This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Unported License.
See http://creativecommons.org/licenses/by-nc-nd/3.0/

A System-Theoretic Clean Slate Approach to Secure Protocols for Wireless Networks

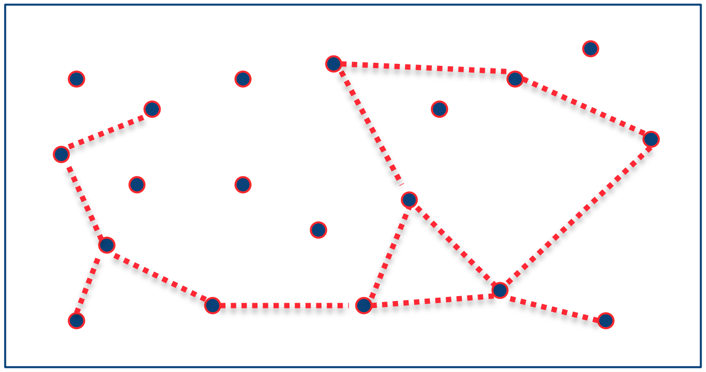
Jonathan Ponniah
Yih-Chun Hu
P. R. Kumar

Dept. of Electrical and Computer Engineering
Texas A&M University

CSoI Big Data Workshop
March 18-20, 2013
Honolulu

Email: prk@tamu.edu
Web: http://cesg.tamu.edu/faculty/p-r-kumar/

Focus: Ad hoc multi-hop wireless networks



- Packets possibly multi-hop from sources to destinations
- Require no pre-existing infrastructure
- No centralized controller
- Distributed decision making: Nodes themselves determine power levels, transmit times, routes, schedules
- Require multiple protocols to operate

Motivation

- Usual approach, including in wireless networks, has been
  - Develop protocols for good performance
- Then
  - Some ATTACK is identified
  - A DEFENSE is developed for that attack
  - Then another ATTACK is identified
  - Another DEFENSE for that attack
  - …

Wormhole attack
Sybil attack
Rushing attack

ARIADNE
TESLA

- Result
  - A sequence of patches
  - An arms race
- Difficulty
  - We don't know what other attacks are possible
  - No guarantees of security

The problem of defending against attacks

- Given a protocol, we can harden it against a particular attack
- Result: a hardened protocol that is immune to that *particular* attack
- But can we develop a protocol that is immune to *all* attacks?
- We cannot even list all attacks, let alone develop defenses attack by attack
- So what can we do?

Need for a system-theoretic approach

- System-theoretic view: Every attack is a *policy* in a given model of the system

- So the goal is to develop a *Model Based Defense*
  - Assume a model of capabilities for the attacker
  - Defend against all capabilities

- Defend against Byzantine behavior of *malicious* nodes

- The *good* nodes have to publish a protocol and follow it
- Now we get a *game* between protocols and Byzantine behavior
- *What is a model of the system for which we can develop such a theory and a complete suite of protocols?*

But what about Performance?

- There may be many protocols that can defend against attacks
  - How do we choose among them?
  - Reminiscent of "throughput optimality" vs. "throughput optimality with low delay"

- We can postulate a performance measure: A *Utility function* $U(x)$
- Now we get a zero-sum game:

$$\underset{\text{Protocols announced and followed by good nodes}}{Max} \quad \underset{\text{Byzantine behavior of bad nodes}}{Min} \quad U(x)$$

- Can we develop a max-min optimal super-protocol?
  - A complete suite of protocols
- Further questions: What type of performance measure?
  - Long term, Transient performance, etc

Goals

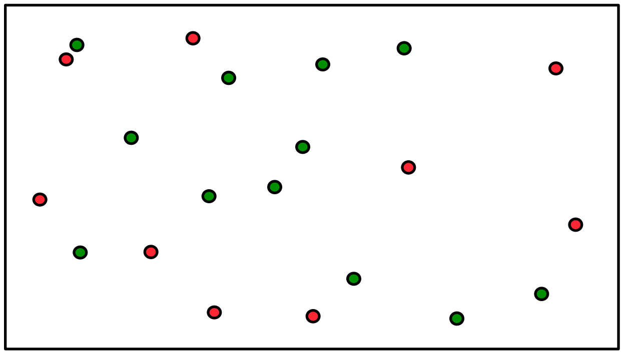
- Can we develop a system-theoretic principled and holistic approach to security?
  - Where Security is addressed first, not an afterthought
  - Performance is addressed second; and it is optimized while preserving security
  - Reverse of the usual approach
- Security objective
  - A system-theoretic clean slate approach to secure wireless networking
  - Provable security: Guaranteed if model assumptions satisfied
    - Subsequently, model assumptions can be attacked/challenged
  - Develop a complete suite of algorithms/protocols
  - An "existence theorem," if you will, or as providing algorithms
- Also a performance guarantee: Max-Min Optimality
  - Max is over protocols
  - Min is over all actions of malicious nodes

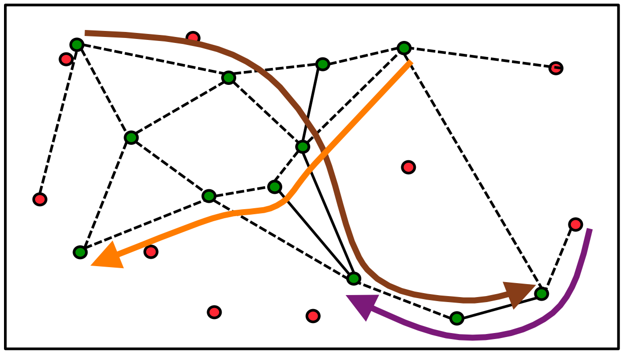
A lot of explanation is clearly needed …

Basic objective

- A complete suite of algorithms/protocols that takes you

- From startup
  - With just a set of nodes
  - Some good
  - Some bad
  - Good nodes don't know who the bad nodes are

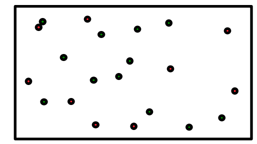




- To an optimized functional network carrying data reliably

What can go wrong with a network formed in presence of bad nodes?

