Querying and Routing
Data-Centric Forays into Networking

Joe Hellerstein
UC Berkeley and Intel Research

Note

- These slides were made on PowerPoint for Mac 2004
- There are incompatibilities between the Mac and Windows versions of PowerPoint, particularly with regard to animations.
- Please email the author with questions.

Road Map

- Emerging synergies in databases and networking
- Internet-Scale Querying: PIER and φ
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- If time permits
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

Background: CS262 Experiment w/ Eric Brewer

- Merge OS & DBMS grad class, over a year
- Eric/Joe, point/counterpoint
- Some tie-ins were obvious:
  - memory mgmt, storage, scheduling, concurrency
- Surprising: QP and networks go well side by side
  - Query processors are dataflow engines.
  - So are routers (e.g. Kohler's CLICK toolkit).
  - Adaptive query techniques look even more like networking idea
    - E.g. “Eddy” tuple routers and TCP Congestion Control
    - Use simple Control/Queuing to “learn”/affect unpredictable dataflows

Networking for DB Dummies (i.e. me)

- Core function of protocols: data xfer
  - Data Manipulation
    - buffer, checksum, encryption, xfer to/fr app space, presentation
  - Transfer Control
    - flow/congestion ctl, detecting xmission probs, acks, muxing, timestamps, framing

    Clark & Tennenhouse, “Architectural Considerations for a New Generation of Protocols”, SIGCOMM ‘90

- Basic Internet assumption:
  - “a network of unknown topology and with an unknown, unknowable and constantly changing population of competing conversations” (Van Jacobson)

C & T’s Wacky Ideas

- Thesis: nets are good at xfer control, not so good at data manipulation
- Some C&T wacky ideas for better data manipulation

Data Modeling!

- Xfer semantic units, not packets (ALF)

Query Opt!

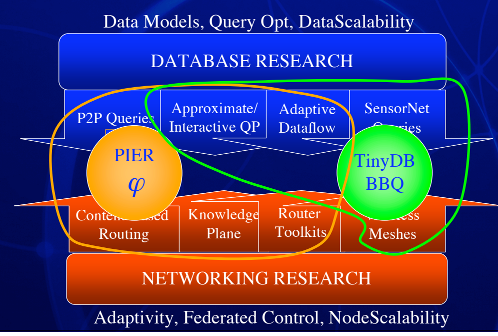
- Auto-rewrite layers to flatten them (ILP)
- Minimize cross-layer ordering constraints
- Control delivery in parallel via packet content

Exchange!

Wacky Ideas in Query Processing

- **What if…**
  - We had unbounded data producers and consumers (“streams” … “continuous queries”)
  - We couldn’t know our producers’ behavior or contents?? (“federation” … “mediators”)
  - We couldn’t predict user behavior? (“CONTROL”)
  - We couldn’t predict behavior of components in the dataflow? (“web services”)
  - We had partial failure as a given? (transactions not possible?)
- **Yes … networking people have been here!**
  - Recall Van Jacobson’s quote

Convergence



Road Map

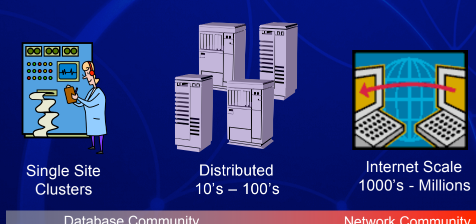
- **Emerging synergies in databases and networking**
- **Internet-Scale Querying: PIER and φ**
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- **If time permits**
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

PIER

- **P2P Information Exchange and Retrieval**
  - An Internet-Scale (peer-to-peer) query engine

Our story at VLDB: What is a Very Large Data Base?

[HHLLSS VLDB 03]



- **Challenge: How to run DB style queries at Internet Scale?!**
- **Challenge: How can DB functionality change the Internet?**

What are the Key Properties?

- **Lots of data that is:**
  - Naturally distributed (where it’s generated)
  - Centralized collection undesirable
  - Homogeneous in schema
  - Data is more useful when viewed as a whole
- **This is the design space we have chosen to investigate.**
  - As opposed to …
    - Enterprise Information Integration
    - Semantic Web
  - Challenges tilted more heavily toward systems/algorithms
    - As opposed to data semantics & cleaning

Who Needs Internet Scale Querying? Example 1: Filenames

- **Simple ubiquitous schemas:**
  - Filenames, Sizes, ID3 tags
- **Early P2P filesharing apps**
  - Napster, Gnutella, KaZaA, etc.
- **Built "in the garage"**
- **"Normal" non-expert users**
- **Not the greatest example**
  - Often used to break copyright
  - Fairly trivial technology
- **But…**
  - Points to key social issues driving adoption of decentralized systems
  - Provide real workloads to validate more complex designs

Example 2: Network Traces

- **Schemas are mostly standardized:**
  - IP, SMTP, HTTP, SNMP log formats, firewall log formats, etc.
- **Network administrators are looking for patterns within their site AND with other sites:**
  - DoS attacks cross administrative boundaries
  - Tracking epidemiology of viruses/worms
  - Timeliness is very helpful
- **Might surprise you just how useful this is:**
  - Network on PlanetLab (distributed research test bed) is mostly filled with people monitoring the network status

Road Map

- **Emerging synergies in databases and networking**
- **Internet-Scale Querying: PIER and φ**
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- **If time permits**
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

φ: Public Health for the Internet

[HPPRSW 04]

- **Thought experiment: A Network Oracle**
  - Queryable entity that knows about all network state
    - Network maps
    - Link loading
    - Point-to-point latencies/bandwidth
    - Event detection (e.g. firewall events)
    - Naming (DNS, ASs, etc.)
    - End-system configuration
    - Router configuration
  - Data from recent past up to near-real-time
  - Available to all end-systems
- **What might this enable?**

Applications of a Network Oracle

- **Performance fault diagnosis**
- **Tracking network attacks**
  - Correlating firewall logs
- **New routing protocols**
  - E.g. app-specific route selection
- **Adaptive distributed applications**
- **"Internet Screensavers"**
  - A la SETI@Home
- **Serendipity!**

Benefits?

- **Short term:**
  - Connect net measurement and security researchers' datasets. Enable distributed queries for network characterization, epidemiology and alerts.
  - E.g. top 10 IP address result from Barford et.al.
- **Medium term:**
  - Provide a service for overlay networks and planetary-scale adaptive applications
  - E.g. feed link measurement results into CDNs, server selection
- **Long term:**
  - Change the Internet: protocols no longer assume ignorance of network state. Push more intelligence into end systems.
  - E.g. Host-based source routing solutions, congestion avoidance (setting timeouts)

A Center for Disease Control?

- **Who owns the Center? What do they Control?**
- **This will be unpopular at best**
  - Electronic privacy for individuals
    - The Internet as "a broadly surveilled police state"?
  - Provider disincentives
    - Transparency = support cost, embarrassment

- **And hard to deliver**
  - Can monitor the chokepoints (ISPs)
  - But inside intranets??
    - E.g. Intel IT
    - E.g. Berkeley dorms
    - E.g. Grassroots WiFi agglomerations?

