
Tracking Hidden Groups Using Communications

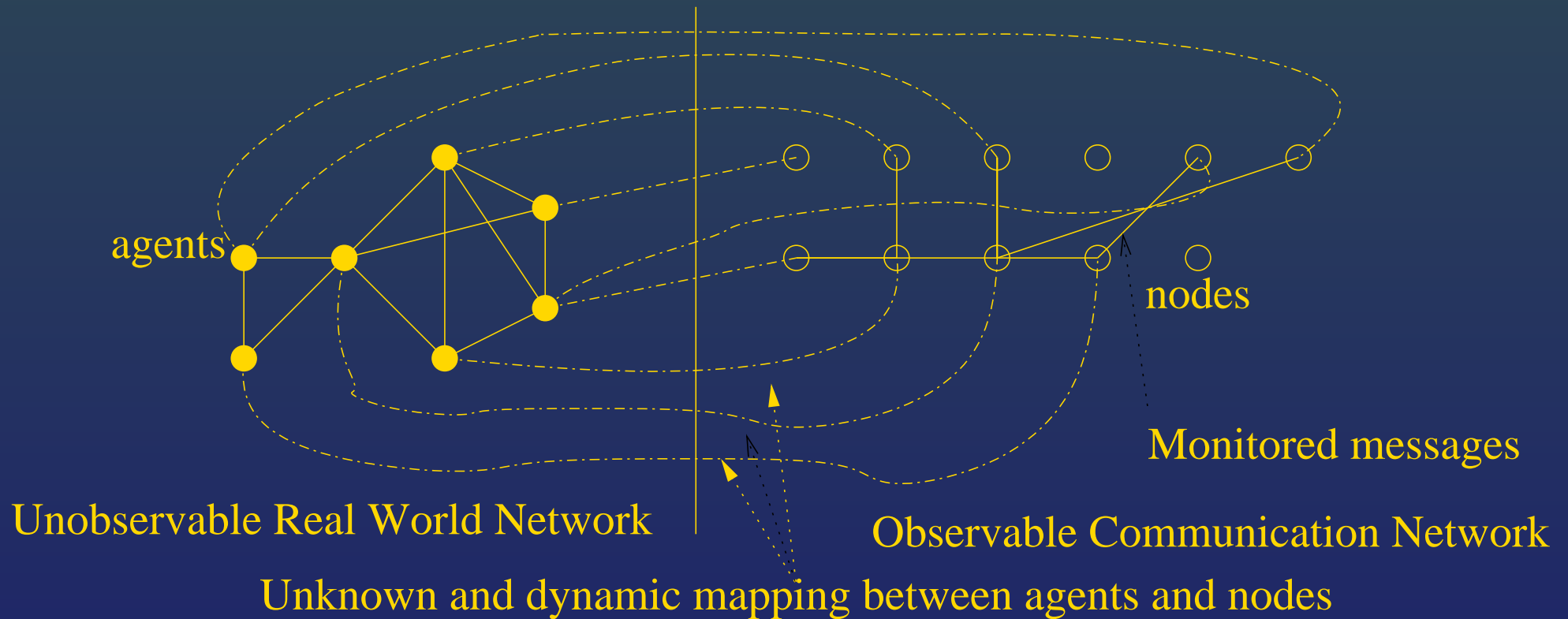
Sudarshan S. Chawathe

University of Maryland, College Park

Motivation

- A group of car thieves who coordinate their actions using disposable prepaid cell phones.
- A group of crackers who attack computer systems using different IP addresses.
- A group of individuals who continually change their mailing addresses.

