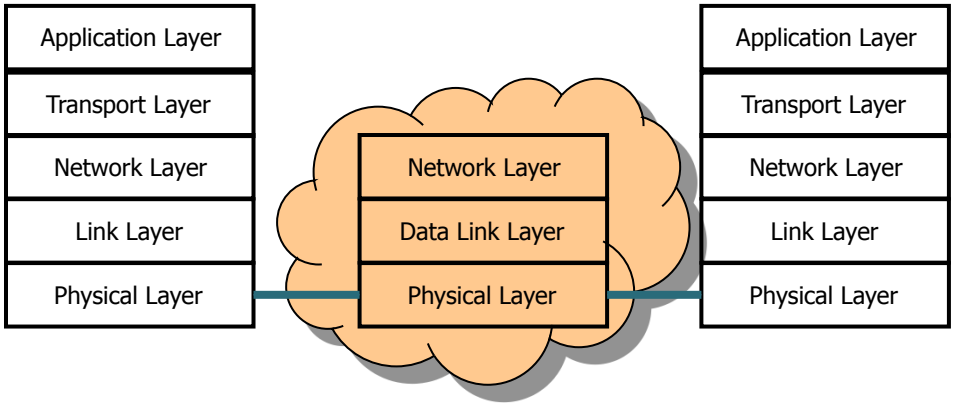
Freie Universität Berlin

Telematics
Chapter 9

Recent Trends



Dr. habil. Emmanuel Baccelli
INRIA / Freie Universität Berlin

Institute of Computer Science
Computer Systems and Telematics (CST)

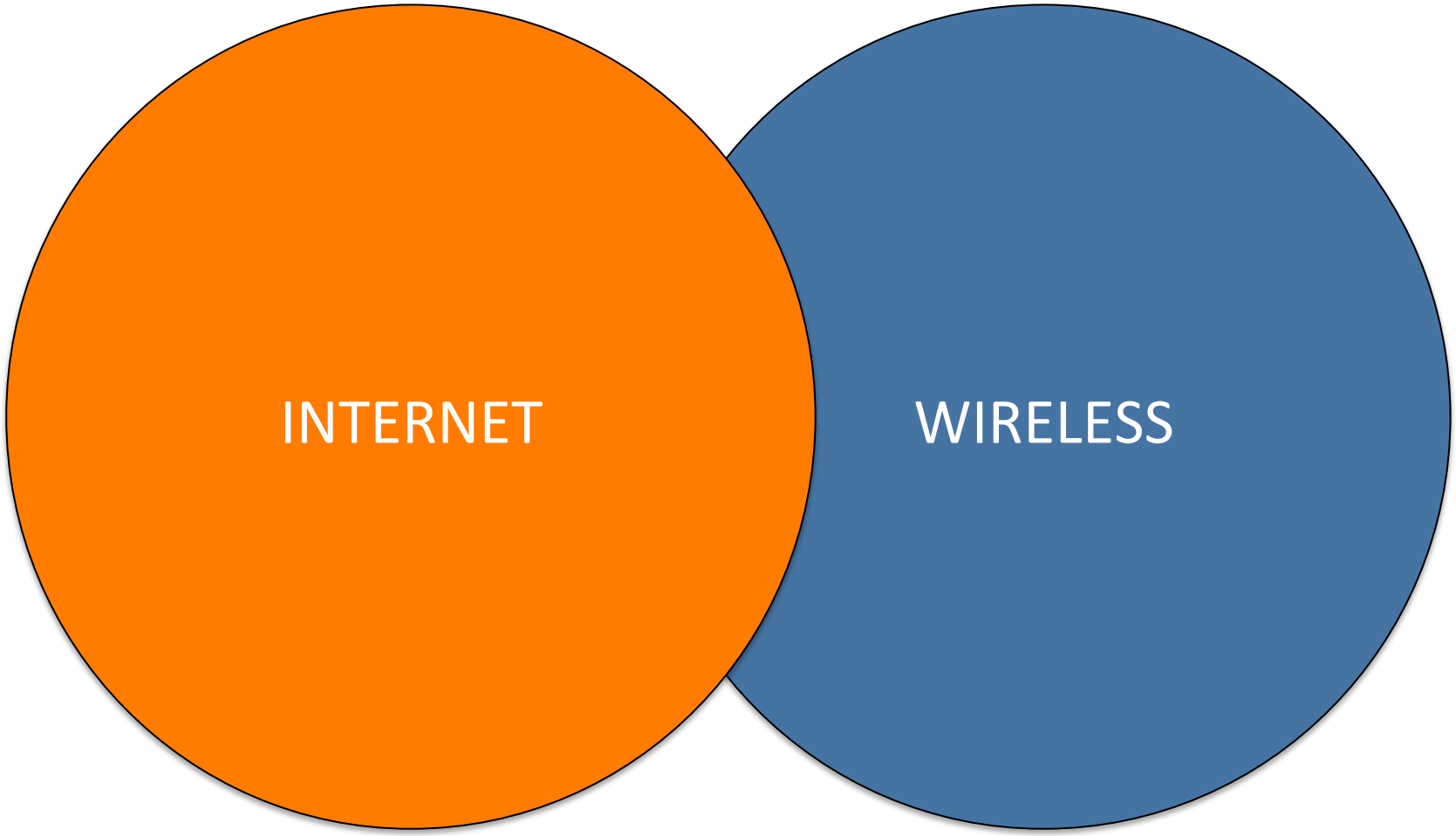
Some Recent/Hot Trends

- Towards more decentralization
  - Spontaneous wireless networks (SWN)
  - Peer-to-Peer networks (P2P)
- Towards more content cache near the user
  - Content Delivery Networks (CDN)
  - Information Centric Networking (ICN)
- Towards less state at the Internet Core
  - LISP/RRG
- Towards more (re)deployment flexibility
  - Software Defined Networks (SDN)

Content of this Chapter

- ❖Spontaneous Wireless Networks
- ❖P2P Networks
- ❖CDN and ICN
- ❖LISP & RRG

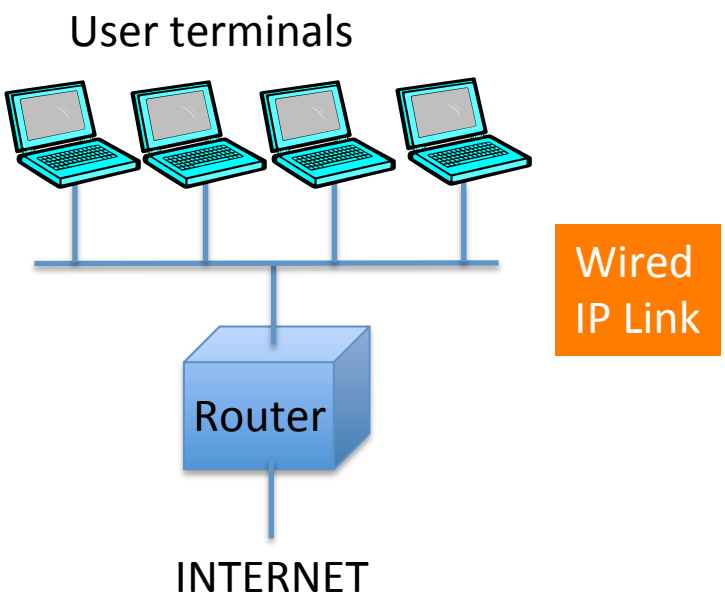
The Big Picture: a Giant Collision



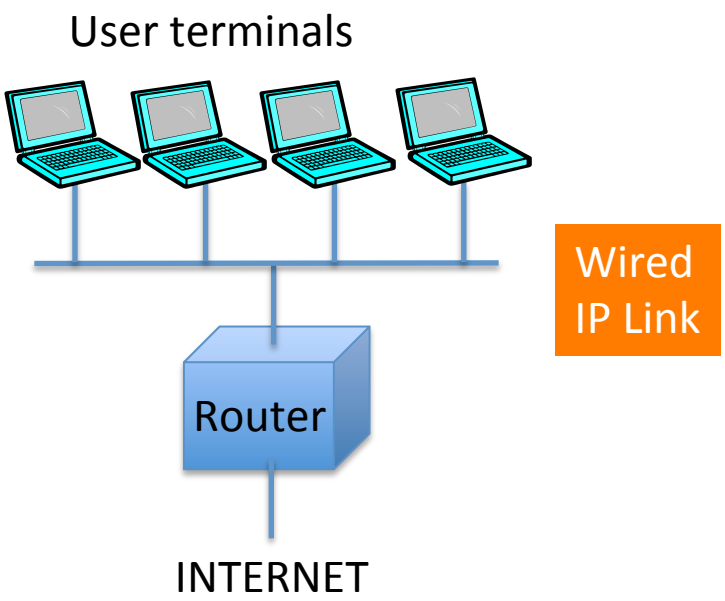
Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