Energizing the End-Users

- **Endpoints are ubiquitous**
  - Internet, intranet, hotspot
  - Toward a uniform architecture

- **End-users *will* help**
  - Populist appeal to home users is timely
  - Enterprise IT can dictate endpoint software
  - Differentiating incentives for endpoint vendors
- **The connection: peer-to-peer technology**
  - Harnessed to the good!
  - Ease of use
  - Built-in scaling
  - Decentralization of trust and liability

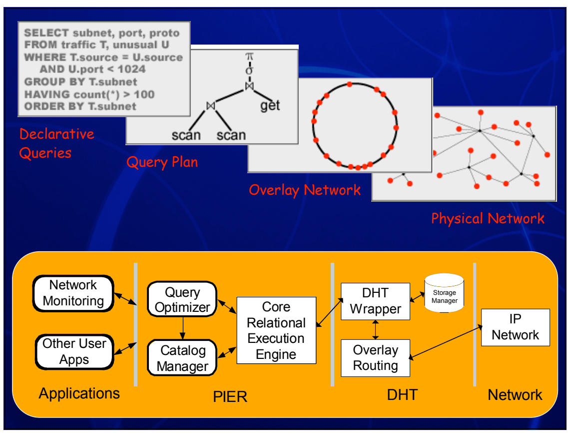
Road Map

- **Emerging synergies in databases and networking**
- **Internet-Scale Querying: PIER and φ**
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- **If time permits**
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

4 Principles for Internet-Scale Querying

- **Relaxed Consistency**
  - ACID transactions not an option
  - We provide best-effort results ("dilated snapshot")
- **Organic Scaling**
  - Applications may start small, without a priori knowledge of size
- **Data in its Natural Habitat**
  - No CREATE TABLE/INSERT
  - No "publish to server"
  - Data must be wrapped at the source
- **Standard Schemas via Grassroots software**
  - Data is produced by widespread software, de-facto schemas
  - Start with data that's easy to homogenize

Coarse Architecture of PIER



Road Map

- **Emerging synergies in databases and networking**
- **Internet-Scale Querying: PIER and φ**
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- **If time permits**
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

Some Background on Overlay Networks

[RH ITR 03]

- **A P2P system like PIER needs to:**
  - Track identities & (IP) addresses of peers currently online
    - May be many!
    - May have significant Churn
    - Best not to have n^2 ID references
  - Route messages among peers
    - If you don't track all peers everywhere, this is "multi-hop"
- **This is an *overlay network***
  - Peers are doing both naming and routing
  - IP becomes "just" the low-level transport
    - All the IP routing is opaque

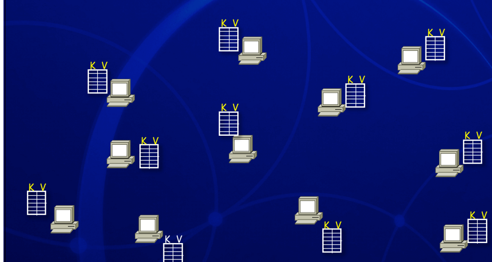
What is a DHT?

- **Hash Table**
  - data structure that maps "keys" to "values"
  - essential building block in software systems
- **Distributed Hash Table (DHT)**
  - similar, but spread across the Internet
- **Interface**
  - insert(key, value)
  - lookup(key)

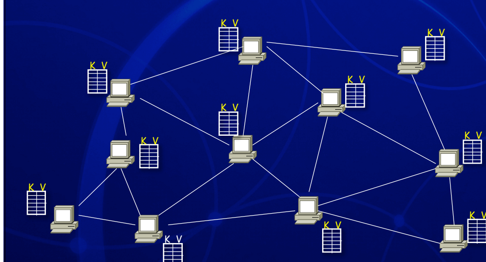
How?

- **Every DHT node supports a single operation:**
  - Given key as input; route messages toward node holding key

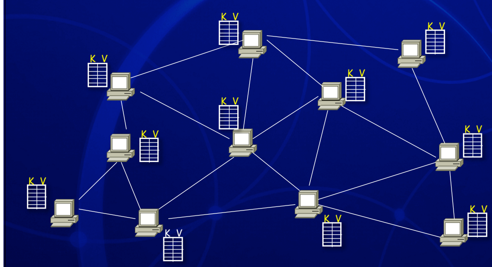
DHT in action



DHT in action

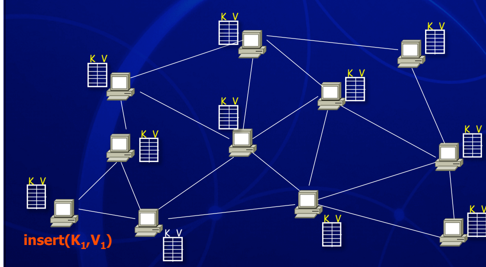


DHT in action



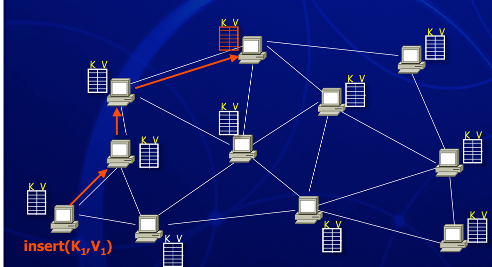
Operation: take *key* as input; route messages to node holding *key*

DHT in action: put()



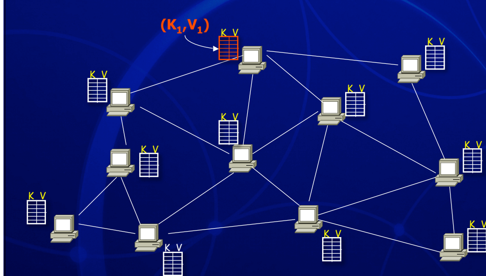
Operation: take *key* as input; route messages to node holding *key*

DHT in action: put()



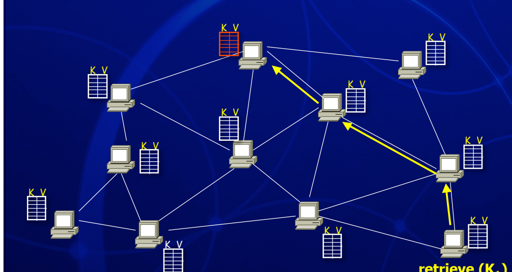
Operation: take *key* as input; route messages to node holding *key*

DHT in action: put()



Operation: take *key* as input; route messages to node holding *key*

DHT in action: get()



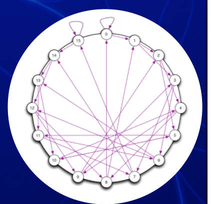
Operation: take *key* as input; route messages to node holding *key*

DHT Design Goals

- **An "overlay" network with:**
  - Flexible mapping of keys to physical nodes
  - Small network diameter
  - Small degree (fanout)
  - Local routing decisions
  - Robustness to churn
  - Routing flexibility
  - Decent locality (low "stretch")
- **A "storage" or "memory" mechanism with**
  - Best-effort persistence (via soft state)

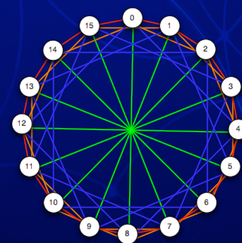
DHT Topologies

- DHTs emulate InterConnect Networks
- These have group-theoretic structure
  - Cayley and Coset graphs
  - Rich families of such graphs with different properties
- We can exploit the structure (i.e. constraints) of the overlay
  - E.g. to embed complex computations with efficient communication
  - E.g. to reason about the "influence" of malicious nodes in the network



