Routing in Fat Trees (RIFT) Update draft-rift-rift-03

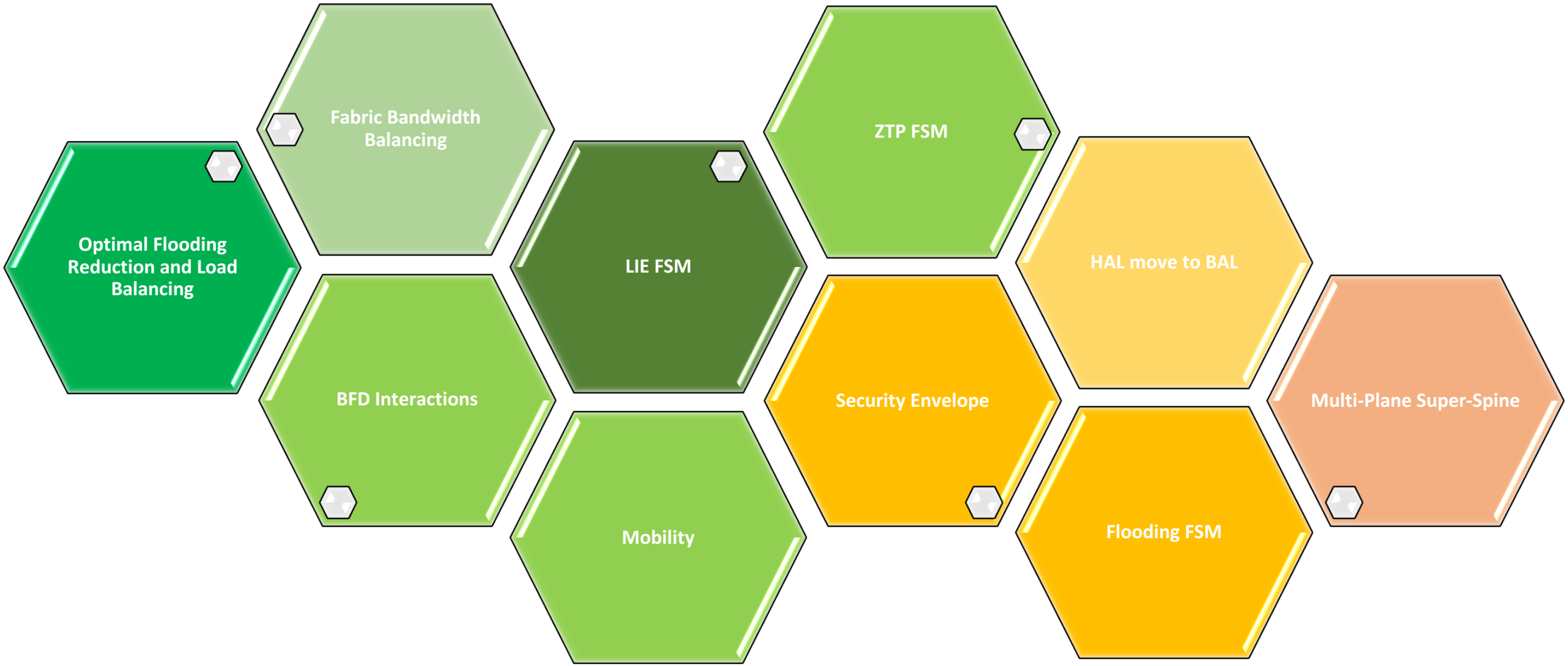
IETF 103, 11/18, Montreal

The RIFT ~~Cabal~~ Authors

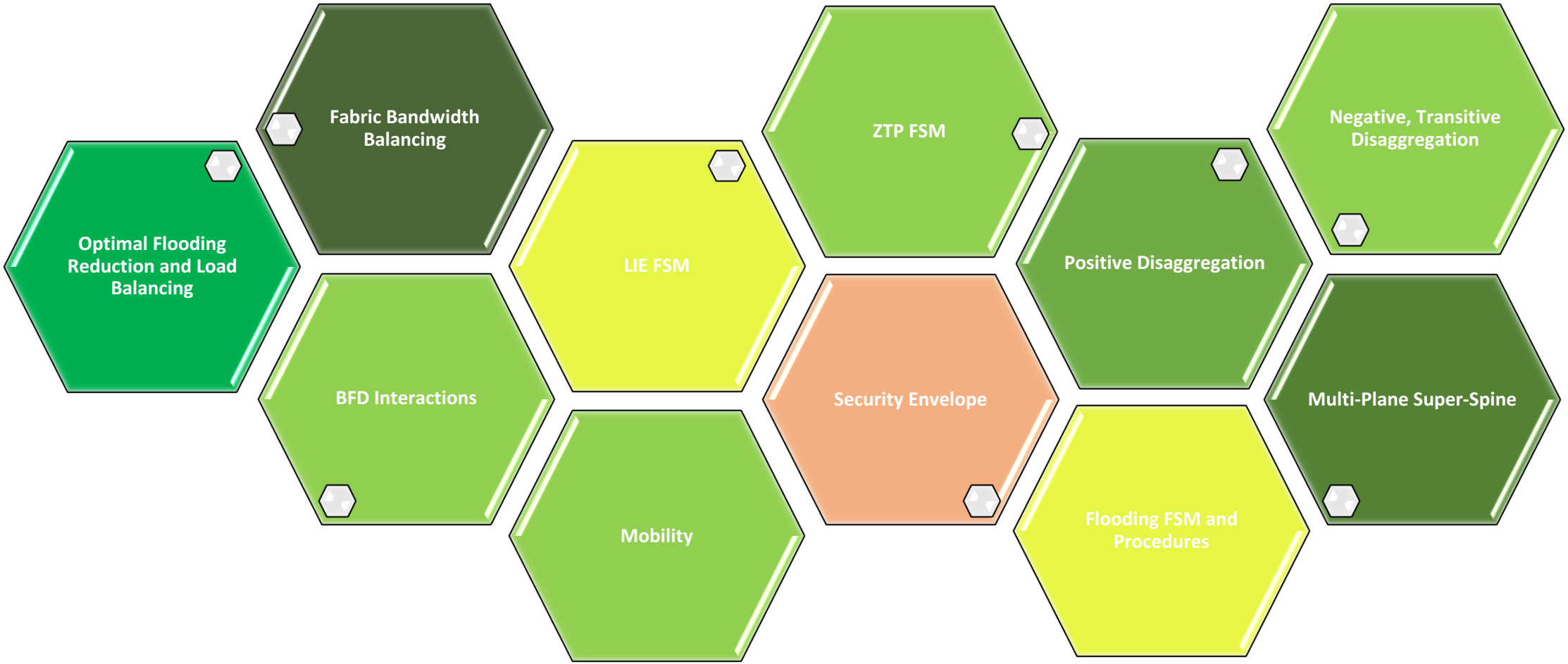
Update from -02

- Last version presented in Canada -02
- We went to -03 since then, -04 already evolving
- Lots of Specification added
- Lost of Open Source Code written and interop'ed
- Once or twice weekly online meetings has been held by the 'core crew'
  - Most meetings recorded and posted to mailing list

Status in -02


Optimal Flooding Reduction and Load Balancing
Fabric Bandwidth Balancing
LIE FSM
ZTP FSM
HAL move to BAL
BFD Interactions
Security Envelope
Multi-Plane Super-Spine
Mobility
Flooding FSM

Update -03/-04, Green is Done



Rough Statistics

- Emails on “core contributor” email threads since last IETF: 300+
- Commits on Open Source version since last IETF without branch merges: 205
- Lines on Open Source version patch since last IETF: 15’897
- Diff Size Between -02 and -03 specification: 6’574 lines of text
  - Flooding procedures
  - Multi-plane fabrics
  - Tons small fry since running code interop is the best teacher
- Objects on encoding model changed: 7
- Ideas Discussed and Scrapped: Dozens ;-)

What did we remove first ;-) ?

- We need to keep the base spec a base spec and basic demands drives the basic content
- PGP goes into separate draft
- SR goes into separate draft
- Key-Value Store will get its own draft
  - A well-known key registry likely

What did we do then 1st ?

- We could not resist changing language since it got confused once we started work on multiple planes on top of fabric
  - ToF: Top of Fabric
  - Spine: Anything between leaf and ToF
  - ToP: Top of Pod
  - Radix South/North: # of ports

What did we do 2nd ?

