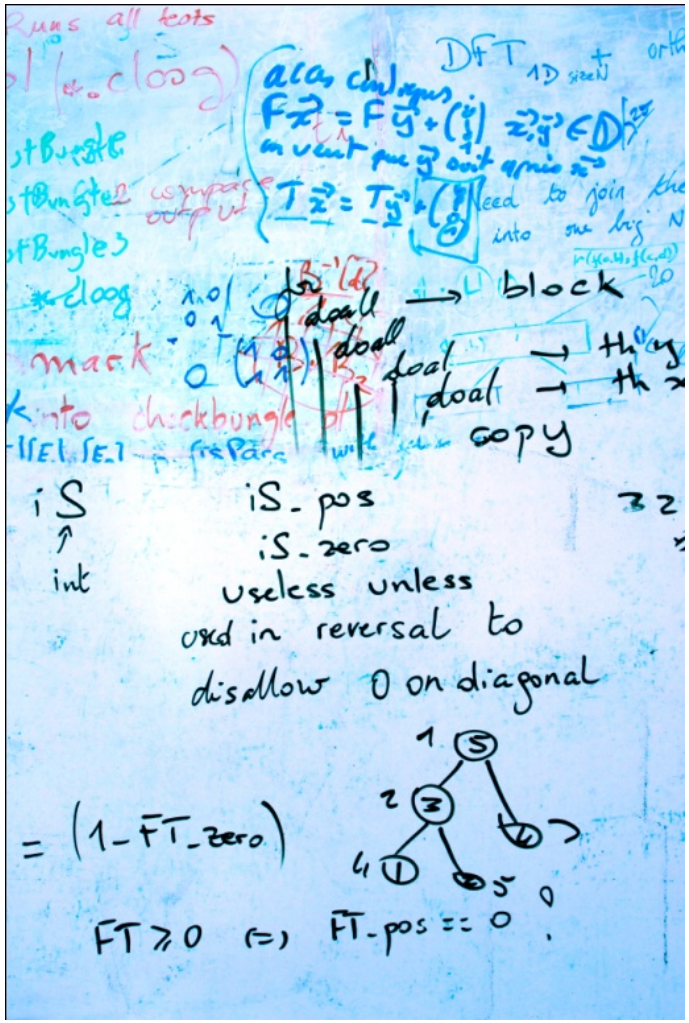
Moving Graph Analytics to Higher Dimensions to Discover Network Activity / Anomalies that are implicit in behavior - Connections in High Dimensions

James Ezick
Muthu Baskaran
Alan Commike
Aditya Gudibanda
Thomas Henretty
M. Harper Langston
Pierre-David Letourneau
Benoit Meister
Jordi Ros-Giralt
Jonathon Cai*
David Bruns-Smith*
Rachel Lawrence*
Grace Cimaszewski*
Richard Lethin
Reservoir Labs, Inc.
New York, NY

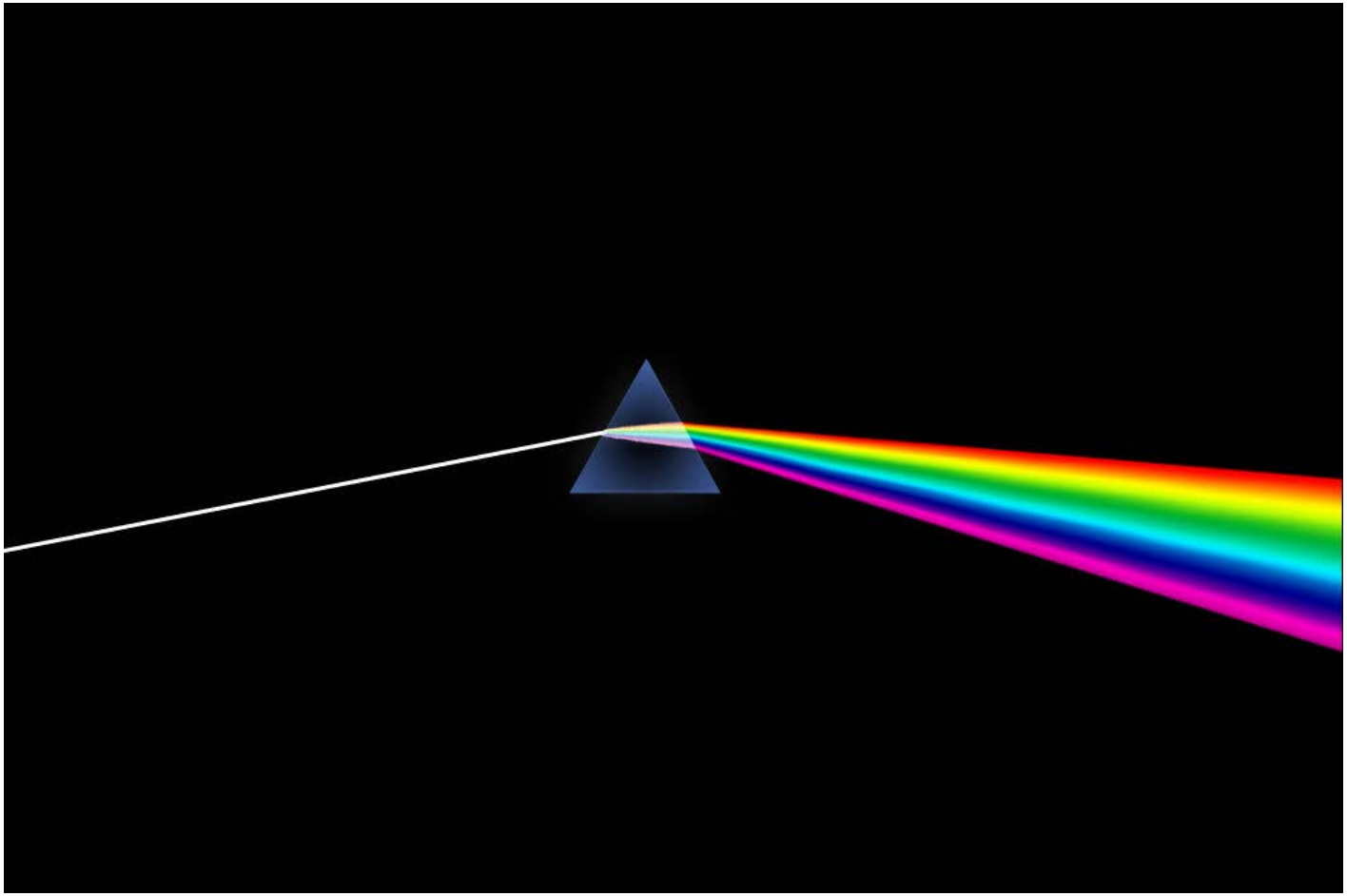
*Formerly – Interns, etc.

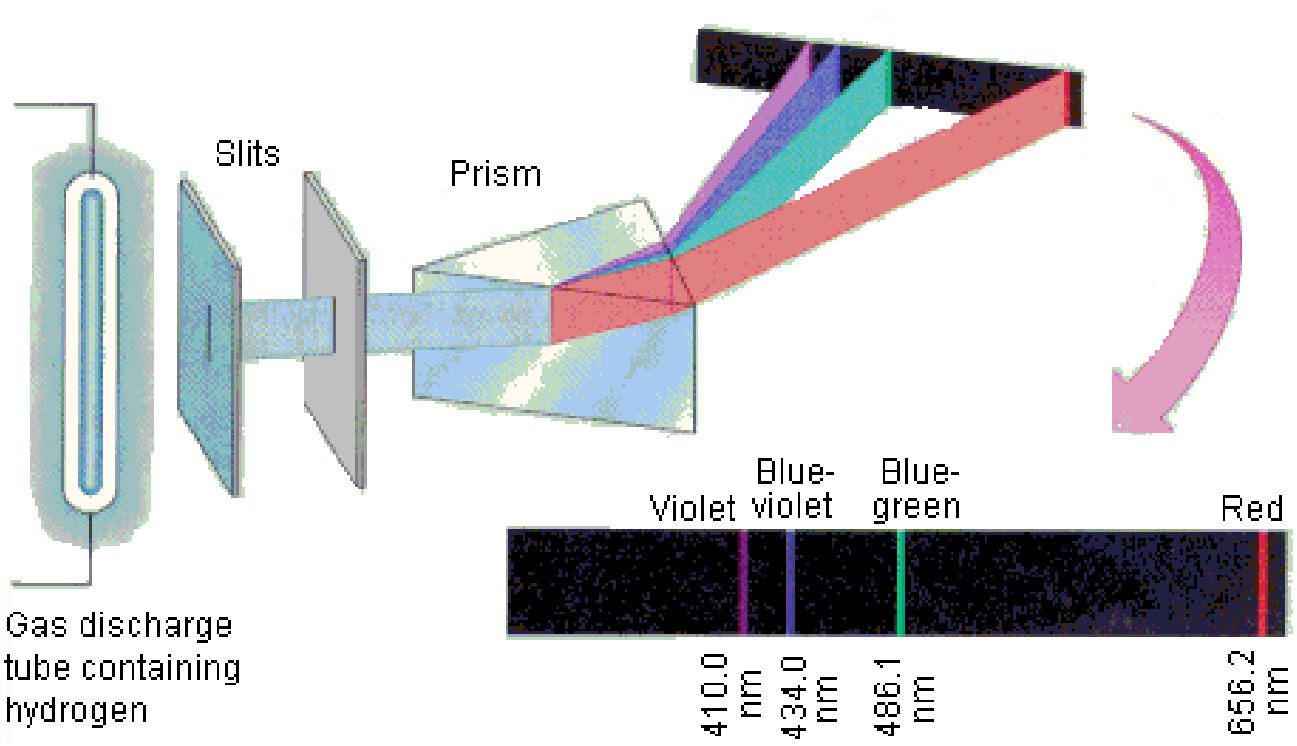


"Just as astronomers study stellar spectra to determine the make-up of distant stars, one of the main goals in graph theory is to deduce the principal properties and structure of a graph from its graph spectrum (or from a short list of easily computable invariants). The spectral approach for general graphs is a step in this direction. We will see that eigenvalues are closely related to almost all major invariants of a graph, linking one extremal property to another. There is no question that eigenvalues play a central role in our fundamental understanding of graphs."

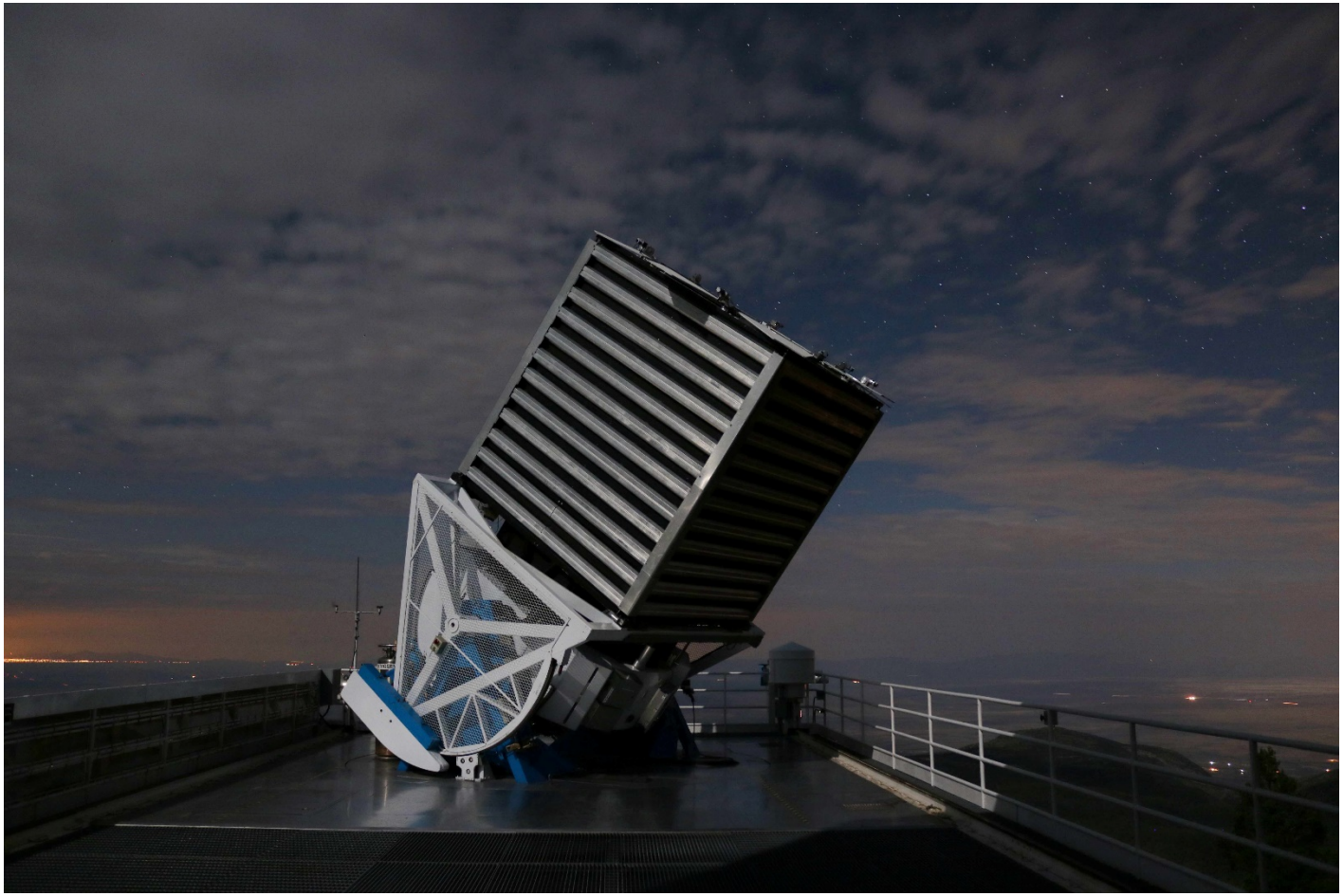
Fan Chung, "Spectral Graph Theory" 2006

Spectrum

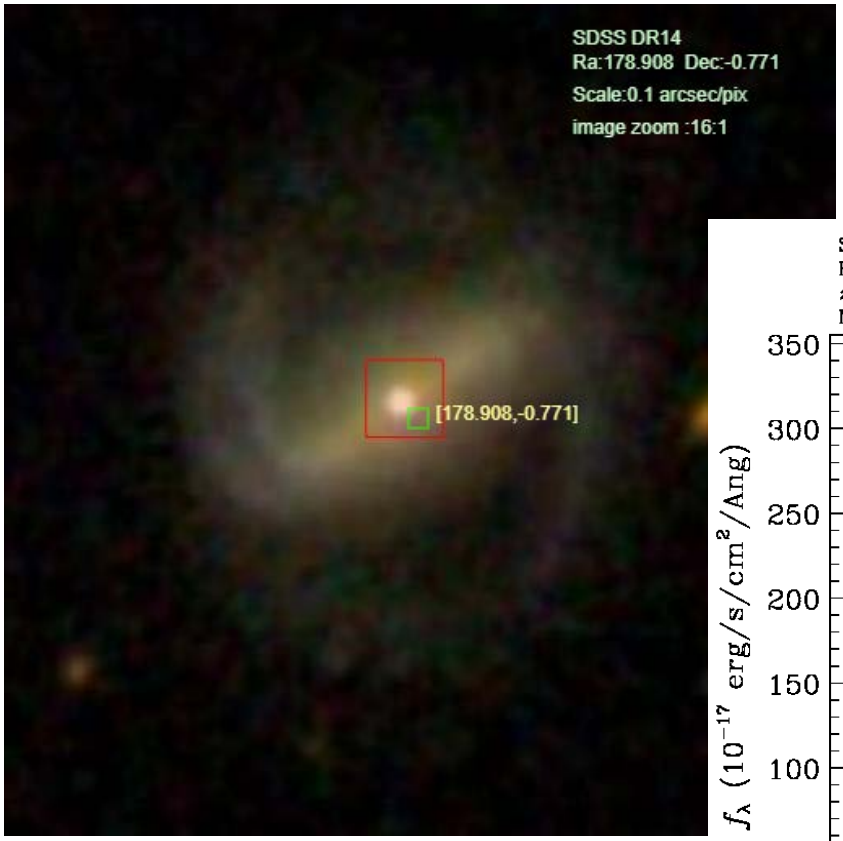


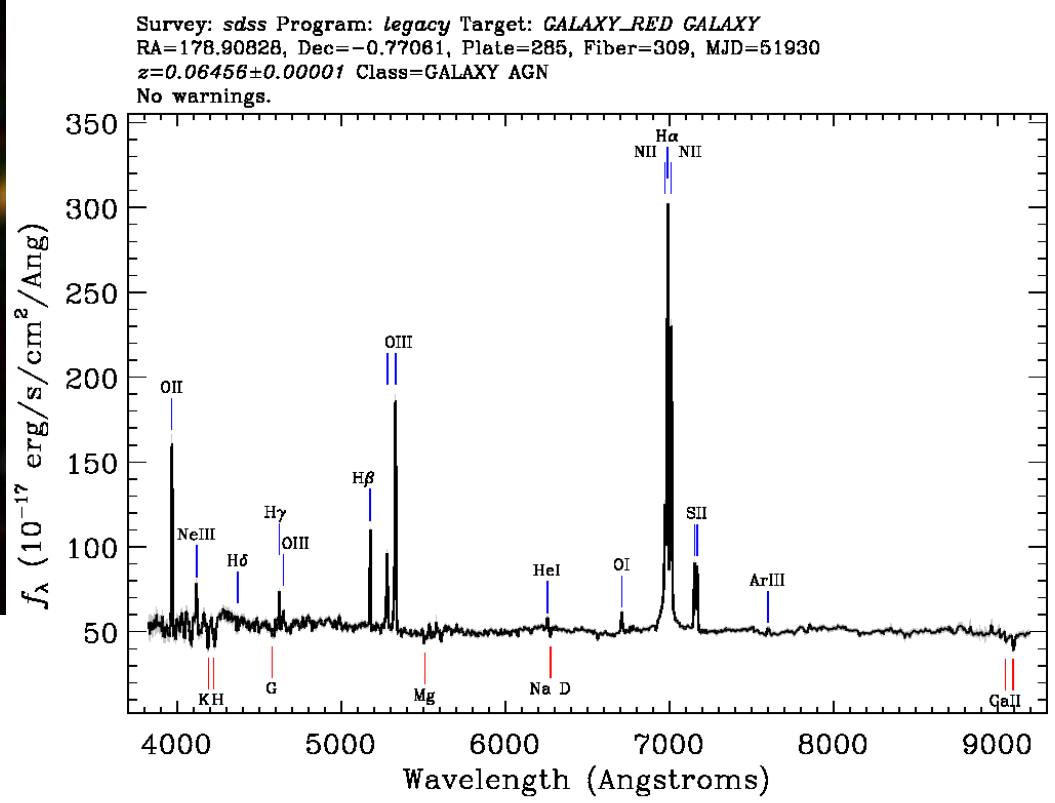

Slits
Prism
Violet
Blue-violet
Blue-green
Red
Gas discharge tube containing hydrogen
410.0 nm
434.0 nm
486.1 nm
656.2 nm

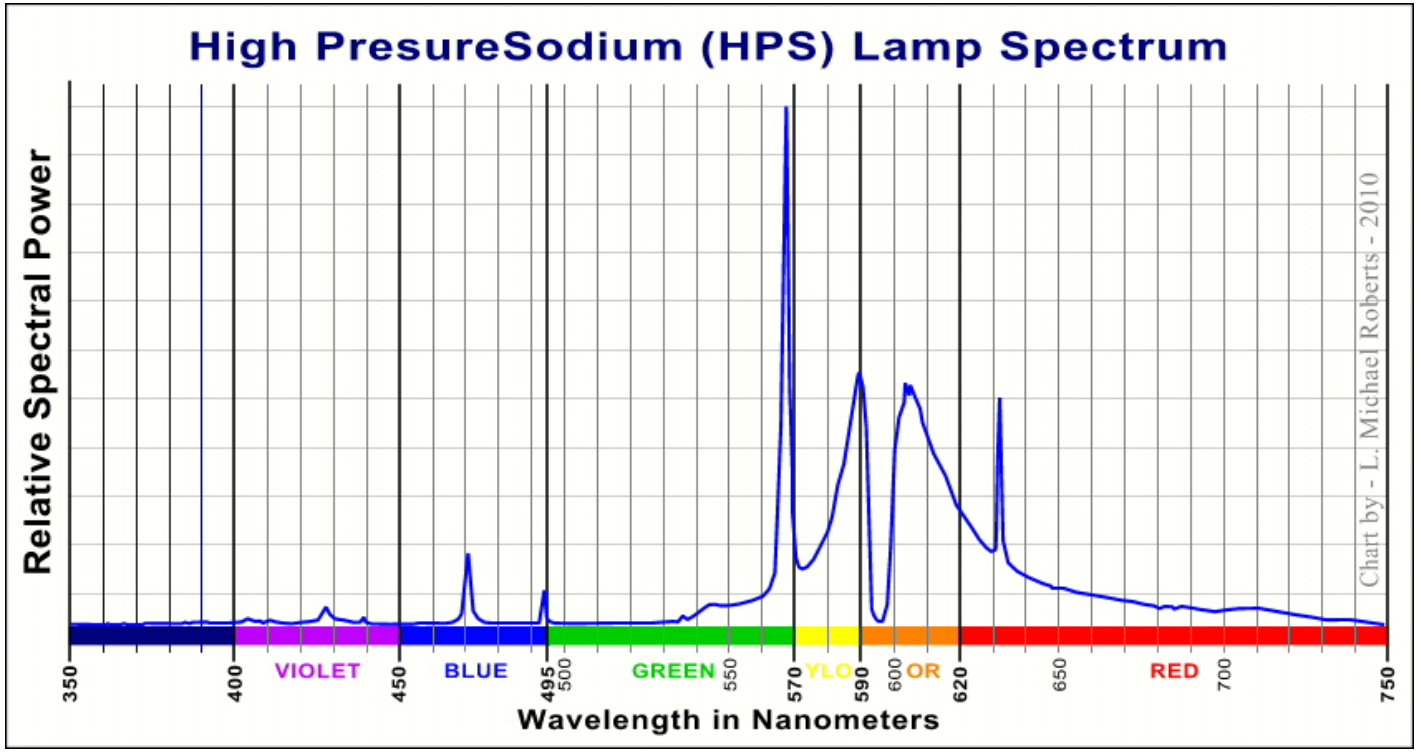
Looking at distant stars



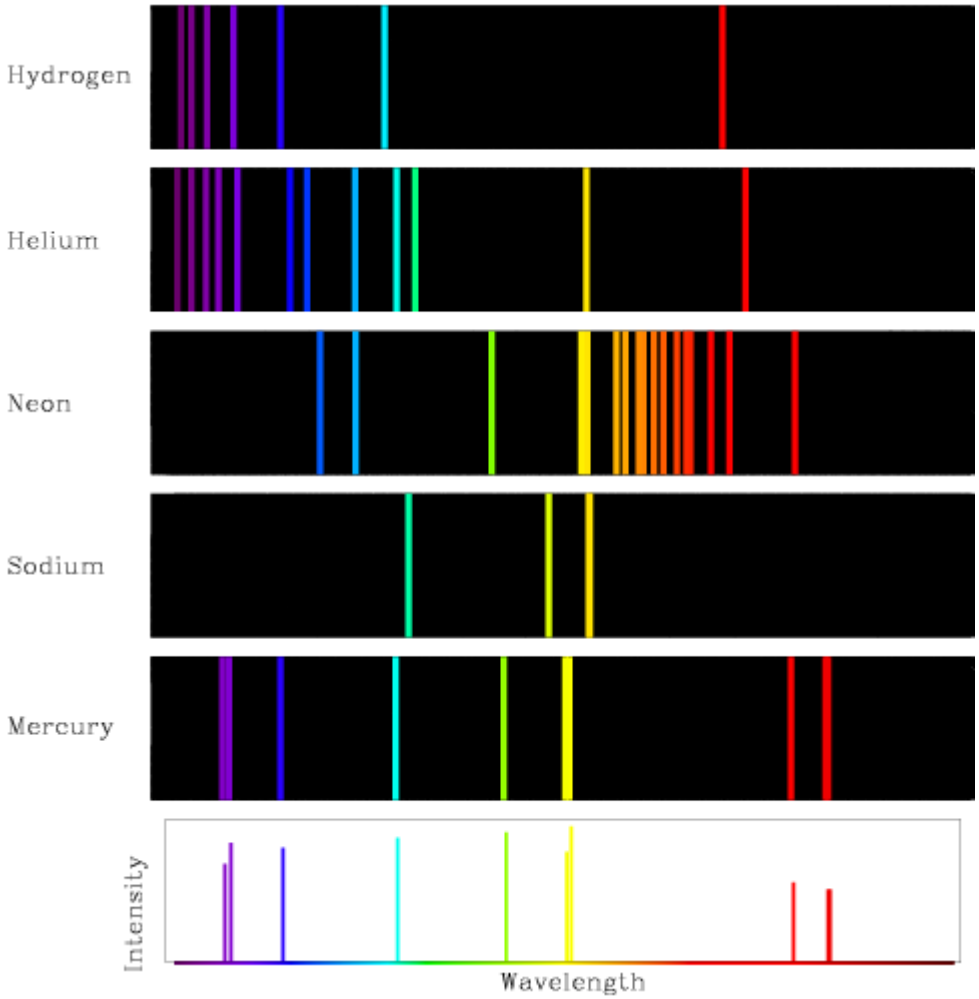
Source: Sloan Digital Sky Survey


SDSS DR14
Ra:178.908 Dec:-0.771
Scale:0.1 arcsec/pix
image zoom :16:1
[178.908,-0.771]

