# New Approaches to Distributed Management to Achieve Scalability and Robustness

*Rolf Stadler*

*Laboratory for Communication Networks*
*KTH Royal Institute of Technology*

*Stockholm, Sweden*

19th NMRG Meeting, Stockholm, Sweden, Jan 12-13, 2006

# Drivers and Enablers for a new Approach

- Drivers
  - increase scalability and robustness
  - decrease reaction times, reduce monitoring data
  - support autonomic operation
- Enablers
  - increasing processing power and memory at low cost in networked devices
  - advances in related disciplines: DHTs, data aggregation, distributed control, game theory, …

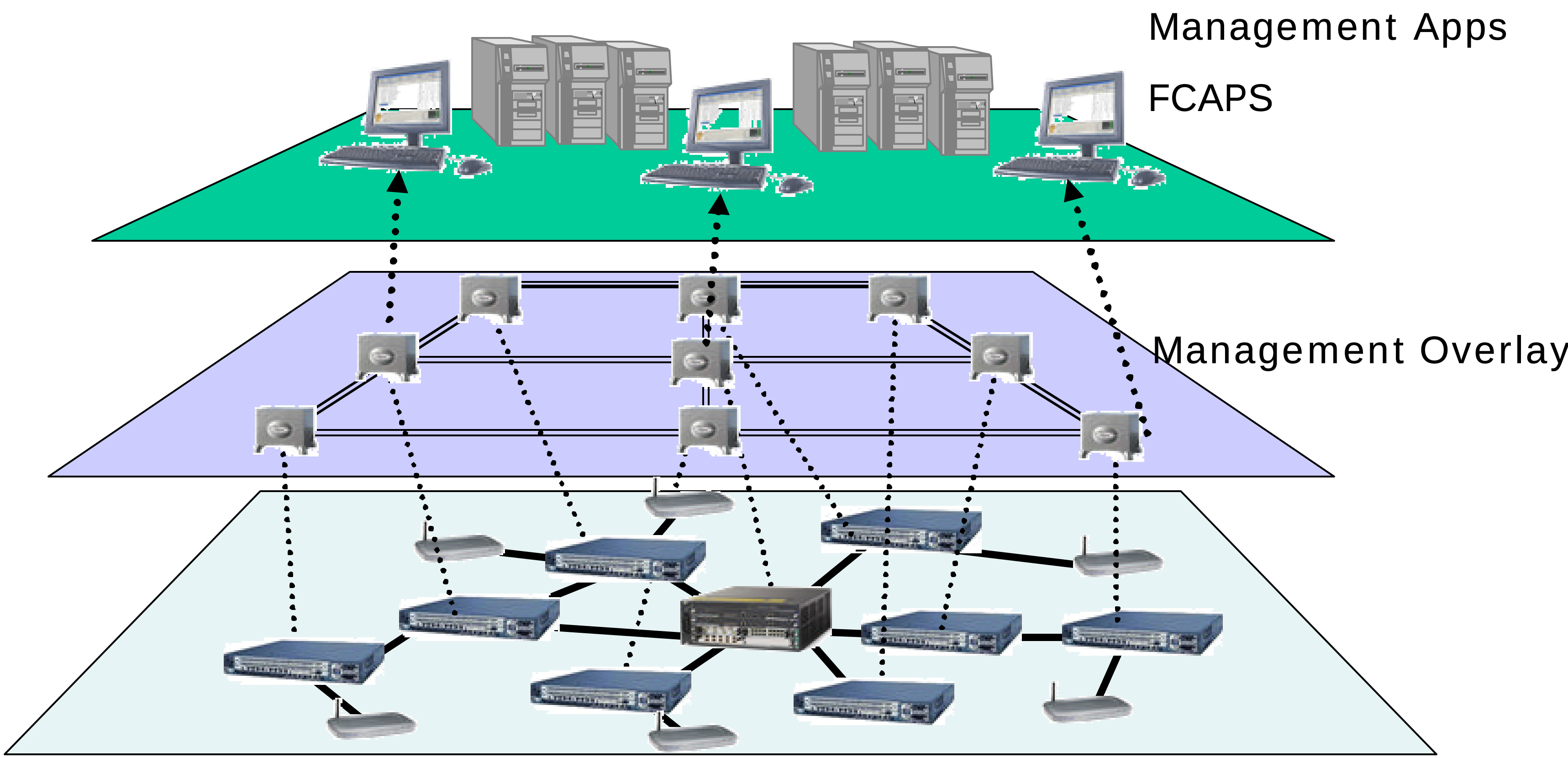
# Why did Earlier Approaches fail? Example: Mobile Agents