Problem Definition



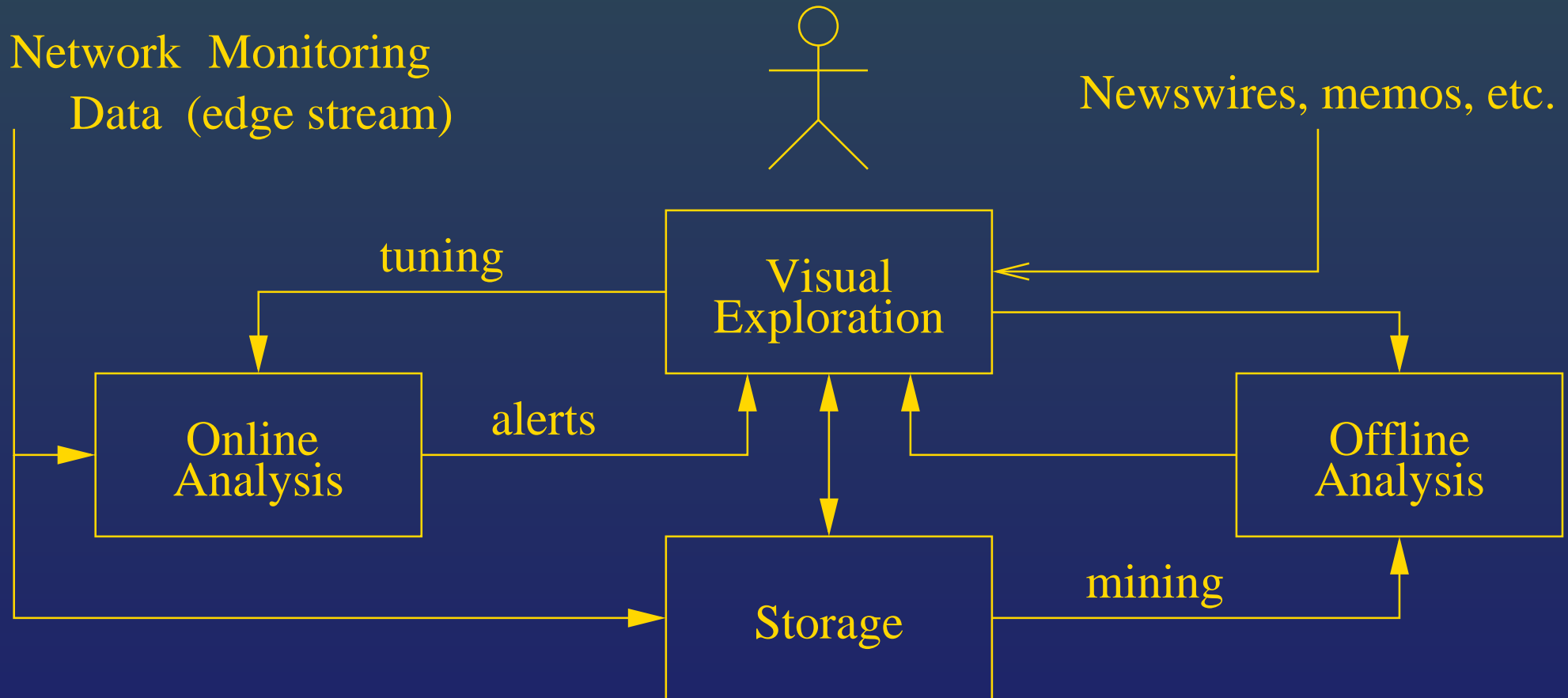
Terminology

- *Network*: the communication network being monitored.
- *Node*: a communication device that is **identifiable** by the network.
 - Phone numbers in telephone networks.
 - IP addresses in computer networks.
 - Postal address in physical mail network.
- *Agent*: an entity that is to be tracked.
 - Use nodes to communicate.
 - Not directly identifiable.

Streaming Data

- Call completion records, TCP/IP sessions, ...
- High volume: cannot simply store everything and use static methods.
- Each data item seen once.
- Order controlled by source of data.
- Recent work: aggregates, order statistics, clustering, ...

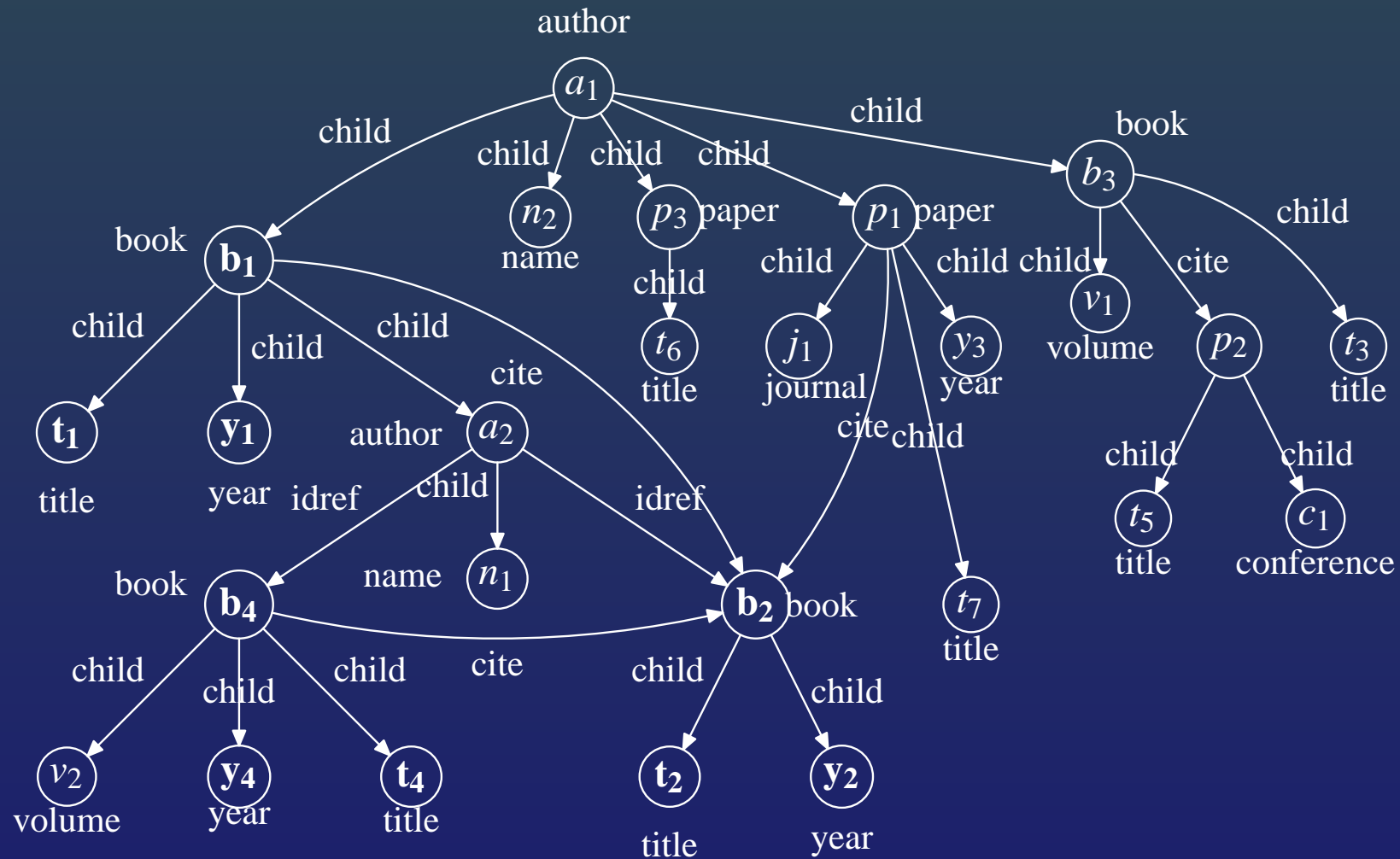
System Architecture



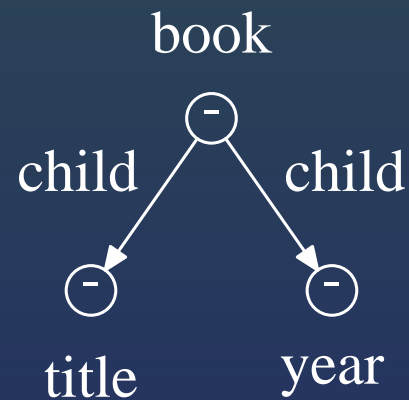
Detecting Frequent Patterns

- Find frequently-occurring substructures in graph structured data.
- Given a graph G , find graphs S (*substructures*) such that G contains *at least* N distinct subgraphs G' that are isomorphic to S .
- Alt.: ... Find the K *most frequent* substructures.
- Trade-off between size and frequency of substructure.
 - Single-node graph occurs very frequently; not very useful.
 - User-controlled parameter serves as input (slider).