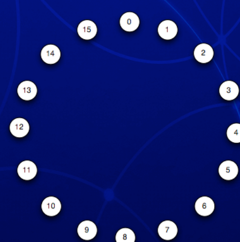
An Example DHT: Chord

- Overlayed 2^k-Gons



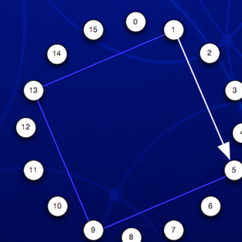
Routing in Chord

- At most one of each Gon
- E.g. 1-to-0



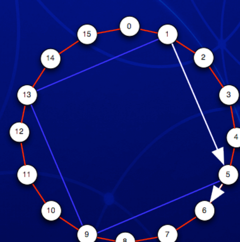
Routing in Chord

- At most one of each Gon
- E.g. 1-to-0



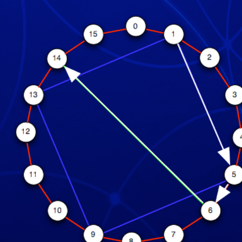
Routing in Chord

- At most one of each Gon
- E.g. 1-to-0



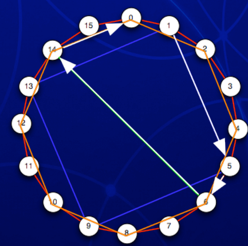
Routing in Chord

- At most one of each Gon
- E.g. 1-to-0



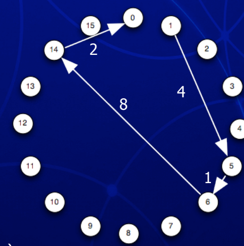
Routing in Chord

- At most one of each Gon
- E.g. 1-to-0



Routing in Chord

- At most one of each Gon
- E.g. 1-to-0
- What happened?
  - We constructed the binary number 15!
  - Routing from *x* to *y* is like computing *y* - *x* mod *n* by summing powers of 2



Diameter: log *n* (1 hop per gon type)
Degree: log *n* (one outlink per gon type)

Deconstructing DHTs

- A DHT is composed of
  - A logical, underlying interconnection network
  - An “emulation scheme”
    - works on a “non-round” #of nodes
    - without global knowledge of network size
  - Self-monitoring components
    - Track and react to churn

Road Map

- Emerging synergies in databases and networking
- Internet-Scale Querying: PIER and φ
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- If time permits
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

DHTs Gave Us Equality Lookups

- That’s a start on database query processing.
- But what else might we want?
  - Range Search
  - Aggregation
  - Group By
  - Join
  - Intelligent Query Dissemination
- Theme
  - All can be built elegantly and opaquely on DHTs!
    - No need to build a “special” DHT for any of these
      - Can leverage advances in DHT design
    - This is the approach we take in PIER

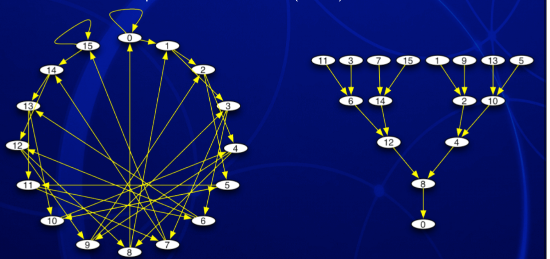
Aggregation in a DHT

- SELECT COUNT(*) FROM firewalls
- One common approach:
  - Everybody routes their firewall records to a particular “collector”
    - This forms a tree
  - Along the way, count up totals
  - At root, form final result
- Note: the shapes of these trees depend on the DHT topology!
  - Can reason about comm costs, sensitivity to failure, influence of malefactors, etc.

binomial tree

Aggregation in Koorde

- **Recall the DeBruijn graph:**
  - Each node x points to $2x$ mod n and $(2x + 1)$ mod n



Grouped Aggregation

- **SELECT COUNT(*)**
  FROM firewalls
  GROUP BY root-domain
- **Everybody routes record *r* to hash(*r*.root-domain)**
  - Simply forms a tree per group

Joins

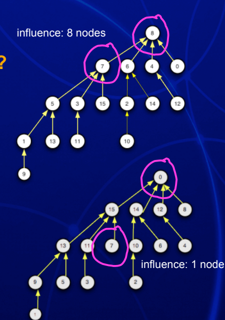
For each of my attackers,
how many sites did they attack,
and how many packets were involved?

```
     SELECT  F.sourceIP,
             COUNT(DISTINCT p.*), COUNT(DISTINCT p.destIP)
       FROM  firewalls F, packets P
      WHERE  F.sourceIP = P.sourceIP
        AND  F.destIP = <myIP>
   GROUP BY  P.sourceIP
```

- **"Rehash" join:**
  - Everybody routes their F and P records to hash(sourceIP)
  - Forms a tree per sourceIP, can combine tuples in each tree independently
  - Automatically parallelizes the join algorithm
    - No notion of parallelism in the code; falls out the DHT
- **Other join algorithms available**
  - "Fetch matches"
  - Semi-join variants
    - Bloom-filter variants

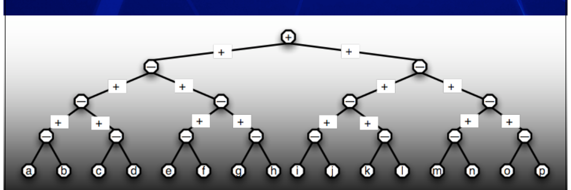
Exploiting Algebraic Topology I

- **Consider malicious aggregators**
- **Identify & limit their influence?**



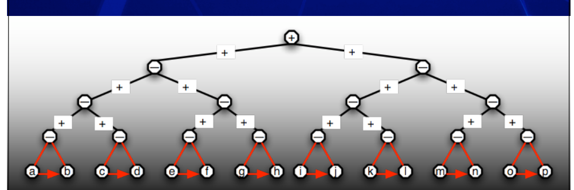
Exploiting Algebraic Topology II

- **Some computations need specific aggregation topologies**
- **Distributed Haar Wavelet**



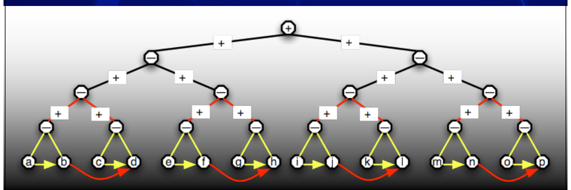
Exploiting Algebraic Topology II

- **Some computations need specific aggregation topologies**
- **Distributed Haar Wavelet**



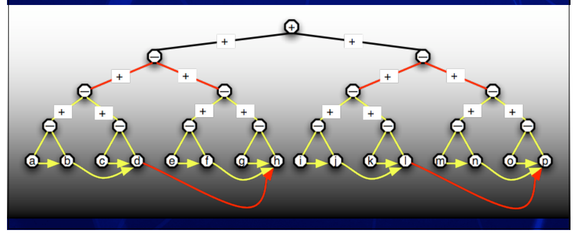
Exploiting Algebraic Topology II

- Some computations need specific aggregation topologies
- Distributed Haar Wavelet



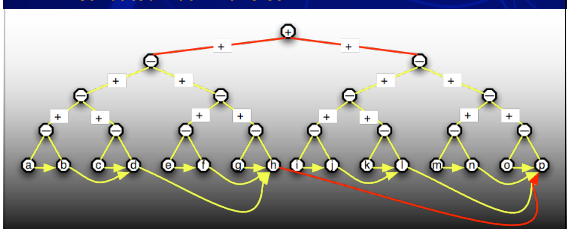
Exploiting Algebraic Topology II

- Some computations need specific aggregation topologies
- Distributed Haar Wavelet



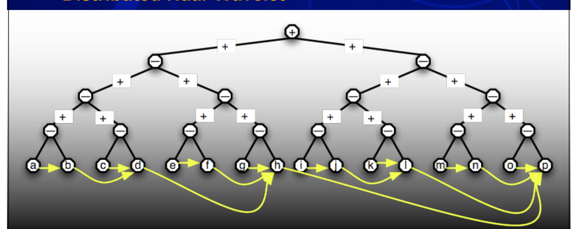
Exploiting Algebraic Topology II

- Some computations need specific aggregation topologies
- Distributed Haar Wavelet



