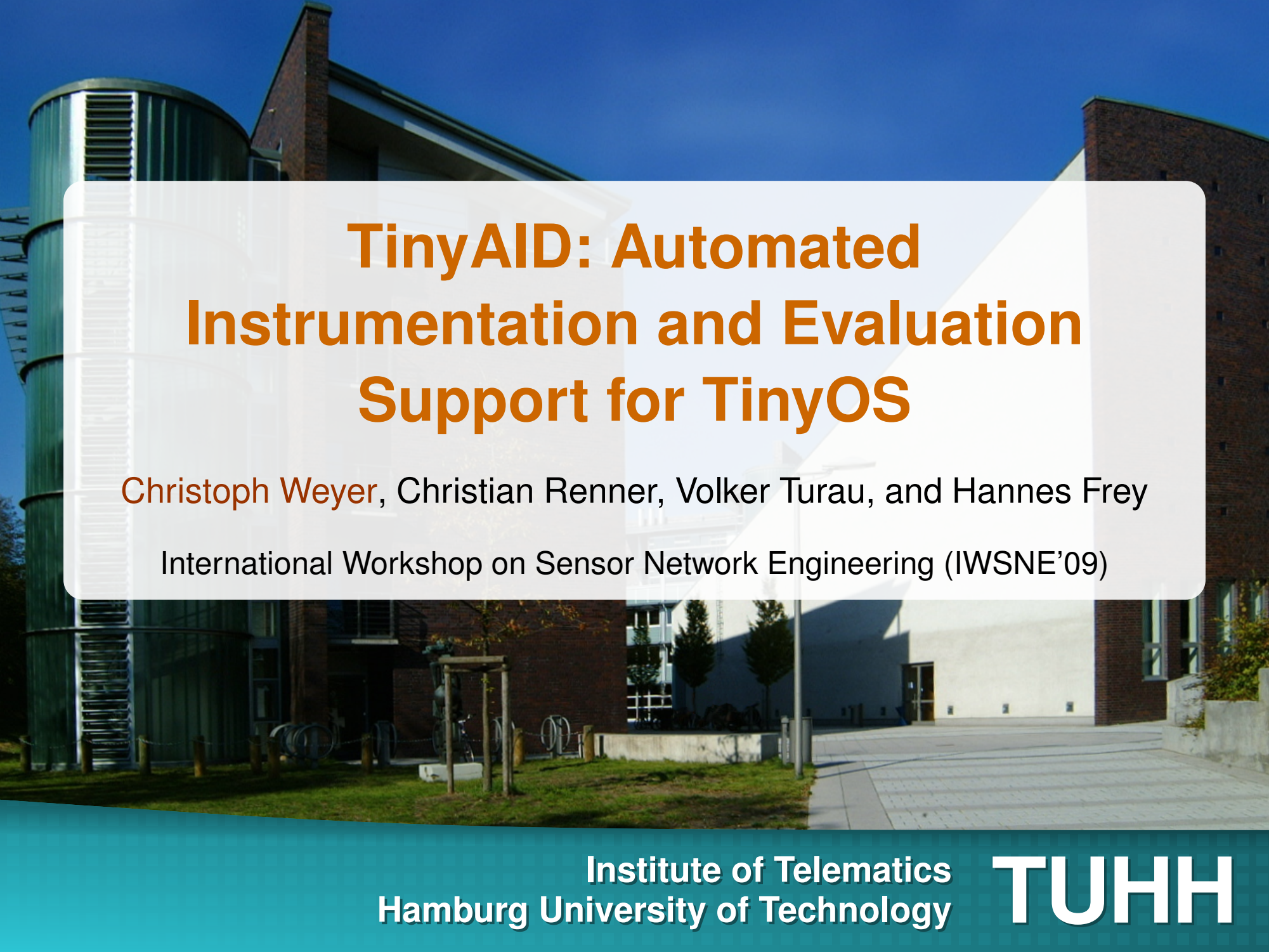


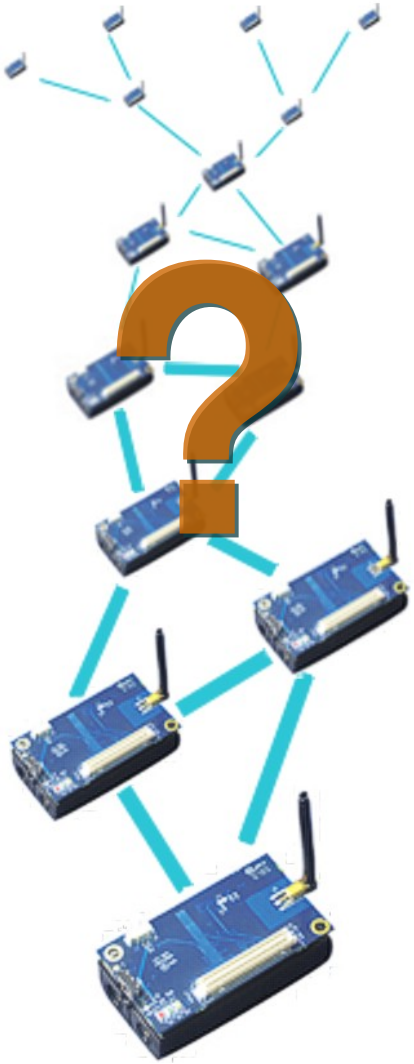
TinyAID: Automated Instrumentation and Evaluation Support for TinyOS

Christoph Weyer, Christian Renner, Volker Turau, and Hannes Frey

International Workshop on Sensor Network Engineering (IWSNE'09)

Institute of Telematics
Hamburg University of Technology
TUHH

Development Support for WSNs



Problem:

- Debugging new implementations
  - → Logging of internal information
