The Real Ad-hoc Multi-hop Peer-to-peer (RAMP) Middleware: an Easy-to-use Support for Spontaneous Networking

Paolo Bellavista
Antonio Corradi
Carlo Giannelli

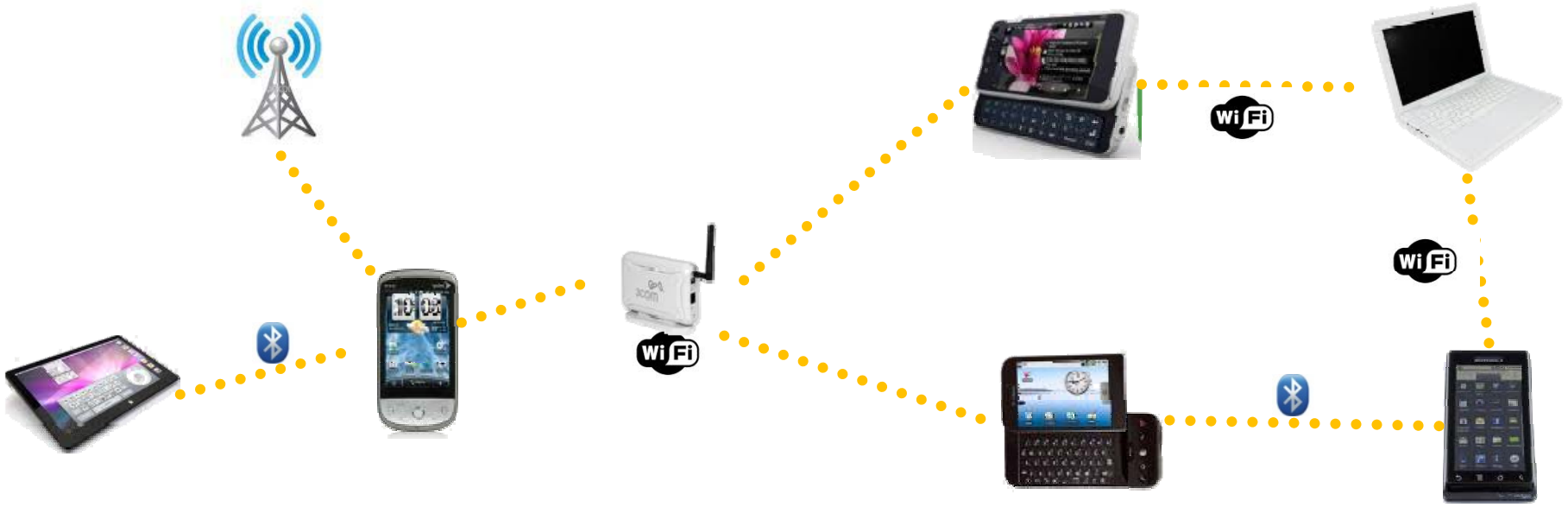
DEIS, Università degli Studi di Bologna,
Viale Risorgimento, 2 - 40136 Bologna Italy
carlo.giannelli@unibo.it

24.6.2010 - ISCC '10

Agenda

- **Spontaneous networking**
  - opportunities and issues
- **RAMP** middleware for spontaneous networking support
  - design guidelines and application-layer routing
  - main features provided to application developers
- **Efficient packet management**
  - dynamic splitting of huge packets
  - performance of File Sharing application

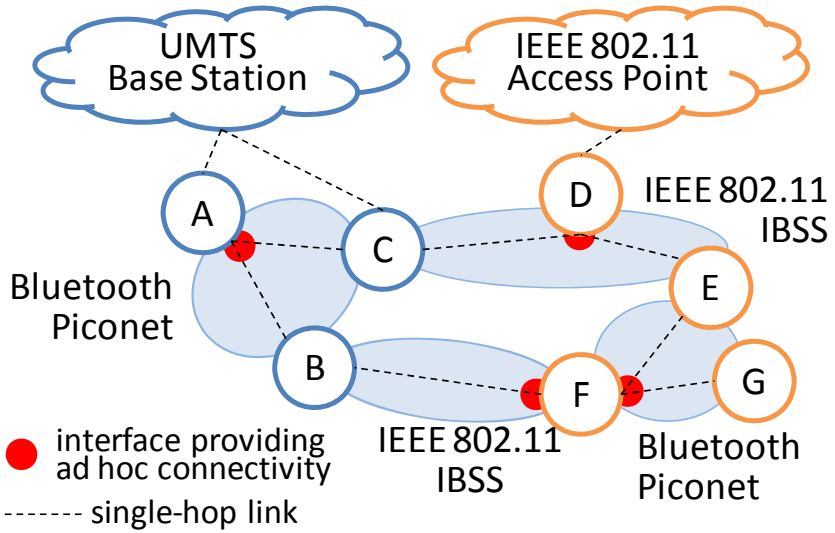
Spontaneous Network (1)



