Graph Analysis Techniques for Network Flow Records Using Open Cyber Ontology Group (OCOG) Format

Robert W. Techentin

David R. Holmes, III

James C. Nelms

Barry K. Gilbert

Presented to FloCon 2016, Daytona Beach, FL

January 12, 2016

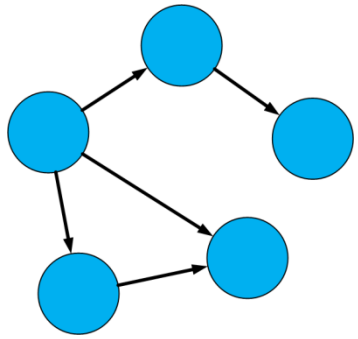
MAYO CLINIC

Outline

- Open Cyber Ontology Group (OCOG) Netflow Format
- SPARQL Query Language for Semantic Graphs
- Examples of SiLK and SPARQL
- Extending the Semantic Data Model
- Graph Characteristics, Patterns, and Algorithms

What Are Semantic Graphs

- W3C created the Resource Description Framework (RDF) standard to facilitate data interchange on the web
  - Links data with named relationships
  - Allows the evolution of schemas over time
- Data objects are vertices in the RDF Graph
- Relationships are the named edges
- Graphs are described as “triples”
  - Subject → Predicate → Object



- See http://www.w3.org/RDF/ for details and tools

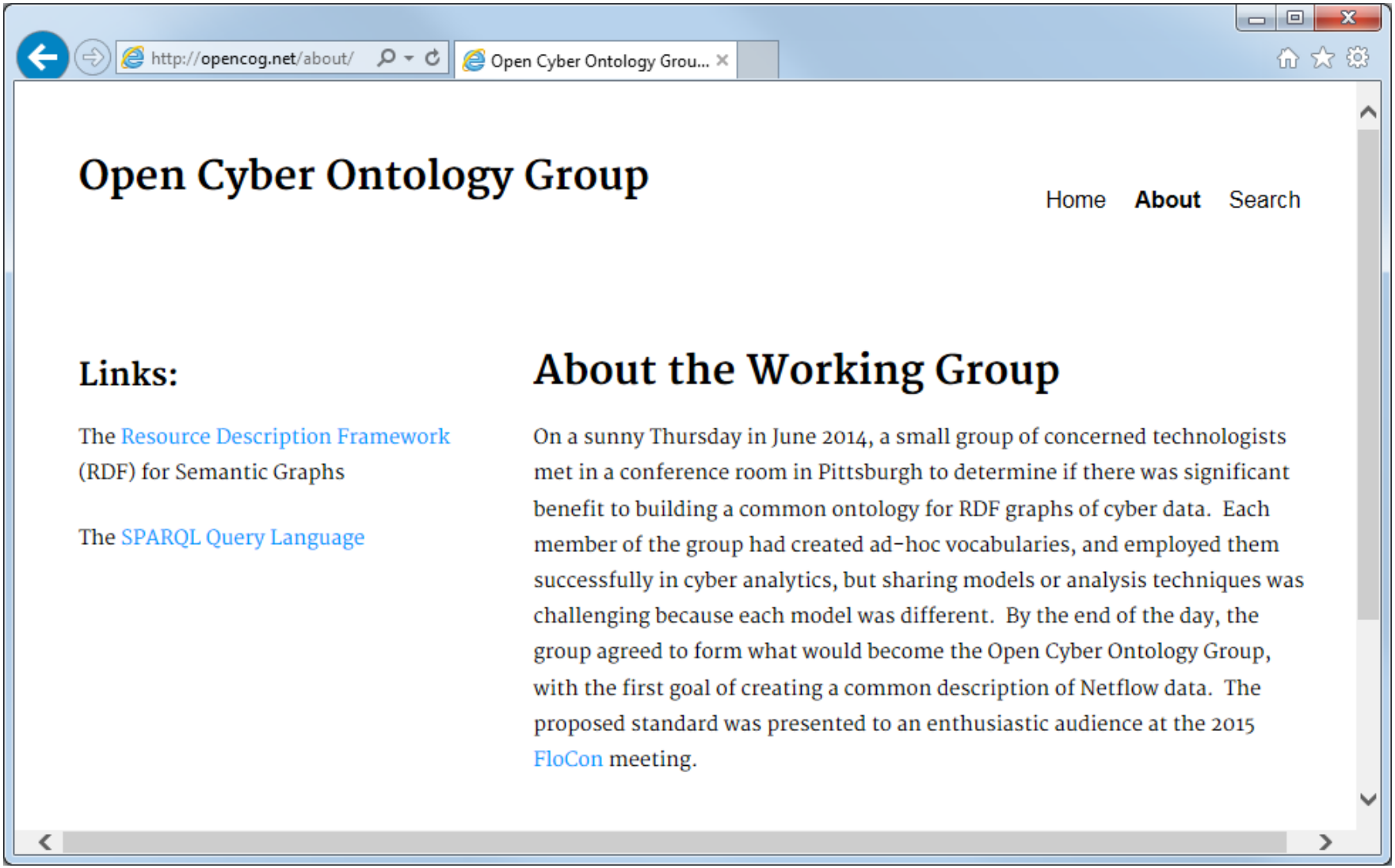
Why Semantic Graph Analysis for Netflow?

