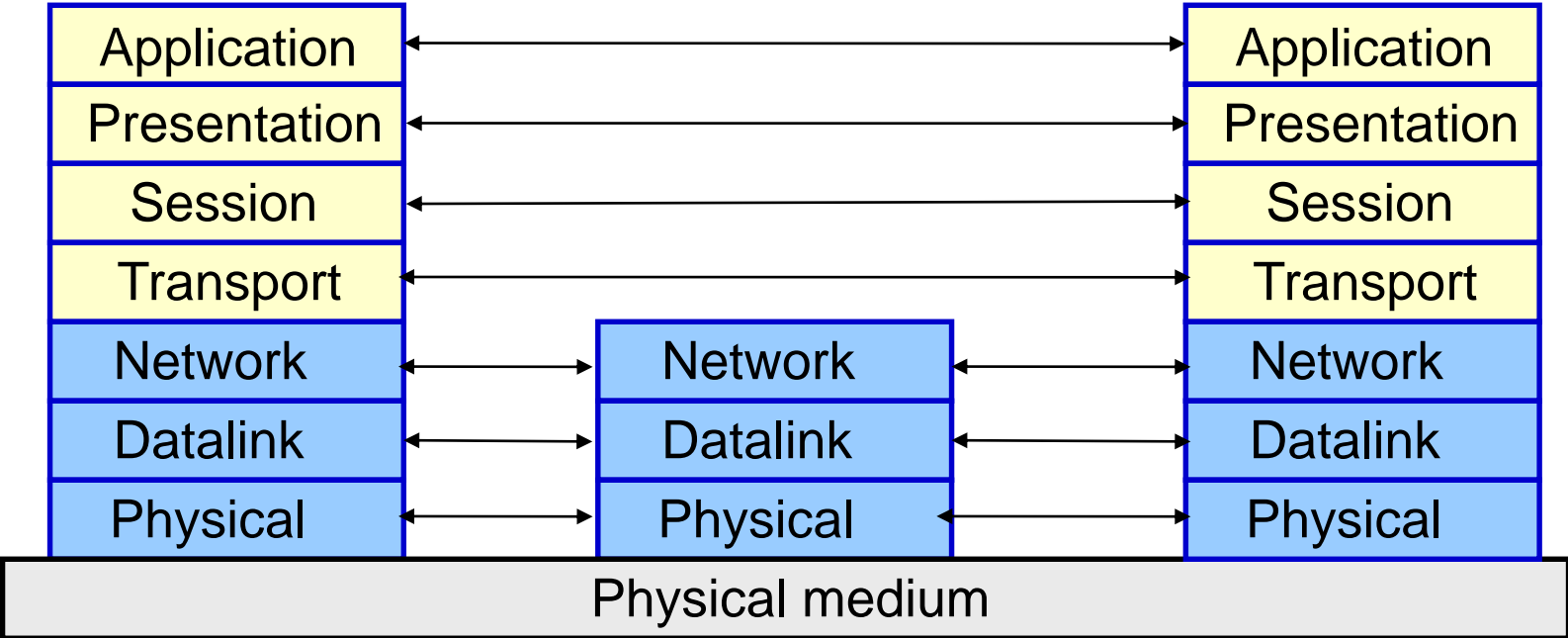


Cross-Layer Design



A Brief Review of Protocol Layers

Premise: Break network tasks into logically distinct entities, each built on top of the service provided by the lower layer entities.



Example: OSI Reference Model



Recap: Layered Protocol Design

- Traditionally, we have a protocol stack
 - OSI Reference model: 7 layers
 - Internet or TCP/IP protocol: 5 layers
- Each layer has specific tasks, communicates via interfaces with adjacent layers and through a protocol to the same layer at other nodes
- On sender: packets are passed down the protocol stack, protocols at each layer may modify the data as it passes along, and typically add a packet header with protocol-specific information
 - TCP: 20 bytes/packet (plus options), UDP: 8 bytes/packet
 - IPv4: 20 bytes/packet (plus options), IPv6: 40 bytes/packet (plus potentially additional headers chained together)
- On receiver: packets are passed up the protocol stack (using protocol fields and port numbers for de-multiplexing), with protocols at each layer stripping off the extra information

Layered Protocol Stack: Advantages

- Each layer only implements a (relatively) small and well-defined set of required network functions
 - Reliability: TCP protocol
 - Access to shared media: MAC (part of Data Link)
 - Mobility support: IP and Mobile IP (or DHCP) at the network layer
- Interface between layers well defined, so it is possible to replace protocols at one layer without changing the whole network protocol stack
 - You can replace the routing protocol (network layer) without impacting how well TCP implements reliability and congestion control or the MAC protocol deals with accessing shared media
 - You can define your own application-layer protocol while enjoying the services of a transport layer
 - You can introduce a new transmission media/networking technology and still run HTTP over it

Layered Protocol Stack: Disadvantages

- Duplication of effort: encrypt data multiple times, send ACKs at multiple layers => wastes resources (CPU, network)
- Each layer adds additional information (header, sometimes trailer)
 - Increases networking overhead
 - ZigBee: packet size is limited to 127 bytes. Voice over IP often uses RTP over UDP over IP. So using IPv6, we have 12 bytes for RTP plus 8 bytes for UDP plus 40 bytes for IP, adding up to 60 bytes or half the MAXIMUM packet size.
 - Solutions:
 -  Have fewer layers → fewer headers (presumably)
 - Motes/TinyOS provides AM interface, which combines UDP and IP
 -  Header compression
 - 6LowPAN: compress IPv6 and TCP/UDP headers down to a few bytes
 - Which option would be better?

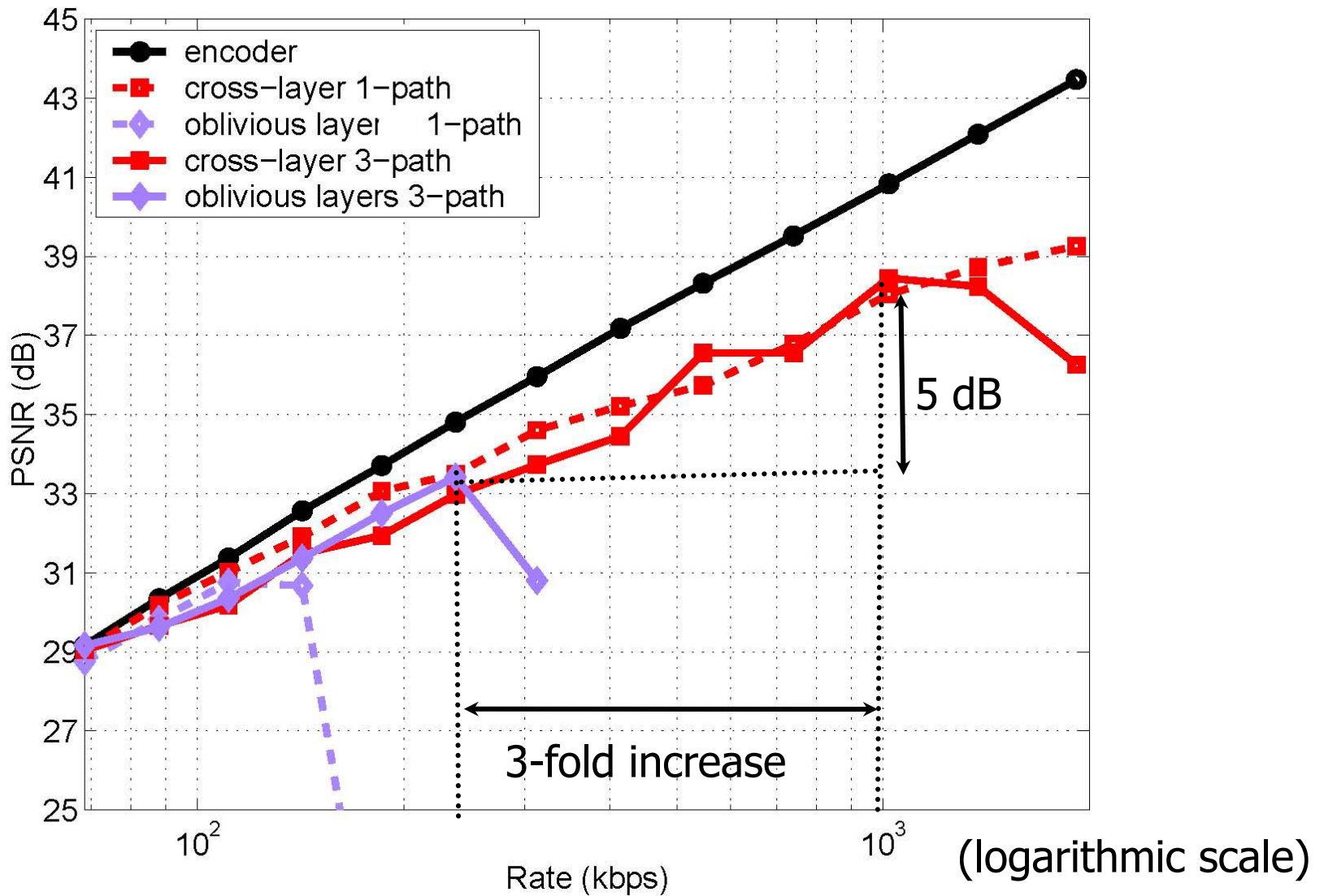
Layered Protocol Stack: Disadvantages

- Layers care only about the functionality assigned to them, and have only limited amount of information about the whole network (mostly information conveyed by neighboring layer):
 - Solution at that layer may not be “globally optimal” (End-to-End Argument again)
 - Examples
 - 📁 IEEE 802.11 implements reliability via re-transmissions, even for applications that could handle occasional packet loss (streaming video, voice, etc.)
 - 📁 Mobile IP handles mobility – what about application that want to use the “closest” printer? Application-level protocols do not know that the device moved
 - 📁 MAC protocols typically promote fair sharing of the media among all devices. What if the traffic is heavily asymmetric (WiFi Access Point carries traffic for all STAs that connect to it)
 - 📁 PHY layer selects coding and modulation rates based on channel conditions. So it knows when the channel is poor, application does not. Yet modern video codecs can adapt video codec in interesting ways to achieve high end-user QoE

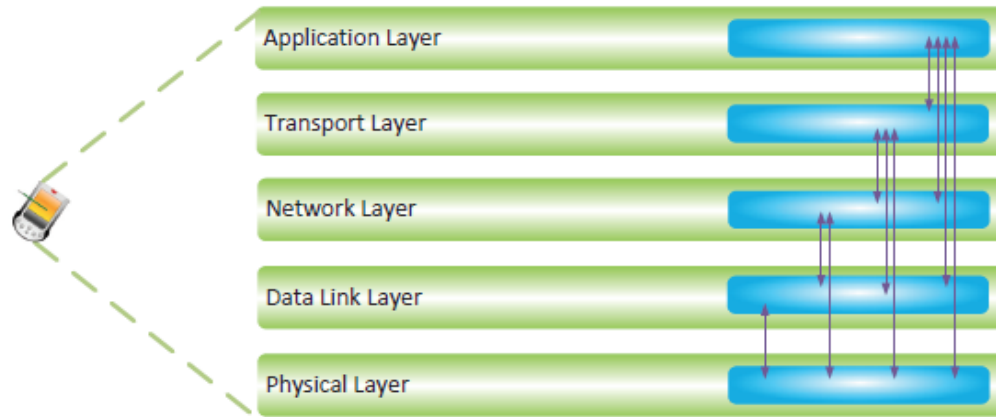
Cross-Layer Designs

- Address the last disadvantage
 - Allow layers to exchange more information, and not only between adjacent layers
 - Idea: each layer can now contribute more appropriately to overall end-user experience by JOINTLY optimizing their operation
 - Questions
 - 📖 How to organize information exchange?
 - 📖 Who ultimately decides relevant protocol parameters (backoff window, packet size, coding rate, etc.): each protocol, or centrally
 - **A Survey of Cross-Layer Designs in Wireless Networks** discusses proposed solutions based on how they answer these two questions.