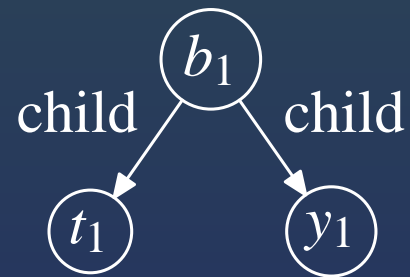
Sample Data



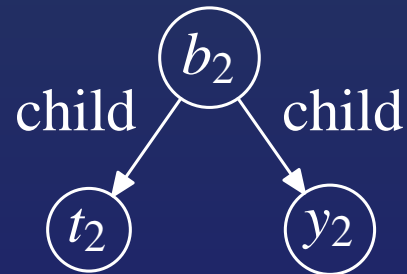
Structures and Instances



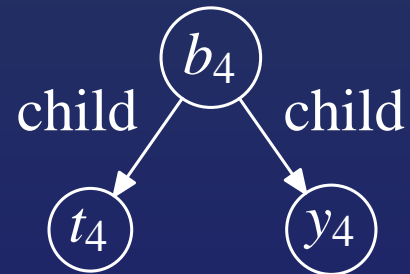
Structure



Subgraph 1



Subgraph 2



Subgraph 3

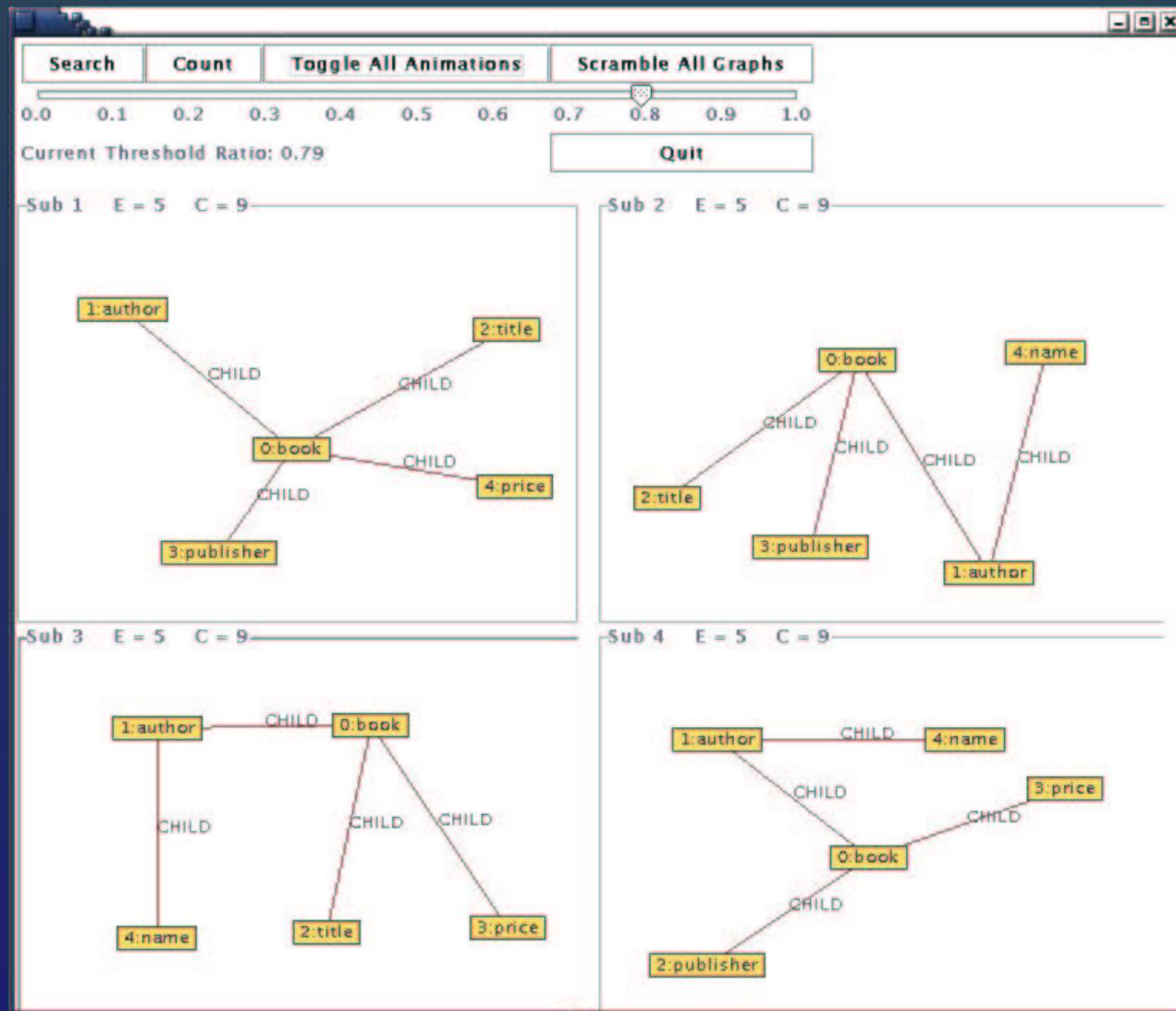
A Hard Problem

- Naive solution:
 - Enumerate all substructures.
 - Count the number of occurrences of each.
 - Return those with at least N occurrences (or top- K).
- Can we do better?
 - In theory, not much; related to hard problems:
 - Subgraph isomorphism.
 - Find support of the most frequent k -vertex structure.
 - Existence of a k -vertex clique in an n -vertex graph.
 - In practice, sure!
 - Good *expected* case performance.