IP: From Wired to Wireless…

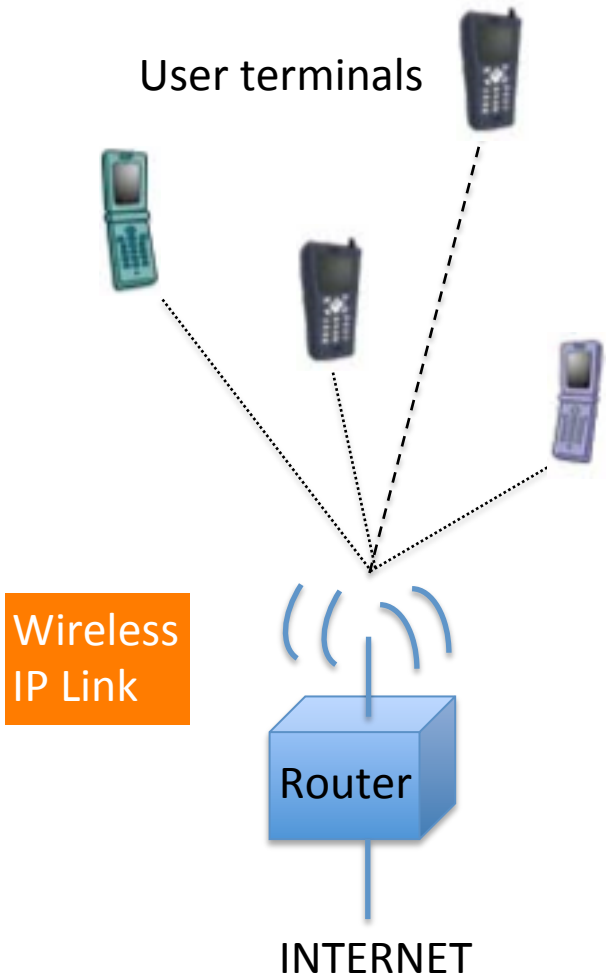


IP: From Wired to Wireless…

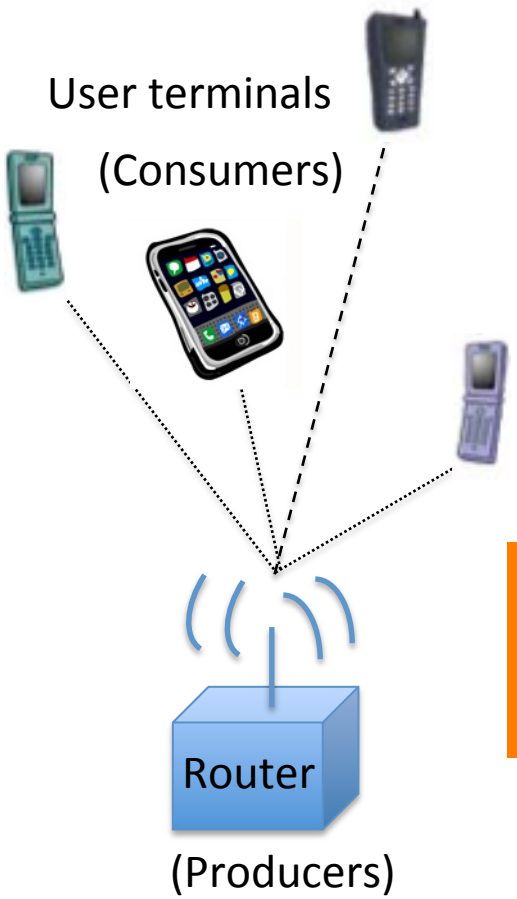


Wired IP Link

Wireless IP Link

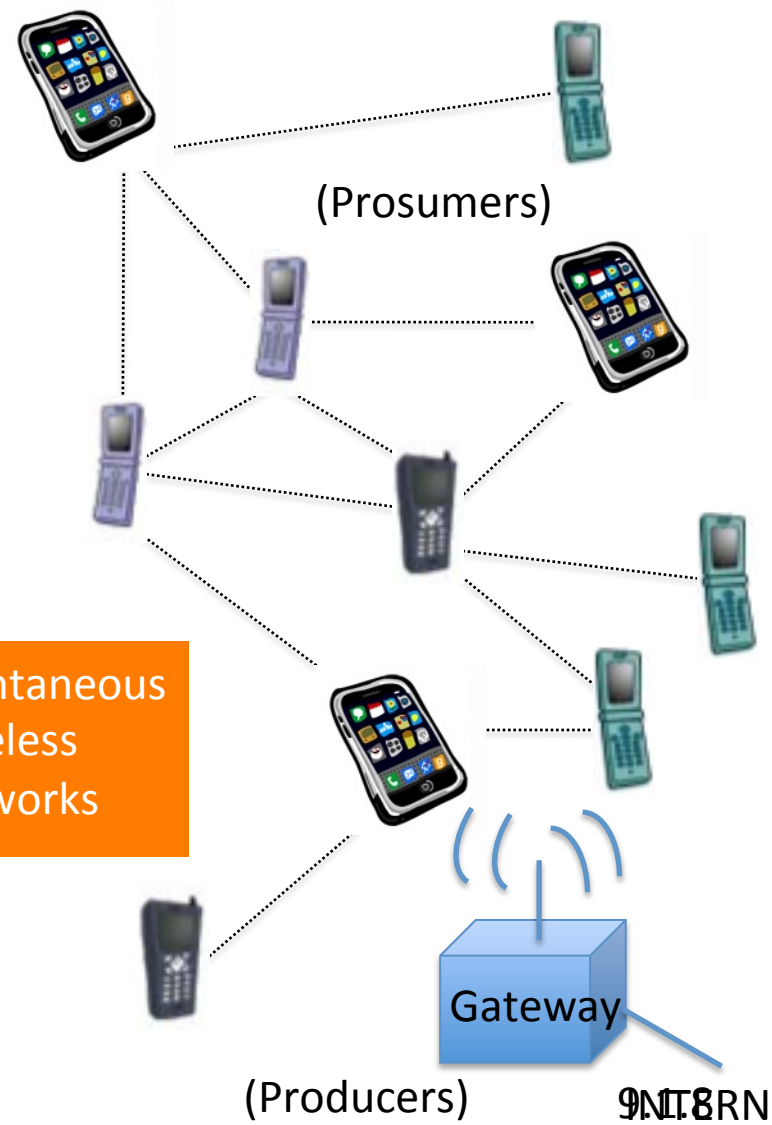


… Towards Wireless Prosumers



Single-hop Wireless Networks

Spontaneous Wireless Networks

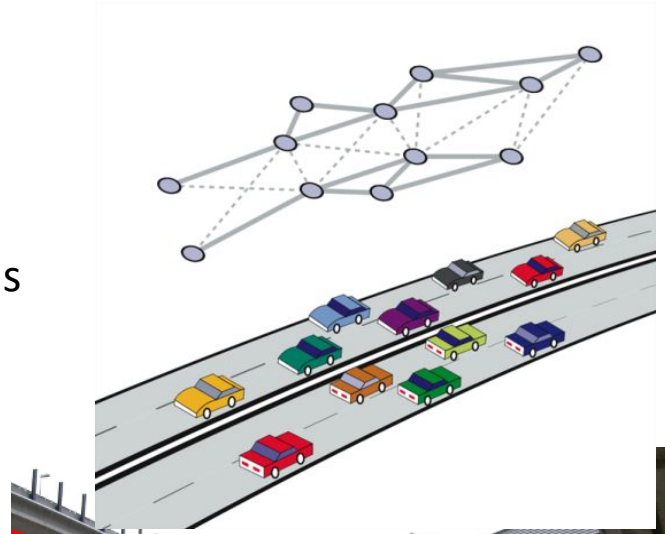


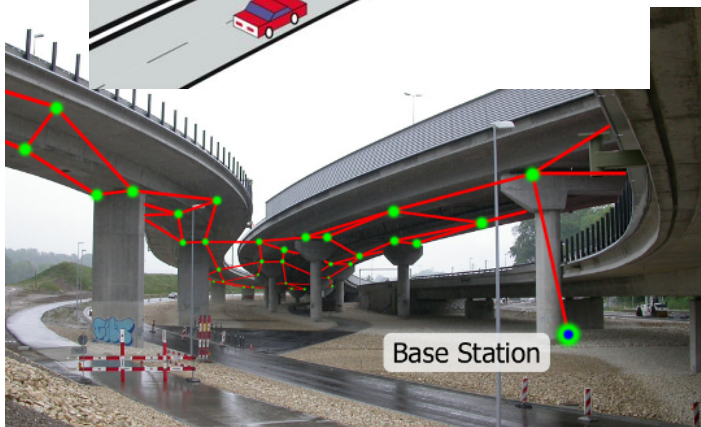
Spontaneous Wireless Networking

Self-organized wireless networking, multi-hop, with or without help from infrastructure.

- Ad hoc networks
- Wireless Mesh networks
- Sensor/actuator networks
- Opportunistic networks, DTNs
- Vehicular Networks
- …

SWN:
Spontaneous
Wireless
Networks





Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

IP-Disruptive Wireless Networking?

Most protocols in the IPv4 or IPv6 suite designed w.r.t. some assumptions:

- Definable IP links
- End-to-end connectivity exists
- Routers are always on, if not dead
- Routers have reasonable memory, CPU, power capacity

IP-Disruptive Wireless Networking?

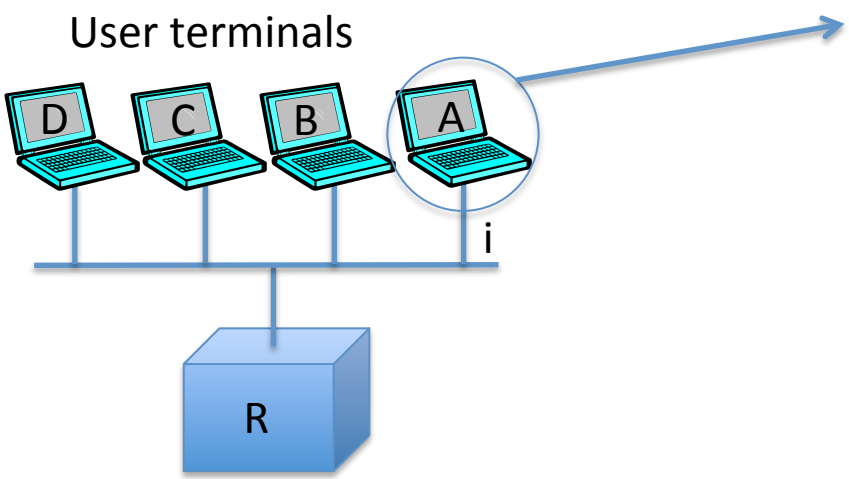
- Definable IP links
- End-to-end connectivity exists
- Routers are always on, if not dead
- Routers have reasonable memory, CPU, power capacity

Most spontaneous wireless networks break these assumptions, thus causing <u>disruption</u>

New/modified protocols needed

IP Links: the Atoms of the Internet

IP links were modeled after Ethernet:

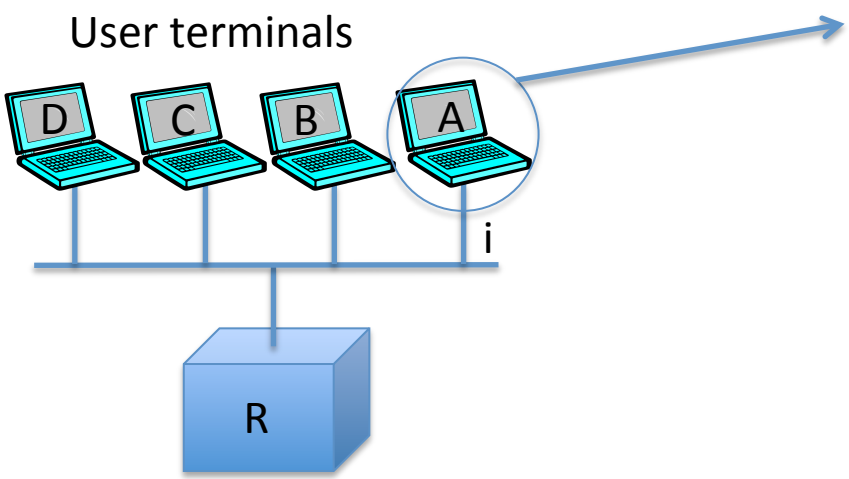


At time T, through interface *i* on the IP link:

Symmetry
Stability
Transitivity

IP Links: the Atoms of the Internet

IP links were modeled after Ethernet:



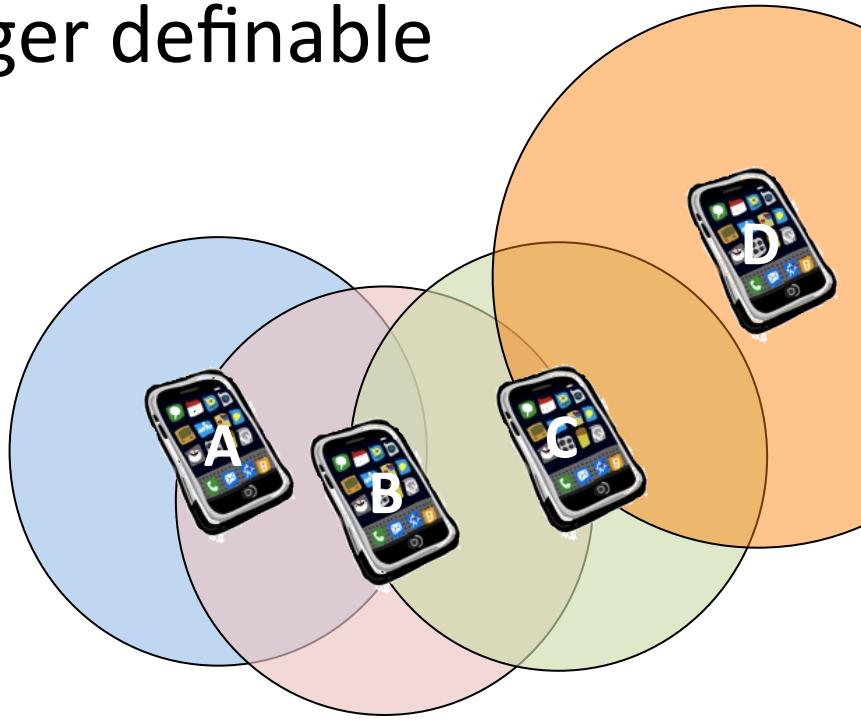
At time T, through interface *i* on the IP link:

Symmetry
Stability
Transitivity

Definable IP links with predetermined properties

Spontaneous Wireless Networks

DISRUPTION: links are no longer definable



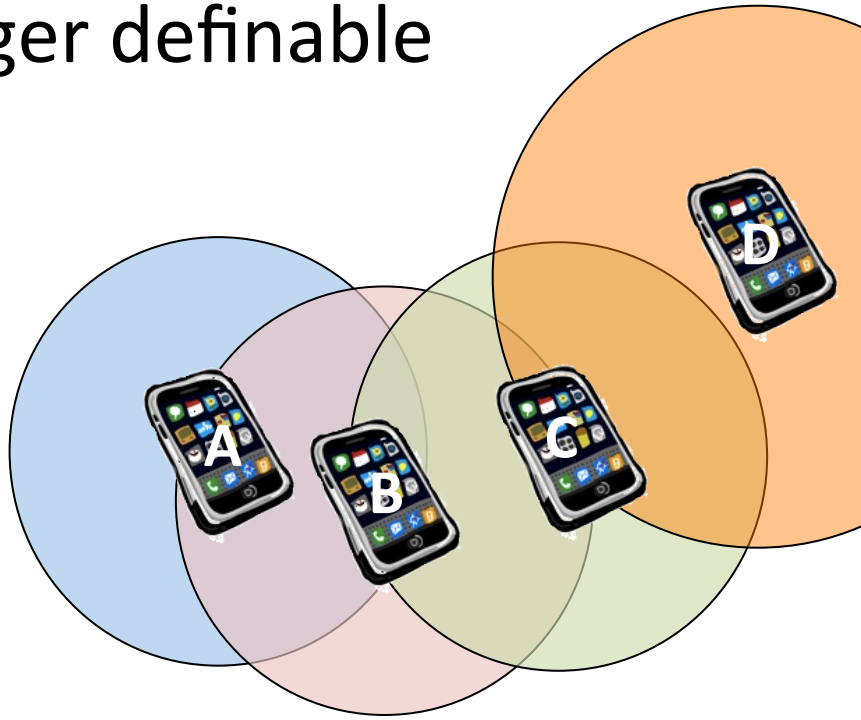
At time T, through interface *i* :

Assymmetry
Time-variation
Non-transitivity

Undetermined links : the IETF has a heart attack

Spontaneous Wireless Networks

DISRUPTION: links are no longer definable



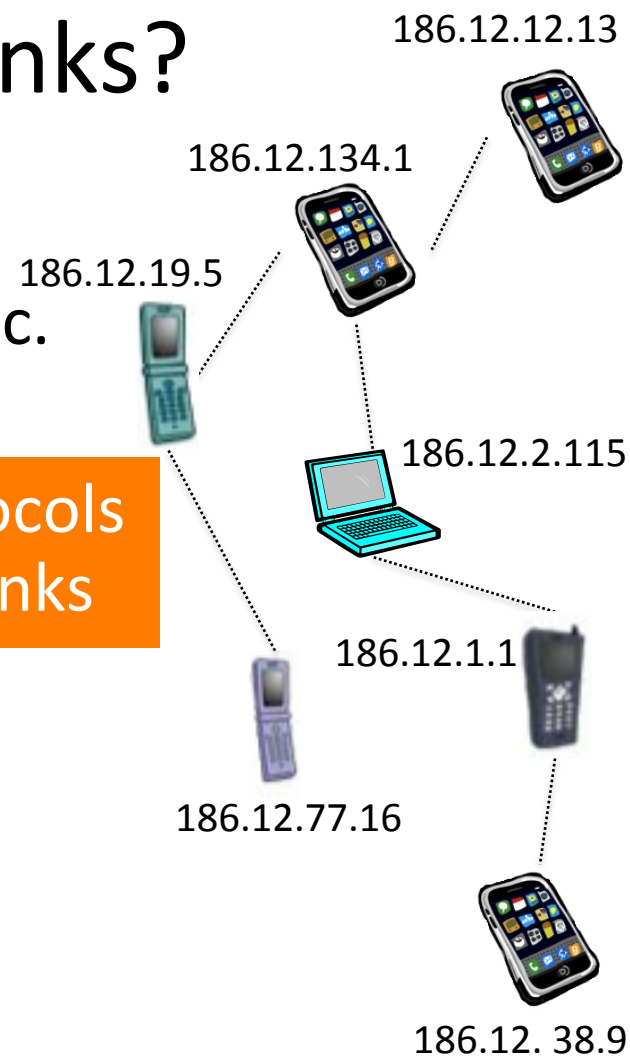
At time T, through interface *i* :

Assymmetry
Time-variation
Non-transitivity

Undetermined links : the IETF has a heart attack

Autoconfiguration on undetermined links?

Prior to routing, devices need appropriate IPv4/v6 configuration: IP address, prefix etc.



Analysis: standard autoconfiguration protocols (ND, SLAAC, DHCP) fail on undetermined links

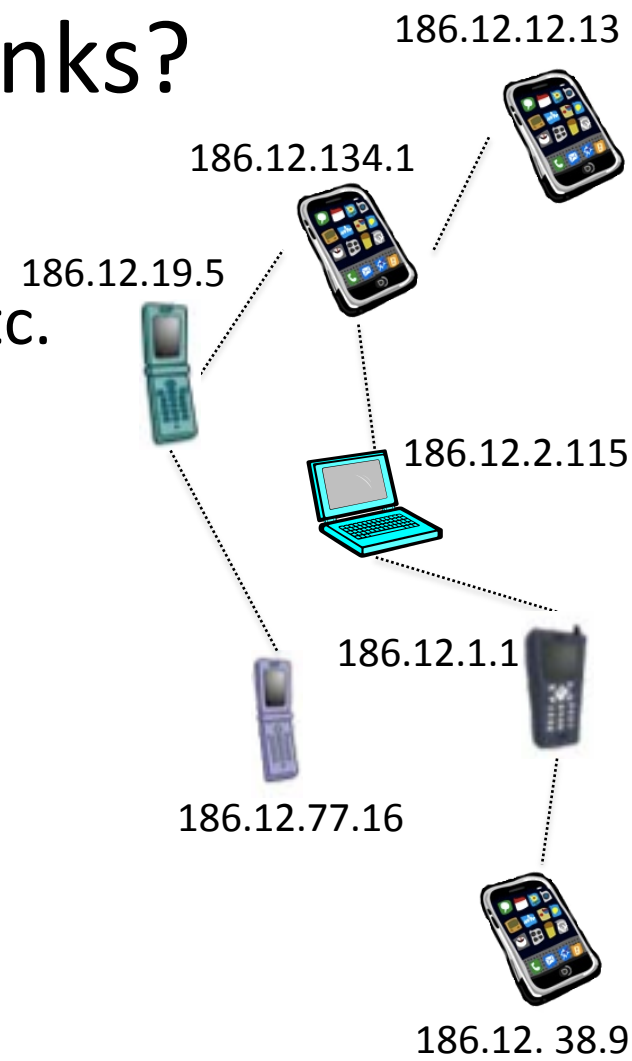
Assymmetry
Time-variation
Non-transitivity

Autoconfiguration on undetermined links?

Prior to routing, devices need appropriate IPv4/v6 configuration: IP address, prefix etc.

RFC 5889

Standard configuration of IP interfaces to undetermined links = ~~on-link prefix~~



Autoconfiguration on undetermined links?

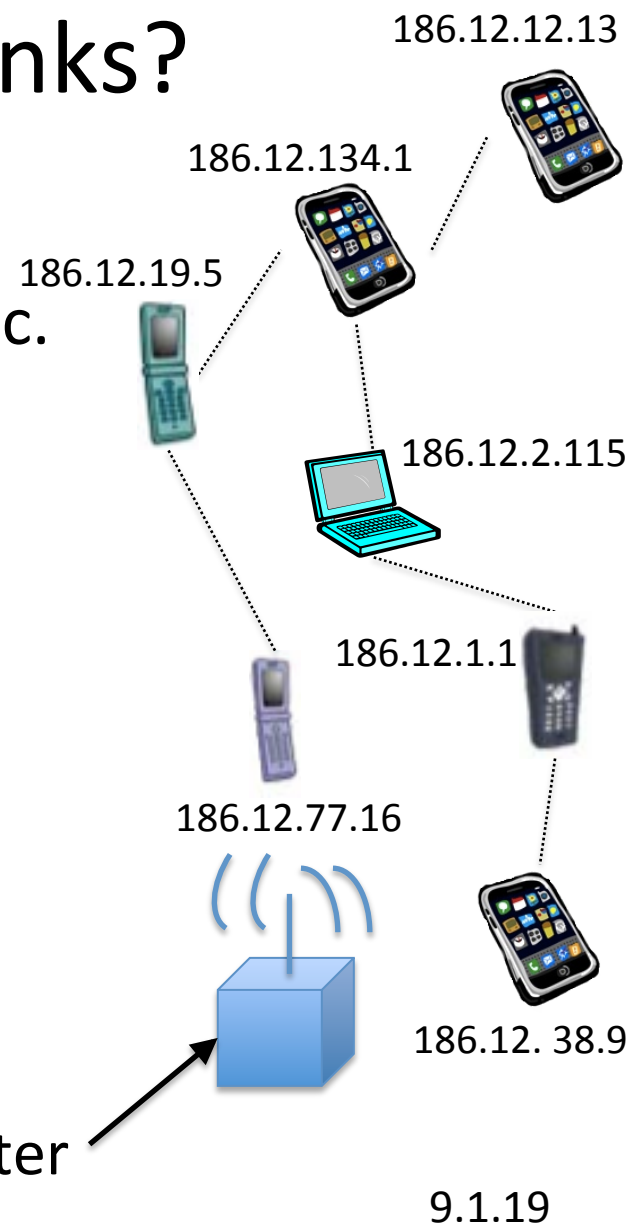
Prior to routing, devices need appropriate IPv4/v6 configuration: IP address, prefix etc.

RFC 5889

Standard configuration of IP interfaces to undetermined links = ~~on-link prefix~~

RFC 6775

6lowpan ND for hosts in a spontaneous wireless network connected to a border router

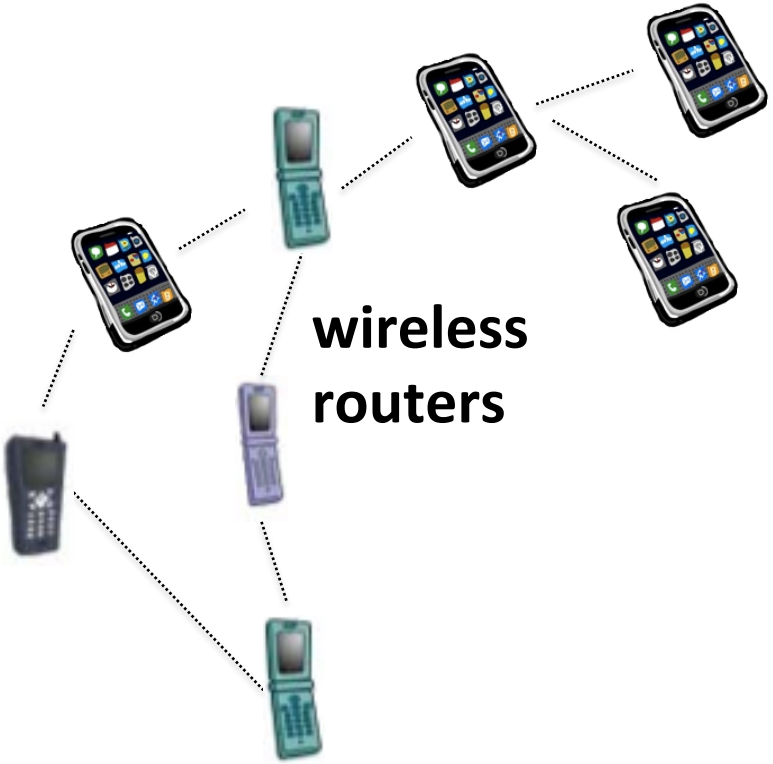


Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

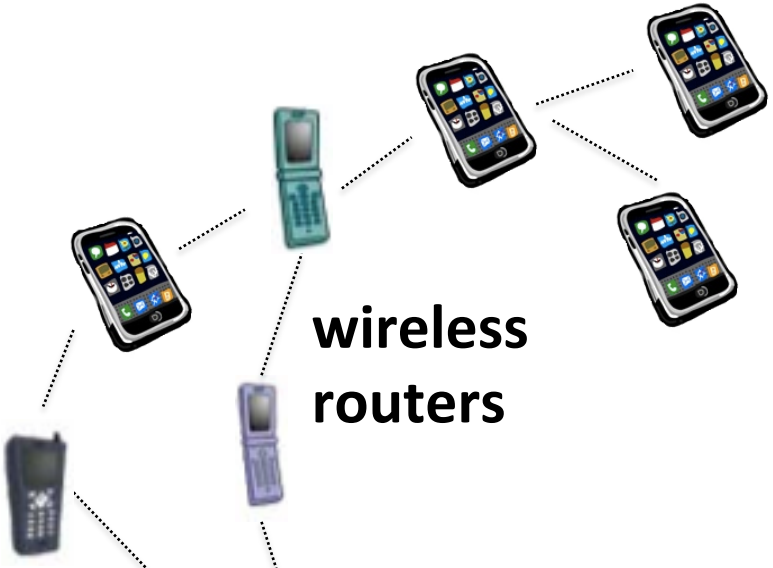
Wireless Ad Hoc & Mesh Networks

- Example of wireless ad hoc network:
  - Multihop wireless networking among smartphones, without access points involved