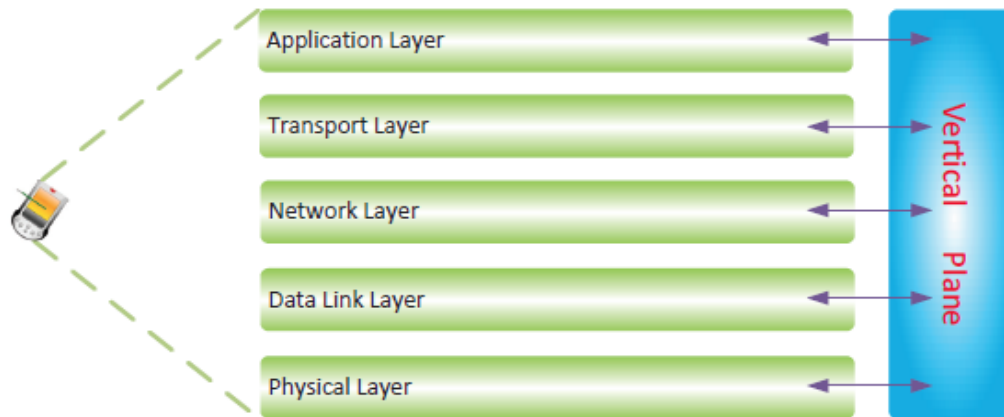
One Example: Adaptive Video Player



Cross-Layer Design Taxonomy 1: How to Exchange Information



(a) Non-manager method

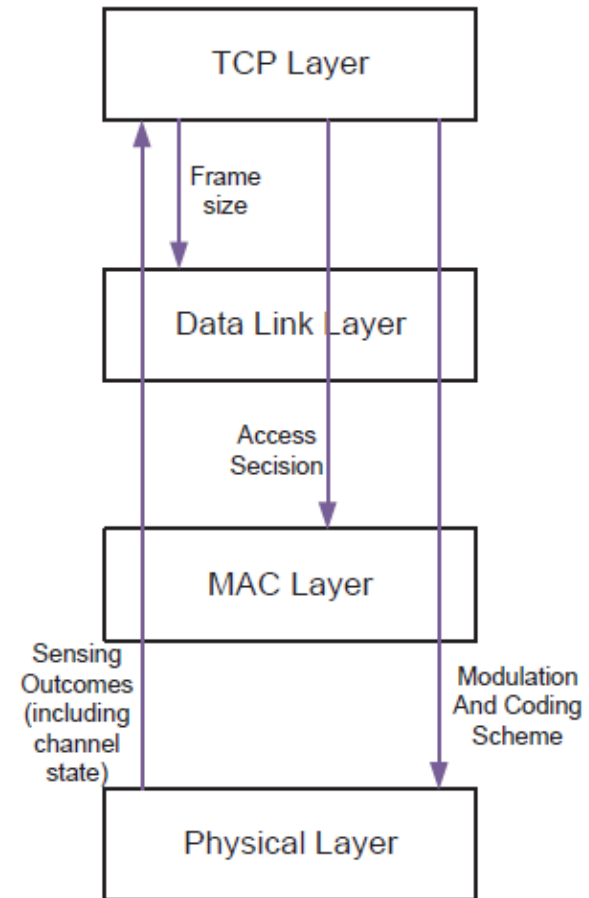


(b) Manager method

Example for Cross-Layer Design without Manager

Goal: improve TCP Throughput

- TCP layer learns about wireless channel conditions
- TCP protocol sets:
 - Frame Size
 - Access Decisions for MAC
 - Coding and Modulation Decisions for PHY

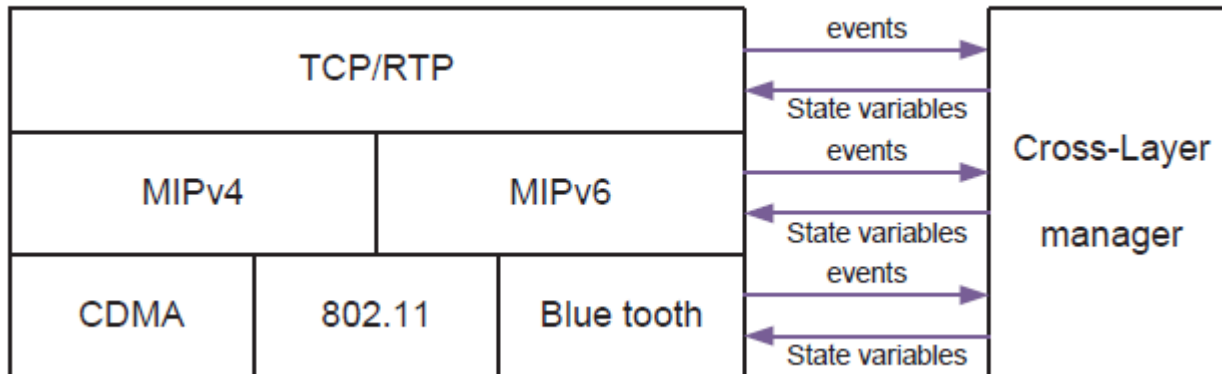


Example for Cross-Layer Design with Manager

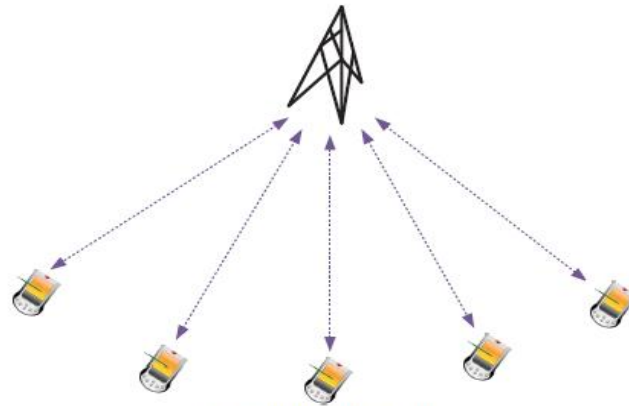
Goal: improve performance of wireless links and mobile terminals

Each layer shares its events with the cross-layer manager and has access to the state variables of protocols at other layers

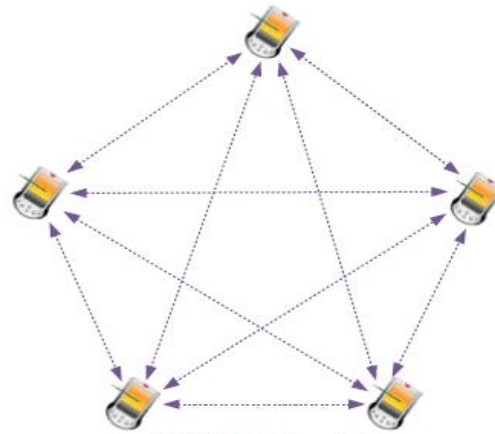
Advantage: interfaces are more clearly defined, reduced number of interfaces (each layer has ONE additional interface with the manager, not N-1 interfaces with the other layers)



Cross-Layer Design Taxonomy 2: Where to Make Optimization Decisions



(a) Centralized method



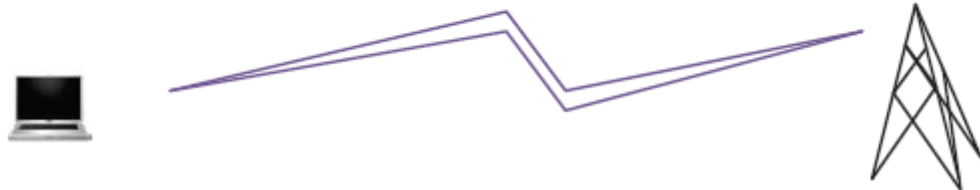
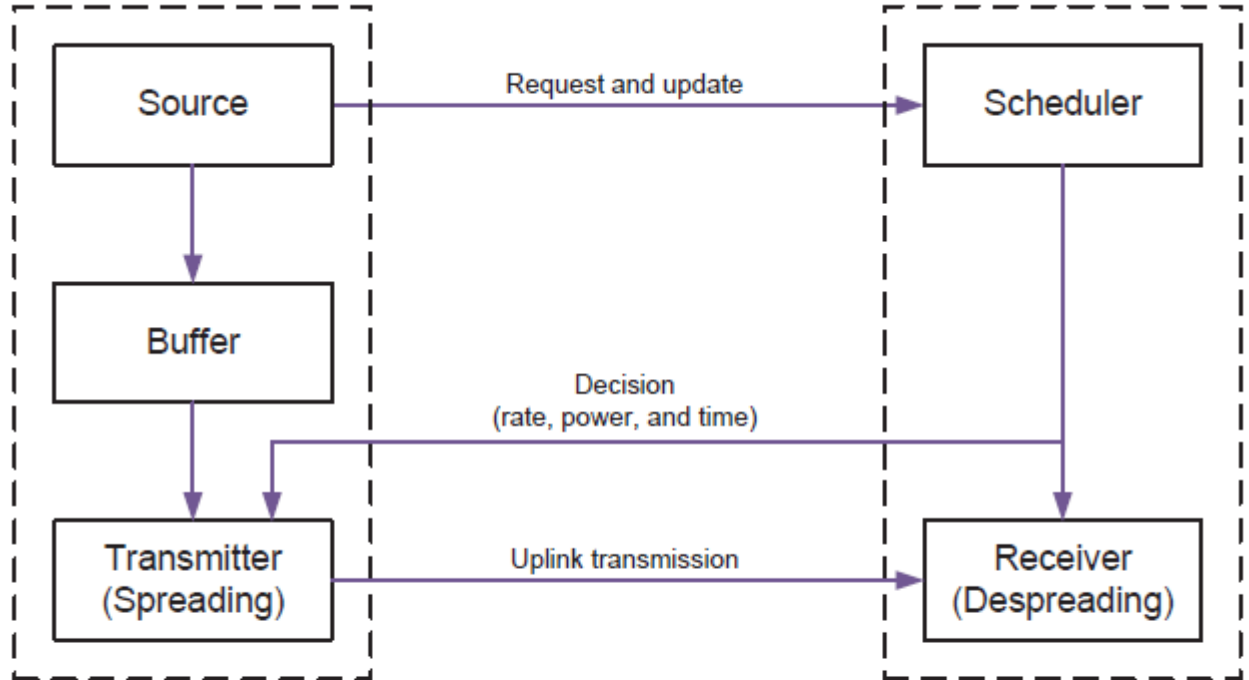
(b) Distributed method

Example for Cross-Layer Design with Centralized Decision Maker

3G Uplink Scheduling

Determine link layer resource allocation of ALL mobile devices based on info from applications and PHY

Optimizes for ALL mobiles and communicates that to them

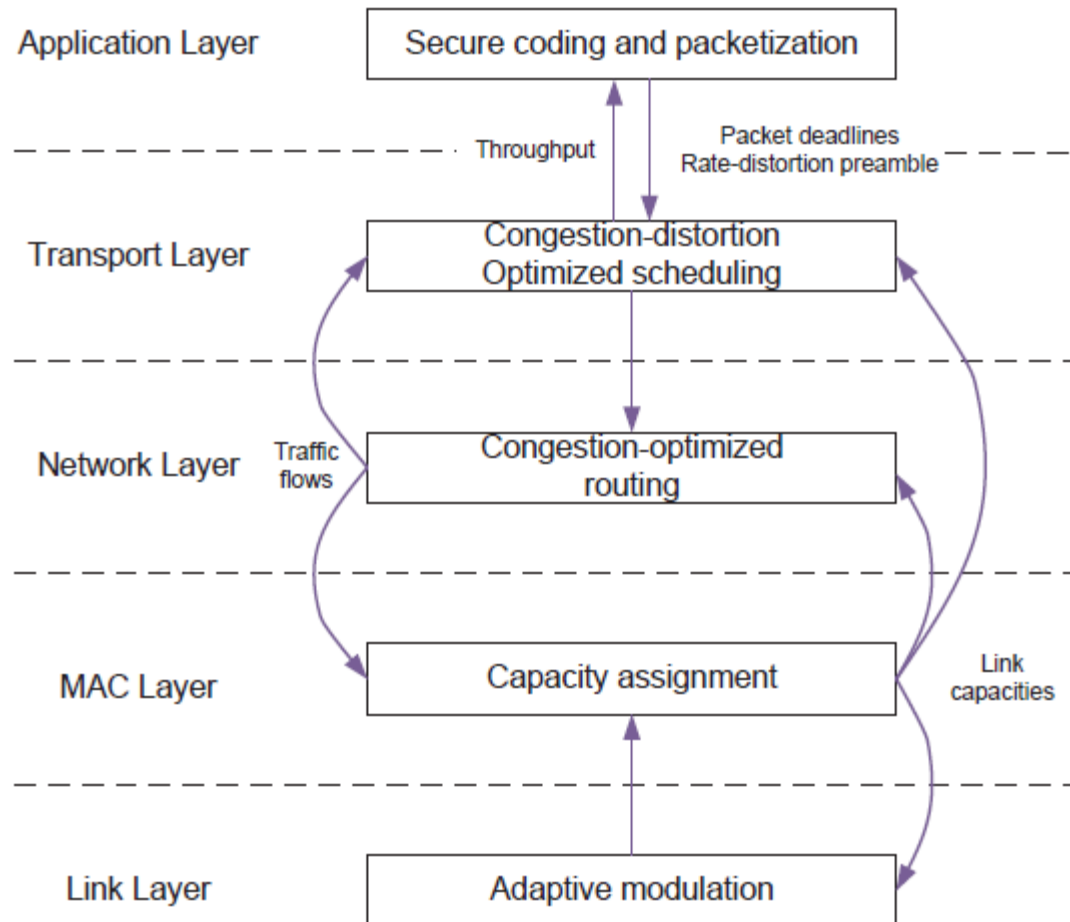


Example for Cross-Layer Design with Distributed Decision Making

Low latency Video Streaming

Each node: layers cooperate to achieve optimal performance

Protocols also spread this information across the network, so optimizations can take network-wide info into account (more later)