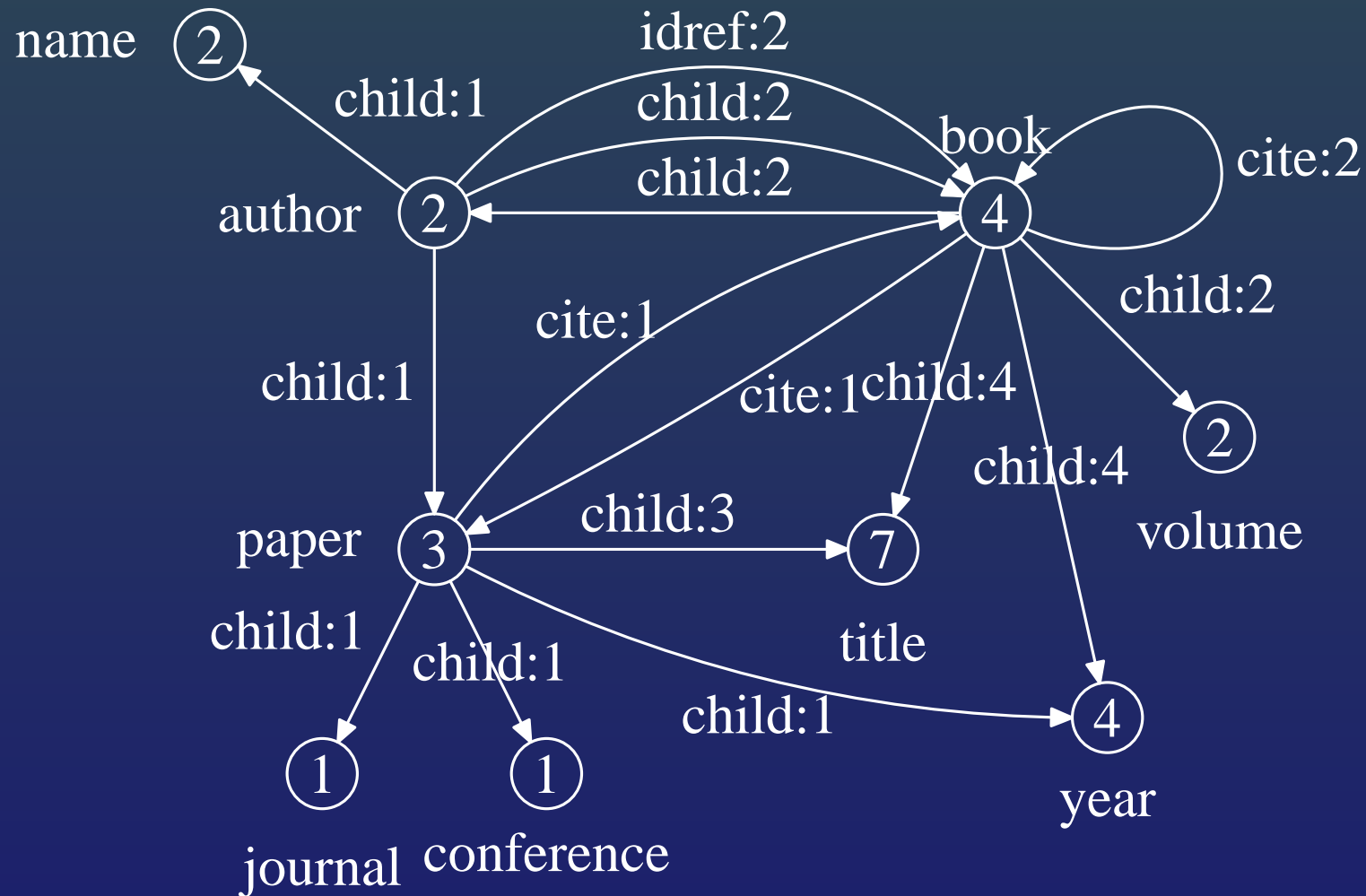
The SEuS Solution

- Phase 1: *Summarization*: Generate a summary structure that concisely captures regularity in data.
 - Similar to *data guides*, *XSketches*, etc.
 - Single sequential scan of database.
- Phase 2: *Candidate Generation*: Use summary to weed out structures that cannot be frequent.
 - *Real-time* feedback based on slider: data exploration.
 - Conservative pruning of search space.
- Phase 3: *Support Counting*: Generate accurate counts of selected substructures.
 - Add edges to current set of substructures (growing); prune away low frequency graphs.