- Distributed AI could not deliver:
  - attempts to address QoS, interdomain negotiations, failure detection and repair, etc.)
  - solutions came from traditional fields (traffic engineering, algorithms, event correlation, etc.)
- Research community developed platforms, instead of management applications
- Some enablers missing at that time

# The Emerging P2P Management Architecture

- Management application layer (old)
- Management overlay (new)
  - allows for communication between management nodes
  - is self-organizing
  - handles node arrivals, departures, failures
    (-->robustness)
  - decentralized processing of management operations
    locality of operations,
    (-->scalability)
  - enables “management by exception”
- Network elements (old)

# A New Management Layer: The Management Overlay



# Exploring the Design Space of the Management Overlay

- Fundamental design choices
  - Where it is *located* ? (“inside” the network, “outside” the network)
  - How does it *interact* with management applications and network elements?
  - Which *functionality* does it provide to management applications?
  - (BTW, is management overlay the right term?)
- Choices in constructing and maintaining topology
  - Following the *physical topology*
  - Constructing overlays with *specific topological properties* (random graph, given, connectivity, etc.)
    - DHTs (Pastry, Chord, …)
    - epidemic algorithms (Newscast, Cyclone, …)
  - Using *context* (different physical paths, distance functions, specialized resources, …)

# Design Choices made for KTH Work

- Management nodes are of *identical functionality*.
- *Every node is access point* for management application.
- Result of an operation is independent of access point that initiates it. (if delays neglected)

Functionality of the overlay

- Executes *generic algorithms* for monitoring and control instantiated at run-time.
  (we call them *management patterns*.)
- Realized as a *thin layer* for deploying and running distributed algorithms.
  (Weaver, [Adam Lim Stadler 05])

# Use of Management Overlay: Aggregation of Device-level Data

- ***aggregation functions***: SUM, AVERAGE, COUNT, MAX, MIN, histograms, relational queries
- possible approaches to compute aggregation functions
  - ***aggregation trees***
    aggregation functions must be (composed of functions that are) commutative and associative
    *strong current research area*
  - ***epidemic protocols***
    aggregation functions must be (composed of functions that are) commutative and associative
    *little current research*
  - ***sampling techniques***
    *little current research*
- related research in ***sensor networks***, using aggregation trees, with similar objectives but different constraints

# Distributed Monitoring using the Management Overlay

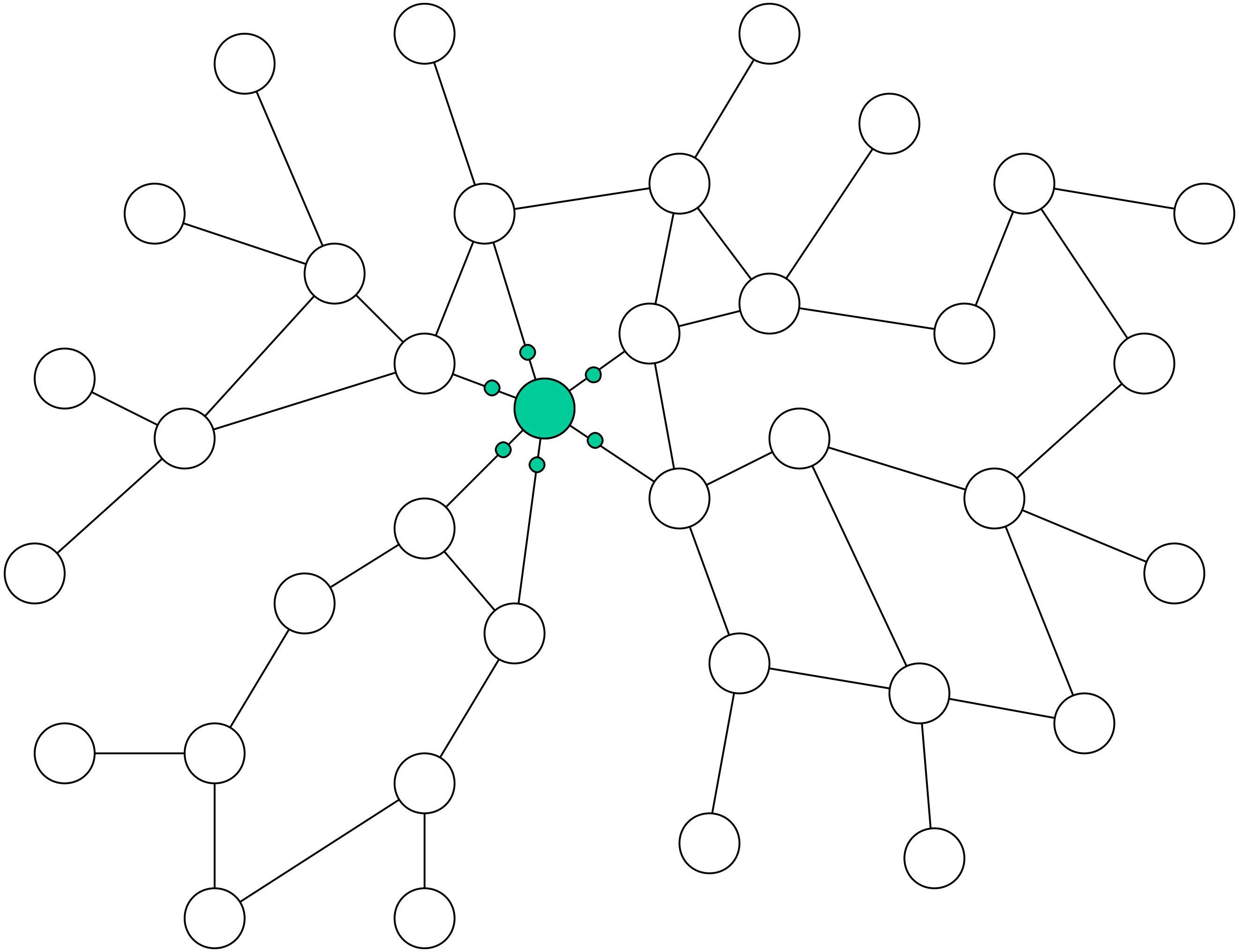
Local device variables are monitored and their values aggregated network-wide