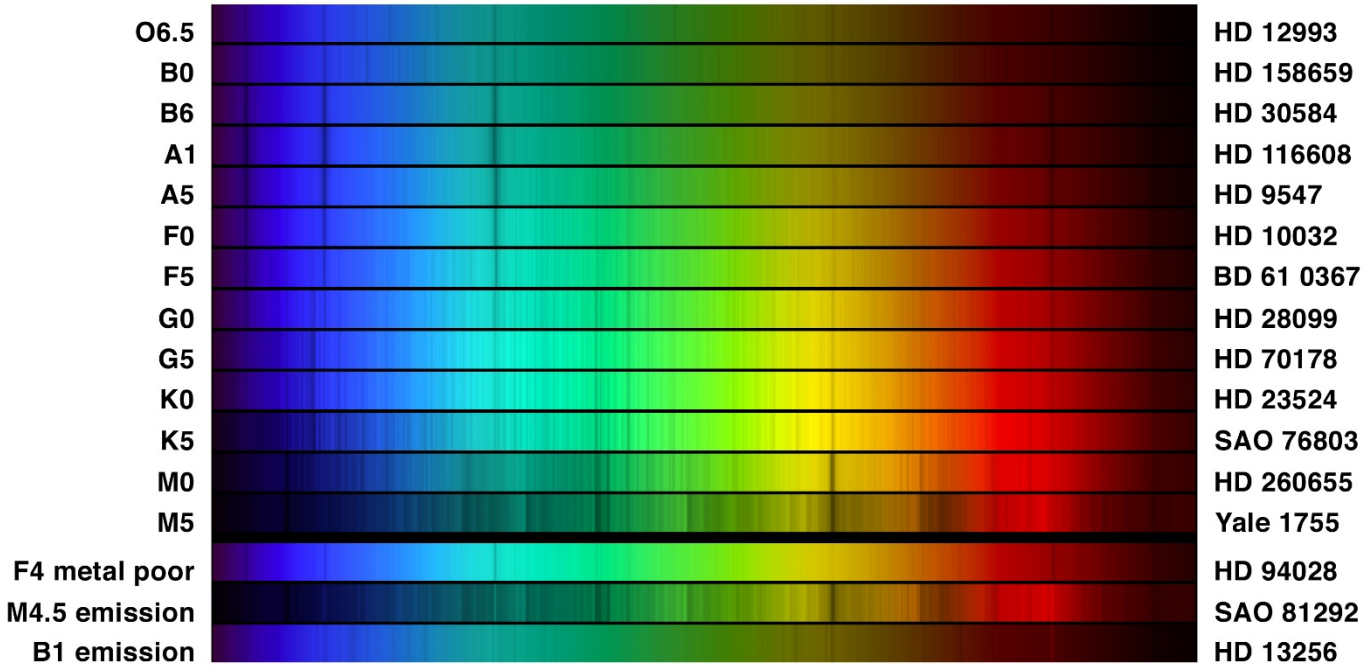

Survey: sdss Program: legacy Target: GALAXY_RED GALAXY
RA=178.90828, Dec=−0.77061, Plate=285, Fiber=309, MJD=51930
z=0.06456±0.00001 Class=GALAXY AGN
No warnings.
f_λ (10^{-17} erg/s/cm^2/Ang)
Wavelength (Angstroms)
OII
NeIII
Hδ
KH
Hγ
OIII
G
Hβ
OIII
Mg
HeI
Na D
OI
Hα
NII
NII
SII
ArIII
CaII



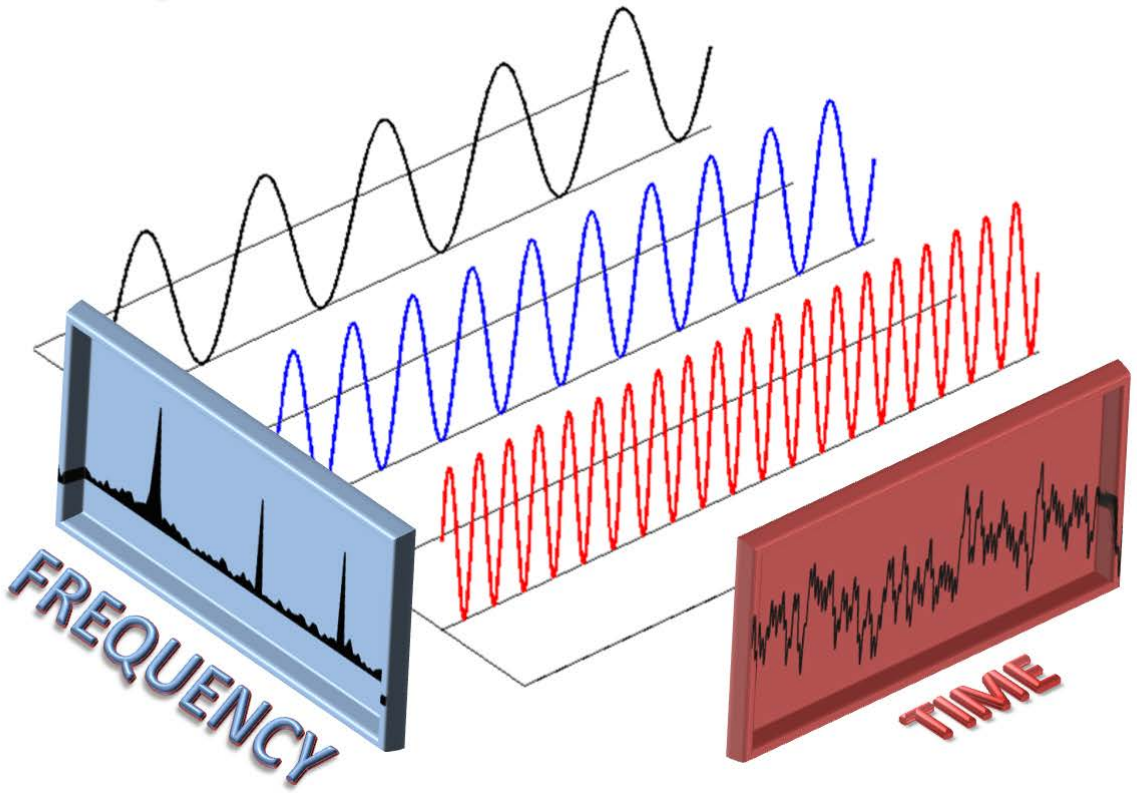
Source: Wikimedia.org

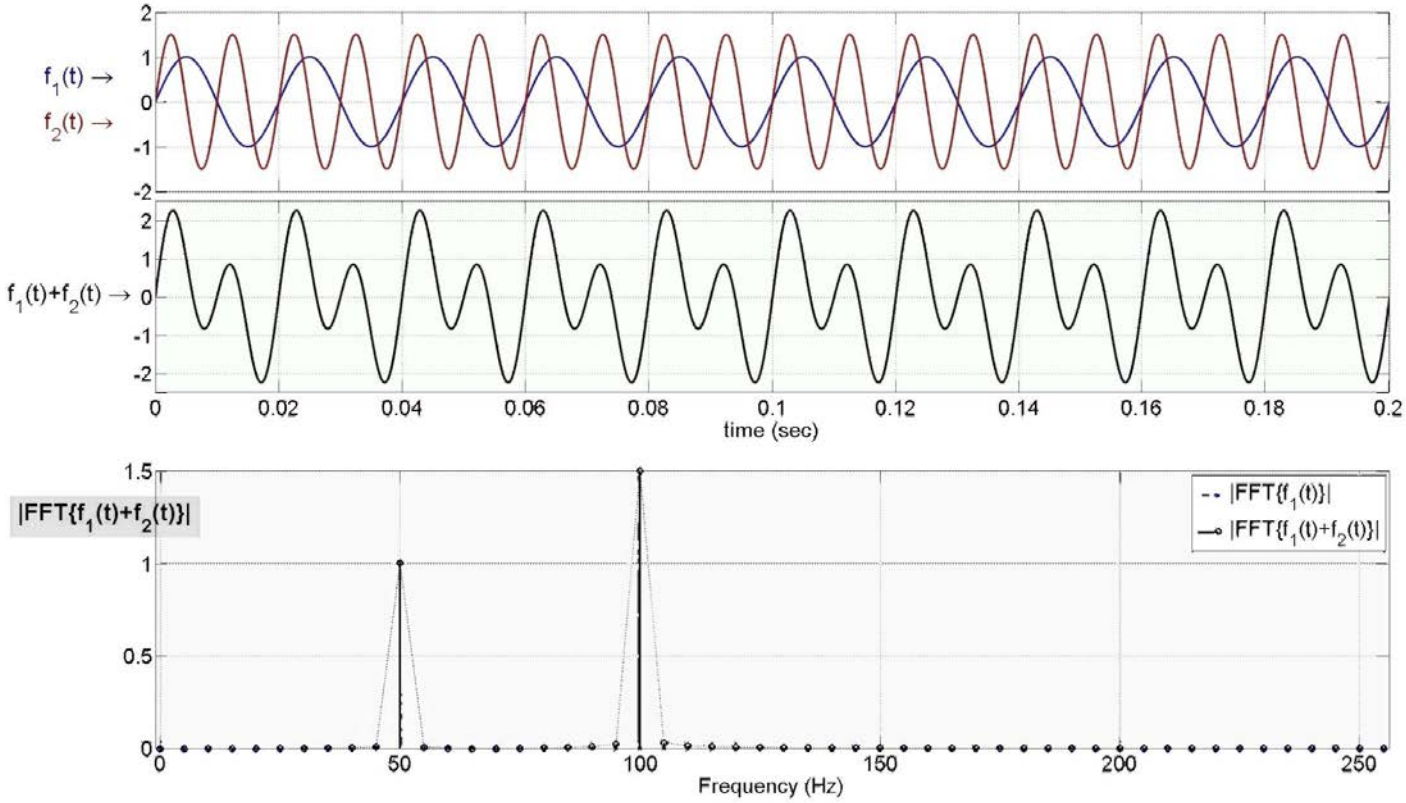

Hydrogen
Helium
Neon
Sodium
Mercury
Intensity
Wavelength

Graphic here identifying type of star from spectrum



Source: Kitts Peak NOAO National Optical Astronomy Observatory
https://www.noao.edu/image_gallery/html/im0649.html


FREQUENCY
TIME

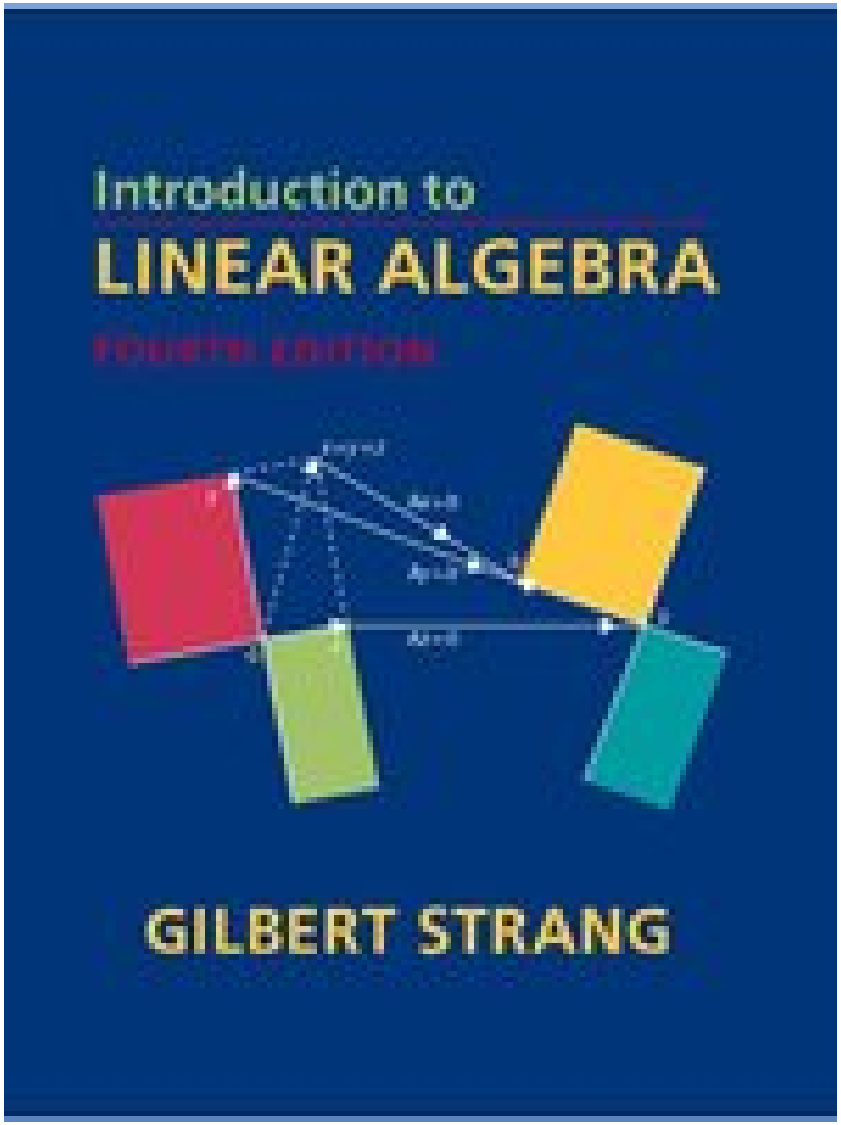


https://www.youtube.com/watch?v=-GYB7khblA0

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

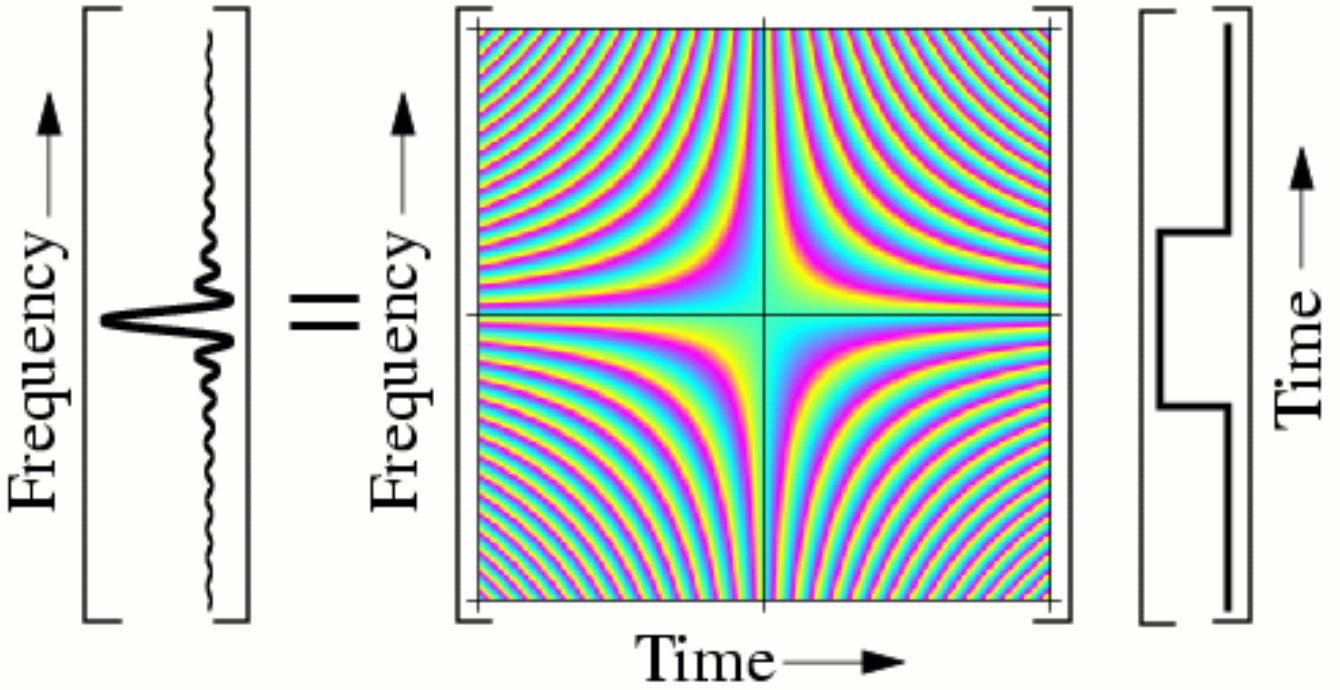
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$$

Remember your Linear Algebra?



The Fourier Transform

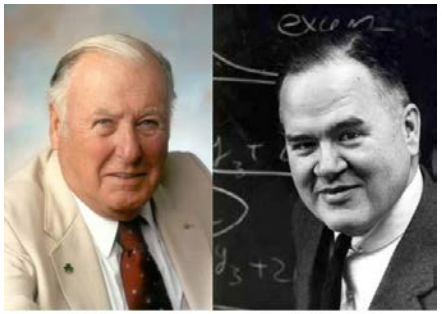
A visualization of the Fredholm kernel of the continuous Fourier transform



Source: WearCam.Org

The Fast Fourier Transform

Carl Friedrich Gauss 1777-1855



Source: https://www.pinterest.com/pin/369435975658121533/

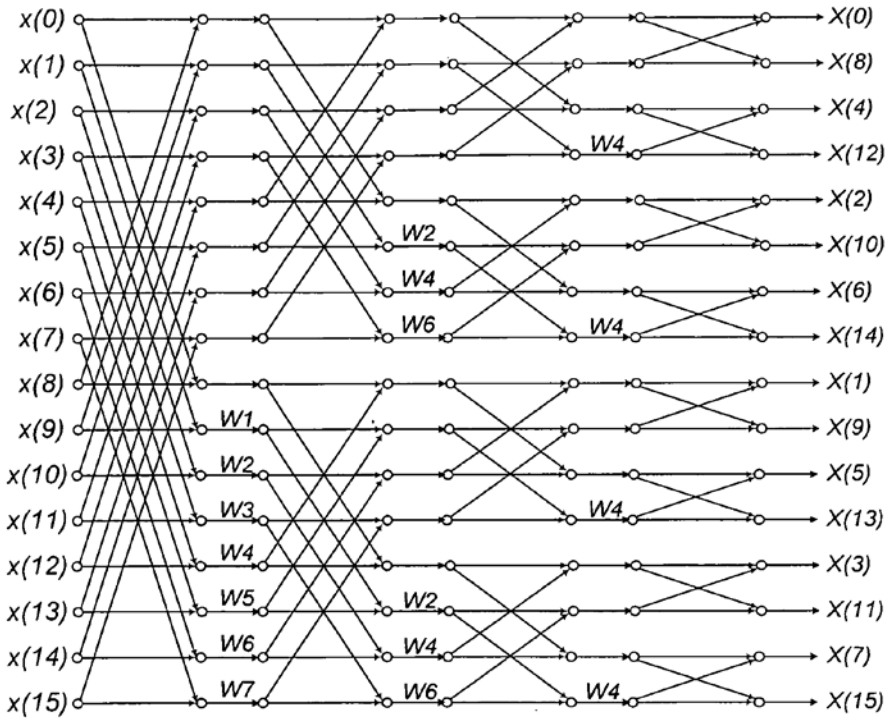


Fig. 4

Source: https://www.google.com/patents/US20120041996

Graph – Matrix Duality

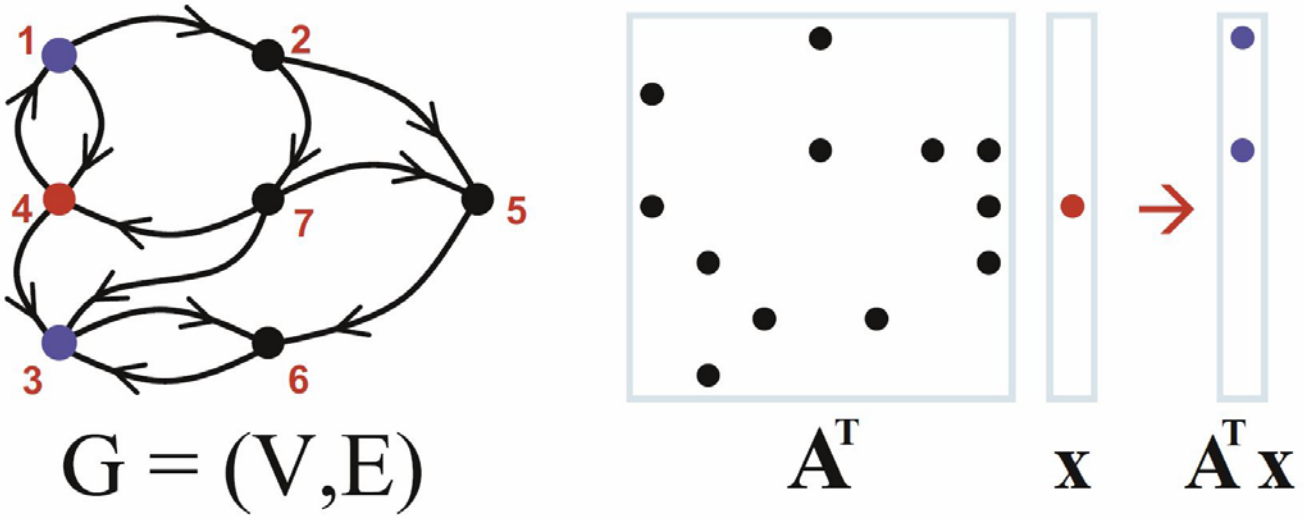
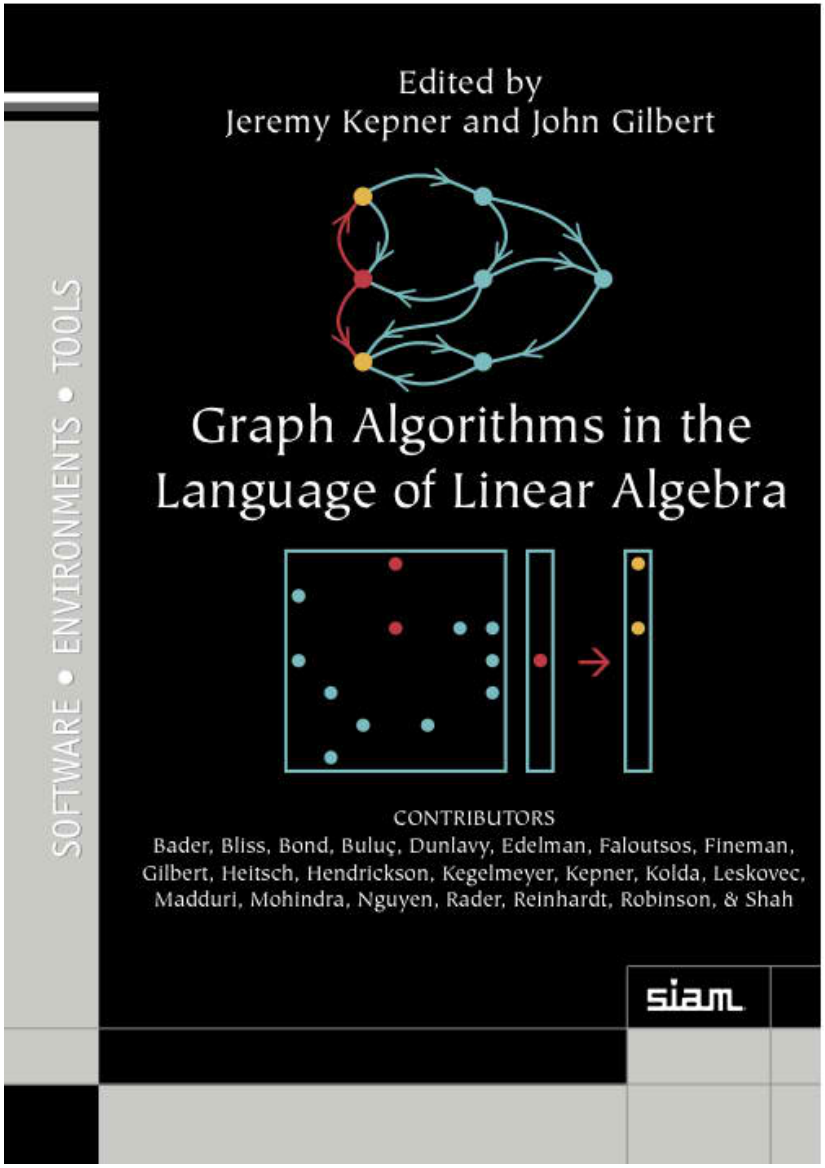


