Context-Aware Network Stack Optimization

Olivier Mehani[1,2] Roksana Boreli[2] Thierry Ernst[1]

[1]Institut National de Rercherche en Informatique
et Automatique — Équipe-projet Imara

INRIA

name.surname@inria.fr

[2]National ICT Australia —
Networked Systems Theme

NICTA

name.surname@nicta.com.au

October, 20 2009

Outline

Outline

Context and Objectives

Heterogeneous environment but some unifying blocks

- Vehicular networks
  - Rapidly changing conditions
  - Mobility geographical and time patterns
- Ad-hoc environment
  - Few or no structural organisation
  - Possibility of insider attacks
- Network mobility and Multihoming
  - Several interfaces, networks and routes available at once to choose from
  - ISO CALM mandates IPv6 use

Context and Objectives

No repository of complete information without cross-layering

- Network information and control scattered between layers
  - All parameters contribute to the overall performance
  - No full read/write access to all of them
- Consideration of cross-layer approaches
  - Share information between layer implementations
  - Each layer makes its own optimizing decision
- Usual issues of such designs
  - Linked layers too specialized for the considered environment
  - Unintended interactions vith each other or other parts of the system
- Out-of-stack cross-layers information busses
  - MobileMAN, ULLA, CALM Manager,. . .
  - Doesn't quite solve the bad interaction problem

Context and Objectives

Use the network resources as soon as possible and as much as supported

- Fully informed decision out of the stack
  - Based on all available information
  - Updates parameters of several layers at once
- Advantages:
  - Avoid over-specialization of layers
  - Work around bad interactions
  - Can use non network-related information
    - History, context, localization, . . .

$\Rightarrow$ Replace a knowledgeable user tweaking their parameters

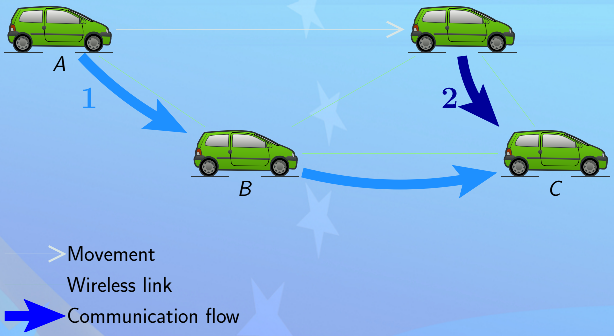
- knows the full current network performance
- has an idea of the current best achievable
- knows reasonably well what is soon to change
- can change any parameter

Outline

Motivating Examples

Transport adaptation on link characteristics change

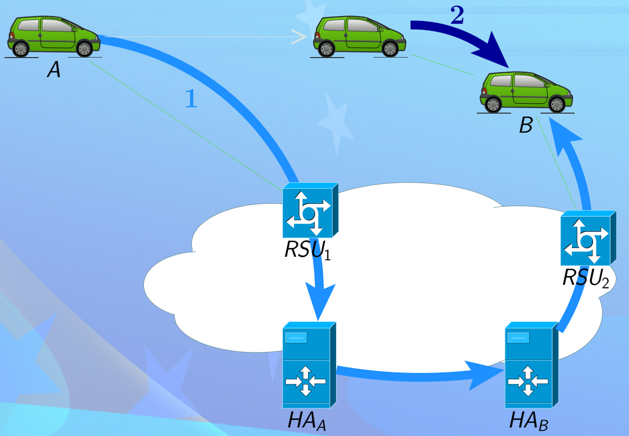
- Several possible changes in routes configuration
  - increase or reduction in the number of hops in a mesh network



Motivating Examples

Transport adaptation on link characteristics change

- Several possible changes in routes configuration
  - increase or reduction in the number of hops in a mesh network
  - switch from NEMO routes to MANET routes (or opposite)



Motivating Examples

Transport adaptation on link characteristics change

- Several possible changes in routes configuration
  - increase or reduction in the number of hops in a mesh network
  - switch from NEMO routes to MANET routes (or opposite)
- Ever-varying link parameters
  - multipath fading
  - interferences

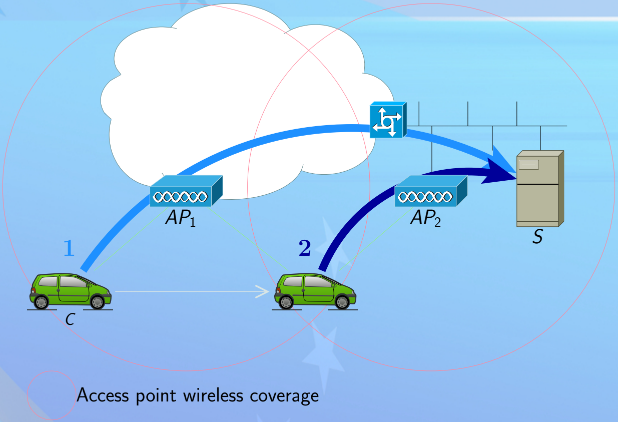
Motivating Examples

Transport adaptation on link characteristics change

- Several possible changes in routes configuration
  - increase or reduction in the number of hops in a mesh network
  - switch from NEMO routes to MANET routes (or opposite)
- Ever-varying link parameters
- Transport protocol not directly aware of such changes
  - Slow feedback-based adaptation to new link and path characteristics
  - Need information about the link and path performances
  - Similar issues for a new connection: expected path throughput as observed on other sockets along the same path
- Application parameters adjustment
  - quality
  - sampling rate