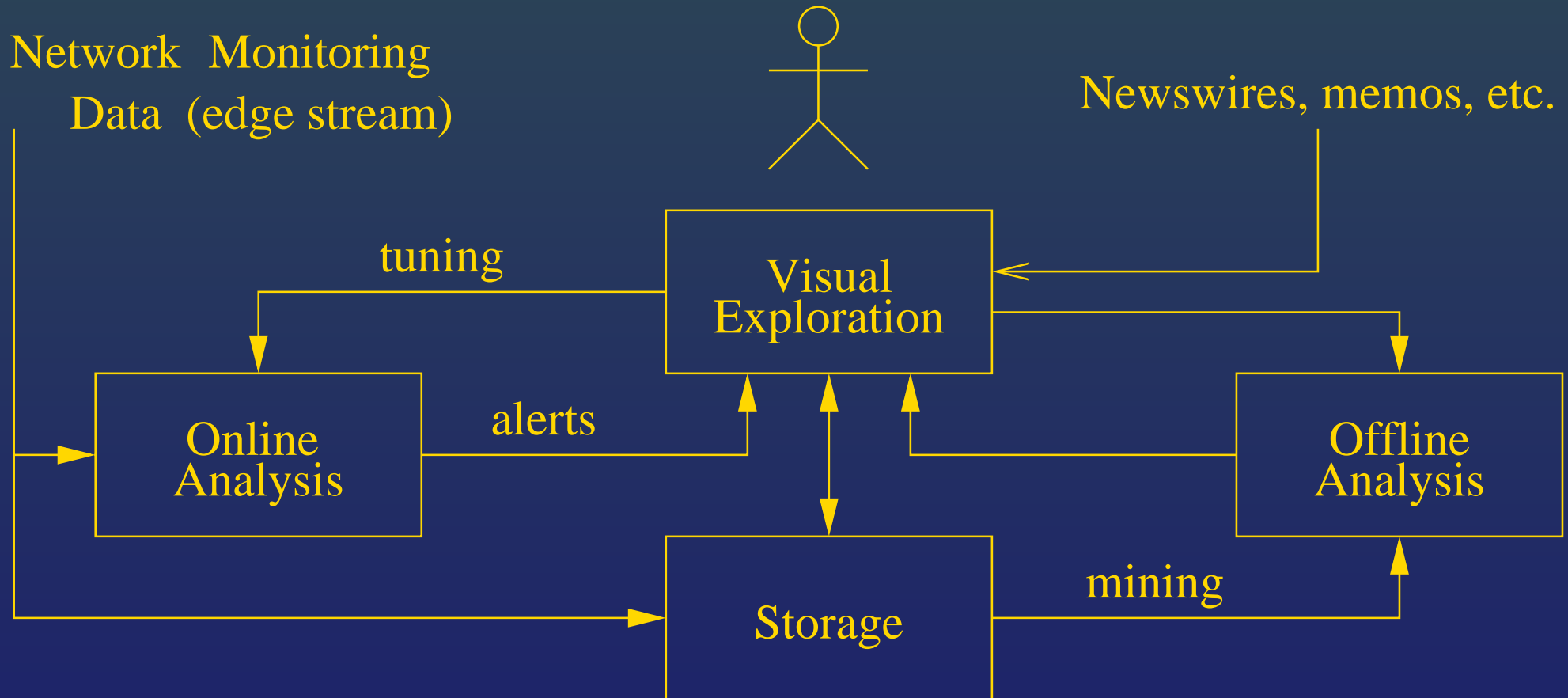
SEuS in Action



Summary Graph



System Architecture



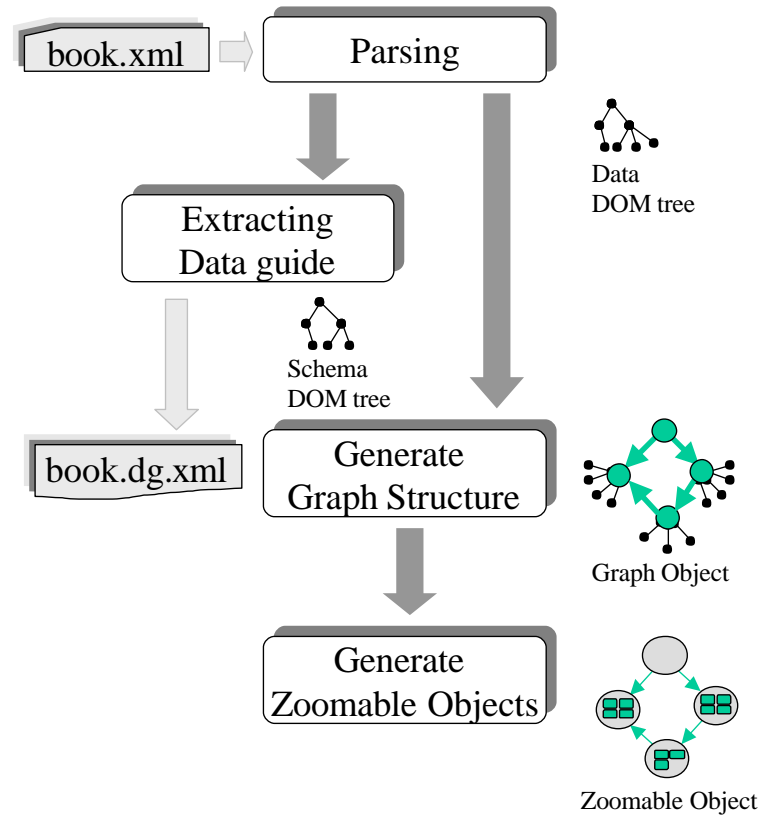
Visual Data Exploration

- Traditional view: *Schema* comes first and is used to browse and query the database instance.
- *Semistructured database* view: Data comes first; schema is inferred from data, and continually changes.
- Two systems based on summarization:
 - VQBD: data guides, zooming graphical interface.
 - FuzzyTree: tree-structured data, directory-tree interface.

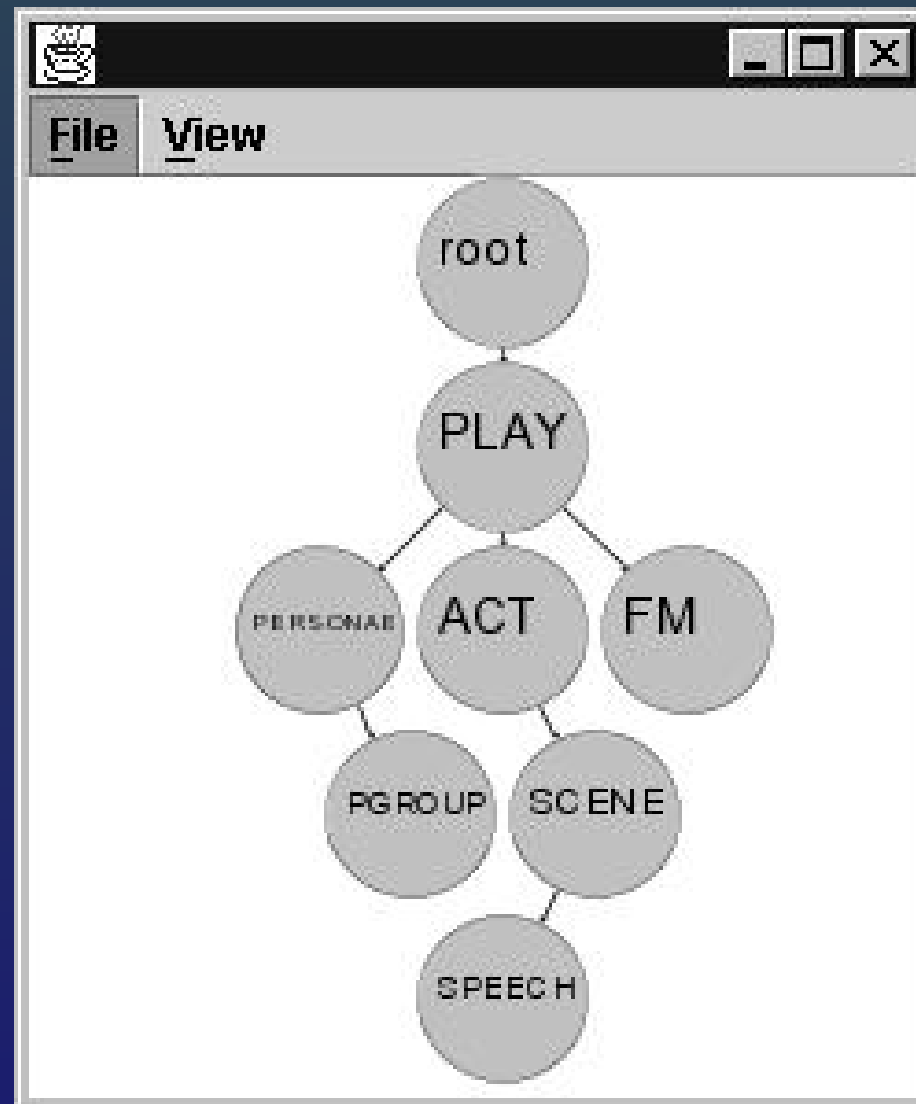
Reading XML

```
<?xml version="1.0"?>
<root>
<PLAY>
<TITLE>The Tragedy of Hamlet, Prince of Denmark</TITLE>
<FM>
<P>ASCII text placed in the public domain by Moby Lexical Tools
<P>SGML markup by Jon Bosak, 1992-1994.</P>
<P>XML version by Jon Bosak, 1996-1999.</P>
</FM>
<PERSONAE>
<TITLE>Dramatis Personae</TITLE>
<PERSONA>CLAUDIUS, king of Denmark. </PERSONA>
<PERSONA>HAMLET, son to the late, and nephew to the present kin
<PERSONA>POLONIUS, lord chamberlain. </PERSONA>
<PERSONA>HORATIO, friend to Hamlet.</PERSONA>
<PERSONA>LAERTES, son to Polonius.</PERSONA>
```