- **Impromptu** interconnection of mobile and fixed nodes
  - users willing to share content and resources
- **Maximize** interconnected nodes and **available services**
  - **heterogeneous** wireless technologies
  - both infrastructure and **ad-hoc** connectivity
  - **multiple** connectivity opportunities

Spontaneous Network (2)

- Node **cooperation** to
  - provide single-hop connectivity
  - manage multi-hop connectivity
  - support peer-to-peer services



- Peer-to-peer **File Sharing**
  - service **advertising**: NodeA provides lesson notes
  - service **discovery**: NodeF looks for nodes that share files
  - service **invocation**: NodeF browses and downloads notes stored on NodeA
- NodeA and NodeF reside in **different layer-3** networks

Issues of Spontaneous Networking

- **Heterogeneous** nodes
  - IEEE 802.11, Bluetooth, Ethernet
  - several operating systems
- **Un-coordinated** network management
  - localized provisioning of layer-2/3 connectivity
  - interconnection of heterogeneous layer-3 networks
- Erratic and **unpredictable behavior**
  - nodes abruptly create and destroy pieces of network
  - nodes dynamically join/move/leave
- Scenario **complexity** makes hard the development of novel applications

RAMP: Real Ad-hoc Multi-hop Peer-to-peer

- **Easy-to-use** middleware for **transparent** spontaneous network management in relation to
  - operating system
  - wireless technology
  - layer-3 network configuration
  - node mobility

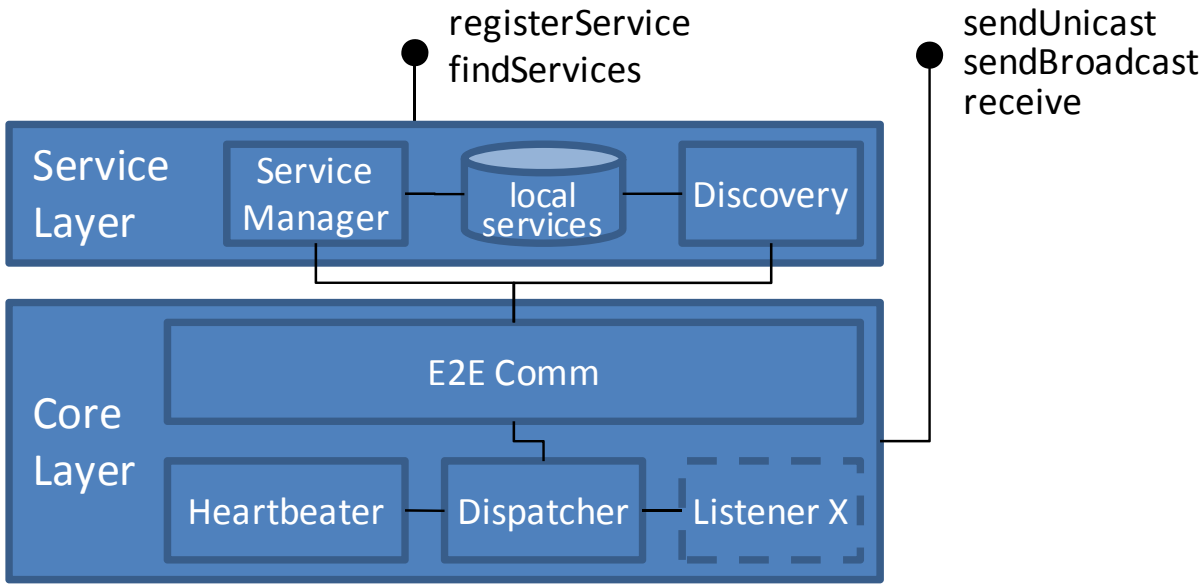
- **Unicast and broadcast** communication
  - per-packet `sendUnicast`, `sendBroadcast`, `receive`
- **Peer-to-peer service** provisioning and discovery
  - per-service `registerService`, `findService`

- RAMP Java prototype available on MS Windows XP/Vista/7, Linux and Mac OS X

RAMP Design Guidelines

- **Application-layer** management
  - layer-3 routing unsuitable for spontaneous networks
  - operating system independency + routing flexibility
- **Local** management decisions
  - nodes have partial topology awareness
  - dynamic path reconfiguration
- **Reactive** and mission-oriented approach
  - resource/path discovery only when required
  - eventually cached information invalidated very soon
- **Stateless** communication
  - per-packet information delivery and path creation
- **Cross-layer** management
  - applications may influence routing mechanism behavior

