

SDPng Update

draft-ietf-mmusic-sdpng-04.txt

Dirk Kutscher dku@tzi.org
Jörg Ott jo@tzi.org
Carsten Bormann cabo@tzi.org

Overview

- **Changes in -04**
- **Open Issues**
- **Next steps**

Changes in -04

- **Generic syntax mechanisms for re-using definitions**
 - **sdpng:use** for referencing named definitions
 - **sdpng:group** to name a group of definitions for later referencing
 - **sdpng:prop** to add properties to definitions
- **Capability negotiation**

Capability Negotiation

- **Offer/Answer:**
 - **Select one (or more) configuration alternatives from the set of offered configurations**
 - **Typically, the answerer matches offer against list of supported configurations**
 - **Problem:**
 - **Some applications (e.g., video) can require many configuration parameters**
 - **Representing each variant as an individual alternative not practical**

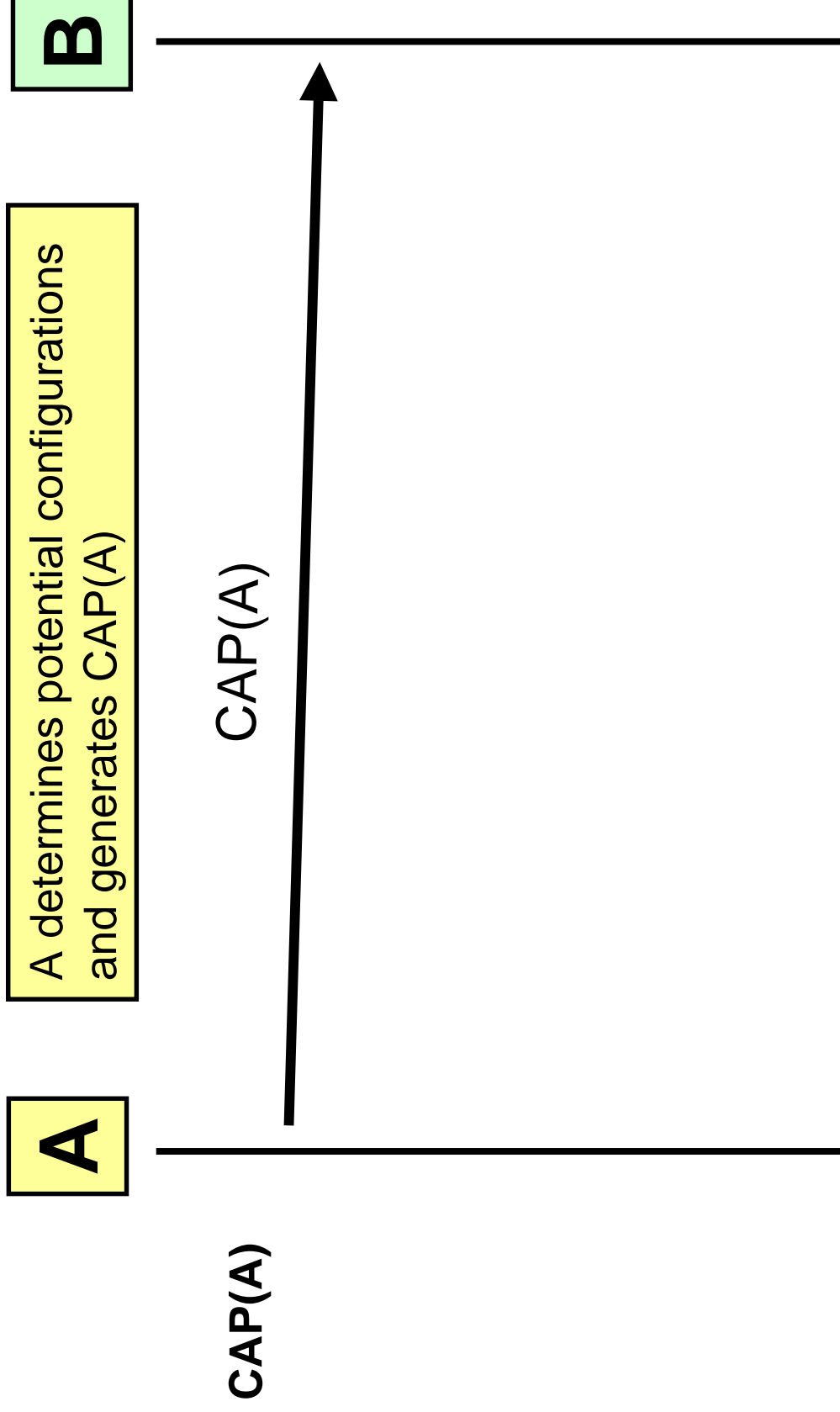
Capability Negotiation in SDPng

- Language mechanisms to describe supported value ranges for configuration parameters
 - “For codec X, I can support frame rates 6,12,24 and codec options foo and bar”
- Algorithms to compare and collapse two descriptions
 - Result: A commonly supported configuration (if it exists)