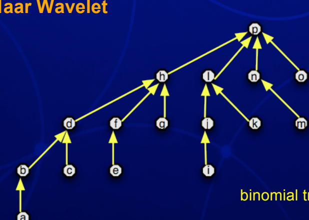
Exploiting Algebraic Topology II

- Some computations need specific aggregation topologies
- Distributed Haar Wavelet



Exploiting Algebraic Topology II

- Some computations need specific aggregation topologies
- Distributed Haar Wavelet



binomial tree

Ephemeral Overlays

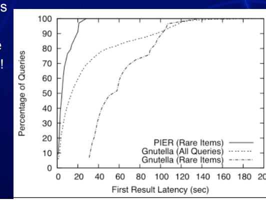
- A new kind of DHT
  - On-Demand overlays for specific computations
  - E.g. for a single operator in a dataflow graph!
- Challenge:
  - Given a DHT that's up and running
  - What's the overhead of constructing a new, appropriate topology among (a subset of) the nodes?
  - How quickly can you re-ID those nodes?
- What is the API
  - When you register an aggregation f'n, what do say about it?
    - E.g. specify the exact agg topology? (bad)
    - E.g. specify some simple algebraic property of the function (better!)
  - This "API definition problem" is where systems and theory really meet?
    - Mathematical abstraction = Engineering abstraction !!

Road Map

- **Emerging synergies in databases and networking**
- **Internet-Scale Querying: PIER and φ**
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- **If time permits**
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

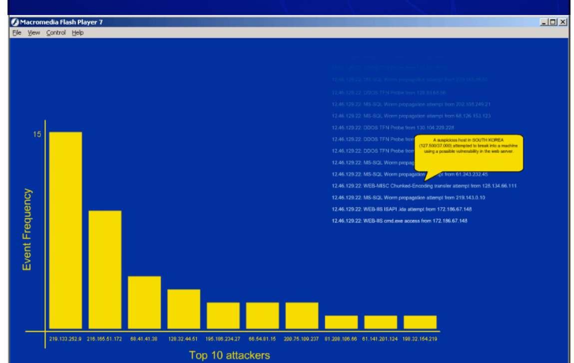
Current PIER Applications (I)

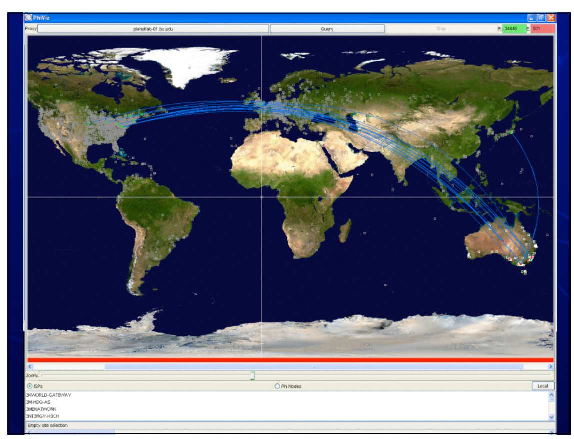
- **Filesharing**
  - Implemented PIERSearch: keyword search over PIER
  - Deployed a hybrid PIERSearch/Gnutella client on PlanetLab
    - Sniffed real Gnutella queries at 50 sites worldwide
  - Results
    - Gnutella is very efficient on popular items
    - PIER *far* better on rare items
      - Both in recall and latency
    - Hybrid solution very tenable
      - Trick: identify rare queries!



Current PIER Applications (II)

- **Engine for φ**





Initial Tidbits from PIER Efforts

[HMR WORLDS 04, HH+ CIDR 04]

- "Multiresolution" simulation critical
  - Native simulator was hugely helpful
  - Emulab allows control over link-level performance
  - PlanetLab is a nice approximation of reality
- Debugging still very hard
  - Need to have a traced execution mode.
    - Radiological dye? Intensive logging?
- DB workloads on NW technology: some mismatches
  - E.g. Bamboo aggressively changes neighbors for single-message resilience/performance
    - Can wreak havoc with stateful aggregation trees
  - E.g. returning results: SELECT * from Firewalls
    - 1 MegaNode of machines want to send you a tuple!
- A relational query processor w/o storage
  - Where's the metadata?

Internet-Scale Querying: Summary

- Query processing on DHT overlays
  - Many traditional querying tasks fall out gracefully
  - Some new opportunities that take advantage of ephemeral overlays
- We're active with two applications
- Major gamble: Network Oracle (φ)
  - Aggregating firewall logs, packet traces, etc.
  - Customizable routing with recursive queries

Parallel Agendas

- Database Agenda
  - Query the Internet?

Be the internet.

- Networks Agenda
  - Network measurement?

Network Oracle.

- Lovely opportunities for synergy here
  - And much research to be done!
- Rallying efforts around an open spec for an Information-Plane/Network-Oracle
  - Rooted in PlanetLab community
  - Data sources, community-building (screensavers?), experimental workloads, applications, protocol definitions, etc.
  - Note: PIER was a prototype *system*
  - Next-gen effort beginning, starting with *protocols*

Acknowledgments

- For specific slides
  - Sylvia Ratnasamy
  - Timothy Roscoe
- Additional Collaborators
  - Ron Avnur, Brent Chun, Tyson Condie, Amol Deshpande, Mike Franklin, Carlos Guestrin, Wei Hong, Ryan Huebsch, Bruce Lo, Boon Thau Loo, Sam Madden, Petros Maniatis, Alexandra Meliou, Vern Paxson, Larry Peterson, Vijayshankar Raman, Raghu Ramakrishnan, David Ratajczak, Sean Rhea, Scott Shenker, Ion Stoica, Nina Taft, David Wetherall

http://pier.cs.berkeley.edu/
http://telegraph.cs.berkeley.edu/tinydb
http://www.cs.berkeley.edu/~jmh

Road Map

- Emerging synergies in databases and networking
- Internet-Scale Querying: PIER and φ
  - Agenda, design space
  - Toward a Network Oracle (φ)
  - The PIER Query Processor
    - Design principles & challenges
    - Overlay Networks: DHTs
    - Query Processing on DHTs
    - PIER in action
- If time permits
  - Routing with queries
  - Related issues in Sensor Networks (TinyDB and BBQ)

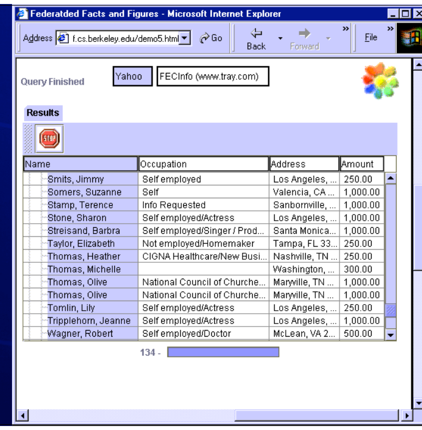
Backup Slides

Adaptive Dataflow Engine

Telegraph
Berkeley Database Research

[CIDR '03]

- **Processing dataflow graphs for unpredictable flows**
  - Unpredictable data properties (sizes, distributions)
  - Unpredictable access/arrival times
- **Originally targeted at querying the "deep web"**
  - Bush/Gore '00 Campaign Finance
- **More recently Continuous Queries over data streams**
  - E.g. packet traces, sensor & RFID reader feeds