- Some nodes are bad. What can go wrong?



- Lots of things. A bad node could
  - Refrain from relaying a packet
  - Advertise a wrong hop count
  - Advertise a wrong logical topology
  - Jam
  - Cause packet collisions
  - Behave uncooperatively vis-à-vis medium access
  - Disrupt attempts at cooperative scheduling
  - Drop an “ACK”
  - Refuse to acknowledge a neighbor’s handshake
  - Behave inconsistently

“Byzantine” behavior

Main results on security-cum-performance

Theorem

- The described protocol suite yields a network that is Max-Min optimal with respect to the utility function

$$\underset{\text{Protocols}}{Max} \quad \underset{\text{All behaviors of bad nodes}}{Min} \quad U(x)$$

- Actually, the protocol suite achieves a stronger result: It attains Min-Max optimality and is thus a *saddle-point*:

$$\underset{\text{All behaviors of bad nodes}}{Min} \quad \underset{\text{Protocols}}{Max} \; U(x)$$

- In fact the protocol suite provides an even stronger result: It attains

$$\underset{\text{Bad nodes can choose to either Jam or Cooperate}}{Min} \quad \underset{\text{Protocols}}{Max} \; U(x)$$

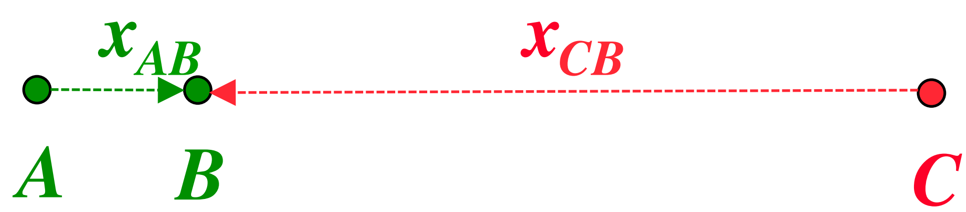
Bottom line

$$\underset{\text{Bad nodes can choose to either Jam or Cooperate}}{Min} \quad \underset{\text{Protocols}}{Max} \; U(x)$$

- Bad nodes are restricted to Jamming or Cooperating consistently on each concurrent transmission set
- Nobody can prevent jamming or cooperating
- Other Byzantine behaviors are ruled out
  - Dropping ACKs, lying, etc.

Why would a bad node ever cooperate?

- $U(x) = \mathrm{Min}(x_i)$



- C is far away
- Low signal/interference at B
- If C jams, it can only slightly reduce x_{AB}

$$\lim_{|BC|\to\infty} x_{AB} = x_{AB}^{Max}$$

- If C pretends to be good, it gets an equal share, and

$$\lim_{|BC|\to\infty} x_{AB} = 0$$

- C causes more harm by cooperating and getting “fair share”

Fundamental ingredients of our approach

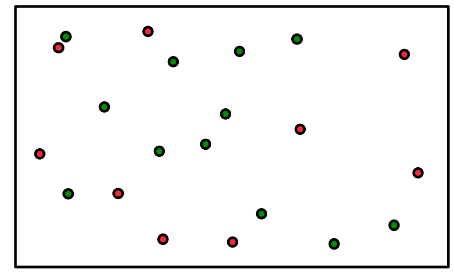
- Standard cryptographic primitives are assumed
  - All packets encrypted
  - Bad nodes cannot create fake packets, or alter good packets without getting caught, etc.

- And, importantly: Clocks and synchronization
  - Without a notion of *time*, we cannot even talk of throughput
  - Without throughput we cannot talk of network Utility
  - So *time* is an essential ingredient

  - With notion of *common* time, nodes can cooperate temporally, can share resources in a time-based way
  - Cooperative scheduling, etc., will be possible
  - So *synchronization* will be a fundamental ingredient

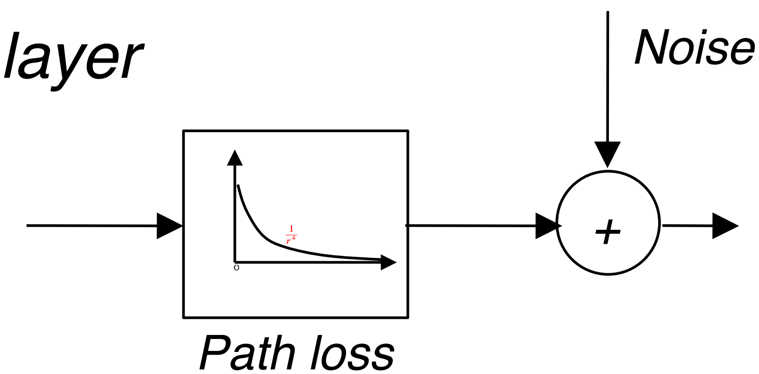
Model: Assumptions – 1

- Bounded domain
- n nodes, some bad
- Minimum distance between any pair of nodes
- Nodes are not mobile



- Finite set of modulation schemes

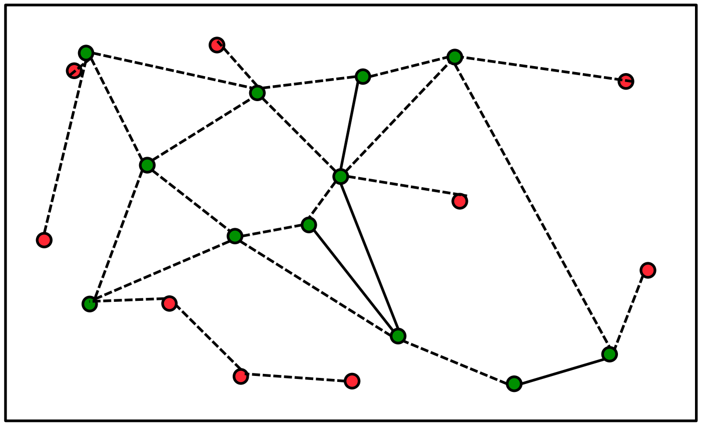
- *Or can assume more about physical layer*
  - *Max power constraint at each node*
  - *Noise at each node*
  - *Path loss is a function of distance*
  - *SINR based rate*