- Integration of other data sources (e.g., IANA, CIDR, DNS, user and asset data) is straightforward
- Graph patterns can identify complex behavioral relationships
- Graph analytic techniques can provide new insights into network data
  - They evaluate relationships and connections, instead of just statistics
- Graph analytic technologies are maturing
  - RDF and SPARQL (e.g., Cray Urika, Apache Jena, Virtuoso)
  - Other languages (e.g., Neo4j, Apache Titan, GraphBase)

Mayo Clinic Cyber Model (MCCM) and Open Cyber Ontology Group (OCOG)

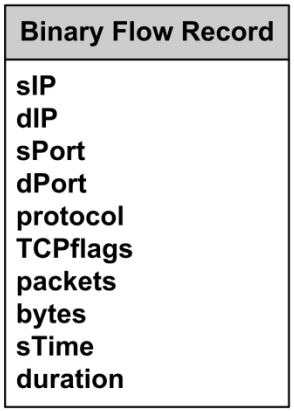
- Mayo began developing MCCM in 2013
  - Includes Netflow, DNS, DHCP, IANA port numbers, network structure, and assets owned by different business units (and other data)
- However, Mayo and Cray (and others) had different approaches and naming conventions, even for simple things like port numbers
- OCOG formed in 2014 to develop a common ontology for common concepts (i.e., don't reinvent the wheel)
- Members: Mayo, CERT, Cray, PSC, PNNL
- *"Semantic Representations of Network Flow"* at FloCon 2015

http://opencog.net/

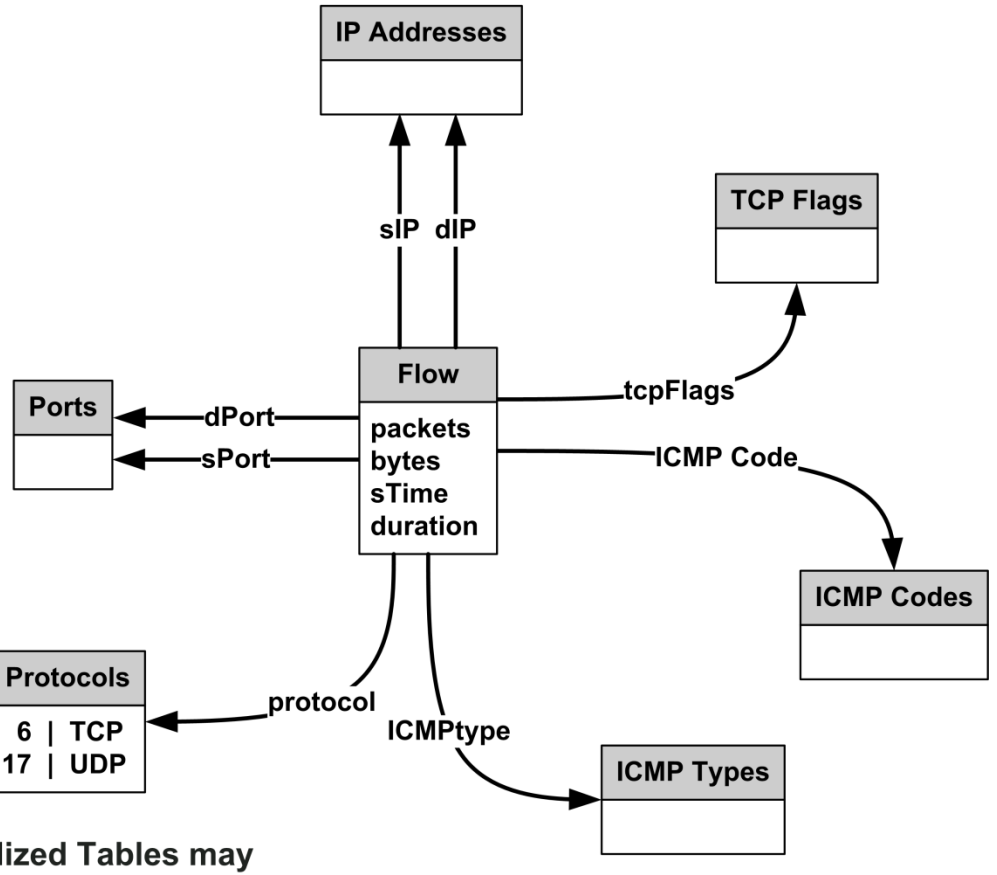


NETFLOW DATA RECORDS MAPPED TO NORMALIZED RELATIONAL DATABASE SCHEMA
(Tables in Notional Schema Represent Lists of Distinct Values from Columns in the Binary Record Format)

Flow Collectors Such as SiLK 'flowcap' typically store flow records in a compact binary format