Federatded Facts and Figures - Microsoft Internet Explorer

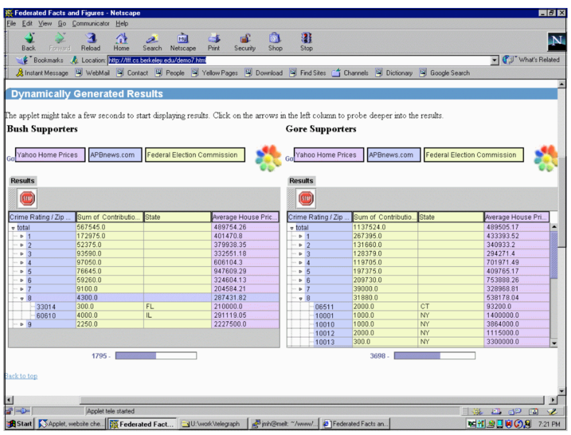
Address f.cs.berkeley.edu/demo5.html Go Back Forward File

Query Finished Yahoo FECInfo (www.tray.com)

Results

Name	Occupation	Address	Amount
Smits, Jimmy	Self employed	Los Angeles, ...	250.00
Somers, Suzanne	Self	Valencia, CA ...	1,000.00
Stamp, Terence	Info Requested	Sanbornville, ...	1,000.00
Stone, Sharon	Self employed/Actress	Los Angeles, ...	1,000.00
Streisand, Barbra	Self employed/Singer / Prod...	Santa Monica...	1,000.00
Taylor, Elizabeth	Not employed/Homemaker	Tampa, FL 33...	250.00
Thomas, Heather	CIGNA Healthcare/New Busi...	Nashville, TN ...	250.00
Thomas, Michelle		Washington, ...	300.00
Thomas, Olive	National Council of Churche...	Maryville, TN ...	1,000.00
Thomas, Olive	National Council of Churche...	Maryville, TN ...	1,000.00
Tomlin, Lily	Self employed/Actress	Los Angeles, ...	250.00
Tripplehorn, Jeanne	Self employed/Actress	Los Angeles, ...	1,000.00
Wagner, Robert	Self employed/Doctor	McLean, VA 2...	500.00

134 -



Federated Facts and Figures - Netscape

Dynamically Generated Results

The applet might take a few seconds to start displaying results. Click on the arrows in the left column to probe deeper into the results.

Bush Supporters

Yahoo Home Prices APBnews.com Federal Election Commission

Results

Crime Rating / Zip ...	Sum of Contributio...	State	Average House Pric...
total	567545.0		489754.26
1	172975.0		401470.8
2	52375.0		379938.35
3	93590.0		332551.18
4	97050.0		606104.3
5	76645.0		947609.29
6	59260.0		324604.13
7	9100.0		204584.21
8	4300.0		287431.82
33014	300.0	FL	210000.0
60610	4000.0	IL	291119.05
9	2250.0		2227500.0

1795 -

Gore Supporters

Yahoo Home Prices APBnews.com Federal Election Commission

Results

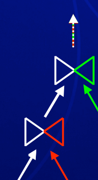
Crime Rating / Zip ...	Sum of Contributio...	State	Average House Pri...
total	1137524.0		489505.17
1	267395.0		433393.52
2	131660.0		340933.2
3	128379.0		294271.4
4	119705.0		701971.49
5	197375.0		409765.17
6	209730.0		753888.26
7	39000.0		328968.81
8	31880.0		538178.04
06511	2000.0	CT	93200.0
10001	1000.0	NY	1400000.0
10010	1000.0	NY	3864000.0
10012	2000.0	NY	1115000.0
10013	300.0	NY	3300000.0

3698 -

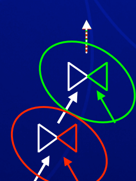
Back to top

Applet tele started

One Challenge in Adaptive Dataflow: Operator Ordering



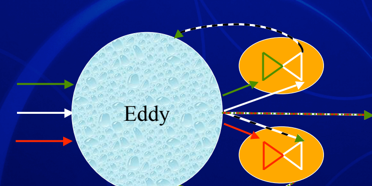
One Challenge in Adaptive Dataflow: Operator Ordering



- **Deal with pipelines of commutative operators**
- **Adapt at very fine granularity**
  - On a tuple-by-tuple basis?
  - *Regional* properties of the data!

Continuous Adaptivity: Eddies

[AH SIGMOD 00, RH SIGMOD 02, MSH SIGMOD 02, RDH ICDE 03, DH VLDB 04]



- **A little more state per tuple**
  - Ready/done bits
- **Routers, not flowgraphs**
  - Query processing = dataflow routing!!
  - Router is adaptive, observing results of prior decisions
  - A recurring theme

Eddies: Two Key Observations

- **Break the set-oriented boundary**
  - Usual DB model: algebra expressions: $(R \bowtie S) \bowtie T$
  - Reasoning about *operators*, not data!
- **Don't re-wire graph. Impose a router.**
  - Any graph can be achieved
  - Router can observe operator consumption/production rates
    - Consumption rate: cost
    - Production: cost * selectivity

Road Map

- **How I got myself into this**
  - CONTROL project
  - Telegraph
- **Connections to Networking**
- **Two arenas over the past few years**
  - Internet: PIER $\Rightarrow \varphi$
  - Sensor networks: TinyDB & BBQ

Coincidence: Eddie Comes to Berkeley

- **CLICK: a NW router is a dataflow engine!**
  - "The Click Modular Router", Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek, SOSP '99

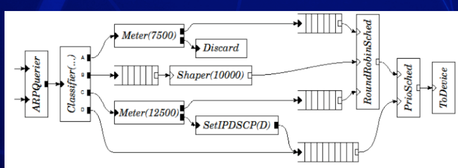


Fig. 15. A sample traffic conditioning block. *Meters* and *Shapers* measure traffic rates in packets per second. A, B, C, and D represent DSCP values.

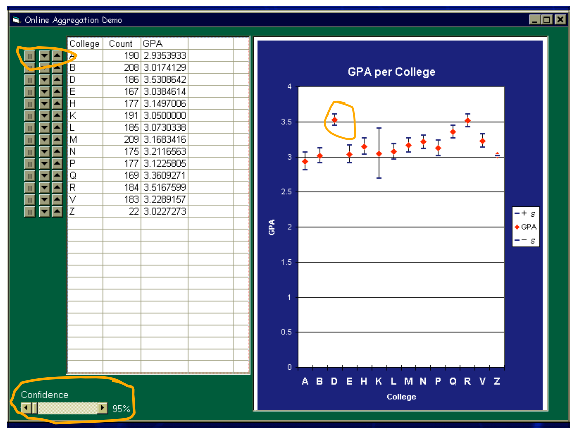
Background: CONTROL

[IEEE Computer 8/99, DMKD 3/2000]

- Continuous Output, Navigation and Transformation with Refinement On Line
  - *Interactive Systems* for long-running data processing
- **Based on**
  - Streaming samples
  - Reactive, pipelining operators
  - Statistical methods
    - approximate queries
    - pattern detection
    - outlier detection
  - Academic & commercial implementation
    - Postgres $\Rightarrow$ Informix
    - Potter's Wheel $\Rightarrow$ PeopleSoft

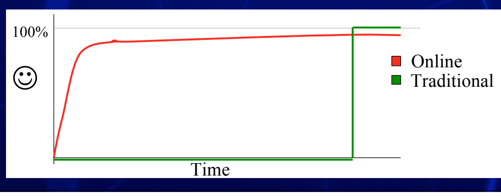
Example: Online Aggregation

[HHW SIGMOD 97, HH SIGMOD 99]



Goals for Online Processing

- **Performance metric:** ☺
  - Statistical (e.g. conf. intervals)
  - User-driven (e.g. weighted by widgets)
- **"Greedy" regime**
  - Maximize 1st derivative of ☺
  - ☺ defined on-the-fly
    ⇒ Feedback & CONTROL