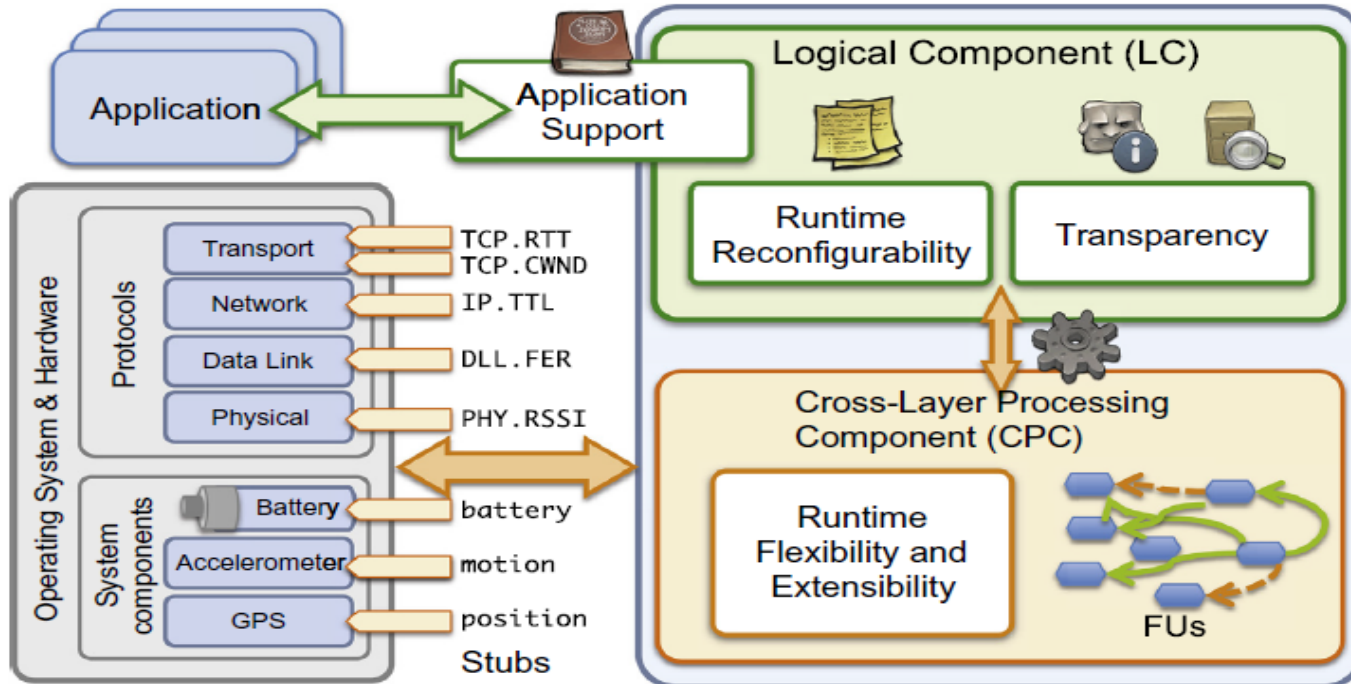


Harnessing cross-layer-design

Second paper not about a specific cross-layer design approach

Rather: talks about a high-level experimental architecture to implement different cross-layer solutions

Paper has some examples of how to use this to implement different cross-layer solutions



Own Work: Cross-Layer Design for MANETs

Goal: Propose Realistic Cross-Layer Design that improves MANET Performance

- Work based on NUM (Network Utility Maximization) framework:
 - Flows are assigned a utility value (typically a convex function of the session rate)
 - Goal is to maximize a function of the flow utilities
 - Different ways to add them results in different fairness criteria
 - Max SUM of utilities, which are $\log(\text{session rate})$: proportional fairness
 - Max MIN UTILITY: max-,min fairness
- Realistic:
 - Keep layered architecture as much as possible
 - Solve optimization problem in a distributed fashion
 - Show that Cross-Layer designs beats Layered approach even:
 - When correctly accounting for additional overheads
 - In the presence of lost messages (which reduce convergence to optimal solution)

Related Work

- Mathematical models exist that jointly solve
 - the MAC layer access probability (the contention problem)
 - the transport layer congestion problem (how data each session should generate to max overall utility) without congesting the network

- Missing:
 - no accounting for mobility,
 - no accounting for lost messages (idealized evaluations),
 - no accounting for sessions coming and going (impact of speed of convergence),
 - no real distributed implementations and accounting of the additional overhead

- Also missing (future work for Ammar): including the network layer
 - find routes other than shortest hop that would max overall utility

Network Model

Starting from the NUM

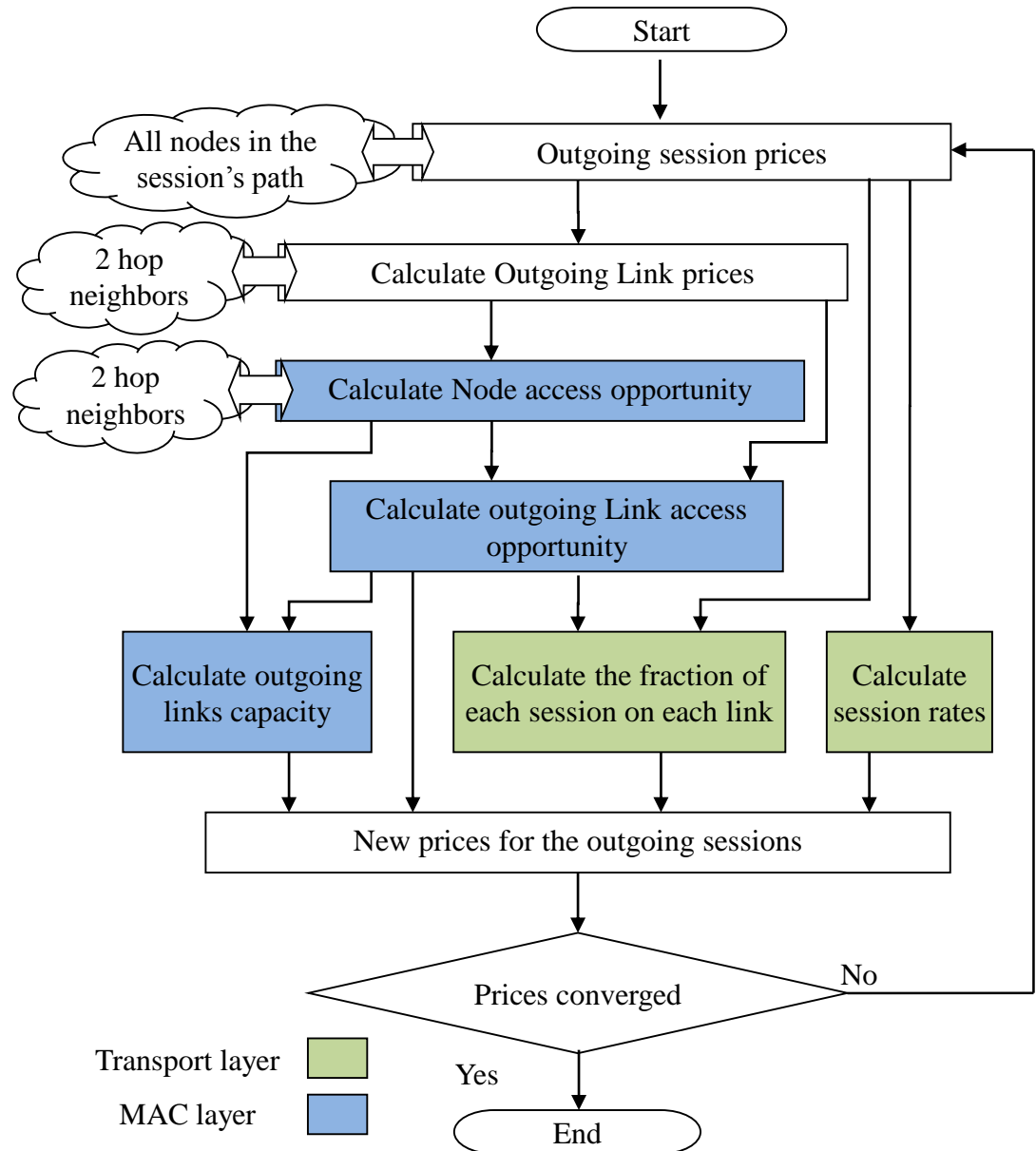
Objective function $\max \sum_s U_s(y_s)$

$U_s(y_s)$ is the utility function and y_s is the rate of the session s

Constraint $\sum_{s \in S(l)} y_s \leq C_l$

C_l is the capacity of the link l and $S(l)$ is the set of sessions that use the link l

Network Model



Network Model (CLD)

For the ALOHA MAC protocol

The transmission opportunity is a transmission probability

The link capacity

$$C_l = C_{max} p_l \prod_{n \in N_{to(l)}} (1 - P_n)$$

C_{max} is the maximum physical capacity, p_l is the link access probability, P_n is the node access probability, and $N_{to(l)}$ is the nodes whose transmission will impact the receiver of the link l .

Network Model (CLD)

For the CSMA-CA MAC protocol

The transmission opportunity is calculated from the throughput

The link capacity

$$C_l = C_{max} * p_l * THR_N$$

THR_N is the throughput of the node N calculated using Bianci's model.

This model will provide us with the saturation throughput as a function of different parameters as follows,

$$THR = \frac{P_s P_{tr} E(P)}{(1 - P_{tr})\sigma + P_s P_{tr} T_s + P_{tr} (1 - P_s) T_c}$$

P_{tr} = Probability that there is at least one transmission in the considered time slot.

P_s = Probability of successful transmission.

$E(P)$ = Average message payload.

T_s = The average time the channel is sensed busy due to a successful transmission.

T_c = The average time the channel is sensed busy during a collision.

σ = Slot time size.

Network Model (OLD)

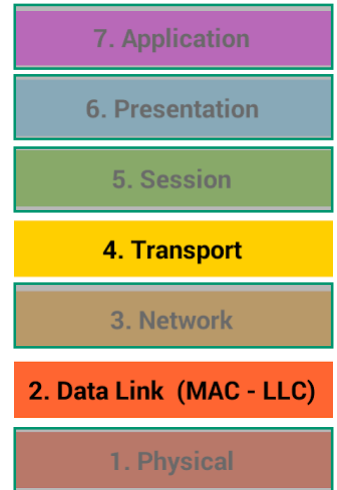
Oblivious layer design (OLD)

No coordination

No prices exchanges

Layer decision based on local information at each layer

The transmission opportunity will be equally divided over the number of active nodes in the contention area



Simulations

As our interest is concentrated on comparing CLD and OLD, the absolute performance results are of less importance at this point.

