The MicroPulse Framework for Adaptive Waking Windows in Sensor Networks

Demetrios Zeinalipour-Yazti (Univ. of Cyprus)

Panayiotis Andreou (Univ. of Cyprus)

Panos Chrysanthis (Univ. of Pittsburgh, USA)

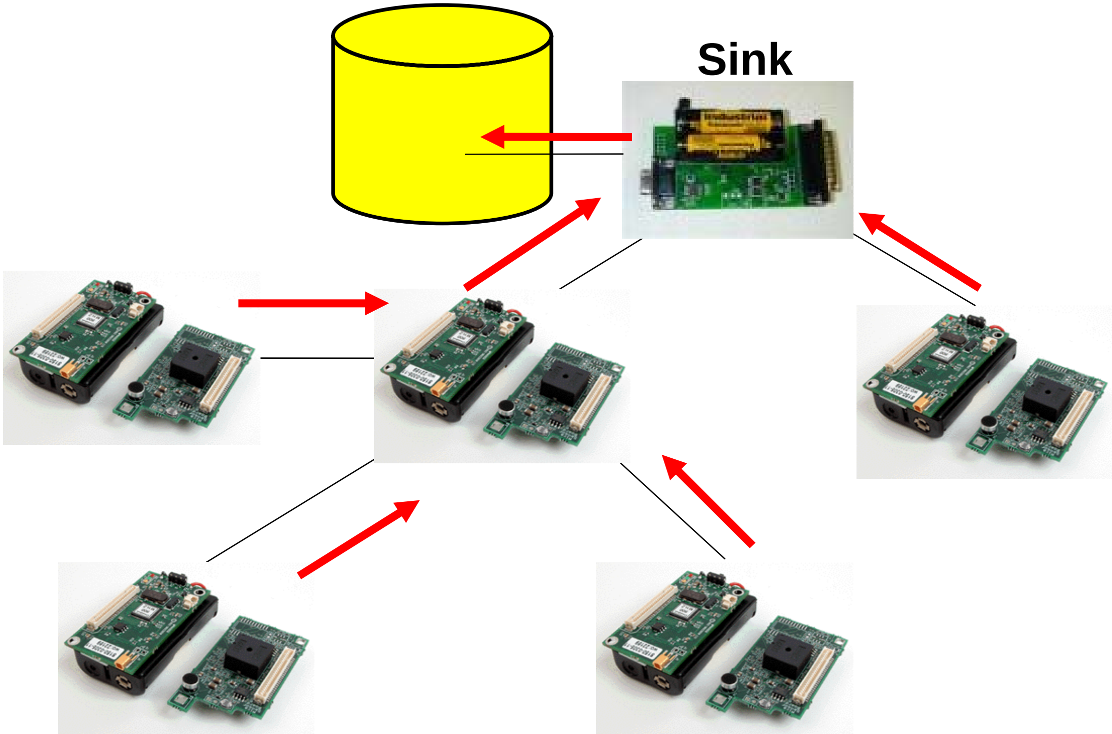
George Samaras (Univ. of Cyprus)

Andreas Pitsillides (Univ. of Cyprus)

http://www.cs.ucy.ac.cy/~dzeina/

System Model

- Distributed Sensing + Centralized Storage
- A **continuous** Data Acquisition Framework
- **Hierarchical** (tree-based) routing



Motivation

Limitations

- **Energy:** Extremely limited (e.g. AA batteries)
- **Communication:** Transmitting 1 bit over the radio consumes as much energy as ~1000 CPU instructions.

Solution

- Power down the radio transceiver during periods of inactivity.
- Studies have shown that a 2% duty cycle can yield lifetimes of 6 months using 2 AA batteries

Definitions

Definition: Waking Window (τ)

The continuous interval during which sensor A:

- **Enables** its Transceiver.
- **Collects** and **Aggregates** the results from its children for a given Query Q.
- **Forwards** the results of Q to A's parent.

Remarks

- τ is continuous.
- τ can currently not be determined in advance.

Definitions

Tradeoff

- **Small τ : Decrease** energy consumption + Increase incorrect results
- **Large τ:** Increase energy consumption + **Decrease** incorrect results

Problem Definition

Automatically tune τ*, locally at each sensor without any global knowledge or user intervention.*