RAMP Architecture

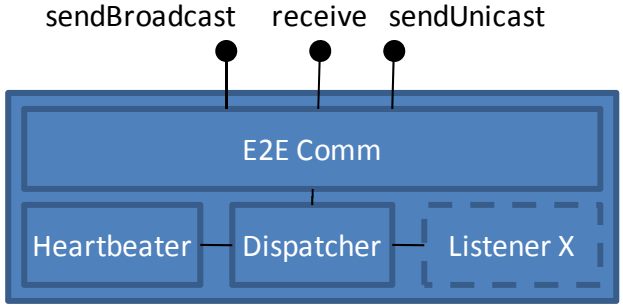


- Service Layer
  - high-level features for peer-to-peer service offering and discovery
- Core Layer
  - low-level primitives for end-to-end multi-hop communication

Service Layer

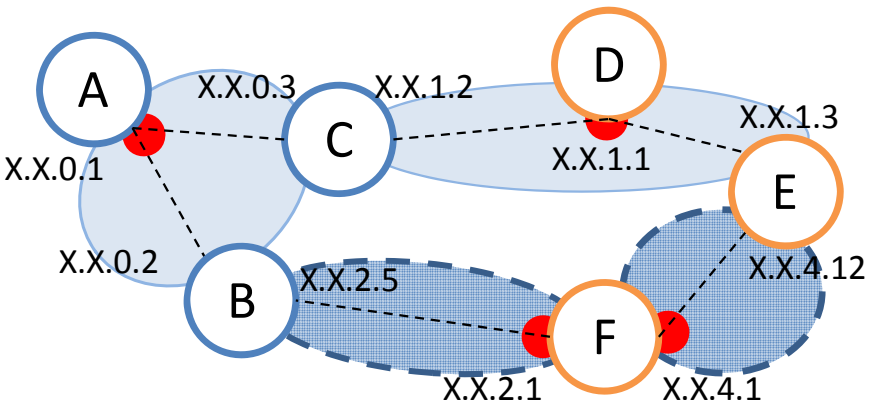
- Discovery
  - `Vector<ServiceResponse> findService(String serviceName, int TTL, int timeout, int responseAmount);`
  - **mission-oriented** TTL-bound broadcast
  - timeout + maximum service amount
- ServiceManager: registration and advertising
  - `void registerService(String serviceName, int servicePort, int prot);`
  - **registration** to the local service DB
  - advertising: it actively listens to requests and then **replies via unicast**
- Service invocation via the Core Layer

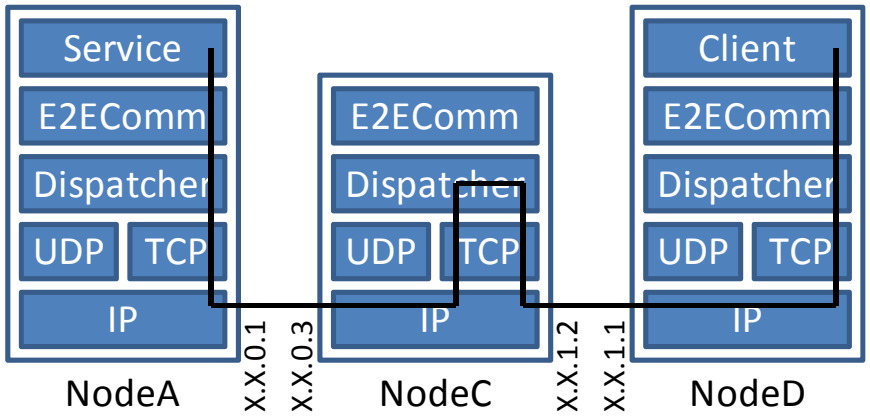
Core Layer



- **E2EComm**: per-packet communication primitives
  - en/decapsulates payload into RAMP packets
  - packet injection
    - `boolean sendUnicast(Vector<InetAddress> dest, int destPort, int prot, int bufferSize, Object payload);`
    - `void sendBroadcast(int TTL, int destPort, int prot, Object payload );` (payload at most 60KB)
  - packet reception
    - `Packet receive(int localPort, int prot, int timeout);`
- **Heartbeater** for local IP addresses gathering and single-hop neighbors discovery (IP broadcast)
- **Dispatcher** remote node interconnection
  - actual inter-node packet forwarding
  - listener-based plug-in for run-time packet management