Some Assumptions

- **This should be simple**
 - Limited set of data types and collapsing rules
- **Usable for SIP, RTSP, SAP, MEGACO**
 - Message size, implementation complexity
 - SIP: SDPng exchange in one offer/answer cycle
- **Extensible**
 - No fixed identifiers
 - Collapsing is possible without knowing semantics of configuration descriptions
 - Can collapse without having to know schema definition etc.

Conceptual Overview



A

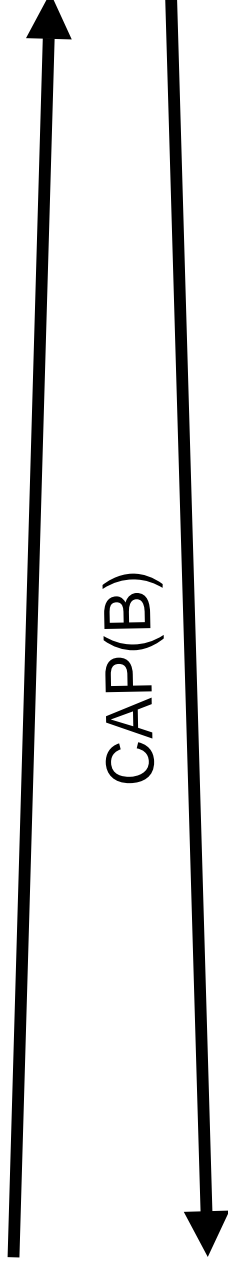
B generates CAP(B), according to the set of components from A's description that B wants to support.

B



CAP(A)

CAP(A)



CAP(B)

CAP(B)

A

CAP(A)

CAP(AB)

Both A and B can compute CAP(AB),
the list of commonly supported
potential configurations.

B



CAP(A)

CAP(B)



CAP(B)

CAP(AB)

A

A selects the actual configuration
CFG(AB).

B



CAP(A)

CAP(AB)
CFG(AB)

CAP(A)



CAP(B)



CAP(B)
CAP(AB)

A

A augments CFG(AB) with A's non-negotiable transport parameters and generates CFG_T(A).

B



CAP(A)

CAP(AB)

CFG(AB)

CFG_T(A)

CAP(A)

CAP(B)

CFG_T(A)

CAP(B)