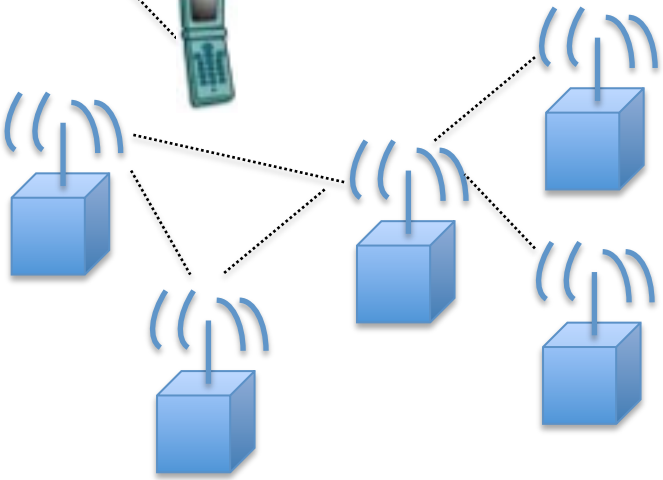


Wireless Ad Hoc & Mesh Networks

- Example of wireless ad hoc network:
  - Multihop wireless networking among smartphones, without access points involved

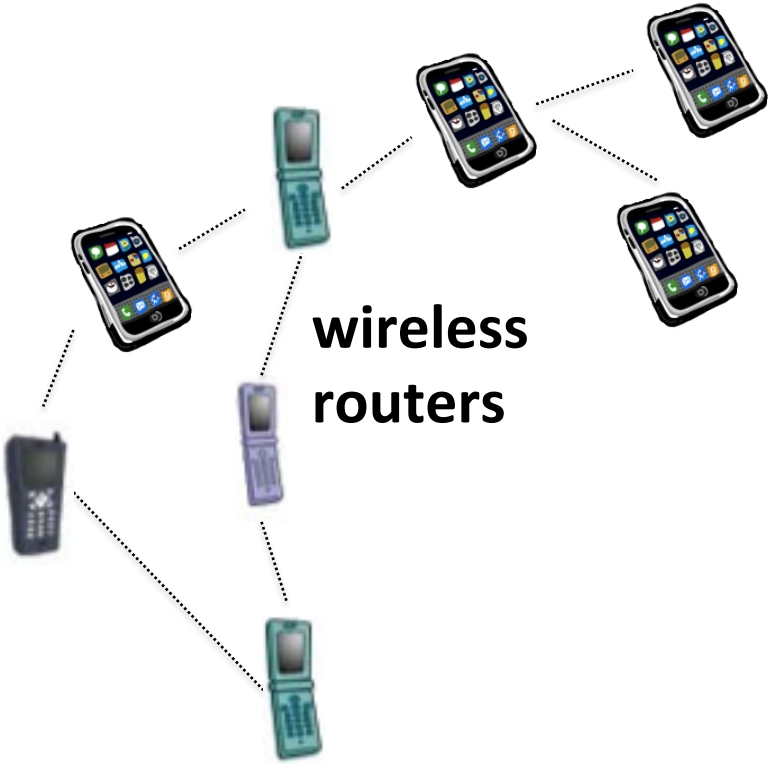


- Example of wireless mesh network:
  - Multihop wireless networking among openWRT wireless routers, e.g. Freifunk (www.freifunk.net)



Routing in Ad Hoc Networks?

- More dynamic links, less bandwidth (wireless)
- Traditional routing protocols (eg **OSPF**) fail
- New routing schemes are needed (e.g. **OLSR, AODV**)



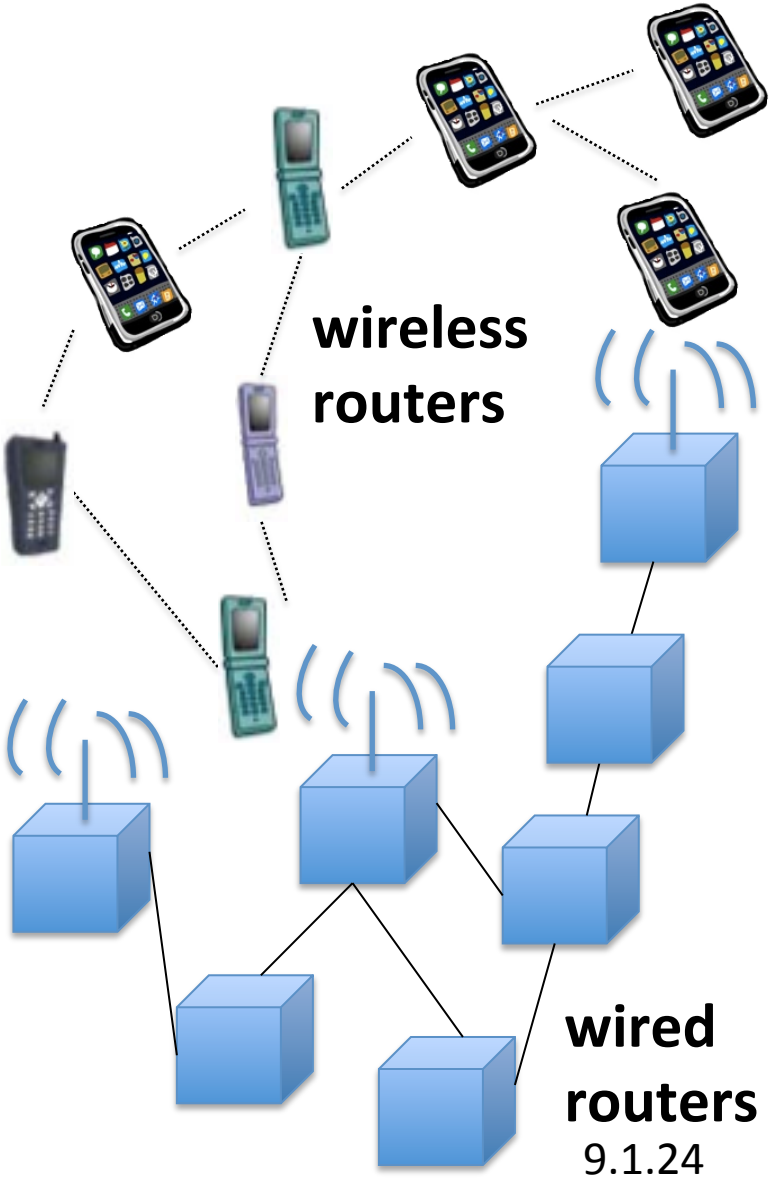
Routing in Hybrid Networks?

<u>Analysis</u>

How to combine OLSR agility with OSPF compatibility?

- Optimized flooding & backup
- Selected & lighter adjacencies
- Reduced topology
- Reduced hello redundancy

RFCs 5449, 5820, 5614



Why are SWNs not Widely Deployed yet?

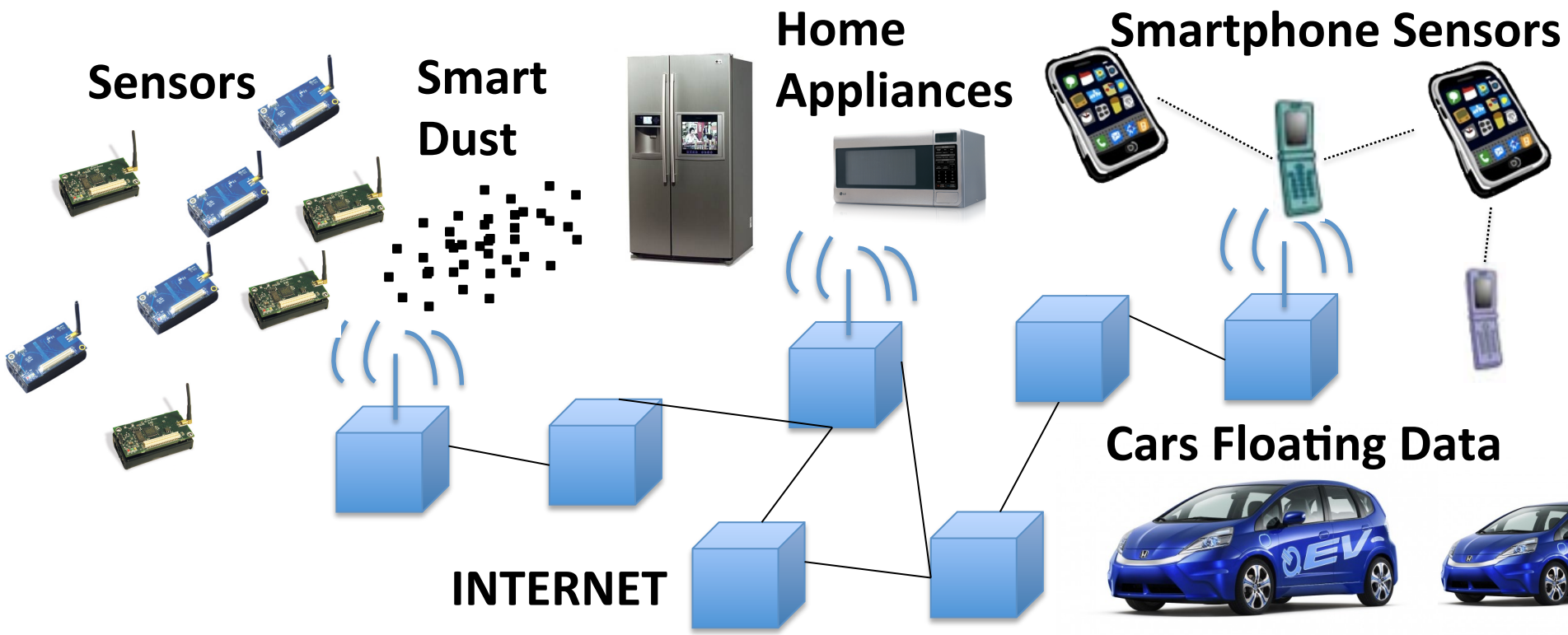
- Reasons include:
  - Suboptimal MAC layers
  - Limited radio bandwidth
  - Disruption of IP protocol suite

Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

The Internet of Things (IoT)

The buzz word for an upcoming avalanche of connected devices beyond hosts and routers



IoT: Slimming Down the IP stack

<u>DISRUPTION</u>: same as ad hoc networks, plus stringent memory, frame-size and power management constraints.

- 8-16 bits micro-controllers, a few kB of RAM/ROM, i.e. **less than the first computers on the ARPANET!**
- IEEE 802.15.4 payload is 81 octets, i.e. **16 times less than IPv6 MTU: 1280 octets!**
- Agressive sleep modes, duty cycles, i.e. **unresponsive routers are not dead!**

Routing in Sensor Networks?