Assumptions

- The model can calculate optimistic time-independent utilities for the CLD and OLD.
- The topology and the feedback information are known instantaneously and also that the algorithm converges to the optimum immediately.
- The random waypoint mobility model, the rerouting events are done instantly based on a shortest hop routing protocol, the maximum physical capacity is 1, and the utility function is the log function that achieves proportional fairness

Simulations

The CSMA-CA model

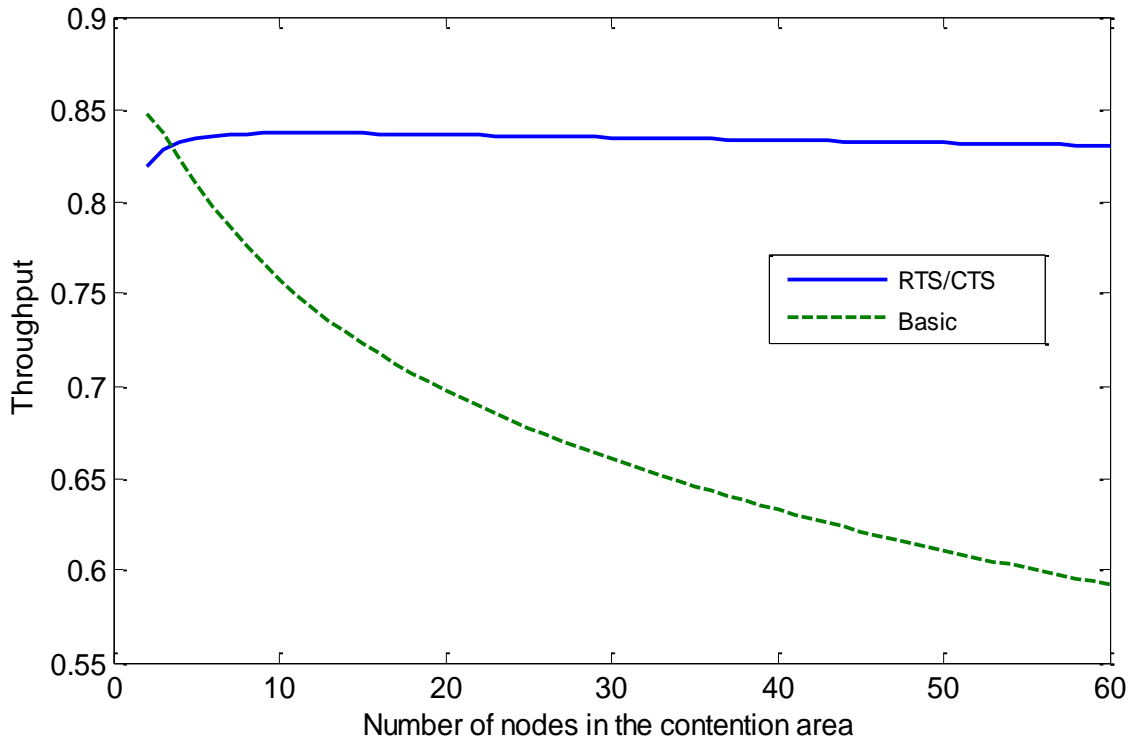
Bianchi model with parameters value for IEEE 802.11 standard

Average message payload	8184 bits
T_c (RTS/CTS)	417 bits
T_s (RTS/CTS)	9569 bits
T_c (basic)	8713 bits
T_s (basic)	8982 bits
σ	50 μ s
Bit rate	1 Mbit/s

Simulations

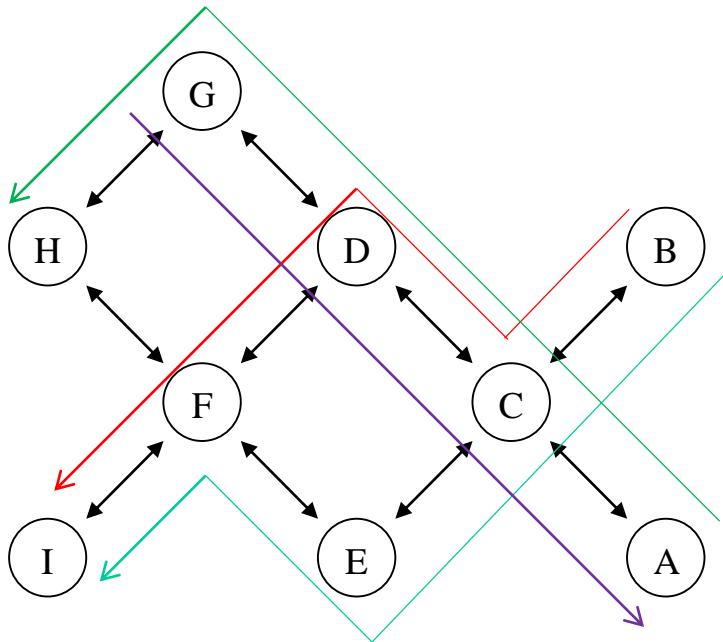
The CSMA-CA model

CSMA-CA Throughput vs. Number of Nodes for the two access mechanisms, RTS/CTS and basic.



Simulations

Fixed network



Session	Path
S_0	B, C, D, F, I
S_1	B, C, E, F, I
S_2	A, C, D, G, H
S_3	G, D, C, A

Simulations - Fixed Network

Link utilization

CSMA-CA protocol with RTS/CTS mechanism				
Link	Cross-layer design		Oblivious layer design	
	Capacity	Utilization (%)	Capacity	Utilization (%)
C-A	0.0687	100	0.0299	100
A-C	0.0609	100	0.1668	18
B-C	0.1217	100	0.1668	36
D-C	0.0687	100	0.0398	75
C-D	0.1212	100	0.0597	100
G-D	0.0687	100	0.1040	29
C-E	0.0614	100	0.0299	100
D-F	0.0604	100	0.0398	75
E-F	0.0614	100	0.1392	21
D-G	0.0609	100	0.0398	75
G-H	0.0609	100	0.1040	29
F-I	0.1217	100	0.1668	36
Total	0.9366	100%	1.0865	57.82%
CSMA-CA protocol with basic mechanism				
Total	0.8933	100%	1.0494	57.32%
ALOHA protocol				
Total	0.8239	100%	0.8621	53.14%

Simulations - Fixed Network

Session rates

		S_0	S_1	S_2	S_3	AGGREGATE LOG RATES
CSMA-CA RTS/CTS	CLD	0.0604	0.0614	0.0609	0.0687	-11.0746
	OLC	0.0299	0.0299	0.0299	0.0299	-14.0444
CSMA-CA BASIC	CLD	0.0575	0.0586	0.0581	0.0655	-11.2637
	OLC	0.0281	0.0281	0.0281	0.0281	-14.2900
ALOHA	CLD	0.0547	0.0590	0.0520	0.0537	-11.6161
	OLD	0.0184	0.0238	0.0184	0.0218	-15.5593

Simulations

Dynamic topology

Simulation period	100 s
Area	200 x 200 m ²
Transmission range	100 m
Node speed	4 m/s
Number of nodes	20
Number of sessions	10
Node density	0.0005 node/m ²
Session density	0.5 session/node

Simulations - Dynamic Topology

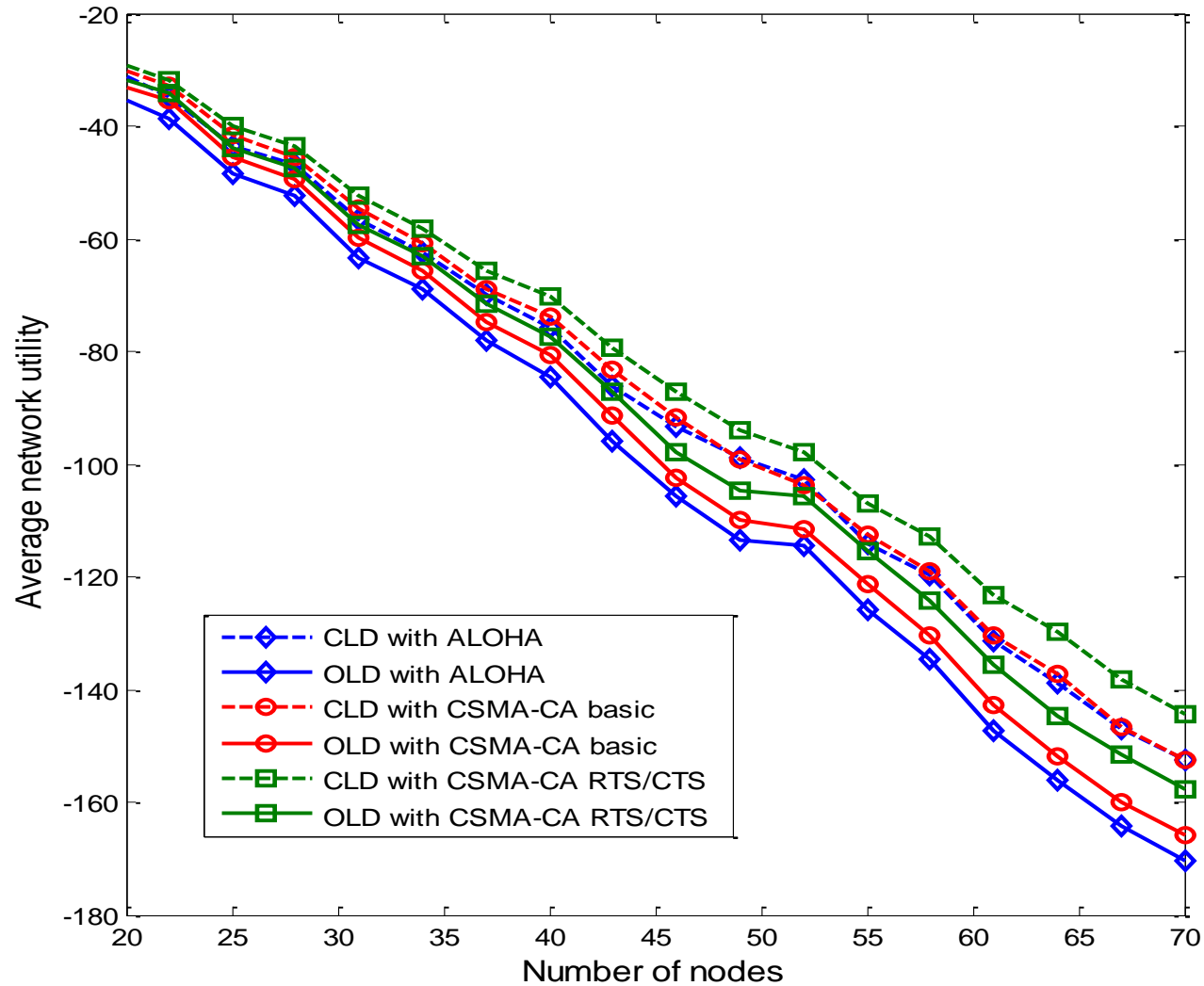
95% confidence interval

Aggregate utilities for 500 snapshots of a dynamic network topology with 70 nodes and 35 sessions.

