

---



# Network Protocol Specification Languages & Compilers

Jasson Casey

[jasson.casey@gmail.com](mailto:jasson.casey@gmail.com)

Dr. Alex Sprintson

[spalex@tamu.edu](mailto:spalex@tamu.edu)

---

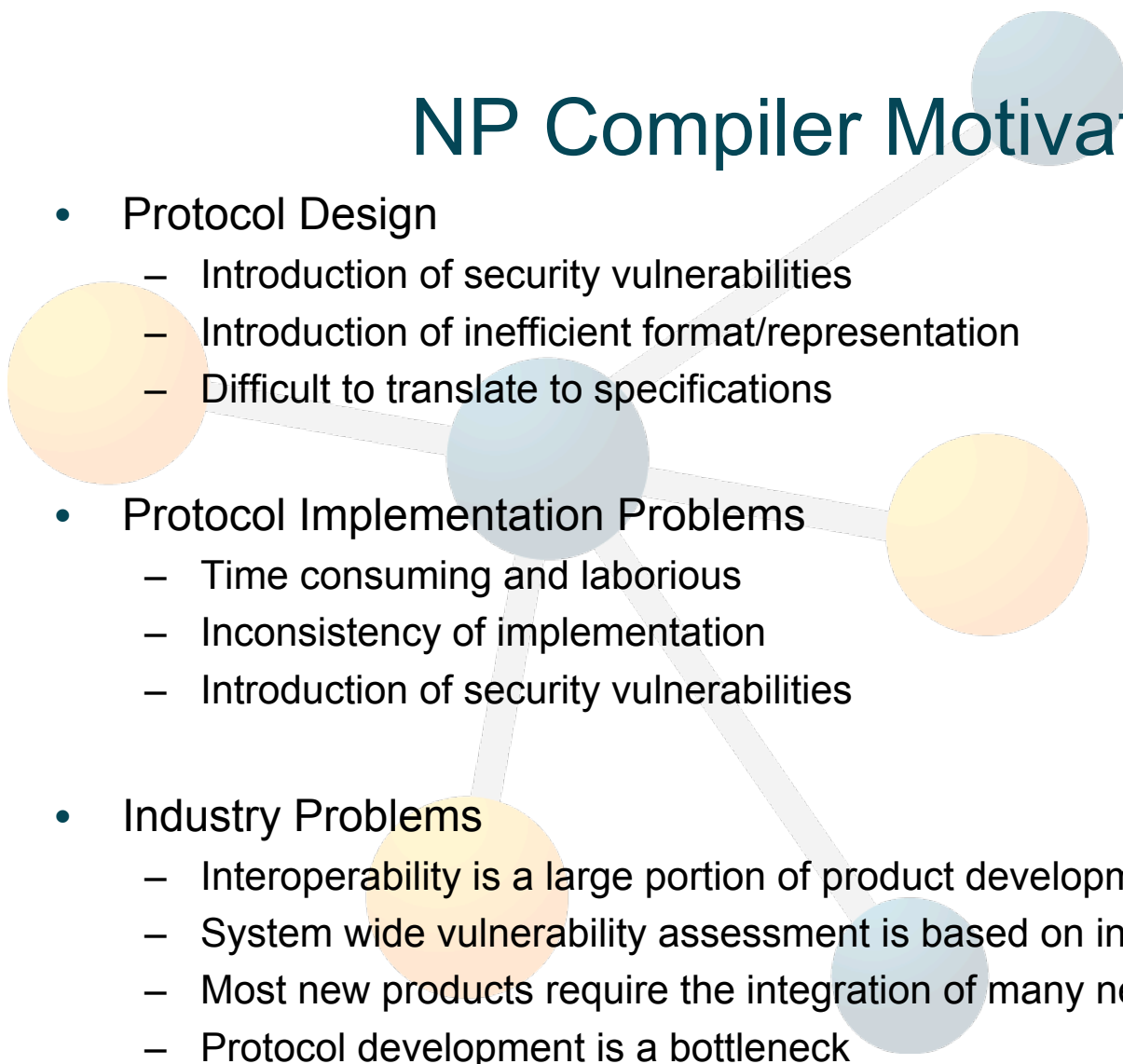
# Background



- Coming from Industry
    - Equipment design and development
    - Service architecture design and development
  - Research Interests
    - Confluence of networking and programming languages
    - Static analysis & abstract interpretation of protocols specs
    - Optimization and target generation
      - Protocol implementations
      - Abstract configuration
-