July 13, 2010 , P. R. K

Model: Assumptions – 2

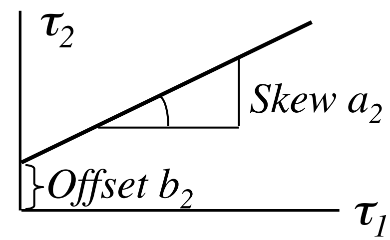
- Connectedness
  - Suppose all nodes transmit at Max power
  - Let us say there is an edge between each pair of nodes (i, j) which can communicate at lowest rate modulation scheme
  - *Or there is an edge between each pair of nodes* (i, j) *an for which* $SINR_{ij}$ *and* $SINR_{ji}$ *both exceed* $SINR_{threshold}$
  - Assumption
    - » Resulting graph is connected
    - » Subgraph of good nodes is also connected



July 13, 2010 , P. R. K

Model: Assumptions – 3

- Affine clock at each node
  - $0 < 1-\varepsilon \leq \text{Skew} \leq 1 + \delta$ for all nodes

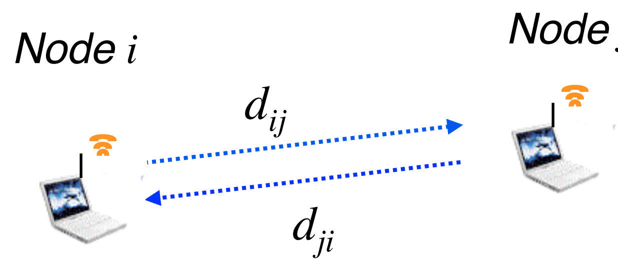


- Digital clocks
  - Clocks tick "digitally" – causing imprecision
  - Clocks wrap around
- System start-up
- All nodes are born within a bounded time of each other
  - Primordial birth

July 13, 2010 , P. R. K

Model: Assumptions – 4

- Packets take a delay d_{ij} from node i to node j



- Cryptographic assumptions
  - Each node has a private key, public key

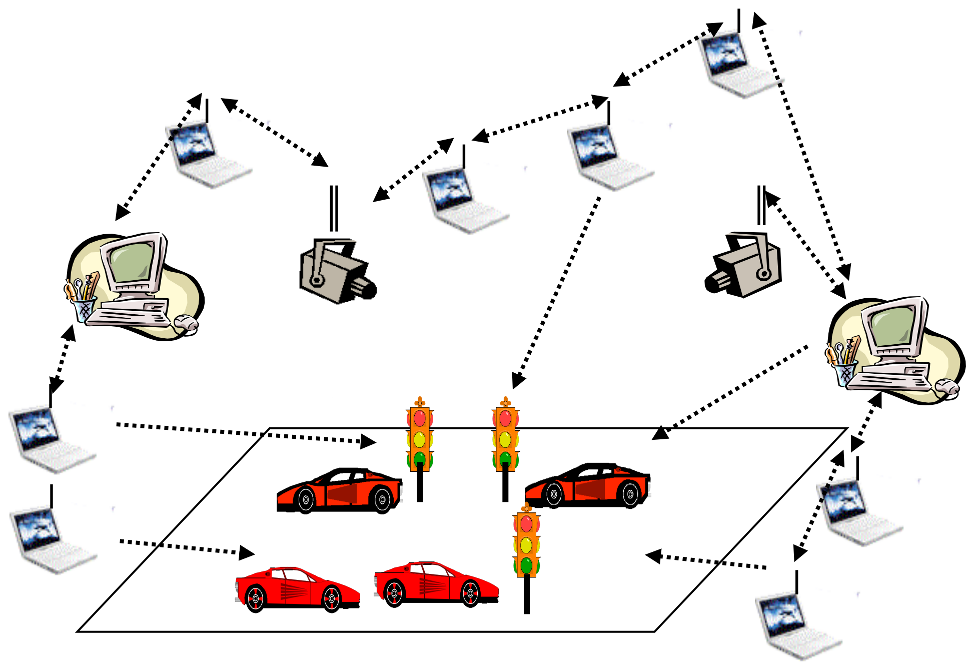
- Network Utility function

$$U(x) = \sum_{\text{All conforming pairs } (i,j)} U_{ij}(x_{ij})$$

Clocks over wireless networks

Clock synchronization over wireless networks

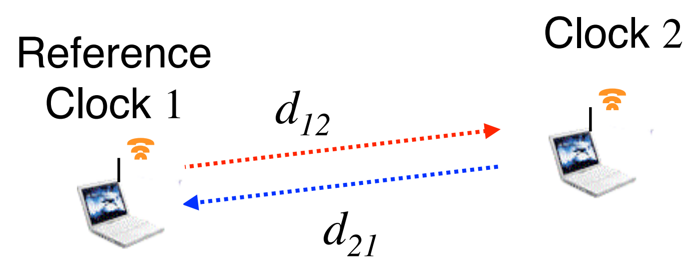
- Knowledge of time is important in Networks
  - Communication network protocols
  - Sensor network applications
  - Networked control
  - *And for security: Clock synchronization can be helpful vis-à-vis security*



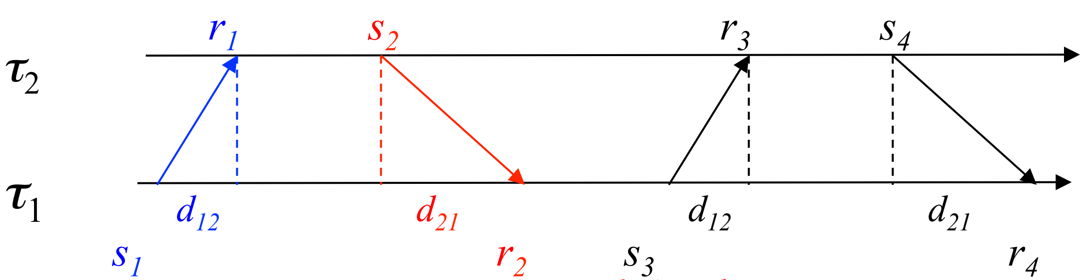
- However no two clocks agree
- How to synchronize clocks in wireless networks?
- And what about security of clock synchronization itself?