VQBD Dataflow



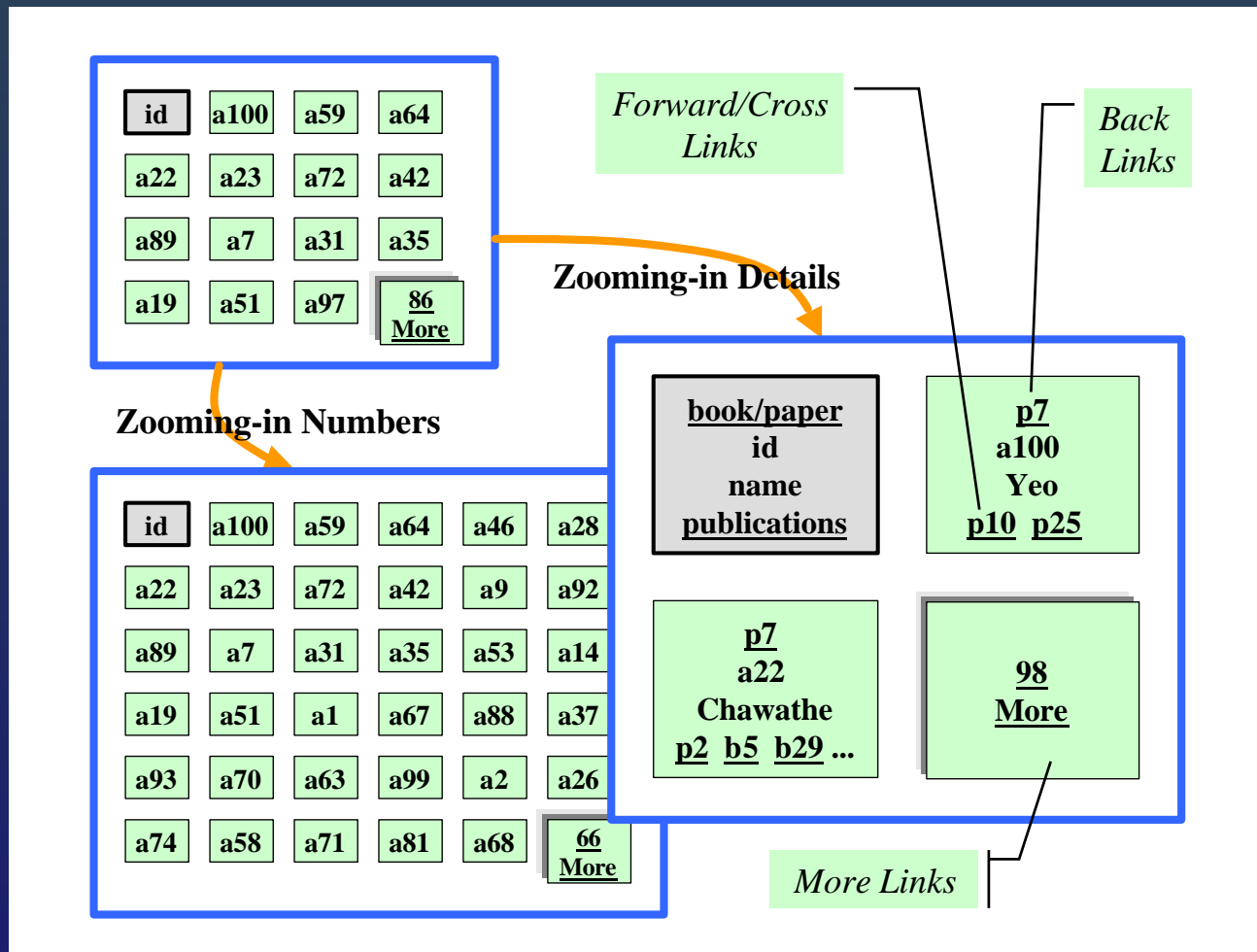
Summary View



Details View

LINE		
	Give him this money and these notes, Reynaldo.	To England or confinement.
Good gentlemen, he hath much talk'd of you;	Welcome, dear Roberto; your errand is welcome.	Madness must needs go.
This physic but prolongs thy sickly days.	If he steal aught that is mine, whilst this play is playing,	And after our judgment,
Of nothing: bring me word of him. Hide fox, and all after.	A thing, my lord!	Shows it a weeper done.
But that this folly doubts it.	Horatio, when thou shalt have overlock'd the door,	He shall please thee with a letter.