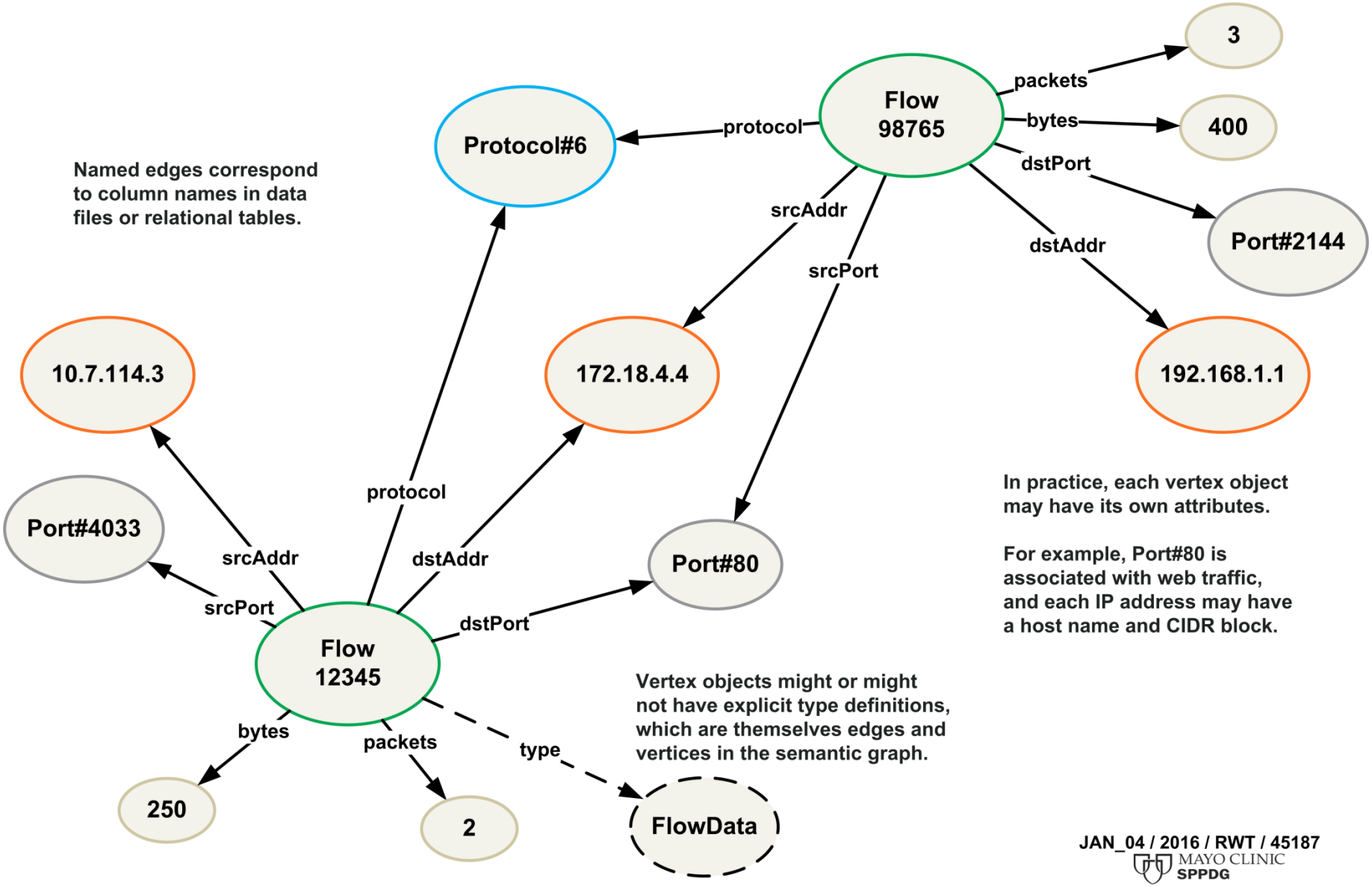


Normalized Relational Database Schemas Map Record Columns into Separate Tables to Minimize Data Redundancy



Normalized Tables may Supply More Information or Link to Other Data Tables

OPEN CYBER ONTOLOGY GROUP (OCOG) NETFLOW MODEL
DEFINES LINKS BETWEEN INSTANCES OF FLOW DATA ELEMENTS
(Semantic Data Graph Vertex Objects are Connected by Labeled Directed Edges; Vertices are Colored for Clarity; Only Two Flows are Shown for Simplicity)



MAYO CLINIC
SPPDG

SPARQL Syntax Example

SELECT describes what we want

The prefix "**oco:**" stands for Open Cyber Ontology, and is a shortcut for readability for constants.

```
PREFIX oco: <http://opencog.net/>

SELECT ?sIP
WHERE {
  ?flow oco:srcAddr ?sIP.
}
```

Variables begin with "?"

This pattern is a "triple" describing a relationship:
"source" "predicate" "object"
Akin to:
"subject" "verb" "direct object"

Comparing SiLK and OCOG/SPARQL

- SiLK examples from the literature†
- SPARQL queries are composed using OCOG syntax to illustrate concepts familiar to SiLK practitioners
- Results are edited to protect proprietary information
- RDF results are formatted for readability
  - For example, this triple

```
<http://opencog.net/collector#9Rs1VNvcZrPu17>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://opencog.net/ocoVersion>
```

  - Is formatted as

```
oco:collector#9Rs1VNvcZrPu17  rdf:type  oco:ocoVersion
```

† ***Network Profiling Using Flow***, **CERT Technical Report, by Austin Whisnant and Sid Faber**