Due to memory constraints, not enough space to store traditional routing database

New/modified protocols needed

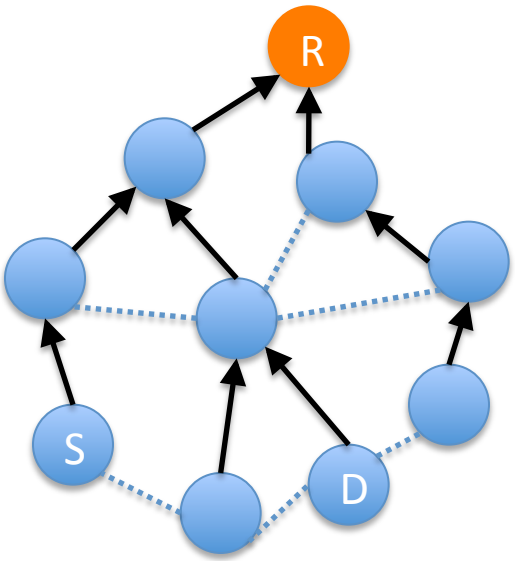
Routing in Sensor Networks?

Due to memory constraints, not enough space to store traditional routing database

New/modified protocols needed

e.g. assumptions on data traffic patterns led to tree-based IPv6 routing standard **RPL** (RFC 6550).

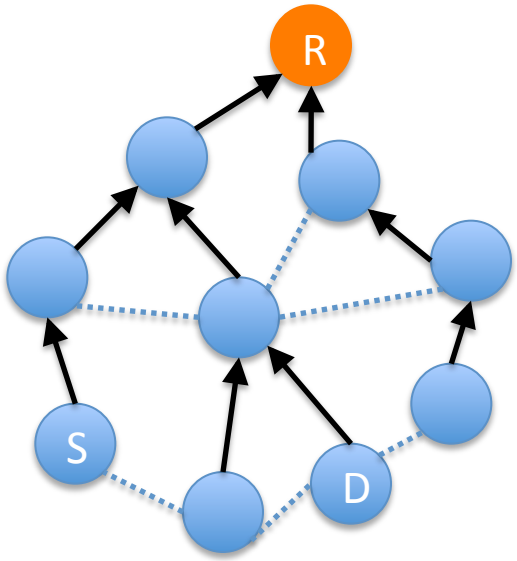
Convergecast with RPL



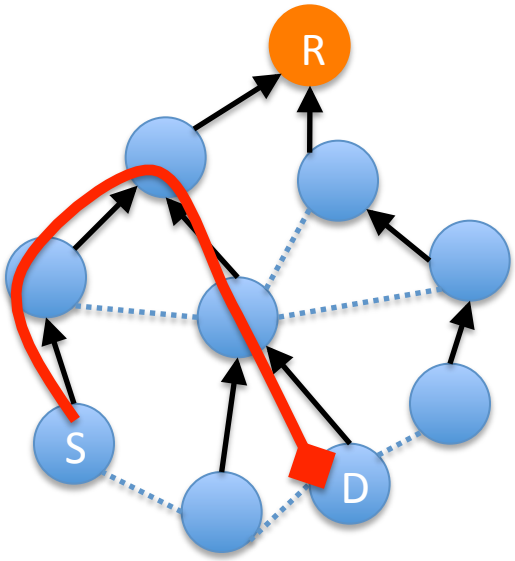
Directed Acyclic Graph (DAG)

RPL Limitations

e.g. very suboptimal routing in home and building automation uses cases (non-convergecast traffic)



DAG

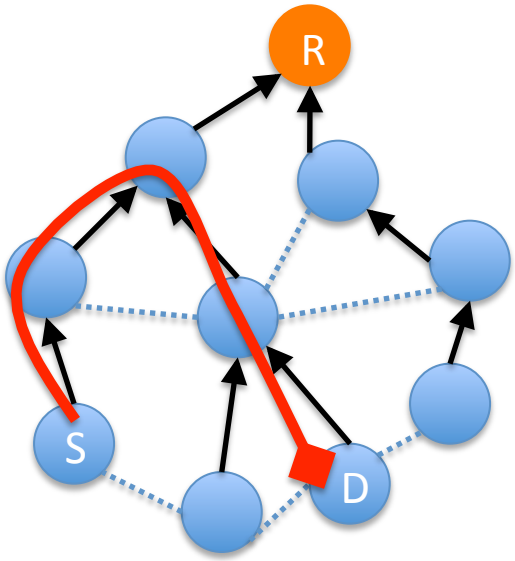


Path with basic RPL

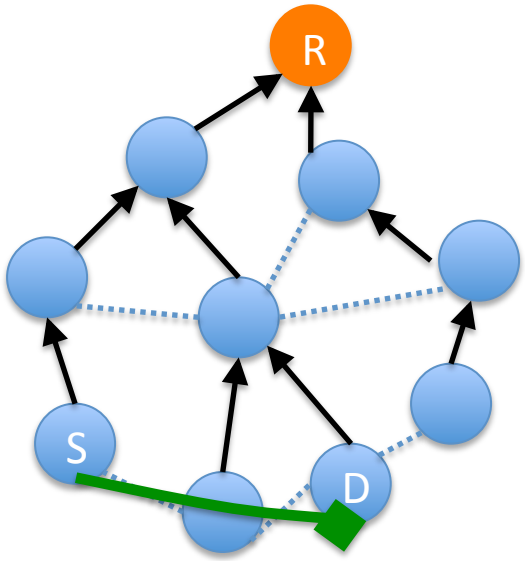
- Much longer paths
- Congestion near root

P2P-RPL Protocol Extension

RFC 6997: extension for on-demand sensor-to-sensor routing



Path with basic RPL

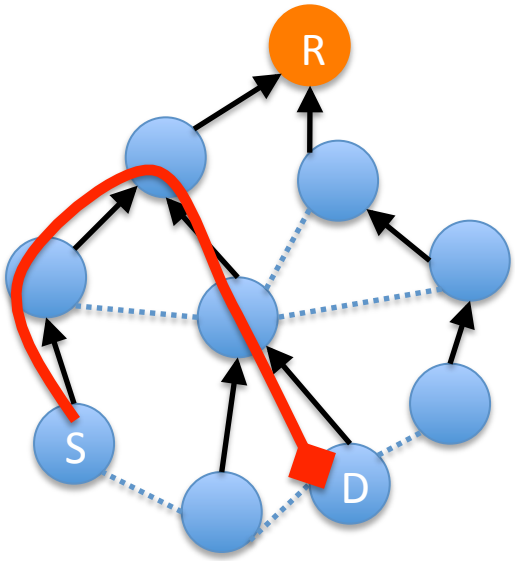


Path with P2P extension

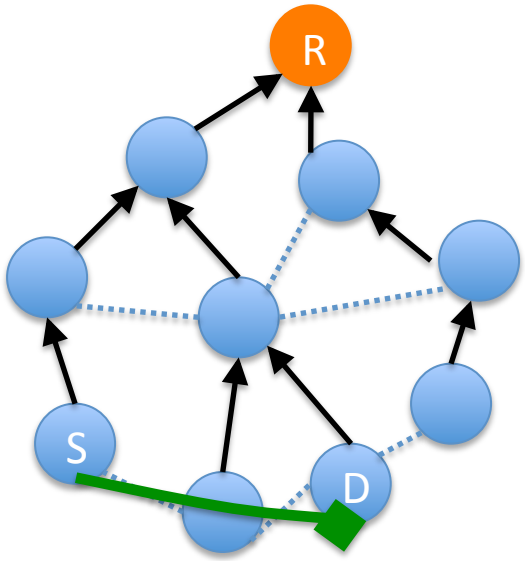
- Path request dissem.
- Path reply unicast

P2P-RPL Protocol Extension

RFC 6997: extension for on-demand sensor-to-sensor routing



Path with basic RPL



Path with P2P extension

Limitations:

- Path acquisition delay
- Unpredictable control traffic
- Limited network diameter

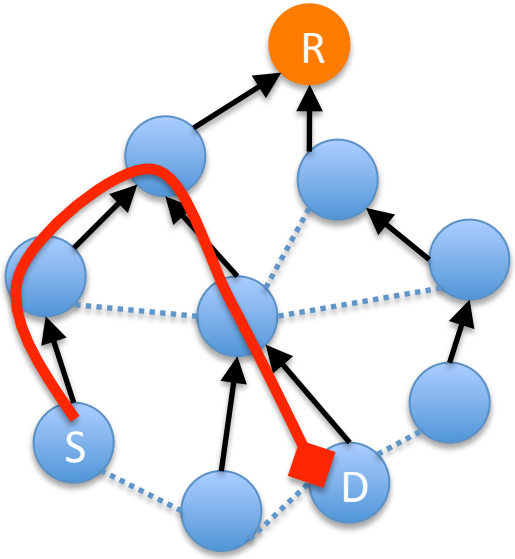
ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009
2010
2011
2012
2013

Phase 1: Problem statement (2009-2010)



<u>Analysis</u>:

- Suboptimal paths
- Congested root

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design (2011-2012)

This phase has to show enough feasability + community interest

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design (2011-2012)

Use case: fixed home/building automation nodes, potentially very constrained
Requirements: shorter paths, on demand and NOT proactive, small state

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

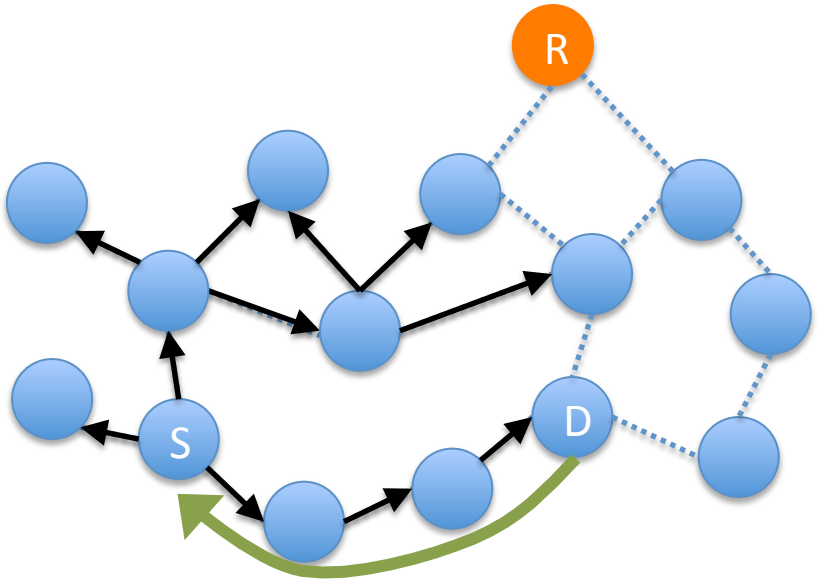
2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design (2011-2012)

Idea: why not a complemenary reactive component, building temporary DAG
Pros: fits requirements, rather well studied technique
Cons: Flooding cost (curbed with fixed nodes + RPL metrics framework)



"good enough" criteria to limit the growth of the temporary DAG

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design (2011-2012)

Idea: why not a complemenary reactive component, à la AODV?
Pros: fits requirements, rather well studied technique
Cons: Flooding cost (curbed with fixed nodes + RPL metrics framework)

- *draft-dt-roll-p2p-rpl-01, Individual Internet Draft, 2010.*
- *draft-ietf-roll-p2p-rpl-02, Working Group Internet Draft, 2011*

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design

Phase 3: Protocol tests (2011-2012)