It is impossible to synchronize two clocks

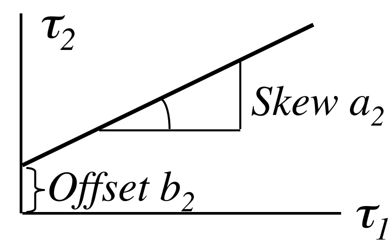
- **Theorem (Graham & K ‘04)**
  - It is impossible to determine $(d_{12}, d_{21}, a_2, b_2)$ through any packet exchanges



$$r_1 = a_2(s_1 + d_{12}) + b_2$$



$$s_2 = a_2(r_2 - d_{21}) + b_2$$



$$\begin{bmatrix} r_1 \\ s_2 \\ r_3 \\ s_4 \\ \cdots \end{bmatrix} = \begin{bmatrix} s_1 & 1 & 0 & 1 \\ r_2 & 0 & -1 & 1 \\ s_3 & 1 & 0 & 1 \\ r_4 & 0 & -1 & 1 \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} a_2 \\ a_2 d_{12} \\ a_2 d_{21} \\ b_2 \end{bmatrix}$$

Rank 3:
Cannot estimate 4 parameters

So what is determinable?

The skew a_2 can be estimated correctly.

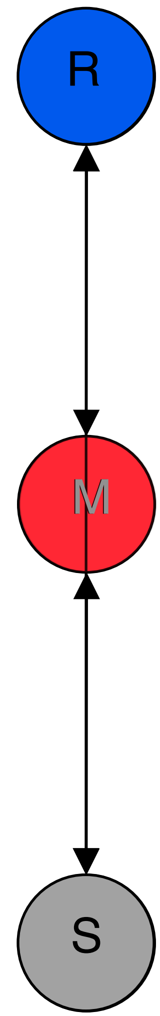
The round-trip delay $(d_{1j} + d_{j1})$ can be estimated precisely.

The sender can predict the receiver's time at which receiver receives a packet.

Interplay between clock synchronization and security

A fundamental possibility in wireless networks: Man-in-the-middle

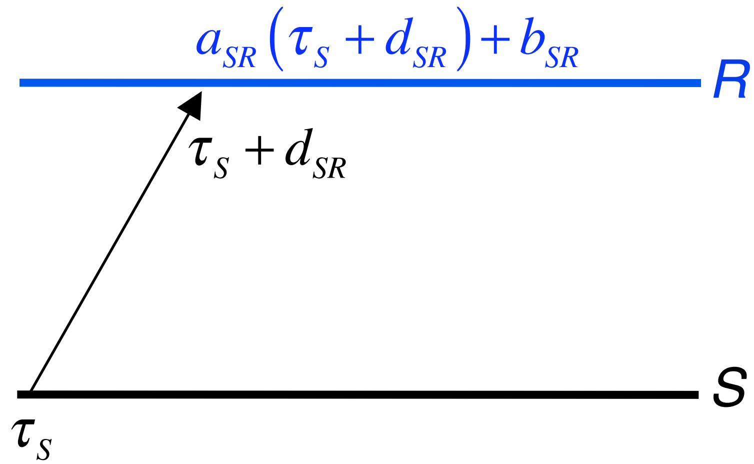
- To what extent can a *Man-in-the-Middle* remain undetected?
- Can we synchronize clocks in spite of the *Man-in-the-Middle?*
- Suppose all messages are encrypted: Then
  - M cannot decrypt any messages between S to R
  - M cannot alter any messages between S and R
  - M cannot create any fake messages between S and R
- So M has to provide a *logical channel* between S and R



But what can Man-in-the-Middle do with respect to *Delay*?

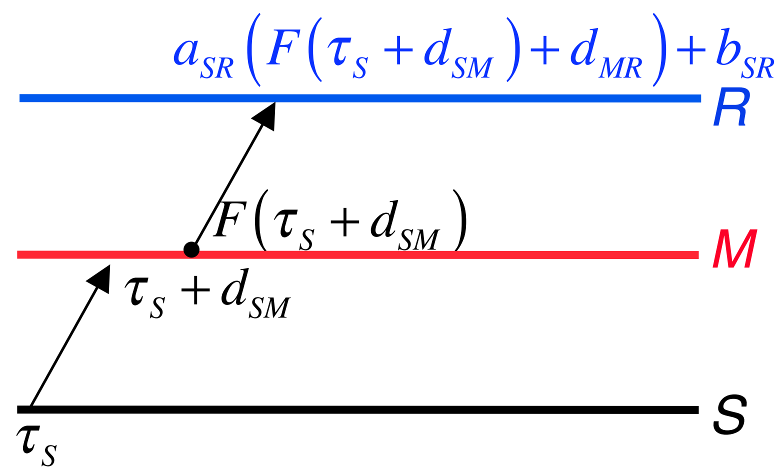
Affine forwarding policy

- Without Man-in-the-Middle
  - Time received is affine in τ_S
  - Coefficient a_{SR} is estimate of skew