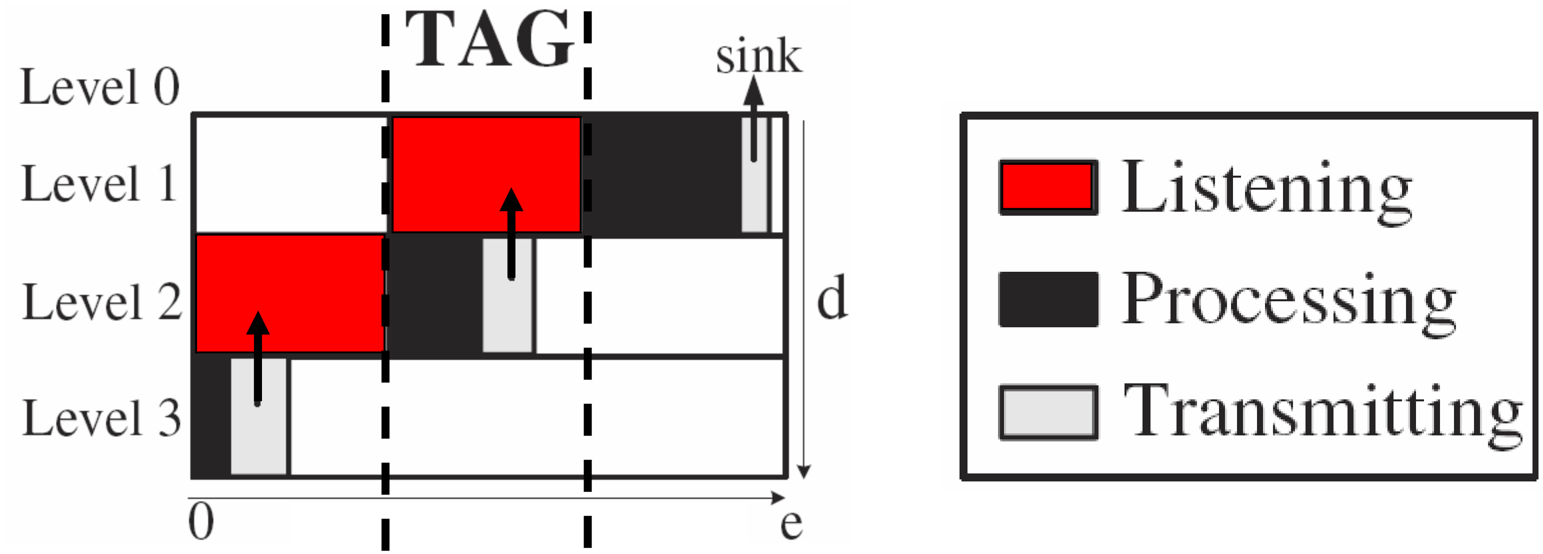
Presentation Outline

- Motivation - Definitions
- **Background on Waking Windows**
- The MicroPulse Framework
  - Construction Phase
  - Dissemination Phase
  - Adaptation Phase
- Experimentation
- Conclusions & Future Work

Background on Waking Windows

The Waking Window in TAG*

- Divide epoch **e** into **d** fixed-length intervals (**d** = depth of routing tree)
- *When nodes at level **i+1** transmit then nodes at level **i** listen.*



* **Madden et. al., In OSDI 2002.**

Background on Waking Windows

Example: The Waking Window in TAG

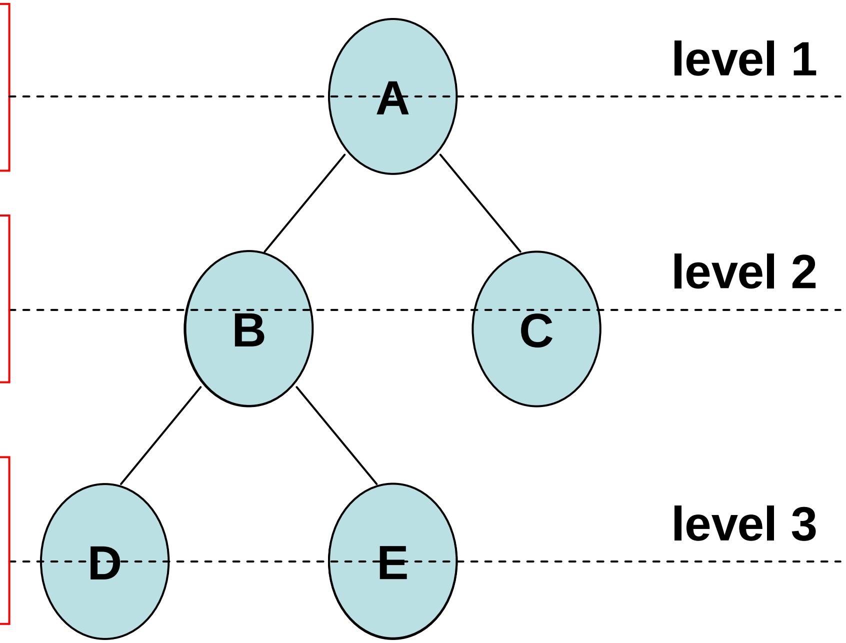
- epoch=31, d (depth)=3

  yields a window $\boldsymbol{\tau_i = \lfloor e/d \rfloor = \lfloor 31/3 \rfloor = 10}$

Transmit: [20..30)
Listen: [10..20)

Transmit: [10..20)
Listen: [0..10)

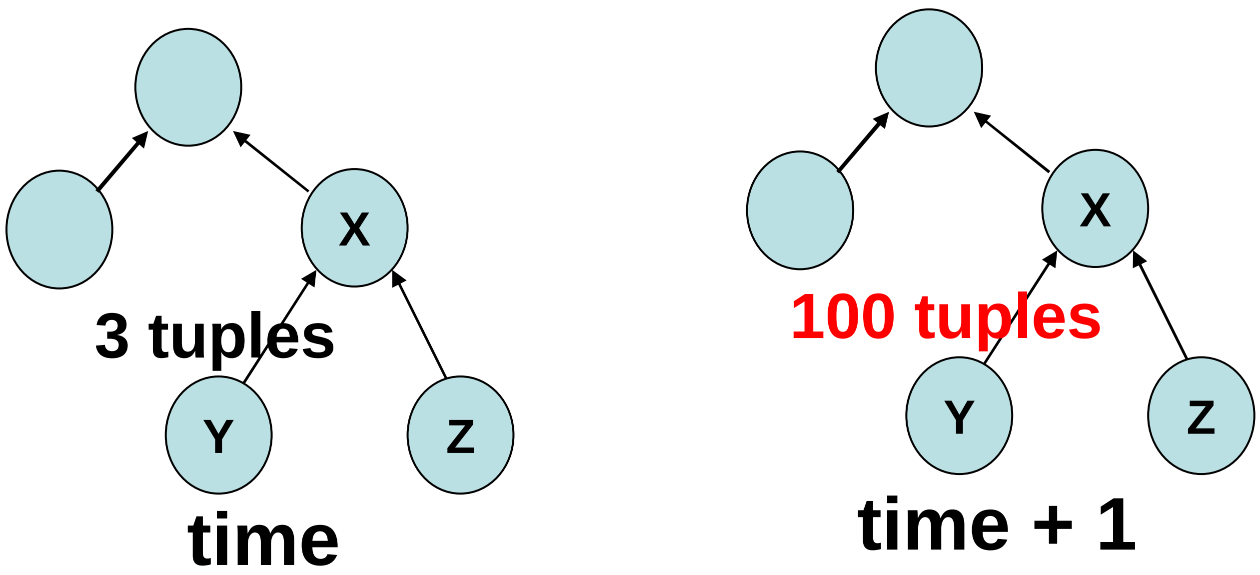
Transmit: [0..10)
Listen: [0..0)