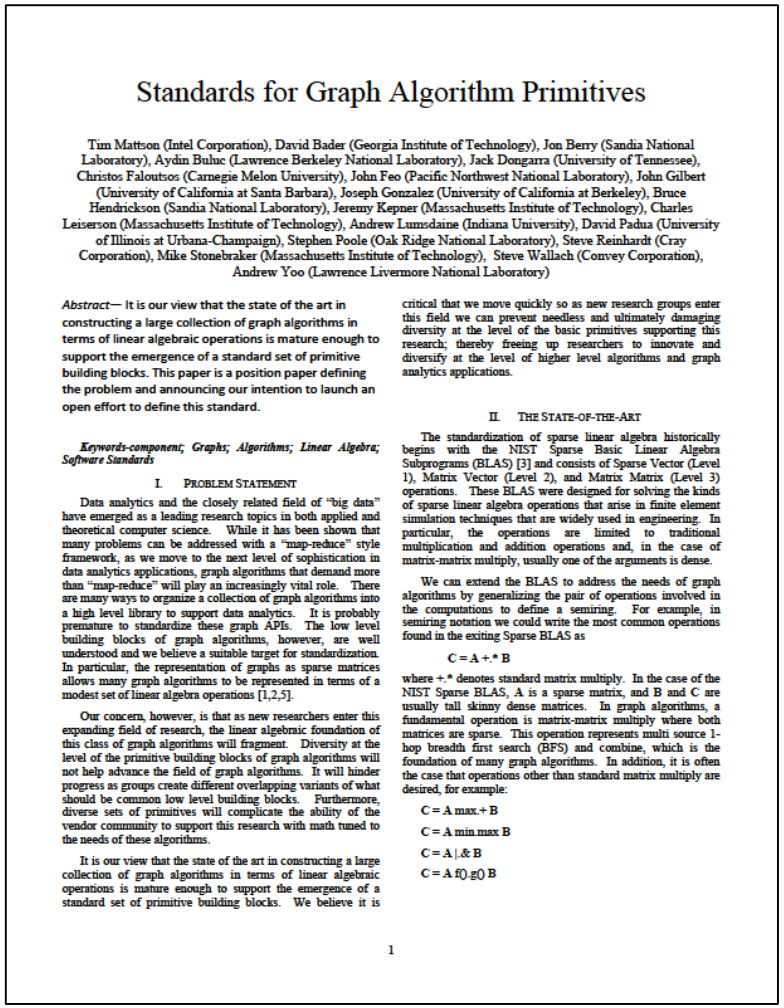
Figure 1.1. Matrix graph duality.
Adjacency matrix **A** is dual with the corresponding graph. In addition, vector matrix multiply is dual with breadth-first search.

Source: Kepner and Gilbert, Graph Algorithms in the Language of Linear Algebra, SIAM Press 2011

Linear Algebra and Graph Analysis



GraphBLAS Manifesto and Specification



Standards for Graph Algorithm Primitives

Tim Mattson (Intel Corporation), David Bader (Georgia Institute of Technology), Jon Berry (Sandia National Laboratory), Aydin Buluc (Lawrence Berkeley National Laboratory), Jack Dongarra (University of Tennessee), Christos Faloutsos (Carnegie Melon University), John Feo (Pacific Northwest National Laboratory), John Gilbert (University of California at Santa Barbara), Joseph Gonzalez (University of California at Berkeley), Bruce Hendrickson (Sandia National Laboratory), Jeremy Kepner (Massachusetts Institute of Technology), Charles Leiserson (Massachusetts Institute of Technology), Andrew Lumsdaine (Indiana University), David Padua (University of Illinois at Urbana-Champaign), Stephen Poole (Oak Ridge National Laboratory), Steve Reinhardt (Cray Corporation), Mike Stonebraker (Massachusetts Institute of Technology), Steve Wallach (Convey Corporation), Andrew Yoo (Lawrence Livermore National Laboratory)

Abstract— It is our view that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. This paper is a position paper defining the problem and announcing our intention to launch an open effort to define this standard.

Keywords-component; Graphs; Algorithms; Linear Algebra; Software Standards

I. Problem Statement

Data analytics and the closely related field of "big data" have emerged as a leading research topics in both applied and theoretical computer science. While it has been shown that many problems can be addressed with a "map-reduce" style framework, as we move to the next level of sophistication in data analytics applications, graph algorithms that demand more than "map-reduce" will play an increasingly vital role. There are many ways to organize a collection of graph algorithms into a high level library to support data analytics. It is probably premature to standardize these graph APIs. The low level building blocks of graph algorithms, however, are well understood and we believe a suitable target for standardization. In particular, the representation of graphs as sparse matrices allows many graph algorithms to be represented in terms of a modest set of linear algebra operations [1,2,5].

Our concern, however, is that as new researchers enter this expanding field of research, the linear algebraic foundation of this class of graph algorithms will fragment. Diversity at the level of the primitive building blocks of graph algorithms will not help advance the field of graph algorithms. It will hinder progress as groups create different overlapping variants of what should be common low level building blocks. Furthermore, diverse sets of primitives will complicate the ability of the vendor community to support this research with math tuned to the needs of these algorithms.

It is our view that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. We believe it is critical that we move quickly so as new research groups enter this field we can prevent needless and ultimately damaging diversity at the level of the basic primitives supporting this research; thereby freeing up researchers to innovate and diversify at the level of higher level algorithms and graph analytics applications.

II. The State-of-the-Art

The standardization of sparse linear algebra historically begins with the NIST Sparse Basic Linear Algebra Subprograms (BLAS) [3] and consists of Sparse Vector (Level 1), Matrix Vector (Level 2), and Matrix Matrix (Level 3) operations. These BLAS were designed for solving the kinds of sparse linear algebra operations that arise in finite element simulation techniques that are widely used in engineering. In particular, the operations are limited to traditional multiplication and addition operations and, in the case of matrix-matrix multiply, usually one of the arguments is dense.

We can extend the BLAS to address the needs of graph algorithms by generalizing the pair of operations involved in the computations to define a semiring. For example, in semiring notation we could write the most common operations found in the exiting Sparse BLAS as

$$C = A +.* B$$

where +.* denotes standard matrix multiply. In the case of the NIST Sparse BLAS, A is a sparse matrix, and B and C are usually tall skinny dense matrices. In graph algorithms, a fundamental operation is matrix-matrix multiply where both matrices are sparse. This operation represents multi source 1-hop breadth first search (BFS) and combine, which is the foundation of many graph algorithms. In addition, it is often the case that operations other than standard matrix multiply are desired, for example:

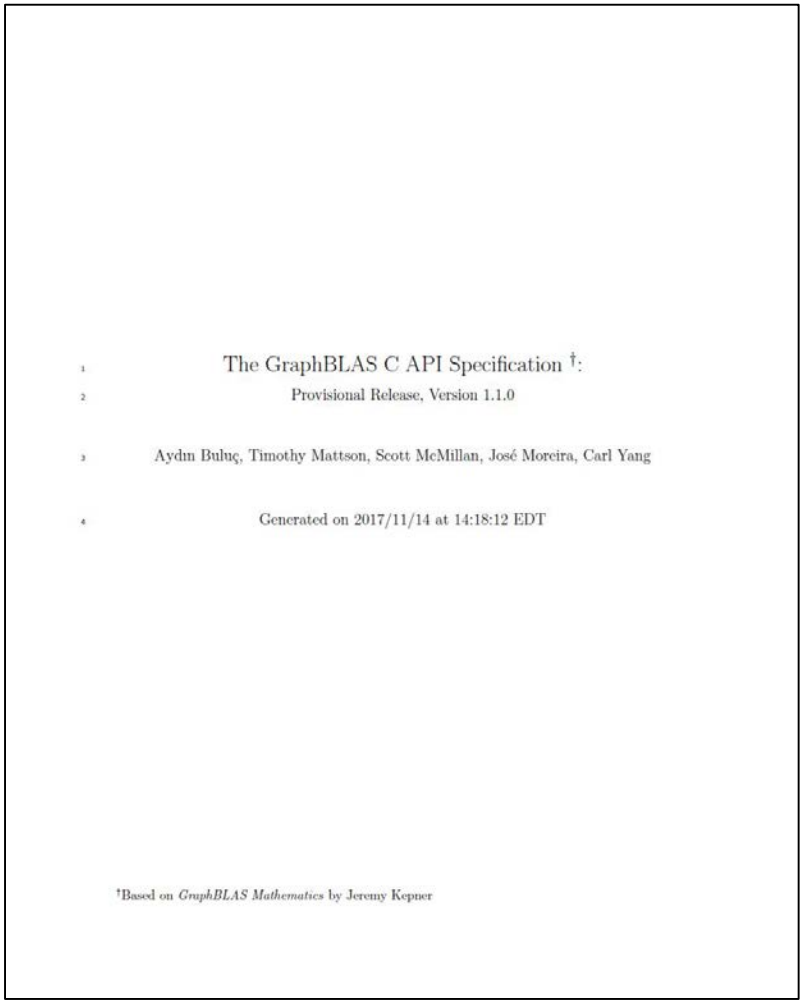
$$C = A \max.+ B$$

$$C = A \min.\max B$$

$$C = A |.\& B$$

$$C = A\ f().g()\ B$$

Mattson, T. et. al.
HPEC 2014



The GraphBLAS C API Specification †:

Provisional Release, Version 1.1.0

Aydın Buluç, Timothy Mattson, Scott McMillan, José Moreira, Carl Yang

Generated on 2017/11/14 at 14:18:12 EDT

†Based on *GraphBLAS Mathematics* by Jeremy Kepner

Bulic, et al. 2017

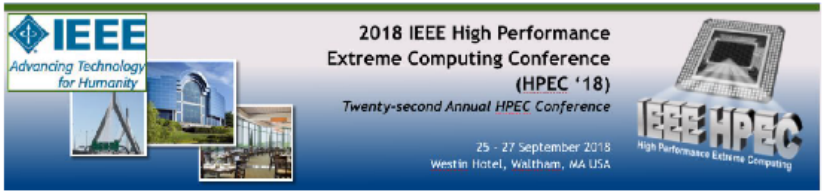
IEEE High Performance Extreme Computing 2018

Waltham MA

September 2018

Papers due May 18

CALL FOR PAPERS



www.ieee-hpec.org

Committees

Chairman & SIAM Liaison
Dr. Jeremy Kepner
Fellow, MIT Lincoln Laboratory

Senior Advisory Board Chair
Mr. Robert Bond
CTO, MIT Lincoln Laboratory

Senior Advisory Board
Prof. Anant Agarwal
MIT CSAIL

Prof. Nadya Bliss
Arizona State University

Dr. Richard Games
Chief Engineer, MITRE Intelligence Center

Mr. John Goodhue
Director, MGHPCC

Dr. Bernadette Johnson
Chief Scientist, DIUX

Dr. Richard Linderman
ASDR&E

Mr. David Martinez
Associate Division Head
MIT Lincoln Laboratory

Dr. John Reynders
CIO Moderna

Dr. Michael Stonebraker
Co-founder SciDB and Vertica;
CTO VoltDB and Paradigm4

Publicity Co-Chairs
Dr. Albert Reuther
MIT Lincoln Laboratory
Mr. Dan Campbell
GTRI

CFP Co-Chairs
Dr. Patrick Dreher
MIT
Dr. Franz Franchetti
CMU

Publications Chair
Prof. Miriam Leeser
Northeastern University

Administrative Contacts
Mr. Robert Alongi
IEEE Boston Section

The IEEE High Performance Extreme Computing Conference (HPEC '18) will be held in the Greater Boston Area, Massachusetts, USA on 25 – 27 September 2018. The HPEC charter is to be the premier conference in the world on the confluence of HPC and Embedded Computing.

The technical committee seeks new presentations that clearly describe advances in high performance extreme computing technologies, emphasizing one or more of the following topics:

- Machine Learning
- Graph Analytics and Network Science
- Advanced Multicore Software Technologies
- Advanced Processor Architectures
- Automated Design Tools
- Big Data and Distributed Computing
- Big Data Meets Big Compute
- Case Studies and Benchmarking of Applications
- Cloud HPEC
- Computing Technologies for Challenging Form Factors
- ASIC and FPGA Advances
- Data Intensive Computing
- Digital Front Ends
- Fault-Tolerant Computing • Embedded Cloud Computing
- General Purpose GPU Computing
- High Performance Data Analysis
- Interactive and Real-Time Supercomputing
- Mapping and Scheduling of Parallel and Real-Time Applications
- New Application Frontiers
- Open System Architectures
- Secure Computing & Anti-Tamper Technologies

HPEC accepts two types of submissions:

1. Full papers (up to 6 pages, references not included), and
2. Extended abstracts (up to 2 pages, references included).

All submissions must be submitted through the Conference Management Toolkit (CMT) at:
https://cmt3.research.microsoft.com/HPEC2018/

IMPORTANT DATES:

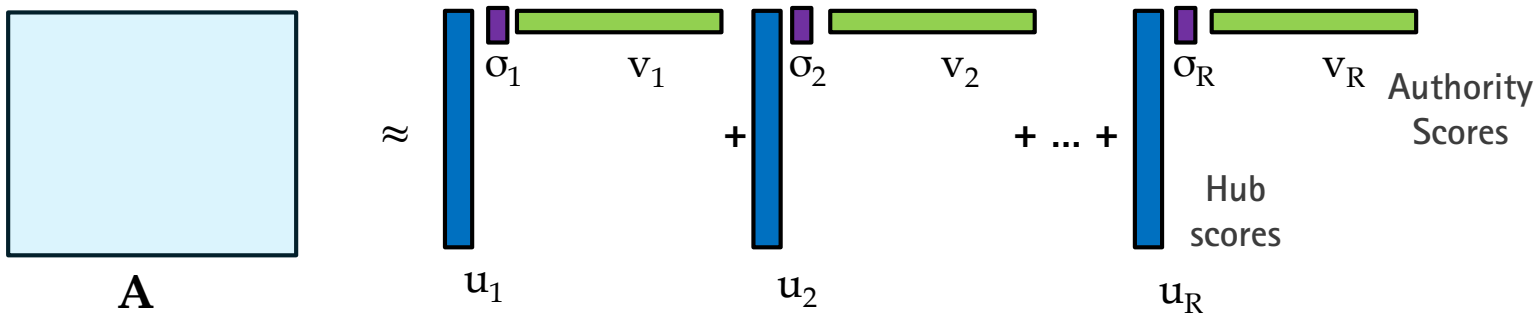
Submission Deadline: **May 18, 2018**
Notification of Acceptance: **July 1, 2018**
Camera Ready Deadline: **August 1, 2018**

Preference will be given to papers with strong, quantitative results, demonstrating novel approaches or describing high quality prototypes. Authors of full papers can mark their preference for a poster display or an oral presentation. Presenters who wish to have hardware demonstrations are encouraged to mark their preference for a poster display. Accepted extended abstracts will be displayed as posters. All paper and extended abstract submissions need to use the approved IEEE templates. Full paper submissions with the highest peer review ratings will be published by IEEE in the official HPEC proceedings available on IEEE eXplore. All other accepted submissions and extended abstracts are published on ieee-hpec.org. Vendors are encouraged to sign up for vendor booths. This will allow vendors to present their HPEC technologies in an interactive atmosphere suitable for product demonstration and promotion. Papers can be declared "student paper" if the first author was a student when doing the presented work, and will be eligible for the "IEEE HPEC best student paper award." Papers should not be anonymized. We welcome input (hpec@ieee-hpec.org) on tutorials, invited talks, special sessions, peer reviewed presentations, and vendor demos. Instructions for submitting will be posted on the conference web site shortly. Full paper submissions should use the approved IEEE templates. The highest scoring submissions will be published by IEEE in the official HPEC proceedings available on IEEE eXplore. All other accepted submissions are published on ieee-hpec.org.

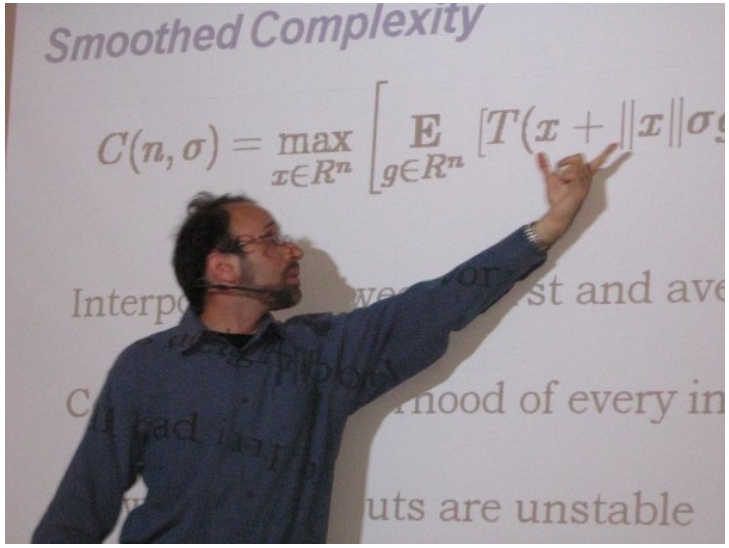
Linear Algebra Analysis or 2D Example
Singular Value Decomposition (SVD)

$$\mathbf{A} = U\Sigma V^T$$

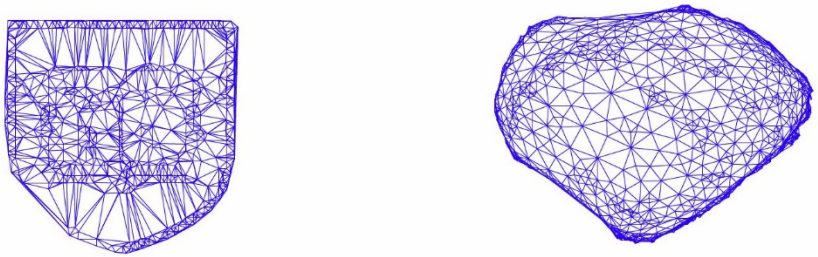
$$\mathbf{A} \approx \sum_{r=1}^{R} \sigma_r u_r \circ v_r$$



HITS "Hubs and Authorities" Algorithm (Kleinberg 1999)



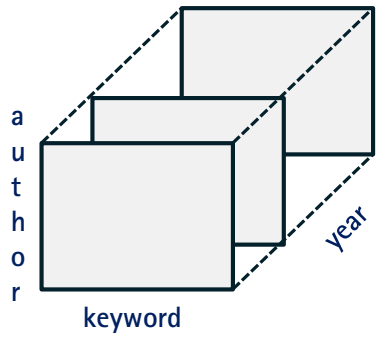
Spectral and Electrical Graph Theory



Daniel A. Spielman
Dept. of Computer Science
Program in Applied Mathematics
Yale Unviersity

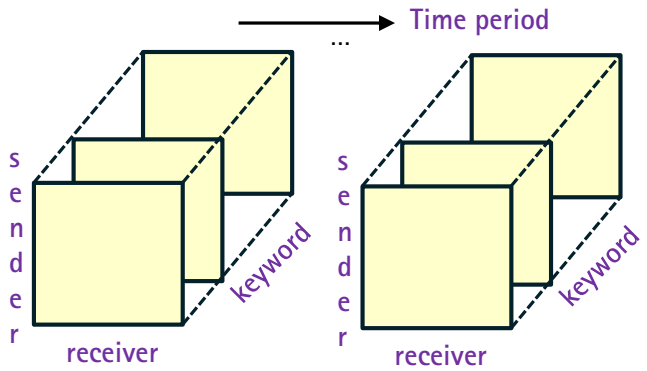
Real-world Data – Representation

Real-world data are multi-dimensional with multiple aspects



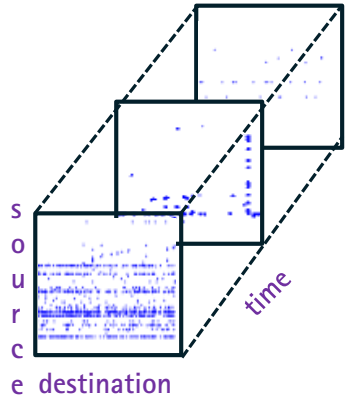
Author x Keyword x Year

Bibiliometric Data



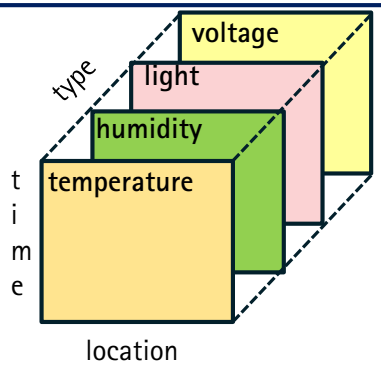
Sender x Receiver x Keyword x Time period