- Comparing existing implementations
  - → Extracting comparable metrics

Looking for: Development support tools

Solution: Automated approach

- Instrumentation
- Evaluation

Automated Development Support

Goals:

- Extract information without any manual interventions
- Gather essential information about network state
  - State of nodes
  - Packet flows within the network
- Small memory footprint and low runtime overhead

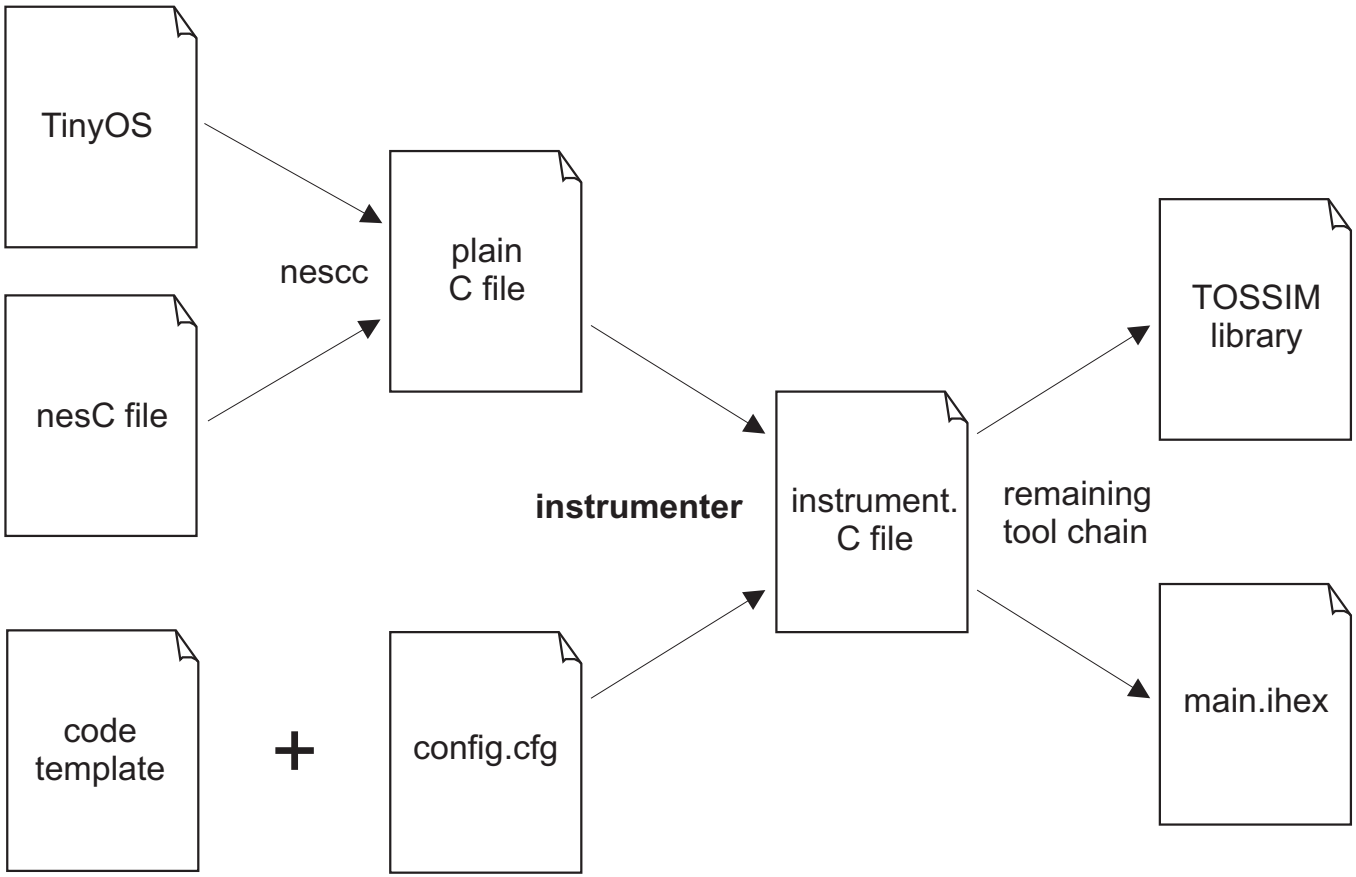
Preconditions:

- Coding conventions are required
- TinyOS provides some kind of conventions

Information gathered by TinyAID

- Call-chain logging
  - Occurrence of events reflects change of node state
    e.g., TinyOS events `Timer.fired`
  - Currently executed component
    Part of the C function name that is called
  - Monitoring what components are turned on/off
    Tracing specific events, e.g., `Radio.startDone`
- Packet logging
  - Sending and receiving a packet
  - Tracking of packets over multiple hops
- Adding timestamps to logged data

Instrumentation



Configuration of Call Chain Logging

```
-d /opt/tinyos-2.x/.*   # exclude everything in /opt/tinyos-2.x
+f Test.nc              # include everything in file Test.nc
+h fired                # include all fired event handler
+h booted               # include all booted event handler
```

- Instrumentation code is inserted based on configuration
- Configuration includes (+) or exclude (-)
  - Directories (d)
  - Files (f)
  - Commands or Events (h)
- First match decides action
- If no match is found no instrumentation is inserted

Call Chain Logging

Node ID	Time [ms]	Direction	Handler ID
5	1320	>	42
5	1322	>	36
5	1323	>	12
5	1324	<	12
5	1328	<	36
5	1333	<	42
3	1648	>	20
3	1649	<	20
7	1930	>	42
7	1931	<	42
...	...	...	...

Call Chain Logging

Node ID	Time [ms]	Direction	Handler ID
5	1320	>	42
5	1322	>	36
5	1323	>	12
5	1324	<	12
5	1328	<	36
5	1333	<	42
3	1648	>	20
3	1649	<	20
7	1930	>	42
7	1931	<	42
...	...	...	...

Message Logging

- Adding additional fields to each packet
  - Originating node
  - Unique sequence number for each node
- Information is inserted by calling `Packet.clear()`
- Trace: creating, sending, and receiving of packets

node	time [ms]	action	type	src	dest	origin	seqno
3	3520	c				3	42
3	3521	s	17	3	12	3	42
5	3524	c				5	14
5	3525	s	34	5	65535	5	14
12	3535	r	17	3	12	3	42
3	3520	c				3	43
...	...	...	...	...	...	...	...

Message Logging

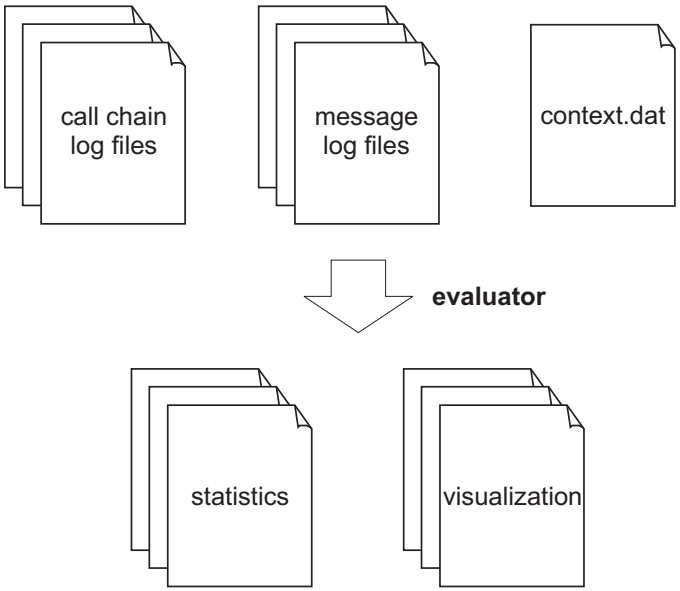
- Adding additional fields to each packet
  - Originating node
  - Unique sequence number for each node
- Information is inserted by calling `Packet.clear()`
- Trace: creating, sending, and receiving of packets

node	time [ms]	action	type	src	dest	origin	seqno
3	3520	c				3	42
3	3521	s	17	3	12	3	42
5	3524	c				5	14
5	3525	s	34	5	65535	5	14
12	3535	r	17	3	12	3	42
3	3520	c				3	43
...	...	...	...	...	...	...	...