Background on Waking Windows

Disadvantages of TAG's τ

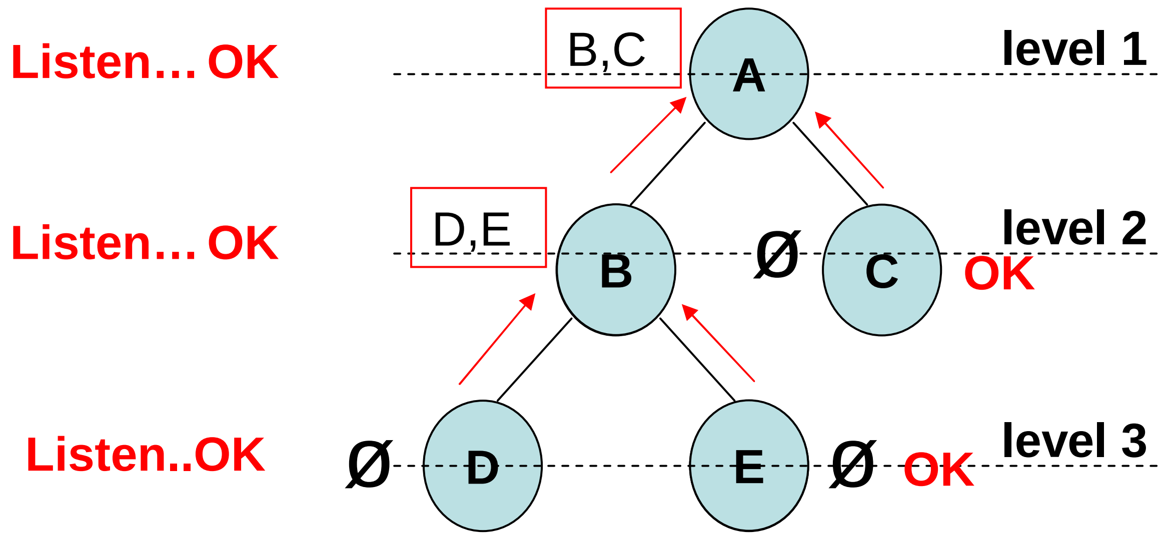
- **τ is an overestimate**
  - In our experiments we found that it is three orders of magnitude bigger than required.
- **τ does not capture variable workloads**
  - e.g., X might need a larger τ in (time+1)



Background on Waking Windows

The Waking Window in Cougar*

- Each node maintains a “waiting list”.



- Forwarding of results occurs when all children have answered (or timer **h** expires)

* Yao and Gehrke, In CIDR 2003.

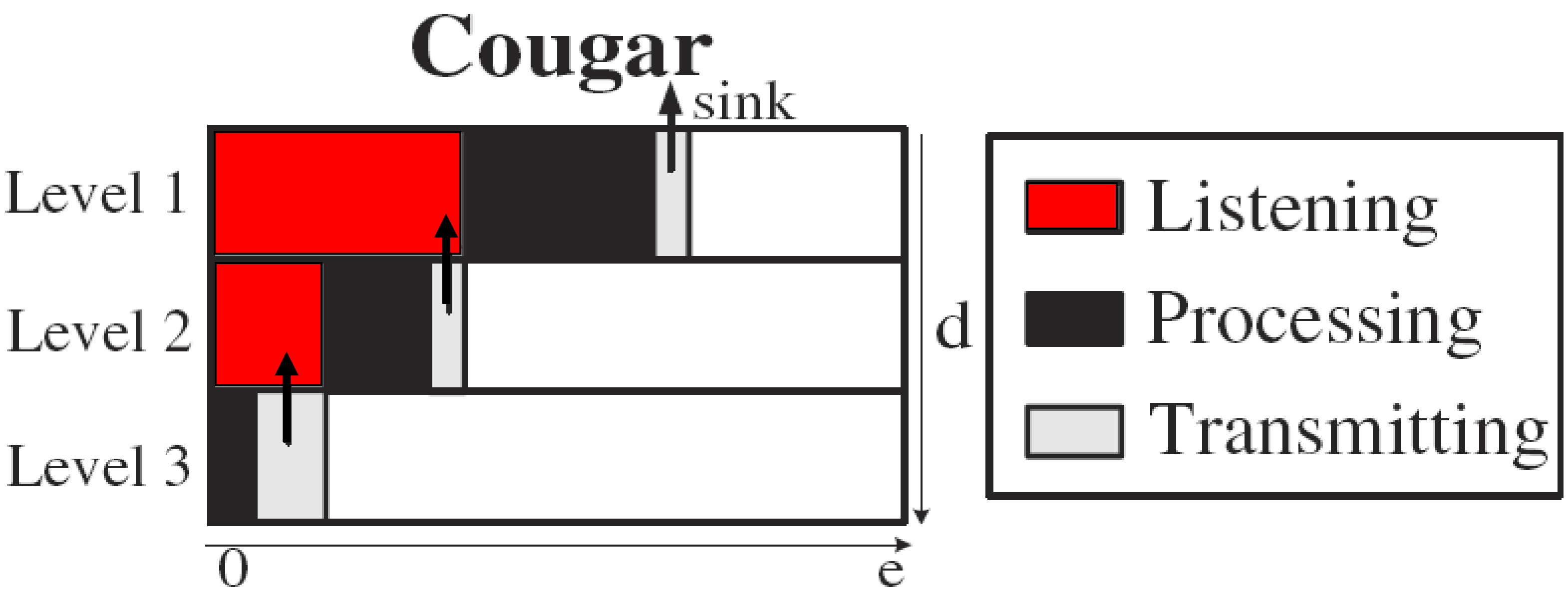
Background on Waking Window

Cougar's Advantage (w.r.t. τ)

- More fine-grained than TAG.