Query Metadata: SiLK

```
$ rwfileinfo sample.rw

sample.rw:
  format(id)          FT_RWIPV6ROUTING(0x0c)
  version             16
  byte-order          littleEndian
  compression(id)     zlib(1)
  header-length       352
  record-length       88
  record-version      1
  silk-version        3.10.2
  count-records       191005464
  file-size           1669946180
```

Query Metadata: OCOG SPARQL - 1

```
SELECT ?property ?value
WHERE {
  ?collector rdf:type oco:Collector .
  ?collector ?property ?value .
}
```

property	value
rdf:type	oco:Collector
oco:exporterAddr	oco:ipv4#10.100.1.1
oco:flowdataFilename	“sample.nt”
oco:conversionStartTime	“2015-12-10T08:37:24”
oco:ocoVersion	“v1.0"
oco:ocoLevel	oco:ocogLevel#3
oco:software	"Mayo Clinic OCOG Reference Translator v1.0"

Query Metadata: OCOG SPARQL - 2

```
SELECT ?collector (COUNT(?flow) AS ?flow_count)
WHERE {
  ?flow oco:collector ?collector .
}
GROUP BY ?collector
```

collector	flow_count
oco:collector#9Rs1VNvcZrPu17	402568585

Query 1: Metadata

SiLK

```
$ rwfileinfo sample.rw
```

SPARQL

```
SELECT ?property ?value
WHERE {
  ?collector rdf:type oco:Collector .
  ?collector ?property ?value .
}
```

The OCOG specification calls for a metadata object in each dataset, associated with the data collector and/or exporter and the software capture pipeline. Every flow may be linked to its collector object, which is useful when integrating many datasets. The links to the collectors may be omitted to save space.

Query 2: Protocol Statistics

SiLK

```
$ rwstats sample.rw --fields=protocol --count=5

INPUT: 10985967 Records for 7 Bins and 10985967 Total Records
OUTPUT: Top 5 Bins by Records
pro| Records| %Records| cumul_%|
  6| 7302815| 66.474030| 66.474030|
 17| 3605304| 32.817357| 99.291387|
  1|   72762|  0.662318| 99.953705|
 50|    5079|  0.046232| 99.999936|
 ...
```

SPARQL

```
SELECT ?protocol (COUNT(?flow) AS ?records)
WHERE {
  ?flow oco:protocol ?protocol .
}
GROUP BY ?protocol
ORDER BY DESC(?records)
LIMIT 5
```

SPARQL Queries can COUNT(), SUM(), AVG() or find MIN() or MAX()

GROUP BY and ORDER BY operate on any parameters

Query 3: Listing Flows

SiLK

```
$ rwcut sample.rw --fields=1-5,packets --num-recs=10

            sIP|             dIP| sPort | dPort | pro| packets |
   192.0.2.226|  192.168.200.39|  11229|  51015|   6|       21|
   192.0.2.226|  192.168.200.39|  34075|  44230|   6|       21|
   192.0.2.226|  192.168.200.39|  23347|  33503|   6|       21|
  203.0.113.15| 192.168.111.219|  59475|  57359|   6|      153|
   ...
```

SPARQL

```
SELECT ?sIP ?dIP ?sPort ?dPort ?protocol ?packets
WHERE {
  ?flow oco:srcAddr ?sIP .
  ?flow oco:dstAddr ?dIP .
  ?flow oco:srcPort ?sPort .
  ?flow oco:dstPort ?dPort .
  ?flow oco:packets ?packets .
  ?flow oco:protocol ?protocol .
}
LIMIT 10
```

This is a “Basic Graph Pattern” in SPARQL. All triples must be matched to produce one record for the solution.

Query 4: Counting Flows

SiLK

```
$ rwuniq sample.rw --fields=sIP | head –n 10

            sIP|   Records|
  10.213.205.29|         4|
  10.108.230.48|      4348|
  10.201.114.31|        34|
 10.232.242.192|        22|
   ...
```

SPARQL

```
SELECT ?sIP (COUNT(?flow) AS ?records)
WHERE {
  ?flow oco:srcAddr ?sIP .
}
GROUP BY ?sIP
LIMIT 10
```

SPARQL COUNT() Queries can be GROUPED BY or ORDERED BY any combination of parameters, or filtered with HAVING clauses with constraints

Relative Performance of SiLK and OCOG/SPARQL

Query	SiLK Time* (s)	SPARQL Time+ (s)
Metadata	5	1 + 3
Statistics	72	45
List Flows	0	61
Count Flows	82	29

* SiLK query times for 191 M records on Cray XT5 compute node, Dual AMD Opteron 2.6 GHz CPU, 12 Cores, 32 GB DDR2 RAM, Lustre RAID file system