- "Distributed Polling"
  - Pull approach
  - Example: *Echo* [Lim Stadler IM01], [Adam Lim Stadler 05]
- "Distributed Event Reporting," "Streaming Queries"
  - Push approach
  - Example: *GAP* [Dam Stadler RVK05], *A-GAP* [Gonzalez Stadler 05]
- Network Threshold Crossing Alerts
  - Push approach
  - Example: *TCA-GAP* [Wuhib et al. DSOM05] , [Wuhib Stadler Clemm 06]

# Echo: A Generic Protocol for Distributed Polling

- Based on a distributed echo algorithm.
- Requires only local knowledge.
- Generic support for local operations and aggregation.
  (We call it a "***pattern***").
- Any node can be the start node for an echo pattern.
- Creates spaning tree on management overlay, with start node as root.
- Two phases of operation:
  - *expansion phase:* local management operations are distributed;
  - *contraction phase*, the results are collected, incrementally aggregated
- More info: [Lim Stadler IM01], [Adam Lim Stadler 05]

# Echo Pattern (expansion)



# Pseudo Code of the Echo Pattern

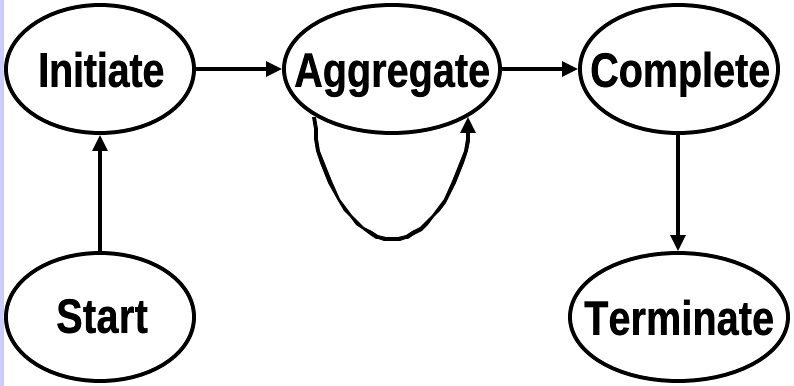
```
visited : boolean          init false;
N       : set of NodeId    init neighbors();
parent: NodeId             init undefined;

Echo( inmsg: bytes, from: NodeId ) {

   if from = undefined
      aggregator->Start( in_msg );
   else
      N := N - from;

   if visited = false {
      parent := from;
      visited := true;
      outmsg= aggregator->Initiate( inmsg );
      if N != empty
          Send_message( N, outmsg);
   } else
      aggregator->Aggregate( inmsg );

   if N = empty {
      outmsg = aggregator->Complete();
      if parent != undefined
         Send_message( parent, outmsg );
      else
         aggregator->Terminate( inmsg );
   }
}
```



# Performance and Robustness of Echo

*Management traffic*

- 2 messages per link, one in each direction.

*Execution time*

- increases linearly with the network diameter d: T = *O(d)*.
  (For most graphs, d grows much slower than n, the number of nodes.)

*Processing load*

- The load is proportional to the number of links of a node (degree).