Cougar's Disadvantage (w.r.t. τ)

- Parents keep their transceivers active until all children have answered….this is recursive.



Presentation Outline

- Motivation - Definitions
- Background on Waking Windows
- **The MicroPulse Framework**
  - **Construction Phase**
  - **Dissemination Phase**
  - **Adaptation Phase**
- Experimentation
- Conclusions & Future Work

The MicroPulse Framework

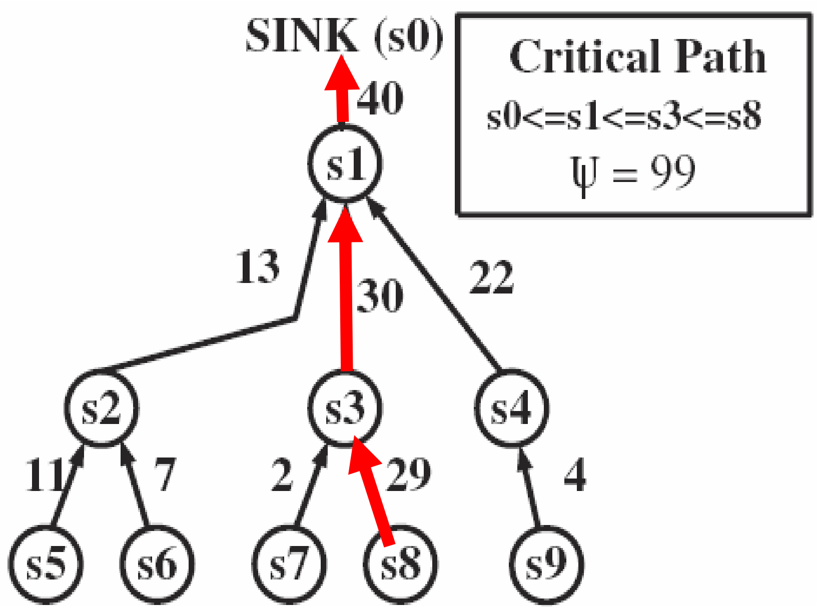
- A new framework for automatically tuning τ.

- **MicroPulse :**
  - Profile recent data acquisition activity
  - Schedule **τ** using an in-network execution of the ***Critical Path Method (CPM)***

- CPM is a graph-theoretic algorithm for scheduling project activities.
- CPM is widely used in construction, software development, research projects, etc.

The MicroPulse Framework

- MicroPulse Phases
  - **Construct** the critical path cost Ψ.
  - **Disseminate** Ψ in the network and define **τ.**
  - **Adapt** the **τ** of each sensor based on Ψ.

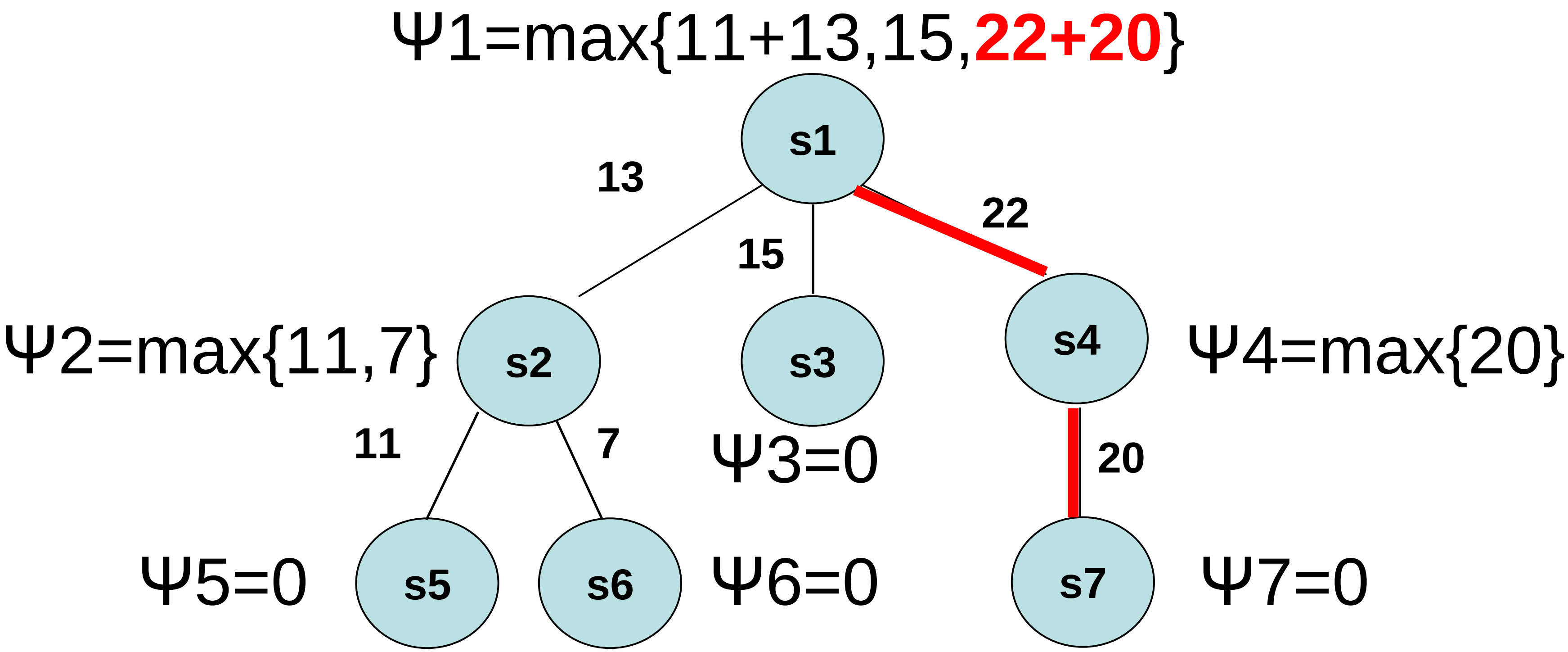


Intuition

Ψ allows a sensor to schedule its waking window.