Motivating Examples

Transport adaptation on link characteristics change

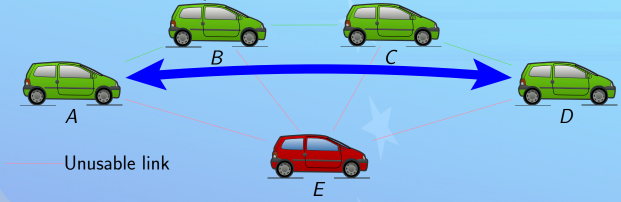


- Choose the most appropriate uplink or route
  - Based on current measurements
  - Based on previous observations
- Inform transport/application as previously mentionned

Motivating Examples

Rogue VANET Node

- E advertises its presence but doesn't forward traffic properly

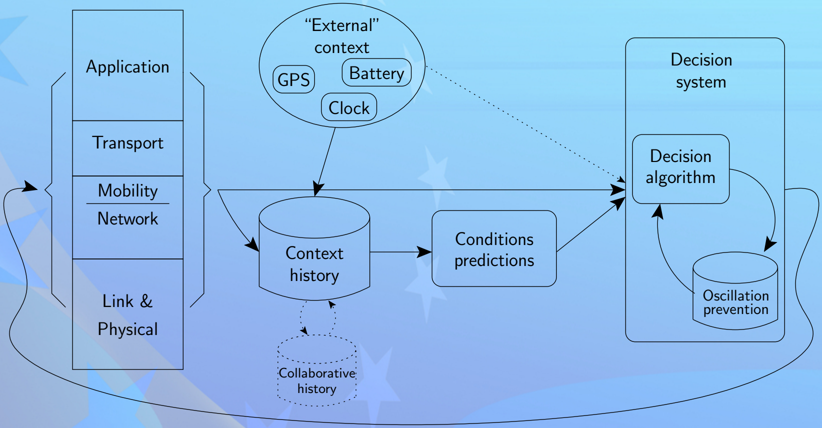


- Communication between A and D not possible along the shortest path
- Switching to another route desirable
- Quick upper layers adjustement needed afterwards

Outline

Target Architecture

Towards a global optimization system



Target Architecture

Combining observation, prediction and decision

- Abstracted metrics from the stack
  - datarate, delay, ETX, RTT, loss, ...
- Context observations and history
  - human timescale: days or weeks
  - predict forthcoming conditions based on previous observations in similar contexts (day of the week, localization, movement pattern)
- Decision engine

  internal history to avoid short-period oscillations

  hints to the stack to finally optimize the network usage

INRIA NICTA

Target Architecture

Optimization decision based on global knowledge

- Globally aggregated information from the stack layers
  - Application: $I_{\mathrm{app}}(s, t) = \{c = \mathrm{codec}(t), \ldots\}$
  - Transport: $I_{\mathrm{trp}}(s, t) = \{thr = \mathrm{throughput}(s, t), \mathrm{rtt}(s, t), \ldots\}$
  - Network: $I_{\mathrm{rt}}(t) = \{nh_B = \mathrm{nextHop}(B, t), \ldots\}$
- Relative impact of the combination on current performance
  - "For socket s, a throughput of thr, needed by codec c, is achievable towards node B along a path starting with nh_B."
  - Identification of other communications with common characteristics (*e.g.* same destination) but different performance (*e.g.* higher throughput)
- Hint the network stack layer to adapt to the possible conditions *e.g.*,
  - change interface modulation or power
  - switch route for an address (or range), perform a handover
  - update transport state parameters
  - modify application parameters (*e.g.* encoding or rate)

INRIA NICTA

Constraint-based Decision Algorithm

Motivation and basic idea

- Large configuration space $\rightarrow$ Combinatorial search techniques
- Modelled as a Constraint Programming problem
- The constraints solver unifies parameters to derives hints for the stack
  - Application quality: $\mathrm{Qual}(Quality, Throughput, Jitter, RTT, PER)$
  - Application socket: $\mathrm{SocketDestination}(Socket, Destination)$
  - Transport and routing conditions: $\mathrm{Routing}(Destination, Route, Interface, Throughput, Jitter, RTT, PER)$
- Variables to unify are
  - observed conditions on the links/networks/paths (offer)
  - possible configurations of the layers (demand)

Constraint-based Decision Algorithm

Example relations for a simplified model

Observed network performances

Destination	Route	Interface	Throughput	Jitter	RTT	PER
Addr1	NH2	eth0	2 Mbps	1×10^{-4} s	10×10^{-3} s	0%
Addr1	NH1	wlan0	900 kbps	1×10^{-3} s	100×10^{-3} s	10%
Addr2	NH1	wlan0	450 kbps	1×10^{-3} s	250×10^{-3} s	30%
			...			