*Robustness*

- Tolerance to node failures can be achieved through time-outs [Adam Lim Stadler 05] or tree reconstruction. See also [Nielsen 05].

# WQL: Distributed Query Processing for Real-time Views using Echo

- Non-trivial application of distributed polling with result aggregation: *dynamic creation of management views for large networks*
  - expressiveness of relational model
  - distributed processing and aggregation of device-level data
- Benefits compared to a centralized management system
  - Scalability--capacity *O(N);* response time *O(d);* processing load evenly distributed
  - Robustness---all management nodes perform identical functions
- More information: [Lim Stadler IM05], [Adam Lim Stadler 05]

k Search Favorites Media

http://brooklyn.imit.kth.se/console/index.php?menu_selected=1

eaver
nsole 1.0

Status | Live View | Query | Help

Query: select SrcIp, DstIp, SrcPort, DstPort, ToS, Application, Max(ByteCount*8)/SamplingInterval as BitRate, Max(PacketCount)/Sampl
where Timestamp > 2004-05-25 11:05:39 group by SrcIp, DstIp, SrcPort, DstPort order by BitRate des

Show top 25 flows Apply

Top Flows

raffic Composition

## Top 25 Flows by Bit Rate

| Src IP | Dest IP | Src Port | Dest Port | TOS | Application | Bitrate (kbps) | Packet Rat |
|---|---|---|---|---|---|---|---|
| 192.168.1.234 | 192.168.32.167 | 2132 | 80 | 0 | http | 279.0 | 44.4 |
| 192.168.4.220 | 192.168.34.140 | 424 | 80 | 0 | http | 200.4 | 31.9 |
| 192.168.3.43 | 192.168.12.220 | 1041 | 389 | 0 | ldap | 196.2 | 31.2 |
| 192.168.3.127 | 192.168.52.226 | 2908 | 389 | 0 | ldap | 162.3 | 25.8 |
| 192.168.3.76 | 192.168.32.134 | 381 | 389 | 0 | ldap | 100.2 | 15.9 |
| 192.168.3.25 | 192.168.2.93 | 901 | 69 | 0 | tftp | 100.2 | 15.9 |
| 192.168.3.163 | 192.168.13.48 | 639 | 69 | 0 | tftp | 100.2 | 15.9 |
| 192.168.3.23 | 192.168.4.169 | 1234 | 194 | 0 | irc | 90.2 | 14.3 |
| 192.168.3.82 | 192.168.13.60 | 270 | 80 | 0 | http | 86.2 | 13.7 |
| 192.168.3.194 | 192.168.2.36 | 840 | 194 | 0 | irc | 70.1 | 11.2 |
| 192.168.1.86 | 192.168.34.72 | 2940 | 25 | 0 | smtp | 40.1 | 6.4 |
| 192.168.1.175 | 192.168.33.37 | 1723 | 25 | 0 | smtp | 38.7 | 6.1 |
| 192.168.4.234 | 192.168.34.178 | 13 | 194 | 0 | irc | 30.1 | 4.8 |
| 192.168.4.233 | 192.168.32.183 | 1939 | 25 | 0 | smtp | 30.1 | 4.8 |
| 192.168.3.170 | 192.168.16.11 | 2978 | 23 | 0 | telnet | 28.4 | 4.5 |
| 192.168.1.224 | 192.168.32.111 | 3188 | 25 | 0 | smtp | 9.5 | 1.5 |
| 192.168.32.2 | 192.168.5.5 | 1966 | 1028 | 0 | other | 3.7 | 1.9 |

Status | Live View | Query | Help

0

ition

Query: select Application, Sum(ByteCount) as ByteCount, Sum(PacketCount) as PacketCount from Flows where Timestamp > 2004-05-25 11:06:02 group
ByteCount desc

## Network Traffic Composition

| Application | Byte Count | Packet Count | |
|---|---|---|---|
| ldap | 342.7 K | 436.5 | (42.6%) |
| tftp | 172.1 K | 219.5 | (21.4%) |
| irc | 152.5 K | 193.4 | (18.9%) |
| http | 90.4 K | 114.9 | (11.2%) |
| smtp | 27.5 K | 34.5 | (3.4%) |
| telnet | 11.8 K | 14.9 | (1.5%) |
| other | 8.0 K | 40.7 | (1.0%) |

# Local Tables on a Management Node

| System table |
| --- |
| WANIp |
| Memory |
| FreeDisk |
| UpSince |

*WAN data*

| Device table |
| --- |
| DeviceIp |
| NumInterfaces |
| Make |
| Model |
| UpSince |

| Interface table |
| --- |
| InterfaceNum |
| InterfaceAddress |
| InterfaceSubnet |
| InterfaceType |
| InterfaceSpeed |

| Flow table |
| --- |
| SrcIp |
| DstIp |
| SrcPort |
| DstPort |
| Application |
| ByteCount |
| PacketCount |
| Protocol |
| Timestamp |
| SamplingInterval |