The Construction Phase

Construct Ψ:



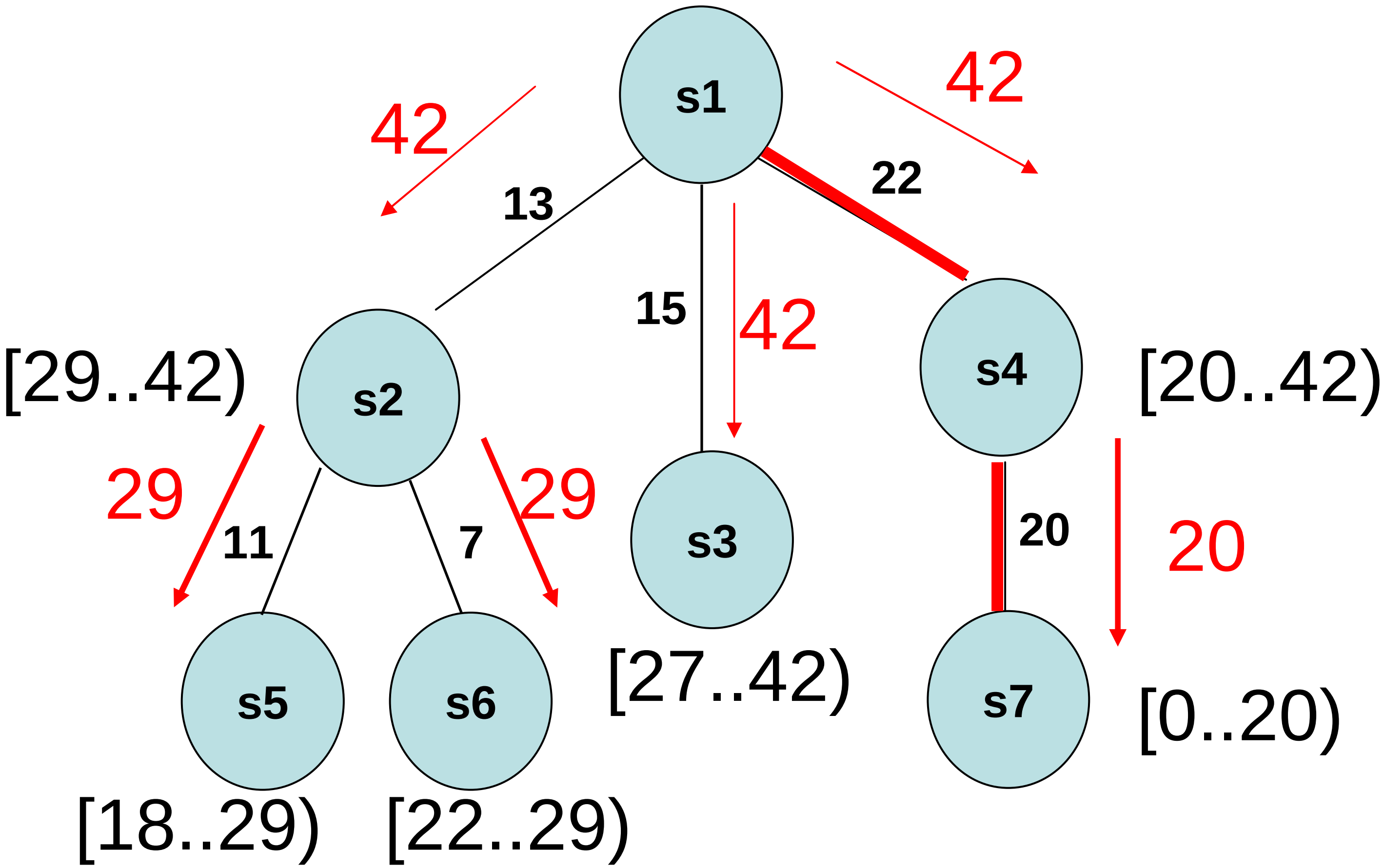
Recursive Definition:

$$\Psi_i = \begin{cases} 0 & \text{, if } s_i \text{ is a leaf node.} \\ \max_{\forall j \in children(s_i)} \{\Psi_j + s_{i,j}\} & \text{, otherwise} \end{cases}$$

The Dissemination Phase

Construct Waking Windows (τ):