- With Man-in-the-Middle

- M's forwarding policy
  - Packet received at τ
  - Forwarded at $F(\tau)$



- Receipt time $a_{SR}\left(F\left(\tau_S + d_{SM}\right) + d_{MR}\right) + b_{SR}$

  has to be affine in τ_S

- So $F(\tau)$ has to be affine in τ

July 13, 2010 , P. R. K

Expansionary affine forwarding policy

- Consider affine forwarding policy

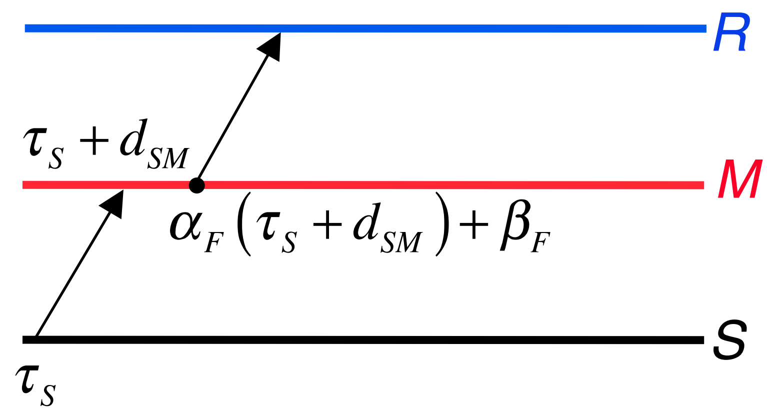
$$F(\tau) = \alpha_F \tau + \beta_F$$

- Causality

- Forwarding packet can only take place *after* receiving packet

$$\alpha_F(\tau_S + d_{SM}) + \beta_F \geq \tau_S + d_{SM} \text{ for all } \tau_S$$

- So $\alpha_F \geq 1$

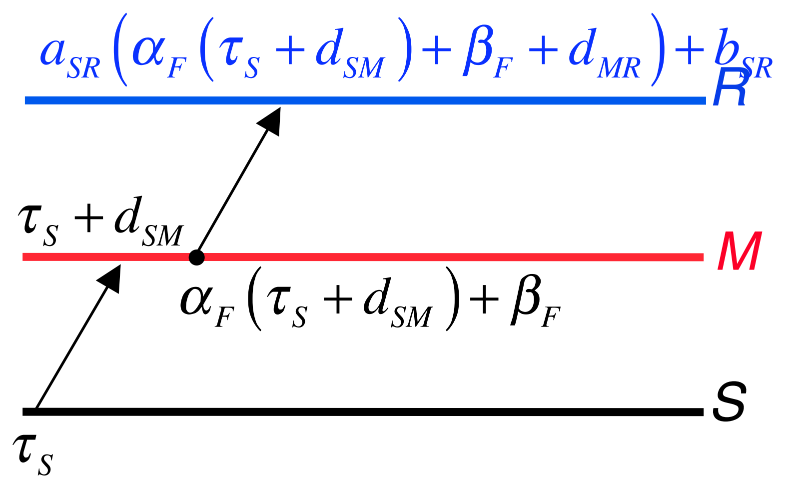


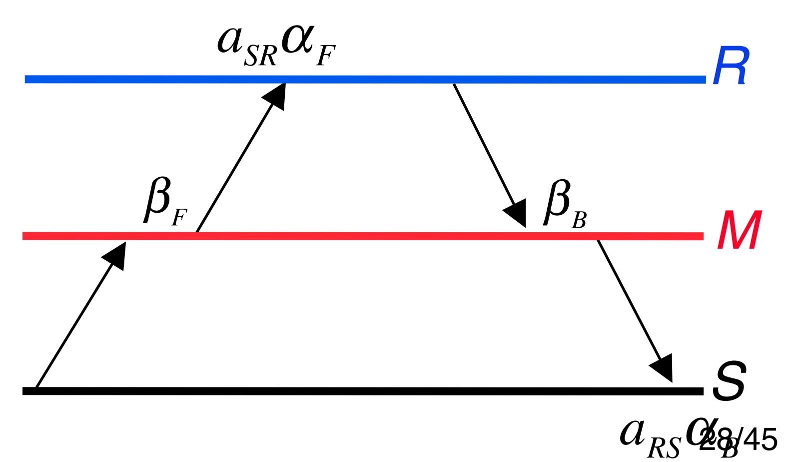
M can only add a *constant delay* to all packets

- Estimate of skew = Coefficient of τ_S
- So skew estimate made by R with reference to S is $a_{SR}\alpha_F$
- *Backward* path skew estimate made by S with reference to R is $a_{RS}\alpha_B$
- But *product* of skew estimates has to be 1