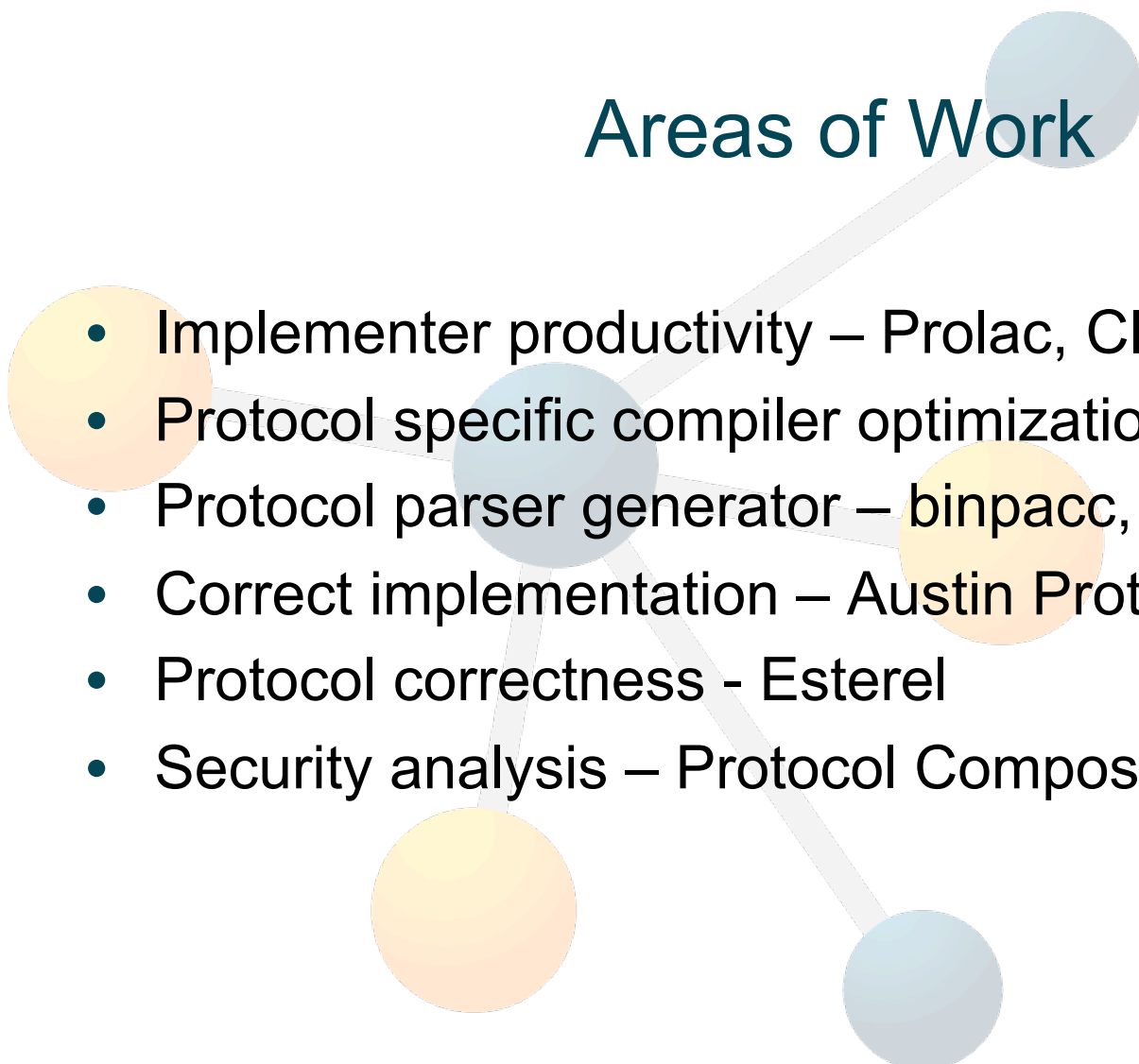
---

# NP Compiler Motivation

- 
- Protocol Design
    - Introduction of security vulnerabilities
    - Introduction of inefficient format/representation
    - Difficult to translate to specifications
  - Protocol Implementation Problems
    - Time consuming and laborious
    - Inconsistency of implementation
    - Introduction of security vulnerabilities
  - Industry Problems
    - Interoperability is a large portion of product development
    - System wide vulnerability assessment is based on incidents
    - Most new products require the integration of many new protocols
    - Protocol development is a bottleneck
-