This phase has to demonstrate running code and protocol performance

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

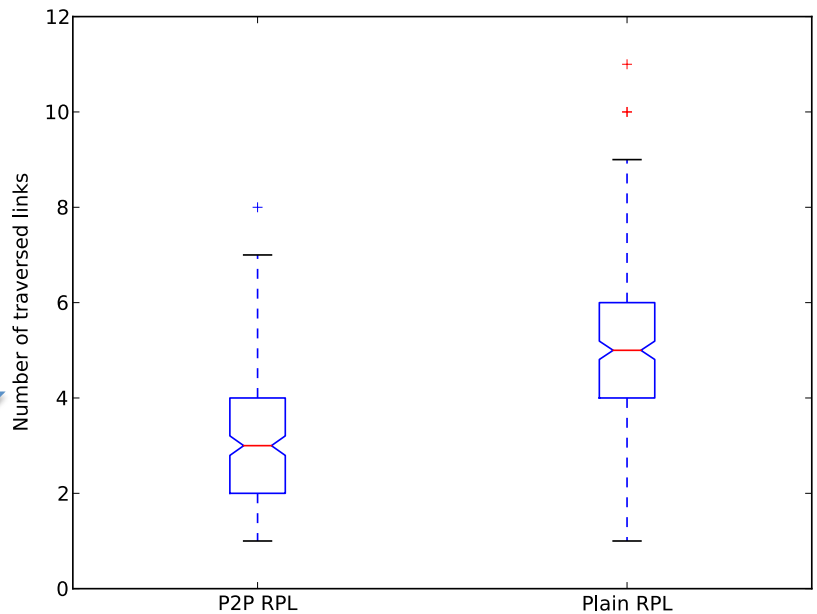
2010

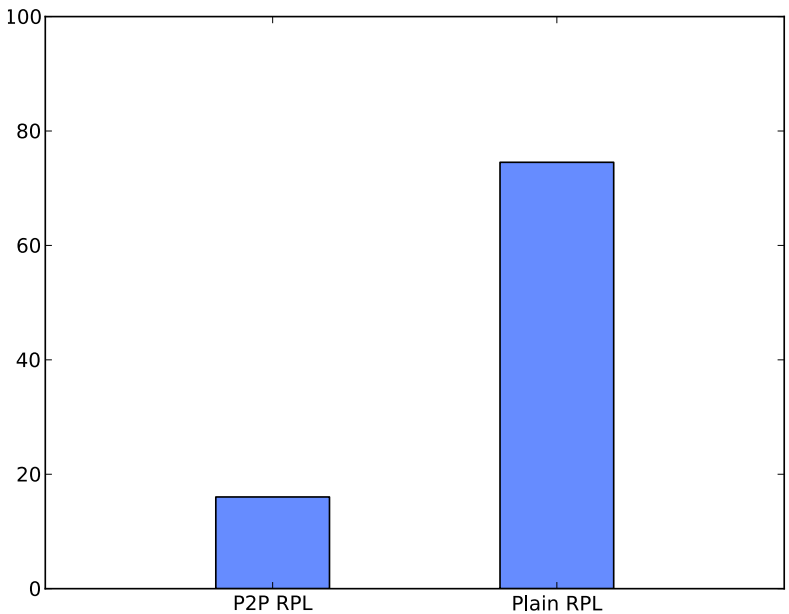
2011

2012

2013

Free open source at http://contiki-p2p-rpl.gforge.inria.fr
Experimental results on MSP430 nodes, 802.15.4 at 2,4GHz





Significant path length reduction

Significant traffic reduction near root

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design

Phase 3: Protocol tests & interop (2011-2012)

This phase has to demonstrate specification quality

independent implementations interoperate

INRIA implementation ⟷ Sigma Designs implementation

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design

Phase 3: Protocol tests & interop

Phase 4: IESG area directors reviews (2013)

This phase checks compatibility with all other existing RFCs

- *Some AD comments: more rationale for recommended default values, IANA registry needs, integrity of payload.*

- *Subsequently produced a new revision of the specification.*

ZOOM

Freie Universität Berlin

Standardizing P2P-RPL

2009

2010

2011

2012

2013

Phase 1: Problem statement

Phase 2: Algorithm design

Phase 3: Protocol tests & interoperability

Phase 4: IESG area experts reviews

Phase 5: RFC publication (July 2013)

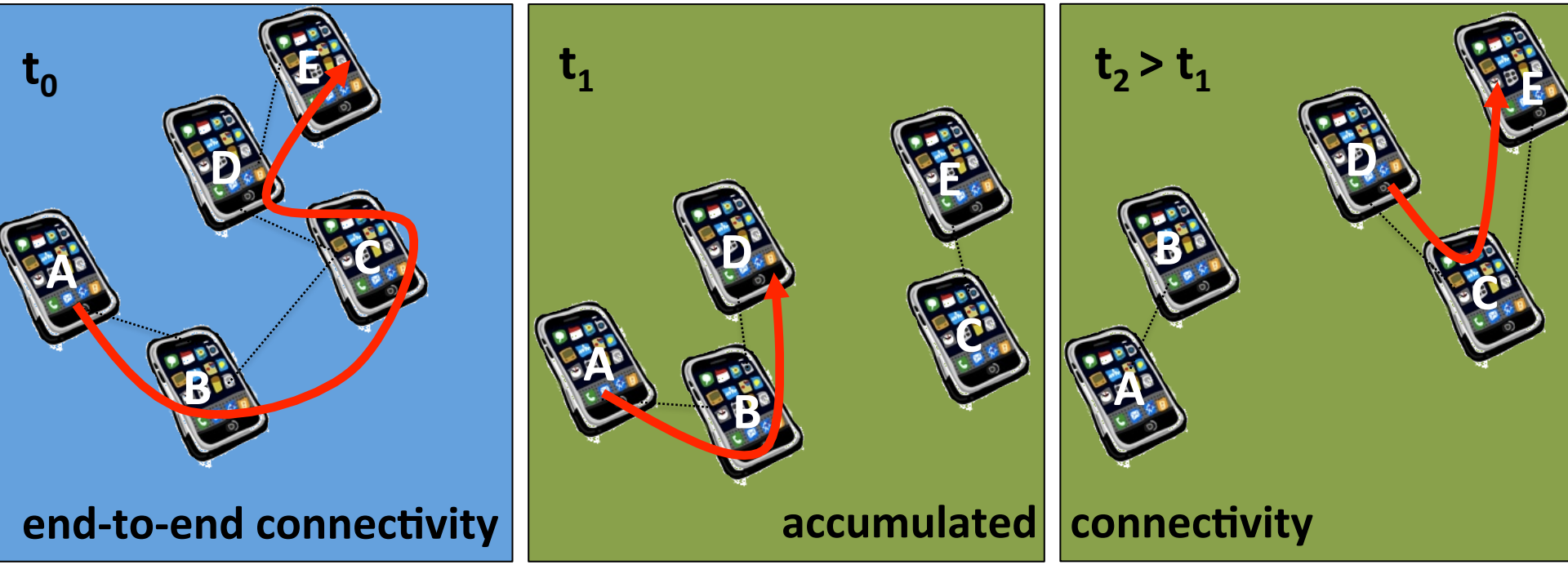
This stage validates all of the above, thus ensuring overall compatibility & performance

Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

Delay Tolerant Networks (DTN)

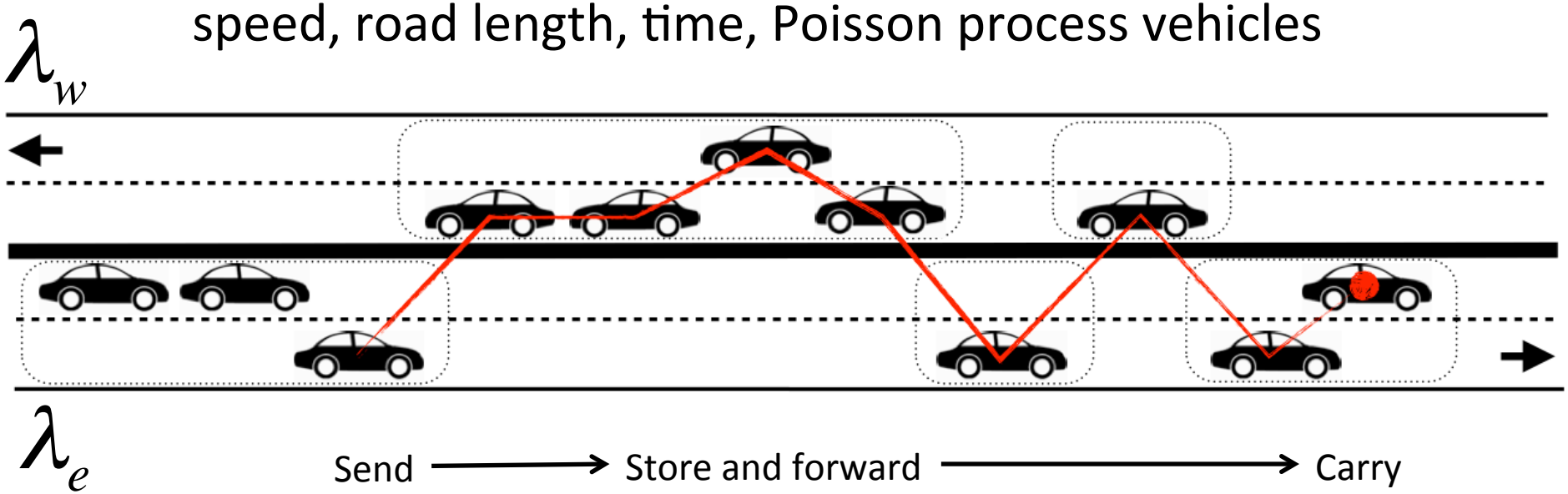
DISRUPTION: no end-to-end connectivity



- Over time accumulated topology becomes connected
- Traditional store-forward cannot use such cumulative connectivity
- DTN = exploiting store-carry-forward strategies

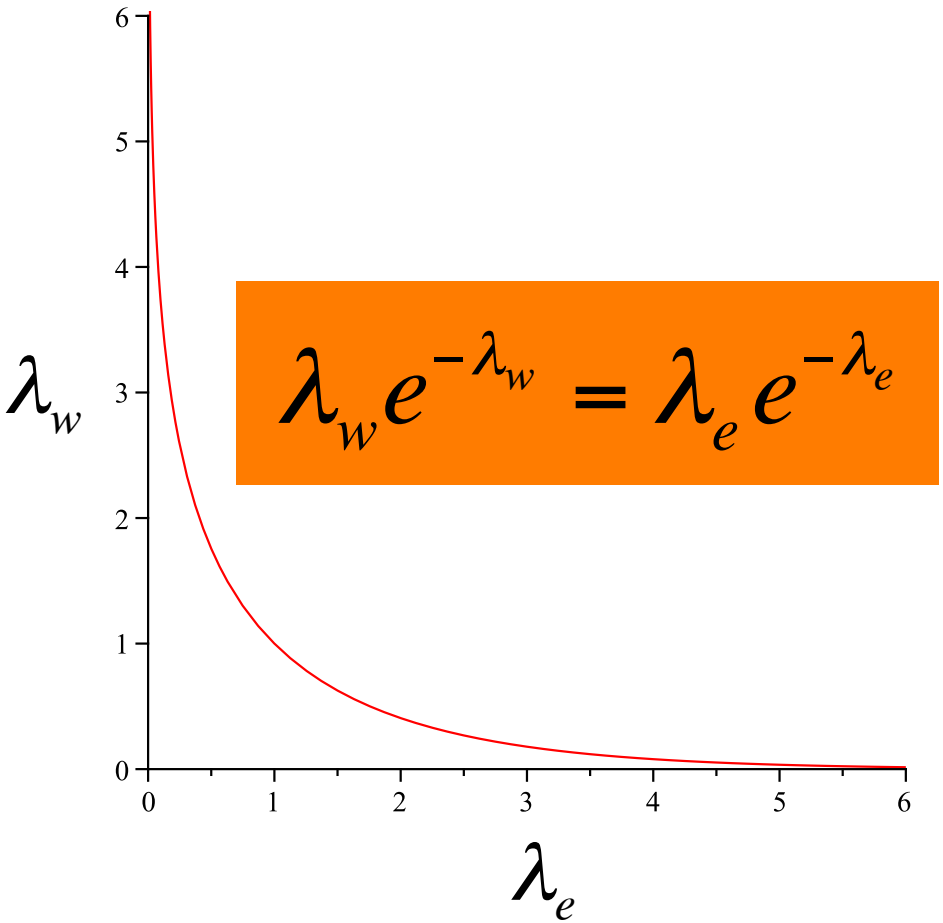
Information Propagation Speed?