Email Data



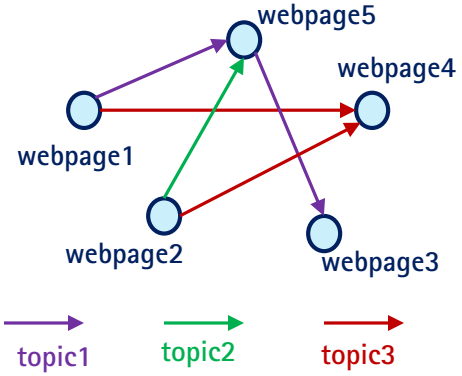
Src x Dest x Time

Network Traffic Data



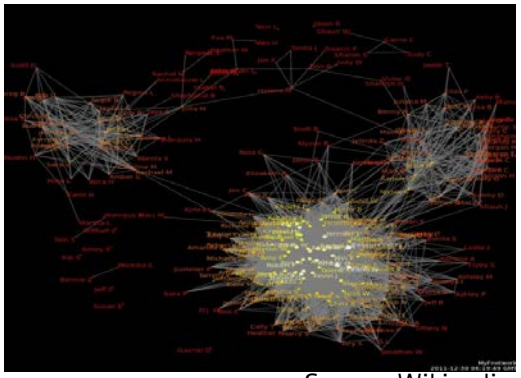
Time x Location x Type

Environmental Sensor Monitoring



Page x Page x Topic

Web graph

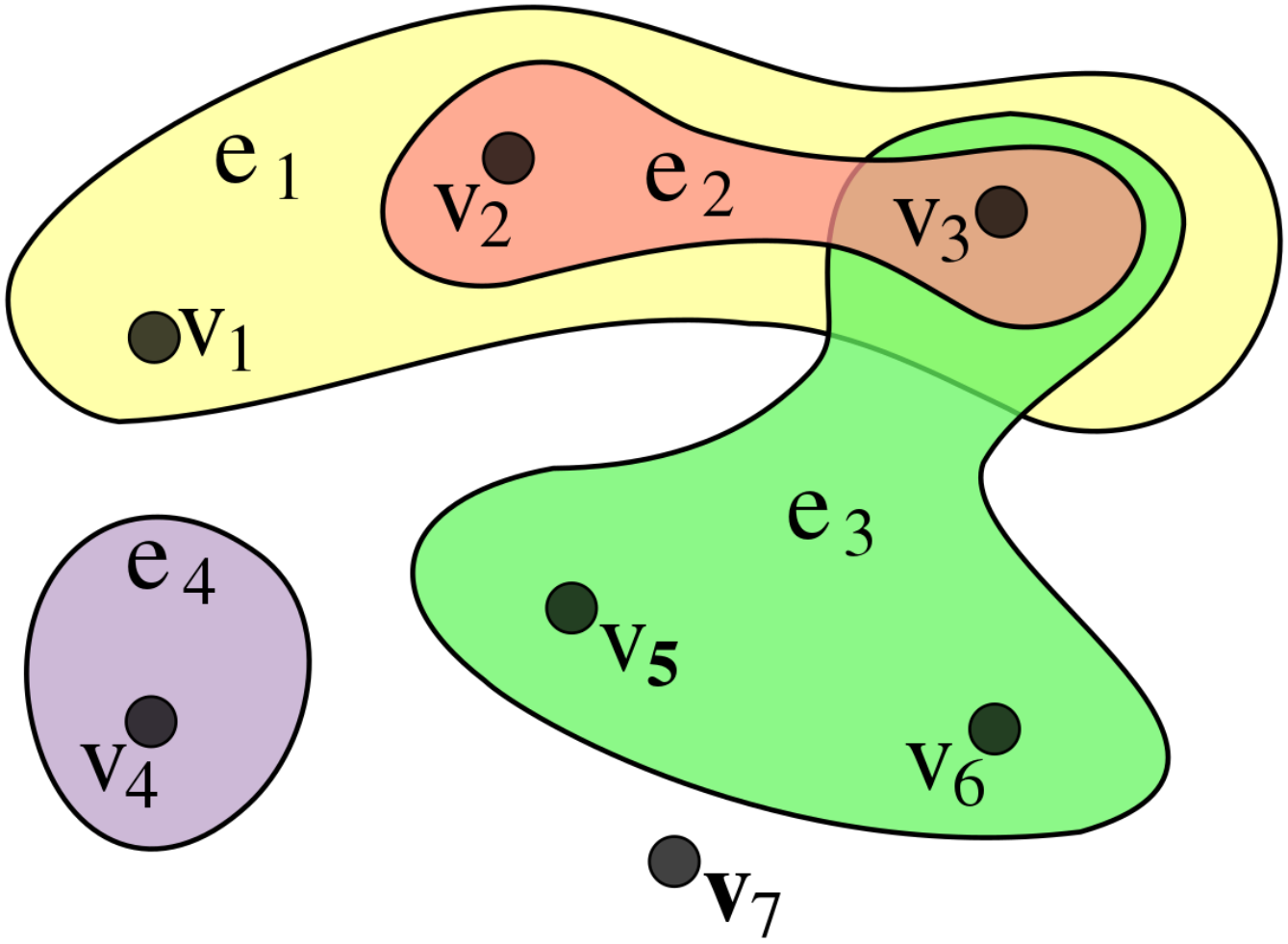


Source: Wikipedia

Person x Person x Relation

Social network graph

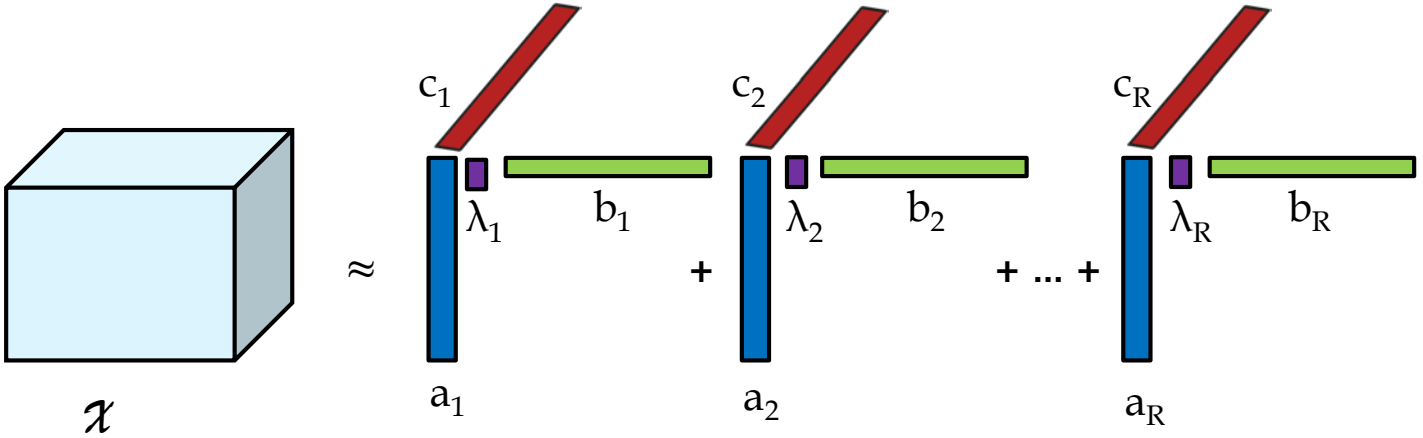
Hypergraph



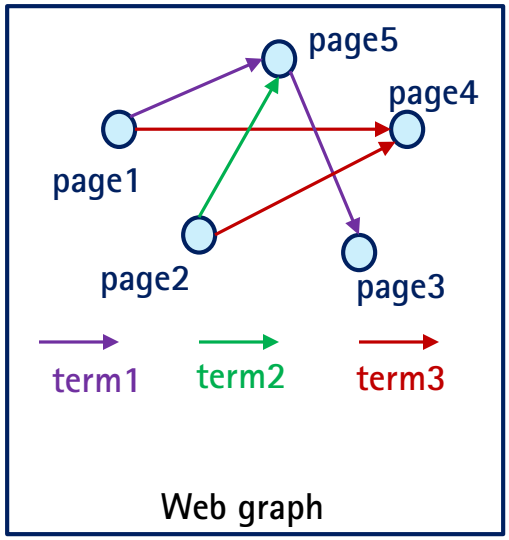
Source: https://en.wikipedia.org/wiki/Hypergraph

Multi-Linear Algebraic Analysis
3D "CP" decomposition

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \, a_r \circ b_r \circ c_r$$

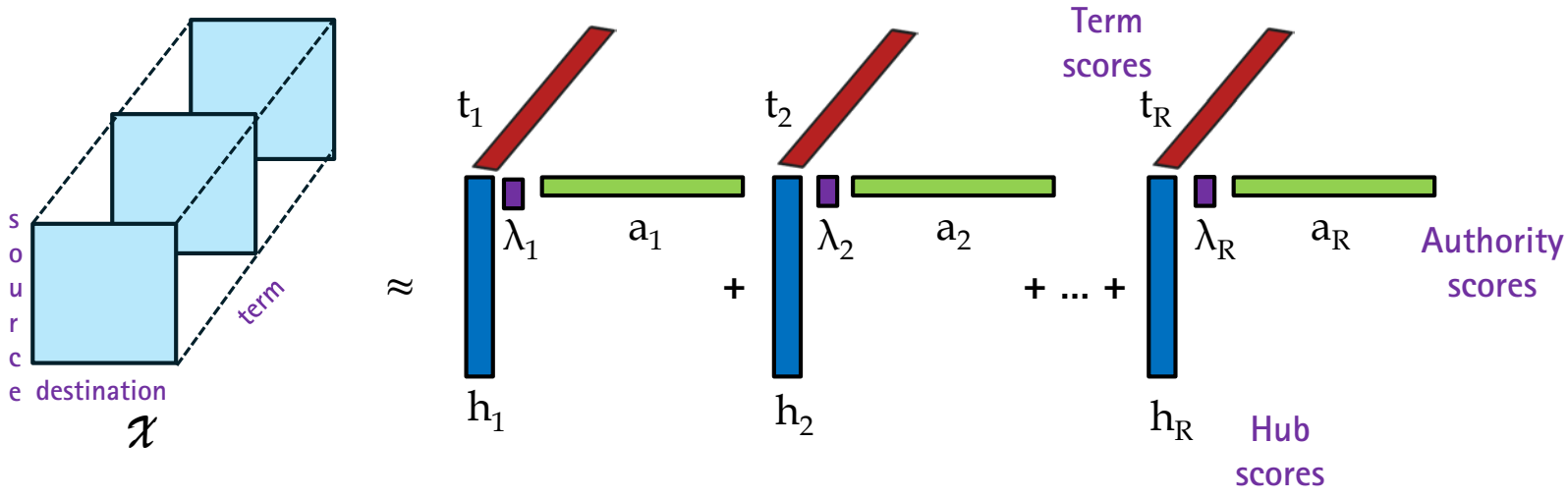


Tensor analogue of HITS



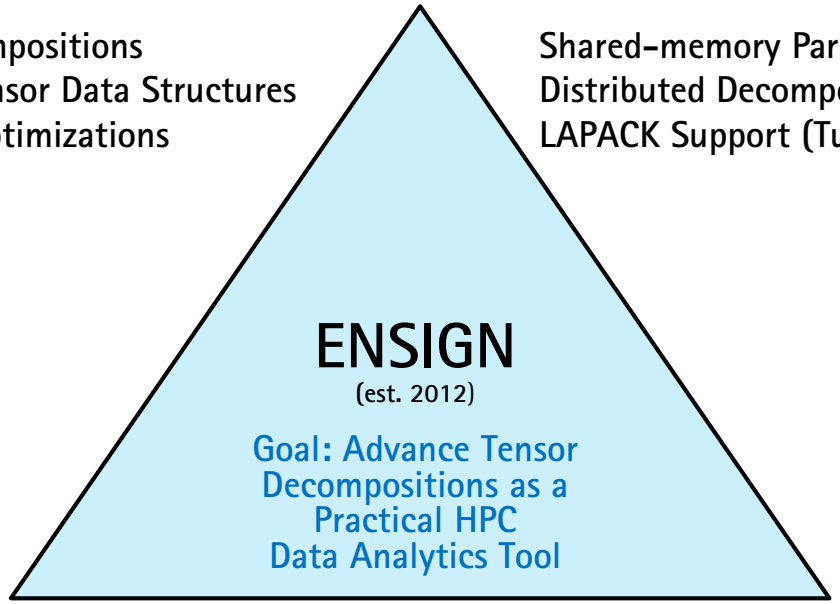
- 3D view of the web graph with topical information added
  - **Very sparse** 3D adjacency "tensor"
- TOPHITS [Kolda et al., 2005]
- Tensor decomposition of the adjacency tensor

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \, h_r \circ a_r \circ t_r$$



Reservoir Labs ENSIGN Project

Version:
ENSIGN 4.1 – Dec 2017



ANSI C for All Decompositions
Optimized Sparse Tensor Data Structures
Memory-efficient Optimizations

Shared-memory Parallelism (OpenMP)
Distributed Decompositions (CP-ALS, CP-APR)
LAPACK Support (Tucker)

CP-ALS
CP-ALS-NN
CP-APR
CP-APR-PDNR
CP-APR-PQNR

CP-OPT (Joint)
Tucker-ALS
CP-APR-Streaming
Tucker-ALS-LowRankUpdate

GUI & CLI
Tensor Building Tools
Visual Exploration Tools
Python Bindings
Modular Installation
Documentation
Tested for CentOS 7.2
Virtual Machine Distributions

Application Support:
Cyber Security
Geospatial Analysis
Genomics

Python Bindings & Jupyter Notebook

jupyter ENSIGN-Jupyter Last Checkpoint: 4 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Python [conda root]

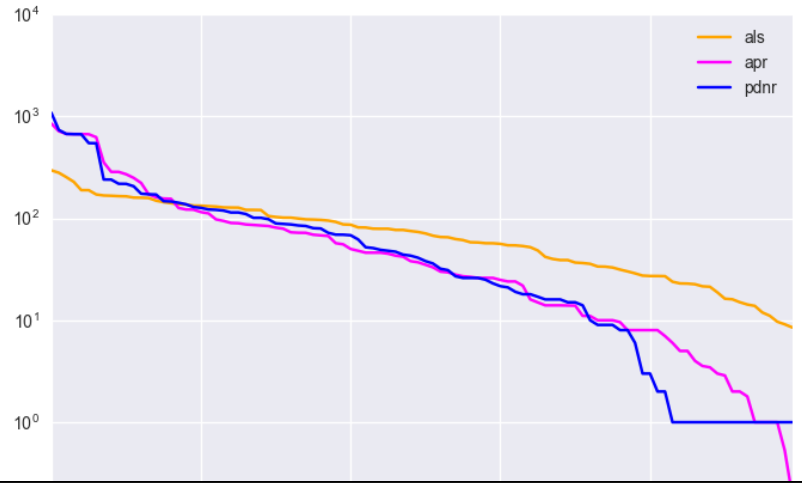
Code | CellToolbar

In [2]:
```
import ensign.cp_decomp as cpd
import ensign.sptensor as spt

# Parameters
rank = '100'
sptensor_file = 'tensor_data.txt'

# Load tensor & decompose
the_tensor = spt.read_sptensor_file(sptensor_file)
als_decomp = cpd.cp_als(the_tensor, rank, "als_save_dir")
apr_decomp = cpd.cp_apr(the_tensor, rank, "apr_save_dir")
pdnr_decomp = cpd.cp_apr_pdnr(the_tensor, rank, "pdnr_save_dir")
```

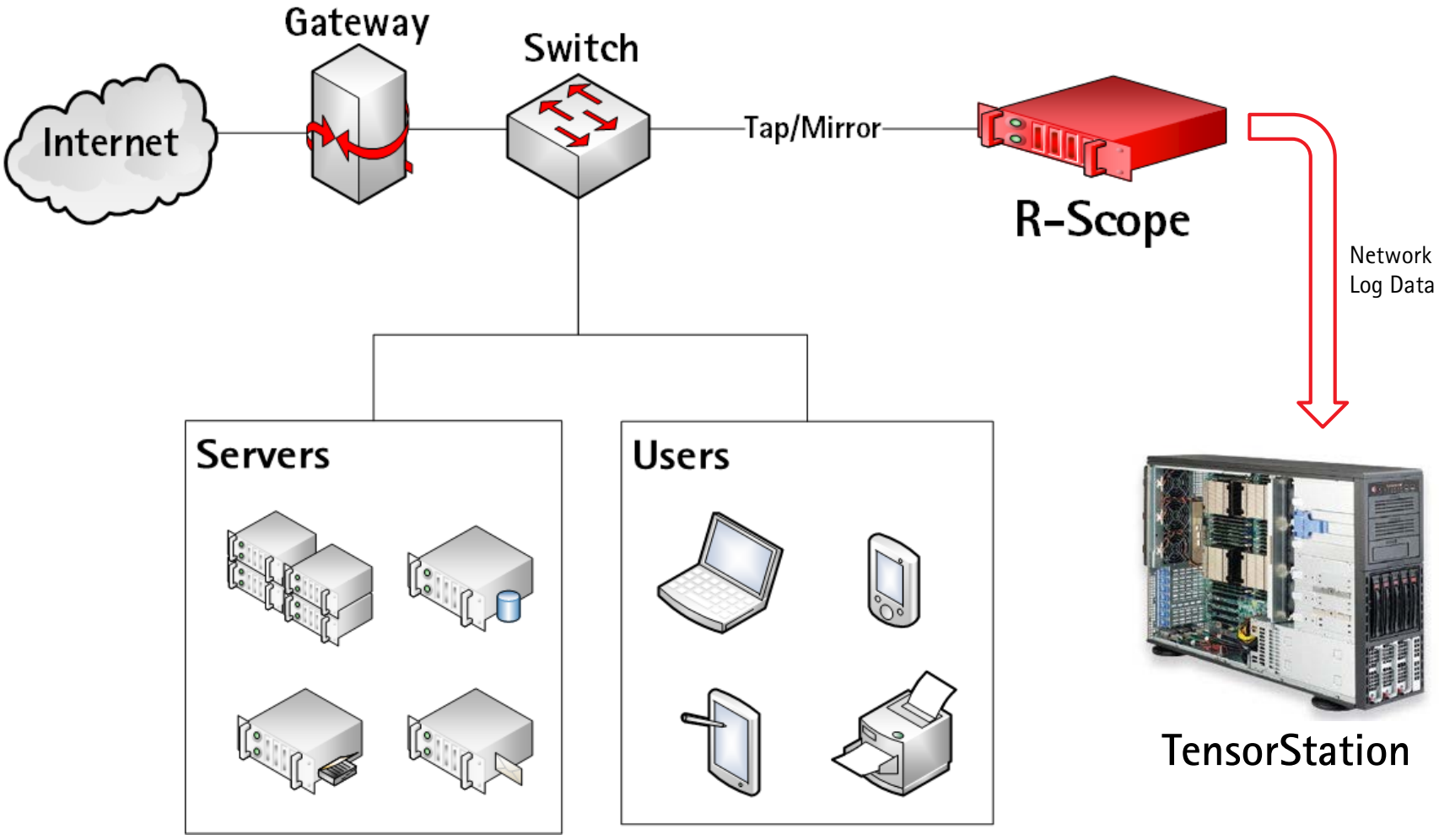
In [3]:
```
als_weights = pd.Series(als_decomp.weights)
apr_weights = pd.Series(apr_decomp.weights)
pdnr_weights = pd.Series(pdnr_decomp.weights)
```