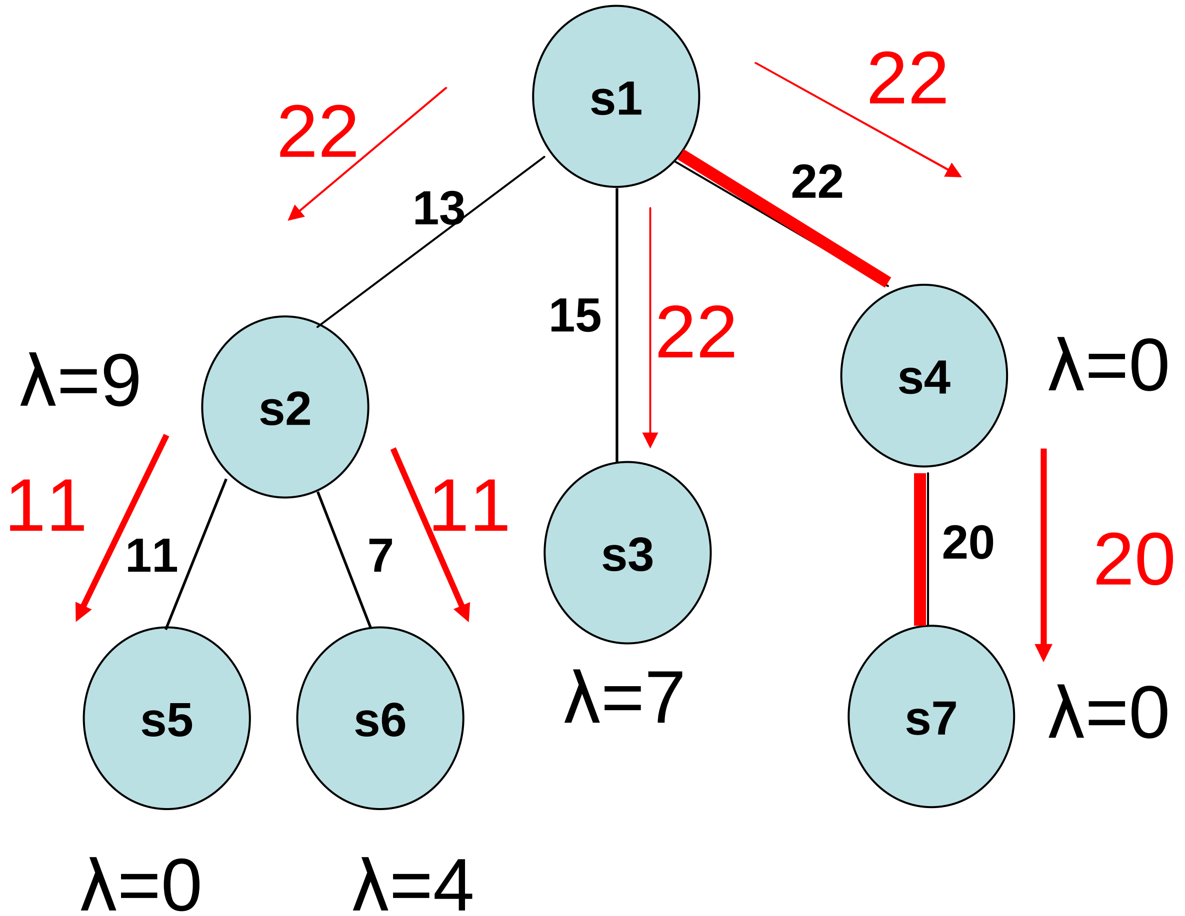
"Disseminate Ψ = 42 to all nodes (top-down)"



The Dissemination Phase

Construct Local Slack (λ):

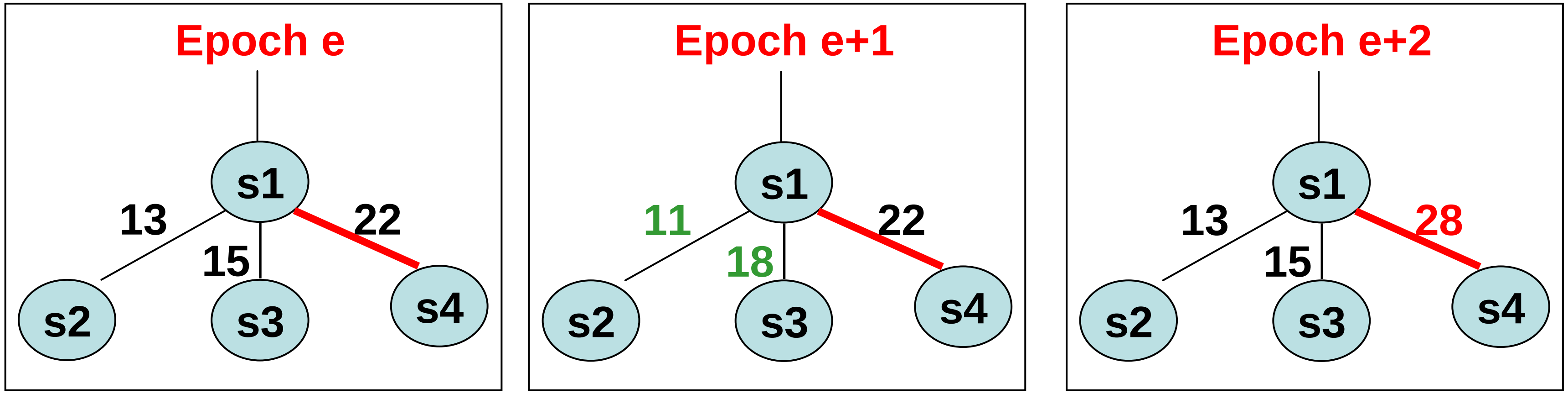
"maximum possible workload increase for the children of a node"



The Adaptation Phase

Intuition

- Workload changes are expected, e.g.,



- **Question:** Should we reconstruct τ?
- **Answer:** Yes/No.
  - No in Case e+1, because s2 & s3 know their local slack.
  - Yes in Case e+2, because the critical path has been affected.

Presentation Outline

- Motivation - Definitions
- Background on Waking Windows
- The MicroPulse Framework
  - Construction Phase
  - Dissemination Phase
  - Adaptation Phase
- **Experimentation**
- Conclusions & Future Work