Application-layer Routing

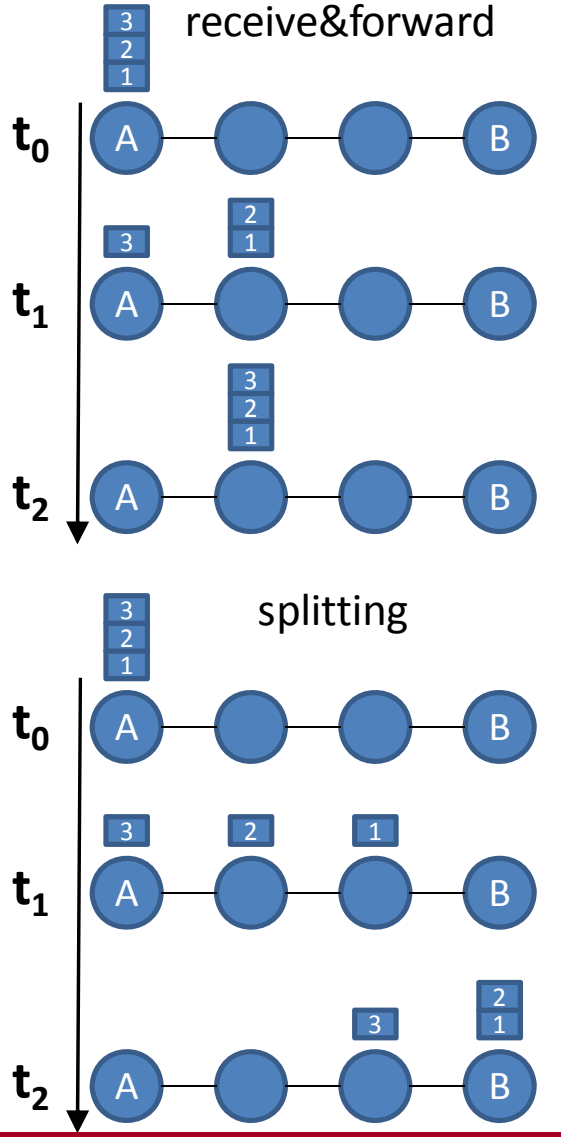




- **Un-coordinated IP address assignment** with local visibility
  - each layer-3 network has a different IP address space
- RAMP packet headers contain **multi-hop routing information**
  - **ordered set** of intermediary nodes **IP addresses**
    - from NodeA to NodeD: [X.X.0.3, X.X.1.1]
    - from NodeD to NodeA: [X.X.1.2, X.X.0.1]
- Dispatchers deliver packets **to the next node:** single-hop TCP/UDP
  - connectionless: each packet managed by a different socket
  - intermediary Dispatchers put the **traversed node sequence** in the header

Packet Splitting

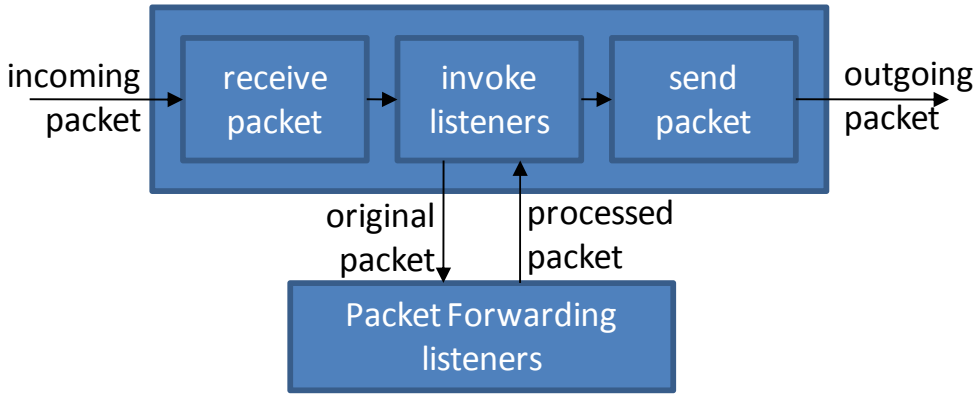
- Transmission time increases linearly with trivial *receive&forward* approach
- **Split unicast huge packets** in data chunks
  - first the **header** with routing information
  - then the **payload** split in small pieces
  - sending a payload chunk while receiving other chunks
- Chunk size tradeoff: `bufferSize`
  - **low value**: prompt transmission and **small local buffers** on intermediary nodes but frequent read/write
  - **high value**: minor communication overhead but delayed transmission and **greater memory usage**



BufferSize Management

- Dynamic and context-aware packet splitting management based on **cross-layer** approach