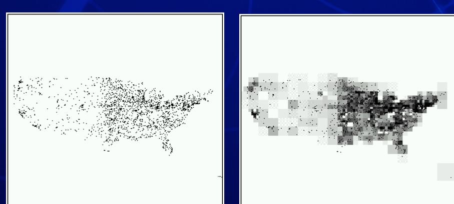


Themes

- **Real-time interaction with streaming data**
  - In this case, streaming samples coming off disks
- **Interactivity ⇒ Unpredictability**
  - Statistical properties
  - User interaction
  - Parameterized on *regions* of the data
- **Followup challenge:**
  - A reusable infrastructure (single-site) for adaptive dataflow

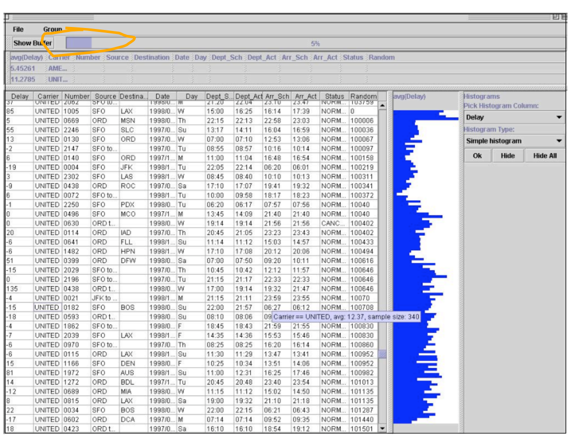
Example: Online Data Visualization

- **CLOUDS**



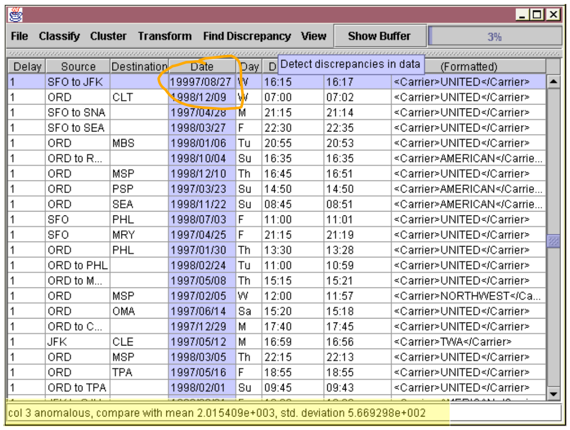
Example: Scalable Spreadsheets

[RCH DMKD 99]



Example: Potter's Wheel

[RH VLDB 01]



File	Classify	Cluster	Transform	Find Discrepancy	View	Show Buffer	3%

Detect discrepancies in data

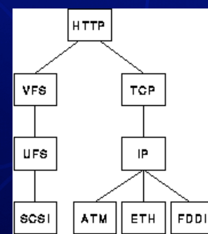
Delay	Source	Destination	Date	Day	D		(Formatted)
1	SFO to JFK		19997/08/27	W	16:15	16:17	<Carrier>UNITED</Carrier>
1	ORD	CLT	1998/12/09	W	07:00	07:02	<Carrier>UNITED</Carrier>
1	SFO to SNA		1997/04/28	M	21:15	21:14	<Carrier>UNITED</Carrier>
1	SFO to SEA		1998/03/27	F	22:30	22:35	<Carrier>UNITED</Carrier>
1	ORD	MBS	1998/01/06	Tu	20:55	20:53	<Carrier>UNITED</Carrier>
1	ORD to R...		1998/10/04	Su	16:35	16:35	<Carrier>AMERICAN</Carrie...
1	ORD	MSP	1998/12/10	Th	16:45	16:51	<Carrier>UNITED</Carrier>
1	ORD	PSP	1997/03/23	Su	14:50	14:50	<Carrier>AMERICAN</Carrie...
1	ORD	SEA	1998/11/22	Su	08:45	08:51	<Carrier>AMERICAN</Carrie...
1	SFO	PHL	1998/07/03	F	11:00	11:01	<Carrier>UNITED</Carrier>
1	SFO	MRY	1997/04/25	F	21:15	21:19	<Carrier>UNITED</Carrier>
1	ORD	PHL	1997/01/30	Th	13:30	13:28	<Carrier>UNITED</Carrier>
1	ORD to PHL		1998/02/24	Tu	11:00	10:59	<Carrier>UNITED</Carrier>
1	ORD to M...		1997/05/08	Th	15:15	15:21	<Carrier>UNITED</Carrier>
1	ORD	MSP	1997/02/05	W	12:00	11:57	<Carrier>NORTHWEST</Ca...
1	ORD	OMA	1997/06/14	Sa	15:20	15:18	<Carrier>UNITED</Carrier>
1	ORD to C...		1997/12/29	M	17:40	17:45	<Carrier>UNITED</Carrier>
1	JFK	CLE	1997/05/12	M	16:59	16:56	<Carrier>TWA</Carrier>
1	ORD	MSP	1998/03/05	Th	22:15	22:13	<Carrier>UNITED</Carrier>
1	ORD	TPA	1997/05/16	F	18:55	18:55	<Carrier>UNITED</Carrier>
1	ORD to TPA		1998/02/01	Su	09:45	09:43	<Carrier>UNITED</Carrier>

col 3 anomalous, compare with mean 2.015409e+003, std. deviation 5.669298e+002

Also Scout

Dataflow Paths key to comm-centric OS

- "Making Paths Explicit in the Scout Operating System", David Mosberger and Larry L. Peterson. OSDI '96.

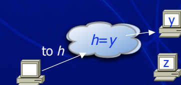


Why Now?

- **The social case (see previous slide)**
- **Technology trends**
  - Conjecture: Net "behavior metadata" grows slower than data
    - Data volume scales with capacity
    - Descriptions of behavior scale with # of end-systems
  - Ample processing power at end-points
    - End-systems have plenty of spare CPU cycles to "think" about traffic
    - This is a differentiation (value-add) opportunity for endpoint vendors
      - HW, OS, Apps
  - Maturation of p2p technologies
    - A Networking/DB nexus!

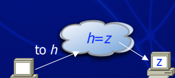
High-Level Idea: Indirection

- **Indirection in space**
  - Logical (content-based) IDs, routing to those IDs
    - "Content-addressable" network
  - Tolerant of *churn*
    - nodes joining and leaving the network



High-Level Idea: Indirection

- **Indirection in space**
  - Logical (content-based) IDs, routing to those IDs
    - "Content-addressable" network
  - Tolerant of *churn*
    - nodes joining and leaving the network



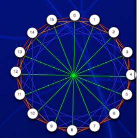
- **Indirection in time**
  - Want some scheme to temporally decouple send and receive
  - Persistence required. Typical Internet solution: soft state
    - Combo of persistence via *storage* and via *retry*
      - "Publisher" requests TTL on storage
      - Republishes as needed
- **Metaphor: Distributed Hash Table**

What is happening here? Algebra!