Limitations

- Monitoring generally influences system
  - Influence depends on precision
  - System behavior should not be changed
- Not all information can be extracted by an automated process
  - Call-chain not always reflects node state
  - Expressiveness of message reception
- Meta or semantical information is needed

Evaluation

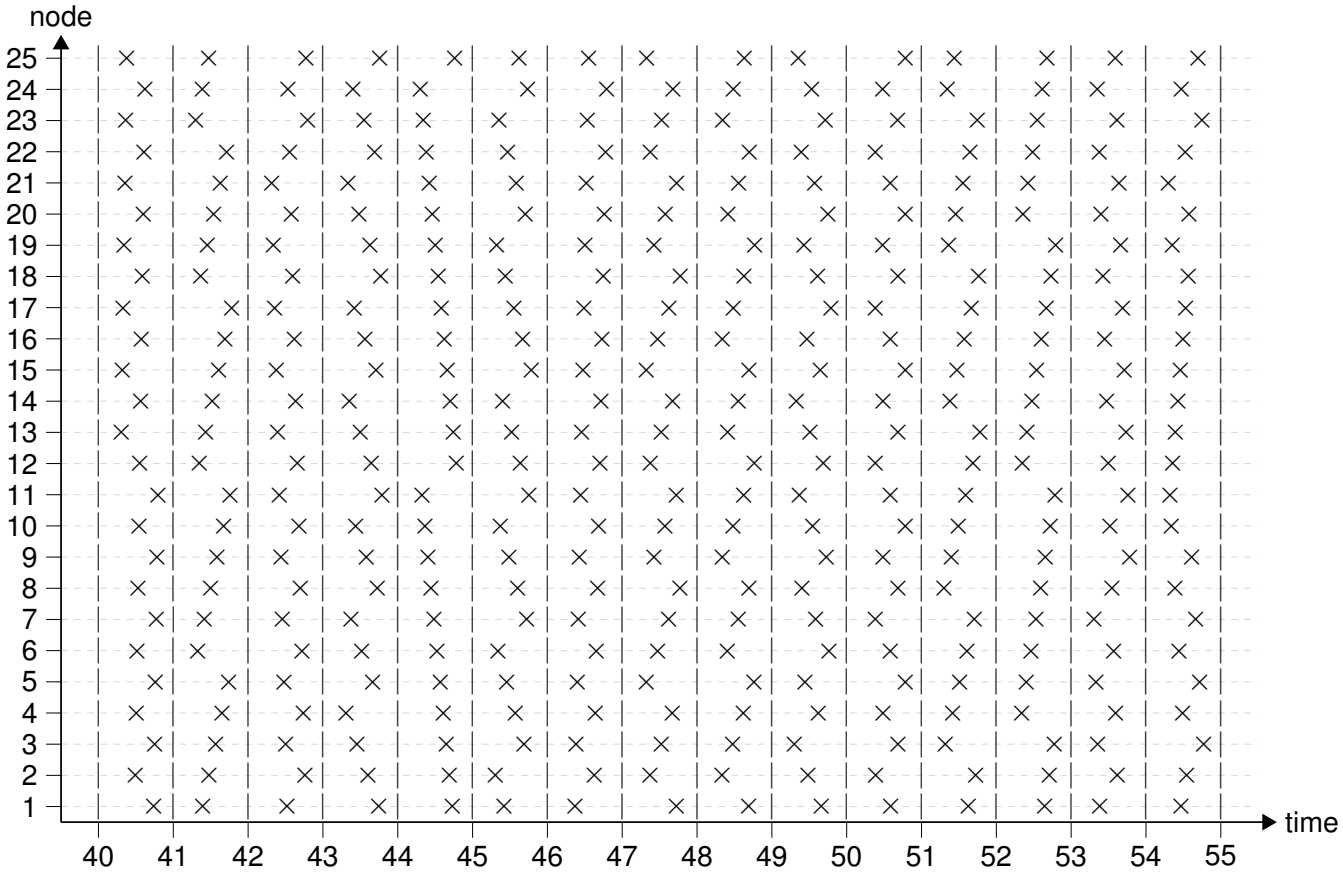


- Runs after instrumentation
- Used input
  - Call-chain logs
  - Message logs
  - Additional semantic information
- Visualization of large amount of data
- Leverages the human visual capabilities