*Router data*

# The Weaver Query Language WQL

- Queries are expressed in WQL, an extension to SQL
  Extensions refer to scoping, aggregate functions, implicit attributes

```
SELECT <columns>
        FROM <tables>
        [ ON <startnode> [ FOR <hops> ]]
        [ WHERE <conditions> ]
        [ GROUP BY <groups> [ HAVING <having> ]]
        [ ORDER BY <ordering> ] [ LIMIT <limit> ]
```

- Queries are executed against virtual global tables
  **System table, Device table, Interface table, Flow table**
- MIB objects can be accessed via a virtual MIB table.

# Identify the heaviest flows currently in the network

```
SELECT    MAX((ByteCount*8)/SamplingInterval) as BitRate, SrcIp, DstIp, DstPort
FROM      Flow
GROUP BY  SrcIp, DstIp, DstPort
WHERE     Timestamp >= “15:23:00” and Timestamp <= “15:26:00”
ORDER BY  BitRate DESCENDING
LIMIT     3
```

| BitRate | SrcIp | DstIp | DstPort |
|---|---|---|---|
| 1245232 | 192.168.1.45 | 192.168.2.27 | 1400 |
| 1212442 | 192.168.2.56 | 192.168.3.42 | 5000 |
| 1022451 | 192.168.3.17 | 192.168.51.24 | 138 |

# Identify all FTP flows currently traversing two given routers

```
SELECT    SrcIp, DstIp, SET_CONCAT(DeviceIp) as PathSet
FROM      Flow, Device
WHERE     Timestamp >= “15:23:00” and Timestamp <= “15:23:05”
         and Application = “FTP”
GROUP BY  SrcIp, DstIp
HAVING    STRSTR(PathSet,”192.168.1.1”) and STRSTR(PathSet, “192.168.4.1”)
```

| SrcIp | DstIp | PathSet |
|---|---|---|
| 192.168.1.24 | 192.168.4.47 | 192.168.1.1<br>192.168.2.1<br>192.168.4.1 |
| 192.168.21.24 | 192.168.6.21 | 192.168.21.1<br>192.168.1.1<br>192.168.4.1<br>192.168.6.1 |

# Mapping a Global Query into Local Queries

```
SELECT    MAX((ByteCount*8)/SamplingInterval)
          as BitRate, SrcIp, DstIP
FROM      Flow
GROUP BY  SrcIp, DstIp
WHERE     Timestamp >= “15:23:00” and Timestamp <= “15:26:00”
          and Application = “FTP”
ORDER BY  BitRate DESCENDING
LIMIT     3
```