In [4]:
```
ax = als_weights.plot(color='orange', logy=True, label='als', legend=True)
ax = apr_weights.plot(color='magenta', logy=True, label='apr', ax=ax, legend=True)
pdnr_weights.plot(color='blue', logy=True, label='pdnr', ax=ax, legend=True)
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f29bd28fe90>



Network Traffic Data analysis using ENSIGN

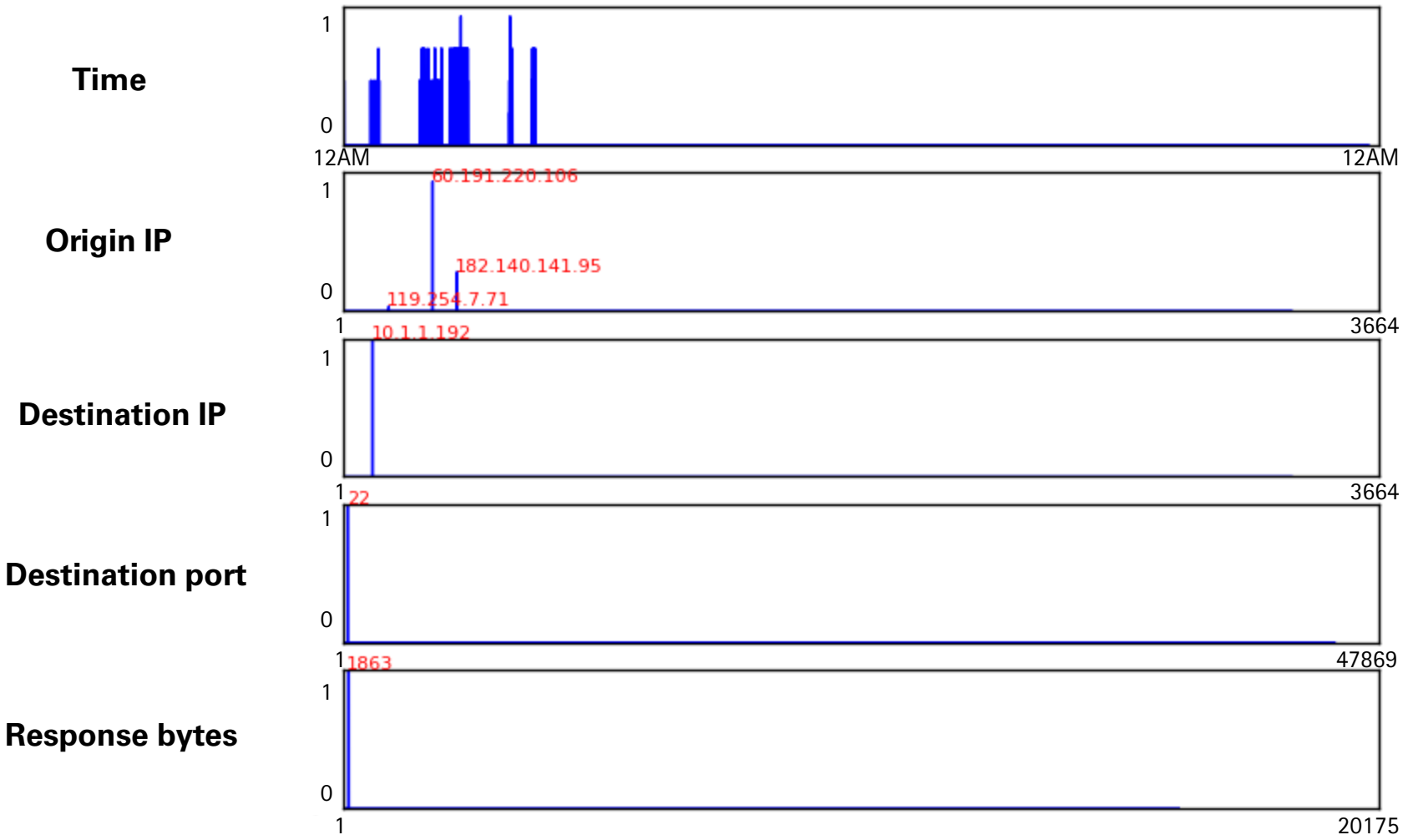


Reservoir Network Traffic Data

1367954284.775933	10.0.2.15	41173	72.22.185.216	80	tcp	0	0	-
1367954284.776223	10.0.2.15	35904	23.6.146.127	80	tcp	0	0	-
1367954284.902059	10.0.2.15	57323	63.251.85.24	80	tcp	2294	797	wt.o.nytimes.com
367954285.233317	10.0.2.15	51934	54.243.131.12	80	tcp	578	184	
3.0950611367954281.046495	10.0.2.15	51237	10.1.1.1	53	udp	32	200	-
1367954281.980046	10.0.2.15	61472	10.1.1.1	53	udp	33	128	-
1367954287.375995	10.0.2.15	55776	74.125.226.214	443	tcp	489	2658	-
1367954282.655601	10.0.2.15	52392	72.22.185.207	80	tcp	0	0	-
1367954282.651031	10.0.2.15	45201	72.22.185.198	80	tcp	0	0	-
1367954282.654099	10.0.2.15	52391	72.22.185.207	80	tcp	0	0	-
1367954282.660982	10.0.2.15	34639	72.22.185.214	80	tcp	0	0	-
1367954282.689652	10.0.2.15	58404	72.22.185.199	80	tcp	0	0	-
1367954283.208290	10.0.2.15	34980	72.21.91.19	80	tcp	394	313	p.typekit.net

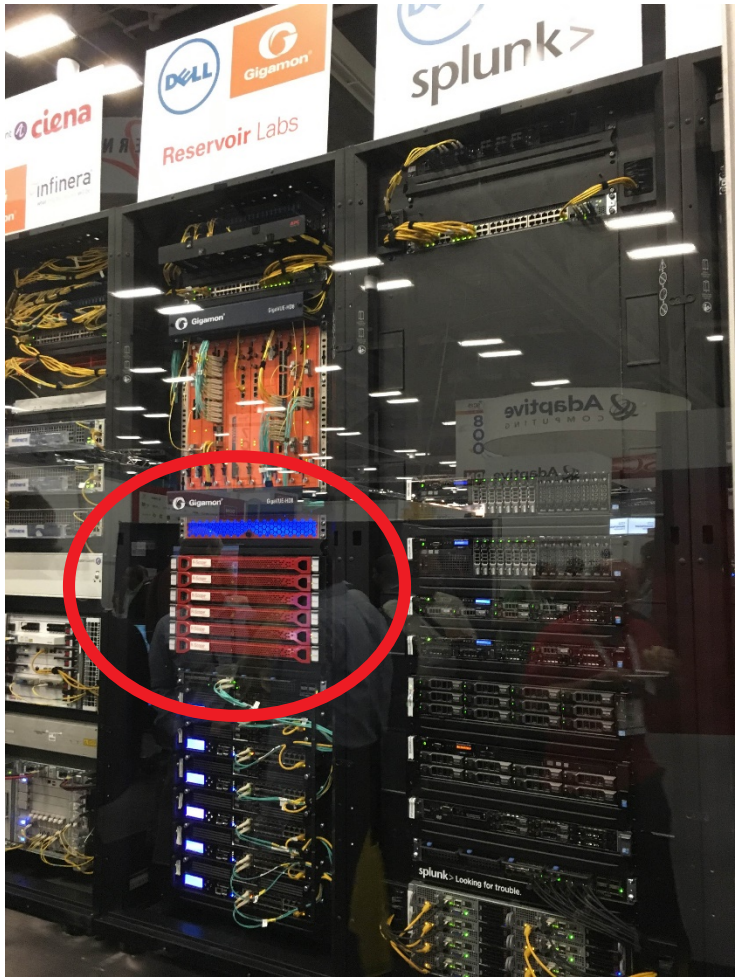
Cyber Data Analysis

SSH Attacks (Component 71 of 120)

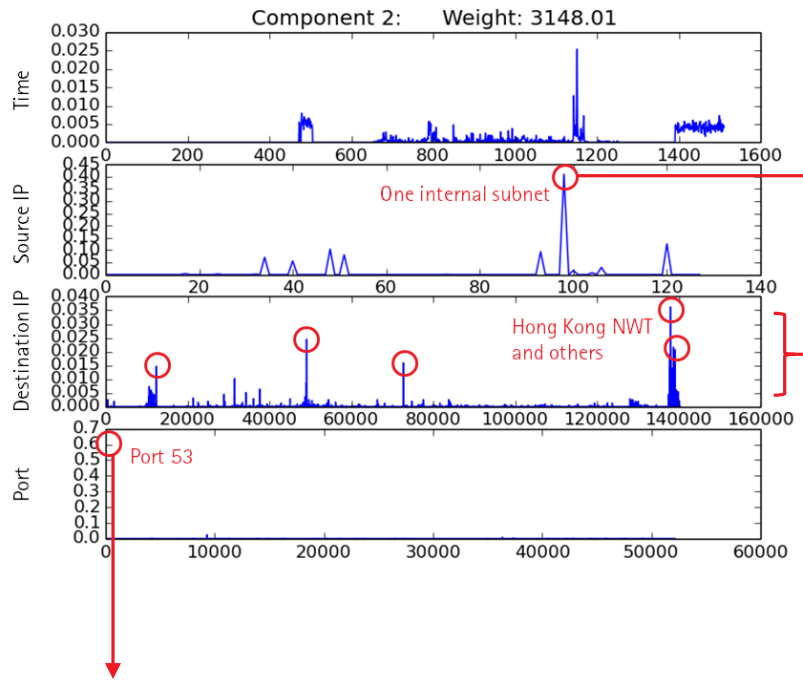


Supercomputing Conference Network: SCInet

~$1M equipment/rack, ~10 racks, >3 Tbps aggregate traffic,
>20000 wireless clients, booth clients, elephant flows, …



Investigation: Suspected Data Exfiltration

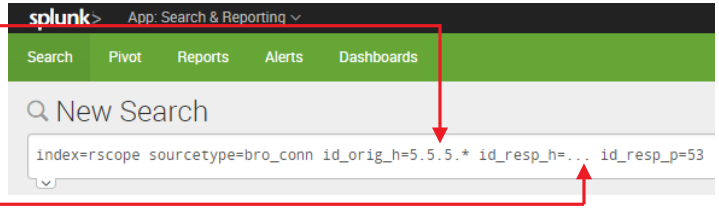


Port 53: Used for DNS

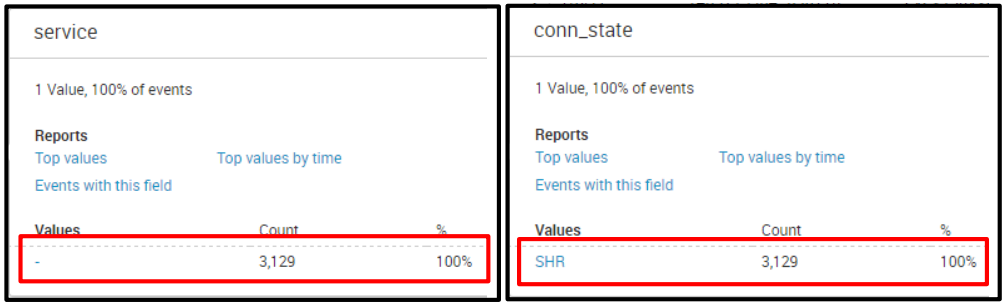
- None of the 4-6 high scoring Destination IPs are the SCinet DNS server
- All destinations are external IPs

Investigating

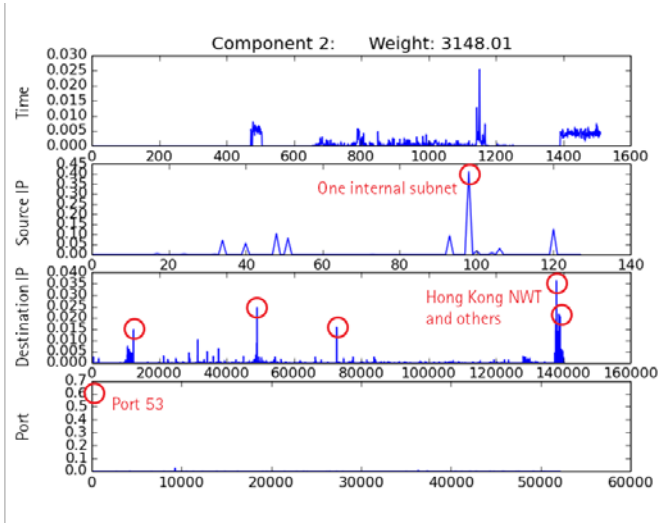
- Search for all connections from this subnet to these destinations with Port 53

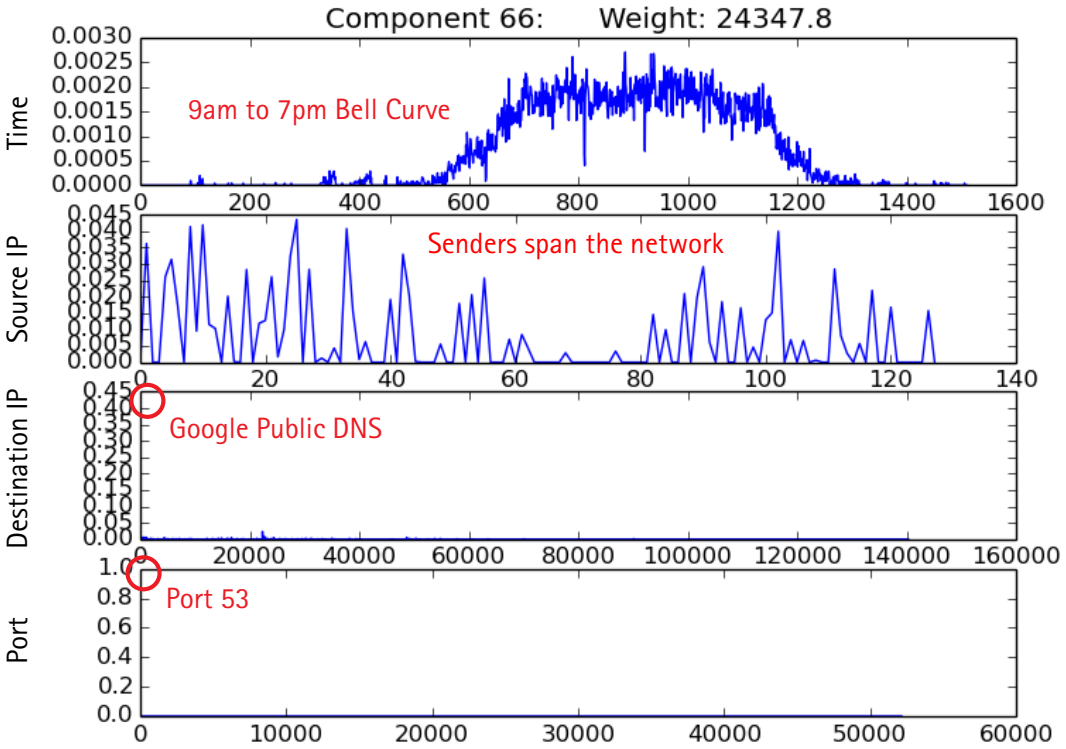


- There is only one IP in this subnet going to those destinations (this is a single internal actor)
- R-Scope labels the service as "-" (not recognized). This is not DNS traffic.
- Connection state is SHR: responder recognized the messages, half-opened communication, and then terminated the connection



Investigation: Suspected Data Exfiltration





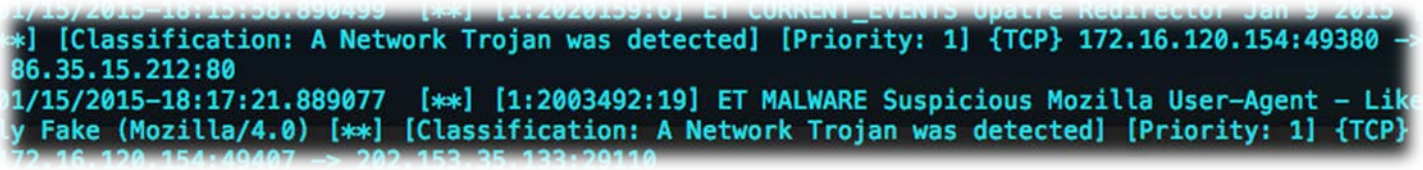
Compare to Typical DNS Traffic

- The component on the right shows an example of typical DNS traffic
- The only Destination IPs are Primary (8.8.8.8) and Secondary (8.8.4.4) DNS
- The time dimension is a highly regular curve between 9 am and 7 pm – the running hours of the conference

Using ENSIGN …

We have uncovered and visualized patterns indicative of:

- Distributed port scans evolving to machine takeover
- Distributed denial of service attacks
- DNS-based data exfiltration/insider threat
- SSH password guessing (apart from scanning)
- Network policy violations
- Exploitation of application-specific port vulnerabilities
- Patterns of traffic indicative of scans for printers or IoT devices
- Broken or misconfigured network services
- Selective, persistent use of cryptographic methods in point-to-point communication



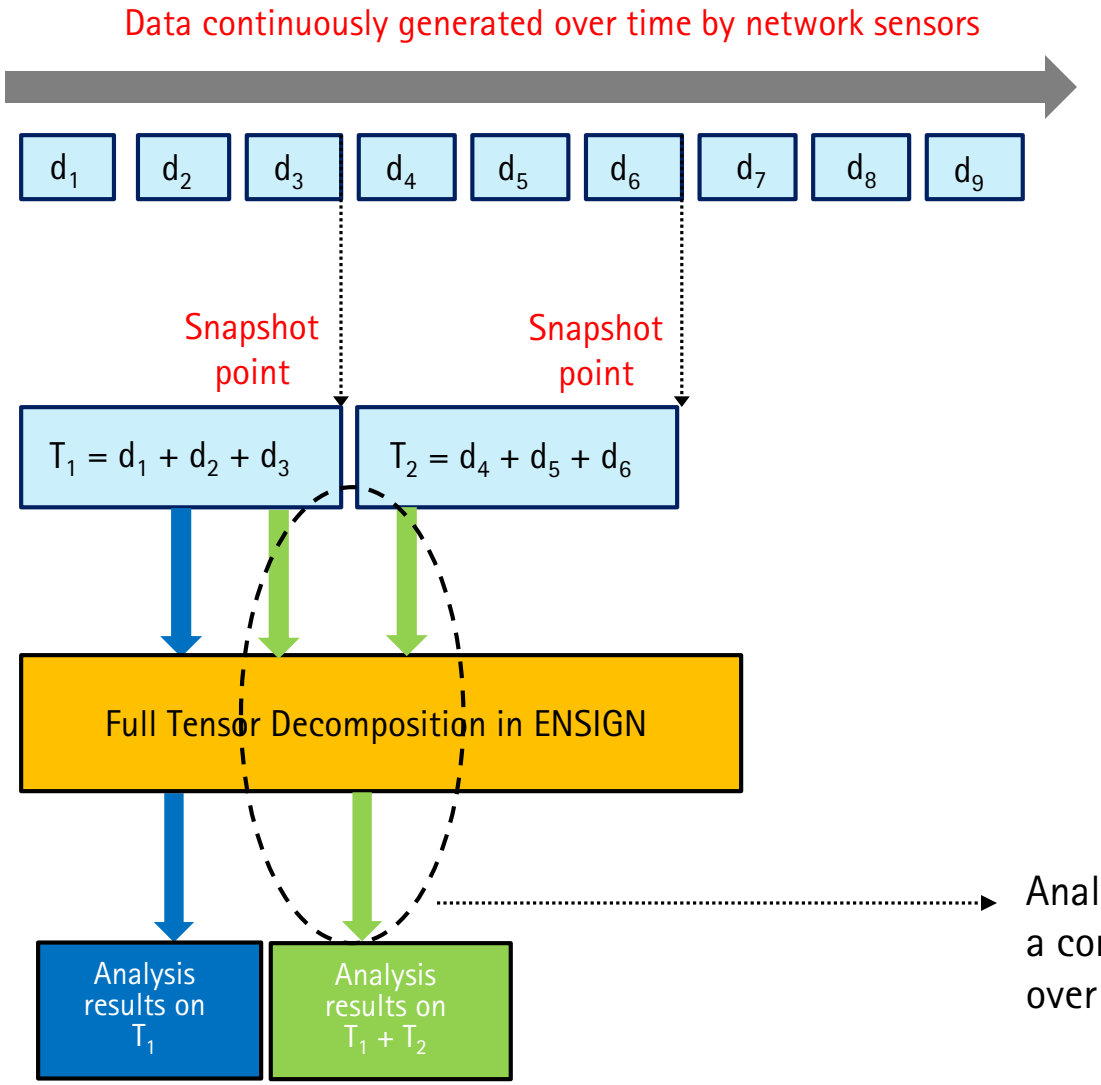
What are the barriers to automated analysis?

1. Provide a faster response to emerging events

2. Automate the classification of components

3. Demonstrate a repeatable process

The Barrier – Respond to Emerging Events

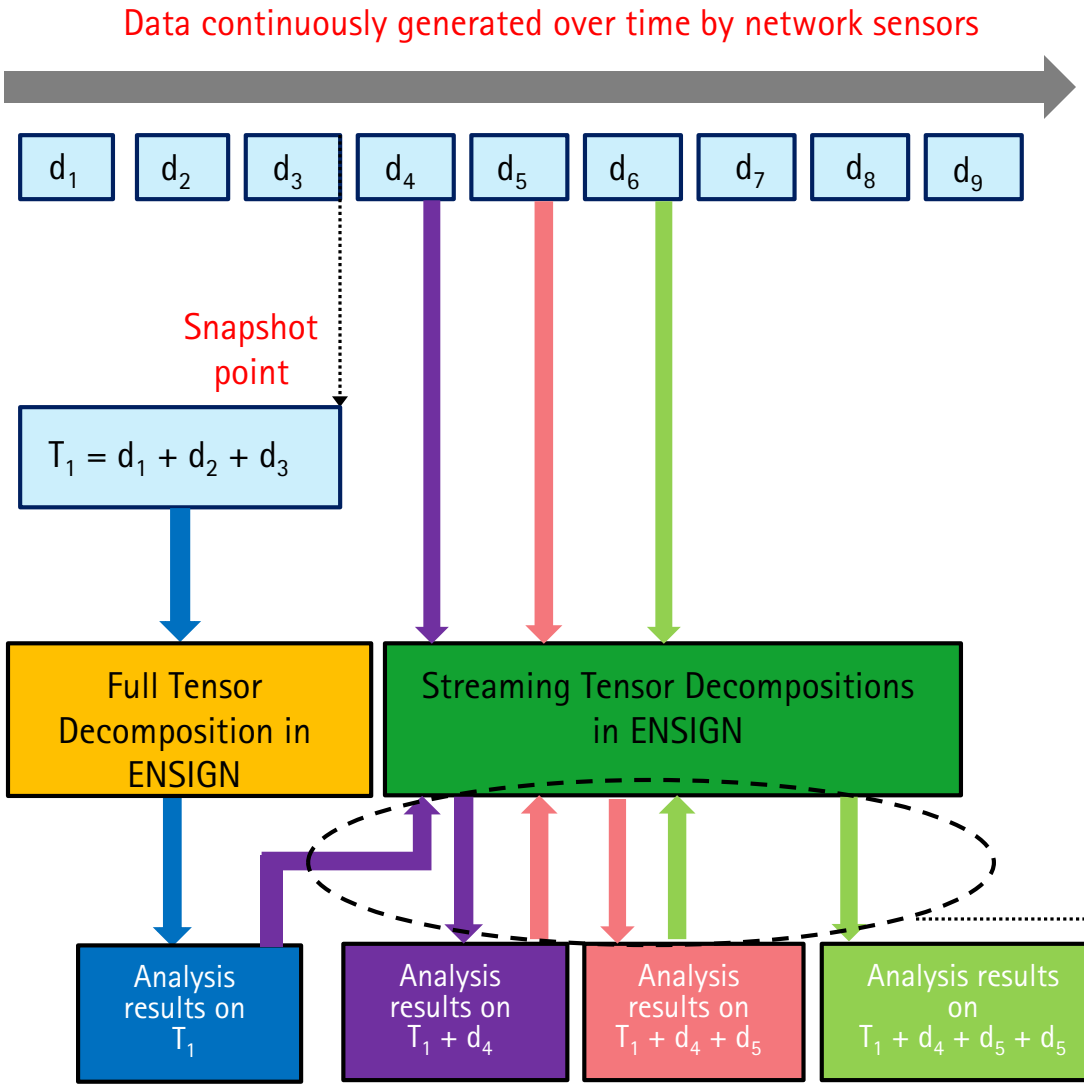


Full Tensor Decomposition

- Complete contextual insights
- Computationally intensive
- **Not ideal for real-time updates**

Analysis of all data is needed to give a complete picture of the activities over the entire timeframe

Streaming Tensor Decompositions

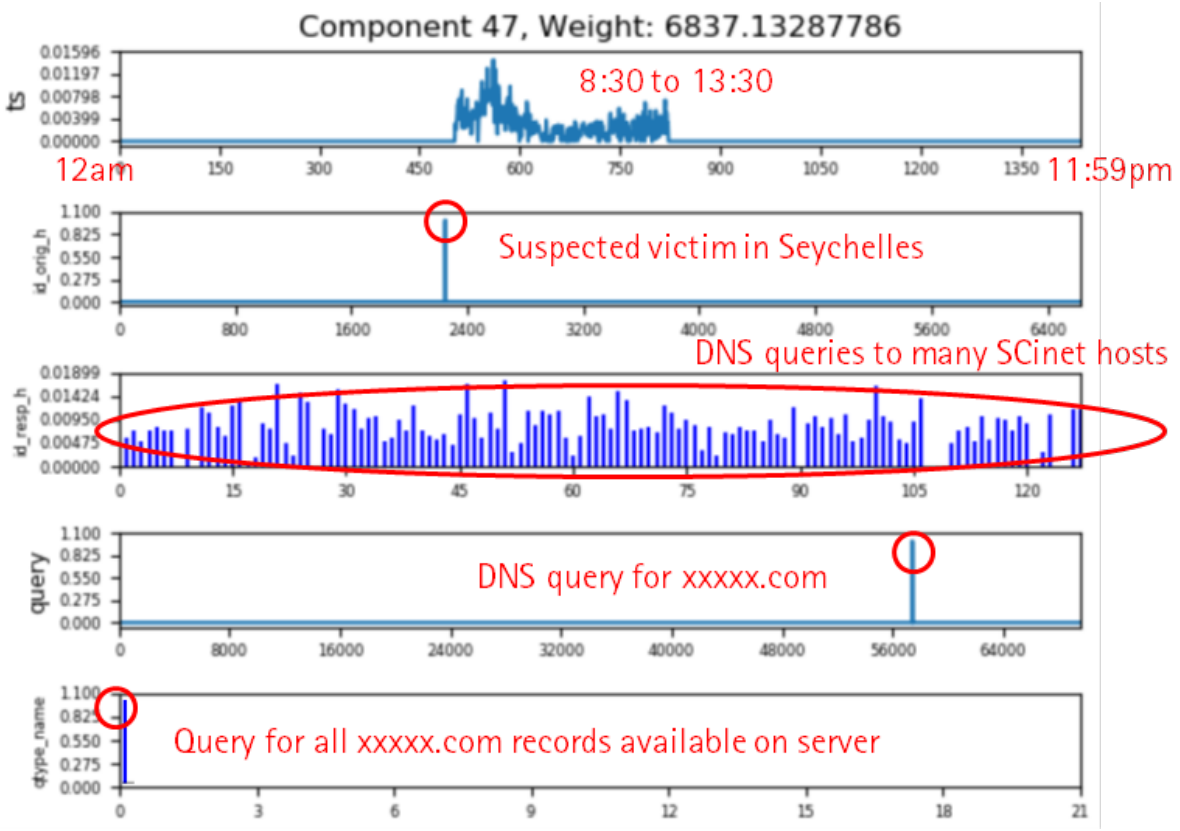


Streaming Tensor Decompositions

- Gives contextual insights with reasonable precision
- Low computational complexity
- **Nearer to real-time updates**

Previous decomposition and only new data streams are processed and analyzed to give a complete picture

DNS Amplification DDoS Attack



DNS amplification DDoS attack identified from full tensor decomposition:

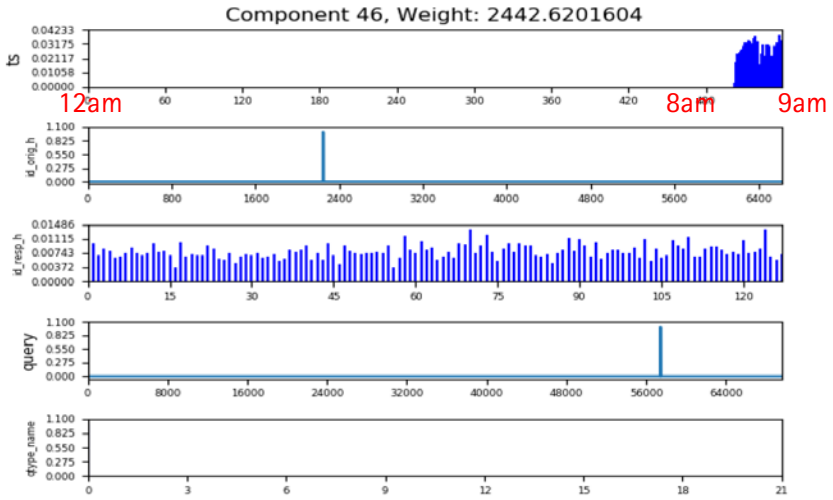
- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
- Tensor created from the log:
  - 5 million non-zeros
- Decomposition took ~500 seconds

Time x Source IP x Destination IP x DNS Query String x DNS Query Type

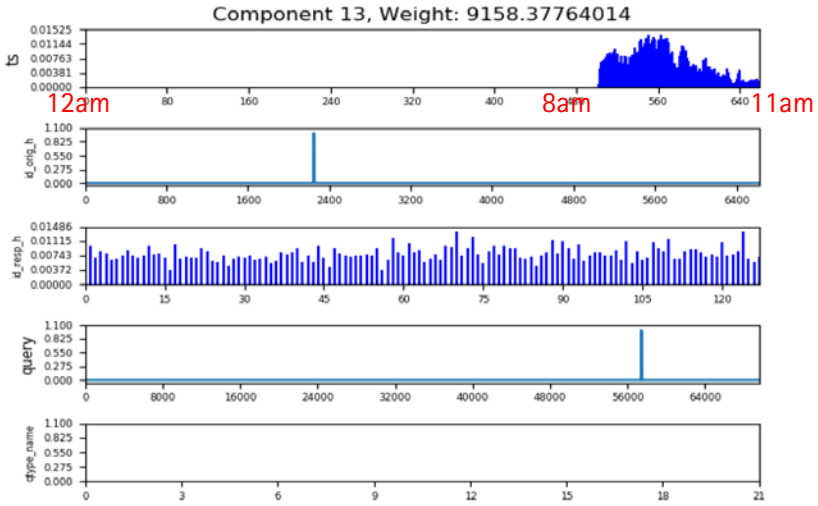
Evolution of the Attack Over Time

… as seen with streaming decompositions

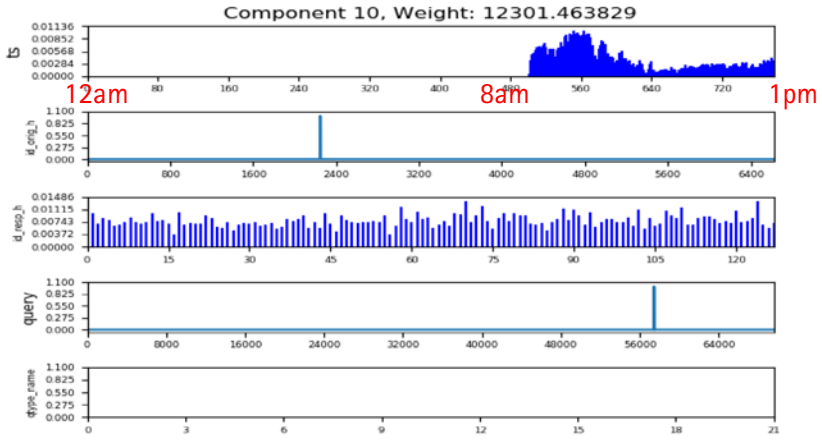
State of the activity at 9am



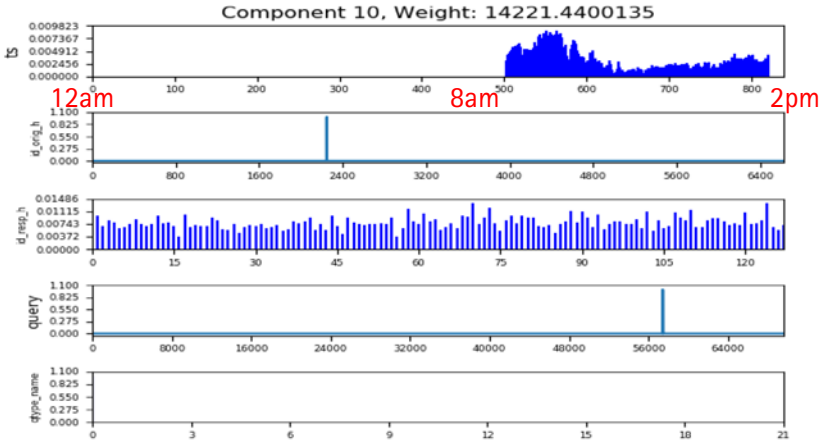
State of the activity at 11am



State of the activity at 1pm

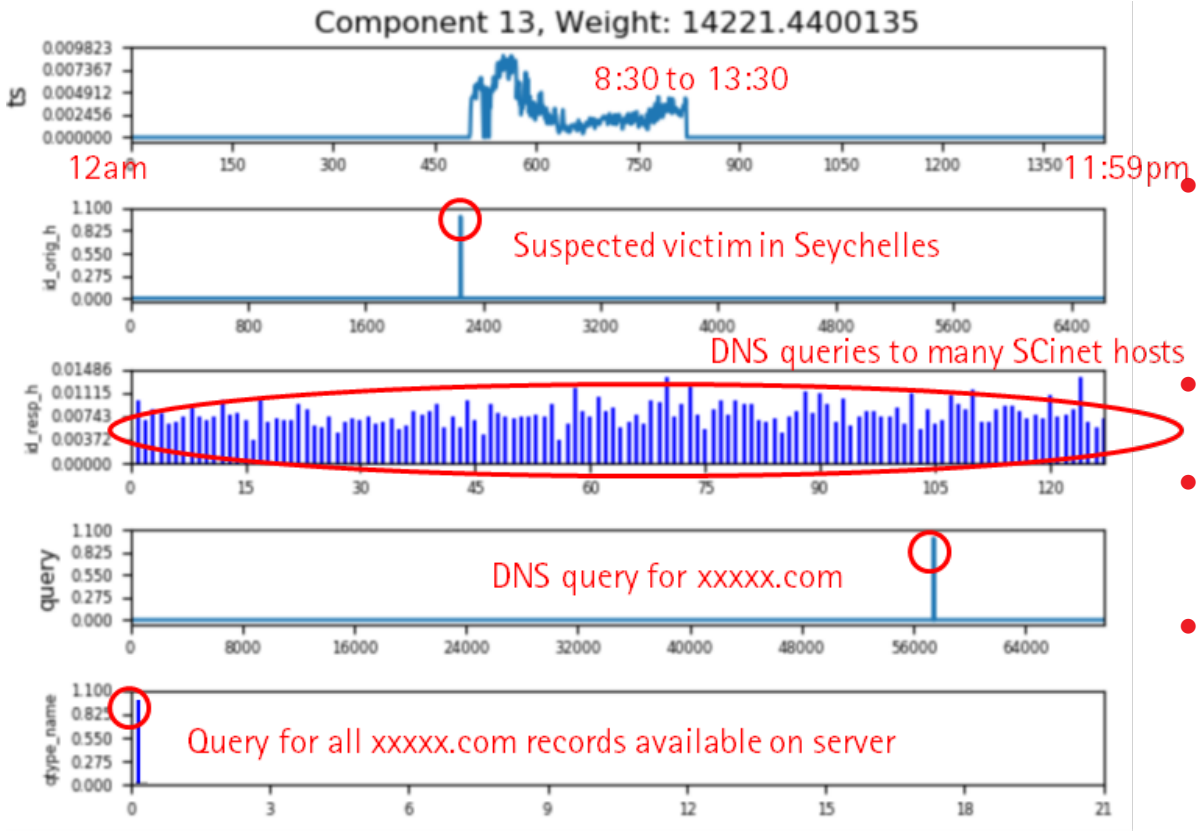