Evaluation of TinyAID

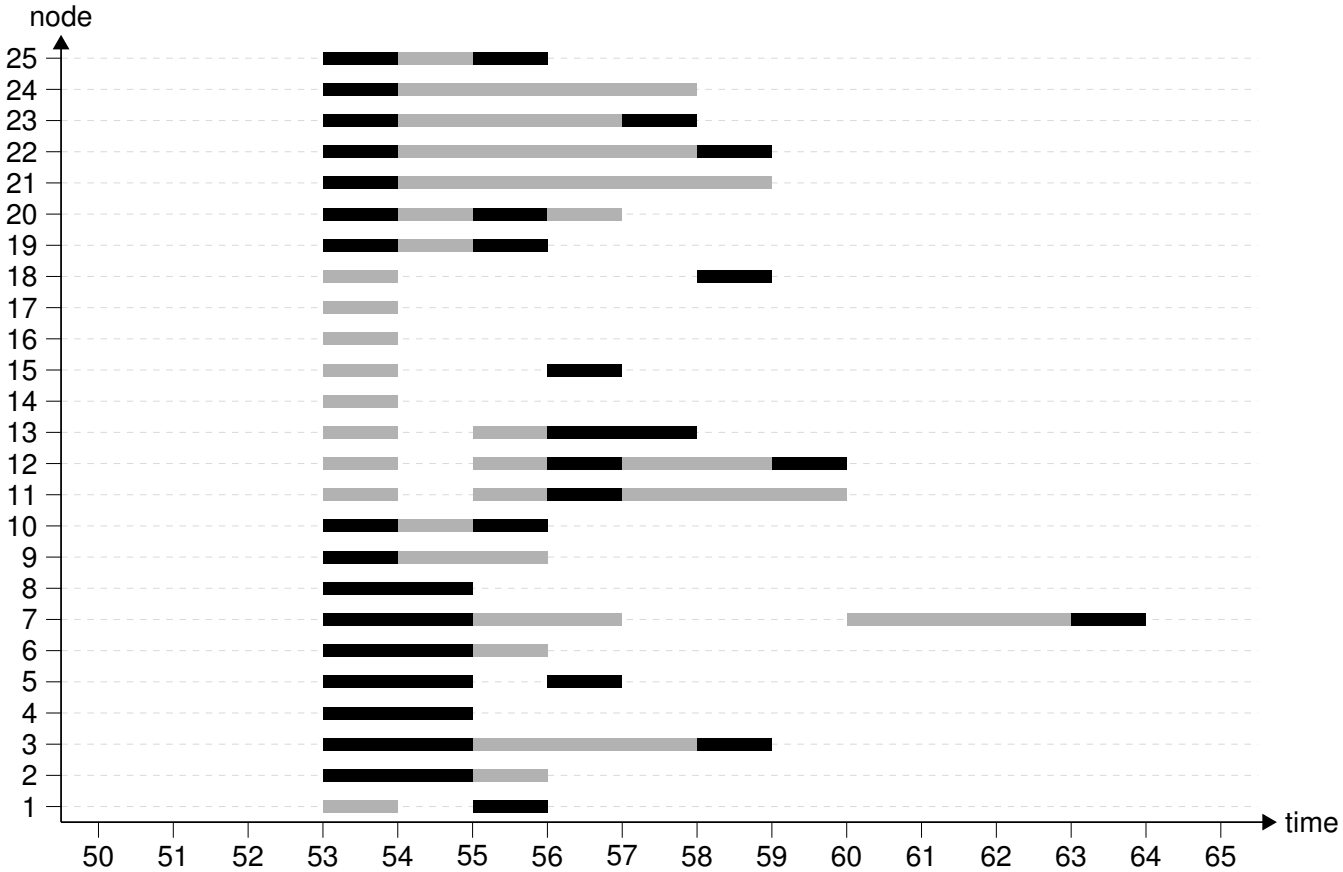
- In real deployments serial interface is the limiting factor
- Overhead due to packet oriented communication
- Presented evaluation is based on TOSSIM only
  - Debugging own implementations
  - Performance comparison of routing algorithms
    - TYMO (part of TinyOS 2.x)
    - Dynamic source routing protocol (DSR)
    - Greedy geographic routing