**G:**
**Global Query**

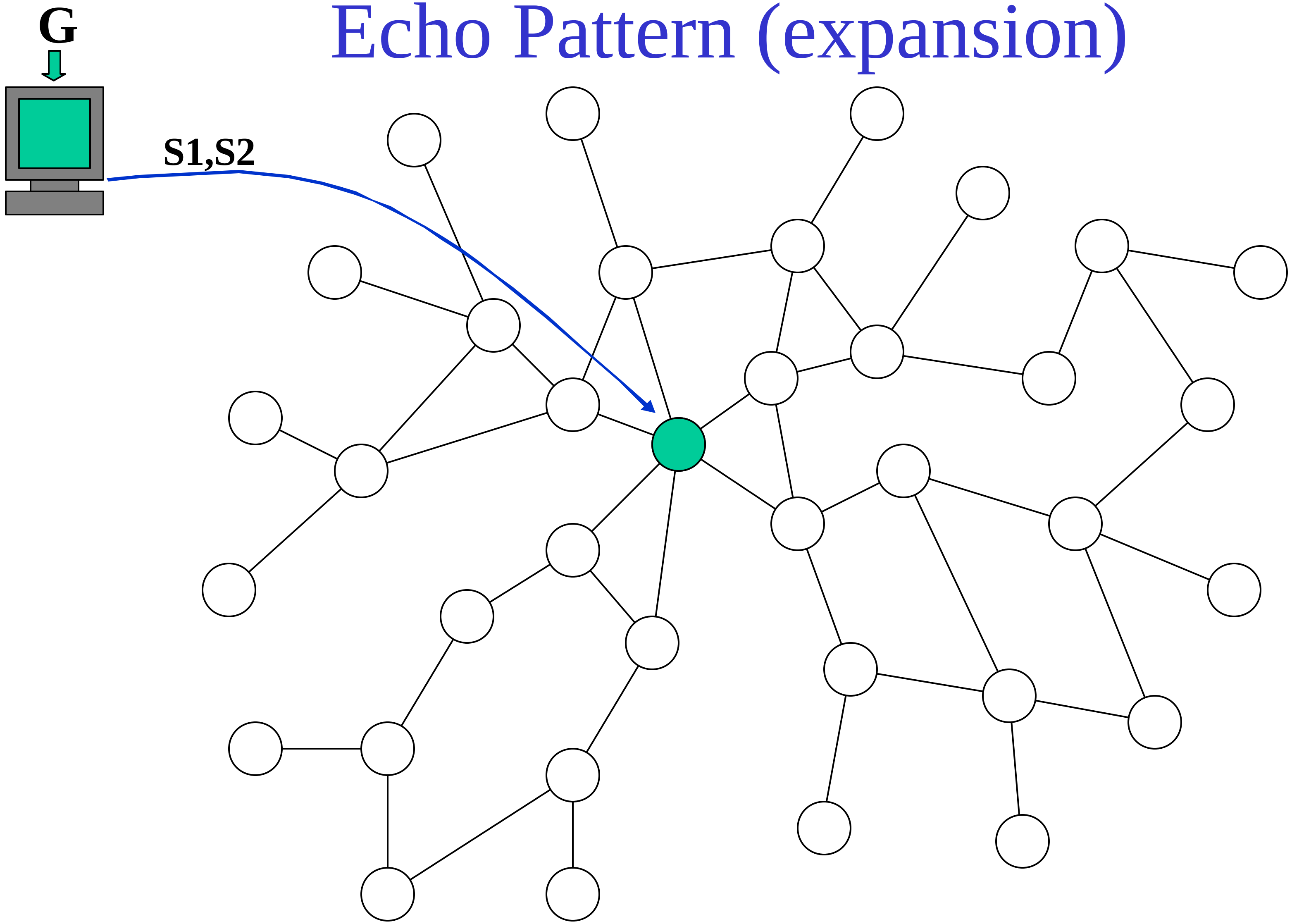
```
SELECT    SrcIp, DstIp, DstPort, MAX(BitRate) as
          BitRate
FROM      TEMP_TABLE
GROUP BY  SrcIp, DstIp
ORDER BY  BitRate DESCENDING
LIMIT     3
```

**S2:**
**Aggregates partial results along spanning tree**

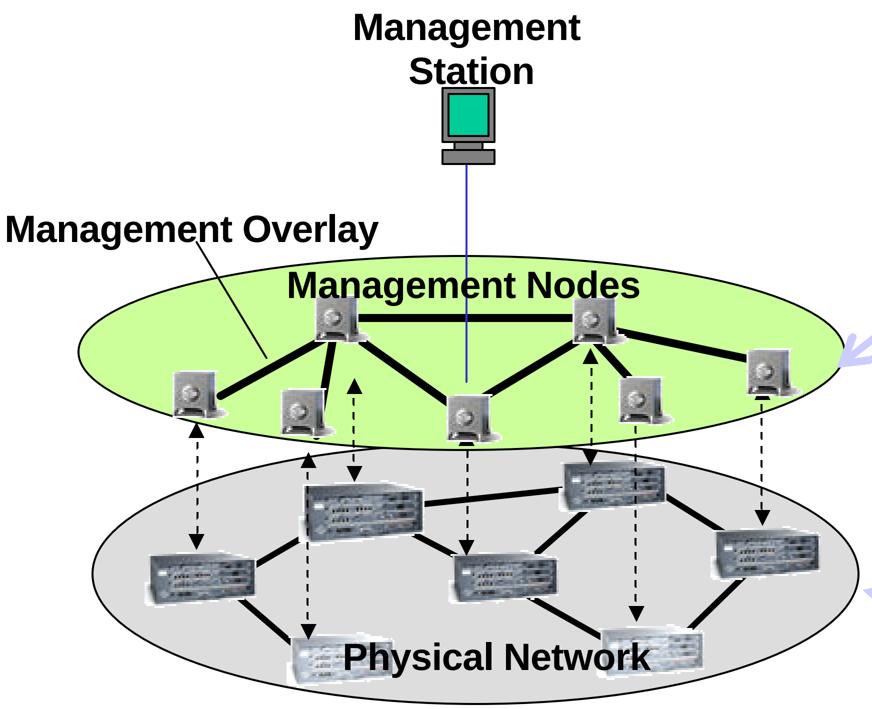
```
CREATE    TEMP_TABLE
SELECT    MAX((ByteCount*8)/SamplingInterval)
          as BitRate, SrcIp, DstIp
FROM      Flow
WHERE     Timest >= “15:23:00” and Timest <= “15:26:00”
          and Application = “FTP”
GROUP BY  SrcIp, DstIp
ORDER BY  BitRate DESCENDING
LIMIT     3
```