State of the activity at 2pm



DNS Amplification DDoS attack

… as seen with streaming decompositions



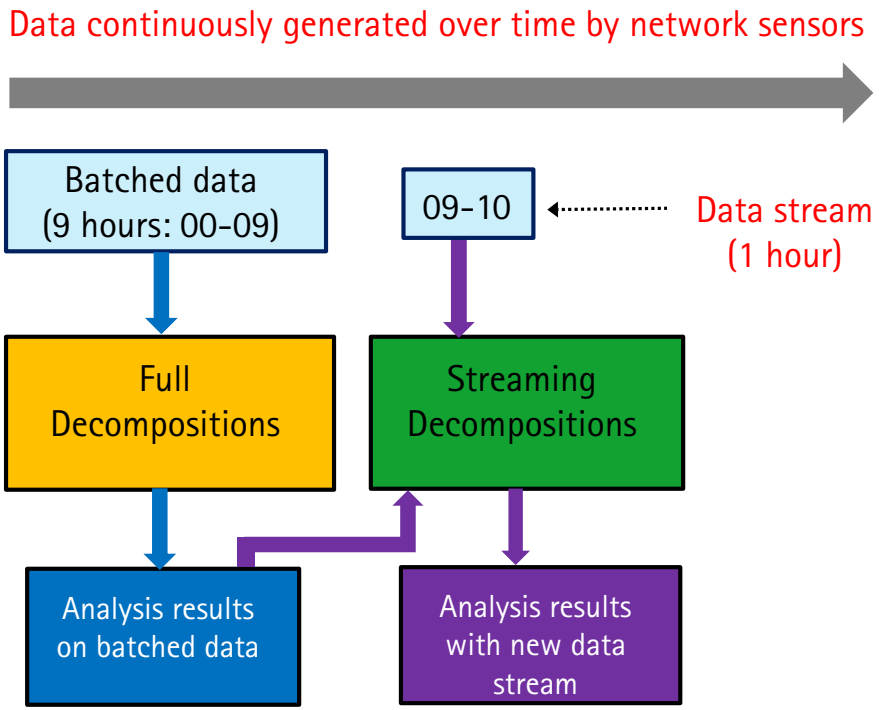
DNS amplification DDoS attack identified from streaming decompositions:

- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
  - Base: 8 hours (12am to 7:59am)
  - Streams: every 1 hour (from 8am)
- Base decomposition on first 8 hours of data took ~200 seconds
- Subsequent streaming decompositions on 1-hour streams took ~3-4 seconds each
- Identified the attack at its onset in near real-time

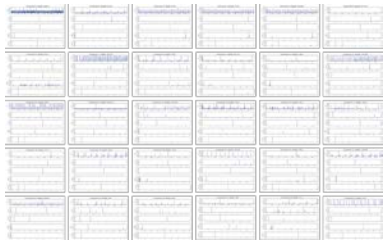
Time x Source IP x Destination IP x DNS Query String x DNS Query Type

ENSIGN Streaming Decompositions

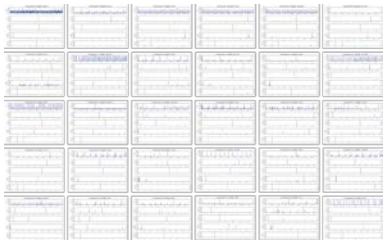
… providing useful meta-information to users



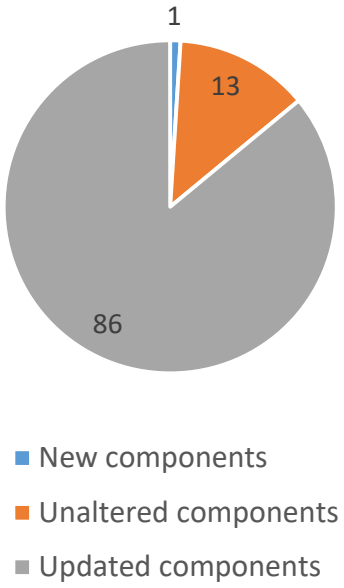
100 Base Components



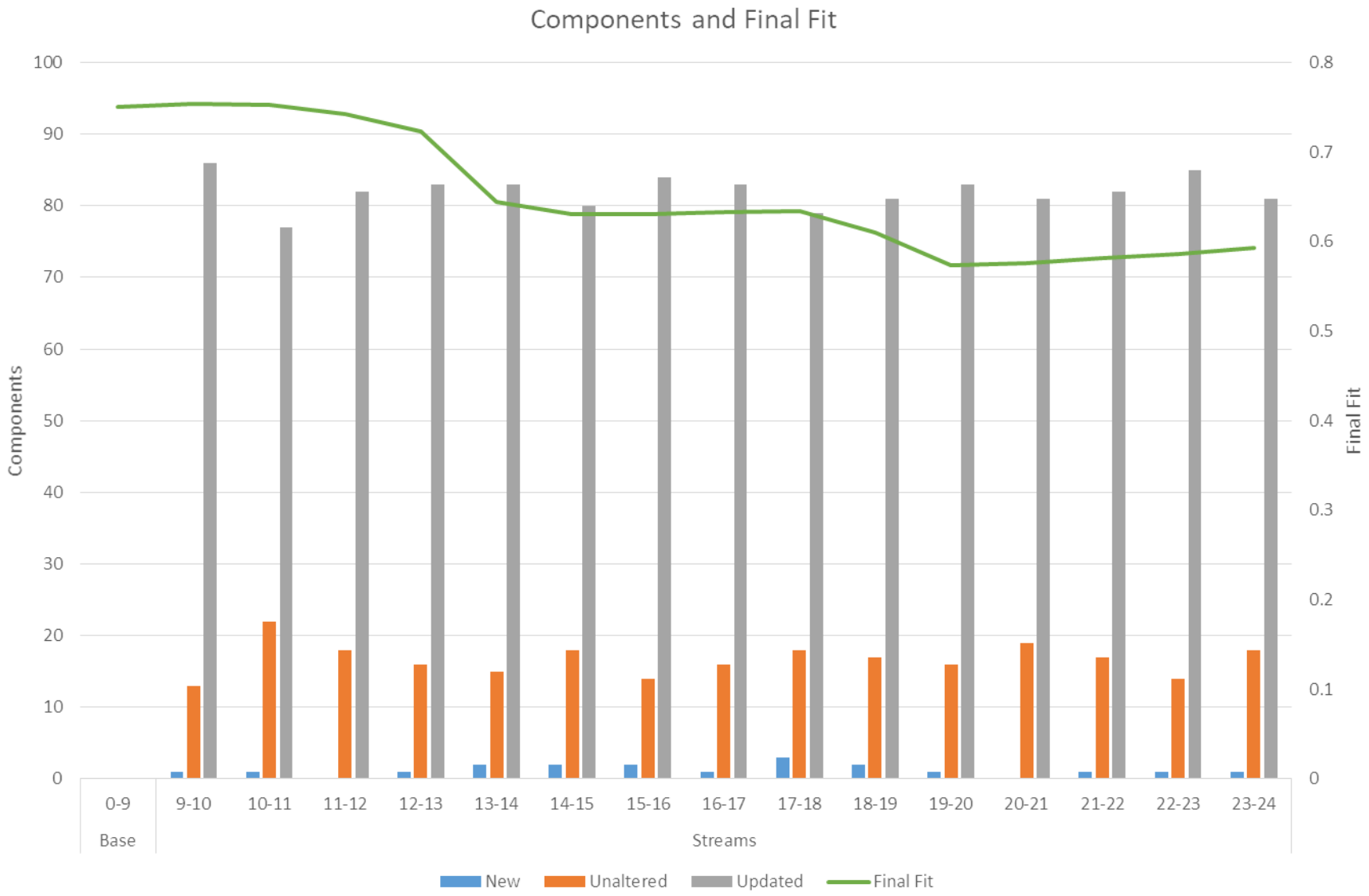
100 Components After Streaming Analysis



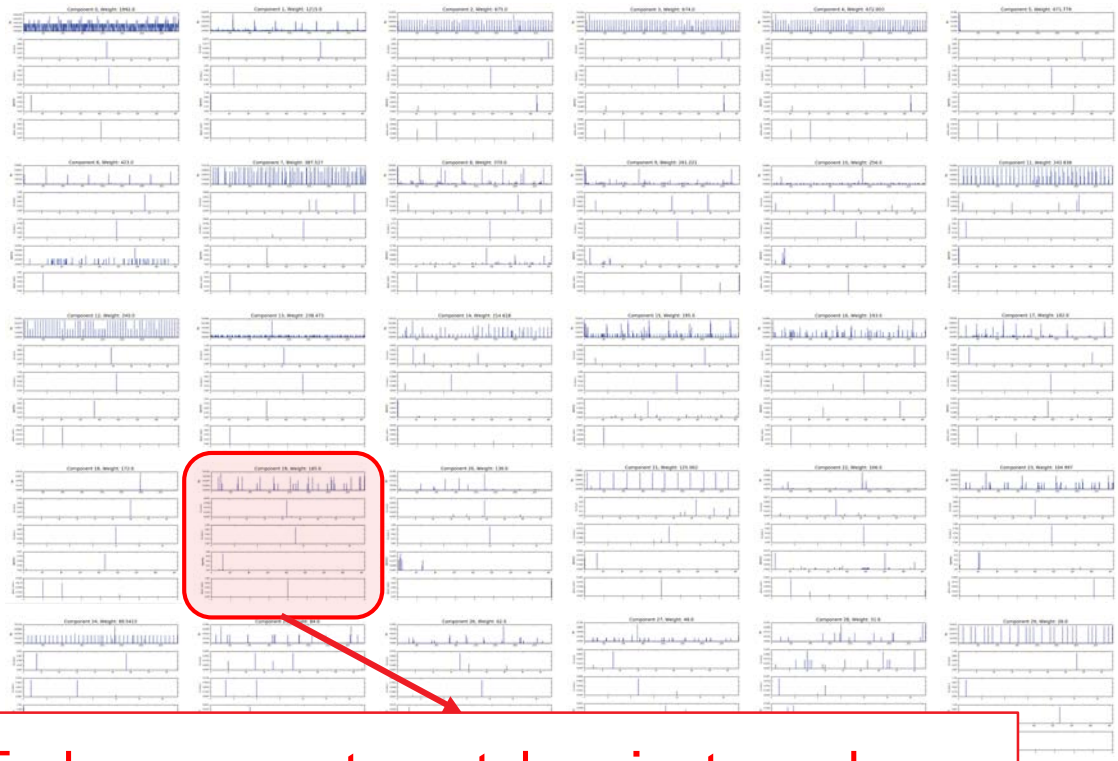
Visualizable report on how components evolve



How the Decomposition Evolves



The Barrier – Automate Component Classification



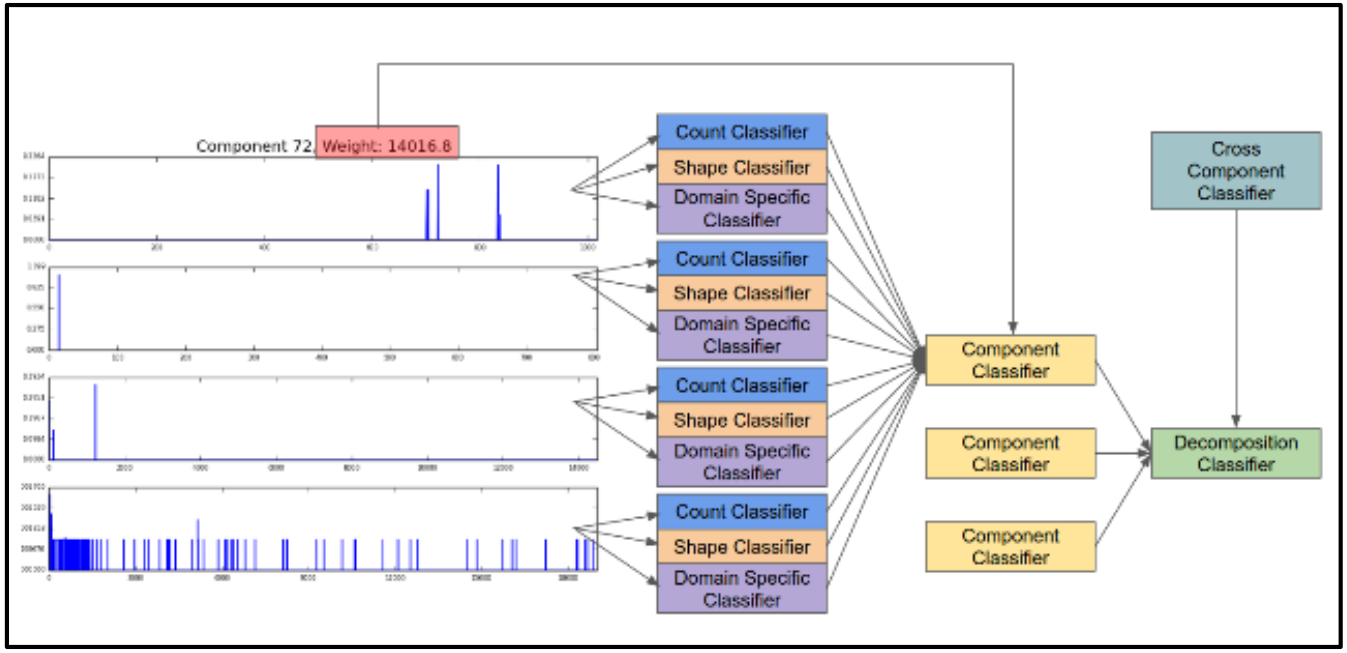
Hundreds of components needed to characterize network traffic

Most components are benign

Each component can take minutes or hours to manually investigate

Which components are interesting?

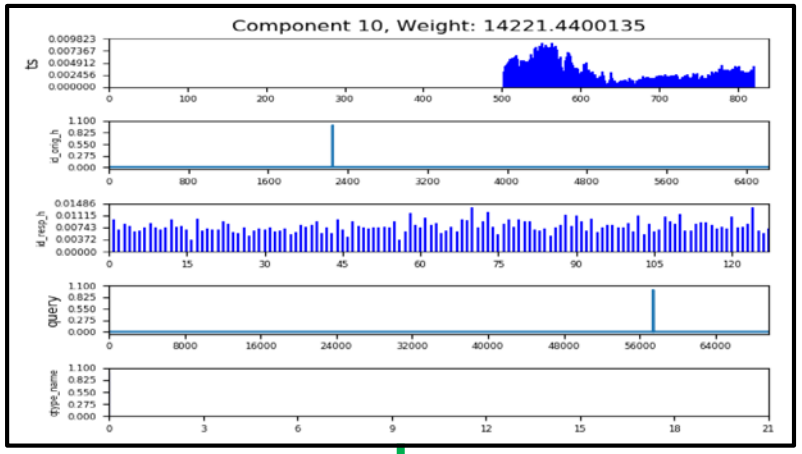
General Classification for Decompositions

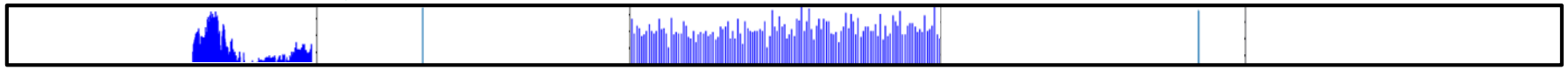


Hierarchical "white box" classification of Modes, Components, and Decomposition

- Count Classifier
  - Finds 1-1, 1-many, many-1 relationships – port scan, network map, DNS amplification
- Shape Classifier
  - Finds recognizable patterns – periodic beaconing, after hours traffic
- Domain Specific Classifier
  - Incorporates domain knowledge – Port 22 is SSH traffic, Ports 80 and 443 are web traffic

Topic Modeling for Component Classification





Latent Dirichlet Allocation (LDA)

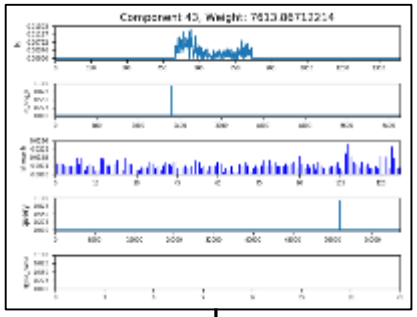
- Well-known Bayesian topic modeling algorithm
- Learns topic model from a corpus of documents
- Infers topic mixture of new documents
- Online updates of topic model
- Commonly used in other applications
  - Bioinformatics
  - Image, video, and sound processing
  - Collaborative filtering

Mapping tensor decompositions to LDA concepts

- Component (as vector) = "document"
- Label = "word"
- Score = "word count"
- Topic = recognizable pattern of network behavior

Topic Model Classification Workflow

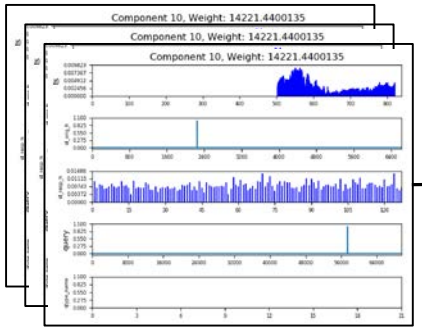
Unseen Component



Analyst Labels Topics

Analyst views decomposition in domain terminology

Historical Decomposition Training Data



2. Label

3. Input

1. Train

Topic Model (LDA)
Topic A: nightly backups
Topic B: DNS amplification
Topic C: nightly testing
...
...
...

4. Infer

Analyst views component as mixture of known network topics

Mixtures without dominant topics indicate anomaly

Topic Mixture of Unseen Component
2% Topic A: nightly backups
98% Topic B: DNS amplification
0% Topic C: nightly testing
...
...
...

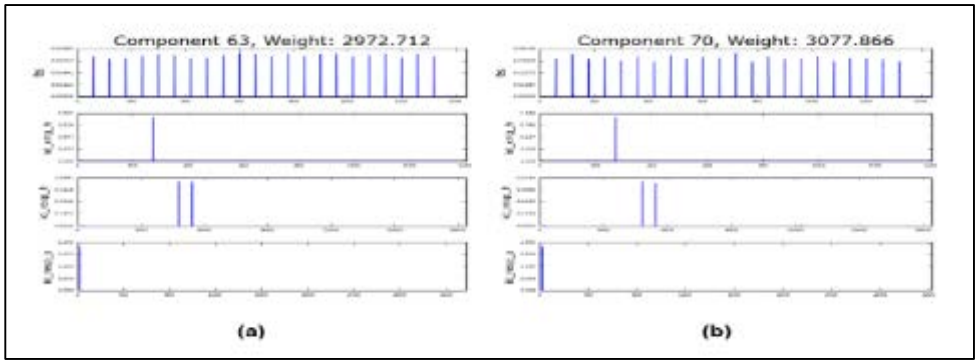
Topic Model Classification Experiments

Small office network traffic decompositions

- Train on 5 days, test on 2 days – 100 topics
- Time, Source IP, Destination IP, Port

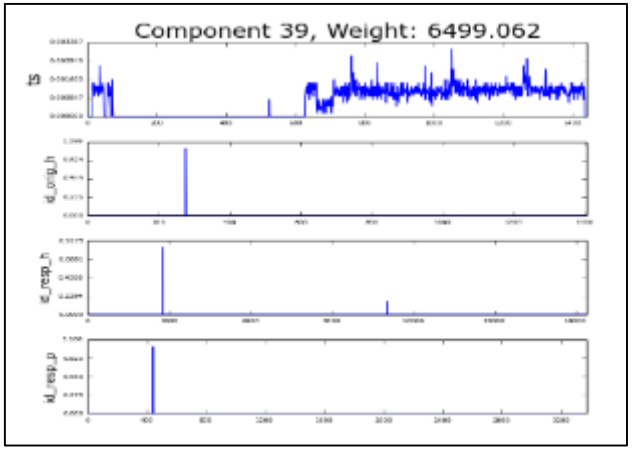
Identified well-known traffic

- Backups, log forwarding, system monitoring, …
- 100% single topic "mixtures"



Identified anomalous traffic

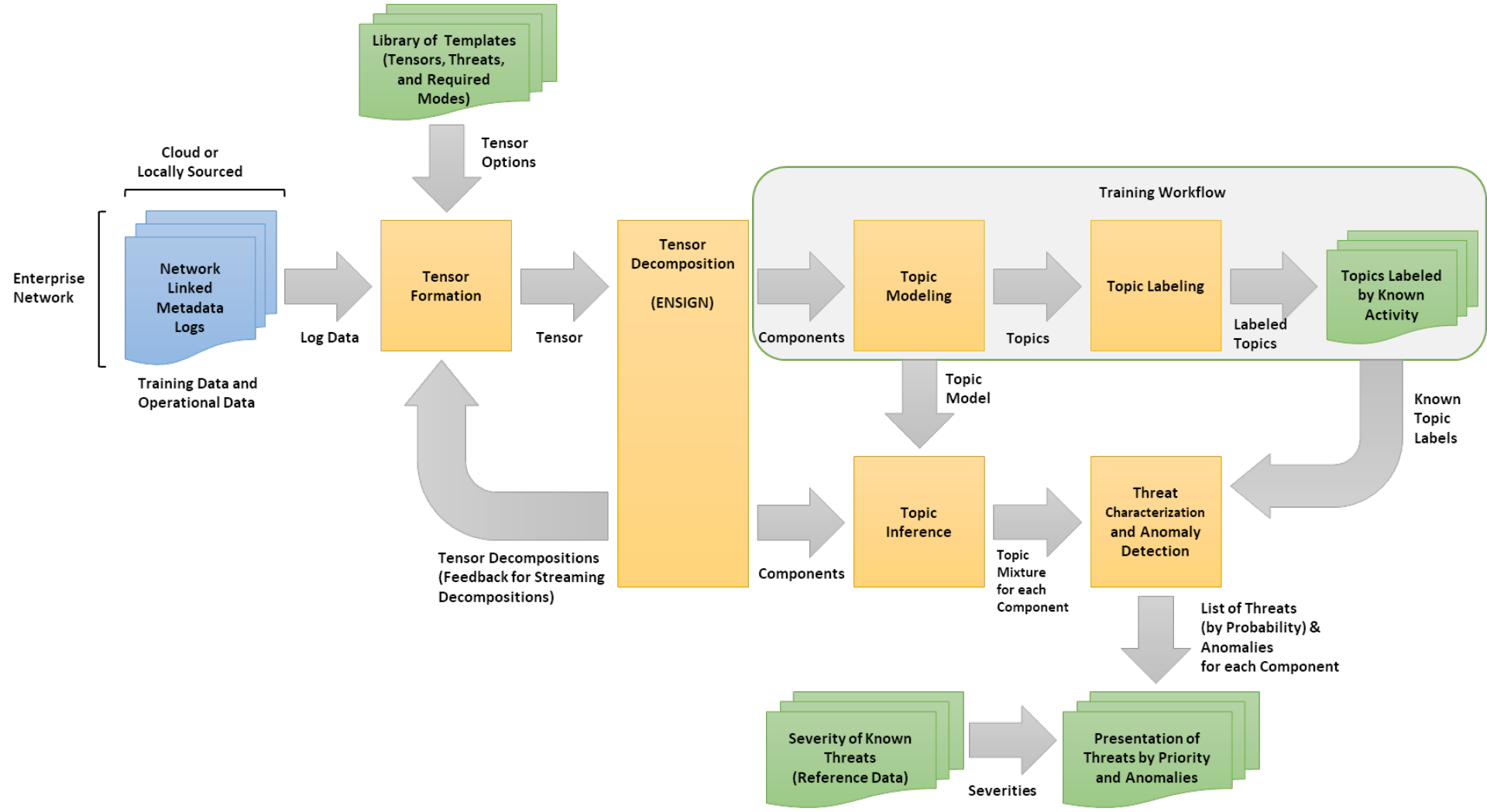
- Unusual long running VPN connection
  - 16 topic mixture
  - Largest part of topic mixture was 16%



Identified clear mixtures of topics

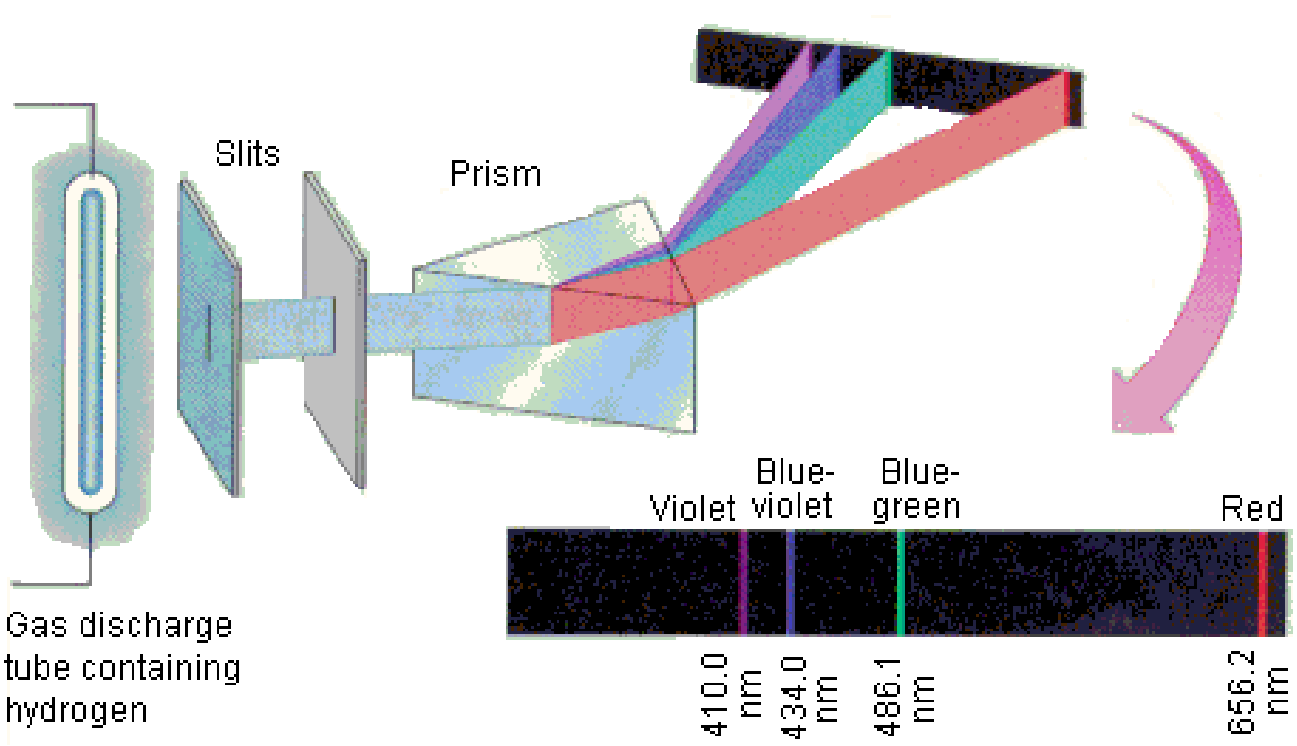
- 2-3 topics matched

The Barrier – Demonstrate a Repeatable Process

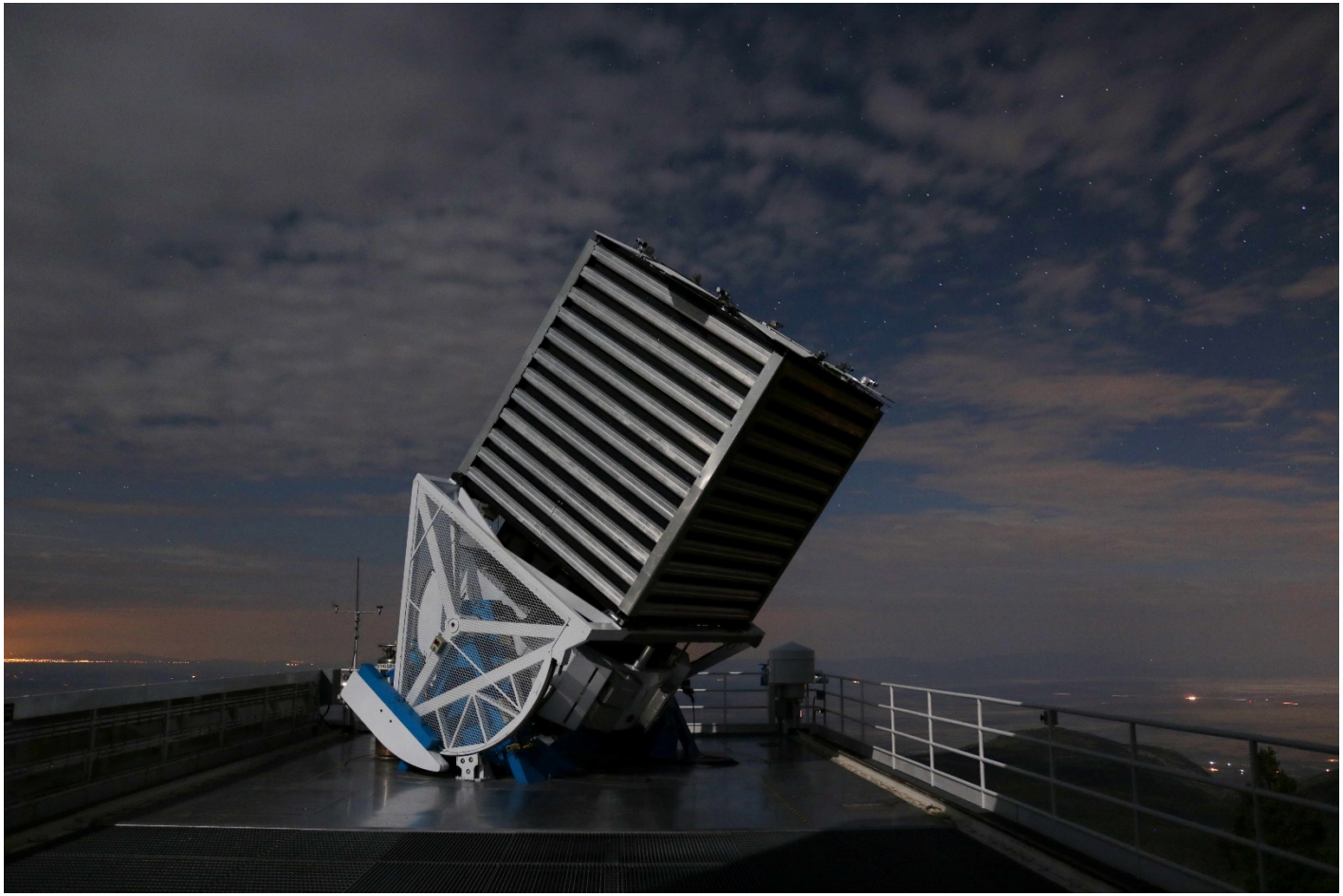


Tensor Creation Library

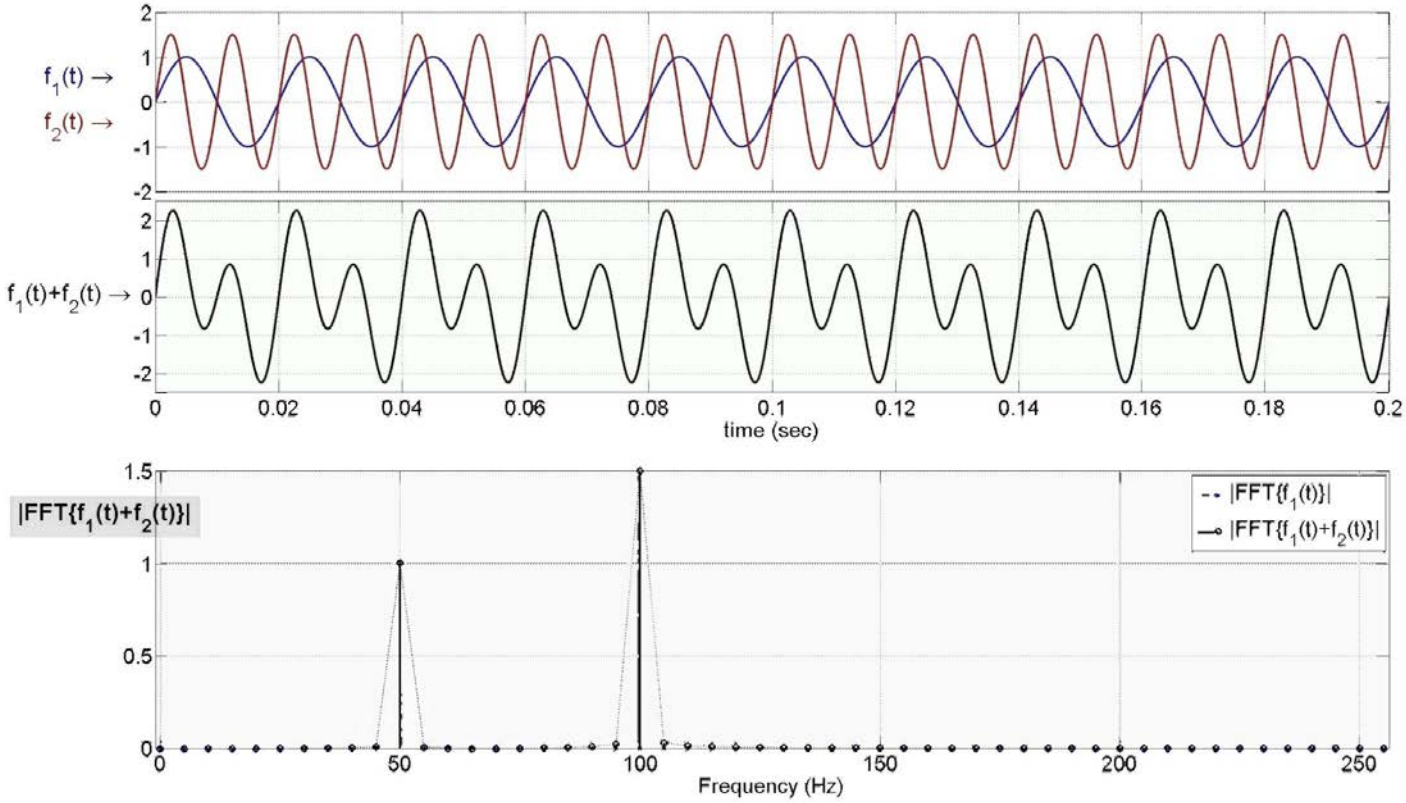
Tensor Name	Bro Log	Tensor Dimensions
Connections	The connections log (conn.log)	Time x Sender IP x Receiver IP x Port
Outgoing	Connections log entries with local sender and external receiver	Time x Sender IP x Receiver IP x Port
Incoming	Connections log entries with local receiver and external sender	Time x Sender IP x Receiver IP x Port
Time Independent	The connections log	Sender IP x Receiver IP x Port x Connection State
File Transfer	The file transfer log (files.log)	Time x Sender IP x Receiver IP x MIME-Type
HTTP	The HTTP traffic log (http.log)	Time x Sender IP x Receiver IP x URI x User Agent
DNS Query	All queries from the DNS log (dns.log)	Time x Sender IP x Receiver IP x Query x Query Type


Slits
Prism
Violet
Blue-violet
Blue-green
Red
Gas discharge tube containing hydrogen
410.0 nm
434.0 nm
486.1 nm
656.2 nm

Looking at distant stars



Source: Sloan Digital Sky Survey

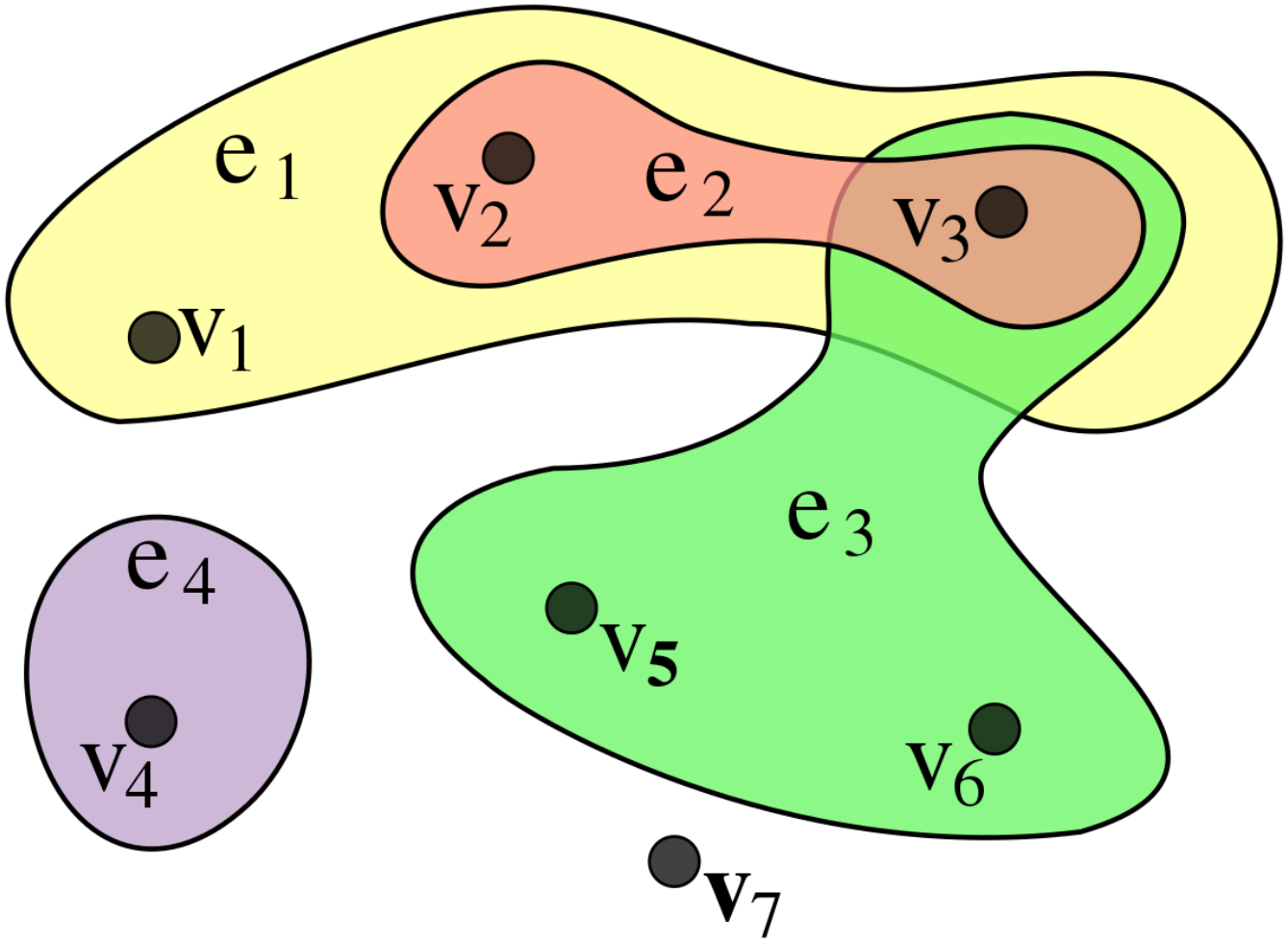


