(\prec\)-less is less likely to produce large results
- compare query patterns with the all-some lifting of \(\prec\)
- nonexclusive groups more weight than exclusive groups
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
  \[(s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\]
- \(\prec\)-less is less likely to produce large results
- compare query patterns with the all-some lifting of \(\prec\)
- nonexclusive groups more weight than exclusive groups
- some patterns are preselected for the lowest level, e.g.

\[
\text{?s} \quad \text{?p} \quad \text{?o}
\]

\[
\text{?s} \text{ rdf:type } \text{?o}
\]
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
  
  \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)

- \(\prec\) -less is less likely to produce large results

- compare query patterns with the all-some lifting of \(\prec\)

- nonexclusive groups more weight than exclusive groups

- some patterns are preselected for the lowest level, e.g.
  
  - \(?s\ ?p\ ?o\)
The selectivity of triple patterns

- Heuristics based on structural variations in triple patterns
- \((s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec \ldots \prec (?, ?, ?)\)
- \(\prec\)-less is less likely to produce large results
- compare query patterns with the all-some lifting of \(\prec\)
- nonexclusive groups more weight than exclusive groups
- some patterns are preselected for the lowest level, e.g.
  - ?s ?p ?o
  - ?s rdf:type ?o
Incremental construction of the cropping:

Priority levels:
- \( t_{31}, \ldots, t_{3m} \)
- \( t_{21}, \ldots, t_{1l} \)
- \( t_{11}, \ldots, t_{1k} \)

Execution process:
- \( R_1 \) to \( R_n \)
- Values
- Select
- Construct
Summary of key properties

- federated query answering is provably sound/complete
- only one HTTP request is sent to each source
- query answering is tractable, i.e. takes only polynomially many computation steps
- source selection is dynamic and once-per-query
Current and future work

• Status:

• a formal theory of federation + join-order optimization
• a working implementation in Scala/Clojure
• a demo that runs against real military databases

• Prioritized future work:

• adapt the theory and the implementation to data streams
• address challenges related to lifting of sensor data
• design a general purpose system around the current core
Current and future work

• Status:
  • a formal theory of federation + join-order optimization

• Prioritized future work:
  • adapt the theory and the implementation to data streams
  • address challenges related to lifting of sensor data
  • design a general purpose system around the current core
Current and future work

- **Status:**
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure

- Prioritized future work:
  - adapt the theory and the implementation to data streams
  - address challenges related to lifting of sensor data
  - design a general purpose system around the current core
Current and future work

- Status:
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure
  - a demo that runs against real military databases

- Prioritized future work:
  - adapt the theory and the implementation to data streams
  - address challenges related to lifting of sensor data
  - design a general purpose system around the current core
Current and future work

- **Status:**
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure
  - a demo that runs against real military databases

- **Prioritized future work:**
Current and future work

- **Status:**
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure
  - a demo that runs against real military databases

- Prioritized future work:
  - adapt the theory and the implementation to data streams
Current and future work

- **Status:**
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure
  - a demo that runs against real military databases

- **Prioritized future work:**
  - adapt the theory and the implementation to data streams
  - address challenges related to lifting of sensor data
Current and future work

- **Status:**
  - a formal theory of federation + join-order optimization
  - a working implementation in Scala/Clojure
  - a demo that runs against real military databases

- **Prioritized future work:**
  - adapt the theory and the implementation to data streams
  - address challenges related to lifting of sensor data
  - design a general purpose system around the current core