$$a_{SR}\alpha_F a_{RS}\alpha_B = \alpha_F \alpha_B = 1$$

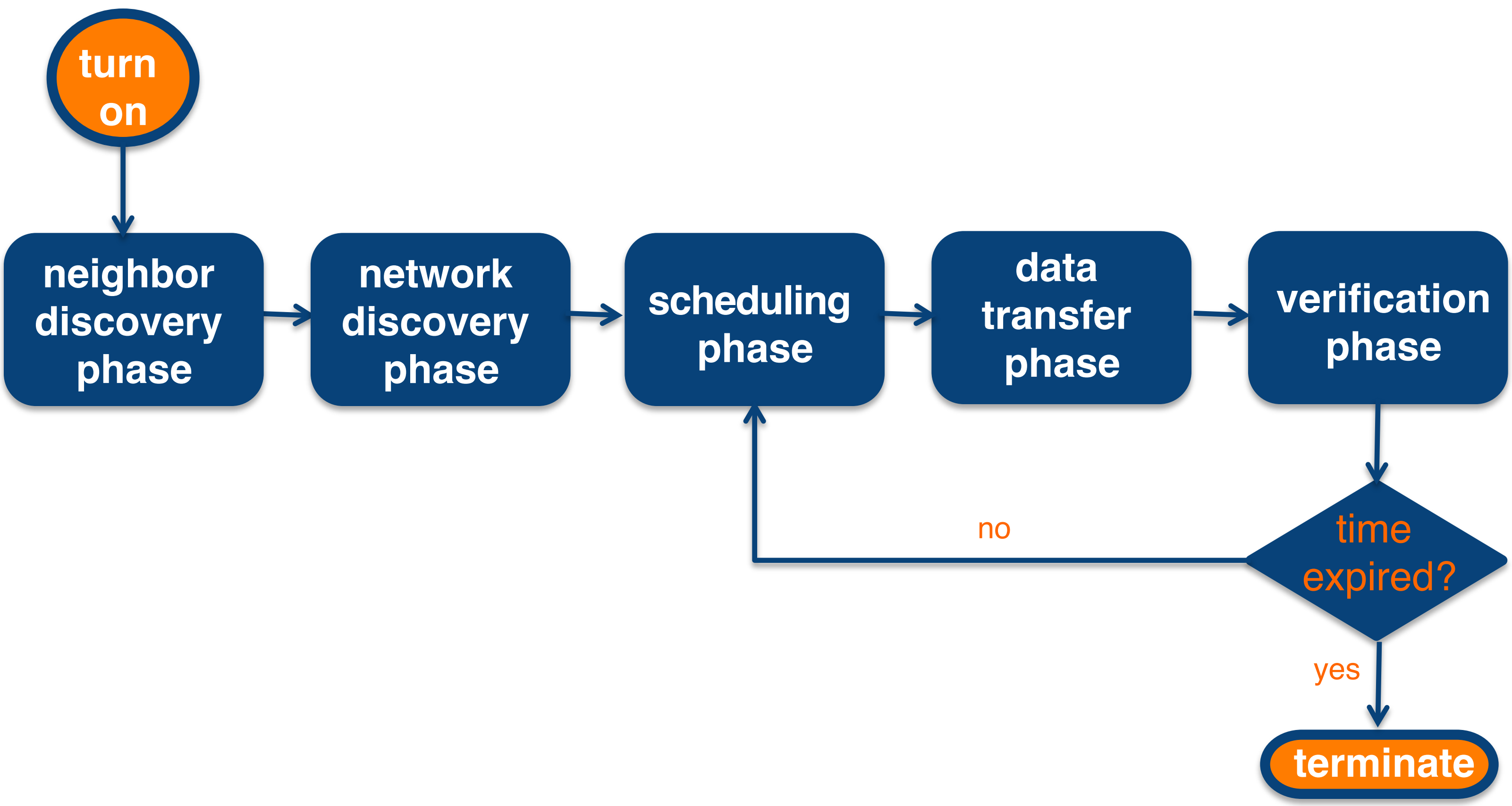
- But $\alpha_F \geq 1$ and $\alpha_B \geq 1$
- So $\alpha_F = \alpha_B = 1$
- Forwarding time is *pure delay*: $F(\tau) = \tau + \beta_F$





Now on to the scheme …

Phases of operation of protocol suite



CC BY NC ND

The challenges – 1

- Nodes need to discover who their neighbors are
  - Require a two-way handshake between the nodes
  - How can we guarantee that any two nodes can communicate packets with each other when other nodes are liable to transmit at the same time and cause collisions?
  - Need an *orthogonal medium access scheme*
  - Must operate with clocks that are not synchronized and tick at different and unknown rates

- Nodes will need to synchronize their clocks with neighbors
  - Need to work with fundamental limitations on clock synchronization
  - Nodes can synchronize their skews but not their offsets which are indistinguishable from delays

The challenges – 2

- Nodes need to form a network
  - Require network wide consistency checks
  - Individual links may look OK, but there could be more complicated hidden inconsistencies
  - Everything has to be done in the presence of malicious nodes while under attack

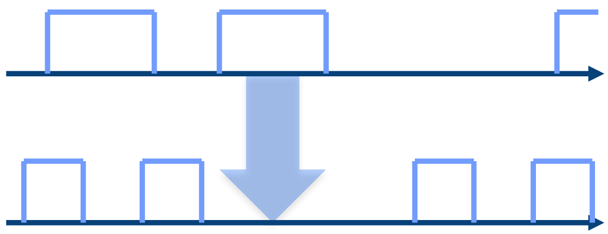
- Nodes draw up a schedule for transmissions and send data
  - Some malicious nodes that conformed hitherto or remained hidden hitherto may not cooperate
  - This requires a check to detect malicious behavior and another round of network wide computation with the un-cooperating nodes being taken into account

The challenges – 3

- Challenge caused by clock wrap-around, which allows "replay attack"
- So above has to be done with a finite bound on clocks
- Also has to be done in the presence of skew errors
- More challenges since we also aim for ε-optimality over network lifetime

Neighbor discovery phase – 1
Orthogonal MAC Code

- Each node attempts to discover its neighbors via two-way handshake
  - Problem of uncoordinated communication
  - Node i has to transmit when node j is listening
  - Clocks have differing skews and times
  - Need a way for every pair of nodes to communicate
  - Orthogonal MAC code



Theorem

There exists an Orthogonal MAC code that allows any pair of neighbors to exchange a message of size W, within a bounded time

Neighbor discovery phase – 2: Clocks

- Each node attempts to discover its neighbors identities and clock parameters
  - Skew can be estimated
  - But not offset

Theorem

There exists a protocol that enables any pair of unsynchronized, half-duplex neighbors to

(i) Determine their relative clock skew to within a desired error

(ii) Bound relative clock offset

(iii) Learn and authenticate each other's identities in a mutually signed link certificate