	CSMA-CA RTS/CTS	CSMA-CA Basic	ALOHA
CLD	[-144.27, -144.64]	[-153.25, -153.66]	[-153.75, -154.16]
OLD	[-159.24, -160.10]	[-168.27, -169.13]	[-171.24, -172.07]

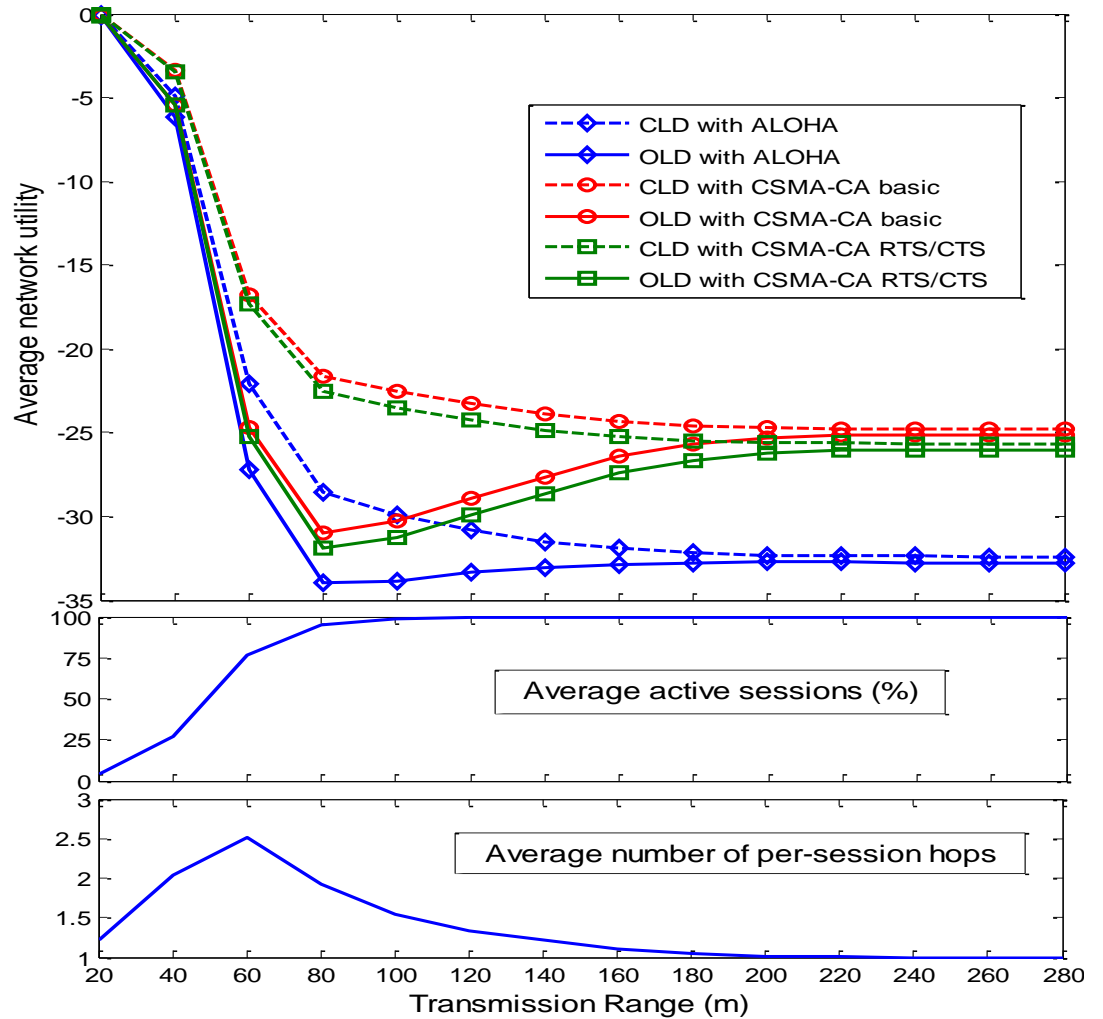
Simulations - Dynamic Topology

Number of nodes



Simulations - Dynamic Topology

Transmission range



Conclusions

- The results indicate the superiority of the CLD over the OLD in dealing with multi-hop network transmissions
 - in medium access opportunity assignments and link capacity utilization.
- The CLD physical capacity gain is roughly proportional to the average number of session hops.
- CLD has the highest impact when sessions travel over relatively longer paths, selecting a good MAC protocol results in the biggest gains when session paths are short.

Next Step: Dealing with Imperfections

- Previous results: assume each node knows everything
 - Its neighbors and their attempt probabilities
 - The sessions and their rates
 - Computations happen instantaneously

- Now: is that still true when
 - Neighborhood changes (and nodes take some time to learn about this)
 - The iterations of the algorithms are taken into account, i.e., there is a delay until we converge to optimal session rates
 - Messages can get lost due to the unreliability of broadcasts in a wireless media

Algorithm Selection

Starting from the NUM

Objective function $\max \sum_{s \in S} U(y_s)$

$U(y_s)$ is the utility function and y_s is the rate of the session s

Session rate Constraint $\sum_{s \in S(l)} y_s \leq C_l$

C_l is the capacity of the link l and $S(l)$ is the set of sessions that use the link l

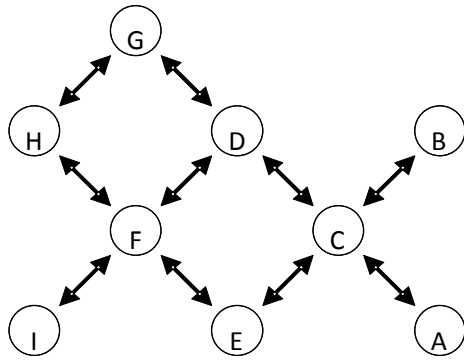
The link capacity constraint $C_l = C_{max} p_l \prod_{n \in N_{to(l)}} (1 - P_n)$

C_{max} is the maximum physical capacity, p_l is the link access probability, P_n is the node access probability, and $N_{to(l)}$ is the nodes whose transmission will impact the receiver of the link l .

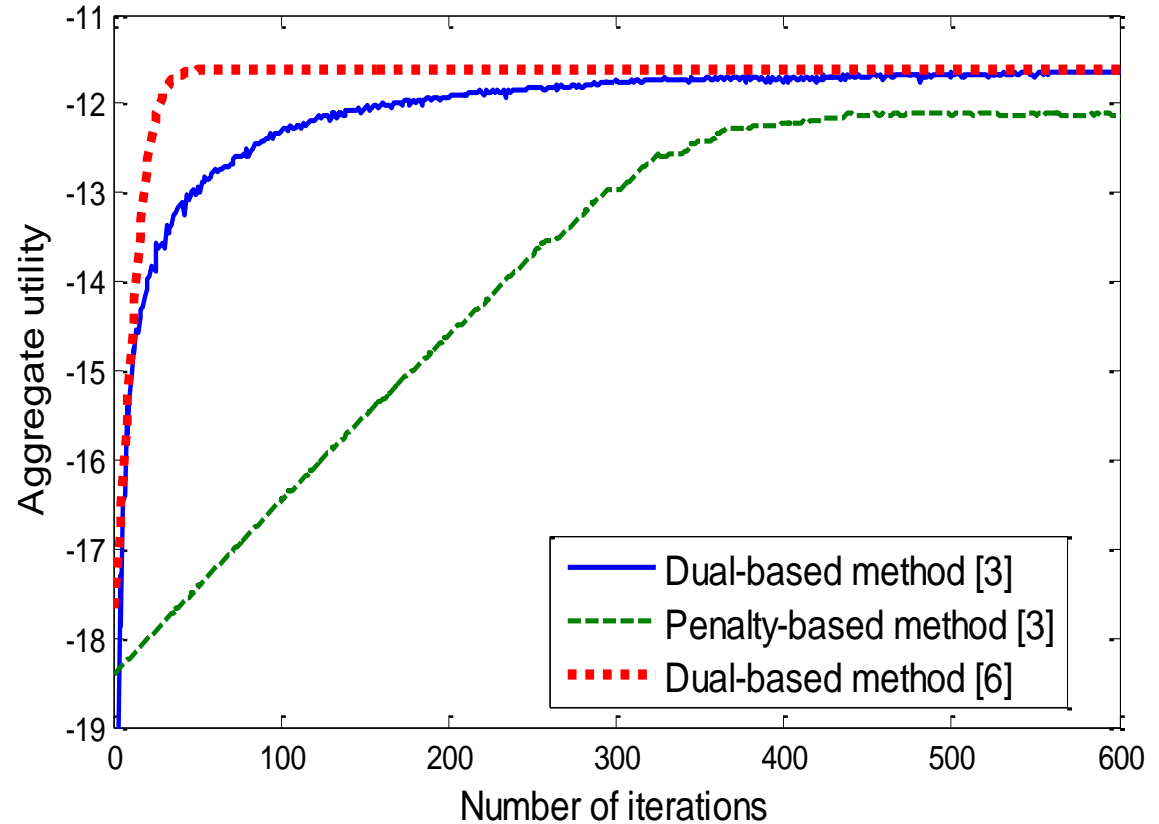
Algorithm Selection: 3 Distributed Algorithms Studied

	Dual-based method [3]	Penalty method [3]	Dual-based method [6]
No. of update functions	2 update functions	2 update functions	1 update function
The update function use	Nodes' attempt probabilities, the link prices, and session rates	Nodes' attempt probabilities, the link prices, and session rates	Link-session prices
Time scale	Two time scales	One time scale	One time scale
Robustness against messages loss	Weak	Very weak	Strong
Update function support	Tests only	Tests only	Mathematically proofed

Algorithm Selection



Session	Path
S_0	B, C, D, F, I
S_1	B, C, E, F, I
S_2	A, C, D, G, H
S_3	G, D, C, A



Algorithm Selection

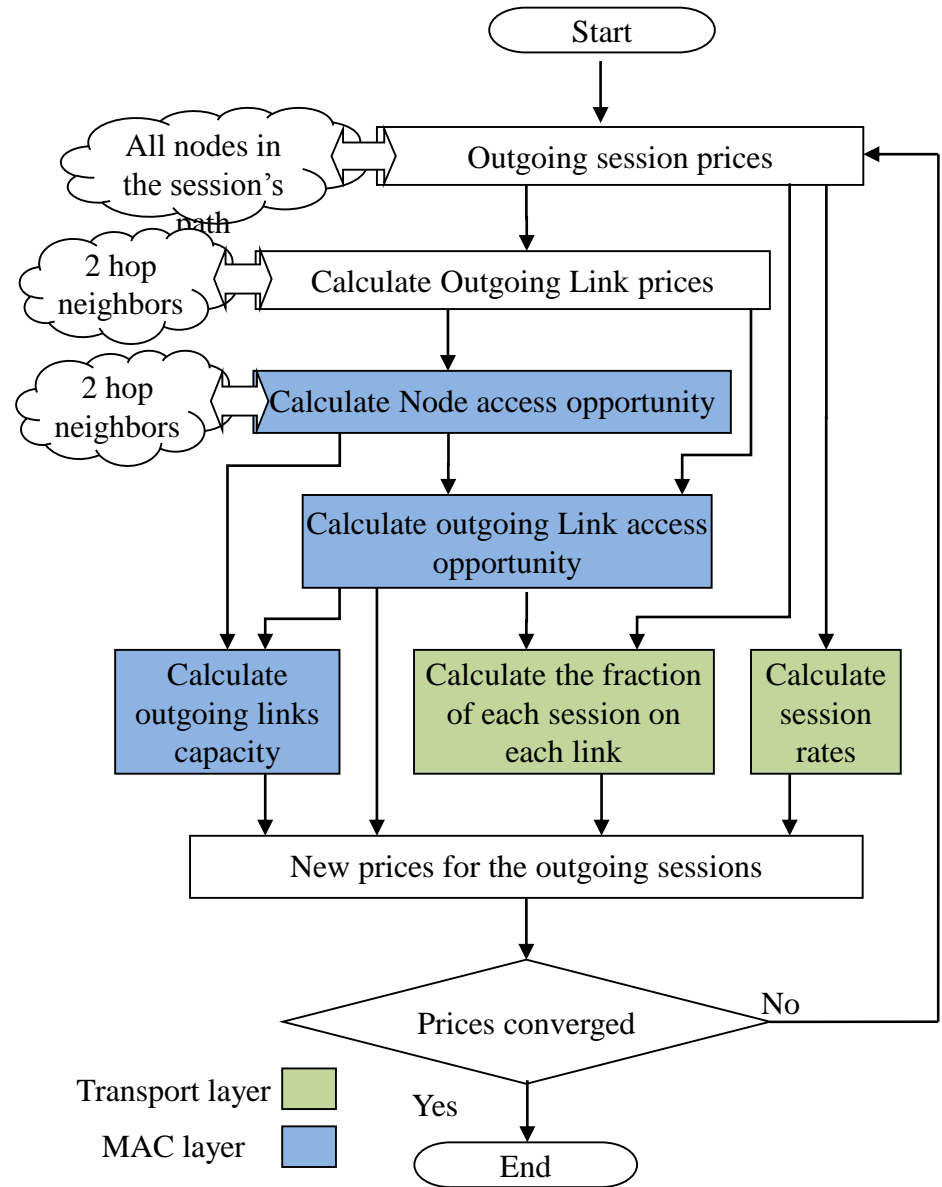
Session	Penalty method [3]	Dual-based method [3]	Dual-based method [6]	Optimum
S_0	0.0513	0.0550	0.0547	0.0545
S_1	0.0608	0.0652	0.0590	0.0632
S_2	0.0390	0.0515	0.0520	0.0492
S_3	0.0453	0.0474	0.0537	0.0535
Utility	-12.1089	-11.646	-11.617	-11.6109

The Simulator

We designed and implemented the simulator in an object-oriented time driven approach using Matlab.