- Significant work on flooding based on clean room open source implementation and the first fallout
- Updated Flooding Scope Table
  - Driven mostly by Bruno's clarifying question (albeit he implemented correctly from old table)
  - ToF changed E-W flooding scopes

Type / Direction	South	North	East-West
node S-TIE	flood if level of originator is equal to this node	flood if level of originator is higher than this node	flood only if this node is not ToF
non-node S-TIE	flood self-originated only	flood only if neighbor is originator of TIE	flood only if self-originated and this node is not ToF
all N-TIEs	never flood	flood always	flood only if this node is ToF
TIDE	include at least all non-self originated N-TIE headers and self-originated S-TIE headers and node S-TIEs of nodes at same level	include at least all node S-TIEs and all S-TIEs originated by peer and all N-TIEs	if this node is ToF then include all N-TIEs, otherwise only self-originated TIEs
TIRE as Request	request all N-TIEs and all peer's self-originated TIEs and all node S-TIEs	request all S-TIEs	if this node is ToF then apply North scope rules, otherwise South scope rules
TIRE as Ack	Ack all received TIEs	Ack all received TIEs	Ack all received TIEs

Table 3: Flooding Scopes

What did we do 3rd bis?

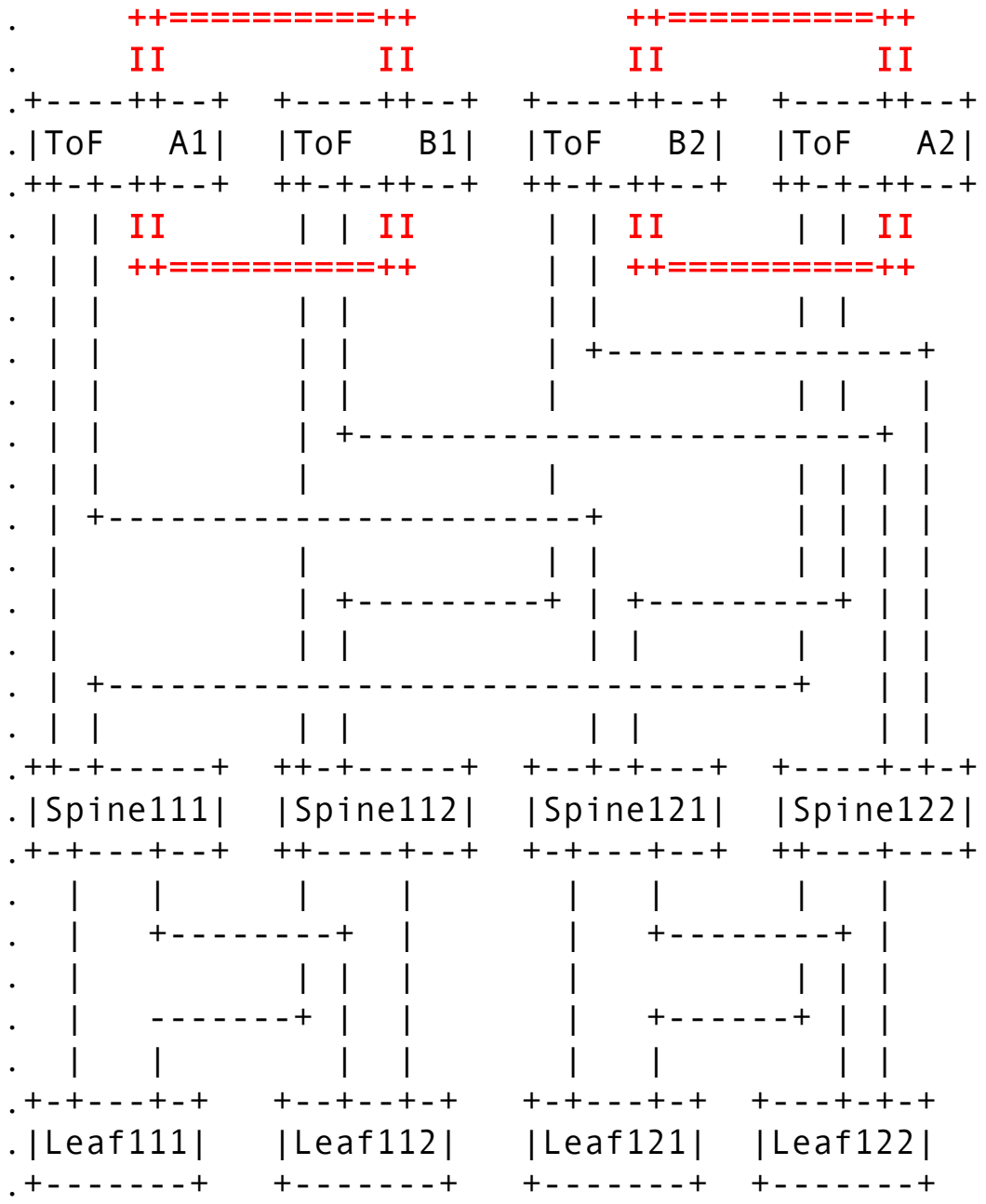
- Wrote all the flooding rules in Appendix B.3
- Flood Structure per Adjacency
  - TIES_TX, TIES_RTX, TIES_REQ, TIES_ACK Queues of TIE Headers conceptually
- TIDE
  - Generation: Generate periodically the set of TIDE describing the database
    - MIN_TIEID and MAX_TIEID were not specified precisely enough
    - Included LifeTime wasn't specified tight enough
    - All has been derived from the fact that we slavishly follow ISIS spec
    - Bunch of ideas along the lines of "let's not sort headers" died in the fry
  - Processing: Based on neighbor's description manipulate the queues
    - Major bug by omission has been found (we didn't put all the "holes" in the middle of the TIDE onto the queues in original text)
    - Very delicate bug with >= vs > on a step has been found