---

## Areas of Work

- 
- Implementer productivity – Prolac, Click
  - Protocol specific compiler optimizations
  - Protocol parser generator – binpacc, packet-types
  - Correct implementation – Austin Protocol Compiler
  - Protocol correctness - Esterel
  - Security analysis – Protocol Composition Logic

---

# Complete Tools

- Prolac
  - Language
    - Functional
    - Simple syntax can cover common network idioms
  - Compiler
    - Removal of dynamic dispatch
    - In-lining of common functions
    - Outlining unlikely error handling
  - Obstacles
    - Actual use requires extensive native language interaction
    - No primitives: encoding, state machine, events, transitions, etc
  - A Readable TCP in the Prolac Protocol Language, E. Kholer, M. Kaashoek, D. Montgomery, ACM SigComm 99

---

# Prolac Example

*// Example 2*

```
module Segment-Arrives has Tcb {  
  field tcb :> *Tcb;  
  check-segment ::=  
    (listen ==> do-listen)  
    || (syn-sent ==> do-syn-sent)  
    ...;  
  listen ::= tcb->listen;  
  syn-sent ::= tcb->syn-sent;  
  ...  
} hide (listen, syn-sent, ...);
```

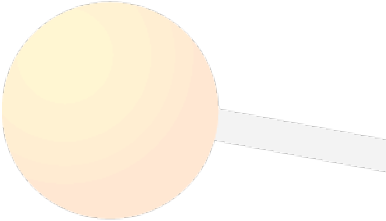
---

# Complete Tools

- Austin Protocol Compiler, Tommy McGuire, Springer
  - Language
    - Functional
    - Encoding primitives for TLV style protocols
    - System primitive support: timers, and UDP IO
  - Compiler
    - Guarantee from abstract to concrete model
  - Obstacles
    - Encoding lacks support for nested objects and lists
    - Encoding lacks primitives for ASCII encoded protocols
    - No support for stream IO
    - Little optimization
  - T. McGuire, M. Gouda, The Austin Protocol Compiler, Springer 2004

---

# APC Example



```
process aserv
var c : address;
    fnd, invld : integer
begin
    rcv query from c →
        resp.id := query.id;      resp.opcode := query.opcode;
        resp.rd := query.rd;      resp.ra := 0;
        resp.aa := 1;
        resp.qdcount := query.qdcount;
        invld := parse_query(query.qdcount, query.body);
        if invld → resp.rcode := response_code();
            resp.ancount := query.ancount;
            resp.nscount := query.nscount;
            resp.arcount := query.arcount;
            resp.tc := query.tc;
            resp.body := query.body;
            resp.size := query.size
        | ¬invld → ...
            resp.tc := response_oversize();
            resp.body := response_body();
            resp.size := response_size() + 12
        fi;
        send resp to c
    end
```