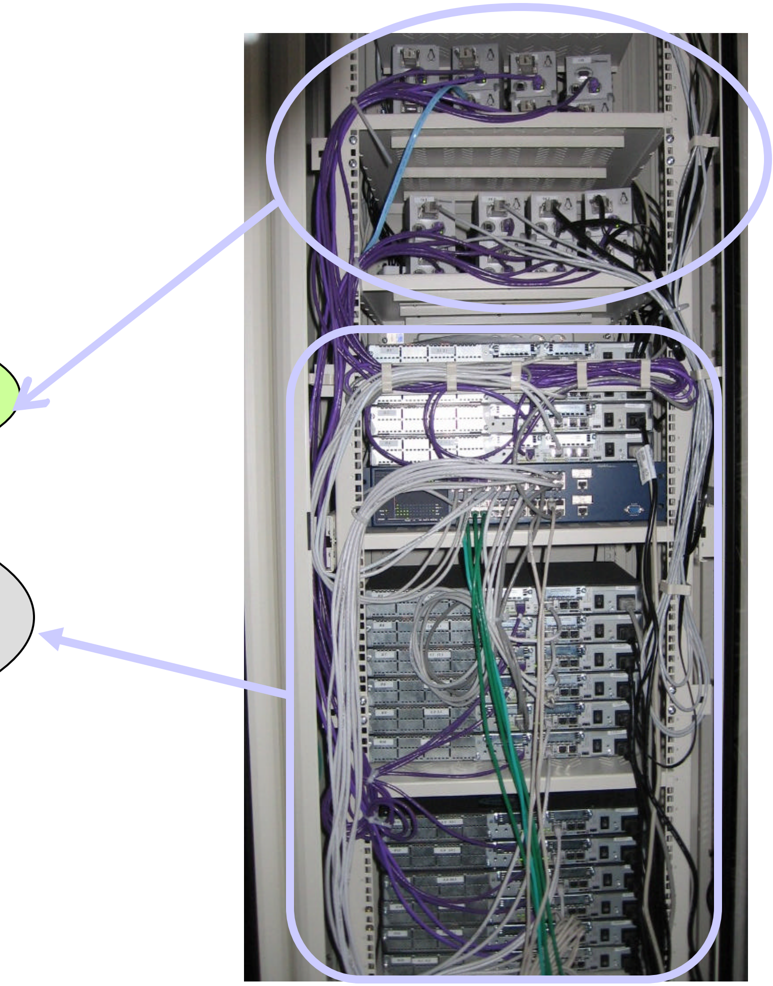
**S1:**
**Executed against local databases**

# Echo Pattern (expansion)



# Implementation on the Lab Testbed

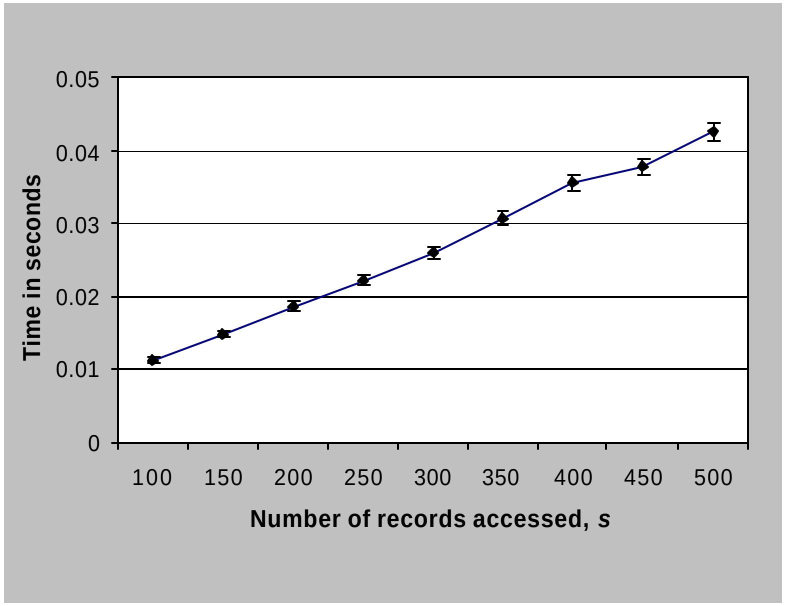


- 16 Cisco 2600 series routers
- 16 Intrinsyc CerfCubes + Linux 2.4.18+MySQL4.0

# The Performance of a WQL Query

- Determining factors
  - Performance of echo patterns [LS01]
    - Execution time: *O(d)*, *d* is network diameter
    - Message complexity: 2 messages per link
  - Performance of DBMS on WANs
    - Execution time: $T = als$
      where
      a: constant
      *l*: record length
      *s*: records in local database
  - Network conditions