The network is constructed from three main object classes

- Mobile nodes
- Links
- Sessions



Implementation Modifications to Selected Algorithm

Price update step-size (β)

Depends on the topology, network size, and number of links

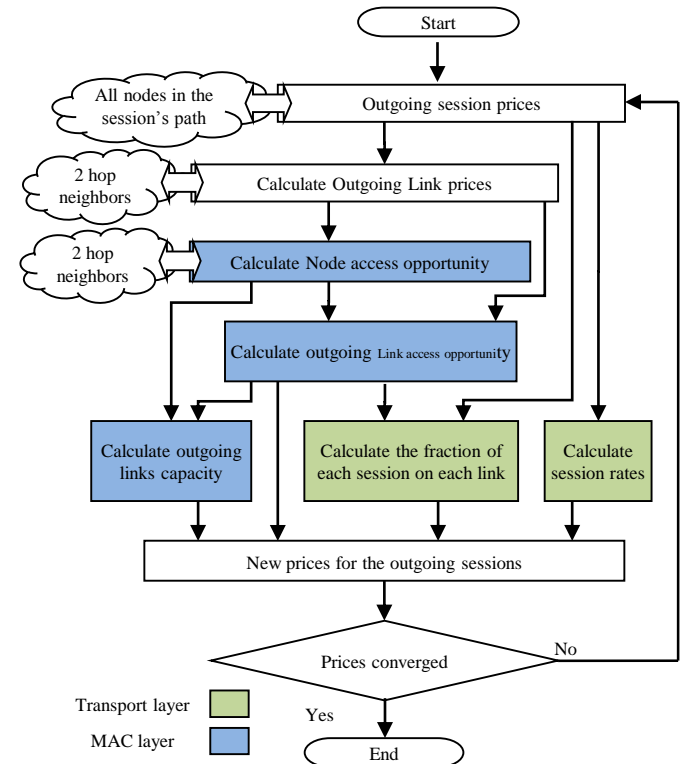
$$\lambda_{ls}^{m+1} = [\lambda_{ls}^m + \beta (y_s^m - x_{ls}^m)]^+$$

Inappropriately *small* β : slow convergence

Inappropriately *large* β : no convergence

Continuous parameters update

It can increase the convergence speed up to 20%



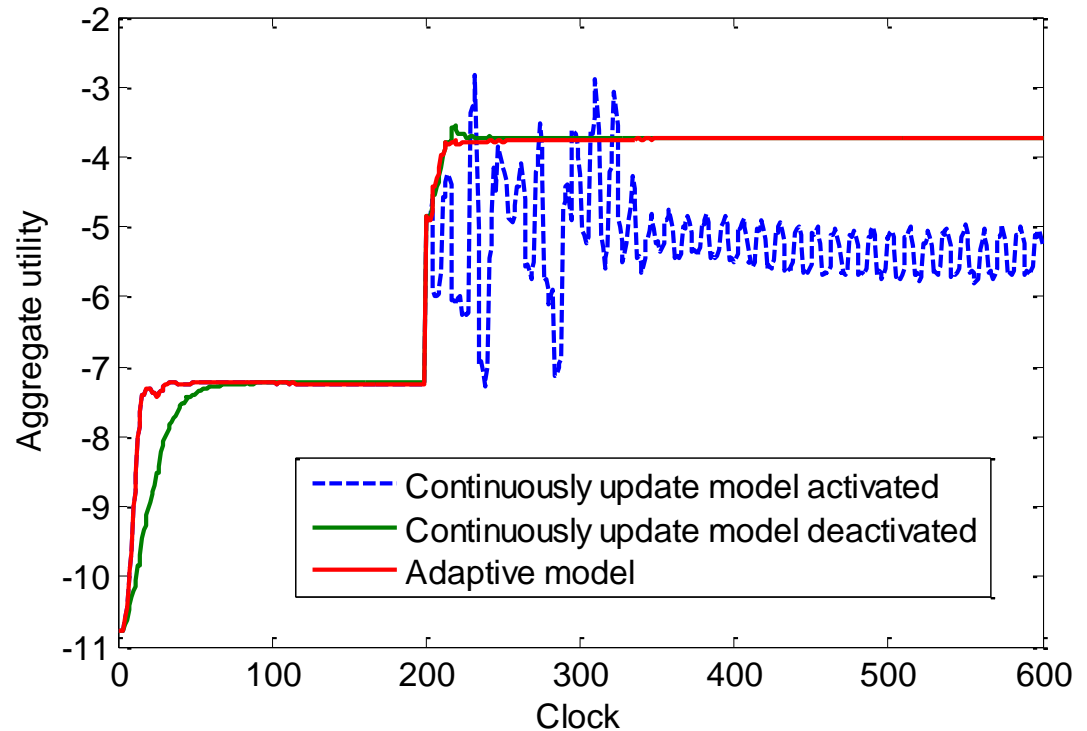
Implementation Modifications

Instability Flag

$$F = \lambda_{ls}^m / (y_s^m - x_{ls}^m)$$

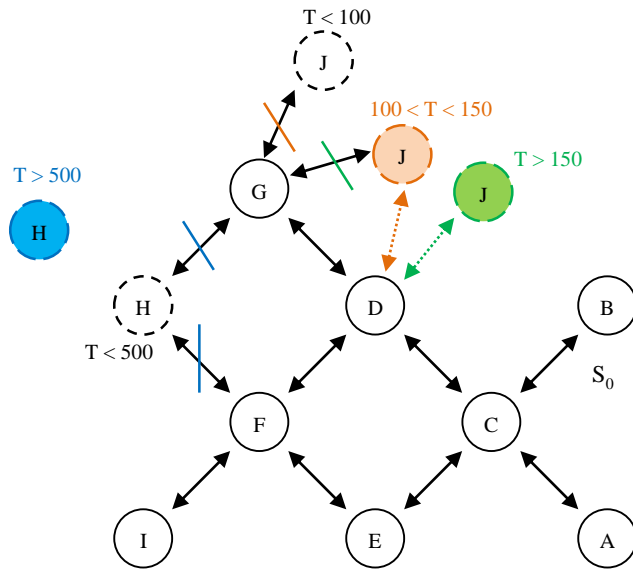
Adaptive β

Adaptive update model



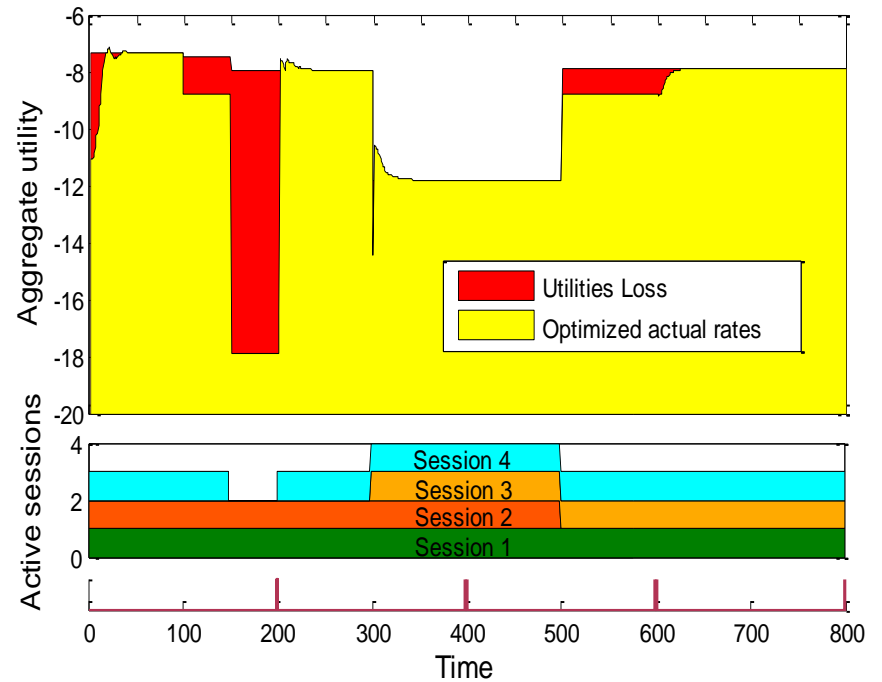
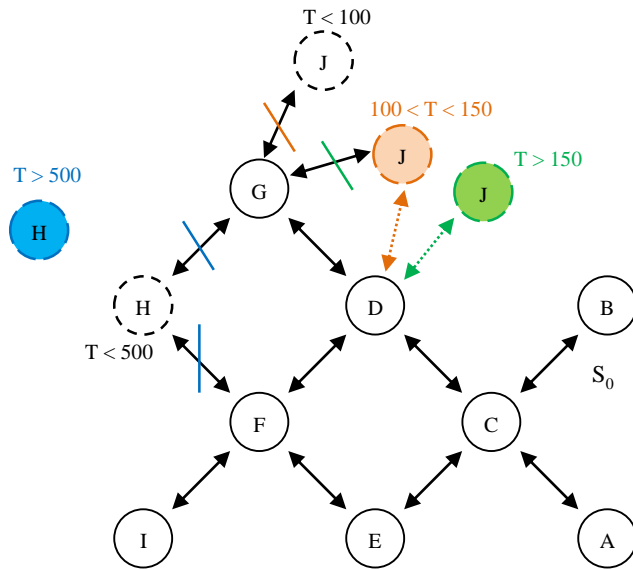
Simulations

Detailed Example



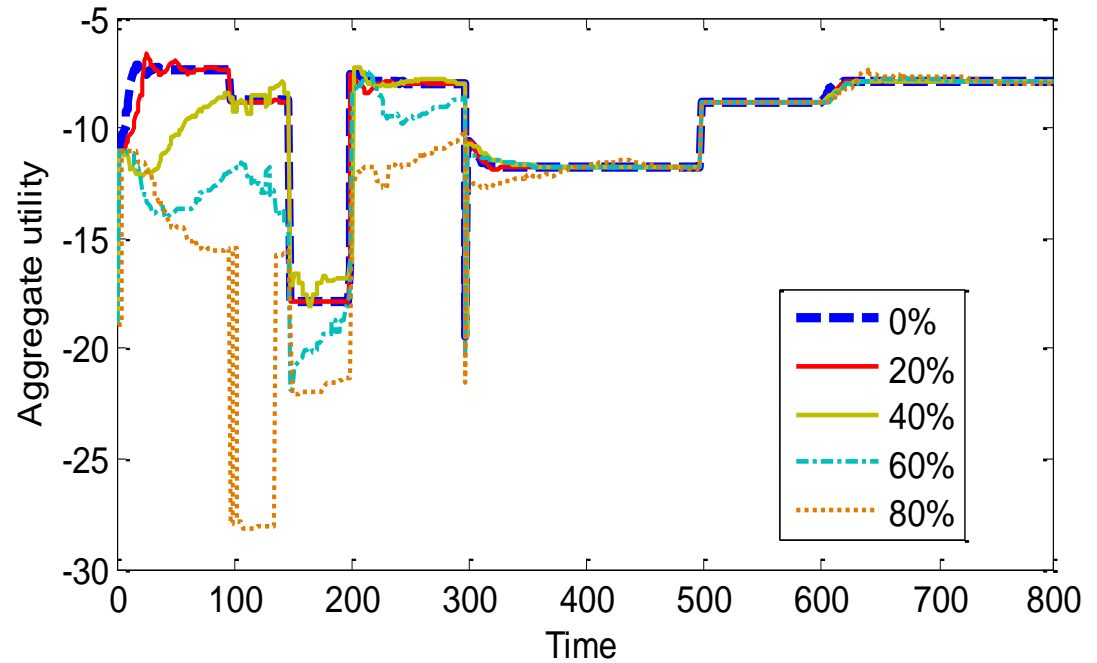
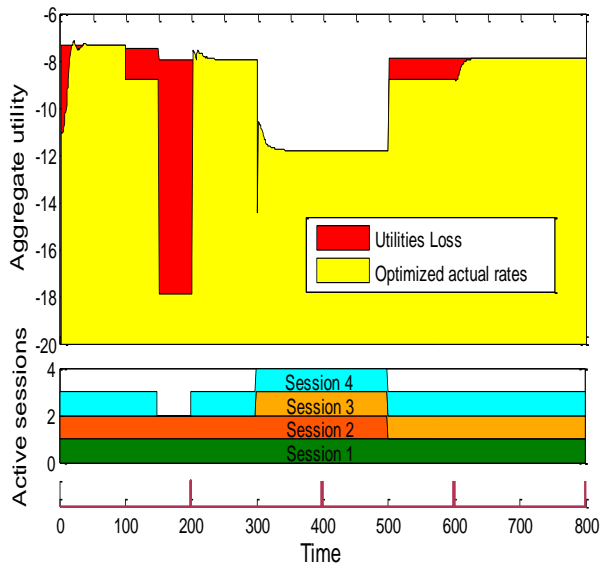
Session	Path	Start time	End time
S_1	B, C, D, F, I	1	800
S_2	A, C, D, G, H	1	800
S_3	G, D, C, A	300	800
S_4	J, G, H, F, E	1	200
rerouted S_4	J, D, C, E	200	800