---



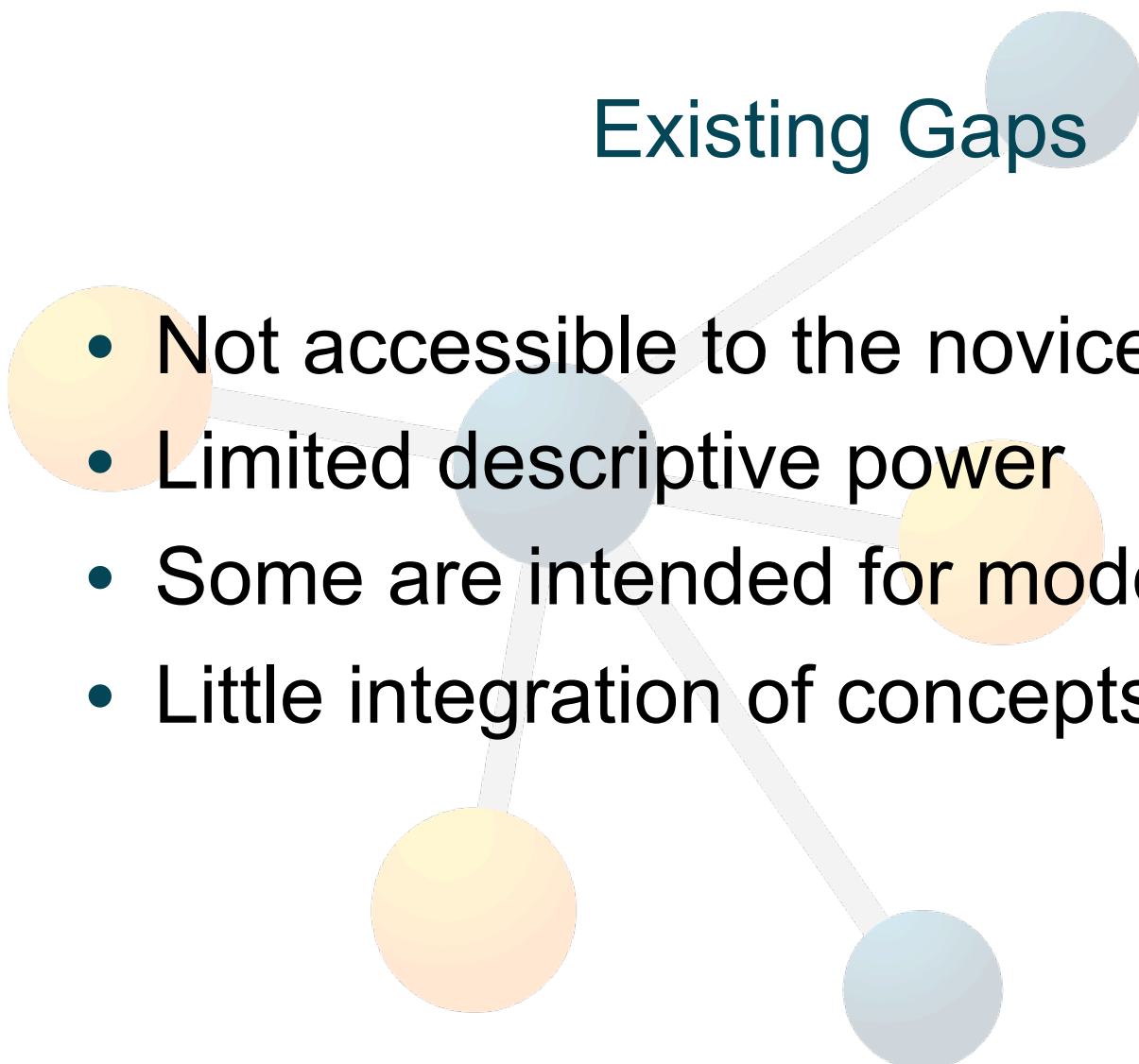
---

## Previous Industry Work

```
# Standard L2TP AVP header for all control message types
pdu L2TP_AVP_Header:
  bool mandatory      = false
  bool hidden         = false
  pad(.4)
  # The base length is always just the header fields
  number(.10) length  = 6
  number(2) vendor_id = 0
  number(2) type      = 0
  # Before writing the packet inc the length and set the type
  encode(length += container.payload.bytesize)
  encode(type   = container.payload.enum)
end
```