Socket between applications and destinations

Socket	Application	Destination
1	App1	Addr1
	...	

Interface costs (switching + usage)

Interface	Cost
eth0	10
wlan0	100
ppp0	250
...	

Application App1 parameters and requirements

Quality	Throughput	Jitter	RTT	PER
1	$\geq$ 1.5 Mbps	$\leq 10^{-3}$	$\leq 10 \times 10^{-4}$ s	$\leq 10 \times 10^{-3}$
2	$\geq$ 1 Mbps	idem	idem	idem
3	$\geq$ 500 kbps	$\leq 10^{-2}$	$\leq 10 \times 10^{-3}$ s	idem

Constraint-based Decision Algorithm

Optimizing a cost function

- Parameter valuation trying to minimize a cost function *e.g.*

$$\min \left(\alpha \cdot rtt - \beta \cdot thr + \gamma \cdot C_{if}\right)$$

  - minimize *rtt*
  - maximize *throughput*
  - minimize interface cost

Current Status

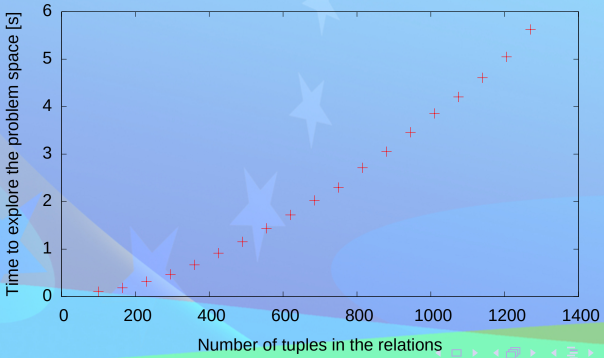
Core compononents under development

- Python/Netlink implementation under Linux
  - NETLINK_ROUTE, pushed by the kernel
    - link configuration parameters (RTMGRP_LINK)
    - neighbor information (RTMGRP_NEIGH)
    - interface addresses (RTMGRP_IPV6_IFADDR)
    - route information (RTMGRP_IPV6_ROUTE)
  - NETLINK_INET_DIAG, upon request
    - socket information
    - transport parameters
  - not widely available (yet)
    - passing hints back to the stack to change parameters
- MiniZinc Constraint Solver
  - current model similar to previously outlined
  - extended by history and cost relations
  - subject to change

Current Status

Timing evaluation of the solver

- CSP model randomly generated (max 5 interfaces, 190 destinations, 95 sockets)
- Coherent data (respecting ranges and correlations of parameters)
- 100 runs on an Intel Core2 Duo 2 GHz, 1 GB RAM
- All optimizations disabled → raw estimate of a higher bound



INRIA NICTA

Outline

1. Context and Objectives
2. Motivating Examples
3. Target Architecture
   - Constraint-based Decision Algorithm
   - Current Status
4. Conclusion and Short-term Plans

Conclusion and Short-term Plans

- Global cross-layer optimization framework
  - External decision process
- Early implementation blocks
- Ongoing work
  - Finish the prototype implementation
    - Generalize the use of Netlink to adjust parameters
    - Unification of data
- Next steps
  - Acquire network and contextual data samples
  - Large scale simulations
  - Consider other decision systems

Thanks

olivier.mehani@inria.fr
olivier.mehani@nicta.com.au

Backup

Target Architecture

Metrics, statuses and parameters

- Physical/Link

  Raw characteristics rate, status, bytes, lost segments (RR), lost fragments (FER), link, noise (RSSI, SNIR) ; Notifications: link_up, link_down, link_quality_changed [ULL], transmission power
  - for each MAC neighbour (*e.g.* AP/Cell Tower in infrastructure mode; all neighbours for ad-hoc modes)

  Events Link Up/Down/Parameters Change/Going Down; Load Balancing; Operator Preferences [802.21]

  Contextual information VLAN [802.1q], SSID [802.11], CellID,...
- Network/Mobility
- Transport
- Application
- See also [RFC4907]

Target Architecture

Metrics, statuses and parameters

- Physical/Link
- Network/Mobility
  - abstract route metrics
  - MTUs
  - possible next hops to an address/range,
  - route addition/removal/change
  - ARP/NDP: mapping from next hop to MAC address
  - single-interface handoff decisions
- Transport
- Application
- See also [RFC4907]

Architecture description

Target Architecture

Metrics, statuses and parameters

- Physical/Link
- Network/Mobility
- Transport
  - throughput (for TCP: cwnd, sstthr, wnd)
  - RTT
  - loss rate
  - congestion information about paths
  - path MSS
  - peers' network address(es)
- Application
- See also [RFC4907]

Architecture description

Target Architecture

Metrics, statuses and parameters

- Physical/Link
- Network/Mobility
- Transport
- Application
  - requirements in end-to-end bandwidth
  - end-to-end delay limits
  - need for packet reliability (implicitely stated when chooosing the transport)
  - configurable modes of operation (*e.g.* codecs), and all of the above for each
  - "satisfaction" (completely ad-hoc to the application *e.g.*, peer feedback on validity of data)
- See also [RFC4907]

◂ Architecture description