WSN Operating Systems					
	Characteristic	Architecture	Programming Model	Scheduling	Communication Protocol Support
TinyOS	1. Concurrent Programming 2. **Low mem requirements**.	1. **Monolithic Arch 整体架构类** 2. **Component Model**—Glue components together as a static image according to different requirements of application	1. Early version did not support multithreading 2. Later, **TOS event driven**	1. **Non-preemptive FIFO** scheduling algorithm 2. 缺点：**Unfair to latter tasks** especially when short tasks are waiting behind longer ones	Provide **2 multi-hop** protocols —dissemination(传播) & TYMO
Contiki	1. Lightweight **C** for WSN 2. **High mem requirement** 3. provide **various features**: preemptive multithreading, TCP/IP, GUI, etc.	1. **Modular Arch 模块化架构** 2. **Event driven**—lightweight event scheduler dispatch events to running processes	**Preemptive multithreading**—use protothreads, good for severely mem constraint devices	1. Event driven OS所以没有复杂的调度算法 2. Interrupt handler runs with **priority**	**Rich set of protocols** reliable (IPv4, IPv6) / lightweight (uIP)
MANTIS —**M**ultimod**a**l 多模式 system for **N**e**T**works of **I**n-situ wireless **S**ensors	1. Multithreading OS for WSNs 2. **Lightweight & energy efficient** 3. **MOS is portable across multiple platforms**	**Layered Arch 分层架构**—Each layer acts as an enhanced VM to above layers	1. **MOS Preemptive multitasking** 2. Space allocated for RAM: a. global var space b. heap	1. Preemptive **priority-based scheduling** (use Round Robin in each priority class) 2. Ready queue: Kernel, Sleep, High, Normal, Idle	Implement the network stack in: a. user space b. MAC & PHY layers
Nano-RK	1. **Fixed**, preemptive **multitasking real-time** (soft and hard real time) 2. High mem requirement	1. **Monolithic Kernel Arch** 2. Programmers can change parameters to meet the overall goal	1. Large memory consumption —saving the sate of each task 2. Reduced performance & High energy consumption —frequent context switch	1. **Priority scheduling** at 2 levels: **process** and **network** 2. **Fully preemptive priority driven** scheduling algorithm	1. **Lightweight networking protocol stack** similar to sockets (communicate via socket) 2. application bind and listen to port

WSN Operating Systems					
LiteOS	**1. Unix-like OS** 2. Low mem requirements	**Modular Arch** a. LiteShell b. LiteFS c. LiteOS Kernel (use RR and priority scheduling)	1. **Multitasking** OS—support multithreading 2. Run application as **separate threads**, each thread has memory space to avoid errors in shared memory space.	**1. Round Robin & Priority-based** **2. sleep** if no active task	1. In the form of **files** 2. Create file corresponding to the radio interface 3. Place data into the radio and transmit later

TinyDB Examples
Sample Period

```
SELECT nodeid, light, temp
    FROM sensors
    SAMPLE PERIOD 1s FOR 10s
```

TinyDB Examples
Aggregates

```
SELECT AVG(volume), room
  FROM sensors
    WHERE floor = 6
    GROUP BY room
    HAVING AVG(volume) >
  threshold
    SAMPLE PERIOD 30s
```

TinyDB Examples
Temporal Aggregates

```
SELECT WINAVG(volume, 30s,
  5s)
    FROM sensors
    SAMPLE PERIOD 1s
```

TinyDB Examples
Event-Based Queries

```
ON EVENT bird-detect(loc):
    SELECT AVG(light),
  AVG(temp), event.loc
    FROM sensors AS s
    WHERE dist(s.loc,
  event.loc) < 10m
    SAMPLE PERIOD 2s FOR 30s
```

TinyDB Examples
Lifetime-Based Queries

```
SELECT nodeid, accel
    FROM sensors
    LIFETIME 30days
```

TinyDB Examples
Network Health Queries

```
SELECT nodeid, voltage
    WHERE voltage < k
    FROM sensors
    SAMPLE PERIOD 10min
```

TinyDB Examples
Lifetime-Based Queries

```
SELECT nodeid, accel
    FROM sensors
    LIFETIME 30days
```

WSN Middlewares					
	Characteristics				
TinyDB	在TinyOS基础上搭建的传感器数据库	**SRT**—Semantic Routing Tree—根据语义进行routing	table[sensors]=column[attribute]+row[time]	当范围查询沿树传播时，节点将仅将查询转发给查询范围内的那些子节点，潜在地使整个分支不必转发或执行查询	
Maté	**Event-driven** On top of TinyOS	Can **Programming at different levels**. Adjusting parameters -> changing complete	Code can be broken into **capsules** of 24 byte-long instructions to fit TinyOS's packet		
Agilla	**Agent-based** based on Maté	WSN网络**不需要预安装**(**no pre-installed applications**)应用，通过代理(agent)在节点间**迁移(clone)**实现 — agents **migrate** to nodes as needed	多个代理可以共同存在于一个节点	Nodes are identified by their **location** rather than address	
Kairos	**Macroprogramming** middleware implemented in **Python** 宏编程中间件	Translates centralized program into **node-specific distributed version**	Execute locally at every node		
Magnet	The whole network appear as a **single , unified JVM** 让整个网络看起来像一个统一的JVM	VM和宏编程方法的混合	Standard Java program, then Magnet translate program into individual components		
SIXTH	**Distributed system**	Provides uniform access to sensing resources (sensors and data) via sensor adaptors	can be programmed directly, via **SQL-like** or **intelligent agent**	Can **add and reprogram at running time**	