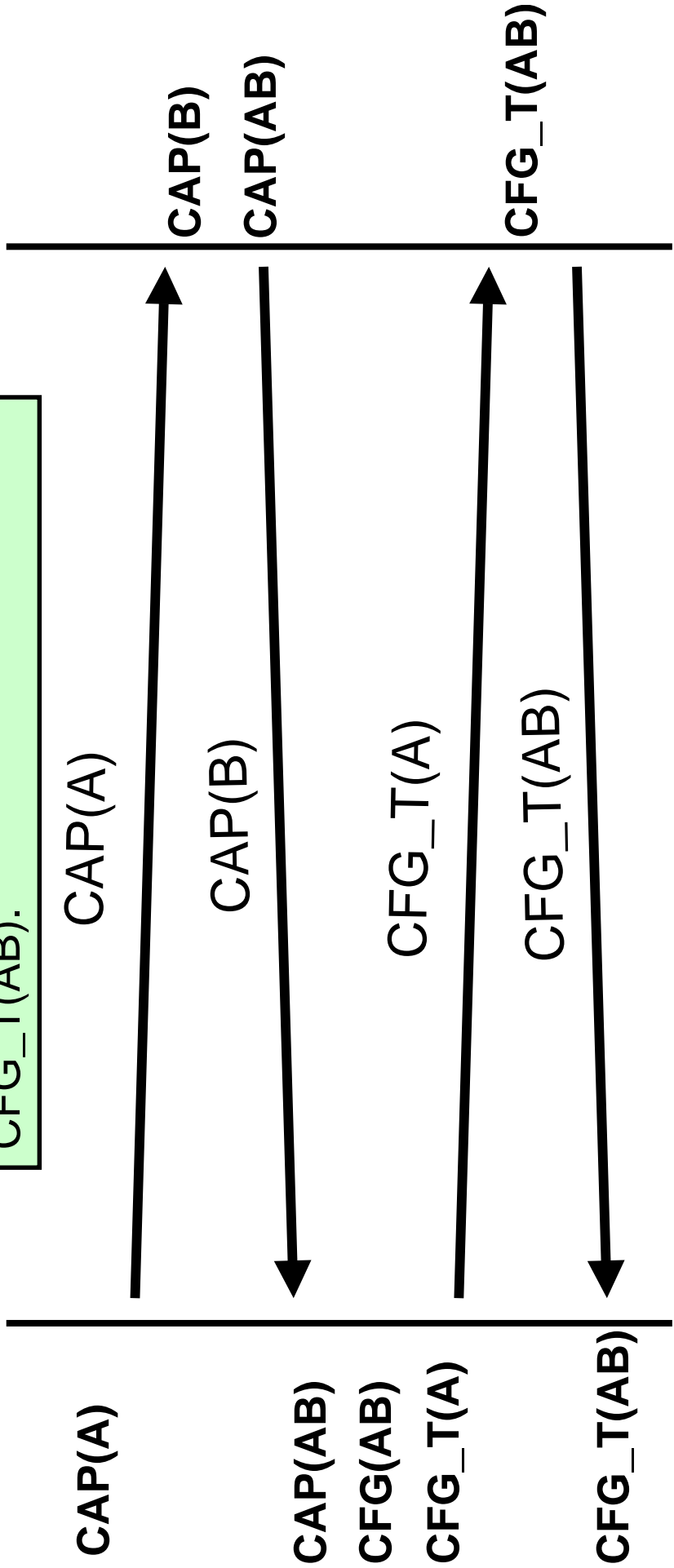
CAP(AB)



A

B receives CFG_T(A) and adds its own transport parameters, resulting in CFG_T(AB).