Zooming



FuzzyTree

The screenshot shows a window titled "Fuzzy Tree XML Viewer - hamlet.xml". At the top, there are three buttons: "Open New File", "Change Parameters", and "Help". The main area displays a tree structure of XML elements:

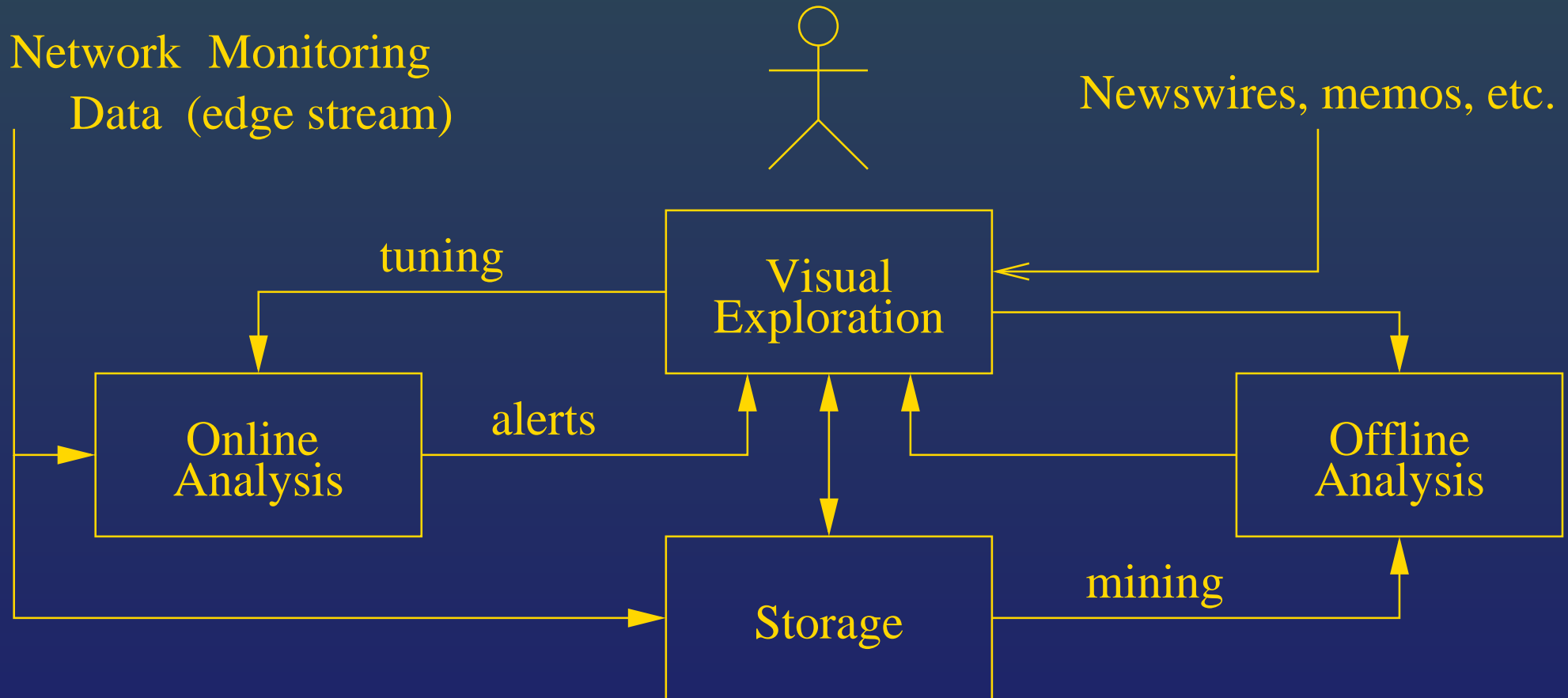
- root
 - PLAY
 - TITLE The Tragedy of Hamlet, Prince of Denmark
 - PERSONAE
 - PERSONA CLAUDIUS, king of Denmark.
 - ACT II
 - STAGEDIR Enter POLONIUS and REYNALDO
 - SPEECH
 - SPEAKER LORD POLONIUS
 - LINE Give him this money and these notes, Reynaldo.
 - SCENE SCENE II (Act II)
 - ACT ACT III
 - SPEECH
 - SPEECH
 - ACT ACT IV
 - SCENE SCENE I
 - SPEECH 2
 - SCENE SCENE III
 - SCENE SCENE IV
 - SCENE

At the bottom, there is a "Displayed Elements" section with a slider ranging from 10 to 60. The slider is currently set to 20. Below the slider, there is a text input field for "Number to display:" with a "Set Value" button. To the right, it says "Displayed: 20 Total: 204".

Selective Display in FuzzyTree

- Each tree node (XML element) is assigned a score.
 - N (slider) highest-scoring nodes displayed.
- Score has two components:
 - Inherent importance of the node (subtree isomorphism).
 - User feedback (GUI interactions).
- Efficient implementation:
 - $O(n \log n)$ preprocessing (n is # of elements).
 - $O(u)$ response (u is user input)

System Architecture



Streaming XML and XPath

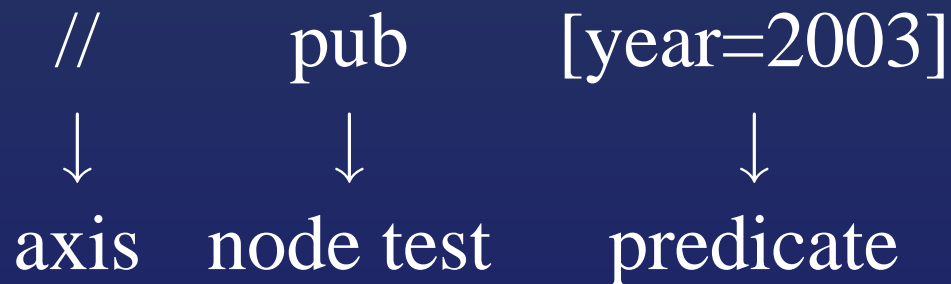
- W3C standard: XPath, XQuery, XSLT

- Location path: a sequence of location steps.