Simulations - Detailed Example



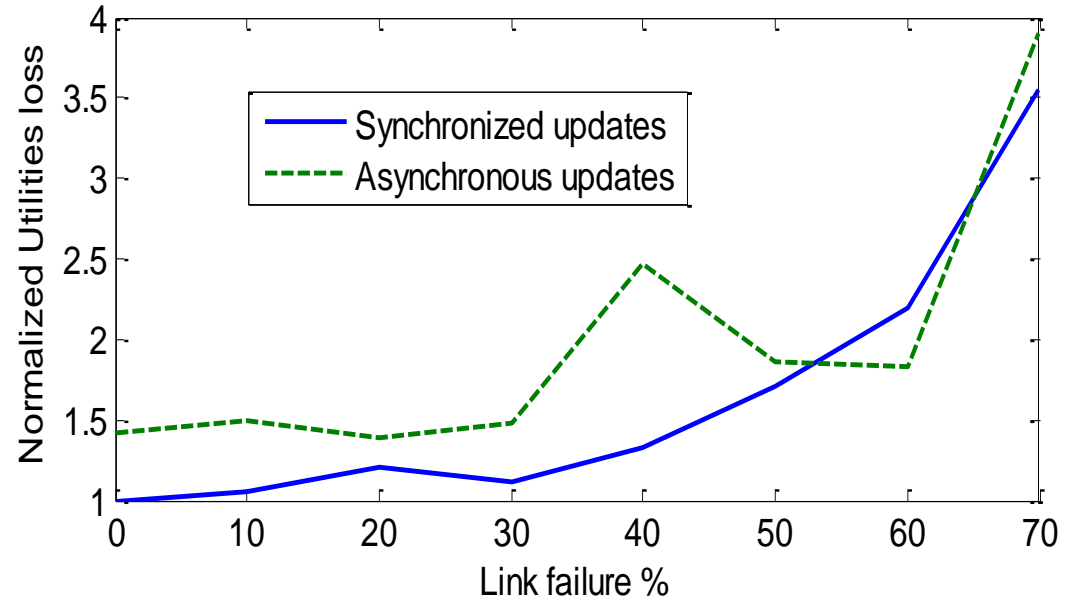
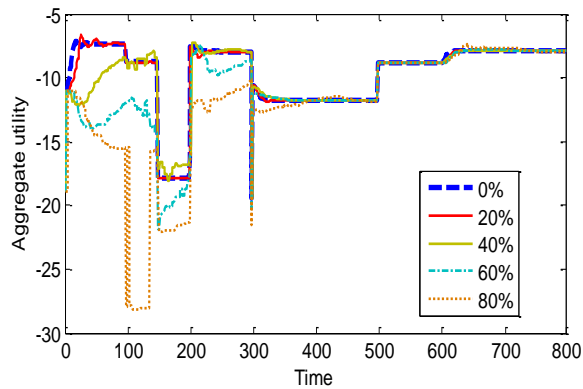
Simulations - Detailed Example

Link Failure



Simulations - Detailed Example

Link Failure



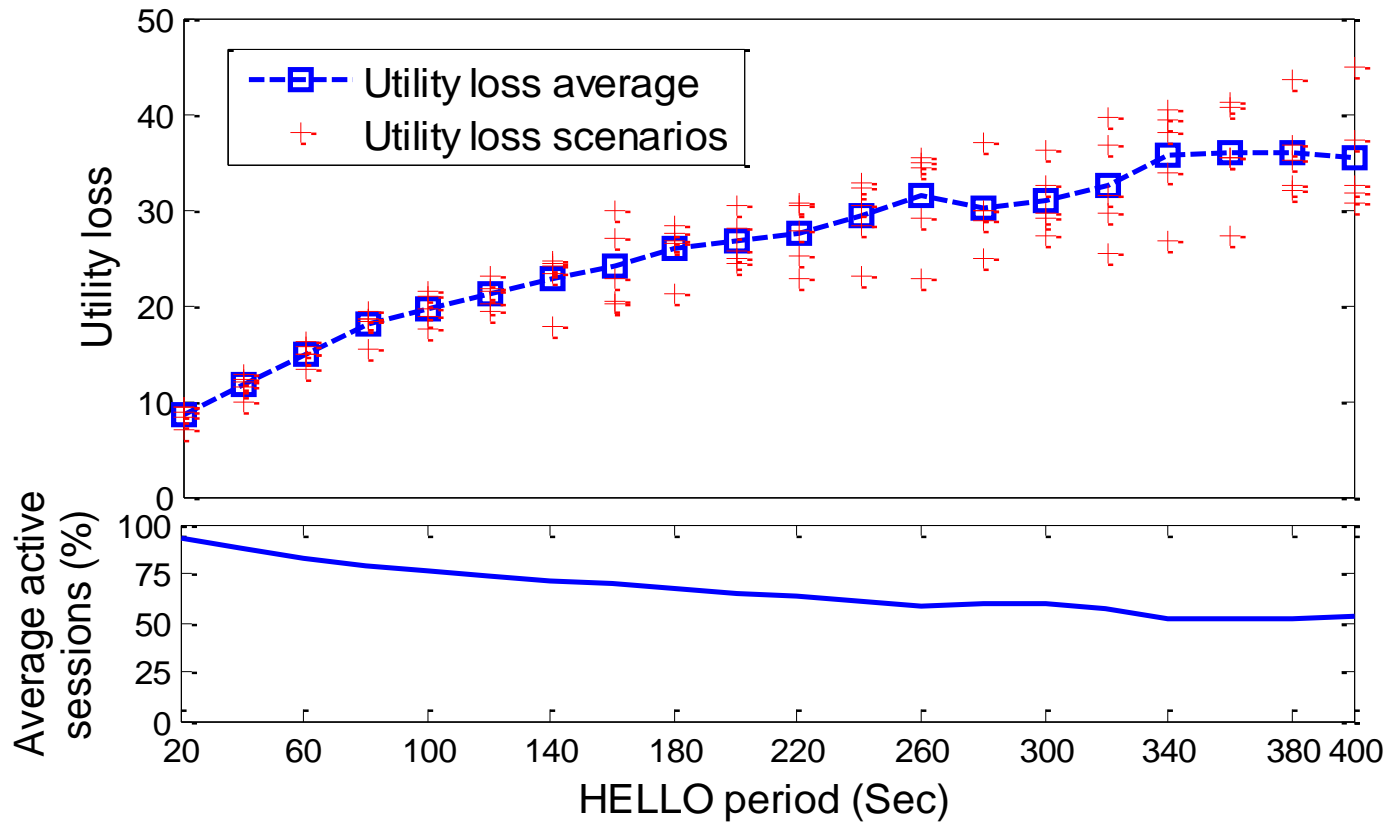
Algorithm Behavior with Different Network Parameters

Default values

Simulation period	200 s
Area	200 x 200 m ²
HELLO period	20 s
Node's transmission range	100 m
Node's speed	4 m/s
No. of nodes	20
No. of sessions	10
Node density	0.0005 node/m ²
Sessions density	0.5 session/node
One-hop transmission time	0.1 s

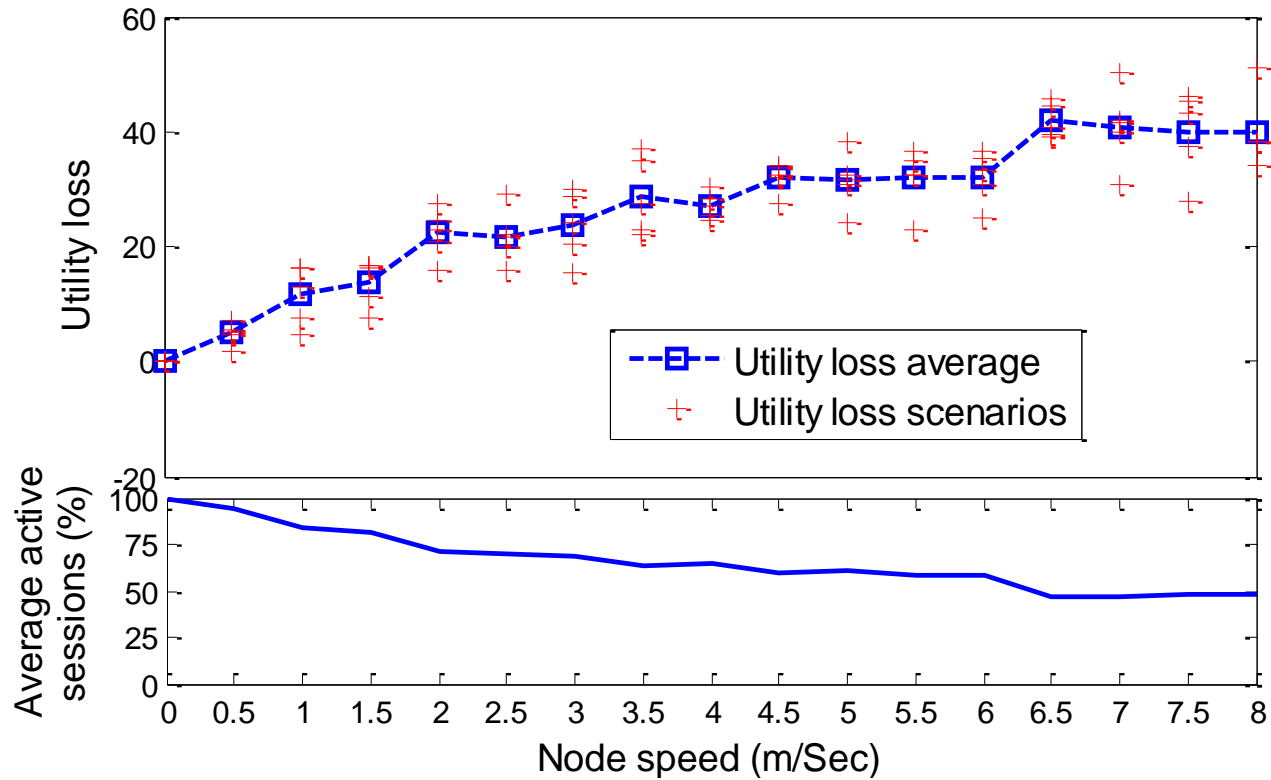
Algorithm Behavior with Different Network Parameters