Network discovery phase – 1: Topological view

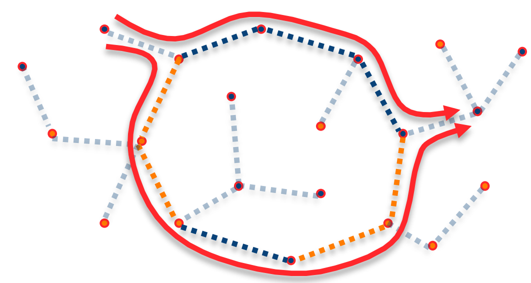
- Each good node attempts to discover topology of the network and relative clock parameters of all the other nodes
  - Views should be common and internally consistent
  - Malicious nodes can lie
  - Each node broadcasts its information about its neighbors
  - Byzantine General's algorithm

Theorem

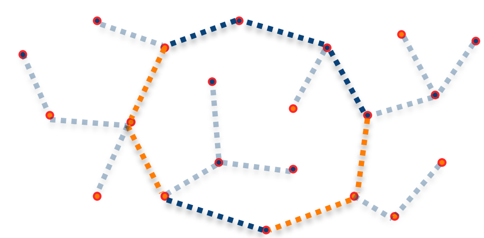
The good nodes will decide on the same topological view after a bounded number of transmissions.

Network discovery phase – 2: Clocks

- Internal information in common views may be *inconsistent*
  - There may be two paths with different clock skew products along paths
  - Impossible to determine which path is correct from declared clock skews alone



- Consistency check protocol
  - Procedure to detect one malicious link
  - There will be an inconsistent cycle, with skew product differing greatly from 1
  - Wait for estimated and actual clock to diverge enough
  - Transmit a packet around cycle that each node must immediately forward



Problem of unsynchronized coordination

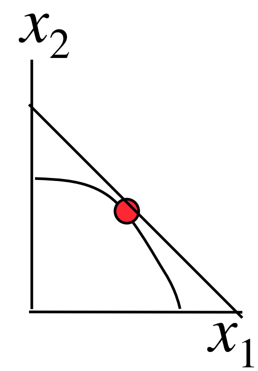
- Problem of coordination is *prior* to synchronization
  - Each stage of Neighbor Discovery phase, Byzantine General's algorithm, and Consistency Check, must be completed simultaneously by network
  - However, clocks have different skews and nodes will proceed through each stage at different speeds
- Solution
  - Assign increasingly larger intervals to each stage so that each node will complete the stage in the same interval regardless of clock skew and offset.

Theorem

There exists a schedule that allows unsynchronized nodes to simultaneously complete a finite number of protocol stages within a bounded time

The Scheduling Phase

- The good nodes determine an optimal schedule
  - Schedule determines optimal end-to-end data rates for each S-D pair
  - Based on time sharing concurrent transmission sets
  - Schedules packets in each concurrent transmission set
  - But estimates of reference clock may diverge because of quantization error in skew estimate – insert “guard bands”

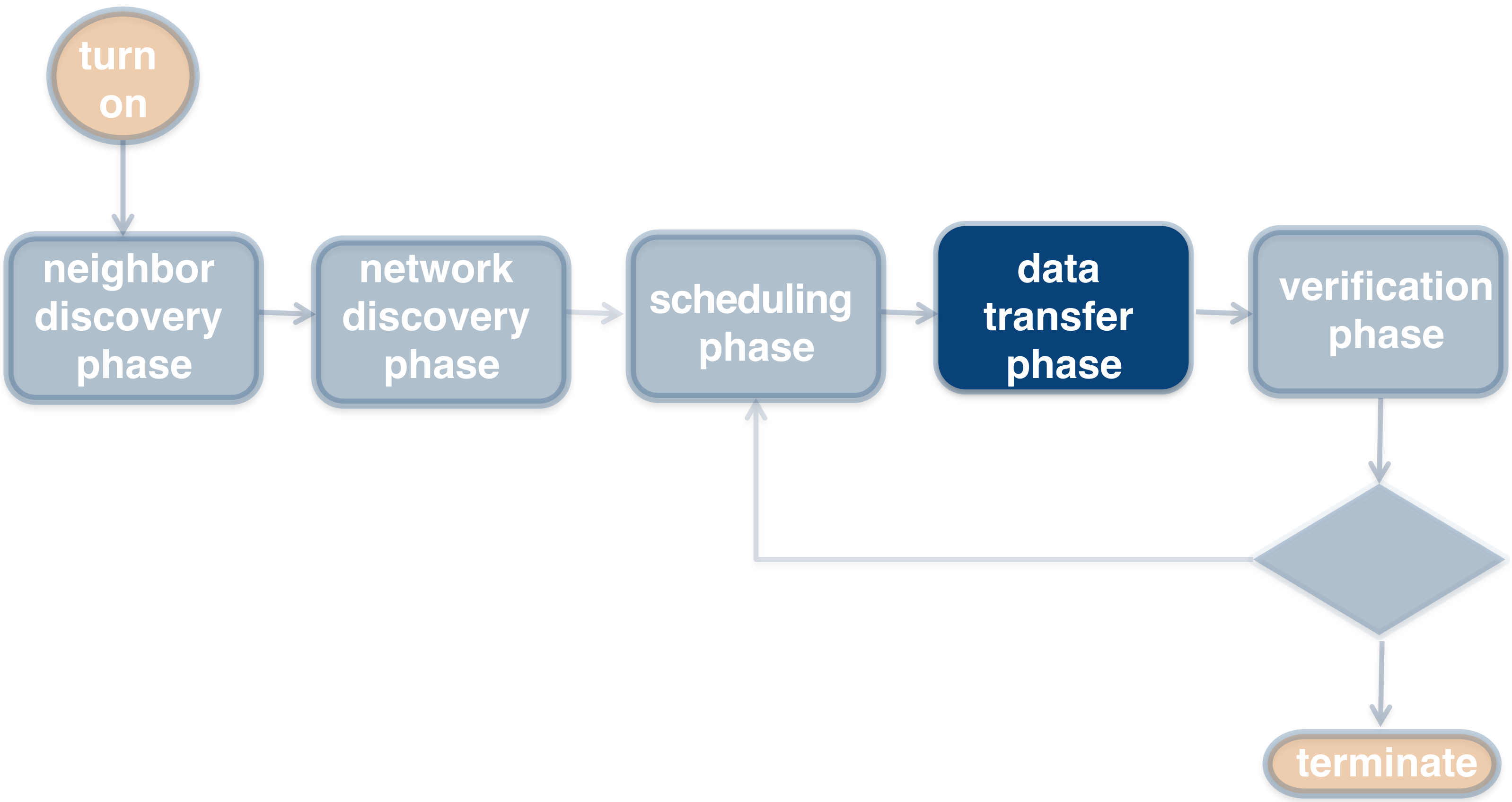


Theorem

There exists a schedule that allows a network of synchronized nodes to maximize its utility over a set of feasible concurrent transmission sets and ensure the rate loss due to clock divergence and overhead is arbitrarily small