- **Underlying group-theoretic structure**
  - Recall a *group* is a set S and an operator • such that:
    - S is closed under •
    - Associativity: $(AB)C = A(BC)$
    - There is an *identity* element $I \in S$ s.t. $IX = XI = X$ for all $X \in S$
    - There is an inverse $X^{-1} \in S$ for each element $X \in S$ s.t. $XX^{-1} = X^{-1}X = I$
- **The *generators* of a group**
  - Elements $\{g_1, \ldots, g_n\}$ s.t. application of the operator on the generators produces all the members of the group.
- **Canonical example: $(Z_n, +)$**
  - Identity is 0
  - A set of generators: {1}
  - A different set of generators: {2, 3}

Cayley Graphs

- **The *Cayley Graph (S, E)* of a group:**
  - Vertices corresponding to the underlying set S
  - Edges corresponding to the *actions of the generators*
- **(Complete) Chord is a Cayley graph for (Z_n,+)**
  - $S = Z \bmod n$ ($n = 2^k$).
  - Generators {1, 2, 4, …, 2^{k-1}}
  - That's what the gons are all about!
- **Fact: Most (complete) DHTs are Cayley graphs**
  - And they didn't even know it!
  - Follows from parallel InterConnect Networks (ICNs)
    - Shown to be group-theoretic [Akers/Krishnamurthy '89]



Note: the ones that aren't Cayley Graphs are *coset graphs*, a related group-theoretic structure

Range Search

- **Numerous proposals in recent years**
  - Chord w/o hashing, + load-balancing [Karger/Ruhl SPAA '04, Ganesan/Bawa VLDB '04]
  - Mercury [Bharambe, et al. SIGCOMM '04]. Specialized "small-world" DHT.
  - P-tree [Crainiceanu et al. WebDB '04]. A "wrapped" B-tree variant.
  - P-Grid [Aberer, CoopIS '01]. A distributed trie with random links.
  - (Apologies if I missed your favorite!)
- **We'll do a very simple, elegant scheme here**
  - Prefix Hash Tree (PHT). [Ratnasamy, et al '04]
  - Works over *any* DHT
  - Simple robustness to failure
  - Hints at generic idea: *direct-addressed distributed data structures*

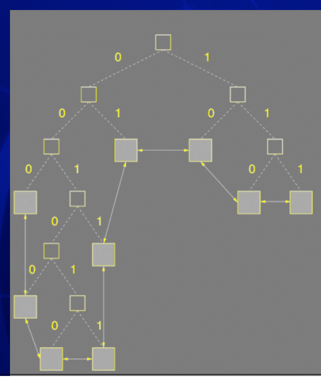
Prefix Hash Tree (PHT)

- **Recall the *trie* (assume binary trie for now)**
  - Binary tree structure with edges labeled 0 and 1
  - Path from root to leaf is a *prefix* bit-string
  - A key is stored at the minimum-distinguishing prefix (depth)
- **PHT is a bucket-based trie addressed via a DHT**
  - Modify trie to allow *b* items per leaf "bucket" before a split
  - Store contents of leaf bucket at DHT address corresponding to prefix
    - So far, not unlike Litwin's "Trie Hashing" scheme, but hashed on a DHT.
    - Punchline in a moment…

PHT

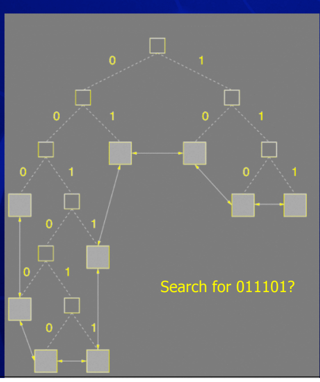
Logical Trie

DHT Content



PHT

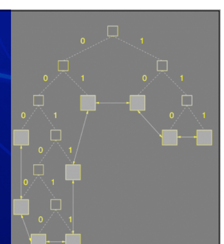
Logical Trie



DHT Contents

Leaf nodes	Keys
000*	000001 000100 000100
00100*	001001
001010*	001010 001010 001010
001011*	001011 001011
0011*	
01*	010000 010101
10*	100010 101011 101111
110*	110000 110010 110011 110110
111*	111000 111010

PHT Search



- **Observe: The DHT allows *direct addressing* of PHT nodes**
  - Can *jump* into the PHT at any node
    - Internal, leaf, or *below a leaf!*
  - So, can find leaf by binary search
    - *loglog* |*D*| search cost!
    - If you knew (roughly) the data distribution,
  - Moreover, consider a failed machine in the system
    - Equals a failed node of the trie
    - Can "hop over" failed nodes directly!
  - And… consider concurrency control
    - A link-free data structure: simple!

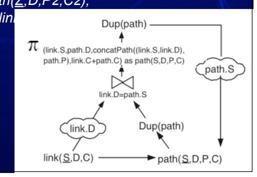
Reusable Lessons from PHTs

- **Direct-addressing a lovely way to emulate robust, efficient “linked” data structures in the network**
- **Direct-addressing requires regularity in the data space partitioning**
  - *E.g.* works for regular space-partitioning indexes (tries, quad trees)
  - Not so simple for data-partitioning (B-trees, R-trees) or irregular space partitioning (kd-trees)

Another Emerging PIER Application

- **Custom Route Constru**
- **Key building block: re**
  - Consider a distributed rou
    link(source, destination,
  - Route construction can easily _ _pressed as recursive queries
    - *path(S,D,P,C) :- link(S,D,C),*
      P = concatPath(link(S,D,C), nil).
    - *path(S,D,P,C) :- link(S,Z,C1), path(Z,D,P2,C2),*
      P = concatPath(lin
    - *Query: path(S,D,P,C).*

Find me all pairs of reachable nodes and paths between them



Minor Variants Give Lots of Options

- ***“Best-Path” Routing***
  - *path(S,D,P,C) :- link(S,D,C),*
    P = concatPath(link(S,D,C), nil).
  - *path(S,D,P,C) :- link(S,Z,C1), path(Z,D,P2,C2),*
    P = concatPath(link(S,Z,C1),P2), C=C1 op C2.
  - *bestPathCost(S,D,AGG<C>) :- path(S,D,P,C).*
  - *bestPath(S,D,P,C) :- bestPathCost(S,D,C), path(S,D,P,C).*
  - *Query: bestPath(S,D,P,C).*
- *Agg* and *op* chosen depending on metric *C*

Minor Variants Give Lots of Options

- ***“Policy-Based” Routing***
  - *path(S,D,P,C) :- link(S,D,C),*
    P = concatPath(link(S,D,C), nil).
  - *path(S,D,P,C) :- link(S,Z,C1), path(Z,D,P2,C2),*
    P = concatPath(link(S,Z,C1),P2), C=C1 + C2.
  - *permitPath(S,D,P,C) :- path(S,D,P,C), excludeNode(S,W),*
    ¬inPath(P,W).
  - *Query: permitPath(S,D,P,C).*

Minor Variants Give Lots of Options

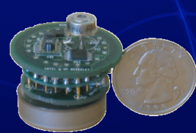
- ***Distance Vector Protocol***
  - *path(S,D,D,C) :- link(S,D,C),*
    P = concatPath(link(S,D,C), nil).
  - *path(S,D,Z,C) :- link(S,Z,C1), path(Z,D,P2,C2),*
    P = concatPath(link(S,Z,C1),P2), C=C1 +C2.
  - *shortestLength(S,D,min<C>) :- path(S,D,Z,C)*
  - *nextHop(S,D,Z,C) :- path(S,D,Z,C), shortestLength(S,D,C)*
  - *Query: nextHop(S,D,Z,C).*

Minor Variants Give Lots of Options

- ***Dynamic Source Routing***
  - *path(S,D,P,C) :- link(S,D,C),*
    P = concatPath(link(S,D,C), nil).
  - *path(S,D,P,C) :- path(S,Z,P1,C1), link(Z,D,C2),*
    P = concatPath(P1, link(Z,D,C2)), C=C1 +C2.
  - *Query: path(N,M,P,C).*
- Uses “left recursion”

Sensor networks

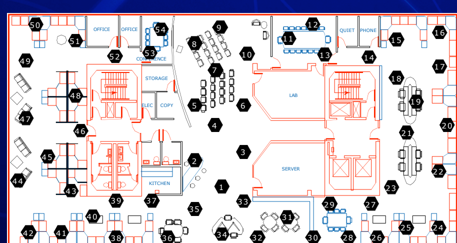
- A collection of devices that can sense, compute, and communicate over a wireless network
- Sensors for temperature, humidity, pressure, sound, magnetic fields, acceleration, visible and ultraviolet light, etc.
- Available resources
  - 4 MHz, 8 bit CPU
  - 40 Kbps wireless
  - 3V battery (lasts days or months)