https://www.youtube.com/watch?v=-GYB7khblA0

Reservoir Network Traffic Data

1367954284.775933	10.0.2.15	41173	72.22.185.216	80	tcp	0	0	-
1367954284.776223	10.0.2.15	35904	23.6.146.127	80	tcp	0	0	-
1367954284.902059	10.0.2.15	57323	63.251.85.24	80	tcp	2294	797	wt.o.nytimes.com
367954285.233317	10.0.2.15	51934	54.243.131.12	80	tcp	578	184	
3.0950611367954281.046495	10.0.2.15	51237	10.1.1.1	53	udp	32	200	-
1367954281.980046	10.0.2.15	61472	10.1.1.1	53	udp	33	128	-
1367954287.375995	10.0.2.15	55776	74.125.226.214	443	tcp	489	2658	-
1367954282.655601	10.0.2.15	52392	72.22.185.207	80	tcp	0	0	-
1367954282.651031	10.0.2.15	45201	72.22.185.198	80	tcp	0	0	-
1367954282.654099	10.0.2.15	52391	72.22.185.207	80	tcp	0	0	-
1367954282.660982	10.0.2.15	34639	72.22.185.214	80	tcp	0	0	-
1367954282.689652	10.0.2.15	58404	72.22.185.199	80	tcp	0	0	-
1367954283.208290	10.0.2.15	34980	72.21.91.19	80	tcp	394	313	p.typekit.net

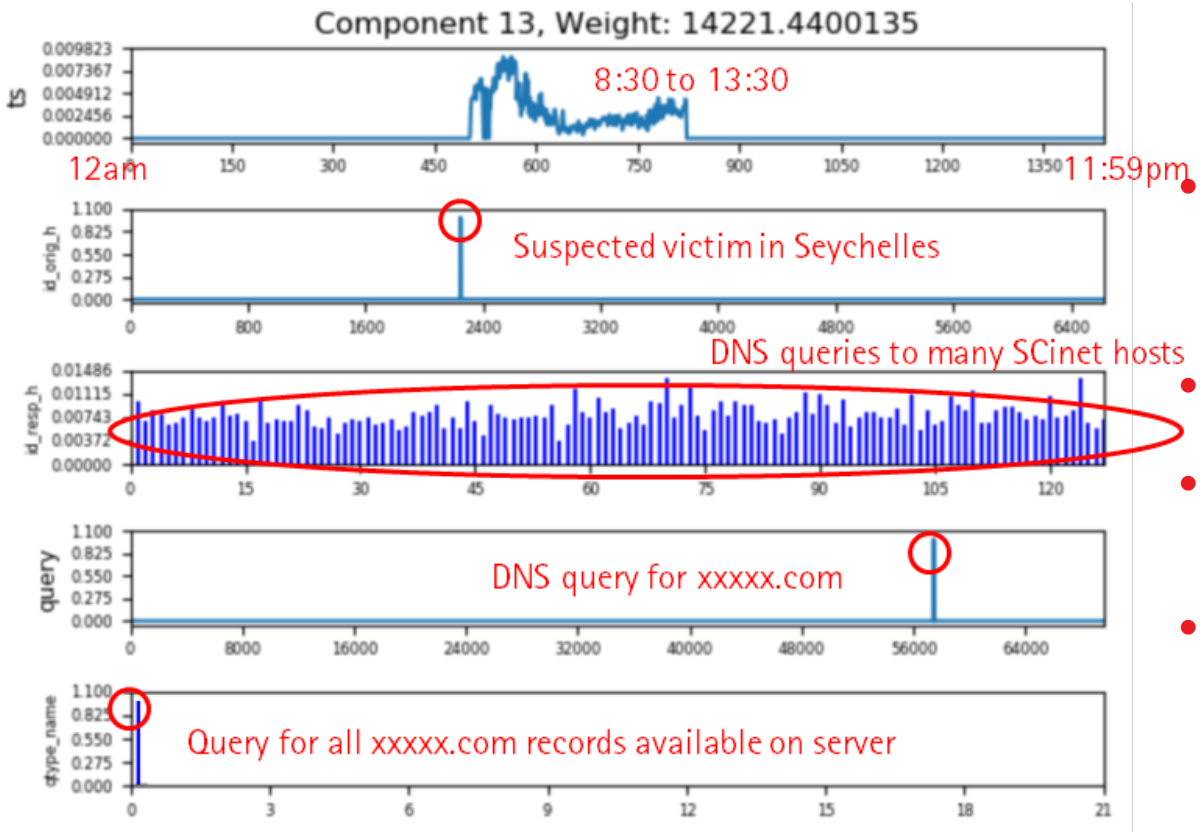
Hypergraph



Source: https://en.wikipedia.org/wiki/Hypergraph

DNS Amplification DDoS attack

… as seen with streaming decompositions



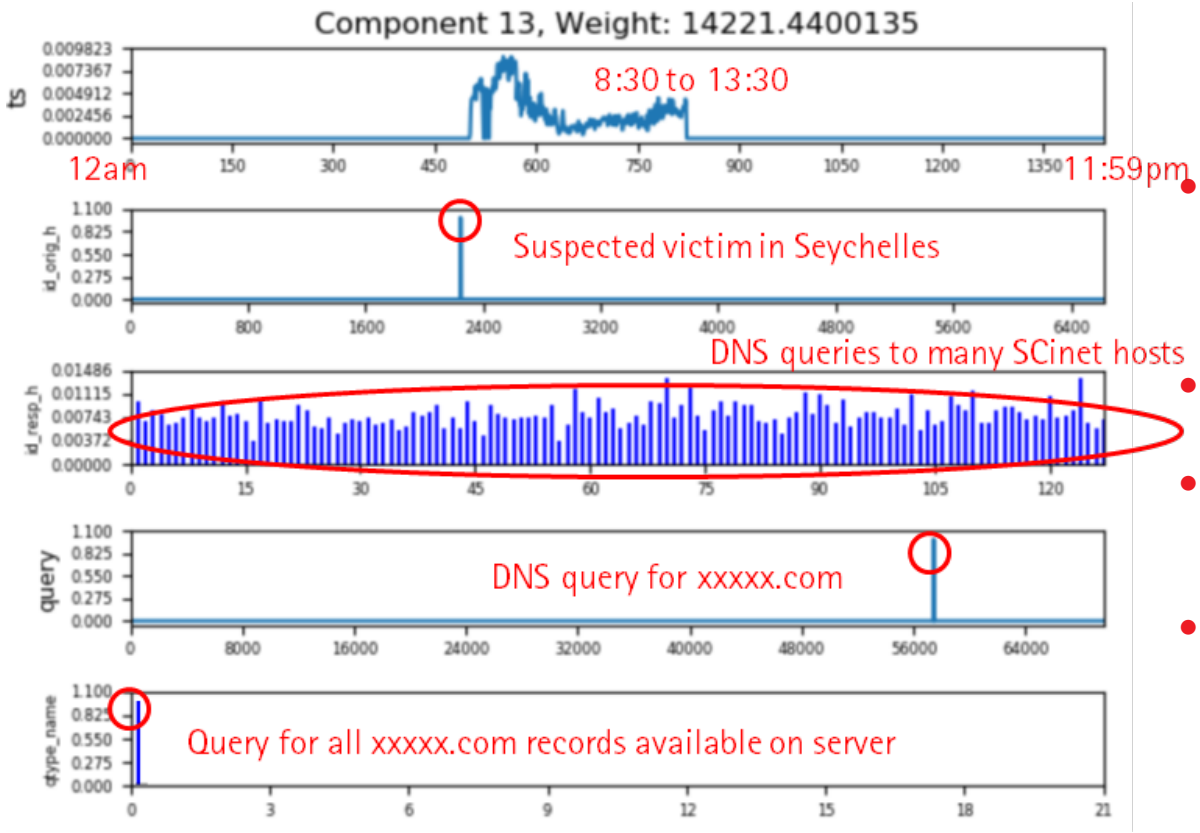
DNS amplification DDoS attack identified from streaming decompositions:

- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
  - Base: 8 hours (12am to 7:59am)
  - Streams: every 1 hour (from 8am)
- Base decomposition on first 8 hours of data took ~200 seconds
- Subsequent streaming decompositions on 1-hour streams took ~3-4 seconds each
- Identified the attack at its onset in near real-time

Time x Source IP x Destination IP x DNS Query String x DNS Query Type

DNS Amplification DDoS attack

… as seen with streaming decompositions



DNS amplification DDoS attack identified from streaming decompositions:

- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
  - Base: 8 hours (12am to 7:59am)
  - Streams: every 1 hour (from 8am)
- Base decomposition on first 8 hours of data took ~200 seconds
- Subsequent streaming decompositions on 1-hour streams took ~3-4 seconds each
- Identified the attack at its onset in near real-time

Time x Source IP x Destination IP x DNS Query String x DNS Query Type

The Sparse Multidimensional Fast Fourier Transform

A Sparse Multi-Dimensional Fast Fourier Transform with Stability to Noise in the Context of Image Processing and Change Detection

Pierre-David Letourneau, M. Harper Langston, and Richard Lethin
Reservoir Labs
letourneau,langston,lethin@reservoir.com

***Abstract*—We present the sparse multidimensional FFT (sMFFT) for positive real vectors with application to image processing. Our algorithm works in any fixed dimension, requires an (almost)-optimal number of samples ($\mathcal{O}\left(R\log\left(\frac{N}{R}\right)\right)$) and runs in $\mathcal{O}\left(R\log\left(\frac{N}{R}\right)\right)$ complexity (to first order) for N unknowns and R nonzeros. It is stable to noise and exhibits an exponentially small probability of failure. Numerical results show sMFFT's large quantitative and qualitative strengths as compared to ℓ_1-minimization for Compressive Sensing as well as advantages in the context of image processing and change detection.**

I. Introduction

Since its popularization in the 1960s [1], the Fast Fourier Transform (FFT) algorithm has played a crucial role in multiple areas of computational mathematics and has rendered possible great achievements in scientific computing [2], signal processing [3] and computer science [4] by reducing the computational cost of computing the Discrete Fourier Transform (DFT) of an N-vector from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log(N))$.

When the vector to be recovered is sparse, it is possible to significantly improve on this cost [5]–[7]. That is, if f is a $N\times 1$ vector corresponding to the DFT of a noisy $N\times 1$ vector $\hat{f}$ containing at most R nonzero elements, it is possible to recover f using a number of samples much smaller than the traditional "Nyquist rate" ($\ll \mathcal{O}(N)$), and in computational complexity much lower than that of the FFT ($\ll \mathcal{O}(N\log(N))$). These schemes are generally referred to as "sparse Fast Fourier Transform" (sFFT) algorithms and fall into two main categories: deterministic (e.g., [7]) or randomized (e.g., [5]) algorithms. Of the two, randomized algorithms have had the most success in practice. Sparse FFT algorithms can further be split into more categories: $1D$ versus multidimensional (d), and exact versus noisy measurements. Also, among the fastest known sFFT algorithms, the probability of failure p of randomized algorithms decays slowly with N [5] and or *a priori* knowledge about the support is required (e.g., contained within a cyclic interval [6], [7]). Further, many require the unknowns N to be a power of two [6], [8], only work in one or two dimensions [6], [8] or do not scale well with ambient dimension d [5], [9].