HELLO period vs. utility loss



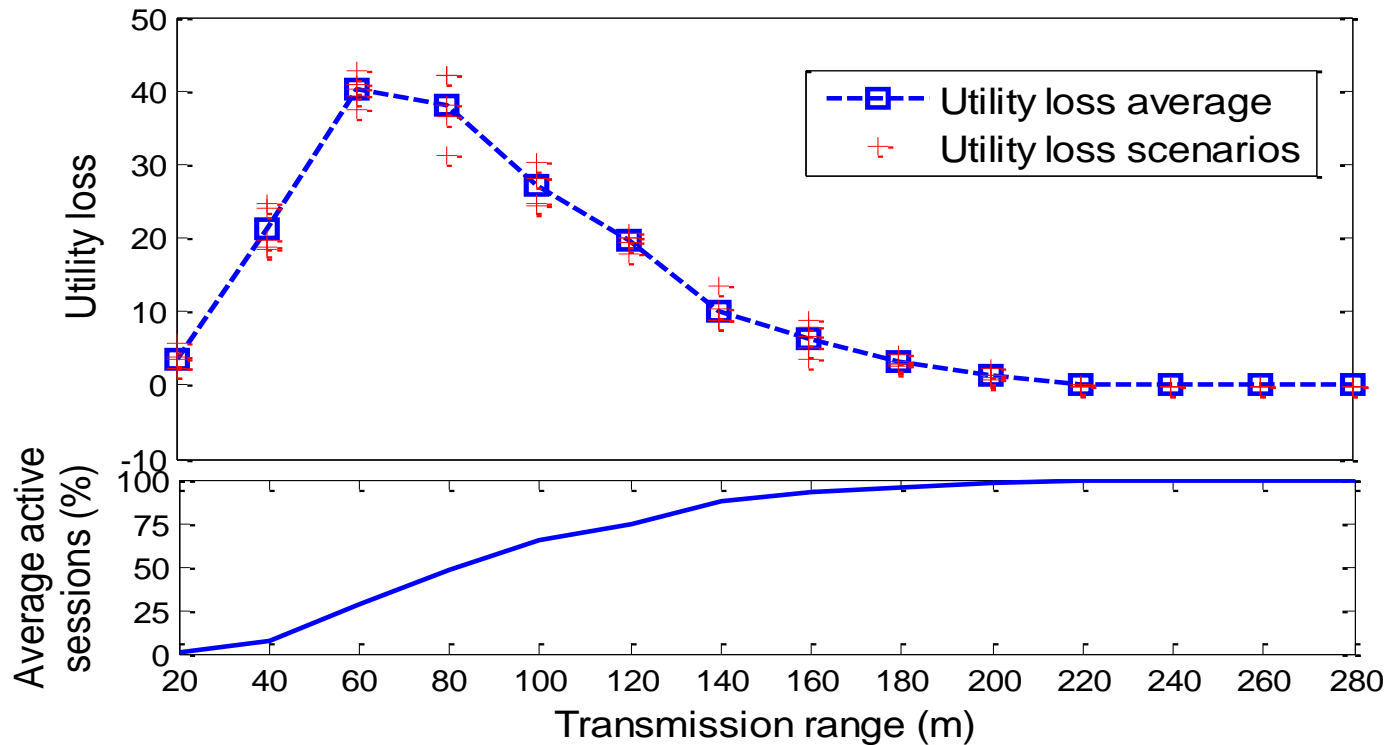
Algorithm Behavior with Different Network Parameters

Node speed vs. utility loss



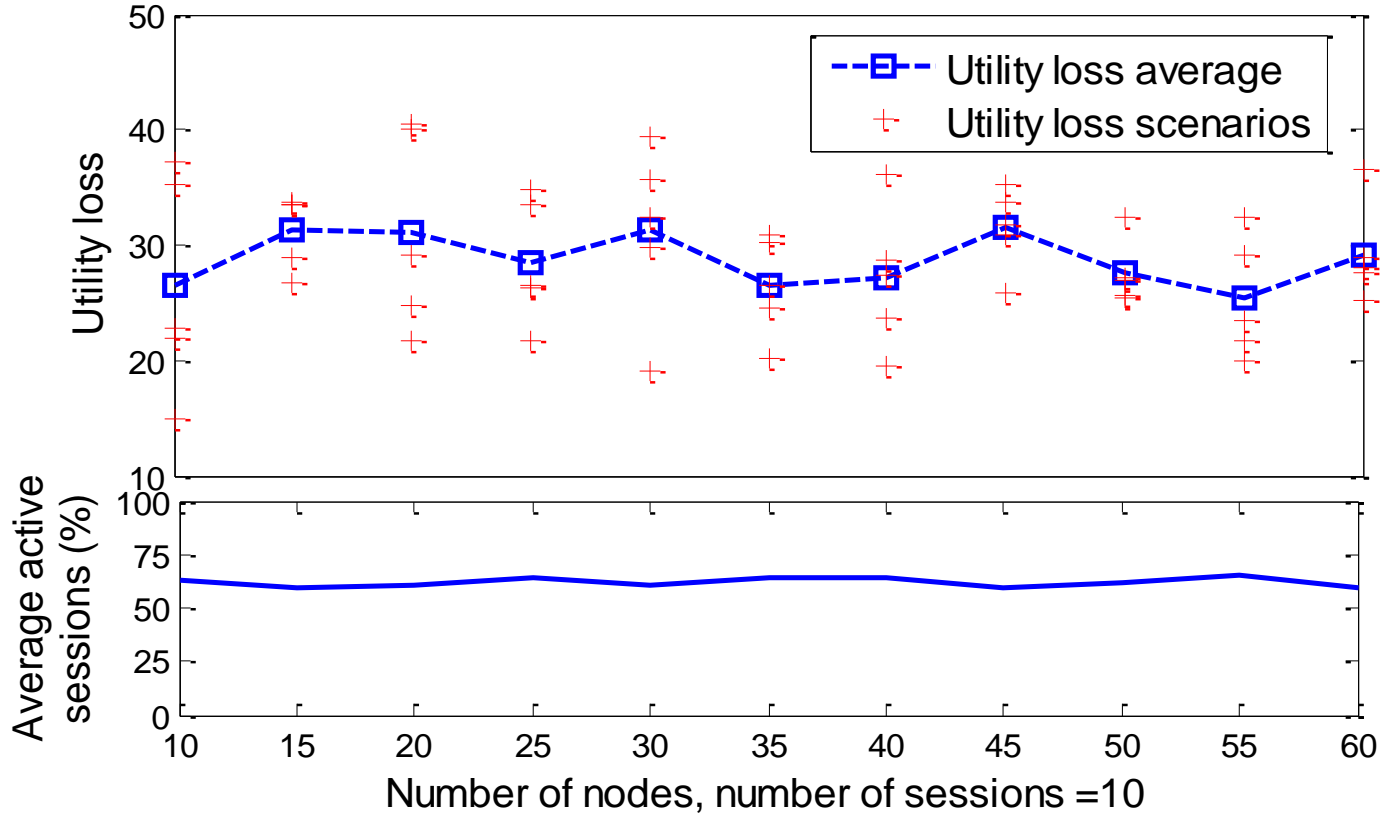
Algorithm Behavior with Different Network Parameters

Transmission range vs. utility loss



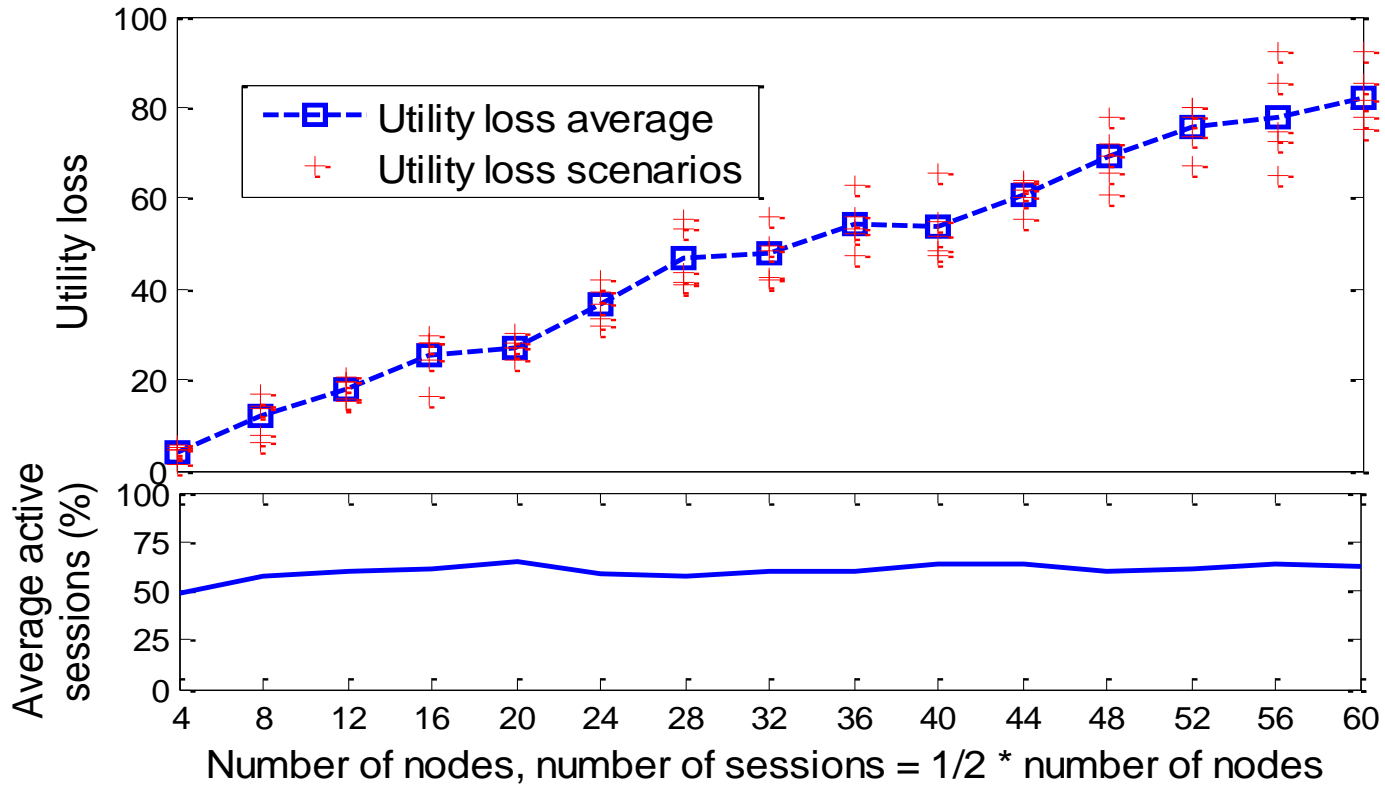
Algorithm Behavior with Different Network Parameters

Node density with fixed number of sessions vs. utility loss



Algorithm Behavior with Different Network Parameters

Nodes density with proportional number of sessions vs. utility loss



Conclusions

- The results show that the selected algorithm, with modifications to ensure fast and stable convergence, performs well even with asynchronous parameter updates.
- With the adaptive update step size the algorithm is robust to different topology changes in the presence of a high link failure rate.
- Link failure only affects the convergence speed and is fairly constant up to the failure rate of about 50%. Increasing link failure rates further tends to exponentially increase the utility losses.
- The convergence speed is proportional to the number of sessions and the ratio of sessions to links.
- Overall, the choice of parameters such as the HELLO interval has to reflect the anticipated change in the local topology as a function to node speed and network density. The higher the rate of local topology change, the shorter a HELLO interval is required to ensure relatively low utility loss.
- With this caveat, however, distributed implementations of optimization algorithms such as the ones studied here in real wireless multi-hop networks seem promising.

Conclusions from Our Work

- CLD holds promise for realistic MANETs
- Still to do
 - Account for messaging overhead
 - Initial results seem to indicate that overheads are small and may be negligible for higher wireless link rates (11 Mbps, etc.), but may kill the idea for low-rate wireless links (tactical radios have a bandwidth of 28 – 128 kbps)
 - Include network layer
 - First: do existing routing protocols matter?
 - Next: how to find best routes based on link and session prices (ant routing analogy)
 - Also: move it all to NS-3 ☹

Cross-Layering: Key Questions

- What is the right framework for crosslayer design?
- What are the key crosslayer design synergies?
- How to manage crosslayer complexity?
- What information should be exchanged across layers, and how should this information be used?
- How to balance the needs of all users/applications?