- Store-Carry-Forward in one direction (e.g. with ProPHET)
- e.g. emergency signaling (accident ahead)
- Model: 1D, 2 lanes, constant vehicle speed, infinite radio speed, road length, time, Poisson process vehicles



Q: How fast does the message propagate?

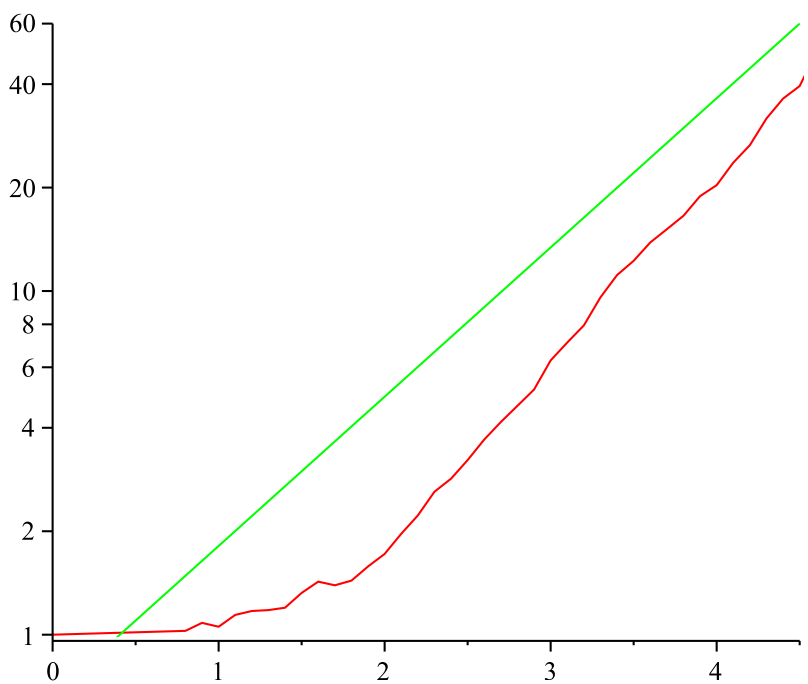
Analysis: Threshold Characterization for Information Propagation Speed



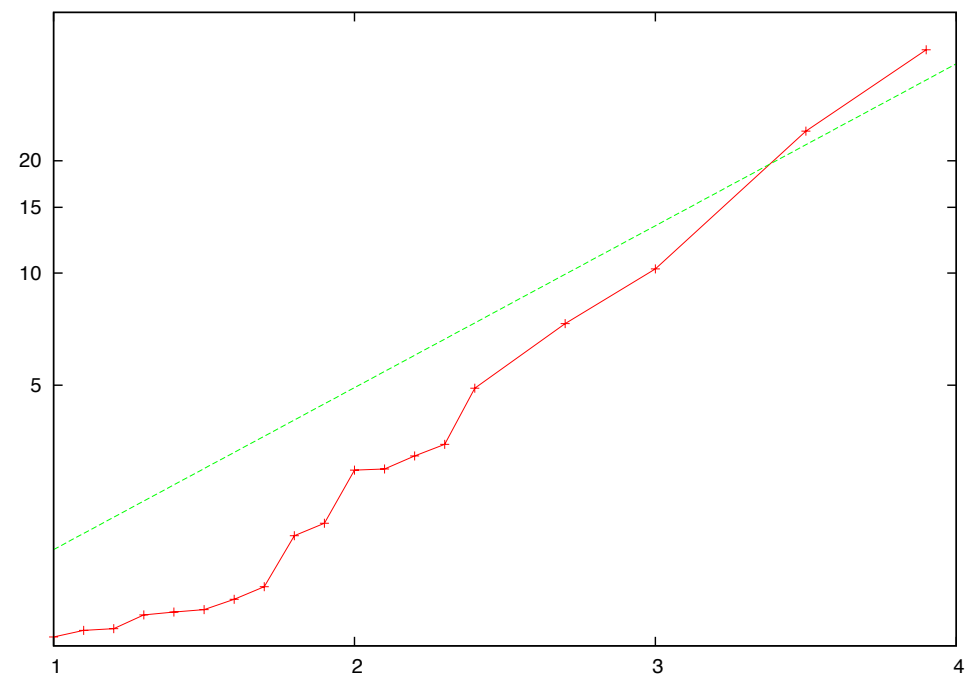
- Below threshold: stuck at vehicule speed: $v_p = v$
- Above threshold: much faster than vehicule speed:

$$v_p \approx 2vC(\lambda_e, \lambda_w)e^{\lambda_e + \lambda_w}$$

Observed Behavior (Simulations)



Maple Simulations



ONE Simulations

Vp for λe = λw, versus the total vehicle density λe + λw , in semi-log scale, compared to the theoretically predicted asymptotic exponential growth.

Content of this section

- Spontaneous Wireless Networks
  - IP-disruptive Wireless Networks
  - Wireless Ad Hoc & Mesh Networks
  - Internet of Things & Sensor Networks
  - Delay Tolerant Networks
- Perspectives for Spontaneous Wireless networks

Perspectives for SWN

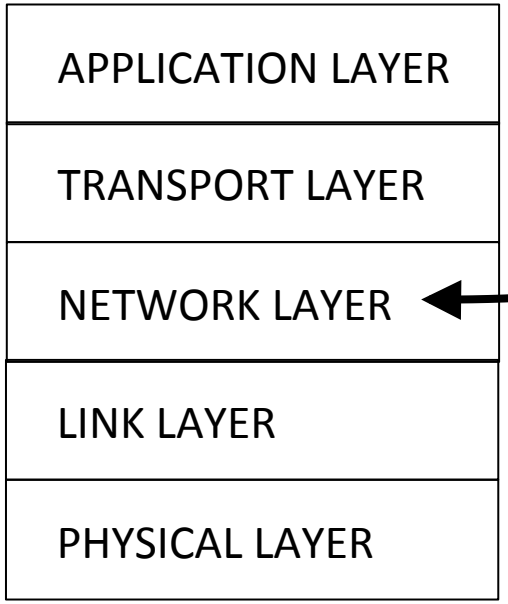
Spontaneous wireless networks are IP-disruptive. How to deal with them?

APPLICATION LAYER
TRANSPORT LAYER
NETWORK LAYER
LINK LAYER
PHYSICAL LAYER

- Top layer developments
- Intra-layer optimizations
- Cross-layer optimizations
- Adaptation layer developments

Perspectives for SWN

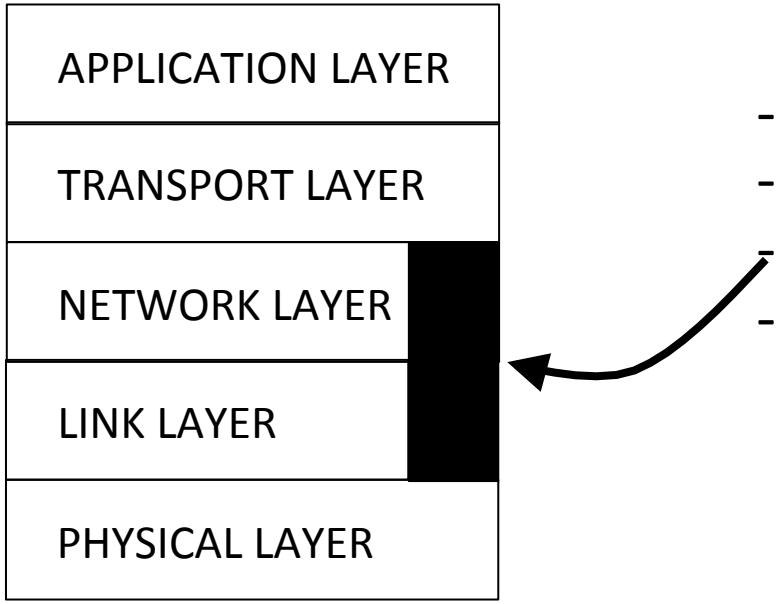
Spontaneous wireless networks are IP-disruptive. How to deal with them?



- Top layer developments
- Intra-layer optimizations
- Cross-layer optimizations
- Adaptation layer developments

Perspectives for SWN

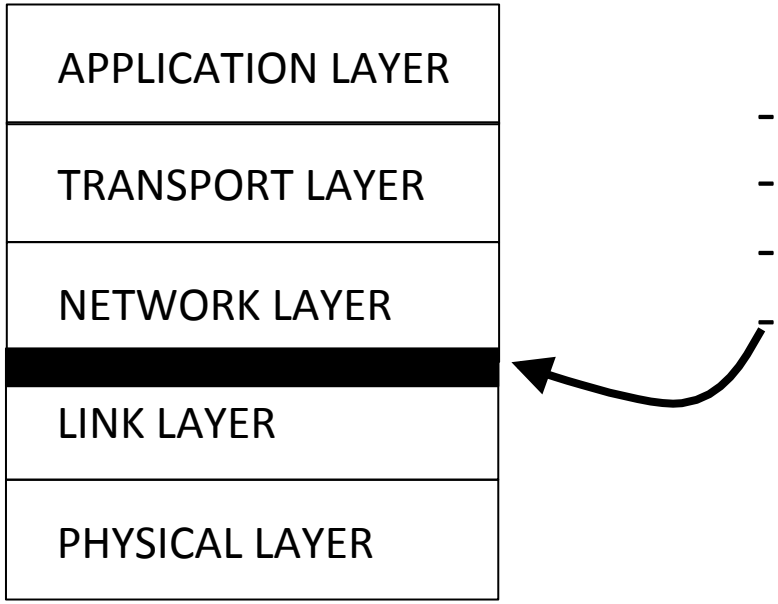
Spontaneous wireless networks are IP-disruptive. How to deal with them?



- Top layer developments
- Intra-layer optimizations
- Cross-layer optimizations
- Adaptation layer developments

Perspectives for SWN

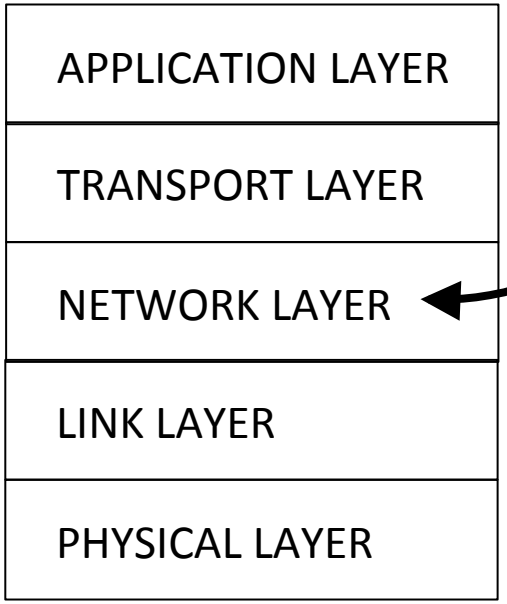
Spontaneous wireless networks are IP-disruptive. How to deal with them?



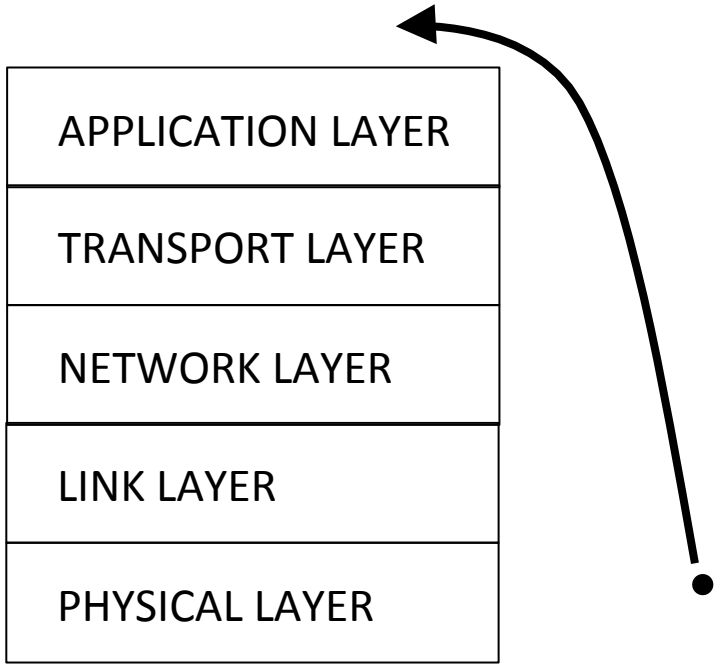
- Top layer developments
- Intra-layer optimizations
- Cross-layer optimizations
- Adaptation layer developments

A Few Related Open Problems

- Scalable and practical routing in spontanenous wireless networks.