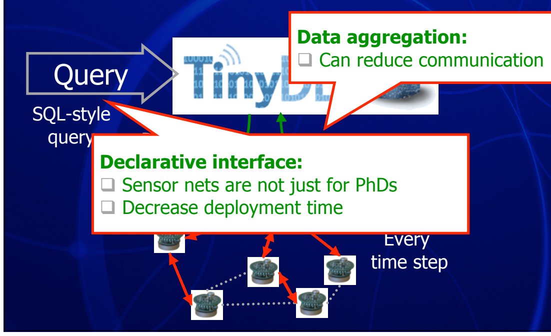
Real deployments

- Great Duck Island
- Redwoods
- Precision agriculture
- Fabrication monitoring

Example: Intel Berkeley Lab deployment



Analogy: SensorNet as a Database

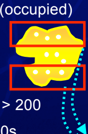


TinySQL Examples

1 SELECT AVG(sound)
FROM sensors
EPOCH DURATION 10s

"Count the number occupied nests in each loud region of the island."

2 SELECT region, CNT(occupied) AVG(sound)
FROM sensors
GROUP BY region
HAVING AVG(sound) > 200
EPOCH DURATION 10s

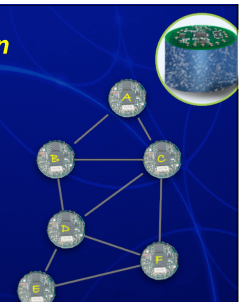


Epoch	region	CNT(…)	AVG(…)
0	North	3	360
0	South	3	520
1	North	3	370
1	South	3	520

Regions w/ AVG(sound) > 200

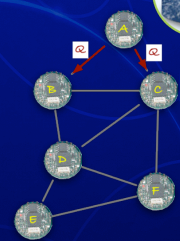
TinyDB execution pattern

- "flood" query to all nodes
  - a tree is formed based on arrival pattern
- periodically communicate up the tree
  - data reduction opportunities
  - tree reconfigures itself online



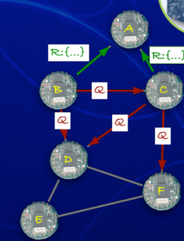
TinyDB execution pattern

- **“flood” query to all nodes**
  - a tree is formed based on arrival pattern
- **periodically communicate up the tree**
  - data reduction opportunities
  - tree reconfigures itself online



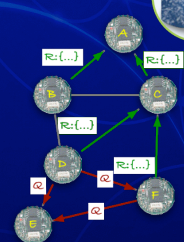
TinyDB execution pattern

- **“flood” query to all nodes**
  - a tree is formed based on arrival pattern
- **periodically communicate up the tree**
  - data reduction opportunities
  - tree reconfigures itself online



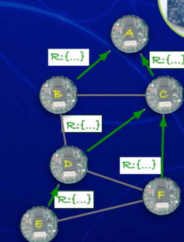
TinyDB execution pattern

- **“flood” query to all nodes**
  - a tree is formed based on arrival pattern
- **periodically communicate up the tree**
  - data reduction opportunities
  - tree reconfigures itself online



TinyDB execution pattern

- **“flood” query to all nodes**
  - a tree is formed based on arrival pattern
- **periodically communicate up the tree**
  - data reduction opportunities
  - tree reconfigures itself online



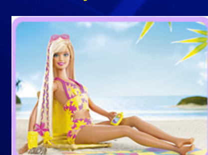
Taxonomy of Aggregates

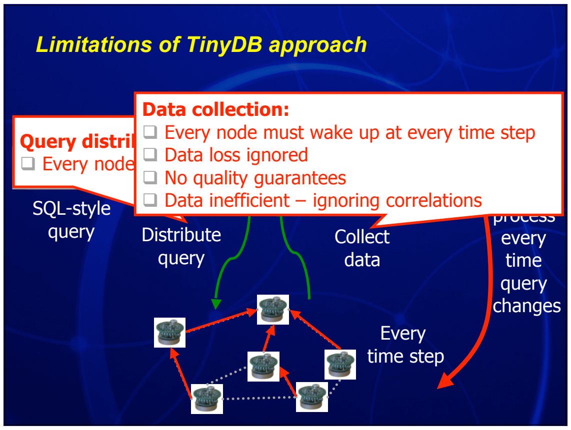
- **TAG insight: classify aggregates according to various functional properties**
  - Yields a general set of optimizations that can automatically be applied
  - Drives an extensibility API to register new aggregates, get them optimized

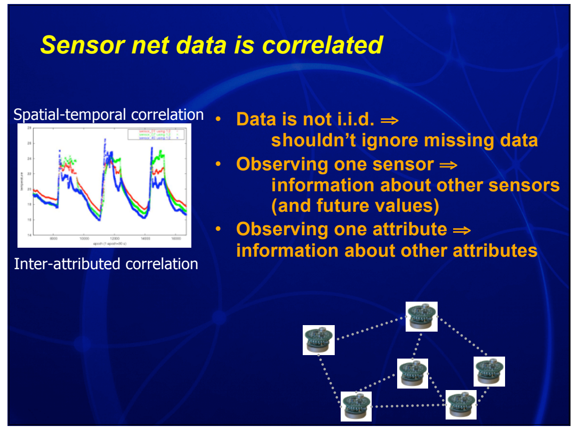
Property	Examples	Affects
Partial State	MEDIAN : unbounded, MAX : 1 record	Effectiveness of TAG
Monotonicity	COUNT : monotonic AVG : non-monotonic	Hypothesis Testing, Snooping
Exemplary vs. Summary	MAX : exemplary COUNT: summary	Applicability of Sampling, Effect of Loss
Duplicate Sensitivity	MIN : dup. insensitive, AVG : dup. sensitive	Routing Redundancy

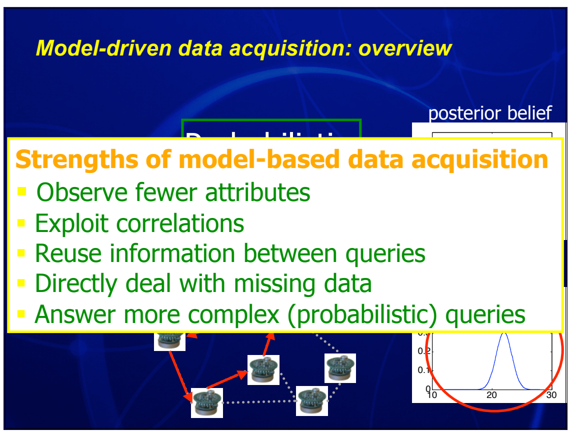
An Alternative Approach

- **BBQ: Model-Drien Data Acquisition for SensorNets**
  - **Tiny Model**-Driven **Q**ueries




Limitations of TinyDB approach
Data collection:
Every node must wake up at every time step
Data loss ignored
No quality guarantees
Data inefficient – ignoring correlations
Query distri
Every node
SQL-style query
Distribute query
Collect data
process every time query changes
Every time step


Sensor net data is correlated
Spatial-temporal correlation
Inter-attributed correlation
Data is not i.i.d. ⇒ shouldn't ignore missing data
Observing one sensor ⇒ information about other sensors (and future values)
Observing one attribute ⇒ information about other attributes


Model-driven data acquisition: overview
posterior belief
Strengths of model-based data acquisition
Observe fewer attributes
Exploit correlations
Reuse information between queries
Directly deal with missing data
Answer more complex (probabilistic) queries
10
20
30