What did we do 3rd bis bis ?

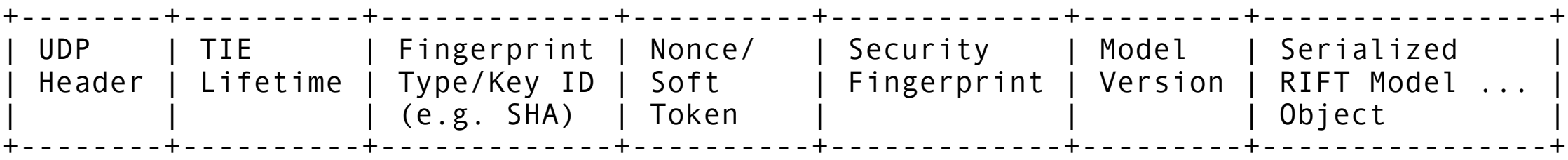
- TIRE
  - Generation: On a regular basis gather TIES_REQ and TIES_ACK queues and advertise
  - Processing: not much different from a single entry in TIDE processing
  - No issues found AFAIR
- TIE Processing
  - Based on TIE Header comparisons accept and ack, regenerate own or queue a new one to transmit

What did we do then 3rd ?

- Multi-plane Fabrics and Negative Disaggregation
- Pascal will spend good amount of time on that
- I can't resist a retro-chic typewriter produced picture though



Secure, Optimized RIFT Information Element Envelope Running Strawman



- Avoids Problems we found over years with traditional link-state protocols when securing them
- Maximizes Flooding Speed (No Re-Serialization, No Lifetime protection)
- Security Fingerprint Does Not Get Affected by TIE LifeTime Changes
  - Security can be solved by forcing advertisement of origin timestamp and clock on fabric
- Serialized Object Keeps Its Fingerprint and Does Not Need Re-Serialization on LifeTime Field Change by Every Node
- Lie Nonces Are Protected by Fingerprint Against Replays, Reflect Neighbors' Nonce. The nonce can be used as Salt to generate softtokens
- Only Node with Private Key (or Shared Secret) Can Generate the Fingerprint (Either for LIEs One-Hop or for TIEs Providing Origin Validation and Integrity)

So still to do as hanging comments

- Explain which parts of specification need to be implemented for leaf/spine/superspine/ToF version in detail
- Write a section on E-W superspine/ToF flooding scope to connect partitions so it becomes clearer
- Get security envelope done, move remaining lifetime out the TIE packet so it can be modified independently of the SHA'd TIE
  - Possibly go to soft token generation to avert the necessity to SHA the nonce on the TIE envelope
- Add an intermediate state on multiple neighbors
- Modify flooding procedure on TIDE reception with the case of stale north TIEs stuck more than one level up (propagate header description southbound)
- Write section on negative disaggregation example
- Move adjacency formation rules onto FSM text and remove 2.4.2</**t**>

THANK YOU FOR YOUR ATTENTION