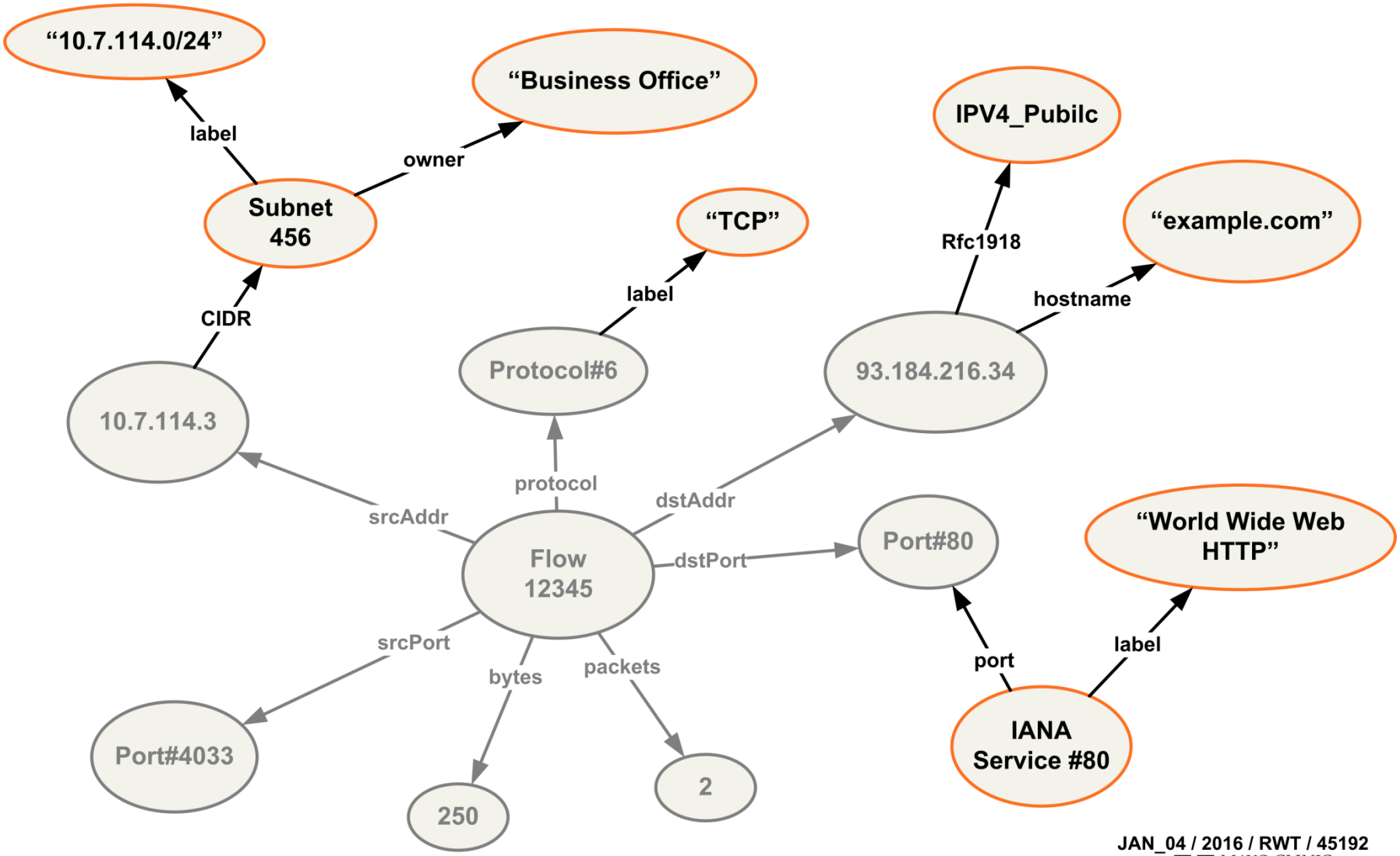
+ SPARQL query times for 400 M records on Cray Urika GD Appliance, 2 TB shared DDR2 RAM, 8192 hardware threads

Extending The Semantic Data Model with SPARQL UPDATE

- We can easily extend the OCOG data model by simply adding more links to the data
- In a similar vein, SiLK supports creation and manipulation of IPsets, Bags, and Prefix Maps
- However, in a semantic graph, any data can be added
  - Annotations of IP address behavior
  - Network topology
  - Qualitative labels for “unusual” things
  - Enterprise data about assets and users

OPEN CYBER ONTOLOGY GROUP (OCOG) NETFLOW MODEL EXTENDED BY ADDING NEW DATA ELEMENTS TO SEMANTIC GRAPH

(Extending the Resource Description Framework (RDF) Model Simply Requires Adding "Triples" Linking to Existing Data Objects)



JAN_04 / 2016 / RWT / 45192

MAYO CLINIC
SPPDG

Example of Extending the Network Data Model

- Example from literature: Identify “TCP Web Talkers” on ports 80, 8080, and 443
- In SiLK, we create an “IP set” of addresses that are (likely) offering web services
- In SPARQL, we add data to the graph
  - You could add almost any reference to the IP address
  - We choose to add a “type” of “mail server”

Identify Email Servers

SiLK

```
$ rwfilter sample.rw --type=out \
--protocol=6 --ack-flag=1 --packets=4- --sport=25,465,110,995,143,993 \
--pass=stdout \
| rwset --sip-file=smtp_servers.set
```