Concept Evaluation: Event Tracing



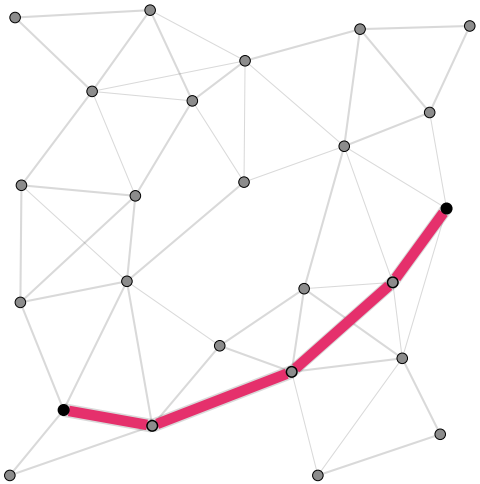
Shortcomings of the random number generator in TOSSIM

Concept Evaluation: State Tracing

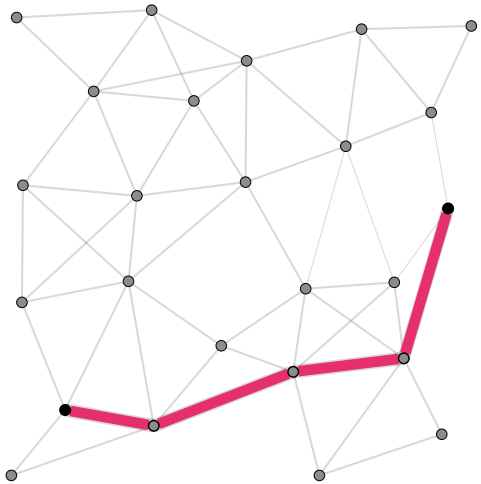


Visualization of program states over time

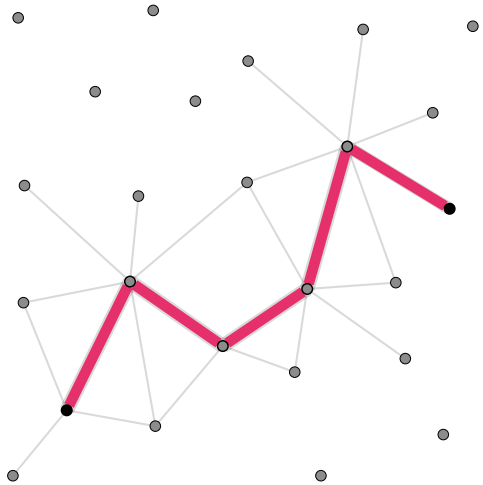
Concept Evaluation: Packet Flow



TYMO



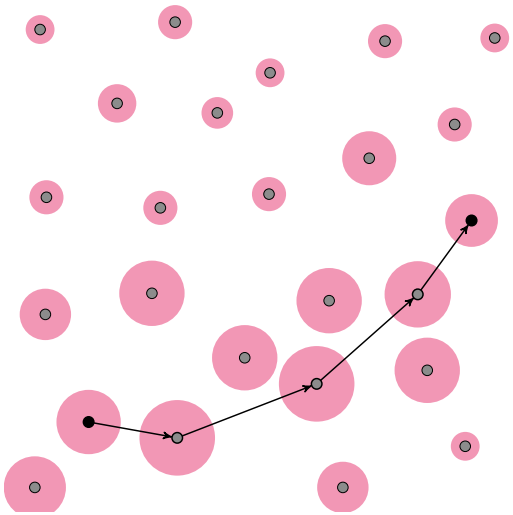
DSR



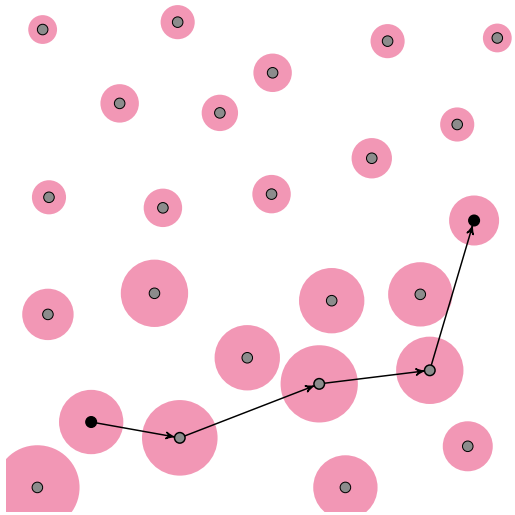
Greedy

- Visualization of number of sent packets over a link
- Identifying routing path decisions