B



CAP(A)

CAP(AB)

CFG(AB)

CFG_T(A)

CFG_T(AB)

CAP(A)

CAP(B)

CFG_T(A)

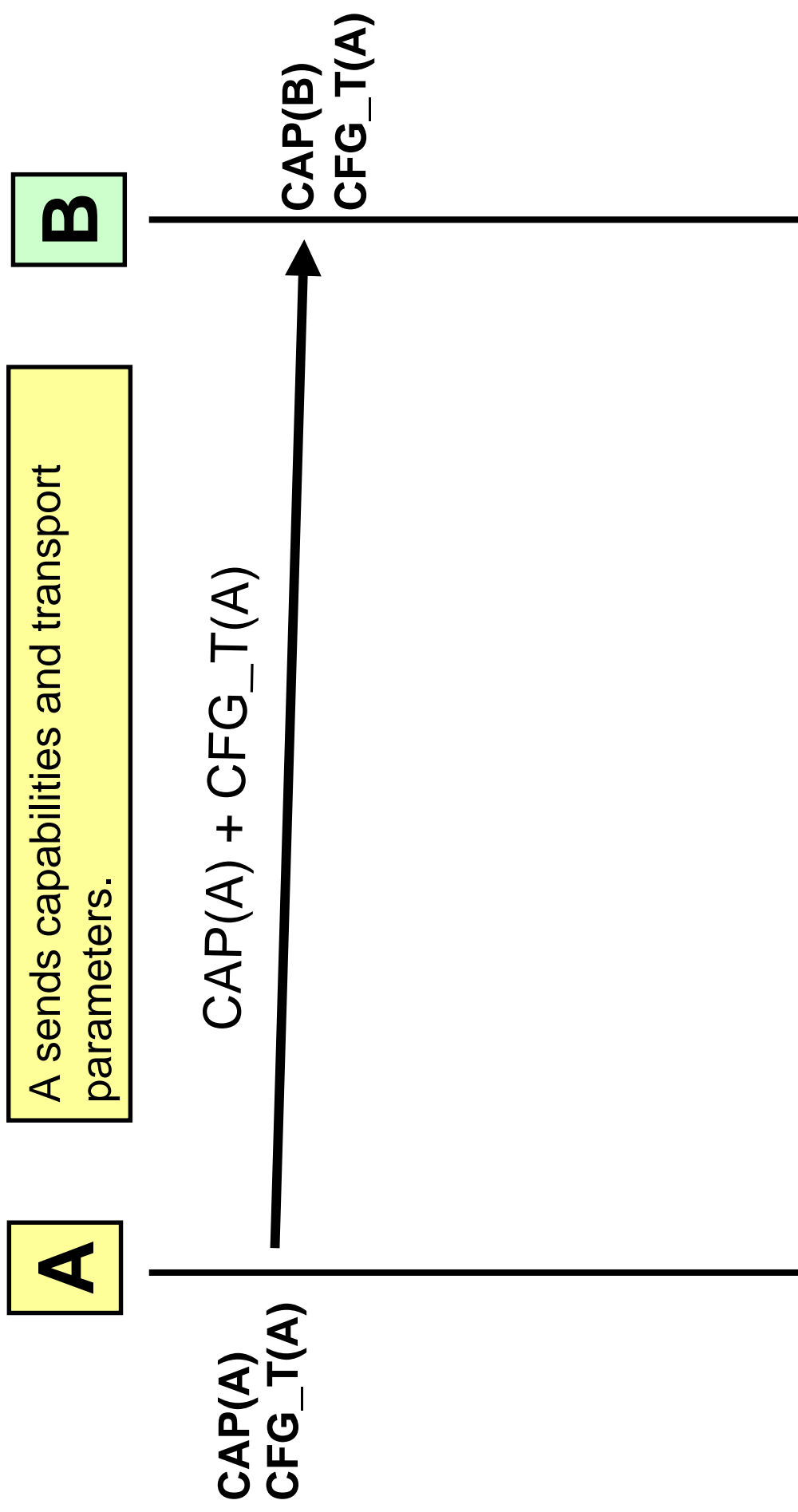
CFG_T(AB)