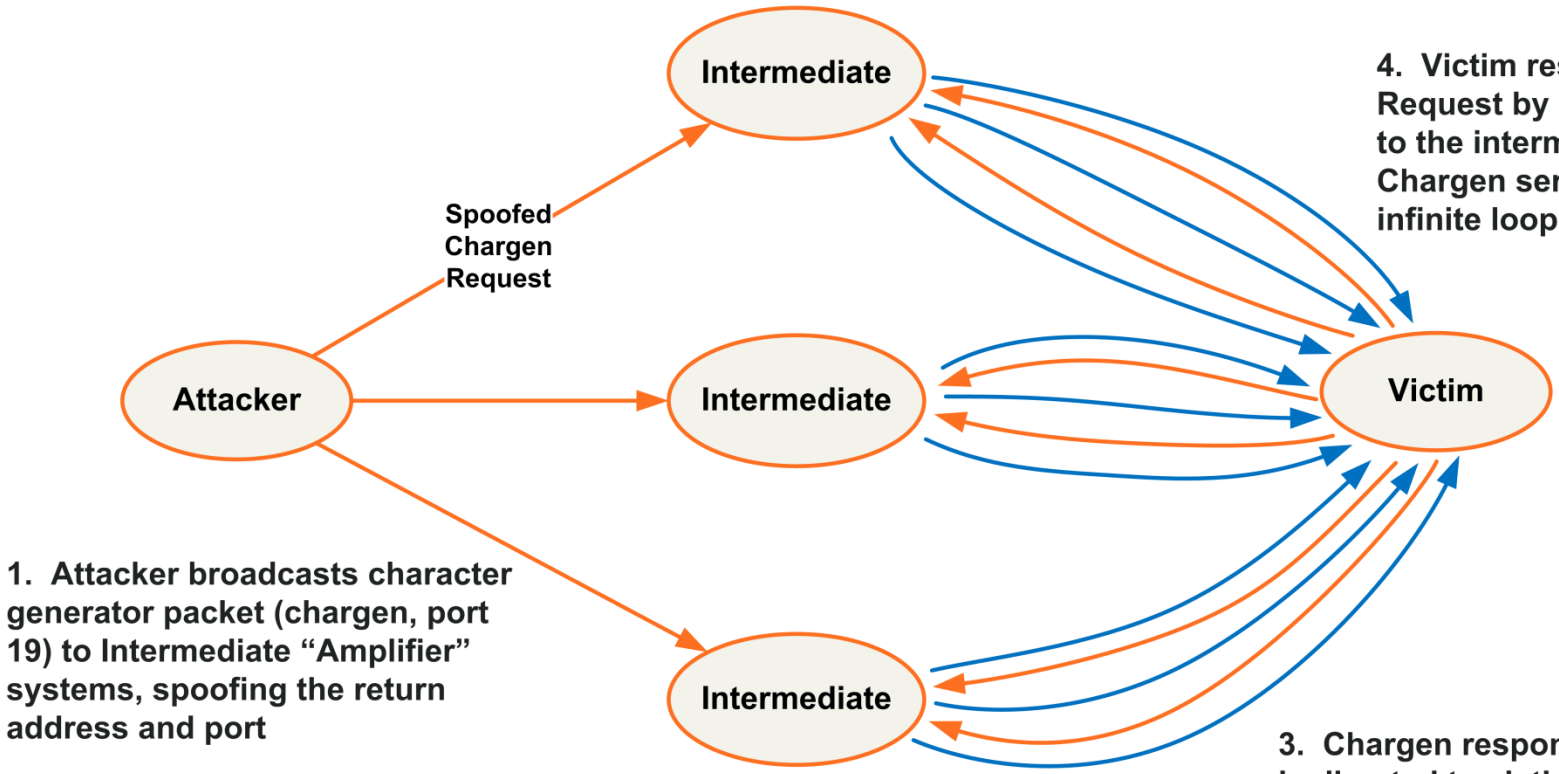
SPARQL

```
INSERT { ?sIP rdf:type <urn:mailServer> . }
WHERE {
  ?flow oco:srcAddr ?sIP .
  ?flow oco:srcPort ?sPort .
  FILTER(?sPort IN( oco:port#25,  oco:port#465, oco:port#110,
                    oco:port#995, oco:port#143, oco:port#993 ))
  ?flow oco:protocol oco:protocol#6 .
  ?flow oco:tcpFlags ?all_flags .
  ?all_flags oco:tcpFlag oco:tcpFlag/ACK .
  ?flow oco:packets ?packets .
}
HAVING(?packets > 4)
```

Graph Characteristics and Patterns

- Graphs have implicit characteristics that can be useful when analyzing netflow data
  - In-Degree and Out-Degree can be a simple metric for characterizing server behavior
- Graph patterns can be more complex than relations between flow data records
  - For example, listing user names for systems that are querying DNS with unusually long domain names
  - Multi-hop patterns between systems might characterize transactions from a client, through a distributed application (e.g., web server, application server, and database server)

GRAPH PATTERN TEMPLATE FOR NETWORK "FRAGGLE ATTACK" VARIATION THAT EXPLOITS CHARACTER GENERATOR SERVICE *



* Godiyal, A., M. Garland, and J.C. Hart: "Enhancing Network Traffic Visualization By Graph Pattern Analysis" (2010).