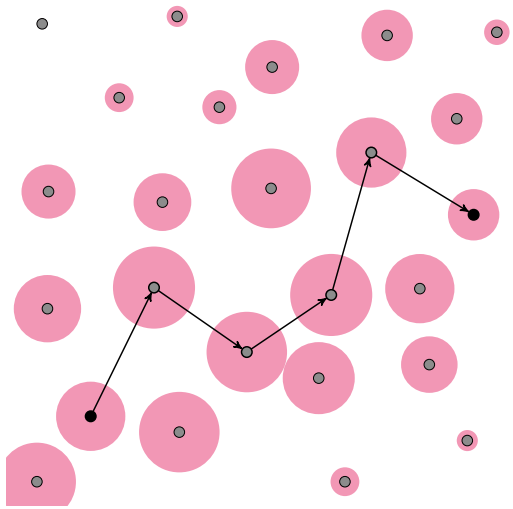
Concept Evaluation: Energy Consumption



TYMO



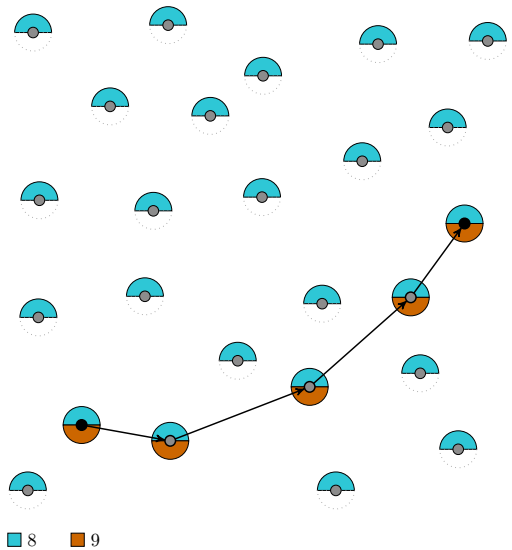
DSR



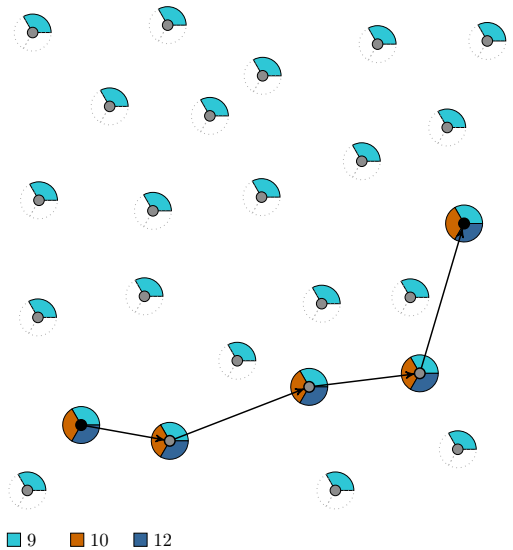
Greedy

- Energy consumption based on communication efforts
- Identification of hot-spots

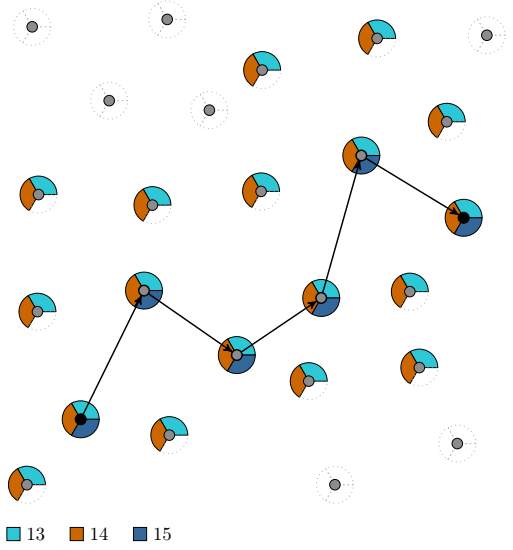
Concept Evaluation: Packet Types



TYMO



DSR



Greedy

- Visualization of activities in the network based on packet types
- Identification of protocol execution

Concept Evaluation: Packet Statistics

	TYMO	DSR	Greedy
Number of Packets			
Created Data Packets	10	10	10
Sent Total	68	68	78
Sent Broadcast	24	24	5
Sent Unicast	44	44	83
Involved Nodes			
Sending	100%	100%	72%
Receiving	100%	100%	72%
Overhearing	56%	56%	96%
Data Packet Latency [ms]			
Minimum	17	24	29
Lower Quartile	23	27	32
Median	26	31	37
Upper Quartile	35	38	45
Maximum	123	144	527