The patent-pending sparse Multidimensional Fast Fourier Transform (sMFFT)[1] algorithm we have developed provably achieves the best known sample and computational complexities while optimizing over key features as detailed here:

- Requires $\mathcal{O}\left(R\sqrt{\log(R)}\log\left(\frac{N}{R}\right)\right)$ samples and $\mathcal{O}\left(R\log^{3/2}(R)\log\left(\frac{N}{R}\right)\right)$ computational complexity;
- Cost scales linearly with dimension d;
- Exponentially small probability of failure;
- Stable to noise;
- Works for any positive integer N;
- Works for positive real vectors only.

In this sense, our algorithm matches the optimal sampling complexity [10] (within a $\log(\log(N))^{-1}$ factor), and the best time complexity achieved so far [6], [7]. Additionally, the sMFFT is the first multidimensional sFFT that scales linearly with dimension. Indeed, most previous endeavors were targeted at the $1D$ sparse FFT, whereas the only currently-existing multidimensional versions [5], [9] have a cost that scales at least exponentially with dimension. Finally, our algorithm does share with [6] the characteristic of being designed for real positive vectors only, but for many applications of interest, e.g. radar imaging [11], this is a valid hypothesis.

Below, we describe the ideas behind sparse FFTs in Section II, followed by key details of of our sMFFT algorithm in Section III. Numerical results are shown in Section V, highlighting the strengths of our approach over standard FFTs as well as Compressive Sensing [12].

II. Description of the Sparse FFT Problem

In Table I, we summarize all quantities of interest. We begin by describing the $1D$ scheme. See Section IV for the

Symbol	Description
$f(\cdot)$	Periodic, bandlimited function of interest.
f	Vector containing Fourier coefficients of $f(\cdot)$.
$\mathcal{S}$	Set containing indices of nonzero elements of $\hat{f}$, i.e., support
N	Total number of unknowns
R	Upper bound on cardinality of the support $\mathcal{S}$ of $\hat{f}$
d	Ambient dimension
p	Probability of failure of randomized algorithm
C	Algorithmic constant

TABLE I
Summary of important quantities

[1]U.S. Patent Application No. 62/286,732

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

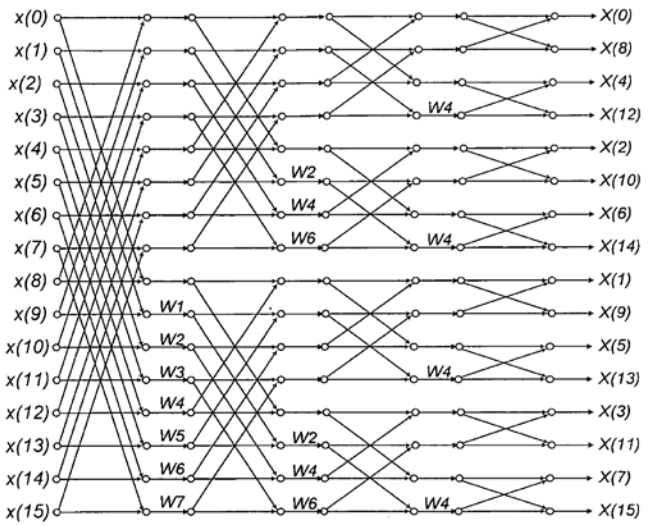


Fig. 4

Source: https://www.google.com/patents/US20120041996

Letourneau et. al, HPEC 2016

Shoulders of Giants

Much of this work builds on foundational contributions in mathematics and algorithms developed over decades by the community of tensor researchers.

In particular, we would like to acknowledge the work of Tamara Kolda and the staff at Sandia National Laboratories.

http://www.sandia.gov/~tgkolda/

- Kolda, T., Bader, B., **Tensor Decompositions and Applications**, *SIAM Review, 51(3), pp. 455-500, 2009*.
- Chi, E., Kolda, T., **On Tensors, Sparsity, and Nonnegative Factorizations**, *SIAM Journal on Matrix Analysis and Applications 33.4 (2012):1272-1299*.
- Tensor Toolbox for MATLAB and C++

Conclusion

Reservoir Labs

- https://www.reservoir.com

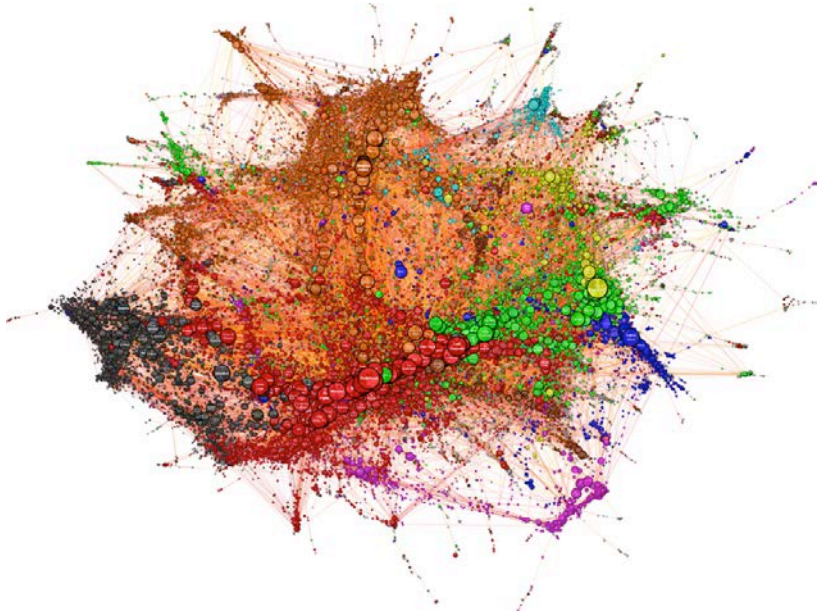
Contact Jim!

- ezick@reservoir.com

Recent Papers

- *Enhancing Network Visibility through Tensor Analysis*
  M. Baskaran, T. Henretty, D. Bruns-Smith, J. Ezick, R. Lethin in SC17 INDIS Workshop, Nov 2017

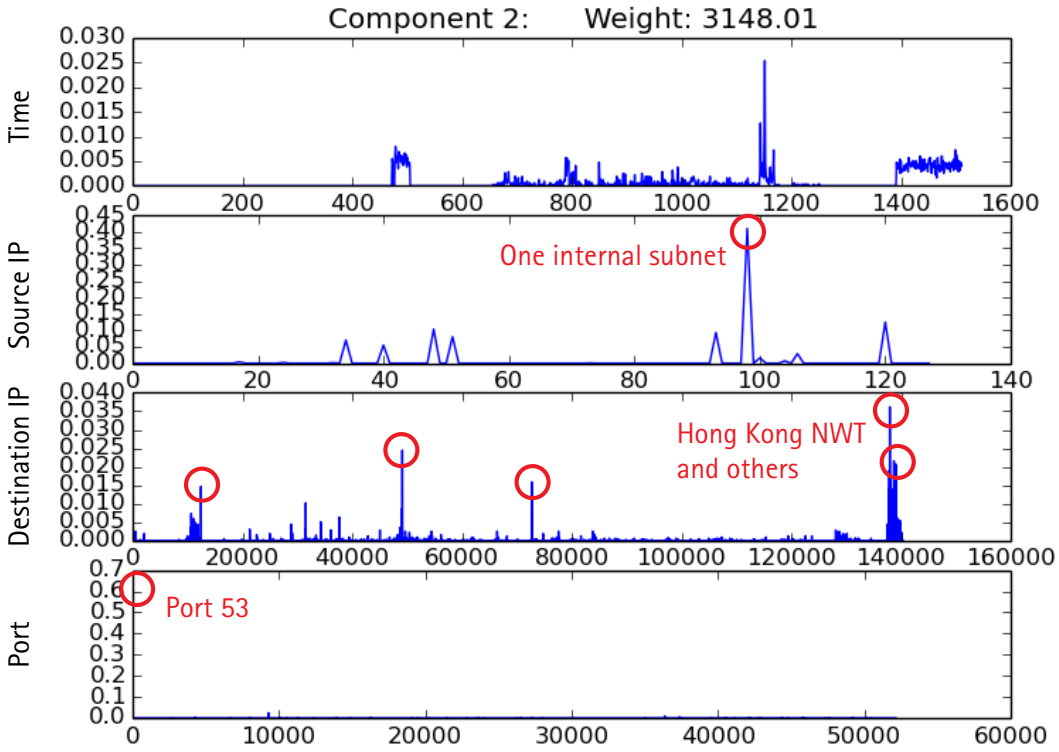
- *Memory-efficient Parallel Tensor Decompositions*
  M. Baskaran, T. Henretty, B. Pradelle, M. H. Langston, D. Bruns-Smith, J. Ezick, R. Lethin in IEEE HPEC, Sep 2017 (Best Paper Award)



Tensor decompositions provide a fast, scalable, practical linear algebra based solution to finding patterns in linked metadata

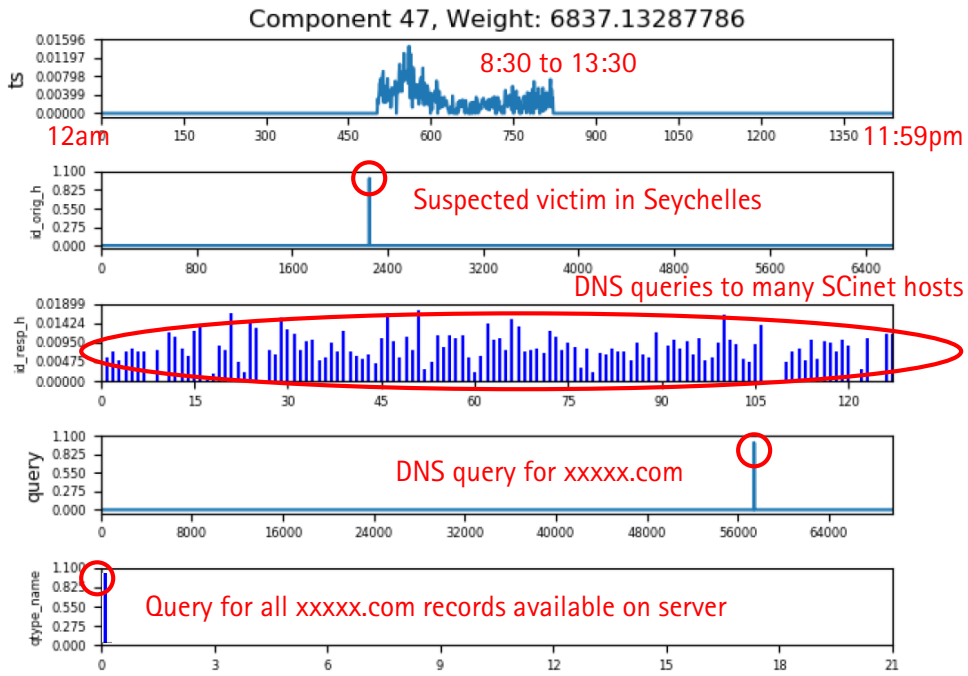
Raw Materials

Example: Suspicious DNS Traffic



Time x Source IP x Destination IP x Port

DNS Amplification DDoS Attack



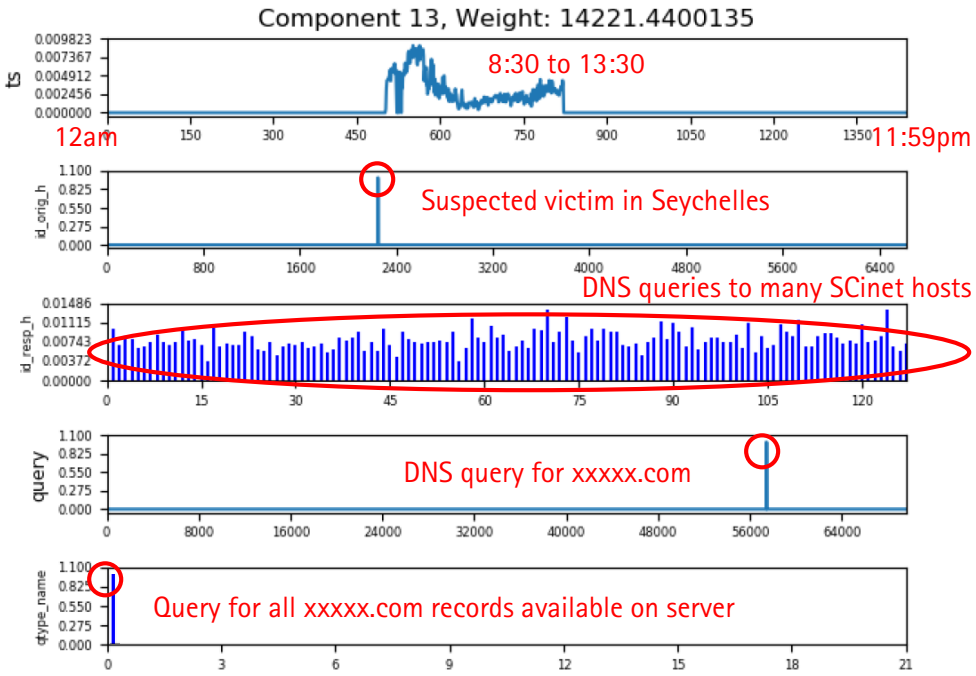
DNS amplification DDoS attack identified from full tensor decomposition:

- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
- Tensor created from the log:
  - 5 million non-zeros
- Decomposition took ~500 seconds

Time x Source IP x Destination IP x DNS Query String x DNS Query Type

DNS Amplification DDoS attack

… as seen with streaming decompositions



DNS amplification DDoS attack identified from streaming decompositions:

- Analyzed DNS log collected over 24 hours (12am to 11:59pm)
  - Base: 8 hours (12am to 7:59am)
  - Streams: every 1 hour (from 8am)
- Base decomposition on first 8 hours of data took ~200 seconds
- Subsequent streaming decompositions on 1-hour streams took ~3-4 seconds each
- Identified the attack at its onset in near real-time

Time x Source IP x Destination IP x DNS Query String x DNS Query Type