`//pub[year=2003] //book[@id] //author /text{ }`



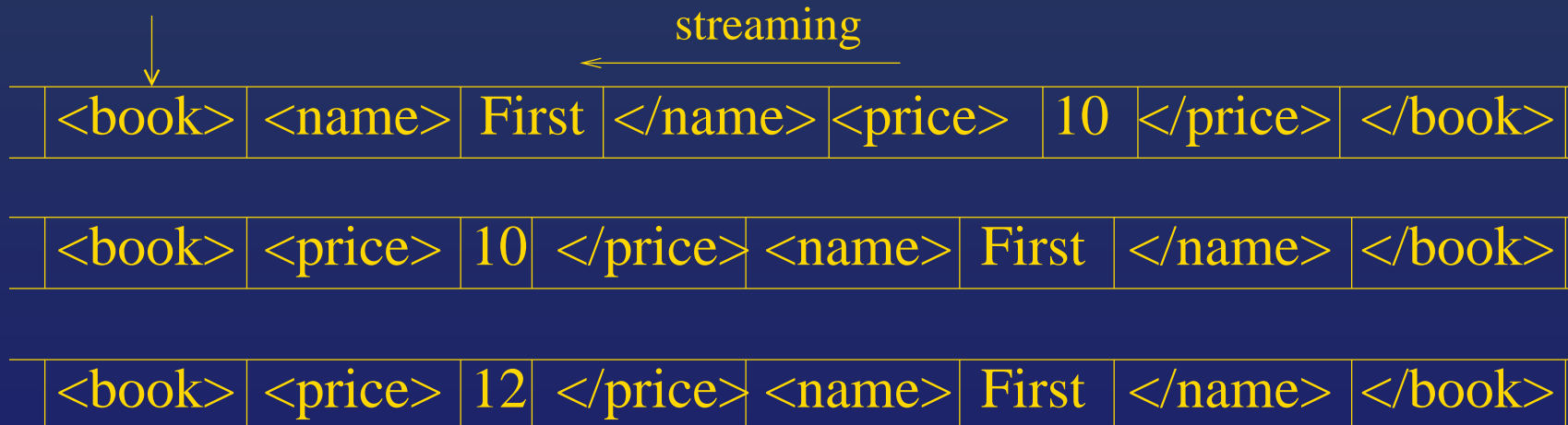
- Location step: axis, node test, predicate(optional)



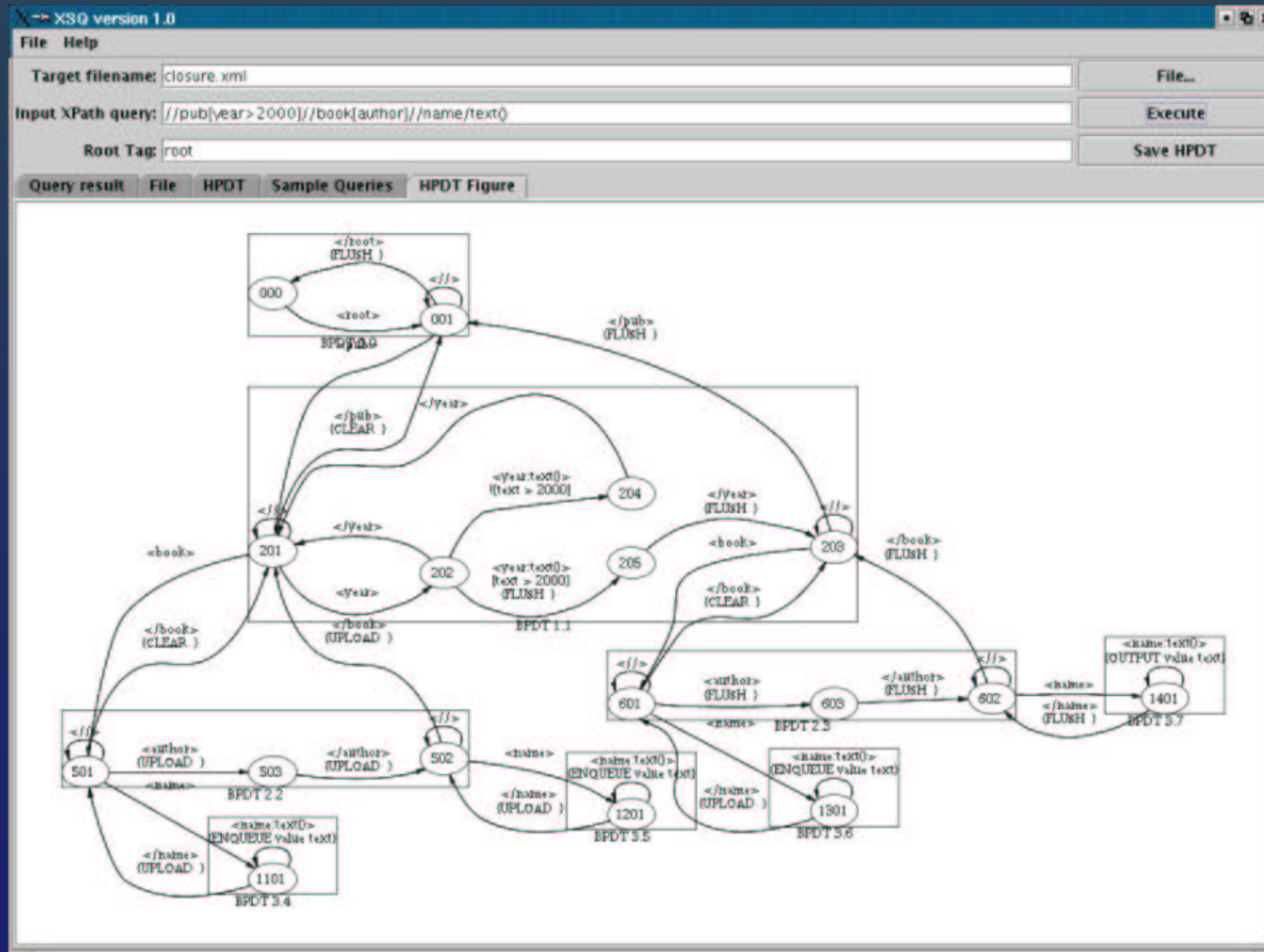
Streaming XPath Evaluation

- Unpredictable sequence of arriving data items.
- Potential result items may arrive before their predicate data.
- Predicates usually not falsified until end of element.

Example: `book[price<11]/name`



XSQ: A Streaming XPath Engine



Windowed Subgraph Similarity

- Match nodes in new and old versions of a call graph.
- New node n' matches old node n if n' communicates with all the partners of n (under the mapping).
- Disregard nodes older than a given version.
- Problem: Graphs too big; change too frequently
- Basic idea: Dynamically maintain a compressed store of node neighborhoods. Lossy compression introduces false positives.

Conclusion

- Problem of tracking agents using nodes.
- Architecture: offline and online analysis, visual exploration, streaming data.
- Modules:
 - SEuS: Finding frequent substructures.
 - VQBD and FuzzyTree: Exploring data interactively (zooming, tree).
 - XSQ: Querying streaming XML data using XPath.
- All freely available (GNU GPL)
 - <http://www.cs.umd.edu/projects/{seus,vqbd,xsq}/>