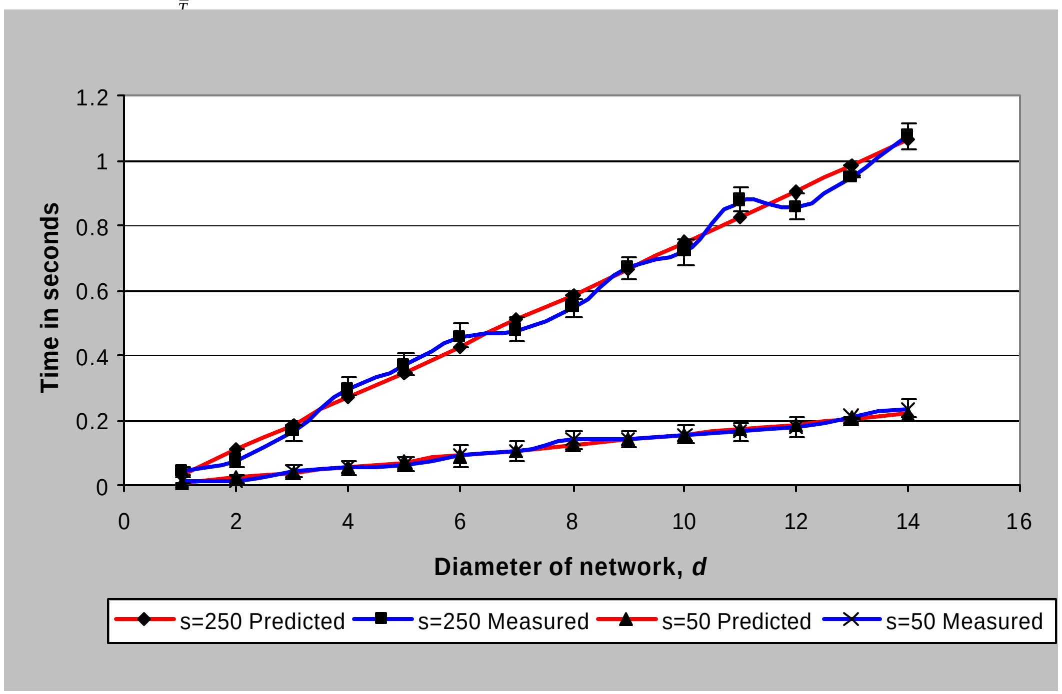


# Modeling and Validating the Execution time of a WQL Query on Weaver

Upper bound on execution time:

$$C_{time} \leq d\left(\bar{c}\,t_q + a\,l\bar{s} + glU + 2t_n\right) + 2a\,lU\left(d - 1\right)$$

where
*d*: diameter of network
*s*: records in local database
*U*: max records of local S1,S2
a: DBMS processing capacity




For supremum queries:

$C_{time}$ proportional to $a\bar{s}d$

# Literature
# www.ee.kth.se/~stadler/nmrg


[Wuhib Stadler Clemm 06] F. Wuhib, R. Stadler, A. Clemm; "Implementation and Evaluation of a Protocol for Detecting Network-Wide Threshold Crossing Alerts", KTH Technical Report, January 2006.

[Gonzalez Stadler 05] A. Gonzalez Prieto, R. Stadler; "Distributed Real-time Monitoring with Accuracy Objectives", KTH Technical Report, December 2005.

[Wuhib at al DSOM05] F. Wuhib, A. Clemm, M. Dam, R. Stadler; "Decentralized Computation of Threshold Crossing Alerts", 16th IFIP/IEEE Distributed Systems Operations and Management (DSOM 05), Barcelona, Spain, October 24-26, 2005.

[Lim Stadler IM05] K.S. Lim and R. Stadler; "Real-time Views of Network Traffic Using Decentralized Management", 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 16-19, 2005.

[Dam Stadler RVK05] M. Dam and R. Stadler; "A Generic Protocol for Network State Aggregation", In Proc. Radiovetenskap och Kommunikation (RVK'05), Linköping Sweden, June 14-16, 2005.

[Adam Lim Stadler 05] C. Adam, K.S. Lim and R. Stadler; "Decentralizing Network Management", KTH Technical Report, December 2005.

[Lim Stadler IM01] K.S. Lim and R. Stadler: "A navigation pattern for scalable Internet management," IFIP/IEEE International Symposium on Integrated Network Management (IM'01), Seattle, Washington, 14-18 May, 2001.