July 13, 2010 , P. R. K

The Data Transfer Phase

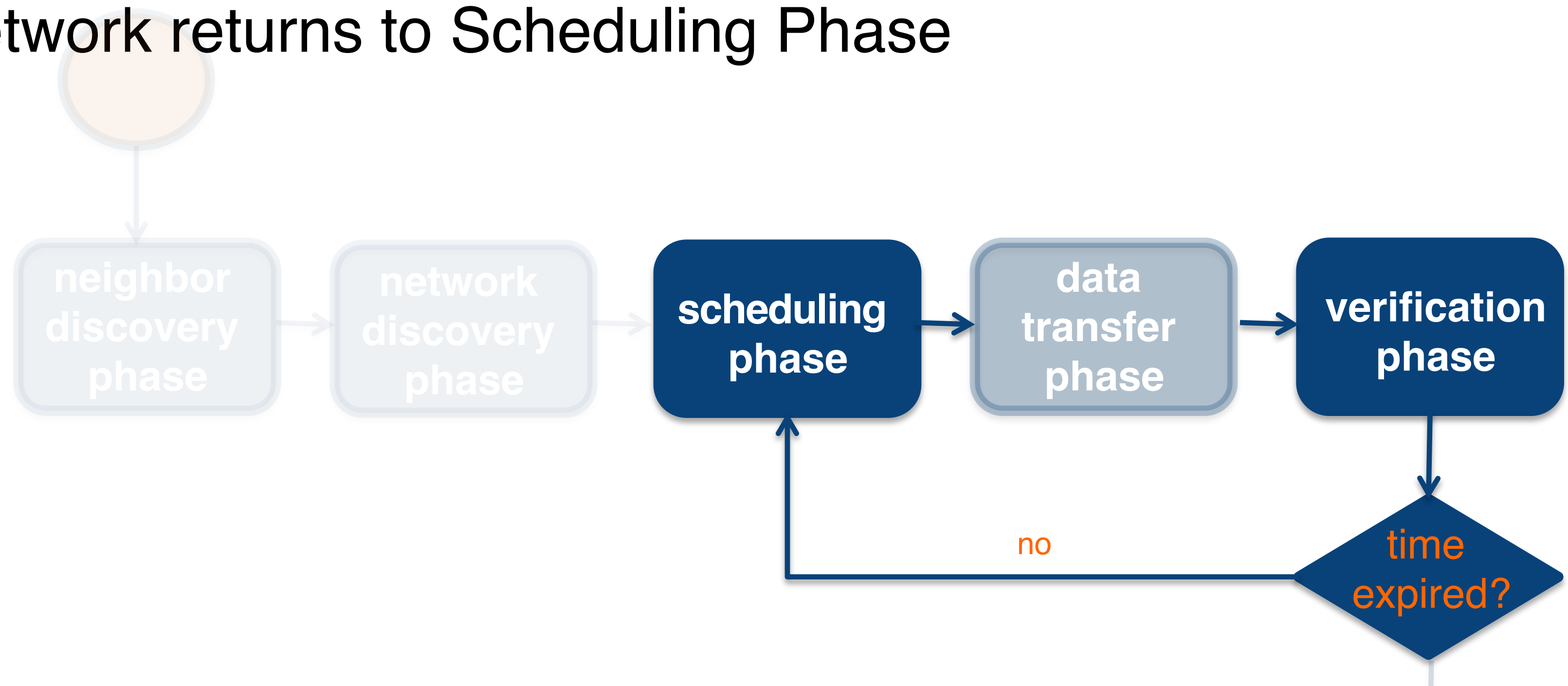


The Data Transfer Phase

- Nodes are expected to conform to the schedule and transmit or relay packets accordingly
- But malicious nodes may disable a concurrent transmission set by not cooperating
- Each node records a “failed” packet and concurrent transmission set
- So we need Verification and iterative pruning

The Verification Phase

- Each node broadcasts the failed concurrent transmission set for the lowest numbered packet that did not arrive
- Byzantine Generals algorithm ensures a common view
- One failed concurrent transmission set is pruned
- Network returns to Scheduling Phase



Min-Max Optimality

- Can bound the loss due to failed concurrent transmission set in data transfer phases, clock skew and divergence errors, and all overheads
- Min-Max Optimality
  - Let Θ_f denote set of disabled concurrent transmission sets
  - Let C denote the set of all concurrent transmission sets

$$\max_{P(C \backslash \Theta_f)} U \geq \min_{\Theta} \max_{P(C \backslash \Theta)} U$$

Theorem

The utility achieved by the protocol over the entire operating lifetime is near min max optimal

CC BY NC ND

Some remarks

- Extensions
  - Nodes not born within bounded time of each other
  - Probabilistic receptions
  - Mobility – gives rise to time-varying system
  - Abstractions/assumptions can be attacked
  - Information theoretic security
- Lots of issues
  - What performance measure?
  - Long transients, overhead in transient period
- Perhaps
  - These results can serve as an “existence proof”
  - Can serve as suggesting an architecture for secure wireless networks
  - Follow up work to mitigate overheads: “Optimization after security”
  - Work may spawn alternative architectures for secure networking

Thank you