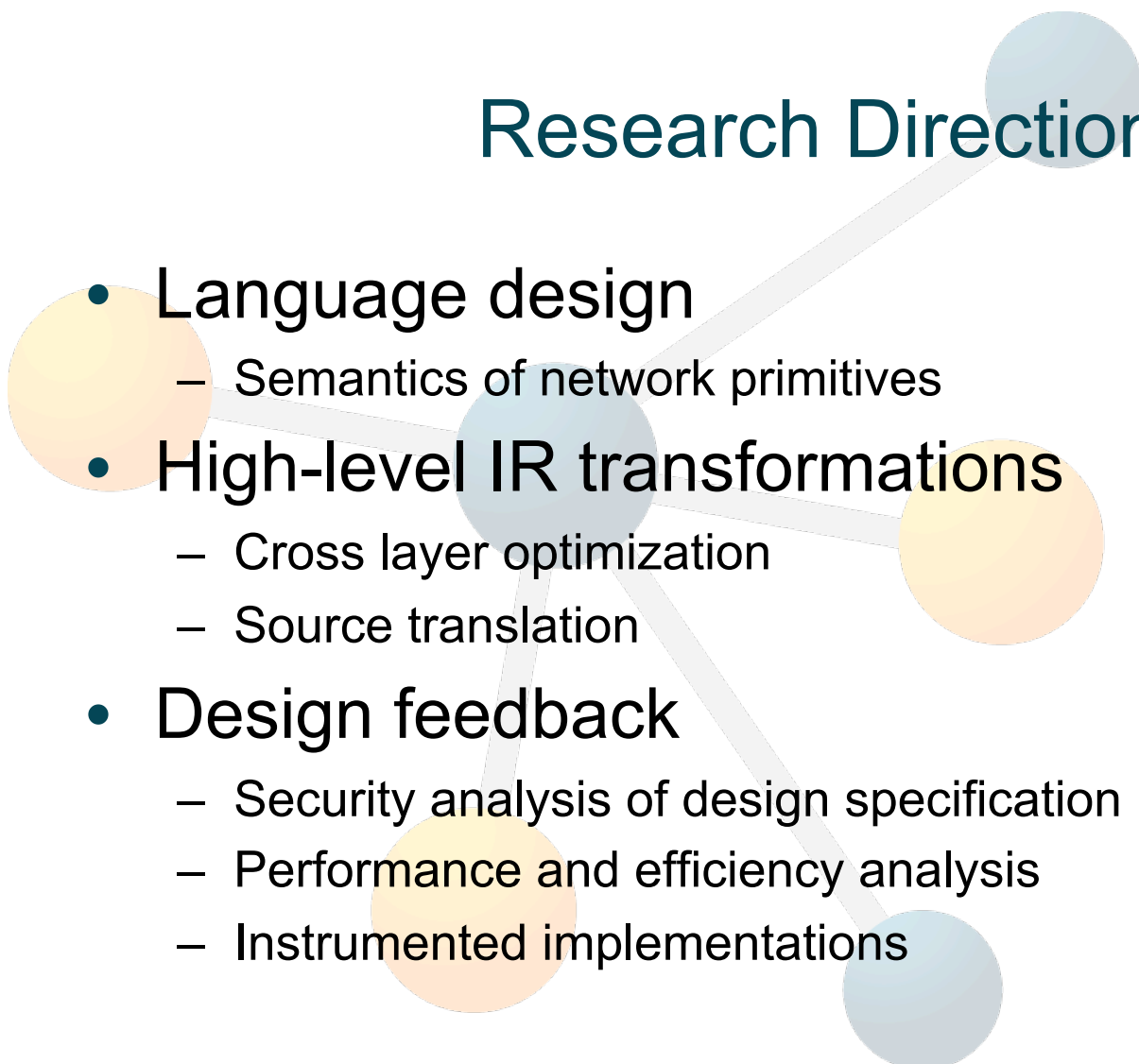
---

## Existing Gaps

- 
- Not accessible to the novice
  - Limited descriptive power
  - Some are intended for modeling only
  - Little integration of concepts