SPARQL Query to Detect Fraggle Attack Variant

```
SELECT ?victim (SUM(?echo_packets) AS ?echo_requests)
WHERE {
  ?echo oco:srcAddr ?intermediate .
  ?echo oco:srcPort oco:port#19 .
  ?echo oco:protocol oco:protocol#17 .
  ?echo oco:dstAddr ?victim .
  ?echo oco:dstPort oco:port#7 .
  ?echo oco:packets ?echo_packets .

  ?chargen oco:srcAddr ?victim .
  ?chargen oco:srcPort oco:port#7 .
  ?chargen oco:protocol oco:protocol#17 .
  ?chargen oco:dstAddr ?intermediate .
  ?chargen oco:dstPort oco:port#19 .
}
GROUP BY ?victim
ORDER BY DESC(?echo_requests)
```

This query identifies and counts complementary flows between “Fraggle Attack” intermediate and victim systems, matching UDP Echo Service and Character Generator Service requests

Encounter Complexes

- Described by Leigh Metcalf, *Encounter Complexes for Clustering Network Flow*, FloCon 2015.
- IP addresses “encounter” each other for the duration of a flow between them
- The Encounter Complex associates flows where
  - They share an IP address in common
  - The end of one occurs within Δ seconds of the start of the next
- Graphs of encounter complexes can be clustered for pattern analysis
  - e.g., Pearson coefficient

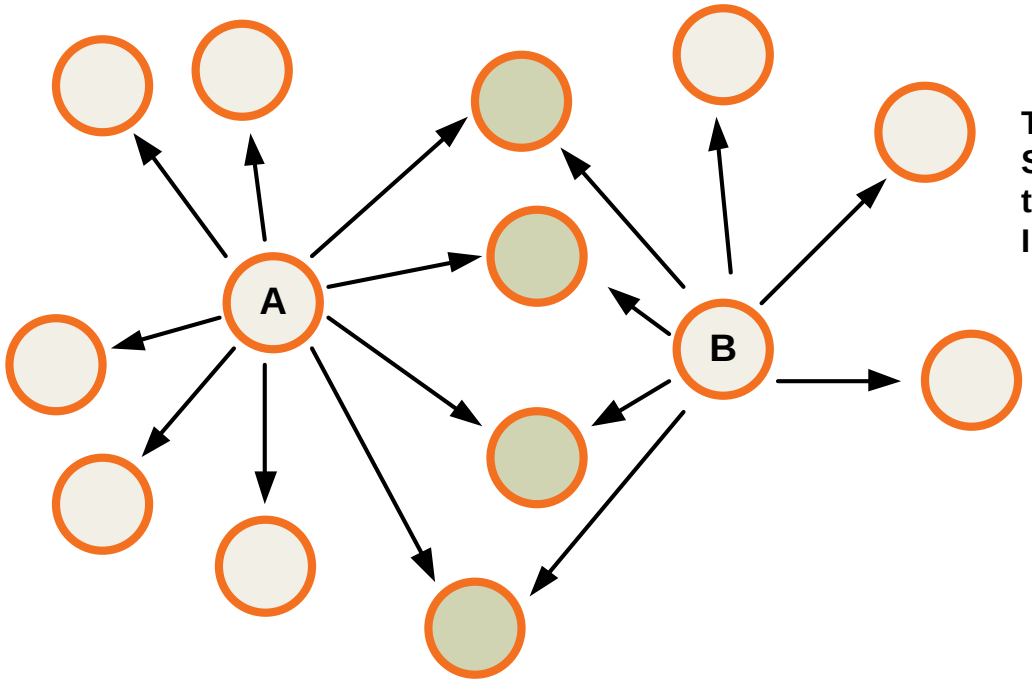
SPARQL Query for Encounter Complexes

```
# Construct new graph with Encounter Complexes
INSERT {
  GRAPH <urn:encounterComplexes> {
    ?flow1 <urn:inComplexWith> ?flow2 .
  }
}
WHERE {
  # Find a flow
  ?flow1 oco:srcAddr ?srcAddr .
  ?flow1 oco:dstAddr ?dstAddr .
  ?flow1 oco:start ?start .
  ?flow1 oco:duration ?duration .

  # Find other flows with matching source or destination
  { {?flow2 oco:srcAddr ?srcAddr .} UNION
    {?flow2 oco:srcAddr ?dstAddr .} UNION
    {?flow2 oco:dstAddr ?srcAddr .} UNION
    {?flow2 oco:dstAddr ?dstAddr .} }

  # Filter based on time similarity
  ?flow2 oco:start ?flow2Start .
  BIND(ABS(?start + ?duration – ?flow2Start) AS ?delta)
  FILTER(?delta <= 1000) # delta time in milliseconds
}
```

Graph Projections and Algorithms

- SPARQL queries and updates make it possible to construct new graphs from the data
- Projections can be made on any dimension
  - e.g., IP address, flow, protocol
- Graph algorithms, such as clustering or betweenness centrality, can reveal interesting behaviors on the network

SUBGRAPH SIMILARITY MEASUREMENT BY JACCARD INDEX
(Similarity of Graph Vertices and Edges Based on Set Theory)



The Jaccard Index Measures Subset Similarity as the Ratio of the Number of Elements in the Intersection and the Union

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

There are Several Options For Semantic Graphs

- **Count Typed Edges**
- **Count Unique Edge Types**
- **Count Incoming vs. Outgoing Edges**
- **Count Vertices**
- **Count Vertex Types**