- **Application-driven** `bufferSize` selection
  - per-packet size selection based on path length and packet size
- Listener-based **BufferManager** plug-in to set `bufferSize` on a per-node granularity
  - users of intermediary nodes can select the proper buffer size of traversing packets to **limit/tune local resource consumption**



File Sharing Application

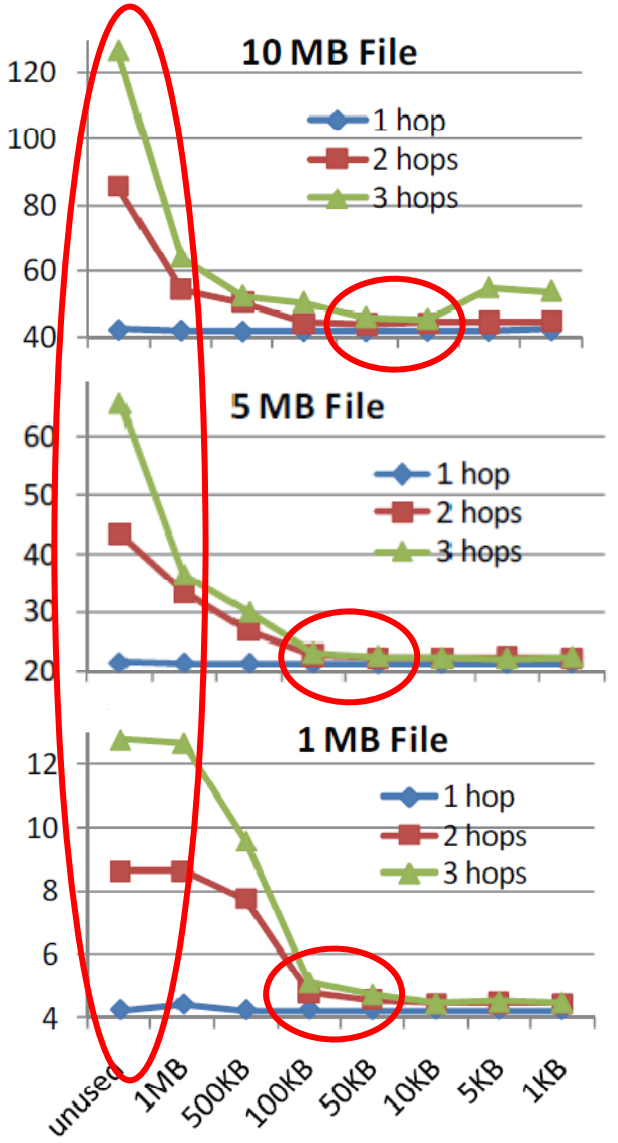
- Testing **RAMP routing mechanism** and huge packets transmission performance in different deployment scenarios
  - **file size** from 100KB to 10MB
  - **buffer size** from 1KB to 1 MB
  - **path length** from 1 to 3
- Comparison with analytical and **layer-3 routing**
  - 2Mbit/s single-hop bandwidth limit

- File Sharing Service
  - `registerService`, `receive`, `sendUnicast`
- File Sharing Client
  - `findService`, `sendUnicast`, `receive`
- Different protocols
  - UDP for service discovery (small data amount)
  - TCP for file content transfer (huge data amount)

File Sharing Performance

File Size	Size/Bandwidth (s)	`iperf`-based (s)
10 MB	40	42.2
5 MB	20	21.4
1 MB	4	4.2
500 KB	1.95	2.0
100 KB	0.39	0.4

- No split: `bufferSize` greater than file size
  - transmission time increases linearly
- **Split**: `bufferSize` lower than file size
  - transmission time **greatly lowers**
  - RAMP introduces **little overhead**
- **Best buffer size**
  - minimum transmission time while limiting read/write
  - depends on path length and packet size
  - sub-optimal **default** value: 50KB
  - `int bestBufferSize(int packetSize, int pathLength);`



Conclusions

- RAMP supports **multi-hop** communication in heterogeneous spontaneous networks
  - **application-layer routing** transparent in relation to OS, wireless technology, network topology
  - performance comparable with traditional layer-3 routing
- RAMP is **service-oriented**
  - easy-to-use API for service development by non-expert programmers
- **RAMP prototype available** to leverage application development and spread spontaneous networking

- Ongoing work
  - live multimedia stream via DVB-T **re-casting**
  - porting to **mobile platforms**

Any Question?

- Prototype code and implementation insights
  - http://lia.deis.unibo.it/research/**RAMP**/
  - http://lia.deis.unibo.it/Staff/**CarloGiannelli**/