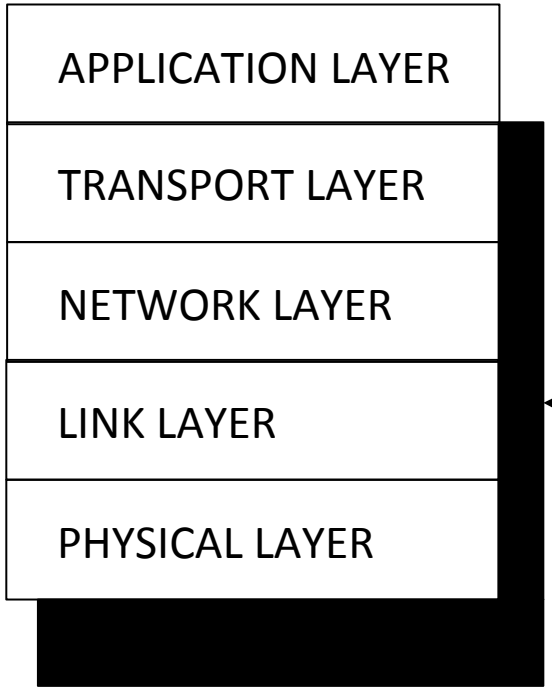


A Few Related Open Problems



- High performance opportunistic networking.

A Few Related Open Problems

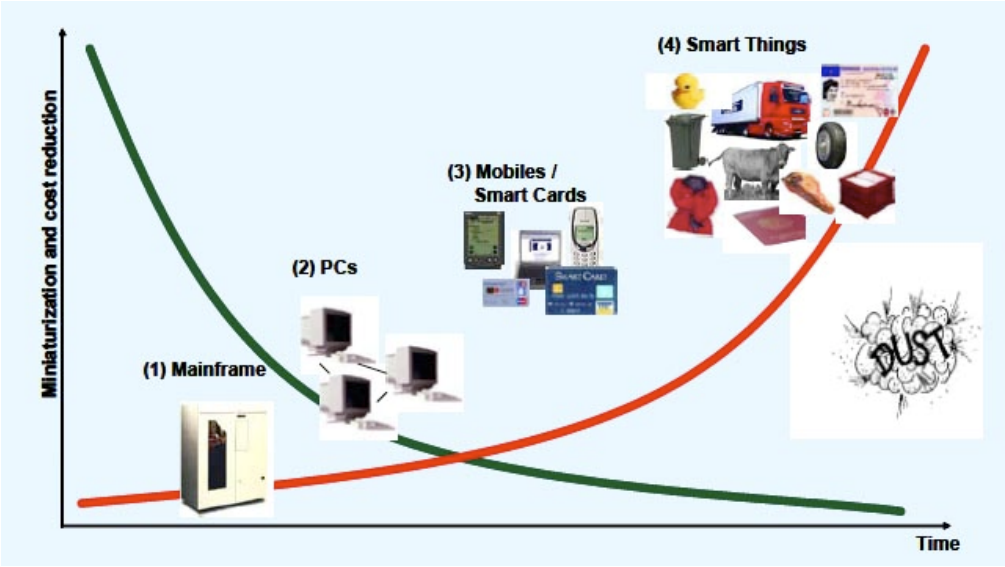


- Optimized software platform for the Internet of Things.

An operating system for the IoT

- The vision of IoT:
  - All our objects are connected
  - Connectivity is spontaneous

- Application scenarios:
  - Smart Metering
  - Home/Building Automation
  - Smart City
  - Smart Grid
  - ehealth
  - Logistics
  - …



An operating system for the IoT

- Challenge 1: connected objects are heterogeneous
  - From 8bit microcontrollers to more powerful smartphones or routers with 32bit architecture
  - Various communication interfaces (mostly, but not limited to wireless networks)

- Challenge 2: most connected objects have constrained capacity
  - Slow CPU, often no FPU
  - Little memory, often no MMU
  - Limited energy resources

An operating system for the IoT

- Challenge 3: connectivity is spontaneous
  - Robustness and self-organization
  - Scalability

- Challenge 4: many IoT applications require advanced services from the OS
  - Real-time requirements
  - Multi-threading

Legacy Operating Systems

- Typical Real-Time Operating Systems:
  - FreeRTOS
  - QNX
  - RTLinux
- Problem: not designed for
  - energy-efficiency
  - constrained networks
- Traditional operating systems for WSN:
  - Contiki
  - TinyOS
- Problem: not a good fit because
  - Event-driven design
  - Single-threaded
  - Specialized programming language

Hello World in TinyOS

```
///////////
#include <stdio.h>
#include <stdlib.h>

module HelloworldM {
provides {
    interface Hello;
}

}
implementation {
    command void
Hello.sayhello() {
        printf("hello world!");
    }
}
```

the hello interface: Hello.nc:

```
///////////
interface Hello{
  command void sayhello();
}
///////////
```

Hello World in Contiki

```
#include "contiki.h"
#include <stdio.h> /* For printf() */
/*---------------------------------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);
/*---------------------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
PROCESS_BEGIN();

printf("Hello, world\n");
PROCESS_END();
}
/*---------------------------------------------------------------------*/
```

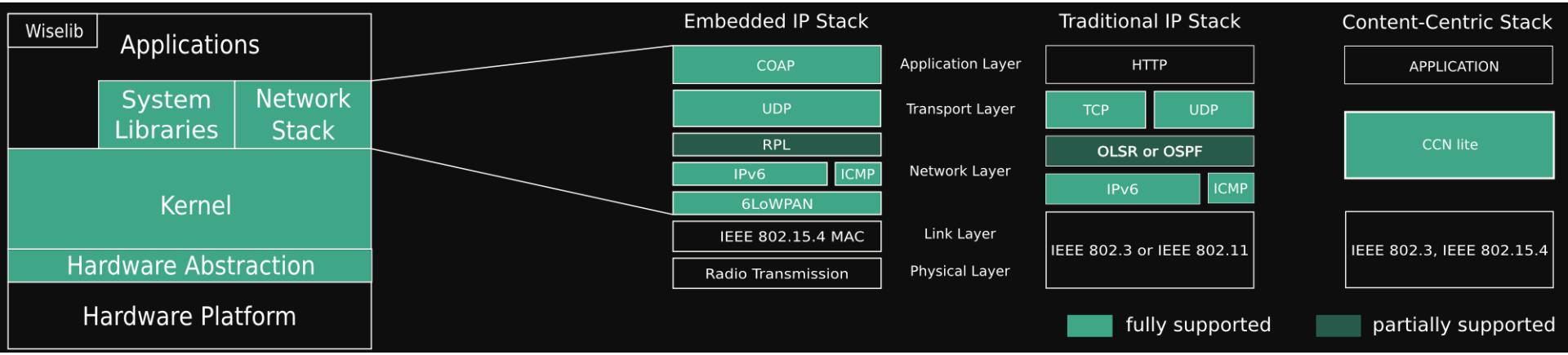
Hello World in RIOT

```
#include <stdio.h>

int main(void)
{
  printf("Hello World!\n");

  return 0;
}
```

RIOT: the friendly OS for the Internet of Things

Freie Universität Berlin

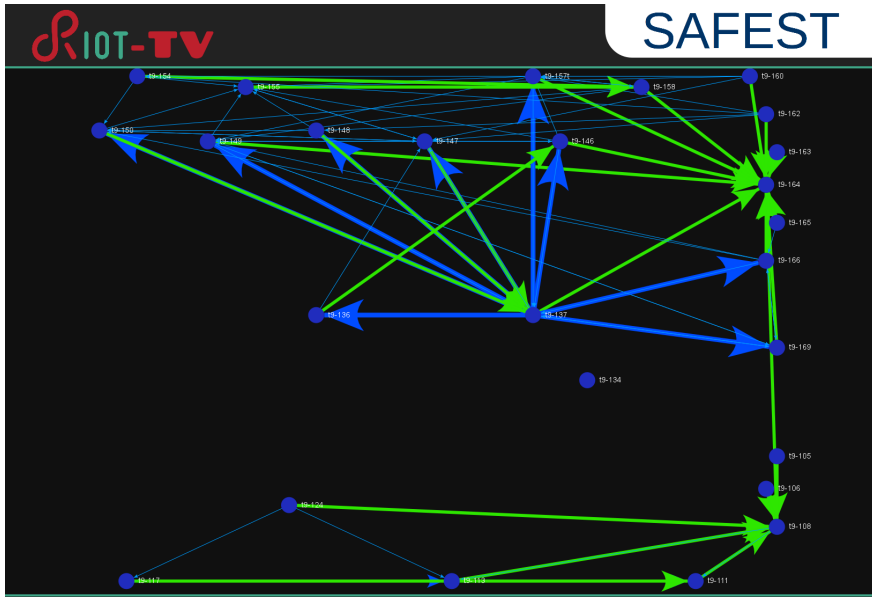
- Microkernel (for robustness)
- Modular structure to adapt to varying requirements
- Tickless scheduler (for energy efficiency)
- Deterministic kernel behaviour (for real-time capability)
- Low latency interrupt handling



- On the web: www.riot-os.org

RIOT as a tool for research

- Experiments on testbeds
  - Network protocols like 6LoWPAN, RPL, CCN etc.
  - Transport protocols
  - MAC protocols
  - Applications
  - Distributed processing



- Simulations on your Linux machine
  - native port with desvirt to emulate various networks

- Debugging on your Linux machine
  - native port with gdb, Valgrind, Wireshark etc.

```
--> ./bin/rpl_udp_router.elf tap0
RIOT native interrupts/signals initialized.
RIOT native uart0 initialized.
LED_GREEN_OFF
LED_RED_ON
RIOT native board initialized.
RIOT native hardware initialization complete.

kernel_init(): This is RIOT! (Version: 2013.08-622-ga723-tbilisi)
Scheduler...[OK]
kernel_init(): jumping into first task...
UART0 thread started.
uart0_init() [OK]
RPL router v1.1
> ps
ps
        pid | name                | state    Q | pri | stack ( used) location  | runtime | switches
          0 | idle                | pending  Q |  31 |  8192 ( 1100) 0x8068ce0 | 99.927% |        1
          1 | main                | running  Q |  15 | 16384 ( 3268) 0x8064ce0 |  0.012% |        6
          2 | uart0               | bl rx    _ |  14 |  8448 (  896) 0x806ad20 |  0.005% |       10
            | SUM                 |            |     | 33024
>
```

RIOT as a tool for app developers

- POSIX API (e.g. sockets as you're used to)
- Develop your application in C or C++
- Use advanced debugging tools
  - gdb, Valgrind, profiler etc.
- Easier debugging of distributed processes via emulated network of several instances of RIOT, right on you Linux machine
  - Native port, desvirt
- Develop once, run everywhere
  - 16bit platforms (e.g. MSP430), 32 bit platforms (e.g. ARM Cortex)

Join the RIOT

- About 40 forks on Github
  - https://github.com/RIOT-OS/RIOT
  - Start your own fork and contribute to RIOT!
- About 60 people on the developer mailing list
  - devel@riot-os.org
- Developers from all around the world
- ~ 250 followers on Twitter