CAP(B)

CAP(AB)

CFG_T(AB)

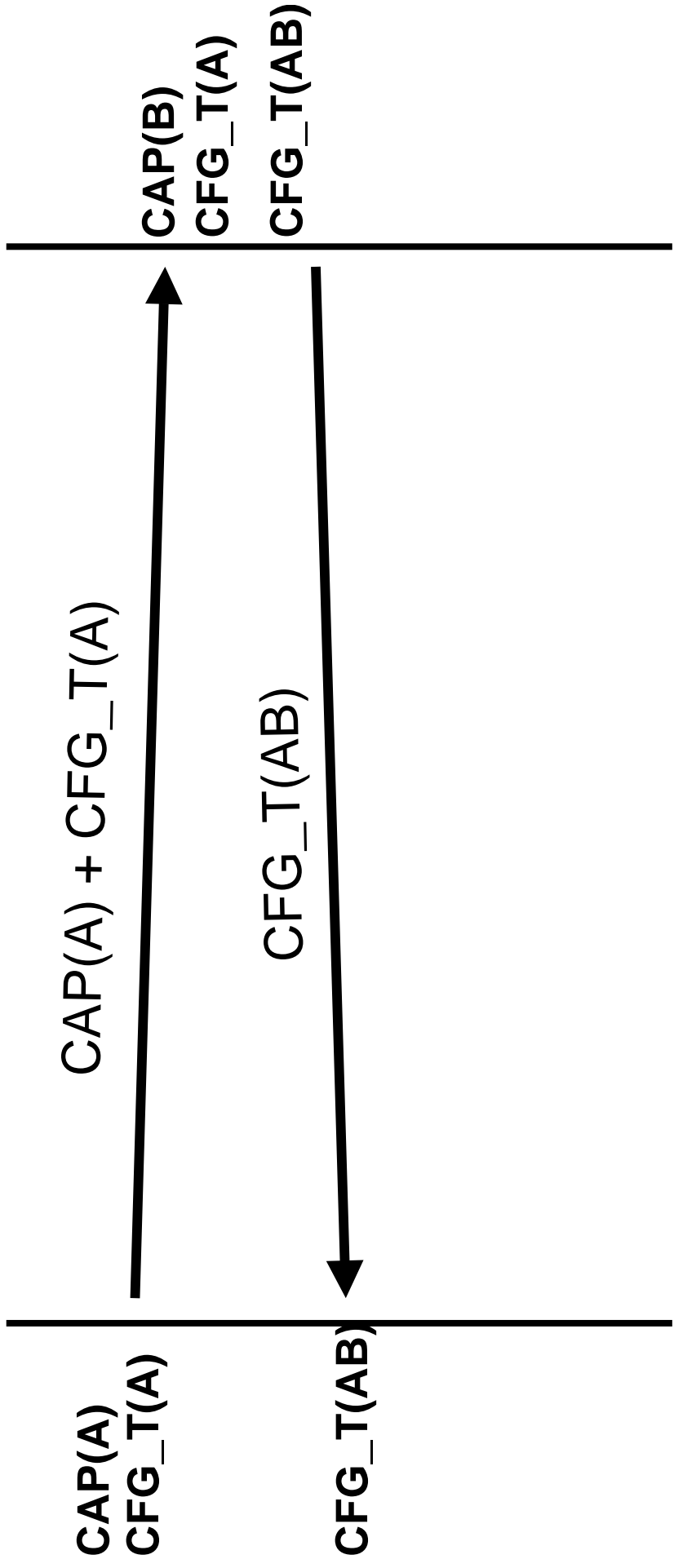
Optimization



A

B computes CAP(AB) and generates CFG_T(AB).

B



Properties of the Negotiation Procedure

- **Integration of transport parameters in “offer”**
 - **Concept of non-negotiable parameters**
- **“Context-free” processing**
 - **Process capability descriptions without knowing schema definition and semantics**
 - **Encode type info into parameters, infer collapsing rule**

Collapsing Example

A

```
<x a=" [BAR, 3, 5, 8]" b="(2,8)" c="foo">  
<y b=" [UVW, XYZ]" />  
</x>
```

B

```
<x a=" [3, 6, 8]" b="(5,10)" c="bar">  
<y b=" [RST, XYZ]" />  
</x>
```

Result

```
<x a=" [3, 8]" b="(5,8)">  
<y b=" [XYZ]" />  
</x>
```

Data Types

- **Set of symbols**
 - [FOO, 4, 5] ☒ [BAR, 3, 4] → [4]
 - Collapsing rule: intersection
- **Numerical ranges**
 - (3, 10) ☒ (5, 12) → (5, 10)
 - Collapsing rule: determine common range
- **Optional string parameters**
 - "FOO" ☒ "BAR" → ""
 - Collapsing rule: test for equality

Open Issue #1: Global View

- **With SIP/SDP, we send participant-specific descriptions**
 - **Can lead to ambiguity (e.g., asymmetric configurations)**
- **Global view:**
 - **Disambiguate all descriptions**
 - **Relate to participant**
 - **Participant-identifier?**
 - **Useful for multi-party scenarios**
 - **Adds some complexity**

Open Issue #2: External Definitions

- **External libraries**
 - **How useful are they in real life?**
 - **Resolving references to libraries**
at call-setup time...
 - **Always include all definitions?**

Open Issue #3: Capability Negotiation

- **What functionality is really needed?**
 - **Current draft provides a simple, limited solution**
 - Limited set of data types
 - Optimized, application specific processing rules
- **Other, more generic work exists**
 - **CC/PP, CONNEG etc.**
 - **How much re-use is useful?**
 - **Technical discussion required!**

Open Issue #4: Capability Negotiation

- **Semantics of descriptions and negotiation results**
 - **Multiple alternatives for a component**
 - **Select one? Support any?**
 - **Ranges for parameters in final descriptions**
 - **Are senders free to select values?**
 - **Do changes require signaling?**
- **Non-negotiable parameters**
 - **Currently under-specified...**

Open Issue #5: Internal References

- SPDng provides mechanisms to name definitions and re-use them
 - Compact definitions, grouping etc.
- For capability negotiation, all references **MUST** be resolved
 - Collapsing process yields a new description instance
 - How to generate a compact description instance again?

Open Issue #6: Profiles

- **Designing profiles**
 - **Current draft provides merely examples**
 - **Audio, RTP, UDP, IPv4**
 - **Complete set of basic profiles for RTP/AVP required**
 - **Payload type assignment**
 - **Security: MIKEY in SDPng**
 - **QoS**
- **Different types of QoS parameters**

Next Steps

- **Restructuring specification**
 - **Split document:**
 - **Framework document**
 - General model, scenarios
 - → Informational?
 - **SPDng spec.**
 - Syntax, processing algorithms
 - → Standards track
 - **Profiles**
 - Some basic profiles covering RTP/AVP
 - → Standards track
- **Editorial changes required**
 - **Make examples consistent, fix bugs**

Next Steps

- **Complete list of open issues**
 - **Discuss (and resolve) on the mailing list**
- **Work on profiles**
- **Fix remaining issues**

SDPng implementation

- **C++ library**
- **Works with expat (XML parser)**
 - Any other event-based parser should be fine
- **Covers sdpng-03**
 - Newer stuff currently being implemented