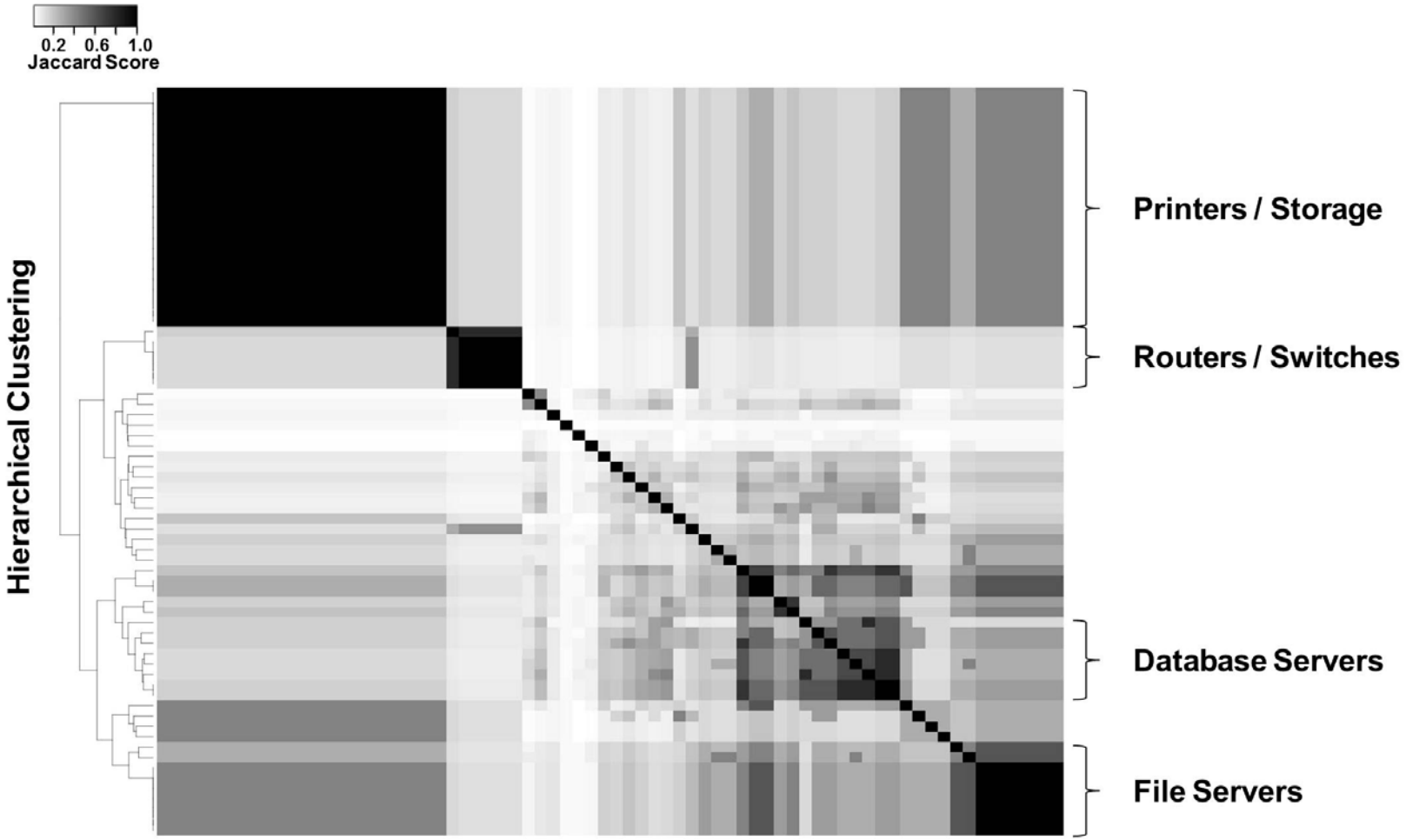
SPARQL Projection of Traffic Graph

```
INSERT {
  GRAPH <urn:ip_traffic> {
    ?srcAddr oco:talksTo ?dstAddr .
  }
}
WHERE {
  SELECT DISTINCT ?srcAddr ?dstAddr
  WHERE {
    ?flow oco:srcAddr ?srcAddr .
    ?flow oco:dstAddr ?dstAddr .
  }
}
```

- While this projection is simply source and destination address, more complex projections are easily implemented
  - Select only traffic for particular ports and protocols
  - Combine address / port / protocol into a distinct destination
  - Relate objects other than network systems (e.g., flows or ports)

NETWORK TRAFFIC SIMILARITY ANALYSIS FOR BIOMEDICAL IMAGING RESOURCE SUBNET

(Netflow Network Metadata Captured in Semantic Graph Database;
Jaccard Similarity Score Computed on Hybrid XMT-2 Supercomputer at Mayo Clinic;
Hierarchical Clustering of Similarity Scores in R-Project Language)



MAR_18 / 2015 / RWT / 44811

MAYO CLINIC
SPPDG

Conclusion

- The Open Cyber Ontology Group (OCOG) defined a common format for the representation of Flow data in RDF semantic graphs
- Data in RDF graphs in OCOG format can be queried for characteristics, much as can be done with the SiLK tool suite
- RDF and SPARQL queries and UPDATES offer added power for analyzing graph characteristics and creating useful projections of large network datasets for graph analytic or other analysis techniques