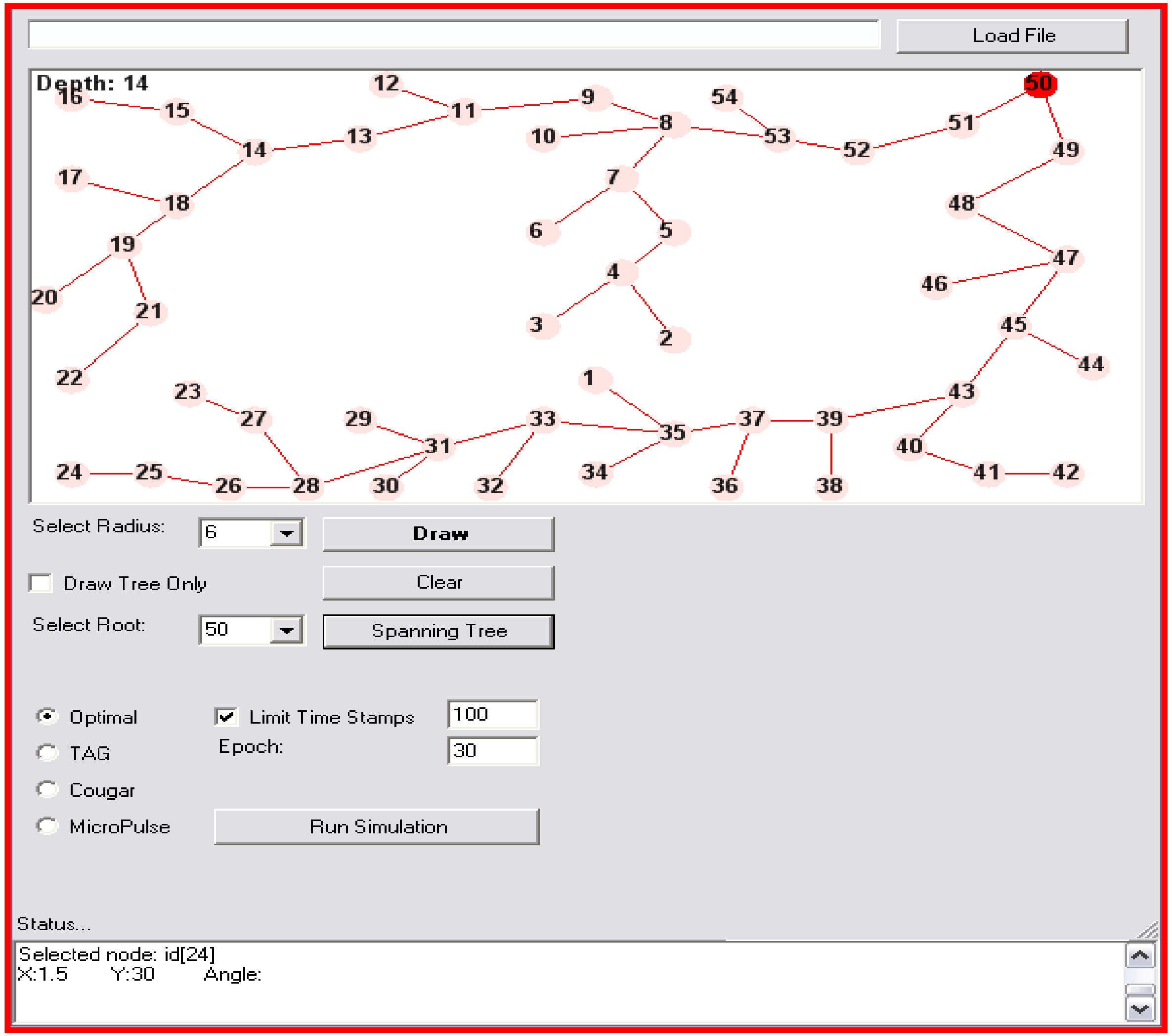
Experimentation

- We have implemented a visual simulator that implements the **waking window algorithm** of TAG, Cougar and MicroPulse.

- Failure Rate = 20%

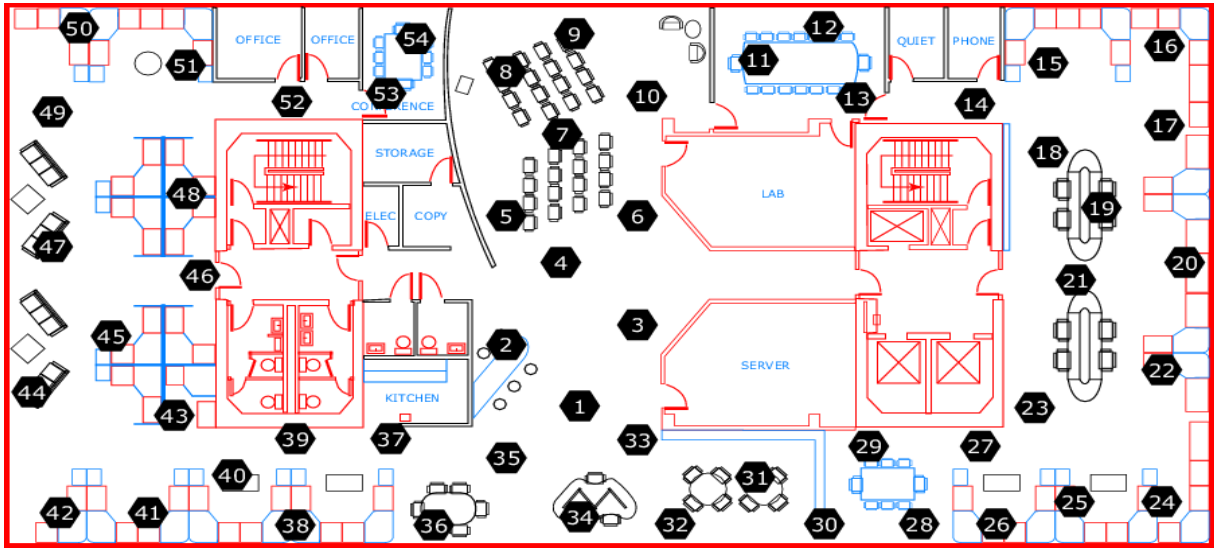
- Child Wait Expiration Timer h = 200 ms

Experimentation



Experimentation

- **Dataset: Intel Lab Data**
  - 58 sensors deployed in the Intel Berkeley Research Lab (28/2/04 – 5/4/04).
  - 2.3 Million Readings: topology info, humidity, temperature, light and voltage
  - Epoch = 31 seconds