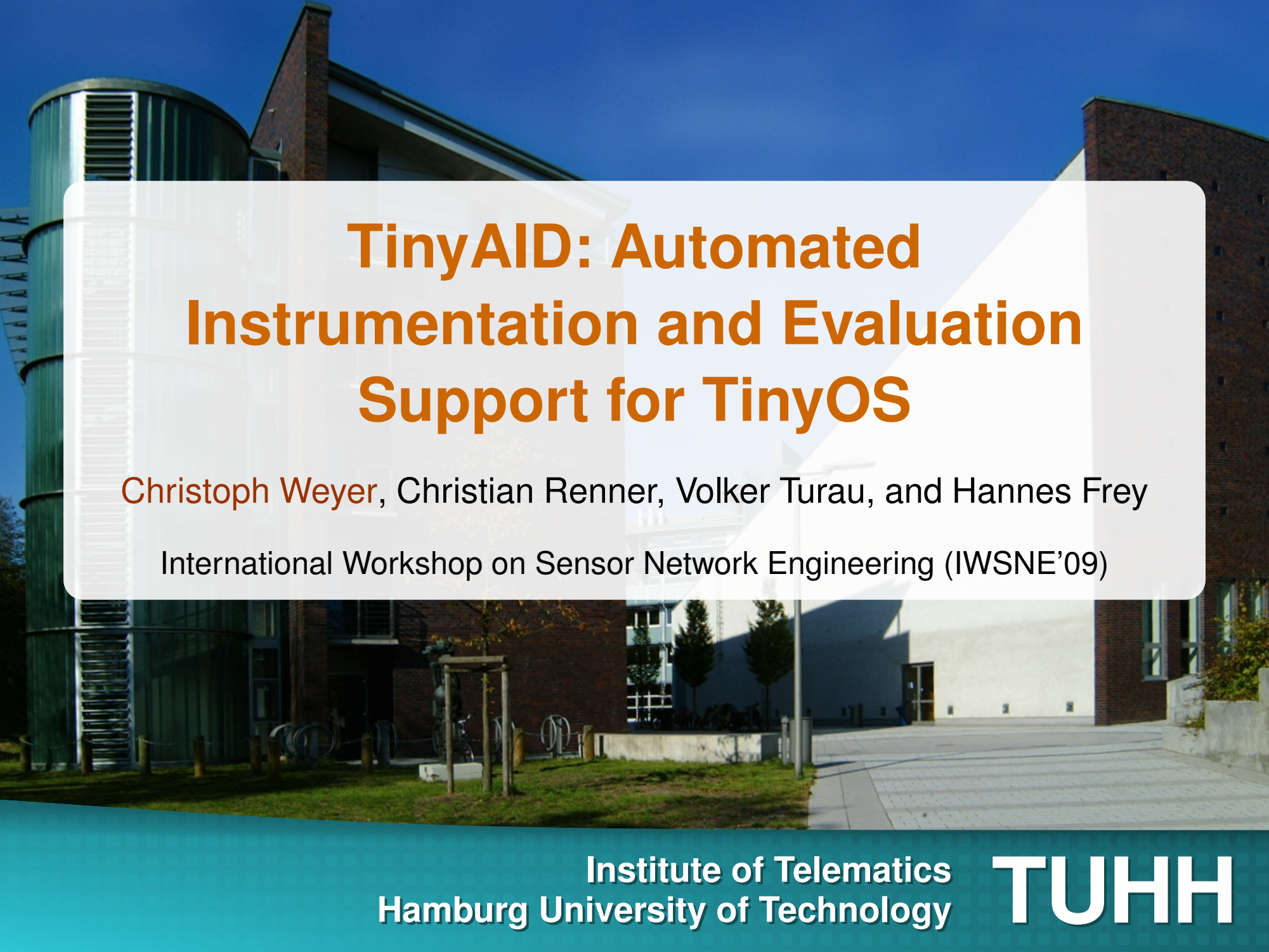
- TinyAID used for performance comparison
- No additional code needed
- Extract statistics from logged data
  - Number of packets
  - Involved Nodes
  - Latency
- Extract reception rate only by
  - Additional information
  - Manual instrumentation

Conclusion

- TinyAID valuable support for TinyOS-based programming
- Simple practicability on legacy source code
- Instant information about internal sequences
- Automated packet tracing

Next steps:

- Apply TinyAID to real hardware
  $\longrightarrow$ hardware interface needed
- Language for specifying instrumentation procedure
  $\longrightarrow$ based on TraceSQL



TinyAID: Automated Instrumentation and Evaluation Support for TinyOS

Christoph Weyer, Christian Renner, Volker Turau, and Hannes Frey

International Workshop on Sensor Network Engineering (IWSNE'09)

Institute of Telematics
Hamburg University of Technology
TUHH