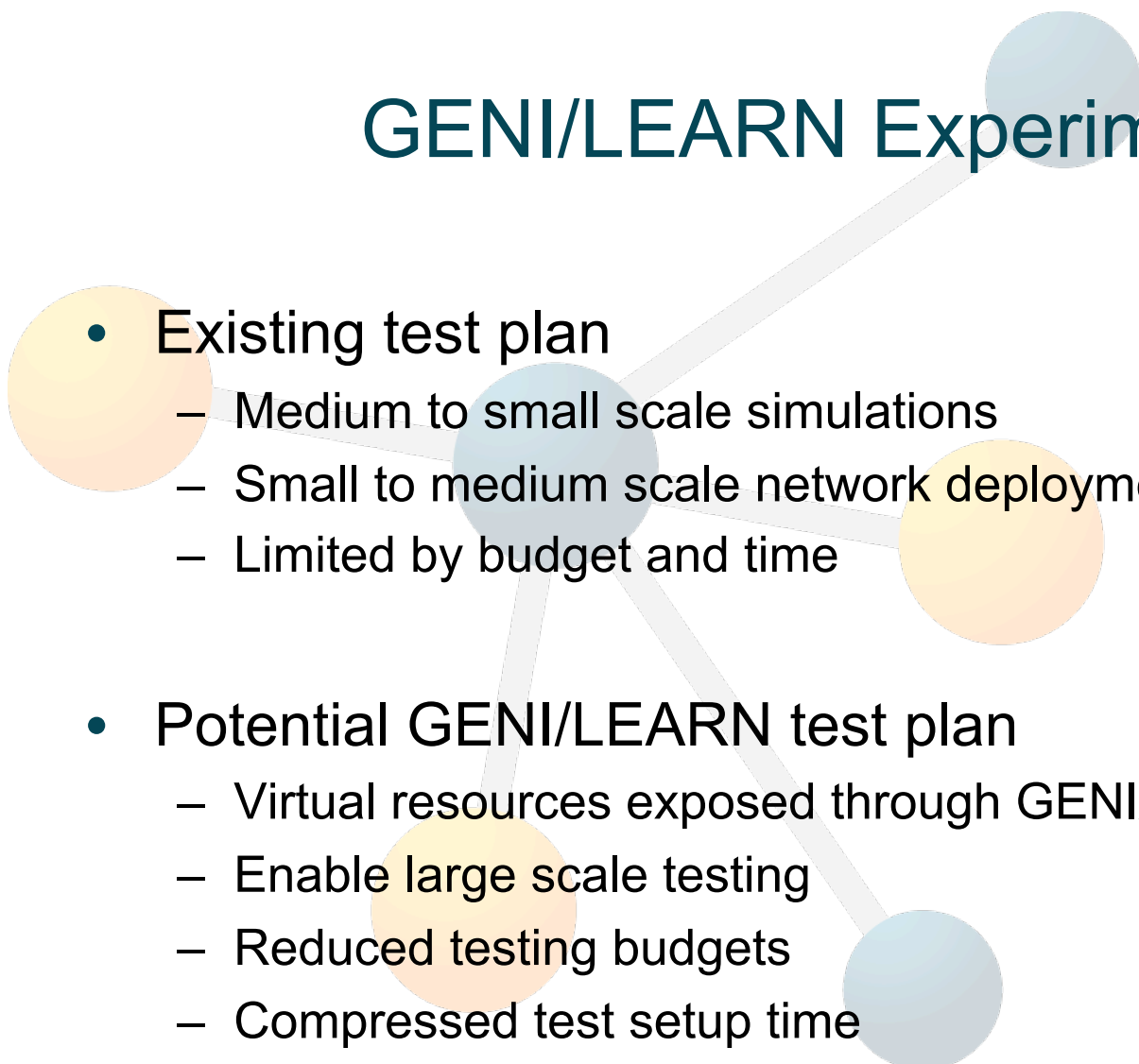
---

# Research Directions

- 
- Language design
    - Semantics of network primitives
  - High-level IR transformations
    - Cross layer optimization
    - Source translation
  - Design feedback
    - Security analysis of design specification
    - Performance and efficiency analysis
    - Instrumented implementations
-

---

# GENI/LEARN Experiments

- 
- Existing test plan
    - Medium to small scale simulations
    - Small to medium scale network deployments
    - Limited by budget and time
  - Potential GENI/LEARN test plan
    - Virtual resources exposed through GENI/LEARN
    - Enable large scale testing
    - Reduced testing budgets
    - Compressed test setup time

---

# Conclusion

- Goals

- Approachable by novice
- Can specify existing common IETF RFC(s)
- Optimize for target system (byte alignment, cache compaction, etc)

- Successful if ...

- Does not compromise existing level of security and performance
- Rapid prototype new and existing protocols