Available at: http://db.csail.mit.edu/labdata/labdata.html

Experimentation

- **Sensing Device**
  - We utilize the energy model of Crossbow's TELOSB Sensor (250Kbps, RF On: 23mA)
  - Trace-driven experimentation using **Energy = Volts x Amperes x Seconds.**

- **Query:**

  SELECT moteid, temperature

  FROM sensors

  EPOCH DURATION 1 min

Energy Consumption
TAG versus **MicroPulse**

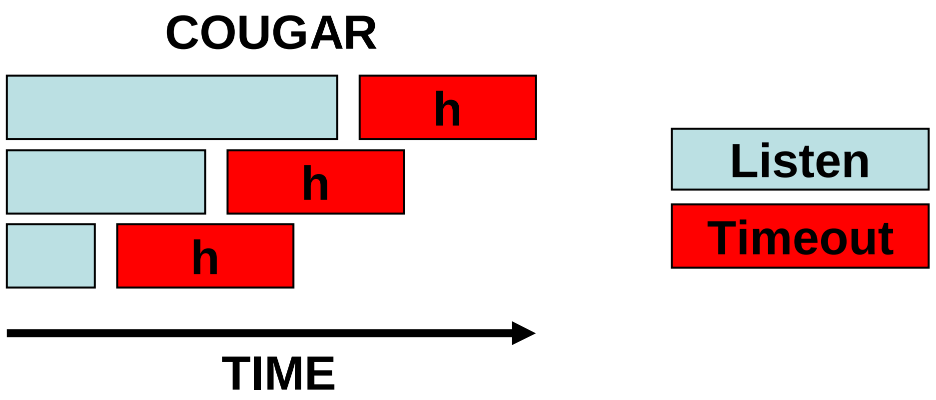
- The waking window in TAG is three orders of magnitude more expensive than in MicroPulse.
  - **TAG:** 7,984mJ
  - **MicroPulse:** 13.75±0.58mJ

- Difference attributed to the waking window **τ :**
  - **τ** in TAG is uniform: 2.21sec. (31 /14 depth)
  - **τ** in MicroPulse is non-uniform: 146ms on average.

Energy Consumption

Cougar versus **MicroPulse**

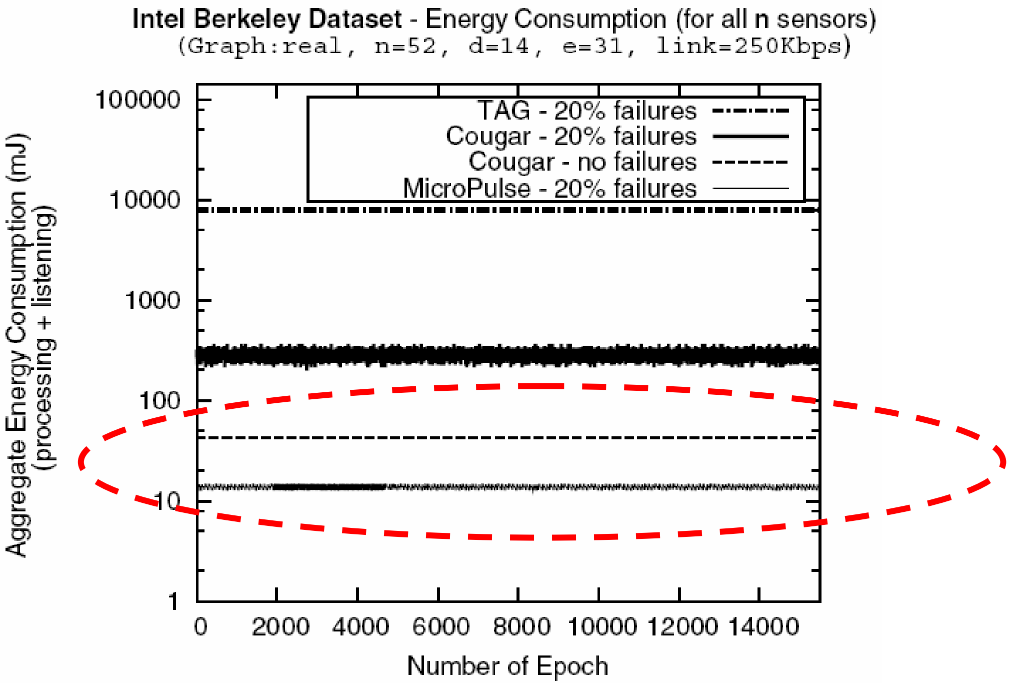
- Same observation holds for Cougar (one order of magnitude more expensive)
  - **Cougar:** 288.97±24.42mJ
  - **MicroPulse:** 13.75±0.58mJ
- **Observation:** A failure at level **K** of the hierarchy results in a **K*h** increase in **τ,** where h is the expiration timer.



Energy Consumption

Cougar versus **MicroPulse**

- We maintain a competitive advantage over Cougar (no failures)
  - **Cougar (no failures):** 42mJ
  - **MicroPulse:** 13.75±0.58mJ



Presentation Outline

- Motivation - Definitions
- Background on Waking Windows
- The MicroPulse Framework
  - Construction Phase
  - Dissemination Phase
  - Adaptation Phase
- Experimentation
- **Conclusions & Future Work**

Conclusions

- We have presented the **design** of MicroPulse that adapts the waking window of a sensing device.

- **Experimentation** with real datasets reveals that MicroPulse can reduce the cost of the waking window by three orders of magnitude.

- We intend to study more carefully the **adaptation algorithms.**

The MicroPulse Framework for Adaptive Waking Windows in Sensor Networks

Thank you!

Questions?

This presentation is available at:
http://www.cs.ucy.ac.cy/~dzeina/talks.html

Related Publications available at:
http://www.cs.ucy.ac.cy/~dzeina/publications.html